Playwright Documentation

Playwright is a Node.js library for end-to-end testing of web applications. It provides a reliable way to automate browsers like Chromium, Firefox, and WebKit, enabling cross-browser testing and debugging.

Getting Started

Installation

To use Playwright, you need Node.js installed on your system. Install Playwright via npm:

bash
Copy code
```
npm install playwright
```

For specific browser dependencies, run:

bash
Copy code
```
npx playwright install
```

Basic Usage

Here’s an example of a Playwright script that launches a browser, opens a page, and interacts with it:

javascript
Copy code
```
const { chromium } = require('playwright');

(async () => {
  const browser = await chromium.launch({ headless: false });
  const context = await browser.newContext();
  const page = await context.newPage();

  await page.goto('https://example.com');
  console.log(await page.title());
  await page.click('text=More information');
  await page.screenshot({ path: 'screenshot.png' });

  await browser.close();
})();
```

Key Concepts

1. Browser Types

Playwright supports three browser engines:

- **Chromium:** Used by Chrome and Edge.
- **WebKit:** Used by Safari.
- **Firefox:** Mozilla Firefox.

You can choose the browser as follows:

javascript
Copy code
```
const { chromium, firefox, webkit } = require('playwright');

// Example: Launching Firefox
const browser = await firefox.launch();
```

---

2. Contexts and Pages

- **Browser Contexts** are like incognito profiles, allowing isolated sessions.
- **Pages** are individual tabs in a browser.

javascript
Copy code
```
const context = await browser.newContext();
const page = await context.newPage();
```

---

3. Locators

Playwright provides flexible locators to find and interact with elements:

- **Text:** `page.locator('text=Sign In')`
- **CSS:** `page.locator('.button')`
- **XPath:** `page.locator('//button[text()="Submit"]')`

javascript
Copy code
```
await page.locator('text=Submit').click();
```

---

4. Actions

Playwright supports various actions on elements, including:

- **Clicking:** `await page.click(selector)`
- **Typing:** `await page.fill(selector, 'Text')`
- **Hovering:** `await page.hover(selector)`

---

5. Assertions

Use built-in assertions for testing:

javascript
Copy code
```
const { expect } = require('@playwright/test');
await expect(page).toHaveTitle(/Example Domain/);
```

Advanced Features

1. Headless Mode

Run browsers in headless mode (no UI) for faster execution:

javascript
Copy code
```
const browser = await chromium.launch({ headless: true });
```

---

2. Screenshots and Videos

Capture screenshots and record videos for debugging:

javascript
Copy code
```
await page.screenshot({ path: 'example.png' });
await context.tracing.start({ screenshots: true, snapshots: true });
```

---

3. Network Interception

Mock network requests and responses:

javascript
Copy code
```
await page.route('**/api/*', (route) =>
  route.fulfill({
    status: 200,
    body: JSON.stringify({ success: true }),
  })
);
```

---

4. Parallel Testing

Run tests concurrently using Test Runner:

bash
Copy code
```
npx playwright test --workers=4
```

---

5. Authentication

Save and reuse authentication state for quicker test execution:

javascript
Copy code
```
await context.storageState({ path: 'auth.json' });
```

Best Practices

1. Use **selectors** judiciously (prefer `data-testid` or text-based selectors).
2. Organize reusable logic into **Page Object Model (POM)**.

3. Run tests in **CI/CD pipelines** for continuous integration.
4. Debug efficiently using the **Playwright Inspector**:

bash
Copy code
```
DEBUG=pw:api npx playwright test
```

Playwright Test Framework

Playwright also includes a test runner with powerful features:

Installation

bash
Copy code
```
npm install @playwright/test
```

Example Test

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

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

Running Tests

Execute tests with:

bash
Copy code
```
npx playwright test
```

Debugging Tips

- **Pause Tests:** Add `await page.pause()` for interactive debugging.
- **Verbose Logs:** Use `DEBUG=pw:api` for detailed logs.

- **Trace Viewer:** Generate and inspect traces for failed tests:

bash
Copy code
```
npx playwright show-trace trace.zip
```