

# Headless testing web applications using Jest and Puppeteer in JavaScript

## Overview

In this tutorial, we will create some headless tests in JavaScript using Puppeteer and Jest framework; we'll use "jest-junit" to generate a compatible JUnit XML report.

## Requirements

- puppeteer
- jest-environment-node
- jest
- jest-junit

## Code

The following code and the setup instructions were inspired on [jest-puppeteer-example](#) tutorial.

The first thing we need to do is configure jest to use the "jest-junit" reporter, in order to produce a JUnit XML report.

### jest.config.js

```
module.exports = {
  globalSetup: './setup.js',
  globalTeardown: './teardown.js',
  testEnvironment: './puppeteer_environment.js',
  reporters: [ "default",
    [ "jest-junit", { "classNameTemplate": "e2e", "titleTemplate": "{classname} {title}" } ]
  ]
}
```

We will create two simple tests based on Google search.

## `__tests__/google.js`

```
const timeout = 5000

describe(
  '/ (Search Home Page)',
  () => {
    let page
    beforeAll(async () => {
      page = await global.__BROWSER__.newPage()
      await page.goto('https://www.google.com', { waitUntil: 'networkidle0' })
    }, timeout)

    afterAll(async () => {
      await page.close()
    })

    it('should return a proper title', async () => {
      // just to make sure
      const title = await page.title();
      expect(title).toBe('Google');
    })
    it('should return link to Marketplace upon searching Xray', async () => {
      await page.waitForSelector('input#lst-ib',{ visible: true});
      await page.type('input#lst-ib', 'Xray test management')

      await page.screenshot({path: 'screenshot1.png'});
      await page.keyboard.press('Enter');

      await page.waitForSelector('div#rcnt',{ visible: true});
      await page.screenshot({path: 'screenshot2.png'});
      const results = await page.evaluate(() => document.querySelector('div#rcnt').innerText);
      expect(results).toContain('Xray - Test Management for Jira | Atlassian Marketplace');
    })
  },
  timeout
)
```

After running the tests and generating the JUnit XML report (e.g. junit.xml), it can be imported to Xray (either by the REST API or through the **Import Execution Results** action within the Test Execution).

```
npm test __test__/google.js
```

			Key	Summary	Test Type	#Req	#Def	Assignee	Status	
	<input type="checkbox"/>	1	CALC-2120	/ (Search Home Page) should return a proper title	Generic	0	0	Administrator	PASS	...
	<input type="checkbox"/>	2	CALC-2121	/ (Search Home Page) should return link to Marketplace upon searching Xray	Generic	0	0	Administrator	PASS	...

Showing 1 to 2 of 2 entries

First Previous **1** Next Last

Each test/"it block" is mapped to a Generic Test in Jira, and the **Generic Test Definition** field contains an unique identifier based on the Test Suite and the test name. In fact, the [jest-junit package](#) allows full customization of some attributes of the JUnit XML, which in turn will allow you to define exactly how to fine tune the value for the "Generic Test Definition" field accordingly with the rules described in [Taking advantage of JUnit XML reports](#).

The Execution Details of the Generic Test contains information about the Test Suite, which in this case corresponds to the name of the "describe" block.

Calculator / Test Execution: CALC-2122 / Test: CALC-2128

Export Test as Text
Return to Test Execution
Previous

/ (Search Home Page) should return link to Marketplace upon searching Xray

Execution Details

Test Description

None

Test Details

Test Type: Generic

Definition: e2e./ (Search Home Page) should return link to Marketplace upon searching Xray

Results

Context	Error Message	Duration	Status
TestSuite / (Search Home Page)	-	1 sec	PASS

## References

- <https://facebook.github.io/jest/docs/en/puppeteer.html>
- <https://github.com/xfumihiro/jest-puppeteer-example>
- <https://www.npmjs.com/package/jest-junit>
- <https://github.com/GoogleChrome/puppeteer/blob/master/examples>