Playwright Testing & Debugging Cheatsheet

A comprehensive cheat sheet covering Playwright's testing and debugging features, including selectors, assertions, debugging techniques, and advanced configurations.

Selectors & Locators

Basic Selectors

`page.locator('text=Submit')`	Selects an element containing the text 'Submit'.
`page.locator('button')`	Selects all `<button>` elements.
`page.locator('#id')`	Selects an element with the ID 'id'.
`page.locator('.class')`	Selects all elements with the class 'class'.
`page.locator('input[name="name"]')`	Selects an `<input>` element with the name attribute 'name'.

Combining Selectors

`page.locator('div > button')`	Selects `<button>` elements that are direct children of `<div>` elements.
`page.locator('div.container button.primary')`	Selects `<button>` elements with class 'primary' inside `<div>` elements with class 'container'.
`page.locator('ul li:nth-child(2)')`	Selects the second `<li>` element inside a `<ul>` element.

Locator Methods

`.first()`	Selects the first matching element.
`.last()`	Selects the last matching element.
`.nth(index)`	Selects the element at the specified index.
`.filter({ hasText: 'text' })`	Filters elements to only include those containing the specified text.
`.locator(':scope > div')`	Find only immediate children `div` elements.

Assertions

Core Assertions

`expect(page).toHaveURL(url)`	Asserts that the page has the specified URL.
`expect(locator).toBeVisible()`	Asserts that the element is visible.
`expect(locator).toBeEnabled()`	Asserts that the element is enabled.
`expect(locator).toHaveText(text)`	Asserts that the element has the specified text.
`expect(locator).toHaveAttribute(name, value)`	Asserts that the element has the specified attribute and value.
`expect(locator).toHaveCount(count)`	Asserts that the locator resolves to the specified number of elements.

Advanced Assertions

`expect(page).toHaveTitle(title)`	Asserts that the page has the specified title.
`expect(locator).toHaveCSS(name, value)`	Asserts that the element has the specified CSS property and value.
`expect(locator).toHaveValue(value)`	Asserts that the element has the specified value.
`expect(locator).toContainText(text)`	Asserts that the element contains the specified text.
`expect(locator).not.toBeVisible()`	Asserts that the element is not visible (negative assertion).

Soft Assertions

Playwright does not have built-in soft assertions. You can implement your own using `try...catch` blocks or custom assertion libraries.

Example:

```
try {
  expect(locator).toBeVisible();
} catch (error) {
  console.warn('Assertion failed: Element not visible');
}
```

Debugging Techniques

Debugging in VS Code

1. Install the Playwright VS Code extension.
2. Set breakpoints in your test code.
3. Run your tests in debug mode using the extension's UI or the `DEBUG=pw:api` environment variable.

```
DEBUG=pw:api npx playwright test
```

Playwright Inspector

The Playwright Inspector is a GUI tool to help you debug your tests. It allows stepping through test execution, inspecting the DOM, and generating selectors.

Run tests in debug mode:

```
npx playwright test --debug
```

This opens the inspector, pauses execution at the first action, and allows stepping through the test.

Browser DevTools

You can use the browser's DevTools for debugging:

1. Set `headless: false` in your Playwright configuration.
2. Use `page.pause()` in your test to pause execution and open DevTools.

```
await page.pause();
```

Tracing

Playwright's tracing feature records detailed information about test executions, including network requests, console logs, and screenshots.

1. Configure tracing in `playwright.config.js`:

```
use: {
  trace: 'on-first-retry',
},
```

2. View the trace using `npx playwright show-trace trace.zip`.

Console Logging

Use `console.log()` statements in your test code to output debugging information to the console.

Example:

```
console.log('Current URL:', page.url());
```

Advanced Configuration

Test Configuration File

The `playwright.config.js` file configures Playwright's behavior. Key settings include `use`, `projects`, `reporter`, and `timeout`.

Example:

```
module.exports = {
  use: {
    baseURL: 'http://localhost:3000',
    headless: true,
    viewport: { width: 1280, height: 720
},
  },
  timeout: 30000,
};
```

Environment Variables

`BASE_URL=http://example.com npx playwright test`	Sets the base URL for the tests.
`HEADLESS=false npx playwright test`	Runs tests in headed mode (shows the browser).
`DEBUG=pw:api npx playwright test`	Enables debug logging for Playwright API calls.

Test Retries

Configure test retries in `playwright.config.js` to handle flaky tests.

```
module.exports = {
  retries: 2,
};
```

Timeouts

`timeout`	Sets the global timeout for tests in `playwright.config.js` (in milliseconds).
`expect(locator).toBeVisible({ timeout: 5000 })`	Sets a custom timeout for a specific assertion.
`page.goto(url, { timeout: 10000 })`	Sets a custom timeout for page navigation.