VS Code Debugger

Debugging Playwright tests in Visual Studio Code (VS Code) allows you to efficiently troubleshoot issues by stepping through the code, inspecting variables, and controlling the flow of execution. Here’s how you can set up and use the VS Code debugger with Playwright.

**1. Setting Up VS Code Debugger**

First, ensure that you have installed Playwright and the VS Code extension for Playwright. Then, create a launch.json file to configure the debugger in VS Code.

* In VS Code, press Ctrl + Shift + P and search for “Debug: Open launch.json”.
* Add the following configuration for debugging Playwright tests:

json

Copy code

{

"version": "0.2.0",

"configurations": [

{

"name": "Debug Playwright Tests",

"type": "node",

"request": "launch",

"program": "${workspaceFolder}/node\_modules/.bin/playwright",

"args": ["test"],

"console": "integratedTerminal",

"internalConsoleOptions": "neverOpen"

}

]

}

This configuration tells VS Code to launch Playwright tests and allows you to debug the test scripts interactively.

**2. Running the Debugger**

With the configuration in place, you can now debug Playwright tests directly in VS Code:

1. Set breakpoints by clicking in the gutter next to the line numbers in your test script.
2. Open the **Run and Debug** tab in VS Code and select **Debug Playwright Tests**.
3. Click the **Start Debugging** button or press F5.

The Playwright tests will run, and the execution will pause at the breakpoints you’ve set, allowing you to inspect the application state.

**3. Inspecting Variables and Flow Control**

When the test execution hits a breakpoint, you can:

* **Inspect variables** in the **Variables** section of the VS Code debug sidebar.
* **Evaluate expressions** in the **Debug Console** to check specific values or run code snippets.
* **Step through** the code line-by-line using the control buttons (Step Over, Step Into, Step Out).

Here’s an example test:

js

Copy code

const { test, expect } = require('@playwright/test');

test('Example VS Code Debugger Test', async ({ page }) => {

await page.goto('https://example.com');

const title = await page.title();

console.log(title); // Set a breakpoint here to inspect the title

expect(title).toBe('Example Domain');

});

When you run this in the debugger, you can pause at console.log(title); and inspect the title value before moving forward.

**4. Advanced Features**

* **Conditional Breakpoints**: Right-click a breakpoint and select **Edit Breakpoint** to set conditions (e.g., stop only if a variable has a specific value).
* **Watch Expressions**: Add variables or expressions to the **Watch** section for real-time evaluation as you step through the test.

**5. Headed Mode for Visual Debugging**

By default, Playwright runs tests in headless mode. To debug with the browser UI open (headed mode), modify your test launch command in the launch.json or use the following code:

js

Copy code

const { chromium } = require('playwright');

(async () => {

const browser = await chromium.launch({ headless: false });

const page = await browser.newPage();

await page.goto('https://example.com');

await browser.close();

})();