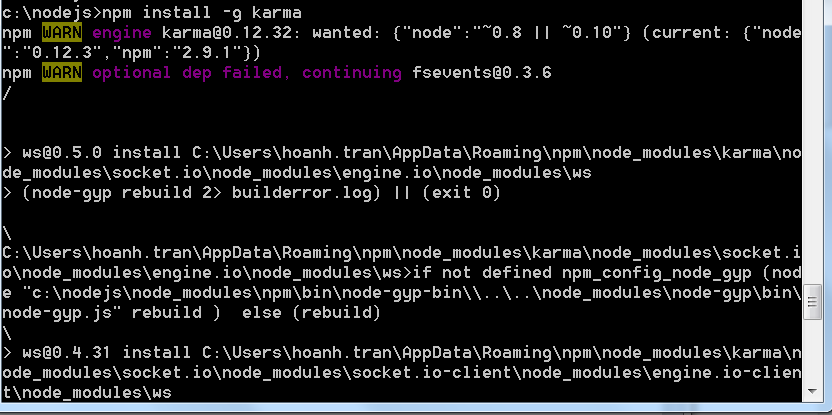
**Karma Unit Test Driver**

Karma is a Javascript unit test driver written in NodeJS. Karma uses Jasmine javascript unit test library and syntax. What make Karma stands out is it run very fast and constantly monitors your Javascript code. As soon you make changes to your JS code, it will run the unit test for you. You don’t have to remember to manually your unit tests. Karma will make you more productive.

Follow these steps to install NodeJS on Windows:

1. Download the latest NodeJS <https://nodejs.org/download/>. As of this writing it’s v0.12.3
2. Run the installer. Let say you install it at c:\node\node0.12.3
3. Edit the system PATH environment variable and add “C:\node\node0.12.3;” to the end of the PATH.
4. Start a new Command Prompt and type “npm install –g karma”. This will download and install karma. You may see some “warnings”, but it’s OK.



1. Once the installation completes, verify the karma is installed correctly.

The –g switch installs the Karma is download in C:\Users\your-user-name\AppData\Roaming\npm\node\_modules\karma

If you leave out the –g switch, Karma will be installed in your node JS installation dir (eg c:\node\node0.12.3\ node\_modules\karma).

For this tutorial, since I use –g, the rest of the steps assume Karma is installed in C:\Users\your-user-name\AppData\Roaming\npm\node\_modules\karma

1. You need to add the following system environment variables:

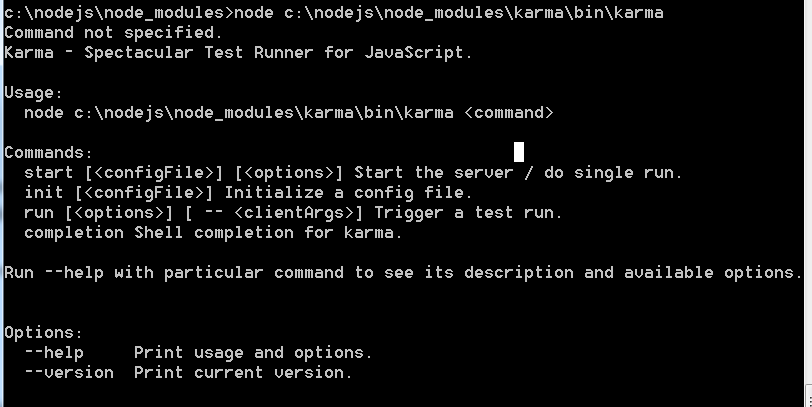
CHROME\_BIN = c:\Program Files (x86)\Google\Chrome\Application\Chrome.exe

FIREFOX\_BIN = c:\Program Files (x86)\Mozilla Firefox\firefox.exe

If you have other browsers you want Karma to run your unit tests against, you will need to add the system env var to them (eg OPERA\_BIN, SAFARI\_BIN). I have not try this yet so you will need to experience with these browsers.

1. Close and start a new Command Prompt window. Type “echo %FIREFOX\_BIN% “ to verify the env variable is set correctly.
2. To make sure node is able to run karma, type this

node c:\Users\hoanh\AppData\Roaming\npm\node\_modules\karma\bin\karma

You should see  indicating node recognized karma and not generating errors.

1. Now you’re ready to create unit test and execute Karma. Assume you have the following layout in your web project

WebContent

|\_\_\_ assets

|\_\_\_ angular.js

|\_\_\_ angular-mocks.js

|\_\_\_ js

|\_\_\_ \*.js

|\_\_\_ test

|\_\_\_ \*Spec.js unit test files

Let say your custom JS files are stored in WebContent/js directory, we will create the karma configuration file in the parent directory of “js”

* cd to WebContent
* Type: “node C:\Users\your-user-name\AppData\Roaming\npm\node\_modules\karma\bin\karma init”

Karma will prompts you with the following questions with some suggested answers.

Which testing framework do you want to use ?

Press tab to list possible options. Enter to move to the next question.

**> jasmine**

Do you want to use Require.js ?

This will add Require.js plugin.

Press tab to list possible options. Enter to move to the next question.

> no

Do you want to capture any browsers automatically ?

Press tab to list possible options. Enter empty string to move to the next quest

ion.

**> Chrome**

**> Firefox**

**>**

What is the location of your source and test files ?

You can use glob patterns, eg. "js/\*.js" or "test/\*\*/\*Spec.js".

Enter empty string to move to the next question.

**> assets/angular.js <<<<<<<<< angular must be the first entries**

**> assets/angular-mocks.js**

**> js/\*.js**

**> test/\*Spec.js**

**>**

Should any of the files included by the previous patterns be excluded ?

You can use glob patterns, eg. "\*\*/\*.swp".

Enter empty string to move to the next question.

>

Do you want Karma to watch all the files and run the tests on change ?

Press tab to list possible options.

**> yes**

Config file generated at "C:\src\plural\protractor-introduction\dynWeb\WebContent\karma.conf.js".

1. Now assume you have some JS files in WebContent\js you want to test and your JS unit test files in WebContent\test. Let say you have the following files:

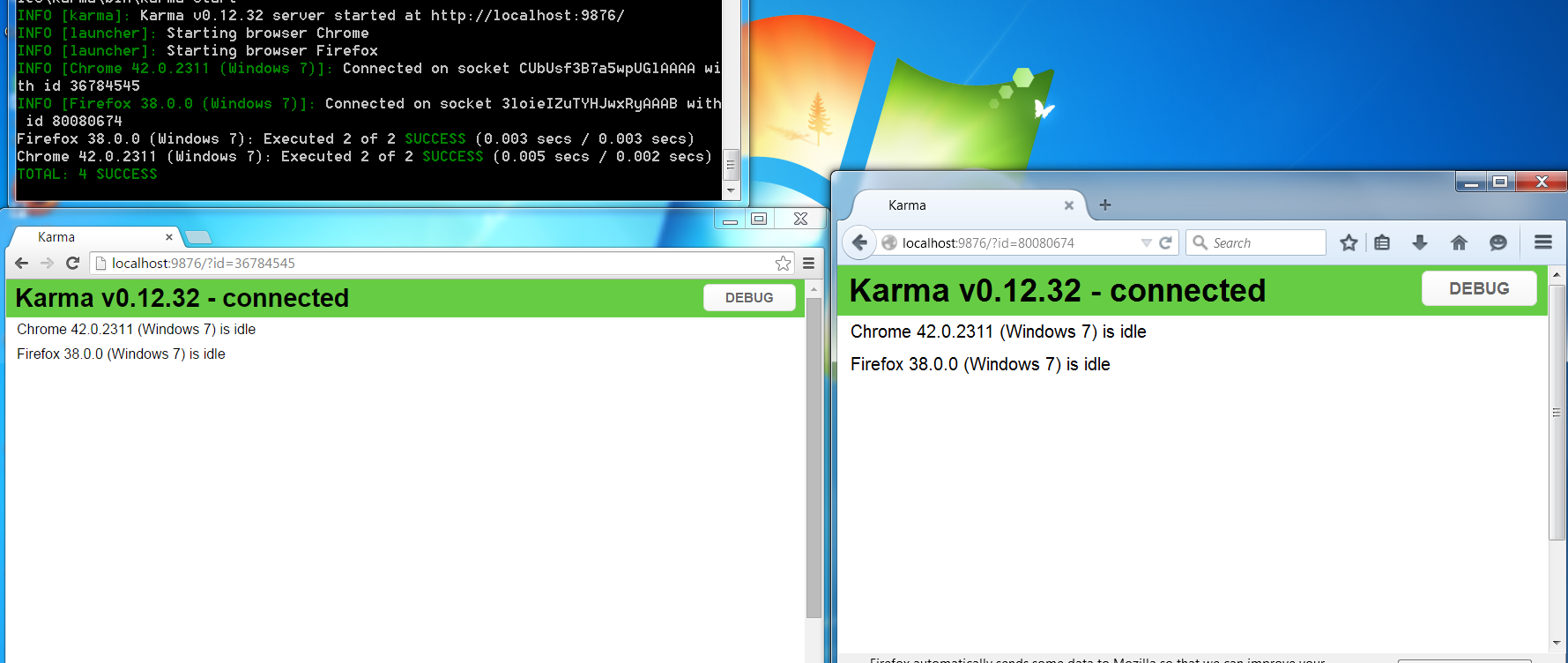
WebContent\js\Code.js

**var *myObj*** = {  
 method1: **function**() {  
 **return** 4;  
 }  
}

WebContent\test\CodeSpec.js <<<<< append “Spec” to the end to signify its unit test.

*describe*(**"My Sample Unit Test"**, **function**() {  
  
 *it*(**'should pass'**, **function**() {  
 *expect*(myObj.method1()).toBe(4);   
 });   
   
 *it*(**'should not do anything else'**, **function**() {  
 *expect*(2).toBe(2);  
 });  
});

1. Now still within WebContent directory, type: “node C:\Users\your-user-name\AppData\Roaming\npm\node\_modules\karma\bin\karma start”. This will launch couple of browsers. For this case since I specified two browsers in the previous steps, Karma starts and launches Chrome and Firefox.



1. It will immediately runs the 2 unit tests. You will see 4 SUCCESS since the 2 unit tests are run on Chrome and again on Firefox.

INFO [karma]: Karma v0.12.32 server started at http://localhost:9876/

INFO [launcher]: Starting browser Chrome

INFO [launcher]: Starting browser Firefox

INFO [Chrome 42.0.2311 (Windows 7)]: Connected on socket CUbUsf3B7a5wpUGlAAAA with id 36784545

INFO [Firefox 38.0.0 (Windows 7)]: Connected on socket 3loieIZuTYHJwxRyAAAB with id 80080674

Firefox 38.0.0 (Windows 7): Executed 2 of 2 SUCCESS (0.003 secs / 0.003 secs)

Chrome 42.0.2311 (Windows 7): Executed 2 of 2 SUCCESS (0.005 secs / 0.002 secs)

TOTAL: 4 SUCCESS

1. Now using a text editor and make some changes to either Code.js or CodeSpec.js and save it. From the Command console, you will notice Karma automatically re-runs the unit tests with 4 SUCCESS.

You can experiment with this by changing the unit test to cause the test to fail. You should see some failed unit tests.

