

Running JavaScript tests with Karma

This tutorial describes how to use the [Karma test runner](#) in the WebStorm IDE.

- [Karma integration in WebStorm](#)
- [Installing Karma](#)
- [Karma configuration](#)
- [Running tests](#)
 - [Run](#)
 - [Run with coverage](#)

Karma integration in WebStorm

Starting with WebStorm 7.0, you can use the Karma test runner to run JavaScript tests for your project. Karma integration in WebStorm:

- Uses local server to run the tests in the selected browsers installed on your computer;
- Allows you to run tests written with the use of Jasmine, QUnit, or Mocha, or to write a simple adapter to use any other framework you like;
- Presents test results and code coverage report by the Istanbul code coverage engine right in the IDE.

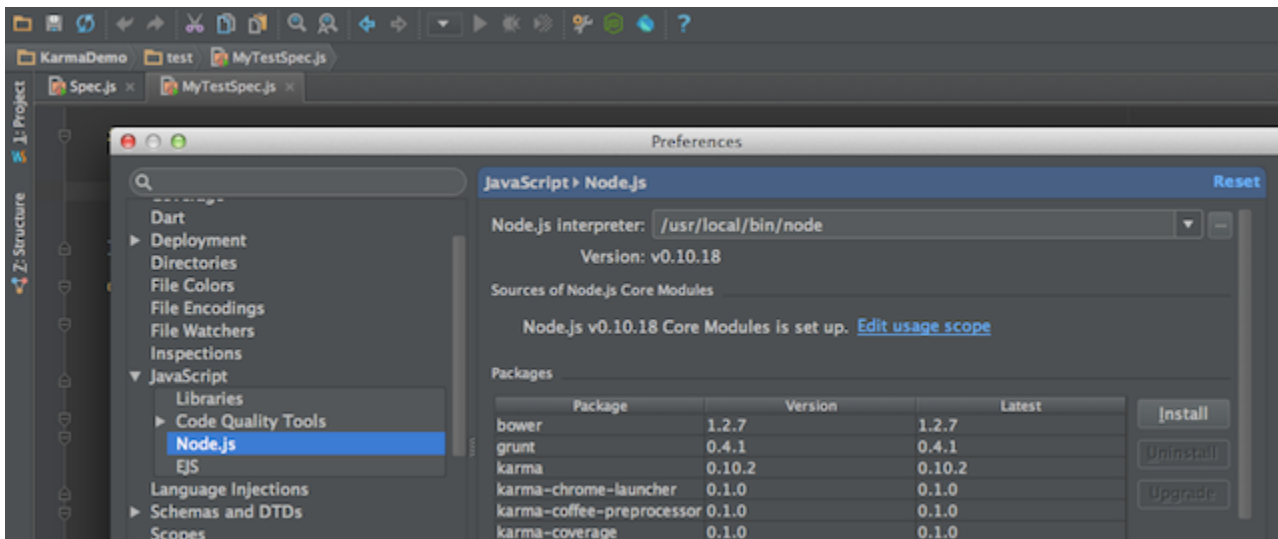
Installing Karma

First, make sure that [Node.js](#) is installed on your computer.

Install Karma using [npm](#):

The [recommended approach](#) is to install Karma (and all the plugins your project needs) locally in the project's directory.

Use WebStorm's built-in npm ([Settings | JavaScript | Node.js and NPM](#)) search and install Karma for your project.



Or install it in the current directory using Terminal:

```
npm install karma
```

Install karma-cli globally:

Use WebStorm's built-in npm and search for karma-cli, add -g option before hitting Install for global installation. Or use built-in Terminal to install karma-cli with:

```
npm install -g karma-cli
```

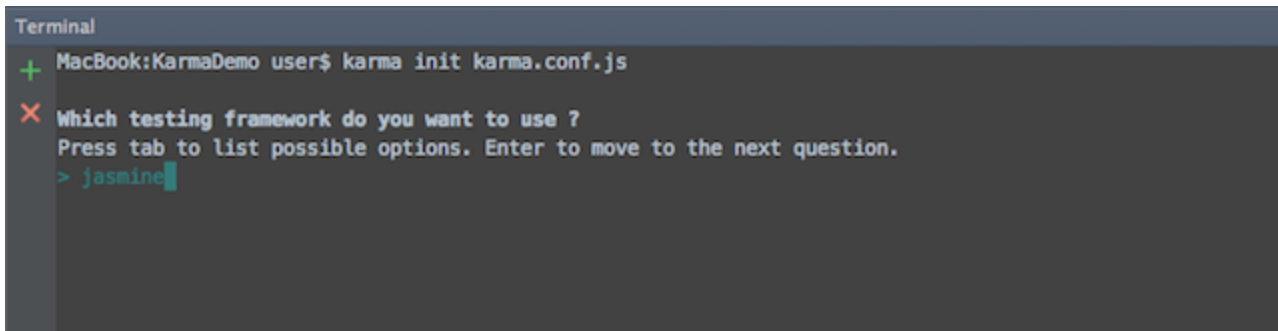
Karma configuration

To run new tests with Karma, you need to have a Karma configuration file ([learn more about it](#)).

To generate new configuration file, for example, karma.conf.js in the project directory run in built-in Terminal:

```
karma init karma.conf.js
```

Follow the suggested steps in the configuration dialog.



```
Terminal
+ MacBook:KarmaDemo user$ karma init karma.conf.js
X Which testing framework do you want to use ?
  Press tab to list possible options. Enter to move to the next question.
  > jasmine
```

Note: Most of the framework adapters, reporters, preprocessors and launchers need to be loaded as [plugins](#).

Create a WebStorm Karma Run Configuration:

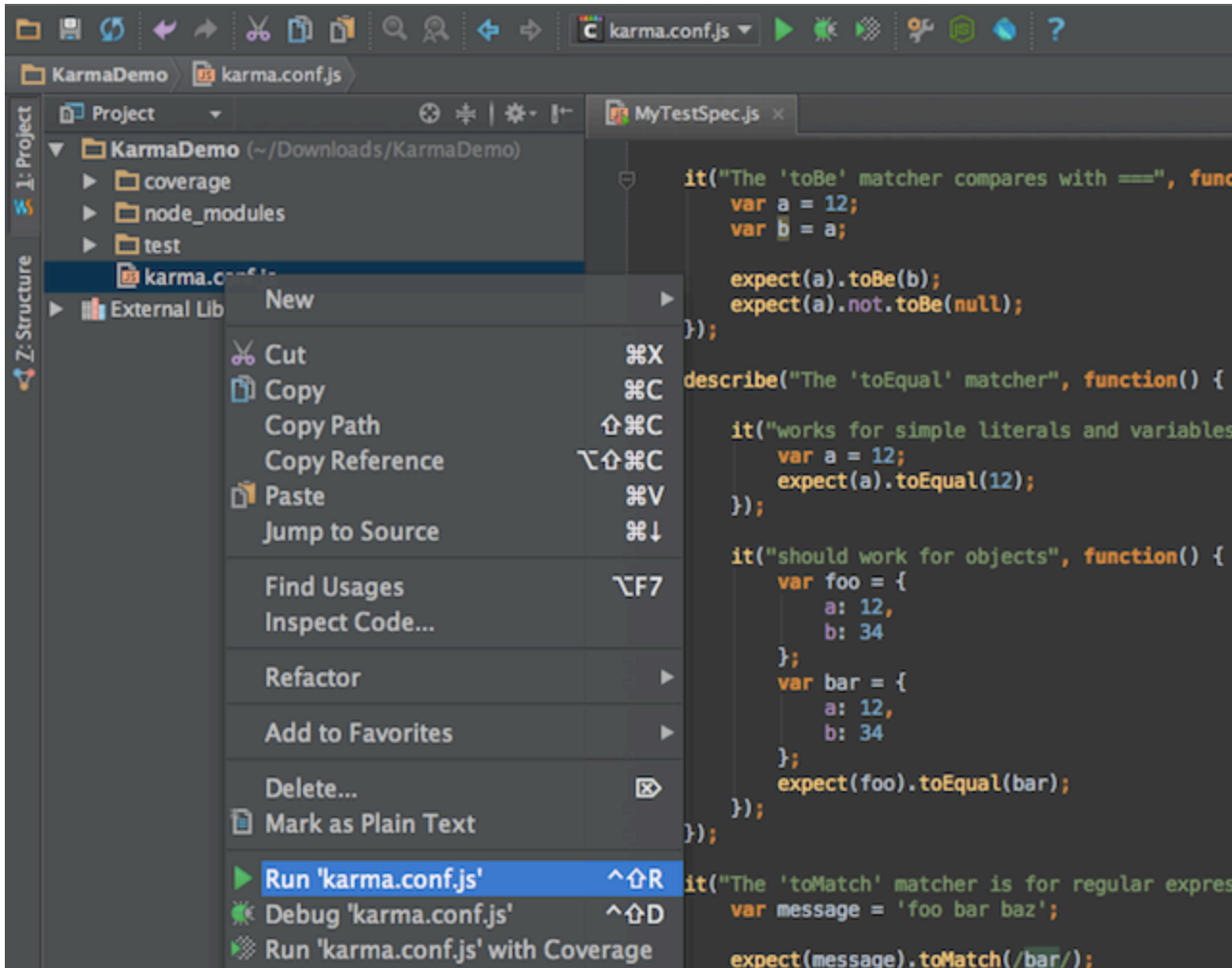
- Click Create 'karma.conf.js' in the context menu for karma.conf.js file
- Check the paths in the Create Run/Debug Configuration window
- Click OK.

Running tests


Run

A new Karma Configuration will be added to the list of Run/Debug Configurations.

To start running your tests, click Run in the toolbar or in the context menu of Karma configuration file.



Note: WebStorm disables autoWatch in Karma configuration. You can rerun your tests with the shortcut Alt+Shift+R on Win/Linux or ctrl-cmd-R on Mac.

Click auto-test  in the WebStorm Test Run window to enable automatic test rerun: The test will be run 10 seconds after your code was changed, if there are no syntax errors.

Run with coverage

WebStorm Karma integration allows you to run tests with code coverage provided by the istanbul code coverage engine.

To use Run with coverage feature, a karma-coverage module should be installed via npm.

Make the required changes in the Karma configuration file, for example:

```
reporters: ["coverage"],
preprocessors: {
  "**/*.js": "coverage"
}
```

You can navigate through test results in the Run window, check statistics for the tests execution, and make HTML reports from it. Code coverage report will be also available.

The screenshot displays an IDE interface with three main panels: a code editor, a coverage summary, and a test results panel.

Code Editor: Shows the following JavaScript code for `karma-jqlite.conf.js`:

```
'use strict';

describe('angular.scenario.Future', function() {
  var future;

  it('should set the sane defaults', function() {
    var behavior = function() {};
    var future = new angular.scenario.Future('test name',
    expect(future.name).toEqual('test name');
    expect(future.behavior).toEqual(behavior);
    expect(future.line).toEqual('foo');
    expect(future.value).toBeUndefined();
    expect(future.fulfilled).toBeFalsy();
    expect(future.parser).toEqual(angular.identity);
  });

  it('should be fulfilled after execution and done callback', function() {
    var future = new angular.scenario.Future('test name',
    done();
  });
  future.execute(angular.noop);
  expect(future.fulfilled).toBeTruthy();
});

  it('should take callback with (error, result) and forward', function() {
    var future = new angular.scenario.Future('test name',
    done(10, 20);
  });
  future.execute(function(error, result) {
    expect(error).toEqual(10);
    expect(result).toEqual(20);
  });
});
```

Coverage karma-jqlite.conf.js

Coverage Summary: 100% files, 90% lines covered

Element	Statistics, %
.idea	
bower_components	100% files, 43% lines covered
coverage	
css	
docs	100% files, 7% lines covered
example	100% files, 100% lines covered
i18n	
images	
lib	
logs	
src	100% files, 94% lines covered
test	100% files, 98% lines covered
.bowerrc	
.gitignore	
.travis.yml	
angularFiles.js	
bower.json	
changelog.js	
CHANGELOG.md	
changelog.spec.js	
changelog.tmp.md	
check-size.sh	
compare-master-to-stab...	
CONTRIBUTING.md	
gdocs.js	
gen_docs.sh	

Run karma-jqlite.conf.js

Test Run Karma Server

Done: 2269 of 2269 (26.132 s)

Test	Time el...	Results
should be fulfilled after execution a	1 ms	Passed
should convert to json with toJson	2 ms	Passed
should convert with custom parser	0 s	Passed
should parse json with fromjson	0 s	Passed
should pass error if parser fails	1 ms	Passed
should set the sane defaults	1 ms	Passed