Monitoring Code Coverage for JavaScript

Code Coverage shows how much of your code is covered with tests and marks covered and uncovered lines visually right in the editor.

In this section:

Prerequisites

1. The *JSTestDriver* or *Karma* test runner is installed and configured.
2. Additional third-party testing frameworks, for example, Jasmine, QUnit, or Mocha, are downloaded and integrated with IntelliJ IDEA.
3. A runner-specific configuration file is generated or written manually.

For more details, see Preparing to Use Karma Test Runner or Preparing to Use JSTestDriver Test Runner depending on your choice.

Enabling running tests with coverage in Karma

Coverage for tests run in *Karma* is calculated by the istanbul coverage tool. The tool is shipped within the `karma-coverage` package. Therefore running tests with coverage in *Karma* requires that you download the `karma-coverage` package and manually update the `karma.conf.js` configuration file.

The `karma-coverage` package should be installed in the same folder with *Karma* (`karma (library home)` package), no matter whether *Karma* is installed globally or in your project. Keep this restriction in mind when installing `karma-coverage` manually in the command line mode. If you are installing `karma-coverage` through the *Node Package Manager* interface, the proper installation folder will be chosen automatically.

1. Install the `karma-coverage` package through the *Node Package Manager (npm)*.
    1. Choose **File | Settings** and in the **Settings** dialog box that opens, click **Node.js and NPM**.
    2. On the Node.js page that opens, click **Install** in the **Packages** area.
    3. In the **Available Packages** dialog box that opens, select the `karma-coverage` package, click **Install Package**, and close the dialog box when ready. IntelliJ IDEA returns to the **Node.js** page, where the `karma-coverage` package is added to the **Packages** list. Click **OK**.

    For details, see Installing and Removing External Software Using Node Package Manager.
2. Open the `karma.conf.js` configuration file and add the following information to it manually:
    1. Locate the `reporters` definition and add `coverage` to the list of values in the following format:

    ```
    reporters: ['progress', 'coverage'],
    ```

    2. Add a `preprocessors` definition and specify the coverage scope according to the following format:

    ```
    preprocessors: {
        '**/*.js': ['coverage']
    }
    ```

Measuring code coverage for tests executed on JSTestDriver

1. Prepare tests manually or have tests generated by IntelliJ IDEA.
2. Create a JSTestDriver configuration file `*.jstd` or `*.conf`. For details, see JSTestDriver Configuration file.
3. Create a run configuration of the type JSTestDriver.

   To have specific files excluded from coverage analysis, specify the paths to them in the **Coverage** tab of the Run/Debug Configuration: JSTestDriver dialog box.
4. Start the JSTestDriver server and capture a browser to run the tests in.
5. On the main toolbar, select the *JSTestDriver* run configuration in the **Run/Debug Configurations** drop-down list and click the **Run with Coverage** button  .
6. Monitor the code coverage in the Coverage tool window.

Measuring code coverage for tests executed on Karma

1. Prepare tests manually or have tests generated by IntelliJ IDEA.
2. Create a Karma configuration file `karma.conf`. For details, see Running Unit Tests on Karma.
3. Create a run configuration of the type Karma.
4. On the main toolbar, select the *Karma* run configuration in the **Run/Debug Configurations** drop-down list and click the **Run with Coverage** button  .
5. Monitor the code coverage in the Coverage tool window. Note that when the tests are executed on `Karma` the **Coverage** tool window does not contain the **Generate Coverage Report** toolbar button  because a coverage report is actually generated on the disk every time *Karma* tests are run. The format of a coverage report can be configured in the configuration file, for example:

   ```
   // karma.conf.js
   module.exports = function(config) {

   config.set({ ...

   // optionally, configure the reporter

   coverageReporter: { type : 'html', dir : 'coverage/' }

   ...

   });};
   ```

   The following `type` values are acceptable:

   - `html` produces a bunch of HTML files with annotated source code
   - `lcovonly` produces an `lcov.info` file
   - `lcov` produces HTML + `.lcov` files. This format is applied by default.
   - `cobertura` produces a `cobertura-coverage.xml` file for easy Hudson integration
   - `text-summary` produces a compact text summary of coverage, typically to the console
   - `text` produces a detailed text table with coverage for all files

See Also

Procedures:

- Code Coverage
- Viewing Code Coverage Results
- Running JavaScript Unit Tests in Browser
- Unit Testing JavaScript

- Testing

Reference:

- Run/Debug Configuration: JSTestDriver
- Coverage Tool Window

Web Resources:

- Developer Community