JavaScript vs TypeScript for Playwright Execution

Feature	JavaScript	TypeScript
Setup	Minimal	Requires TypeScript compiler (tsc) and config
Execution	Directly with Node.js	Needs compilation to JavaScript before running
Type Safety	None	Strong typing helps catch errors early
IDE Support	Good	Excellent (autocompletion, refactoring, inline docs)
Scalability	Suitable for small projects	Ideal for large, maintainable codebases
Test Writing	Simple syntax	More robust with type definitions
Learning Curve	Easier for beginners	Slightly steeper due to types and config

Technology Stack Overview

Language: TypeScript (typed superset of JavaScript)

Framework: Playwright Test

Runtime: Node.js

IDE: Visual Studio Code

Package Manager: npm

Test Runner: Playwright's built-in test runner

Reporting: HTML reports, trace viewer, screenshots, videos

Supported Browsers and Platforms

Browsers:

Chromium (Chrome, Edge)

Firefox

WebKit (Safari)

Platforms:

Windows

macOS

Linux

CI/CD environments (GitHub Actions, Azure DevOps, etc.)

Mobile emulation (iPhone)

Recommended Locators:

---

page.getByRole() – based on ARIA roles

page.getByLabel() – for form inputs

page.getByText() – for visible text

page.getByPlaceholder() – for input placeholders

page.locator() – for custom selectors

XPath vs CSS Selectors

Feature--- XPath--- CSS Selectors

Syntax ---Complex--- Simple and readable

Performance --Slower---- Faster

Browser Support--- Limited in some cases--- Widely supported

Use Case ---- Traversing complex DOM trees--- Styling and basic DOM access

Best Practice:

Prefer CSS selectors or Playwright's built-in locators over XPath for readability and performance.

Assertion

==========

In Playwright with TypeScript, assertions are used to validate expected outcomes during automated testing. Playwright uses the expect API from its built-in test runner (@playwright/test) to perform assertions.

Here's a categorized list of commonly used assertions:

1.Basic Assertions

These are general-purpose assertions used to validate values or conditions.

```
import { expect ,test} from '@playwright/test';

test('test', async () => {

const value = 5;

expect(value).toBeGreaterThan(3);

expect(value).toEqual(5);

expect(value).not.toBeNull();

});
```

2. Element Assertions

These are used to validate the state or properties of DOM elements.

```
import { test, expect } from '@playwright/test';


test('element assertions', async ({ page }) => {
  await page.goto('https://playwright.dev');


  const getStartedLink = page.getByRole('link', { name: 'Get started' });
```

```
  await expect(getStartedLink).toBeVisible();
  await expect(getStartedLink).toHaveText('Get started');
  await expect(getStartedLink).toHaveAttribute('href', '/docs/intro');
});
```

Page-Level Assertions

These validate properties of the entire page

```
test('page assertions', async ({ page }) => {
  await page.goto('https://playwright.dev');

  await expect(page).toHaveTitle(/Playwright/);
  await expect(page).toHaveURL('https://playwright.dev/');
});
```

==========

Locator Concepts

xpath:

```
import { test, expect } from '@playwright/test';


test('Locate element by XPath on GitLab', async ({ page }) => {
        await page.goto('https://gitlab.com'); // Replace with your target website

        // Locate the element using XPath
  //  const signInButton = page.locator('//a[text()="Sign in"]');
```

```
    const signInButton = page.locator('//*[@id="be-navigation-desktop"]/div/div/div[2]/div/a');

    // Verify the element is visible
    await expect(signInButton).toBeVisible();

    // Perform actions on the located element
    await signInButton.click();
});
```

==========

text content:

```
import { test, expect } from '@playwright/test';

test('Locate element by text content on Sauce Labs', async ({ page }) => {
    await page.goto('https://saucelabs.com'); // Replace with your target website

    // Locate the element by its text content
    const element = page.locator('text=Sign In');

    // Verify the element is visible
    await expect(element).toBeVisible();

    // Perform actions on the located element
    await element.click();
});
```

==========

RoleBy:

```
import { test, expect } from '@playwright/test';

test('locate by role', async ({ page }) => {
  await page.goto('https://www.google.com');
  await page.getByRole('button', { name: 'Google Search' }).click();
  await expect(page.getByRole('button', { name: 'Google Search' })).toBeVisible();
});
```

==========

page.getByPlaceHolder()

==========

Selects form input elements (like textboxes, checkboxes, etc.) by their associated label text.

```
await page.getByLabel('Username').fill('myuser');
import { test, expect } from '@playwright/test';

test('Login using various locator strategies', async ({ page }) => {
  // Navigate to the login page
  await page.goto('https://www.saucedemo.com');

  //getByPlaceholder – for input fields with placeholder text
  await page.getByPlaceholder('Username').fill('standard_user');
  await page.getByPlaceholder('Password').fill('secret_sauce');

  //getByRole – for accessible roles like buttons
```

```
  await page.getByRole('button', { name: 'Login' }).click();

  //getByText – for visible text on the page
  await expect(page.getByText('Products')).toBeVisible();

  //locator – for custom CSS selectors
  const productTitle = page.locator('.title');
  await expect(productTitle).toHaveText('Products');
});
```

==========

Locating by Attributes:

```
import { test, expect } from '@playwright/test';

test('Sauce Labs login test', async ({ page }) => {
        await page.goto('https://www.saucedemo.com');

        // Type into the username field using an attribute selector
        await page.fill('input[data-test="username"]', 'standard_user');

        // Type into the password field using an attribute selector
        await page.fill('input[data-test="password"]', 'secret_sauce');

        // Click the "Login" button using an attribute selector
        await page.click('input[data-test="login-button"]');

        // Verify that the login was successful by checking the URL
```

```
    await expect(page).toHaveURL('https://www.saucedemo.com/inventory.html');
});
```

==========

Locate by Classname:

```
import { test, expect } from '@playwright/test';

test('Sauce Labs login test', async ({ page }) => {
    await page.goto('https://www.saucedemo.com');

    // Type into the username field using a class name selector
    await page.fill('.input_error.form_input', 'standard_user');

    // Type into the password field using a class name selector
    await page.fill('.input_error.form_input[type="password"]', 'secret_sauce');

    // Click the "Login" button using a class name selector
    await page.click('.btn_action');

    // Verify that the login was successful by checking the URL
    await expect(page).toHaveURL('https://www.saucedemo.com/inventory.html');
});
```

Input textbox automation

```
import { test, expect, chromium } from '@playwright/test';

test('Sauce Labs login test', async () => {
  // Launch Chromium browser with headless mode disabled
  const browser = await chromium.launch({ headless: false });
  const page = await browser.newPage();

  // Navigate to Sauce Demo
  await page.goto('https://www.saucedemo.com');

  // Type into the username field using an ID selector
  await page.fill('#user-name', 'standard_user');

  // Type into the password field using an ID selector
  await page.fill('#password', 'secret_sauce');

  // Click the "Login" button using an ID selector
  await page.click('#login-button');

  // Verify that the login was successful by checking the URL
  await expect(page).toHaveURL('https://www.saucedemo.com/inventory.html');

  // Close the browser
  await browser.close();
});

=========================================================================
=========
```