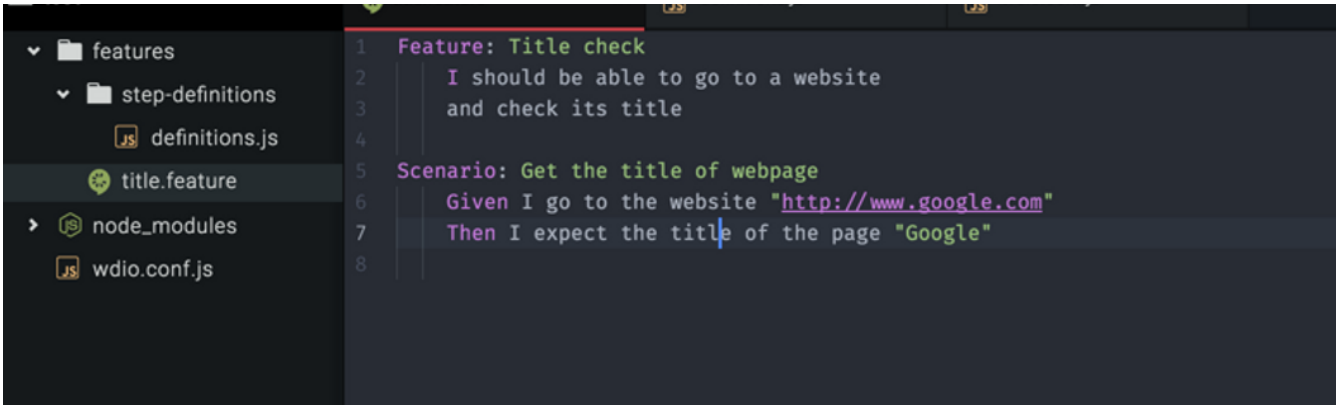
Jeremy Rajan FOLLOW
Full Stack Developer (OSS lover)

Acceptance Testing in JavaScript with Cucumber.js & WebdriverIO

Published Oct 11, 2016



When it comes to testing your JS code, it is of utmost importance that you include acceptance tests apart from unit tests.
This is where tools like Selenium or PhantomJS come into play. These tools help us run tests against our JS code in a real browser. If we put in a bit of effort, we can run those on multiple devices or platforms using services like Sauce Labs or BrowserStack.

Enter WebdriverIO!! WebdriverIO lets you write tests, with actions through which you can control the browser and assert the results. You can use services such as Mocha, Cucumber, or Jasmine.

Here we will discuss how to setup Cucumber.js with WebdriverIO. If you are new to Cucumber.js or have not heard about it.

information regarding it since it's a bit spread out (might want to use your Google-ing skills 😃).

While using Cucumber.js for testing, you basically write feature files to test multiple scenarios. For instance, the following is a feature file:

```
# title.feature
Feature: Title check
  I should be able to go to a website
  and check its title

Scenario: Get the title of webpage
  Given I go to the website "http://www.google.com"
  Then I expect the title of the page "Google"
```

If you've noticed, the syntax is pretty much self-explanatory and readable. You must be thinking, really... that's it ? Well, yes and no.
The code above will work as an entry point to your test, but WebdriverIO (the test runner), does not know what those lines mean. This is where `step definitions` come in.
Step definitions, as the name suggests is JS code (which the runner understands) for each step in the test. For the example above, the following are the step definitions:

```
// stepdefinitions.js
const expect = require('chai').expect;

module.exports = function () {
  this.Given(/^I go to the website "([^"]*)"$/, (url) => {
    browser.url(url);
  });

  this.Then(/^I expect the title of the page "([^"]*)"$/, (title) => {
    expect(browser.getTitle()).to.be.eql(title);
  });
}
```

You will notice here that for each `Given` and `Then` statements, we have corresponding JS definitions. These definitions can be used in other feature or test files as well.

> *To make things clear, all your test files (or feature files .feature) and step definition files (*.js) are separate.*

Now we have our test files and step definitions in place, it's time to link them using WebdriverIO, which will bring all these files together and run it for us.

The folder structure looks something like this:

Before we get started, let's install WebdriverIO and Cucumber deps first.

```
[sudo] npm i -g WebdriverIO
[sudo] npm i -g wdio-cucumber-framework
```

In order to start the browser for testing, we need to install and start Selenium.

```
[sudo] npm i selenium-standalone@latest -g
```

Once all of them is installed, navigate to the directory where your feature files and step definitions are present—assuming that, we have them in `~/project_name/test` directory. Let's continue.

Invoke WebdriverIO

In order to do so, we need to create a configuration file. To create one, type...

```
wdio
```

...inside the `test` folder. This will prompt you to create a configuration file and give you four options.
For the time being, we will concentrate on the first option as that would help us get WebdriverIO up and running on our local machine.

When prompted, select the following options:

```
Where do you want to execute your tests? On my local machine

Which framework do you want to use? cucumber

Shall I install the framework adapter for you? No # we already have installed t

Where are your feature files located? (./features/**/*.feature) # this is a glo

Where are your step definitions located? (./features/step-definitions)  # Your

Which reporter do you want to use?  dot - https://github.com/webdriverio/wdio-d

Shall I install the reporter library for you? Yes

Do you want to add a service to your test setup? Selenium Standalone

Level of logging verbosity silent

In which directory should screenshots gets saved if a command fails? ./errorSho

What is the base url? http://localhost # In case you did not provide a full url
```

After answering all the questions, it will look like this:

Once that is done, you might want to install the assertion library of your own choice. I am using Chai here. Install and save Chai, using

```
npm i --save-dev chai
```

Once you are happy with the configuration, run the tests (in the current

```
selenium-standalone install # install selenium
selenium-standalone start # start selenium
wdio wdio.conf.js
```

If everything went according to plan, you will see:

Instead of starting Selenium separately, there is a plugin which is part of WebdriverIO. This will start Selenium and stop Selenium according to tests. In order to do that, install the plugin first:

```
npm i wdio-selenium-standalone-service --save-dev
```

And, in the `wdio.conf.js` file, add the following as parent key

```
exports.config = {
  ...
  services: ['selenium-standalone'],
  ...
}
```

That's it, you can try running `wdio wdio.conf.js` again without starting the selenium-standalone. Please make sure that the selenium-standalone is not running in the background.

Hope this helps someone.

In the next tutorial, we will discuss on how to automate the whole process using Gulp.

Related tutorials you might be interested in

- Unit Testing React Components: Jest or Enzyme?
- How to (Finally) Learn Test-Driven Development

Selenium webdriver | Testing | Cucumberjs | JavaScript

Enjoy this post? Give **Jeremy Rajan** a like if it's helpful.

4 4 SHARE

Jeremy Rajan

Full Stack Developer (OSS lover)

##My self-summary## I have been working on developing web applications from 2009, during the course of my career I have worked with different kind of websites and technologies. A problem solver by nature, I am very particular abo...

FOLLOW

4 Replies

Leave a reply

Pavan Royal a month ago

Thanks Jeremy. Please can you share me the link for the next tutorial - 'How to automate the whole process using Gulp'?

Reply

4 4

Thanks Jeremy!
In cucumber 2.+ version many things have changed (completely new syntax, etc). Do you plan to release updated version?

Reply

Jeremy Rajan a year ago

Yes! I have not personally gone thru the updated version. Will give it a shot and try to setup a revised version :).

Reply

xgqfrms a year ago

js test ?

Reply

Show more replies

Łukasz Makuch

React and visual automata-based programming

4 4

This post consists of three parts. In the first one I'm telling about **my journey to visual automata-based programming**. The second one shows **the process of building a GUI using visual automata-based programming**. The code is written in JavaScript and React is used to render the view. The last part is **a summary of my experience with state machines** where I explain how did they change the way I think about state management.

READ MORE