

Testing Node.js apps using Cucumber.js in JavaScript

- [Overview](#)
- [Requirements](#)
- [Description](#)
 - [Using Jira and Xray as master](#)
- [FAQ and Recommendations](#)
- [References](#)

Overview

In this tutorial, we will create tests for Node.js, in JavaScript, with Cucumber.js.

The test (specification) is initially created in Jira as a Cucumber Test and afterwards, it is exported using the UI or the REST API.

Requirements

- nodejs
- npm packages
 - cucumber

Description

For the purpose of this tutorial, we'll use a simple Javascript class implementing a very basic calculator.

lib/calculator.js

```
class Calculator {  
  
  constructor(x, y) {  
    this.x = x;  
    this.y = y;  
  }  
  
  add() {  
    this.result = this.x + this.y;  
  }  
  
  getResult() {  
    return this.result;  
  }  
}  
  
module.exports = Calculator;
```

We aim to test the sum operation.

However, before moving into the actual implementation, you need to decide is which workflow we'll use: do we want to use Xray/Jira as the master for writing the declarative specification, or do we want to manage those in Git?

[This tutorial only showcases using Xray/Jira as the master for editing the Cucumber Scenarios/Scenario Outlines.](#)



Learn more

Please see [Testing in BDD with Gherkin based frameworks \(e.g. Cucumber\)](#) for an overview of the possible workflows.

Using Jira and Xray as master

This section assumes using Xray as master, i.e. the place that you'll be using to edit the specifications (e.g. the scenarios that are part of .feature files).

The overall flow would be something like this:

1. create Scenario/Scenario Outline as a Test in Jira; usually, it would be linked to an existing "requirement"/Story (i.e. created from the respective issue screen)
2. implement the code related to Gherkin statements/steps and store it in Git, for example
3. generate .feature files based on the specification made in Jira
4. checkout the code from Git
5. run the tests in the CI
6. import the results back to Jira

Usually, you would start by having a Story, or similar (e.g. "requirement"), to describe the behavior of a certain feature and use that to drive your testing.

If you have it, then you can just use the "Create Test" on that issue to create the Scenario/Scenario Outline and have it automatically linked back to the Story/"requirement".

Otherwise, you can create the Test using the standard (issue) Create action from Jira's top menu.

The screenshot shows the Jira Software interface. The top navigation bar includes 'Jira Software', 'Your work', 'Projects', 'Filters', 'Dashboards', 'People', 'Apps', and a 'Create' button. The left sidebar lists project navigation options: 'Calculator' (Classic software project), 'CALC board' (Board), 'Backlog', 'Active sprints', 'Reports', 'Issues' (selected), 'Components', and 'Code'. The main content area displays the issue 'As a user, I can calculate the sum of 2 numbers' (ID: CALC-629). Below the title are action buttons: 'Attach', 'Create subtask', 'Link issue', 'Test Coverage', and a menu icon. The 'Description' section contains the text 'As a user, I can calculate the sum of 2 numbers'. The 'Test Coverage' section shows 'No Tests are associated with this issue.' and two buttons: 'Create new Sub Test Execution' and 'Create new Test' (highlighted with a red box). A blue 'UNCOVERED' badge is visible at the bottom right.

In this case, we'll create a Cucumber Test, of Cucumber Type "Scenario".

We need to create the Test issue first and fill out the details on the next screen, from within the Test issue.

Add two numbers



Description

Tests As a user, I can calculate the sum of 2 numbers

Linked issues



tests

 CALC-629 As a user, I can calculate the sum of 2 numbers  **TO DO**

Test Details



Cucumber

▼

Test Repository

Scenario

1

Given the numbers 2 and 3

2

When they are added together

3

Then should the result be 5

✓

✕

After the Test is created it will impact the coverage of related "requirement", if any.

The coverage and the test results can be tracked in the "requirement" side (e.g. user story). In this case, it changed from being UNCOVERED to NOTRUN (i.e. covered and with at least one test not run).

As a user, I can calculate the sum of 2 numbers



Description

As a user, I can calculate the sum of 2 numbers

Linked issues

is tested by

CALC-630 Add two numbers ↑ TO DO

Test Coverage

Calculate the Test Coverage for the following scopes.

Create new Sub Test Execution

Create new Test

Latest Version Test Plan

Test Environment

All Environments



NOTRUN

☒ Final statuses have precedence over non-final.

Status	Key	Summary	Test Status
↑.. TO DO	CALC-630	Add two numbers	↑.. TO DO
Prev	1	Next	

The related statement's code is managed outside of Jira and stored in Git, for example.

features/step_definitions/addition_steps.js

```
const assert = require('assert')
const {Before, Given, When, Then} = require('cucumber');
const Calculator = require('../lib/calculator');

let calculator;

Given('the numbers {int} and {int}', function (x, y) {
  calculator = new Calculator(x, y);
});

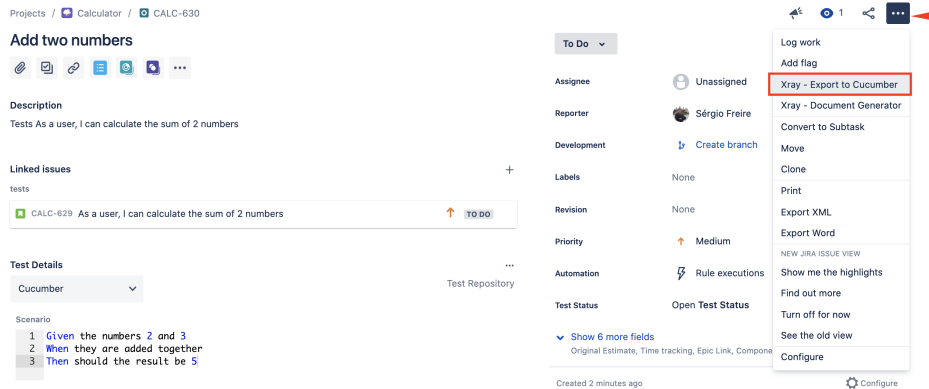
When('they are added together', function () {
  calculator.add();
});

Then('should the result be {int}', function (expected) {
  assert.equal(calculator.getResult(), expected)
});
```

You can then export the specification of the test to a Cucumber .feature file via the REST API, or the **Export to Cucumber** UI action from within the Test /Test Execution issue or even based on an existing saved filter. A plugin for your CI tool of choice can be used to ease this task.

So, you can either:

- use the UI



- use the REST API (more info [here](#))

○ #!/bin/bash

```
token=$(curl -H "Content-Type: application/json" -X POST --data @"cloud_auth.json" https://xray.cloud.getxray.app/api/v2/authenticate| tr -d ' ')
curl -H "Content-Type: application/json" -X GET -H "Authorization: Bearer $token" "https://xray.cloud.getxray.app/api/v2/export/cucumber?keys=CALC-630" -o features.zip

rm -rf features/*.feature
unzip -o features.zip -d features
```

- use one of the available CI/CD plugins (e.g. see an example of [Integration with Jenkins](#))

After being exported, the created .feature(s) will contain references to the Test issue key and the covered "requirement" issue key, if that's the case. The naming of these files is detailed in [Generate Cucumber Features](#).

new feature after export (features/1_CALC-7895.feature)

```
@REQ_CALC-7895
Feature: As a user, I can calculate the sum of 2 numbers

  # Addition is great as a verification exercise to get the Cucumber-js infrastructure up and running
  @TEST_CALC-4763 @features/addition.feature
  Scenario: Add two number
    Given the numbers 2 and 3
    When they are added together
    Then should the result be 5
```

To run the tests and produce a Cucumber JSON report, we can use the `cucumber-js` binary.

```
./node_modules/cucumber/bin/cucumber-js -f json:report.json
```

After running the tests, results can be imported to Xray via the REST API, or the **Import Execution Results** action within the Test Execution, or by using one of the available CI/CD plugins (e.g. see an example of [Integration with Jenkins](#)).

```
BASE_URL=https://xray.cloud.getxray.app
token=$(curl -H "Content-Type: application/json" -X POST --data @"cloud_auth.json" "$BASE_URL/api/v2/authenticate" | tr -d ' ')
curl -H "Content-Type: application/json" -X POST -H "Authorization: Bearer $token" --data @"report.json" "$BASE_URL/api/v2/import/execution/cucumber"
```



Which Cucumber endpoint/"format" to use?

To import results, you can use two different endpoints/"formats" (endpoints described in [Import Execution Results - REST](#)):

1. the "standard cucumber" endpoint
2. the "multipart cucumber" endpoint

The standard cucumber endpoint (i.e. `/import/execution/cucumber`) is simpler but more restrictive: you cannot specify values for custom fields on the Test Execution that will be created. This endpoint creates new Test Execution issues unless the Feature contains a tag having an issue key of an existing Test Execution.

The multipart cucumber endpoint will allow you to customize fields (e.g. Fix Version, Test Plan), if you wish to do so, on the Test Execution that will be created. Note that this endpoint always creates new Test Executions (as of Xray v4.2).

In sum, if you want to customize the Fix Version, Test Plan and/or Test Environment of the Test Execution issue that will be created, you'll have to use the "multipart cucumber" endpoint.

A new Test Execution will be created (unless you originally exported the Scenarios/Scenario Outlines from a Test Execution).

Projects / Calculator / CALC-631

Execution results [1604333427544]

[Attach](#) [Create subtask](#) [Link issue](#) [Tests](#) [...](#)

Description

Add a description...

Tests

Create Test

+ Add

Overall Execution Status

TOTAL TESTS: 1

1 PASSED

Filters 100 Columns

Rank	Key	Summary	Test Type	Status	Actions
1	CALC-630	Add two numbers	Cucumber	PASSED	Expand More

Prev 1 Next

Total 1 issues

The execution screen details of the Test Run will provide overall status information and Gherkin statement-level results; you'll need to click on the arrow to expand and see the detailed Gherkin statements.

Jira Software

Your work

Projects

Filters

Dashboards

People

Apps

Create

Q Search

🔍 ? ⚙️ 👤

▶ Execution Details

Test Description

Tests As a user, I can calculate the sum of 2 numbers

Test Issue Links (1)

tests

CALC-629

As a user, I can calculate the sum of 2 numbers

↑

TO DO

Custom Fields

There are no Test Run Custom Fields defined.

Test Details

Test Type: Cucumber

Scenario Type: Scenario

Scenario:

1

Given the numbers 2 and 3

2

When they are added together

3

Then should the result be 5

Results

Context

Steps

Given the numbers 2 and 3

When they are added together

Then should the result be 5

Duration

2ms

1 millisecond

0 millisecond

1 millisecond

Status

PASSED

PASSED

PASSED

PASSED

Results are reflected on the covered item (e.g. Story). On its issue screen, coverage now shows that the item is OK based on the latest testing results, that can also be tracked within the Test Coverage panel below.

Projects / Calculator / CALC-629

As a user, I can calculate the sum of 2 numbers

Attach

Create subtask

Link issue

Test Coverage

...

Description

As a user, I can calculate the sum of 2 numbers

Linked issues

is tested by

CALC-630

Add two numbers

↑

TO DO

Test Coverage

Calculate the Test Coverage for the following scopes.

Latest

Version

Test Plan

Create new Sub Test Execution

Create new Test

Test Environment

All Environments

▼

→

OK

☒ Final statuses have precedence over non-final.

Status	Key	Summary	Test Status
↑ .. TO DO	CALC-630	Add two numbers	→ PASSED

Prev 1 Next

FAQ and Recommendations

Please see [this page](#).

References

- <https://github.com/cucumber/cucumber-js>
- [Testing in BDD with Gherkin based frameworks \(e.g. Cucumber\)](#)
- [Automated Tests \(Import/Export\)](#)
- [Exporting Cucumber Tests - REST](#)