

Cucumber+WebDriverJS= CucumberJS



[My before article](#) was about WebDriverJS and its conveniences such as less dependencies and working on nodeJS. Although there are a lot of good sides of WebDriverJS, it is more close TDD. That's why it may be little hard to understand the tests and their steps for the people that haven't any coding knowlage like BA or costumers.

To solve this problem we can use Cucumber Test Framework which is more BDD. Cucumber lets software development teams describe how software should behave in plain text. The text is written in a business-readable domain-specific language and serves as documentation, automated tests.

If we use the Cucumber and WebDriverJS together, it is called CucumberJS. With CucumberJS tests are written in Gherking language in a file whose extension is feature. And then each step in test should be coded with JS in a folder which is called step definations.

```
test2.feature
Feature: Test SonyMobile Web Page

Scenario Outline: Test SonyMobile Web Title Description
    Given I visit "https://www.sonymobile.com/<locale>"
    And Wait "2000"

Examples:
| locale |
| sa     |
| mx     |
```

For Given step in Feature file, its task should be coded in related JS file like below. And if we use this Given step in any other Feature, we don't have to code again and again. It will call its JS code in step definition folder.

```

test.js
var assert = require('assert');

module.exports = function () {
  this.Given(/^I visit "([^"]*)"$/, function (arg1) {
    return this.driver.get(arg1);
  });

  this.Then(/^I see title SonyMobile$/, function() {
    this.driver.getTitle().then(function (title) {
      assert.equal(title, "The Official Xperia™ Website – Sony Mobile (UK)");
    });
  });

  this.Then(/^Wait "([^"]*)"$/, function (milisec) {
    this.driver.sleep(milisec);
  });

  this.Given(/^I visit "([^"]*)"$/, function (arg1) {
    return this.driver.get(arg1);
  });
};

```

So if we code enough steps via JS in step definition folder, we can create automated test cases by using only Cucumber Gherking language. At first, it can take more time comparing to develop with only JS. Because in this framework we should create JS and Feature files. But after all necessary step are coded, duration of creating automated test cases will reduce very much in future. And the best feature of Cucumber for me is using tables with scenario outlines. Because we are working on has multi-regional platform and it helps to run test on each region easily. When WebdriverJS good sides are made combine with Cucumber conveniences, it could be very charming and there is no reason to use this tool to develop automated tests.

How to Install

I have already created a folder structure and samples to able to use quickly. You can reach it from attachments of this page. Download it and need to go to this folder and apply below commands:

install Cucumber and Selenium webdriver

\$ npm install cucumber@1.3.3 --save-dev (It works properly only with this version now.)

\$ npm install selenium-webdriver

install browser drivers


\$ npm install chromedriver geckodriver

install an assertion tool

\$ npm install assert

Then run "*npm test*" command in same path, and a browser should be up and sample CucumberJS test should be run on it.

```
TRSEUISTMAC033:~ trtopcmu$ cd cucumberjs/  
TRSEUISTMAC033:cucumberjs trtopcmu$ npm test  
  
> @ test /Users/trtopcmu/cucumberjs  
> cucumber-js -f pretty  
  
Feature: Test SonyMobile Web Page  
  
  Scenario: Test SonyMobile Web Title Description  
    ✓ Given I visit "https://www.sonymobile.com/sa"  
    ✓ And Wait "2000"  
  
  Scenario: Test SonyMobile Web Title Description  
    ✓ Given I visit "https://www.sonymobile.com/mx"  
    ✓ And Wait "2000"  
  
2 scenarios (2 passed)  
4 steps (4 passed)  
0m16.193s  
TRSEUISTMAC033:cucumberjs trtopcmu$
```

 **Note**
After download [cucumberjs](#) file from attachment, better take a look in **package.json** and **world.js** and **hook.js** in support folder to understand to process of this framework.
And please don't change the folder structure to work successfully.

