🎯 Playwright Automation

📑 Table of Contents

1️⃣ Introduction to Playwright

Playwright is a **modern end-to-end automation framework** built for **speed, reliability, and scalability**. This guide is structured to reflect **how automation engineers work in real companies**, not how tutorials are written.

2️⃣ Why Playwright for Modern QA Teams

Playwright is widely adopted because it provides:

- Built-in **auto-waiting** (reduces flaky tests)
- **Single API** for Chromium, Firefox, and WebKit
- Native **debugging, tracing, screenshots, and videos**
- Seamless **CI/CD integration**

This makes Playwright ideal for **fast-moving agile teams**.

3️⃣ Environment Prerequisites

- Node.js **v18+**
- VS Code (recommended)
- Basic JavaScript knowledge

Verify installation:

```
node -v
npm -v
```

4️⃣ Playwright Installation (Step-by-Step)

npm init playwright@latest

Installation Choices (Interview-Relevant)

Option	Recommended	Reason
Language	JavaScript	Widely used in industry
Tests Folder	`tests/`	Standard Playwright convention
GitHub Actions	Yes	CI-ready setup
Browsers	Chromium, Firefox, WebKit	Cross-browser coverage

This command scaffolds a **production-grade automation framework**.

5️⃣ Updated Project Folder Structure (Industry-Ready)

```
playwright-project/
│
├── .github/
│   └── workflows/
│       └── playwright.yml   # CI execution
│
```

```
├── playwright-report/       # HTML execution reports
├── test-results/            # Screenshots, traces
│
├── tests/
│   ├── example.spec.js
│   └── spec_file.spec.js    # Core validation tests
│
├── .gitignore
├── package.json
├── package-lock.json
└── playwright.config.js     # Central configuration
```

📌 Recruiter Perspective

- Clean separation of concerns
- CI/CD ready
- Execution proof via reports

6️⃣ Test Lifecycle Hooks (`beforeEach`, `afterEach`)

Hooks ensure **test isolation**, a key automation quality metric.

```
const { expect, test } = require('@playwright/test');

// Runs before each test
test.beforeEach(async ({ page }) => {
  await page.goto('https://opensource-demo.orangehrmlive.com/web/index.php/auth/login');
});

// Runs after each test
test.afterEach(async ({ page }) => {
  await page.waitForTimeout(3000); // Debug/demo purpose only
});
```

Real-world usage:

- `beforeEach`: Navigation, login, test data setup
- `afterEach`: Cleanup, screenshots on failure

7️⃣ Code Explanation – Core Validations

Validate Page Title (TDD Style)

```
test('Validate Title', async ({ page }) => {
  const title = await page.title();
  console.log(`Title: ${title}`);
  await expect(title).toBe('OrangeHRM');
});
```

Validate Page URL

```
test('Validate URL', async ({ page }) => {
  const url = await page.url();
  console.log(`URL: ${url}`);
  await expect(url).toBe(
    'https://opensource-demo.orangehrmlive.com/web/index.php/auth/login'
  );
});
```

Validate UI Heading (XPath + Auto-Wait)

```
test('Validate Heading', async ({ page }) => {
  const text = await page.locator('//h5').innerText();
  console.log('Heading Text:', text);
  await expect(page.locator('//h5')).toHaveText('Login');
});
```

✔ Demonstrates **TDD-style assertions** with Playwright's auto-waiting.

8️⃣ Real-World Challenges & Solutions

❌ Flaky Tests

Cause: Hard waits, async UI rendering

✅ Solution:

- Use Playwright assertions (`toHaveText`, `toHaveURL`)
- Avoid `waitForTimeout()` in real test suites

❌ Dynamic Page Loads

await page.waitForLoadState('networkidle');

9️⃣ Benefits of Using Playwright & Inspector

await page.pause();

Key Advantages

- Step-by-step debugging
- Live DOM inspection
- Selector generation
- Faster root-cause analysis than Selenium

🔟 Test Execution Commands

Run All Tests

npx playwright test

Run Specific Spec File

npx playwright test --spec spec_file.spec.js

Debug Mode (Highly Valued in Interviews)

npx playwright test --spec spec_file.spec.js --browser=chromium --headed --debug

View HTML Report

npx playwright show-report

1️⃣1️⃣ Best Practices Recruiters Expect

✔ Assertion-driven validation
✔ Auto-waiting over sleeps
✔ Clean, scalable folder structure
✔ Debug-ready automation
✔ CI/CD compatible execution

🔚 Conclusion

This Playwright project reflects **real-world automation standards** by demonstrating:

- Industry-aligned setup
- Scalable folder structure
- Reliable UI validations
- Strong debugging capabilities
- Execution confidence across environments

✍️ Author

Routh Kiran Babu
Aspiring SDET | Playwright Automation | CI/CD Ready Frameworks

---