Playwright Reporters

In Playwright, **reporters** are used to format and display the test results.

They help visualize the output in different styles, such as a simple list, a single line, dots, or even as an HTML or JSON file.

1. HTML Reporter

The **HTML reporter** provides a rich visual interface to explore test results in the browser.

Default Behavior:

- Opens automatically **only when tests fail**.

Control Open Behaviour:

You can customize when the HTML report should open using the **open** option:

Option	Behaviour
open: 'never'	Never opens the report automatically
open: 'always'	Always opens the report
open: 'on-failure'	Opens only when a test fails (**default**)

Configuration Example:

```
export default defineConfig({
  reporter: [['html', { open: 'never' }]],
});
```

Save Report to Custom Folder:

```
export default defineConfig({
  reporter: [['html', { open: 'never', outputFolder: 'my-report' }]],
});
```

View the Report:

- If using default folder:

```
npx playwright show-report
```

- If using custom folder:

```
npx playwright show-report my-report
```

Set via Command Line:

```
npx playwright test --reporter=html
```

2. List Reporter (Default)

The **List reporter** prints one line per test. It’s the default option when running tests.

Configuration:

```
export default defineConfig({
  reporter: 'list',
});
```

Or via CLI:

```
npx playwright test --reporter=list
```

3. Line Reporter

The **Line reporter** displays only the last executed test on a single line, and shows failures when they occur—more compact than the list reporter.

Configuration:

```
export default defineConfig({
  reporter: 'line',
});
```

Or via CLI:

```
npx playwright test --reporter=line
```

4. Dot Reporter

The **Dot reporter** is extremely concise—it shows one character (dot) per test. It's ideal for minimal terminal output.

Configuration:

```
export default defineConfig({
  reporter: 'dot',
});
```

Or via CLI:

```
npx playwright test --reporter=dot
```

5. JSON Reporter

The **JSON reporter** outputs test results in JSON format, useful for programmatic processing or integrations.

Configuration:

```
export default defineConfig({
  reporter: [['json', { outputFile: 'results.json' }]],
});
```

Or via CLI:

```
npx playwright test --reporter=json
```

6. JUnit Reporter

The **JUnit reporter** creates results in XML format, compatible with tools like Jenkins and CI systems.

Configuration:

```
export default defineConfig({
  reporter: [['junit', { outputFile: 'results.xml' }]],
});
```

Or via CLI:

```
npx playwright test --reporter=junit
```

7. Custom Reporter

You can build your own **custom reporter** by implementing the Playwright **Reporter** interface.

Example (my-awesome-reporter.ts):

```
import type { FullConfig, FullResult, Reporter, Suite, TestCase, TestResult } from
'@playwright/test/reporter';
class MyReporter implements Reporter {
  onBegin(config: FullConfig, suite: Suite) {
    console.log(`Starting the run with ${suite.allTests().length} tests`);
  }
  onTestBegin(test: TestCase, result: TestResult) {
    console.log(`Starting test ${test.title}`);
  }
  onTestEnd(test: TestCase, result: TestResult) {
    console.log(`Finished test ${test.title}: ${result.status}`);
  }
  onEnd(result: FullResult) {
    console.log(`Finished the run: ${result.status}`);
  }
}
export default MyReporter;
```

Use in Config:

```
export default defineConfig({
  reporter: [['./my-awesome-reporter.ts', { customOption: 'some value' }]],
});
```

Or via CLI:

```
npx playwright test --reporter="./myreporter/my-awesome-reporter.ts"
```

We can configure all Playwright reporters together in a single playwright.config.ts file:

```
export default defineConfig({
  reporter: [
    ['list'], // Default list reporter
    ['line'], // Line reporter
    ['dot'],  // Dot reporter
    ['html', { open: 'never', outputFolder: 'html-report' }], // HTML reporter with custom folder
    ['json', { outputFile: 'results.json' }], // JSON reporter
    ['junit', { outputFile: 'results.xml' }], // JUnit reporter
    [MyReporter], // Custom reporter
  ],
});
```

Generate Allure Reports in Playwright

Allure Reports provide beautiful, detailed test execution reports for your Playwright automation tests.

Pre-requisites

Install Allure CLI

You need the **Allure Command Line Interface (CLI)** to generate and view reports.

Install via npm:

```
npm install -g allure-commandline --save-dev
```

Note: After installing, make sure the path is added to your system's environment variables → Path.

Install Allure Playwright Reporter

This package connects Playwright with Allure.

Install using:

```
npm install -D allure-playwright
```

This adds it as a **dev dependency** in your project.

Steps to Generate and View Allure Reports

Step 1: Configure the Allure Reporter

You can choose **either** of the two options below:

Option 1: Update Playwright Config File (playwright.config.ts or .js)

Add the reporter config:

```
export default defineConfig({
  reporter: 'allure-playwright',
});
```

Option 2: Use Command Line (Without Changing Config File)

Run tests with the reporter via CLI:

npx playwright test --reporter=allure-playwright

Step 2: Generate and Open the Report

After test execution, Playwright stores Allure results in the **./allure-results** folder.

Generate the HTML report:

```
allure generate ./allure-results -o ./allure-report
```

Or, to clean old reports:

```
allure generate ./allure-results -o ./allure-report --clean
```

Open the report in your browser:

```
allure open ./allure-report
```

This command launches a browser showing a **detailed test report**, including:

- Step-by-step logs
- Screenshots
- Attachments
- Test durations