Modern Web Development

Chapter 10

Unit Testing JavaScript

Chapter Objectives

In this chapter we will:

- Learn about unit testing JavaScript code
- Demonstrate assertion frameworks and testrunners

Unit Testing JavaScript

Intro to Unit Testing

Introducing Karma

Introducing Jasmine

Testing Applications

- Testing provide safeguards
- Tests document systems
- Types of tests
  - **Unit**: Small tests covering individual functions
  - **Integration**: Integrated modules (combined functionality)
  - **Functional**: Focused on end result of larger actions
  - **Acceptance**: Typically client-side, end-to-end testing

Unit Tests

- Unit tests to see if a given module works
- All the dependencies are stubbed
  - we are providing fake dependencies for a module
- Test the API
  - write the test for the exposed methods, not for the internal workings of the given module

Unit tests have the following structure:

- Test setup
- Calling the tested method
- Asserting

What tech is available to help test?

- Jasmine
- Karma
- Chai
- Mocha
- QUnit
- many others...

Unit Testing JavaScript

Intro to Unit Testing

Introducing Karma

Introducing Jasmine

Karma Test Runner

- A JavaScript test runner that can be configured for your project needs
  - which browsers to test, where files are located
- Karma is testing framework agnostic
  - Can describe your tests with Jasmine, Mocha, Qunit
  - Can write adapters for others
- Karma allows you to test your code on real devices
  - test your code on real browsers and real devices such as phones

Adding Karma to a project

- Install karma and other dependencies

how could we add these to a project?

```
npm install --save-dev karma
karma-chrome-launcher
karma-webpack etc...
```

```
"devDependencies": {
  "karma": "^2.0.5",
  "karma-chrome-launcher": "^2.2.0",
  "karma-webpack": "^3.0.0",
  "karma-jasmine": "^1.1.2",
  "jasmine-core": "^3.1.0",
  "puppeteer": "^1.6.1",
  "webpack": "^4.16.3",
  "babel-core": "^6.26.3",
  "babel-loader": "^7.1.5",
  "babel-preset-env": "^1.7.0"
```

Configuring Karma

- Can start with just **karma** and use `npx karma init`
  - Or use ./node_modules/.bin/karma init
  - Answer questions to generate config file and modify package.json
    - karma.conf.js

```
C:\coursedev\test-example\setup>npx karma init
npx: installed 1 in 4.139s
Path must be a string. Received undefined
C:\coursedev\test-example\setup\node_modules\karma\bin\karma

Which testing framework do you want to use ?
Press tab to list possible options. Enter to move to the next question.
> jasmine
```

```
Config file generated at "C:\coursedev\test-example\setup\karma.conf.js".
```

Karma init:

- modifies package.json
- creates karma.conf.js

```
"devDependencies": {
  "karma": "^4.0.1",
  "karma-chrome-launcher": "^2.2.0",
  "karma-jasmine": "^2.0.1"
```

karma.conf.js

```
1  // Karma configuration
2  // Generated on Fri Mar 15 2019 06
3
4  module.exports = function(config)
5    config.set({
6
7      // base path that will be used
8      basePath: '',
9
10
11     // frameworks to use
12     // available frameworks: https
13     frameworks: ['jasmine'],
14
15
16     // list of files / patterns to
17     files: [
18       'src/*.js',
19       'test/**/*.test.js'
20     ],
```

Unit Testing JavaScript

Intro to Unit Testing

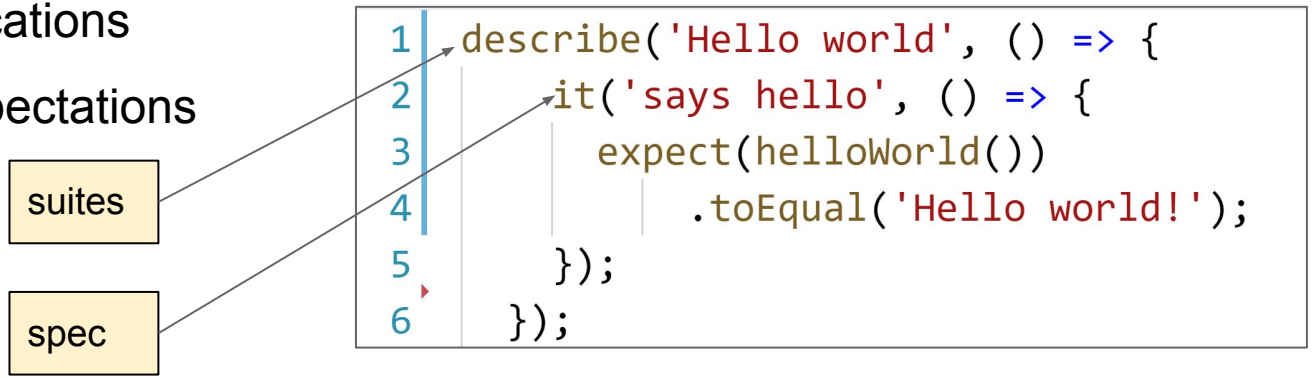
Introducing Karma

Introducing Jasmine

Introducing Jasmine

- A behavior driven testing framework for testing JavaScript code
  - (“readable”)
- Testing framework
- Suites possess a hierarchical structure, multiple specs
- Tests as specifications
- Matchers for expectations

npm i -D jasmine-core



Specs

example-expectations.test.js

```
2  describe('example-expectations', () => {
3      // See https://jasmine.github.io/api/edge/matchers.html
4      // for more matchers
5
6      it('is true', () => {
7        let someBoolean = true;
8        expect(someBoolean).toBeTruthy();
9      });
10
11     it('is 42', () => {
12       const value = 42;
13       expect(value).toEqual(42);
14     });
```

Typically a single spec will be written for each .js file in an app

Example Jasmine Matchers

- toBe('expected') //exact compare (===)
- toEqual('expected') //more general compare, can compare objects
- toMatch(/regex/) //matches against regex
- toBeNull(/regex/) //checks if a var is null
- toBeTruthy( ) //checks if var is truthy
- toBeFalsy( ) //checks if var is falsy
- toBeLessThan(number) //checks if value is less than number
- toBeGreaterThan(number) //checks if value is greater than

Can nest describe blocks for organization

all-matchers.test.js

```
1  describe("Included matchers:", function() {
2
3      describe("The 'toEqual' matcher", function() {
4
5        it("works for simple literals and variables", function() {
6          var a = 12;
7          expect(a).toEqual(12);
8        });
9
10       it("should work for objects", function() {
11         var foo = {
```

If > 3 it blocks, best practice is to wrap in nested describe()

Test Setup & Teardown: beforeEach and afterEach

shared-setup.test.js

```
describe("A suite with some shared setup", function() {
    var foo = 0;

// the beforeEach function is called once before each spec

    beforeEach(function() {
      foo += 1;
    });

 //the afterEach function is called once after each spec.

    afterEach(function() {
      foo = 0;
    });
```

Test Setup & Teardown: beforeAll and afterAll

```
describe("A suite with some shared setup", function() {
  // called only once before all the specs in describe are run
  let bar = 0;
  beforeAll(function() {
    bar = 1;
  });

  // the afterAll function is called after all specs finish

  afterAll(function() {
    bar = 0;
  });
});
```

DEMO: Chapter10 Testing

Running unit tests

- Start test runner by issuing following command if karma is globally installed

```
$ karma start karma.conf.js
```

- Save typing, keep in project - use scripts of package.json and: **npm test**

```
"scripts": {
  "test": "karma start karma.conf.js"
},
```

Resources

- http://karma-runner.github.io/0.13/index.html
- http://jasmine.github.io/2.4/introduction.html