Transpiling CoffeeScript to JavaScript

IntelliJ IDEA supports integration with the coffee-script transpilation tool. The tool translates CoffeeScript code into JavaScript and creates source maps that set correspondence between lines in your CoffeeScript code and in the generated JavaScript code, otherwise your breakpoints will not be recognised and processed correctly.

In IntelliJ IDEA, transpiler configurations are called *File Watchers*. For each supported transpiler, IntelliJ IDEA provides a predefined *File Watcher* template. Predefined *File Watcher* templates are available at the IntelliJ IDEA level. To run a transpiler against your project files, you need to create a project-specific *File Watcher* based on the relevant template, at least, specify the path to the transpiler to use on your machine.

On this page:

- Prerequisites
- Installing the CoffeeScript transpiler
- Installing the CoffeeScript transpiler globally
- Installing the CoffeeScript transpiler in a project
- Creating a File Watcher
- Examples of customizing the behaviour of a transpiler
- Transpiling the CoffeeScript code
- Previewing the transpilation results without running a transpiler

Prerequisites

1. Download and install Node.js. The framework is required for two reasons:
   - The CoffeeScript transpiler is started through *Node.js*.
   - *NPM*, which is a part of the framework, is also the easiest way to download the CoffeeScript transpiler.

   For details on using *Node.js* in IntelliJ IDEA, see the section Node.js

   > Alternatively, you can define Node.js as an external tool, as described in the section Configuring third-party tools. This approach is helpful, when you need facilities that are missing in the plugin, for example, the possibility to pass certain parameters as wildcards.

2. If you are going to use the command line mode, make sure the following paths are added to the `PATH` variable:
   1. The path to the parent folder of the *Node.js* executable file.
   2. The path to the `npm` folder.

   This enables you to launch the CoffeeScript transpiler and *npm* from any folder.

3. Install and enable the *NodeJS* repository plugin.

   The plugin is not bundled with IntelliJ IDEA, but it is available from the JetBrains plugin repository. Once enabled, the plugin is available at the IDE level, that is, you can use it in all your IntelliJ IDEA projects. See Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins for details.

4. Install and enable the *File Watchers* repository plugin.

   The plugin is not bundled with IntelliJ IDEA, but it is available from the IntelliJ IDEA plugin repository. Once enabled, the plugin is available at the IDE level, that is, you can use it in all your IntelliJ IDEA projects. See Installing, Updating and Uninstalling Repository Plugins and Enabling and Disabling Plugins for details.

Installing the CoffeeScript transpiler

The easiest way to install the CoffeeScript transpiler is to use the *Node Package Manager (npm)*, which is a part of Node.js.

Depending on the desired location of the CoffeeScript transpiler executable file, choose one of the following methods:

- Install the transpiler *globally* at the IntelliJ IDEA level so it can be used in any IntelliJ IDEA project.
- Install the transpiler in a specific project and thus restrict its use to this project.
- Install the transpiler in a project as a development dependency.

In either installation mode, make sure that the parent folder of the CoffeeScript transpiler is added to the `PATH` variable. This enables you to launch the transpiler from any folder.

IntelliJ IDEA provides user interface both for *global* and *project* installation as well as supports installation through the command line.

Installing the CoffeeScript transpiler globally

Global installation makes the transpiler available at the IntelliJ IDEA level so it can be used in any IntelliJ IDEA project. Moreover, during installation the parent folder of the transpiler is automatically added to the `PATH` variable, which enables you to launch the transpiler from any folder. To install the transpiler *globally*, do one of the following:

1. Run the installation from the command line in the *global* mode:
    1. Switch to the directory where *NPM* is stored or define a `PATH` variable for it so it is available from any folder, see Installing NodeJs.
    2. Type the following command at the command line prompt:

        ```
        npm install -g coffee-script
        ```

        The `-g` key makes the transpiler run in the *global* mode. Because the installation is performed through *NPM*, the CoffeeScript transpiler is installed in the `npm` folder. Make sure this parent folder is added to the `PATH` variable. This enables you to launch the transpiler from any folder.

        For more details on the *NPM* operation modes, see npm documentation. For more information about installing the CoffeeScript transpiler, see https://npmjs.org/package/coffee-script.
2. Run *NPM* from IntelliJ IDEA using the **Node.js and NPM** page of the **Settings** dialog box.
    1. Open the project settings and then click **Node.js and NPM**.
    2. On the **Node.js and NPM** page that opens, the **Packages** area shows all the Node.js-dependent packages that are currently installed on your computer, both at the *global* and at the *project* level. Click +.
    3. In the **Available Packages** dialog box that opens, select the `coffee-script` package.
    4. Select the **Options** check box and type `-g` in the text box next to it.
    5. Optionally specify the product version and click **Install Package** to start installation.

Installing the CoffeeScript transpiler in a project

Installing the transpiler in a specific project restricts its use to this project. To run *project* installation, do one of the following:

- In the command line mode, switch to the project root folder and type the following command at the command line prompt:

```
npm install coffee-script
```

- Run *NPM* from IntelliJ IDEA using the **Node.js and NPM** page of the **Settings** dialog box.
  1. Open the project settings and click **Node.js and NPM**.
  2. On the **Node.js and NPM** page that opens, the **Packages** area shows all the Node.js-dependent packages that are currently installed on your computer, both at the *global* and at the *project* level. Click +.
  3. In the **Available Packages** dialog box that opens, select the `coffee-script` package.
  4. Optionally specify the product version and click **Install Package** to start installation.

Project level installation is helpful and reliable in template-based projects of the type *Node Boilerplate* or *Node.js Express*, which already have the `node_modules` folder. The latter is important because *NPM* installs the CoffeeScript transpiler in a `node_modules` folder. If your project already contains such folder, the CoffeeScript transpiler is installed there.

Projects of other types or *empty* projects may not have a `node_modules` folder. In this case *npm* goes upwards in the folder tree and installs the CoffeeScript transpiler in the first detected `node_modules` folder. Keep in mind that this detected `node_modules` folder may be *outside* your current project root.

Finally, if no `node_modules` folder is detected in the folder tree either, the folder is created right under the current project root and the CoffeeScript transpiler is installed there.

In either case, make sure that the parent folder of the CoffeeScript transpiler is added to the `PATH` variable. This enables you to launch the transpiler from any folder.

Creating a File Watcher

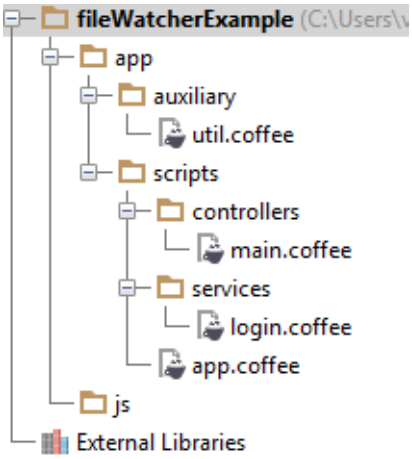
IntelliJ IDEA provides a common procedure and user interface for creating *File Watchers* of all types. The only difference is in the predefined templates you choose in each case.

1. To start creating a *File Watcher*, open the **Project Settings** by choosing **File | Settings** on the main menu, and then click **File Watchers**. The File Watchers page that opens, shows the list of *File Watchers* that are already configured in the project.
2. Click the **Add** button + or press `Alt+Insert` and choose the **CoffeeScript** predefined template from the pop-up list. Your code will be translated to JavaScript and supplied with generated source maps.
3. In the **Program** text box, specify the path to the `coffee.cmd` file. Type the path manually or click the **Browse** button ... and choose the file location in the dialog box that opens.
4. Proceed as described on page Using File Watchers.

Examples of customizing the behaviour of a transpiler

Any *transpiler* is an external, third-party tool. Therefore the only way to influence a *transpiler* is pass arguments to it just as if you were working in the command line mode. Below are two examples of customizing the default output location for the *CoffeeScript transpiler*.

Suppose, you have a project with the following folder structure:



By default, the generated files will be stored in the folder where the original file is. You can change this default location and have the generated files stored in the the `js` folder. Moreover, you can have them stored in a flat list or arranged in the folder structure that repeats the original structure under the `app` node.

- To have all the generated files stored in the output `js` folder without retaining the original folder structure under the `app` folder:

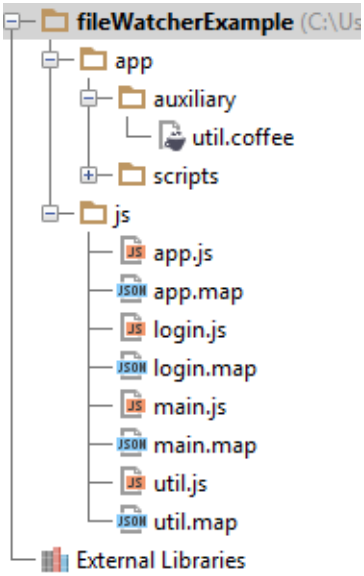
  1. In the **Arguments** text box, type:

     ```
     --output $ProjectFileDir$\js\ --compile --map $FileName$
     ```

  2. In the **Output paths to refresh** text box, type:

     ```
     $ProjectFileDir$\js\$FileNameWithoutExtension$.js:$ProjectFileDir$\js\$FileNameWithoutExt
     ```

  As a result, the project tree looks as follows:



- To have the original folder structure under the `app` node retained in the output `js` folder:

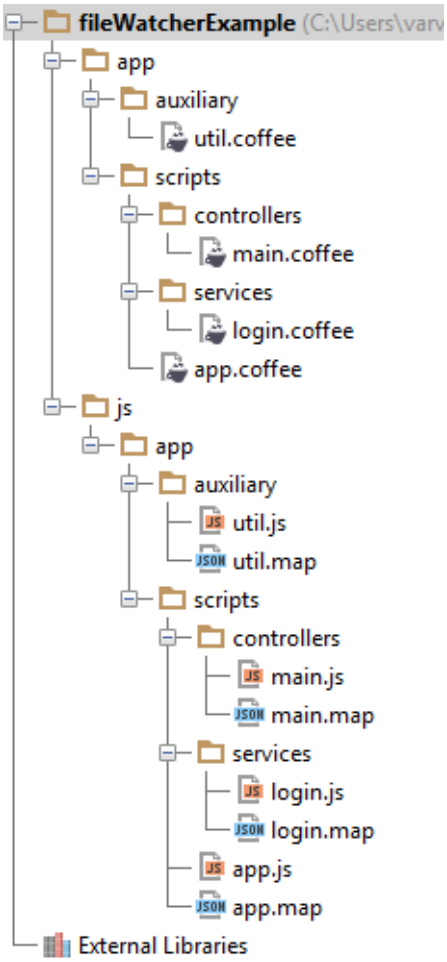
  1. In the **Arguments** text box, type:

     ```
     --output $ProjectFileDir$\js\$FileDirRelativeToProjectRoot$\ --compile --map $FileName$
     ```

  2. In the **Output paths to refresh** text box, type:

     ```
     $ProjectFileDir$\js\$FileDirRelativeToProjectRoot$\$FileNameWithoutExtension$.js:$Project
     ```

  As a result, the project tree looks as follows:



Transpiling the CoffeeScript code

When you open a CoffeeScript file, IntelliJ IDEA checks whether an applicable file watcher is available in the current project. If such file watcher is configured but disabled, IntelliJ IDEA displays a pop-up window that informs you about the configured file watcher and suggests to enable it.

If an applicable file watcher is configured and enabled in the current project, IntelliJ IDEA starts it automatically upon the event specified in the New Watcher dialog.

- If the **Immediate file synchronization** check box is selected, the *File Watcher* is invoked as soon as any changes are made to the source code.
- If the **Immediate file synchronization** check box is cleared, the *File Watcher* is started upon save (**File | Save All**, `Ctrl+S`) or when you move focus from IntelliJ IDEA (upon frame deactivation).

The *transpiler* stores the generated output in a separate file. The file has the name of the source *CoffeeScript* file and the extension `js` or `js.map` depending on the *transpiler* type. The location of the generated files is defined in the **Output paths to refresh** text box of the New Watcher dialog. Based on this setting, IntelliJ IDEA detects the *transpiler* output. However, in the **Project Tree**, they are shown under the source `.coffee` file which is now displayed as a node.

Previewing the transpilation results without running a transpiler

IntelliJ IDEA can perform static analyses of your CoffeeScript code without actually running a transpiler and display the predicted transpilation output in the dedicated read-only viewer.

1. Open the desired CoffeeScript file in the editor, and right-click the editor background.
2. On the context menu, choose **Preview Compiled CoffeeScript File**. The preview is opened in the dedicated read-only viewer: the left-hand pane shows the original CoffeeScript source code and the right-hand pane shows the JavaScript code that will be generated by the transpiler when it runs.

See Also

Procedures:

- Using File Watchers

Language and Framework-Specific Guidelines:

- JavaScript-Specific Guidelines
- Node.js

Reference:

- File Watchers
- New Watcher Dialog

External Links:

- http://coffeescript.org/
- http://zsitro.com/coffeescript-compiler-setup-for-webstorm/

Web Resources:

- Developer Community