Running and Debugging Node.js

Running and debugging of Node.js applications in IntelliJ IDEA is supported through the *Node.js* plugin. Make sure you have installed and enabled the plugin before running or debugging.

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.

Running a Node.js application in IntelliJ IDEA is supported only in the *local* mode. This means that IntelliJ IDEA itself starts the Node.js engine and the target application according to a run configuration and gets full control over the session.

Debugging can be performed in two modes:

- Locally, with the Node.js engine started from IntelliJ IDEA.
- Remotely, when IntelliJ IDEA connects to an already running Node.js application. This approach gives you the possibility to re-start a debugging session without re-starting the Node.js server.

In either case, the debugging session is initiated through a debug configuration

You can also enable the *Live Edit* functionality during a Node.js debugging session. This functionality is provided through a `JavaScript Debug` run configuration, so technically you need to create two configurations: **Node.js** for initiating a debugging session and `JavaScript Debug` to have *Live Edit* at disposal.

In this section:

- Running a Node.js application.
- Debugging a Node.js application locally
- Debugging a running Node.js application
- Creating a Node.js run/debug configuration
- Enabling Live Editing during a Node.js debugging session
- Creating a Node.js Remote Debug configuration.

To run a Node.js application

1. Create a Node.js run configuration.
2. To launch the application, select the run configuration from the list on the main tool bar and then choose **Run | Run <configuration name>** on the main menu or click the **Run** toolbar button  .
3. The Run tool window opens, possibly showing the following information message:

   ```
   Server running at http://<host>:<port>/
   ```

4. Open the browser of your choice and open the page with the URL address generated through the server.listen function based on the `port` and `host` parameters. The page shows the result of executing your Node.js application.

To debug a Node.js application locally

1. Set the breakpoints in the Node.js code, where necessary. At least one breakpoint is necessary otherwise the program will be just executed. If you want the debugging tool to stop at the first line of your code, set a breakpoint at the first line.
2. Create a Node.js run/debug configuration.
3. To start a debugging session, select the debug configuration from the list on the main tool bar and then choose **Run | Debug <configuration name>** on the main menu or click the **Debug** toolbar button  .
4. In the Console tab of the Debug tool window, that opens, copy the URL address at which the application is running. The URL address is specified in the following information message:

   ```
   Server running at http://<host>:<port>/
   ```

5. Open the browser of your choice and open the page with the copied URL address. Control over the debugging session returns to IntelliJ IDEA.
6. Switch to IntelliJ IDEA, where the controls of the **Debug** tool window are now enabled. Proceed with the debugging session step through the breakpoints, switch between frames, change values on-the-fly, examine a suspended program, evaluate expressions, and set watches.

To debug a running Node.js application

With IntelliJ IDEA, you can connect to an already running Node.js applications. The application can be started either on the same machine or on a *physically remote host*.

When the application to debug is running on a *physically remote host*, you need to run a proxy or any other software that ensures port forwarding on the Node.js server. This is necessary because the debug port can open only on the `localhost` network interface. The `localhost` network interface cannot be accessed from another machine therefore IntelliJ IDEA cannot connect to it upon initiating a debugging session.

1. Make sure the application to debug has been launched in the target environment with the following parameters: `--debug-brk=<port through which the debugger on the remote host interacts with the network interface which accepts external connections>`

   Please note the following:

   - With the `--debug-brk` option, the execution of the application suspends right after launch. This option allows you to debug the code executed on start.
   - With the `--debug` option, the code that has to be executed on the application start is executed whereupon the application waits for a debugger to connect to it. This option is useful when you are not going to debug Node.js right now, but you want to debug it later.

2. Create a Node.js Remote Debug configuration: in the **Debug port** text box, type the port number through which you will interact with the remote host according to the *server access configuration*, see Creating a Remote Server Configuration.
3. With the application still running, launch the **Node.js Remote Debug** configuration (select the configuration in the list and click the **Debug** toolbar button ).
4. In the **Run** tool window, copy the URL address of the server and open the corresponding page in the browser. Control over the debugging session returns to IntelliJ IDEA.
5. Switch to IntelliJ IDEA. In the **Debug** tool window, step through the breakpoints, switch between frames, change values on-the-fly, examine a suspended program, evaluate expressions, and set watches.

To create a Node.js run/debug configuration

1. On the main menu, choose **Run | Edit Configurations**.
2. In the **Edit Configuration** dialog box, that opens, click the **Add New Configuration** toolbar button +, and choose **Node.js** on the context menu.
3. In the Run/Debug Configuration: Node.js dialog box, that opens, specify the following:
   - The name of the configuration.
   - The path to the Node.js executable file.

     This field is already filled in if you have appointed a default installation.
   - To enable remote debugging of the application, specify the following option in the **Node parameters** text box: `--debug-brk=<port for connect to debugger remotely>`

     - With the `--debug-brk` option, the execution of the application suspends right after launch. This option allows you to debug the code executed on start.
     - With the `--debug` option, the code that has to be executed on the application start is executed whereupon the application waits for a debugger to connect to it. This option is useful when you are not going to debug Node.js right now, but you want to debug it later.
   - The location of the file to start running the Node.js application.
   - If the file to run references any other files, specify their location in the **Working directory** field.
   - If applicable, in the **Application parameters** text box, specify the arguments to be passed to the application on start through the process.argv array.
4. If necessary, enable the *Live Edit* functionality during a Node.js debugging session. This functionality is provided through a `JavaScript Debug` run configuration, so technically you need to create two configurations: **Node.js** for initiating a debugging session and `JavaScript Debug` to have *Live Edit* at disposal.
5. Click **OK**, when ready.

To enable Live Editing during a Node.js debugging session

You can enable the *Live Edit* functionality during a Node.js debugging session. This functionality is provided through a `JavaScript Debug` run configuration, so technically you need to create two configurations: **Node.js** for initiating a debugging session and `JavaScript Debug` to have *Live Edit* at disposal.

1. Start creating a Node.js debug configuration.
2. Switch to the **Browser / Live Edit** tab.
3. Select the **After launch** check box to have a browser started automatically after a debugging session is launched. Specify the browser to use in the drop-down list next to the check box.
   - To use the system default browser, choose **Default**.
   - To use a custom browser, choose it from the list. Note that *Live Edit* is fully supported only in Chrome.
   - To configure browsers, click the **Browse** button ... and adjust the settings in the **Web Browsers** dialog box that opens. For more information, see Configuring Browsers.
4. Select the **With JavaScript debugger** check box to enable the JavaScript debugger in the selected browser.

To create a Node.js Remote Debug configuration

1. On the main menu, choose **Run | Edit Configurations**.
2. In the **Edit Configuration** dialog box, that opens, click the **Add New Configuration** toolbar button +, and choose **Node.js Remote Debug** on the context menu.
3. In the Run/Debug Configuration: Node.js Remote Debug dialog box, that opens, specify the following:
   - The name of the configuration.
   - The host where the target application is running.
   - The port to connect to. Copy the port number from the information message in the Run tool window that controls the running application.
4. Click **OK**, when ready.

See Also

Concepts:

- Running, Debugging and Testing

Procedures:

- Node.js
- Running
- Debugging
- Creating and Editing Run/Debug Configurations

Reference:

- Run/Debug Configuration: Node JS
- Run/Debug Configuration: Node JS Remote Debug
- Run Tool Window
- Debug Tool Window

Web Resources:

- Developer Community