Configuring JavaScript Libraries

In IntelliJ IDEA, you can download and install a number of the most popular JavaScript libraries. These include Dojo, ExtJS, jQuery, jQuery UI, Prototype, and others. Once installed, these libraries, by default, are visible in all files within the project.

You can also download JavaScript libraries and frameworks yourself, and use them to set up your own, custom JavaScript libraries. For such libraries, you have to specify their usage scope. Only then these libraries become available in the corresponding scope.

The libraries associated with your project, among other things, provide the basis for code completion, highlighting, and navigation, and also Documentation Lookup.

On this page:

Visibility and Scope

Each IntelliJ IDEA library is characterized by its *visibility* status and *scope*. The *visibility* status of a library determines whether it can be used in one project (*Project*) or can be re-used at the IDE level (*Global*).

- Once configured, a *Global library* can be associated with any of IntelliJ IDEA projects. The library itself can be located wherever you need, its settings are stored with other IntelliJ IDEA settings in the dedicated directories under the IntelliJ IDEA home directory, depending on your operating system.

  The *advantage* of configuring a framework as a global library is that you can store such library in one location and re-use it in unlimited number of your projects without copying the library under the project root every time.

  The *disadvantage* of this approach is that to enable team work on the project all the team members have the library stored on their machines in the same location relative to the project root.

- A *Project library* is visible only within one single project. Therefore a project library can be associated only with this project or its part, in other words, the scope of a project library is restricted to the current project or its part. This means that project libraries cannot be re-used, so if you later try to use a framework configured as a project library with another project, you will have to configure the library anew.

  The *advantage* of configuring a JavaScript framework as a project library is that you can share the library definition among other team members through the project settings so each developer does not have to configure the library separately,

  To use a self-developed JavaScript library, configure it as a IntelliJ IDEA library.

- *Predefined* libraries bring JavaScript definitions (also known as "stubs") for the standard DOM, HTML5 and EcmaScript API, as well as for Node.js global objects. These libraries make the basis for coding assistance in accordance with the API provided by the corresponding JavaScript engine. By enabling a certain predefined library you can ensure that your code fits the target environment.

  Predefined libraries are by default enabled in the *scope* of the entire project. A predefined library can be disabled or associated with another scope but it cannot be deleted.

The *scope* of a library defines the set of files and folders in which the library is considered as *library*, that is, it is write-protected, excluded from check for errors and refactoring, affects the completion list and highlighting.

Viewing the libraries associated with a file

1. Open the file in the editor.
2. Click the **Hector** icon on the **Status** bar. The pop-up window that opens shows the list of libraries associates with the current file. To change the list, click the **Libraries in scope** links and edit the scope settings in the Manage Scope dialog box that opens.

To download and install one or more of the most popular JavaScript libraries

IntelliJ IDEA provides a dedicated user interface for downloading and installing the most popular *official* JavaScript libraries. Using this interface, you can download and install Dojo, ExtJS, jQuery, jQuery UI, Prototype, and other libraries.

Besides the above listed *official* libraries, you can download stubs for TypeScript definition files.

1. Open the Settings dialog.
2. In the left-hand pane, select **JavaScript Libraries**.
3. In the right-hand part, click **Download**.
4. In the **Download Library** dialog that opens, choose the group of libraries in the drop-down list. The available options are **Official libraries** and **TypeScript community stubs**. Depending on your choice, IntelliJ IDEA displays a list of available libraries. Select the one to be downloaded and installed, and click **Download and Install**.

   Note that only one library can be selected and downloaded at a time.
5. If necessary, download and install more libraries one by one.
6. If necessary, change the usage scope for the downloaded libraries. (By default, the downloaded libraries are available from all the files within the project.)
7. Click **OK** in the **Settings** dialog.

To create a custom JavaScript library

1. Do one of the following:
   - Open the Settings dialog. In the left-hand pane, select **JavaScript Libraries**. In the right-hand part, click **Configure**.
   - Open the Project Structure dialog and click **Global Libraries** in the left-hand pane.
2. Click + on the toolbar, and select **JavaScript**.
3. In the **Select Library Files** dialog, select the necessary files and folders. These may be individual JavaScript files and the directories containing such files.
4. If appropriate, the **Detected Roots** dialog is displayed showing the library resources found in the folders you have selected. Use the check boxes to deselect unnecessary items.
5. If necessary, edit the library name.
6. Specify the library usage scope.

Specifying the usage scope for JavaScript libraries

The *scope* of a library defines the set of files and folders in which the library is considered as *library*, that is, it is write-protected, excluded from check for errors and refactoring, affects the completion list and highlighting.

By default, all predefined libraries and libraries downloaded from within IntelliJ IDEA provide completion, resolution, highlighting and are treated as libraries in any file within the project. In other words, their usage scope is the whole project.

Libraries that you create yourself are not considered libraries in any of the files unless you specify their usage scope explicitly.

1. Open the Settings dialog.
2. In the left-hand pane, expand the **JavaScript Libraries** node and select **Usage Scope**.
3. In the right-hand part (on the Usage Scope page):
   - To make the library available in all the files within the project, click the **Library** cell in the topmost (**Project**) row and select the library in the list that appears.
   - To specify a narrower usage scope, expand the nodes in the **File/Directory** column to see the folder or file you want to limit the library usage scope to. Click the **Library** cell to the right of the corresponding folder or file and select the library in the list.
4. Click **OK** in the **Settings** dialog.

Adding files to a JavaScript library

1. Do one of the following:
   - Open the Settings dialog. In the left-hand pane, select **JavaScript Libraries**. In the right-hand part, click **Configure**.
   - Open the Project Structure dialog and click **Global Libraries** in the left-hand pane.
2. In the area under + − , select the library of interest.
3. In the right-hand part, click + and select one of the following options:
   - **Attach Files or Directories.** Use this option to let IntelliJ IDEA decide which category the selected JavaScript file or files belong to.

     IntelliJ IDEA will analyze the selected files and folders, and automatically assign the JavaScript files to the appropriate categories. Minified files will be assigned to the Release category; ordinary (uncompressed) files will be assigned to the Debug category.
   - **Attach Debug Versions.** Use this option to add a single uncompressed JavaScript file or a directory containing such files.
   - **Attach Release Versions.** Use this option to add a single compressed (minified) JavaScript file or a directory containing such files.
4. Select the required file or directory in the dialog box that opens. If you used **Attach Files or Directories** at the previous step, you can select a number of files and directories simultaneously.
5. If appropriate, the **Detected Roots** dialog is displayed showing the library resources found in the folders you have selected. Use the check boxes to deselect unnecessary items.

Specifying links to external documentation

If the library files have associated external documentation available online, you can specify the URL of such external documentation to be able to view this documentation in a Web browser at a later time. (See Documentation Look-up in External JavaScript Libraries.)

1. Do one of the following:
   - Open the Settings dialog. In the left-hand pane, select **JavaScript Libraries**. In the right-hand part, click **Configure**.
   - Open the Project Structure dialog and click **Global Libraries** in the left-hand pane.
2. In the area under + − , select the library of interest.
3. In the right-hand part, click +, select **Specify Documentation URL** and specify the URL in the dialog that opens.

Downloading documentation for jQuery

If jQuery is included in one of your libraries, you can download the jQuery documentation and associate it with this library.

1. Do one of the following:
   - Open the Settings dialog. In the left-hand pane, select **JavaScript Libraries**. In the right-hand part, click **Configure**.
   - Open the Project Structure dialog and click **Global Libraries** in the left-hand pane.
2. In the area under + − , select the library of interest.
3. In the right-hand part, click + and select **Download Documentation**.

Removing items from a library

1. Do one of the following:
   - Open the Settings dialog. In the left-hand pane, select **JavaScript Libraries**. In the right-hand part, click **Configure**.
   - Open the Project Structure dialog and click **Global Libraries** in the left-hand pane.
2. In the area under + − , select the library of interest.
3. Select the library item or items to be removed, and click − in the right-hand part of the dialog.

Deleting a library

1. Do one of the following:
   - Open the Settings dialog. In the left-hand pane, select **JavaScript Libraries**. In the right-hand part, click **Configure**.
   - Open the Project Structure dialog and click **Global Libraries** in the left-hand pane.
2. In the area under + − , select the library to be deleted.
3. Click − on the toolbar.

See Also

Procedures:

- Viewing External Documentation
- Viewing Inline Documentation
- Enabling JavaScript Unit Testing Support

Reference:

- JavaScript Libraries
- Project Library and Global Library Pages

Web Resources:

- Developer Community