Running Playwright Tests

Introduction

Playwright is a powerful end-to-end testing framework for web applications. It allows running tests on multiple browsers in both headless and headed modes. There are two ways to run tests in Playwright:

1. **Using Playwright Test Runner (`@playwright/test`)** – This approach uses Playwright's built-in test framework with configuration support .
2. **Using Playwright API (Standalone Mode)** – This approach allows direct browser control without the test runner.

Below are detailed instructions on executing Playwright tests using different commands.

---

1. Running Tests with Playwright Test Runner (`@playwright/test`)

Running All Tests in Headless Mode

To execute all Playwright tests across all browsers in headless mode, use the following command:

```
npx playwright test
```

Explanation:

- Runs all test files present in the test directory.
- By default, it runs in headless mode across all browsers.
- Uses `playwright.config.js` for configuration.
- Automatically detects available browsers.

Running a Specific Test File

To run a specific test file, use:

```
npx playwright test MyTest.spec.js
```

Explanation:

- Executes only the test cases in `MyTest.spec.js`.
- Runs in headless mode by default.
- Test execution is controlled by Playwright Test Runner.

Running Tests on a Specific Browser

To run a specific test file on a particular browser, use:

```
npx playwright test MyTest.spec.js --project=chromium
```

Explanation:

- Runs `MyTest.spec.js` using the Chromium browser.
- The `--project` flag specifies the browser (e.g., `chromium`, `firefox`, or `webkit`).
- Uses the browser configurations from `playwright.config.js`.

Running Tests in Headed Mode

To run tests with a visible browser UI, use:

```
npx playwright test MyTest.spec.js --project=chromium --headed
```

Explanation:

- Executes `MyTest.spec.js` in Chromium with the browser UI visible.
- Useful for debugging test failures.
- Helps observe test execution visually.

Running Tests in Debug Mode

For debugging, use the following command:

```
npx playwright test MyTest.spec.js --project=chromium --headed --debug
```

Explanation:

- Runs `MyTest.spec.js` in Chromium in headed mode.
- Opens an additional Playwright Inspector window, showing the script execution in real-time.
- `--debug` pauses the test execution at each step for inspection.
- Useful for debugging test scripts interactively.
- Provides enhanced debugging with step-by-step execution.

---

2. Running Tests with Playwright API (Standalone Mode)

When using Playwright API directly without the test runner, use:

```
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();
})();
```

Explanation:

- This approach does **not** use `@playwright/test` but directly controls browsers programmatically.
- The CLI commands (`npx playwright test`, `--project`, etc.) **do not apply** here.
- Instead, you manually launch browsers and write custom scripts using Playwright's API.
- To run such scripts, use `node myscript.js` instead of `npx playwright test`.

Running in Headless or Headed Mode

```
const { chromium } = require('playwright');
(async () => {
  const browser = await chromium.launch({ headless: false }); // Set to
true for headless
  const page = await browser.newPage();
  await page.goto('https://example.com');
  await browser.close();
})();
```

Explanation:

- `headless: false` runs the browser in headed mode (UI visible).
- `headless: true` runs the browser in headless mode.

Running in Debug Mode

```
const { chromium } = require('playwright');
(async () => {
  const browser = await chromium.launch({ headless: false, slowMo: 1000 });
// Adds delay for debugging
  const page = await browser.newPage();
  await page.goto('https://example.com');
  await page.pause(); // Opens Playwright Inspector for debugging
  await browser.close();
})();
```

Explanation:

- `slowMo: 1000` slows down execution to observe steps.
- `page.pause()` opens an interactive debugging session.

Key Differences

Feature	`@playwright/test`	Playwright API
Uses `npx playwright test`	✅ Yes	❌ No
Uses `playwright.config.js`	✅ Yes	❌ No
Provides CLI arguments (`--headed`, `--project`)	✅ Yes	❌ No
Runs via `node script.js`	❌ No	✅ Yes

If you're using the **Playwright API** (without `@playwright/test`), you'd write and execute tests differently.