Debugging JavaScript

On this page:

Basic terms

When an HTML file with injected JavaScript code is opened in the browser, the script execution starts and suspends on reaching the first breakpoint, whereupon you can analyze the suspended program, step through it, resume execution when ready, etc.

> This HTML file does not necessarily have to be the one that implements the starting page of the application.

Depending on the location of the application files, IntelliJ IDEA distinguishes between *local* and *remote* debugging modes.

1. *Local debugging*. In this mode, the application files remain where they are in the IntelliJ IDEA project on you machine. IntelliJ IDEA runs them on the built-in Web server with the root URL `http://localhost:<built-in server port>/<project root>`. The *built-in server port* is specified on the Debugger. JavaScript page of the **Settings** dialog box. A debugging session in this mode can be initiated in two ways:
   - Open the HTML file with the JavaScript injection to be tested and launch the temporary run/debug configuration that IntelliJ IDEA has generated automatically. This approach it helpful when you do not need to debug the entire application but just one script.
   - Create and launch a permanent debug configuration.

   For details, see Temporary and Permanent Run/Debug Configurations.
2. *Remote debugging*. In this mode, the application files are on an external web server and you have their copies in a IntelliJ IDEA project on your computer. No matter, whether the web server itself is running on a physically remote host or on your machine, application files deployed on it are treated as *remote*. For details, see Working with Web Servers: Copying Files.

   You can deploy the application files to a web server and launch the application yourself or debug an already running application.

   When a remote HTML file with a JavaScript injection is opened, the debugger tells IntelliJ IDEA the name of the currently processed file and the number of the line to be processed. IntelliJ IDEA opens the local copy of this file and indicates the line with the provided number. This behaviour is enabled by specifying correspondence between files and folders on the server and their local copies. This correspondence is called *mapping*, it is set in the debug configuration.

   A *Remote* debugging session can be initiated only through a *permanent* debug configuration that you have to define yourself manually.

Starting a debugging session by opening an HTML file from IntelliJ IDEA (local debugging)

1. Set the breakpoints in the JavaScript code, as required.
2. Open the **HTML** file with the JavaScript injection to debug or select the HTML file in the Project view.
3. On the context menu of the editor or the selection, choose **Debug <file name>**. The automatically generated debug configuration starts a debugging session. This configuration is temporary but you can save it later.

   The file opens in the default browser and the Debug tool window appears.

4. If you are using Chrome or Dartium, you need a Chrome extension. If the extension is not installed and activated in your browser, the **Debug** tool window shows a message with a link to the web store where the extension is available. Click the link and install the extension. For details, see Using JetBrains Chrome Extension.
5. In the **Debug** tool window, proceed as usual: step through the program, stop and resume program execution, examine it when suspended, view actual HTML DOM, edit the code, etc.

Starting a debugging session through a user-defined configuration (local debugging)

1. Set the breakpoints in the JavaScript code, as required.
2. Create a permanent debug configuration.
   1. On the main menu, choose **Run | Edit Configurations**.
   2. Click the **Add New Configuration** toolbar button +, and choose **JavaScript Debug** on the context menu.
   3. In the Run/Debug Configuration: JavaScript Debug dialog box that opens, specify the following:
      - The URL address of the HTML file that implements the page to start debugging from. Use the following format: `http://localhost:<built-in server port>/<project root>/<path to the HTML file relative to the project root>`
      - Choose the browser to debug the application in (*Firefox* or *Chrome*).
   4. Click **OK** to save the configuration settings.
3. Choose the newly created configuration in the **Select run/debug configuration** drop-down list on the tool bar and click the **Debug** toolbar button.

   The HTML file specified in the run configuration opens in the chosen browser and the Debug tool window appears.

4. If you are using Chrome or Dartium, you need a Chrome extension. If the extension is not installed and activated in your browser, the **Debug** tool window shows a message with a link to the web store where the extension is available. Click the link and install the extension. For details, see Using JetBrains Chrome Extension.
5. In the **Debug** tool window, proceed as usual: step through the program, stop and resume program execution, examine it when suspended, view actual HTML DOM, edit the code, etc.

Remote debugging

1. Set the breakpoints in the JavaScript code, as required.
2. Create a permanent debug configuration.
   1. On the main menu, choose **Run | Edit Configurations**.
   2. Click the **Add New Configuration** toolbar button +, and choose **JavaScript Debug** on the context menu.
   3. In the Run/Debug Configuration: JavaScript Debug dialog box that opens, specify the following:
      - The URL address of the HTML file that implements the page to start debugging from. This URL address should correspond with the *server access configuration*.
      - Choose the browser to debug the application in (*Firefox* or *Chrome*).
      - Optionally define mappings between local files and URL addresses of their copies on the server. These mappings are required only if the local folder structure under the project root differs from the folder structure on the server. If the structures are identical, IntelliJ IDEA itself "retrieves" the paths to local files by parsing the URL addresses of their copies on the server.
   4. Click **OK** to save the configuration settings.

   Click **OK** to save the configuration settings.
3. Choose the newly created configuration in the **Select run/debug configuration** drop-down list on the tool bar and click the **Debug** toolbar button.

   The HTML file specified in the run configuration opens in the chosen browser and the Debug tool window appears.
4. If you are using Chrome or Dartium, you need a Chrome extension. If the extension is not installed and activated in your browser, the **Debug** tool window shows a message with a link to the web store where the extension is available. Click the link and install the extension. For details, see Using JetBrains Chrome Extension.
5. In the **Debug** tool window, proceed as usual: step through the program, stop and resume program execution, examine it when suspended, view actual HTML DOM, edit the code, etc.

See Also

Procedures:

- Debugging
- Debugging JavaScript Unit Tests

Reference:

- Run/Debug Configuration: JavaScript Debug

External Links:

- http://wiki.jetbrains.net/intellij/Debugging_JavaScript_with_IntelliJ_IDEA

Web Resources:

- Developer Community