**Summary of Topics, Methods, and Mechanisms Learned**

**1. Playwright Fixtures and Browser Context**

* **Playwright Fixtures**:
  + **Fixtures** are like **global variables** that provide the necessary **browser, context, page**, or other utilities to your tests.
  + When using Playwright fixtures, it's essential to wrap them in **curly braces** ({ browser }, { context }, { page }).
  + Without curly braces, they will be treated as **strings or parameters**, not fixtures.
* **Example**:

javascript

Copy code

test('First Playwright Test', async ({ browser }) => {

// Using the browser fixture for testing

const context = await browser.newContext();

});

**2. Browser Context and Page Creation**

* **Browser Context**:
  + Created a new **browser context** using await browser.newContext(). This allows for **isolated browser sessions**, similar to **incognito** mode in browsers.
  + Each **browser context** has its own **cookies, storage**, and **session**. You can create contexts with specific properties, such as cookies or proxy settings.
* **Page Creation**:
  + Created a new **page** using await context.newPage(). Pages exist within a browser context, and multiple pages can be created in the same context.

**3. Using Cookies in Browser Context**

* **Adding Cookies** to a Browser Context:
  + Used context.addCookies(cookies) to add **pre-configured cookies** to the context.
  + This allows you to set cookies before navigating to the target URL, providing an easy way to **simulate authenticated sessions**.
* **Example**:

javascript

Copy code

const cookies = [

{ name: 'session\_id', value: 'abc123', domain: 'example.com', path: '/', httpOnly: true, secure: true, sameSite: 'Lax' }

];

await context.addCookies(cookies);

**4. Using Proxy in Browser Context**

* **Launching Browser with Proxy Settings**:
  + Configured the browser to use a **proxy** during its launch by passing a proxy parameter to chromium.launch().
  + This feature is especially useful for **testing geo-location-specific features** or using a **corporate proxy**.
* **Example**:

javascript

Copy code

const browser = await chromium.launch({

proxy: {

server: 'http://myproxyserver.com:3128',

username: 'myUsername',

password: 'myPassword'

}

});

**5. Using Page Fixture Directly**

* **Page Fixture**:
  + When there are no prerequisites like **cookies** or **proxy**, you can use the **page fixture** directly to **open a fresh incognito page**.
  + **Page fixture** makes it convenient to create a new page without worrying about context creation.
* **Example**:

javascript

Copy code

test.only('Page Title Validation', async ({ page }) => {

await page.goto('https://google.com');

let title = await page.title();

console.log("Title is - ", title);

await expect(page).toHaveTitle("Google");

});

**6. Assertions with expect()**

* **Assertions** help verify the expected outcome, making sure the test passes only if conditions are met.
* **Page Title Validation**:
  + Used await expect(page).toHaveTitle() to validate the **page title** after navigation.
  + This ensures that you are on the **correct page** with the expected title.
* **Direct Title Check**:
  + Used await expect(title).toBe('Google') to directly validate the **page title**.
* **Example**:

javascript

Copy code

await expect(page).toHaveTitle("Google");

**7. Wait Mechanisms in Playwright**

* **Auto-Waiting**:
  + Playwright has **auto-waiting** for most **actionable events** (like .click(), .fill(), .goto()). This ensures that the elements are **visible**, **enabled**, and **attached** to the DOM before interaction.
  + This feature allows you to avoid explicitly adding **wait statements** for most interactions, unlike Selenium.
* **Use of Explicit Waits**:
  + When elements are **loaded asynchronously**, you may need **explicit waits** like page.waitForLoadState('networkidle') to make sure all resources are loaded before proceeding.
* **Using waitFor() for Elements**:
  + Used productCards.waitFor() to ensure that **elements** are present before calling a method like .allTextContents().
  + This kind of waiting mechanism can be crucial for **getter methods** like .textContent() or .allTextContents(), where Playwright’s auto-waiting does not apply.

**8. Text Content Retrieval**

* **Single Element Text Retrieval**:
  + Used textContent() to extract the **text** of an individual element.
* **Multiple Elements Text Retrieval**:
  + Used allTextContents() to extract the **text content** from **multiple elements** as an array.
  + To avoid **flakiness**, ensured that the **page was stable** (networkidle) before calling allTextContents().

**9. Proxy and Cookie Management**

* **Context with Cookies**:
  + Adding cookies to a **browser context** simulates **logged-in sessions** and helps bypass login steps during tests.
* **Proxy Settings**:
  + Using a proxy with **browser context** is useful when your application has **regional restrictions** or needs to simulate requests from a specific location.

**Key Takeaways for an Interview:**

* **Fixtures in Playwright**:
  + Fixtures ({ browser }, { page }, etc.) are reusable, easy to manage, and provide **automatic setup** and **teardown**. Always wrap them in **curly braces** to ensure they are recognized as fixtures.
* **Browser Context and Its Use**:
  + Browser contexts allow **isolated sessions**, making them ideal for simulating multiple users or different environments without interference.
  + You can add **cookies** or **proxy settings** to a context, which makes testing authenticated or region-specific behaviors straightforward.
* **Handling Dynamic Content**:
  + Playwright has **auto-wait** for most **actions**, which is a key difference compared to Selenium, reducing the need for explicit waits.
  + However, when using getter methods (textContent(), allTextContents()), it may still be necessary to use **explicit waits** to avoid flaky results when dealing with **dynamic content**.
* **Title Assertions and Page Fixture Usage**:
  + Used assertions like expect(page).toHaveTitle() to ensure the correctness of page navigation.
  + The **page fixture** is useful when there's no requirement for cookies or proxies—allows the creation of a **clean, fresh page**.
* **Proxy and Cookies in Playwright**:
  + Added **cookies** to a browser context to maintain sessions, simulating scenarios where the user is already authenticated.
  + Configured a **proxy** during **browser launch** for testing applications behind **geolocation-based restrictions** or **corporate firewalls**.

This summary covers the methods, practices, and use cases you encountered in the script, providing a concise overview of key Playwright features, which will be valuable to discuss during interviews. It highlights your understanding of **fixtures**, **context management**, **wait mechanisms**, and **dynamic content handling**, all of which are crucial skills for a test automation role.