Installing and Removing External Software Using Node Package Manager

A number of tools are started through *Node.js*, for example, the *CoffeeScript*, *TypeScript*, and *LESS* transpilers, *YUI*, *UglifyJS*, and *Closure* compressors, *Karma* test runner, *Grunt* task runner, etc. The *Node Package Manager (npm)* is the easiest way to install these tools, the more so that you have to install *Node.js* anyway.

On this page:

Installing Node.js and Node Package Manager (npm)

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

   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 tool 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.

Installing an external tool

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

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

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

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

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

Installing an external tool globally

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

- 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  <tool name>
     ```

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

     For more details on the *NPM* operation modes, see npm documentation. For more information about installing the tool, see https://npmjs.org/package/.
- 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 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 an external tool in a project

Installing a tool 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  <tool name>
```

- 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 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 tool in a `node_modules` folder. If your project already contains such folder, the tool 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 tool 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 tool is installed there.

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

Installing an external tool as a development dependency

If a tool is a documentation or a test framework, which are of no need for those who are going to re-use your application, it is helpful to have it excluded from download for the future. This is done by marking the tool as a development dependency, which actually means adding the tool in the `devDependencies` section of the `package.json` file.

With IntelliJ IDEA, you can have a tool marked as a *development dependency* right during 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 --dev  <tool name>
```

- Run *NPM* from IntelliJ IDEA using the **Node.js and NPM** page of the **Settings** dialog box.
  1. Open the project settings and click **NPM and Node.js**.
  2. On the **NPM and Node.js** 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 package.
  4. Select the **Options** check box and type `--dev` in the text box next to it.
  5. Optionally specify the product version and click **Install Package** to start installation.

After installation, a tool is added to the `devDependencies` section of the `package.json` file.

See Also

Procedures:

- Installing and Removing Bower Packages
- Transpiling CoffeeScript to JavaScript
- Transpiling TypeScript to JavaScript
- Using File Watchers

Reference:

- Node.js and NPM

Web Resources:

- Developer Community