Playwright Test generator(Codegen)

What is Codegen in Playwright?

Codegen in Playwright is a built-in tool that **automatically generates test code** based on your actions in the browser.

Think of Codegen like a **test recorder**.

1. You open a browser using Codegen.
2. You interact with the website (like clicking buttons, typing in fields, etc.).
3. Playwright watches what you do and **writes the test code for you**!

Why Use Codegen?

- Saves time — no need to write test code manually.
- Helps beginners learn how Playwright test scripts are written.
- Great for quickly generating test scripts for UI flows.

How to Use Codegen

Open your terminal and run:

```
npx playwright codegen https://example.com
```

This will:

- Launch a browser window.
- Open Playwright Inspector side by side.
- Record your clicks, inputs, and navigation.
- Show the **generated code live** as you interact.

Codegen Commands

1. Launch Browser & Playwright Inspector

This opens a blank browser window along with the Playwright Inspector:

```
npx playwright codegen
```

2. Launch with a Specific URL

Opens the specified webpage directly and starts recording actions:

```
npx playwright codegen https://example.com
```

3. Record and Save Code to File

Generates the recorded script and saves it to a specific file path.

Options:

```
npx playwright codegen -o tests/mytest.spec.ts
npx playwright codegen --output tests/mytest.spec.ts
npx playwright codegen --output=tests/mytest.spec.ts
```

All three variants do the same job—use whichever you prefer.

4. Specify Browser Type

You can choose a specific browser engine (chromium, firefox, or webkit) for recording.

Examples:

```
npx playwright codegen --browser chromium
npx playwright codegen --browser=firefox
npx playwright codegen -b chromium
npx playwright codegen -b=firefox
```

5. Emulate a Specific Device

Start recording in the emulated view of a specific device.

Examples:

```
npx playwright codegen --device "iPhone 15"
npx playwright codegen --device='iPhone 15'
```

6. Set Custom Browser Viewport Size

Customize the browser window's viewport dimensions during recording:

```
npx playwright codegen --viewport-size "1280,720"
```

Format: "width,height"

7. Combine Multiple Options

You can combine all supported options in a single command for full control.

Example:

```
npx playwright codegen --browser=chromium --output=tests/mytest.spec.ts --viewport-size "1280,720" https://example.com
```

This command:

- Opens Chromium
- Navigates to https://example.com
- Sets the viewport to 1280x720
- Records and saves the script to tests/mytest.spec.ts

Summary Table

Purpose	Command Example
Open browser & inspector	npx playwright codegen
Open with a URL	npx playwright codegen https://example.com
Save to file	--output tests/test.spec.ts
Choose browser	--browser=chromium
Emulate device	--device "iPhone 15"
Set viewport size	--viewport-size "1280,720"

Debugging Playwright Tests

When writing automated tests, it's common to run into issues where things don't work as expected. Playwright provides a powerful tool to help you troubleshoot — **Playwright Inspector**.

What is Playwright Inspector?

The **Playwright Inspector** is a graphical tool that helps you **debug your tests step by step**. It lets you:

- Run your test one step at a time (play/pause/step).
- Pick and test locators interactively.
- See actionability logs and real-time browser highlights.
- Make live edits to selectors.

How to Start Debugging

To open the Playwright Inspector, run the following command in your terminal:

```
npx playwright test tests/mytest.spec.ts --debug
```

This launches your test in debug mode and opens the Inspector window.

Stepping Through Tests

Once inside the Inspector:

- Use the **toolbar** to play, pause, or step through each action.
- The **currently executing line of code** will be highlighted.
- The **targeted elements** on the page will also be highlighted in the browser.

This helps you understand exactly what each part of your test is doing.

Pause at a Specific Step with page.pause()

Instead of stepping through every line, you can tell Playwright exactly **where to pause** by adding:

```
await page.pause();
```

This acts like a **breakpoint** in your test.

- Add it right before the part you want to inspect.
- Then run the test in debug mode (as shown above).
- The test will pause at page.pause(), and you can interact with the browser and test code in the Inspector.
- Click **"Resume"** when you're ready to continue.