# Playwright Interview Questions

Abhishesh Mishra

December 21, 2025

## 1. What is Playwright?

Playwright is a modern end-to-end automation framework developed by Microsoft to test web applications across multiple browsers using a single API.

**Key points:**

- Supports Chromium, Firefox, WebKit
- Works with JavaScript, TypeScript, Python, Java, C#
- Designed for fast, reliable, flaky-free automation
- Built-in auto-wait, network handling, parallel execution

**Code Example**

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

test('Open homepage', async ({ page }) => {
  await page.goto('https://example.com');
  await expect(page).toHaveTitle(/Example/);
});
```

**Interview tip:** Playwright tests the browser the same way real users interact — not through drivers.

## 2. Why use Playwright over Selenium?

Playwright solves many pain points of Selenium.

| Selenium | Playwright |
|---|---|
| Needs WebDriver | No WebDriver |
| Manual waits | Auto-waiting |
| Slower | Faster |
| Flaky tests | Stable tests |
| Limited network control | Full network control |

**Code Example**

```
// Auto-wait example
await page.click('#loginBtn'); // waits automatically
```

**Note:** In Selenium, you'd need explicit waits.

## 3. Which languages does Playwright support?

Playwright supports:

- JavaScript
- TypeScript (recommended)
- Python
- Java
- C# (.NET)

**Code Example**

```
// TypeScript Example (recommended)
import { test } from '@playwright/test';

test('TS example', async ({ page }) => {
  await page.goto('https://google.com');
});
```

**Why TypeScript?** Better autocomplete, Type safety, and fewer runtime errors.

## 4. Which browsers are supported by Playwright?

Playwright supports real browser engines:

- Chromium → Chrome, Edge
- Firefox
- WebKit → Safari (macOS/iOS behavior)

**Code Example**

```
test.use({ browserName: 'firefox' });
```

**Note:** One script = all browsers (cross-browser testing made easy).

## 5. What is Playwright Test Runner?

Playwright Test Runner is a built-in test framework that provides:

- Test execution, Assertions, Parallel runs
- Retries, HTML reports, Fixtures

**Code Example**

```
test('Login test', async ({ page }) => {
  await page.goto('/login');
});
```

**Note:** No need for Mocha/Jest separately.

## 6. How do you install Playwright?

**Step 1: Install**

**Code Example**

```
npm init playwright@latest
```

**Step 2: Run test**

**Code Example**

```
npx playwright test
```

**Install browsers**

**Code Example**

```
npx playwright install
```

**Note:** This installs browsers, config, and example tests automatically.

## 7. What is auto-waiting in Playwright?

Auto-waiting means Playwright automatically waits for:

- Element visibility
- Clickability
- Navigation
- Network idle

**Code Example**

```
await page.click('#submit'); // waits automatically
```

**Note:** No `sleep()`, no flaky waits.

## 8. What is a Locator?

A Locator is Playwright's smart way to find and interact with elements.
**Benefits:**

- Re-evaluated automatically
- Safer than static selectors
- Handles dynamic DOM

**Code Example**

```
const loginBtn = page.locator('#login');
await loginBtn.click();
```

**Note:** Locators are lazy — evaluated only when action is performed.

## 9. Difference between page.locator() and page.$()?

| page.locator() | page.$() |
|---|---|
| Auto-wait | No auto-wait |
| Retry mechanism | One-time fetch |
| Recommended | Not recommended |

**Code Example**

```
// Example (bad)
const btn = await page.$('#login');
await btn?.click();

// Example (good)
await page.locator('#login').click();
```

**Note:** Always use `locator()` in Playwright.

## 10. How to launch a browser in Playwright?

**Using Test Runner**

**Code Example**

```
test('Launch browser', async ({ page }) => {
  await page.goto('https://example.com');
});
```

**Manually (rare case)**

**Code Example**

```
const browser = await chromium.launch({ headless: false });
const page = await browser.newPage();
await page.goto('https://example.com');
```

**Note:** Test Runner handles browser lifecycle automatically.