
Debug Chrome with Command

1 message

Michael Tadyshak <michael.tadyshak@nfm.com> Fri, Aug 23, 2024 at 2:03 PM
To: Michael Tadyshak <mike.tadyshak@gmail.com>

ChatGPT 4o mini prompt:

how to configure VS Code to debug a Javascript / HTML / CSS web application?

Debugging does not work with Live Server unless the Debugger for Chrome extension is running

Additional Tips

- **Ensure Source Maps Are Enabled**: If you're using tools like Webpack, ensure that source maps are enabled in your build configuration. This allows VS Code to map the minified code in the browser back to your original source files.
- **Check the Console**: If you encounter issues, check the Debug Console in VS Code for error messages or additional information about what's going wrong.
- **Browser Sync**: If you make changes to your code and want to see them immediately, consider using tools like Live Server (for static sites) or BrowserSync (for dynamic applications) that automatically refresh the browser when files change.

//

Live Server (Optional)

For a simpler setup without advanced debugging features, you can use the "Live Server" extension:

- Install the **Live Server** extension from the Extensions view.
- Right-click on your HTML file in the Explorer and select "Open with Live Server."

This will start a local server and open your application in the browser, but it won’t provide the same debugging capabilities as the Debugger for Chrome.

//

By default, the Debugger for Chrome extension in VS Code connects to port 9222

//

To configure Visual Studio Code (VS Code) to launch Chrome with specific command-line arguments, such as --remote-debugging-port=9222, you need to use a combination of launch.json for debugging and a separate setup for starting Chrome with those arguments.

Here's a step-by-step guide to set it up:

1. Install the Required Extensions

Make sure you have the **Debugger for Chrome** extension installed:

- Open VS Code.
- Go to the Extensions view by clicking the square icon in the sidebar or pressing Ctrl+Shift+X (Cmd+Shift+X on macOS).
- Search for "Debugger for Chrome" and install it.

2. Create a launch.json Configuration

This file tells VS Code how to launch and debug your application. Here’s how to configure it:

1. **Open Your Project in VS Code**.
2. **Open the Debug View**:
    - Click the play icon in the sidebar or press Ctrl+Shift+D (Cmd+Shift+D on macOS).
3. **Create or Edit launch.json**:
    - Click the gear icon (⚙) to open or create a launch.json file.
    - Select "Chrome" if prompted.
4. **Add a Launch Configuration**:

- Use the following configuration to launch Chrome with the desired command-line arguments:

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "chrome",
      "request": "launch",
      "name": "Launch Chrome with Debugging",
      "url": http://localhost:8000, // Replace with your app's URL
      "webRoot": "${workspaceFolder}",
      "runtimeExecutable": "/path/to/chrome", // Adjust for your OS
      "runtimeArgs": [
        "--remote-debugging-port=9222",
        "--new-window"
      ],
      "sourceMapPathOverrides": {
        "webpack:///*": "${webRoot}/*",
        "webpack:///./~/*": "${webRoot}/node_modules/*"
      }
    }
  ]
}
```

1. **Details**:
   - **url**: The URL of the application you want to debug. Adjust this to match your setup.
   - **webRoot**: The root directory of your web application.
   - **runtimeExecutable**: Path to the Chrome executable. Replace /path/to/chrome with the correct path for your operating system:
      - **Windows**: "C:/Program Files/Google/Chrome/Application/chrome.exe"
      - **macOS**: "/Applications/Google Chrome.app/Contents/MacOS/Google Chrome"

        - **Linux**: Typically just "google-chrome" or "google-chrome-stable"
    - **runtimeArgs**: Command-line arguments for launching Chrome. --remote-debugging-port=9222 enables debugging, and --new-window ensures Chrome opens in a new window.

3. Launch and Debug

1. **Run Your Application**:
    - Start your application on the specified URL (e.g., http://localhost:8000) if it’s not already running.
2. **Start Debugging**:
    - In VS Code, go to the Debug view.
    - Select "Launch Chrome with Debugging" from the dropdown.
    - Click the green play button or press F5 to start the debugging session.

4. Optional: Set Up a Task to Launch Chrome Separately

If you want to separately launch Chrome with specific arguments and then attach VS Code to it:

1. **Create a Task**:
    - Open or create .vscode/tasks.json and add a task to launch Chrome:

json

```
{
  "version": "2.0.0",
  "tasks": [
    {
      "label": "Launch Chrome",
      "type": "shell",
      "command": "/path/to/chrome", // Adjust for your OS
      "args": [
        "--remote-debugging-port=9222",
        "--new-window",
        http://localhost:8000 // URL of your app
      ],
      "windows": {
        "command": "C:\\Program Files\\Google\\Chrome\\Application\\chrome.exe" // Adjust for Windows
      }
    }
  ]
}
```

1. **Run the Task**:
    - Open the Command Palette (Ctrl+Shift+P or Cmd+Shift+P on macOS).
    - Type Tasks: Run Task, and select the "Launch Chrome" task.

Summary

By configuring launch.json with the runtimeArgs field or setting up a task, you can launch Chrome with the desired command-line arguments directly from VS Code. This setup streamlines the debugging process, integrating Chrome's debugging features with VS Code's powerful development environment.

Michael Tadyshak
Customer Service Specialist
P: +19726683000 Ext. 67265
michael.tadyshak@nfm.com
Omaha | Des Moines | Kansas City | Dallas/Fort Worth | Austin - Opening 2026

Think before you print

Newsweek's
America's Best Retailer
Home Goods - Furniture
2024
WINNER