

Playwright using JavaScript

Introduction

Just like **Selenium**, **Playwright** supports:

- ✓ **Multiple Operating Systems** (Windows, Mac, Linux)
- ✓ **Multiple Languages** (JavaScript, Python, Java, etc.)
- ✓ **Multiple Browsers** (Chrome, Edge, Safari, Firefox)



 **Official Website** – playwright.dev

Playwright is used for testing **modern web applications**.

It can also automate **APIs** and **browser-based applications on mobile devices**. However, it does **NOT** support **mobile native apps**.


What is a Native App?

A **native app** is an application built for a specific mobile operating system, like:


-  **Android** – Built using Java/Kotlin
-  **iOS** – Built using Swift/Objective-C

For example,

1. Purely Native App (Not Browser-Based)

-  **Snapchat** – Only works as a native mobile app, no browser-based version.

2. Purely Browser-Based Mobile App (No Native Version)

-  **Google Search (mobile site: google.com)** – Fully web-based, no need to install an app.

Since **Playwright** can only automate **browser-based apps**, it can work with **Google Search (mobile site)** but **not** with **Snapchat**.

What Features Does Playwright Support?

1. Supports Different Types of Applications

- **Web Apps** – Example: Automating the NBCU login page in a web browser.
- **Mobile Web Apps** – Example: Testing NBCU's website in a mobile browser like Chrome or Safari.
- **API Automation** – Example: Testing NBCU's backend APIs for fetching movie details.

2. Supports Multiple Programming Languages

Playwright works with:

- **JavaScript** (Most commonly used)
- **TypeScript**
- **Java**
- **Python**
- **.NET (C#)**

Why is **JavaScript Mostly Used** in Playwright?

- Playwright is built using **JavaScript**.
- Most web applications are developed using JavaScript.
- JavaScript makes Playwright tests run faster and integrate easily with web apps.

3. Supports Multiple Browsers

Playwright can run tests on:

- **Google Chrome (Chromium-based)**
- **Microsoft Edge**
- **Safari (for Mac/iOS)**
- **Firefox**
- All browsers support **headed mode (with UI)** and **headless mode (without UI, runs faster)**.

What is **Chromium**?

Chromium is an open-source browser engine. It powers:

- **Google Chrome**
- **Microsoft Edge**
- **Opera**
- **Brave**

4. Supports Multiple Operating Systems



Playwright runs on:

- **Windows**
 - **MacOS**
 - **Linux**
 - **CI/CD Environments** (Example: Running Playwright tests in **Jenkins, GitHub Actions, or Azure DevOps**).
-

Why Prefer Playwright Over Selenium?

Comparison of Features

1. Free and Open Source

- **Selenium** –  Yes, Selenium is an open-source automation framework.
 - **Playwright** –  Yes, Playwright is also open-source and maintained by Microsoft.
-

2. Supported Browsers

- **Selenium** – Supports **Chrome, Edge, Firefox, Safari, and Opera**.
 - **Playwright** – Supports **Chromium (Chrome & Edge), Firefox, WebKit (Safari)**.
-


3. Programming Language Support

- **Selenium** – Supports **Java, Python, C#, JavaScript, Ruby, PHP**.
 - **Playwright** – Supports **JavaScript, TypeScript, Java, Python, C# (.NET)**.
-

4. Supported Operating Systems

- **Selenium** – Supports **Windows, MacOS, Linux**.
 - **Playwright** – Supports **Windows, MacOS, Linux + CI/CD environments**.
-

5. Headless Mode (Runs Without Opening the Browser UI)

- **Selenium** –  Yes, using **ChromeOptions**.

```
ChromeOptions options = new ChromeOptions();
options.addArguments("--headless");
WebDriver driver = new ChromeDriver(options);
```


- **Playwright** –  **Yes, using headless mode by default.**

```
const { chromium } = require('playwright');
(async () => {
  const browser = await chromium.launch({ headless: true });
  const page = await browser.newPage();
  await page.goto('https://www.peacocktv.com');
  console.log(await page.title());
  await browser.close();
})();
```

6. Cross Browser & Cross OS Testing


- **Selenium** –  **Yes, using config.properties & Selenium Grid.**

```
browser=chrome
platform=windows
DesiredCapabilities cap = new DesiredCapabilities();
cap.setBrowserName("chrome");
cap.setPlatform(Platform.WINDOWS);
WebDriver driver = new RemoteWebDriver(new
URL("http://localhost:4444"), cap);
```


- **Playwright** –  **Yes, using Playwright Test Runner.**

```
npx playwright test --project=chromium
npx playwright test --project=firefox
```

7. Parallel Execution

- **Selenium** –  **Yes, using TestNG.**
testng.xml:

```
<suite name="NBCU Test Suite" parallel="tests" thread-count="2">
  <test name="Login Test">
    <classes>
      <class name="tests.LoginTest"/>
    </classes>
  </test>
  <test name="Search Test">
    <classes>
      <class name="tests.SearchTest"/>
    </classes>
  </test>
</suite>
```

- **Playwright** –  Yes, built-in parallel execution.


```
test.describe.parallel('NBCU Tests', () => {
  test('Login Test', async ({ page }) => {
    await page.goto('https://www.peacocktv.com');
    // Perform login test
  });
});
```

8. Auto Waits (Handles Element Waiting Automatically)

- **Selenium** –  No, needs **Implicit & Explicit Waits**.


```
// Implicit Wait
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

// Explicit Wait
WebDriverWait wait = new WebDriverWait(driver,
Duration.ofSeconds(10));
WebElement element =
wait.until(ExpectedConditions.elementToBeClickable(By.id("start")));
```


- **Playwright** –  Yes, **Auto-Waiting** is built-in.

```
await page.click('#start'); // Automatically waits for the element to
be clickable
```

9. Multi-Tab & Multi-Window Handling

- **Selenium** –  Yes, using `switchTo().window()`.

```
Set<String> windows = driver.getWindowHandles();
for (String window : windows) {
  driver.switchTo().window(window);
}
```

- **Playwright** –  Yes, using `context.newPage()`.

```
const newPage = await context.newPage();
await newPage.goto('https://www.peacocktv.com');
```

10. Real-Time Test Execution

● Only Cypress supports real-time execution.

✦ What is real-time execution?

It means seeing the test run **step-by-step in real-time**, like watching a video. Selenium & Playwright don't have this feature.

? Why only Cypress supports real-time execution?

Cypress runs directly in the browser and provides a **real-time dashboard** with automatic reloads.

11. Debugging

- **Selenium** – ✓ Yes, using **DevTools integration**.

```
ChromeOptions options = new ChromeOptions();
options.setExperimentalOption("debuggerAddress", "localhost:9222");
```

- **Playwright** – ✓ Yes, has **built-in debuggers**.

```
await page.pause(); // Opens Playwright Inspector
```

12. API Testing

- **Selenium** – ✗ No, needs **RestAssured** for API testing.

```
Response response =
RestAssured.get("https://api.peacocktv.com/movies");
```

- **Playwright** – ✓ Yes, built-in API testing support.

```
const response = await
request.get('https://api.peacocktv.com/movies');
console.log(await response.json());
```

13. Supported Frameworks

- **Selenium** – Works with **TestNG, JUnit, Cucumber**.
 - **TestNG** – Framework for running parallel tests.
 - **Cucumber** – BDD framework for writing tests in Gherkin.
- **Playwright** – Works with **Jest, Mocha**.
 - **Jest** – Fast JavaScript test framework.
 - **Mocha** – Flexible JS test framework.

- **Common Framework Support** –  **Both** support integration with **CI/CD tools** like Jenkins & GitHub Actions.
-

Why is Playwright Faster than Selenium?

- ◆ Selenium interacts with browsers through drivers (e.g., ChromeDriver, GeckoDriver).
- ◆ Playwright directly controls browsers via WebSockets, eliminating extra layers.

This direct interaction makes **Playwright tests faster & more stable**.
