Advanced Playwright Interview Questions:

Select Dropdown

Q1: What is the basic method to select an option in a dropdown using Playwright?

APPROACH:

Introduce the simple commands used for interacting with dropdowns.

ANSWER:

In Playwright, the `selectOption()` method is used to interact with `<select>` elements. This method can select an option by its value, label, or index.

EXAMPLE:

```
await page.selectOption('select#yourDropdownId', { value: 'optionValue' });
```

Q2: How can you handle a dropdown that loads options dynamically in Playwright?

APPROACH:

Discuss strategies for interacting with asynchronously loaded dropdown options.

ANSWER:

Use the `waitForSelector()` method to ensure that the options are loaded and visible before attempting to select them.

EXAMPLE:

```
await page.waitForSelector('option[value="dynamicOption"]');
await page.selectOption('select#yourDropdownId', 'dynamicOption');
```

Q3: Can Playwright select multiple options from a dropdown? If yes, how?

APPROACH:

Explain the process of selecting multiple options from a multi-select dropdown.

ANSWER:

Playwright can select multiple options if the `<select>` element allows multiple selections. Pass an array of values or labels to `selectOption()`.

EXAMPLE:

```
await page.selectOption('select#yourMultiSelectDropdown', ['option1', 'option2']);
```

Q4: What challenges might you face when automating dropdowns with Playwright and how can you overcome them?

APPROACH:

Discuss common issues like hidden options and dynamic content.

ANSWER:

Challenges include hidden or dynamically loaded options. Overcome these by using methods like `waitForSelector()` or `waitForFunction()` to ensure elements are ready for interaction.

EXAMPLE:

```
await page.waitForSelector(
  'select#yourDropdownId option[value="optionValue"]',
  { state: 'attached' }
);
```

Q5: How does Playwright handle dropdowns within iframes?

APPROACH:

Describe accessing and interacting with elements within iframes.

ANSWER:

Use the `frameLocator()` method to target the specific iframe, then proceed with the `selectOption()` on the desired dropdown.

EXAMPLE:

```
const frame = await page.frameLocator('iframe');
await frame.selectOption('select#dropdownInsideIframe', 'optionValue');
```

Auto Waiting

Q6: What is auto-waiting in Playwright and how does it benefit test automation?

APPROACH:

Define auto-waiting and its advantages in automated testing scenarios.

ANSWER:

Auto-waiting in Playwright automatically manages waits, ensuring that elements are ready before performing actions. This reduces flaky tests and improves reliability.

Q7: How can you customize the wait conditions for a specific element in Playwright?

APPROACH:

Discuss methods to wait for elements to be in a certain state.

ANSWER:

Customize wait conditions using options in `waitForSelector()` like `visible`, `hidden`, or specifying a timeout.

EXAMPLE:

```
await page.waitForSelector('#buttonId', { state: 'visible' });
```

Q8: Describe a scenario where auto-waiting might fail and how you would handle it.

APPROACH:

Identify limitations of auto-waiting and propose solutions.

ANSWER:

Auto-waiting may fail in cases of extremely dynamic content. Use manual waiting strategies like `waitForTimeout()` or specific event listeners.

EXAMPLE:

```
await page.waitForTimeout(5000);
```

Q9: How does Playwright ensure that actions on elements are performed at the right time?

APPROACH:

Explain the synchronization mechanisms of Playwright.

ANSWER:

Playwright uses auto-waiting mechanisms to ensure that elements are interactable before performing actions, thus ensuring that tests are not flaky.

Q10: Can Playwright wait for network requests to complete before proceeding? How?

APPROACH:

Discuss network-related waiting mechanisms.

ANSWER:

Yes, use `waitForRequest()` or `waitForResponse()` to ensure that specific network calls are made or completed before proceeding.

EXAMPLE:

```
await page.waitForResponse(response =>
  response.url() === 'https://leaftaps.com/api' && response.status() === 200
);
```

Alert (Dialogs)

Q11: How can you handle browser alerts using Playwright?

APPROACH:

Introduce methods to interact with alerts in Playwright.

ANSWER:

Handle browser alerts by listening to the `dialog` event and responding with `accept()` or `dismiss()`.

EXAMPLE:

```
page.on('dialog', async dialog => {
  await dialog.accept();
});
```

Q12: What strategies are available in Playwright to verify the text of a dialog box?

APPROACH:

Explain how to access and check dialog properties.

ANSWER:

Capture the dialog instance using the `dialog` event, and use the `message()` method to verify its text.

EXAMPLE:

```
page.on('dialog', async dialog => {
  if (dialog.message() === 'expected message') {
    await dialog.accept();
  }
});
```

Q13: How can you automate testing a confirmation dialog in Playwright?

APPROACH:

Describe steps to interact with confirmation dialogs.

ANSWER:

Listen to the `dialog` event and choose to `accept()` or `dismiss()` based on the test scenario. Verify the outcome of the action.

EXAMPLE:

```
page.on('dialog', async dialog => {
  await dialog.dismiss();
});
```

Q14: What challenges might you face when automating dialog handling in Playwright and how can you overcome them?

APPROACH:

Discuss potential pitfalls like timing issues or missing dialogs.

ANSWER:

Challenges include dialogs that appear unexpectedly or fail to trigger events. Overcome these by preemptively setting up dialog handlers and using `try/catch` to manage exceptions.

EXAMPLE:

```
page.on('dialog', async dialog => {
  try {
    await dialog.accept();
  } catch (error) {
    console.error('Failed to handle dialog:', error);
  }
});
```

Q15: Can Playwright interact with non-native browser dialogs, such as custom JavaScript pop-ups?

APPROACH:

Describe handling complex custom dialogs.

ANSWER:

Yes, by targeting the elements within the dialog using standard selector methods. Playwright can interact with any element rendered in the DOM.

EXAMPLE:

```
await page.click('div.custom-popup > button.close');
```

Q16: How do you automate a scenario where selecting a dropdown option triggers an alert?

APPROACH:

Combine knowledge of dropdowns and alerts.

ANSWER:

Set up a dialog listener before interacting with the dropdown. Select the option and handle the triggered alert.

EXAMPLE:

```
page.on('dialog', async dialog => {
  await dialog.accept();
});
await page.selectOption('select#triggerAlertDropdown', 'alertOption');
```

Q17: Describe a complex scenario involving dropdowns, waiting, and alerts, and how you would automate it using Playwright.

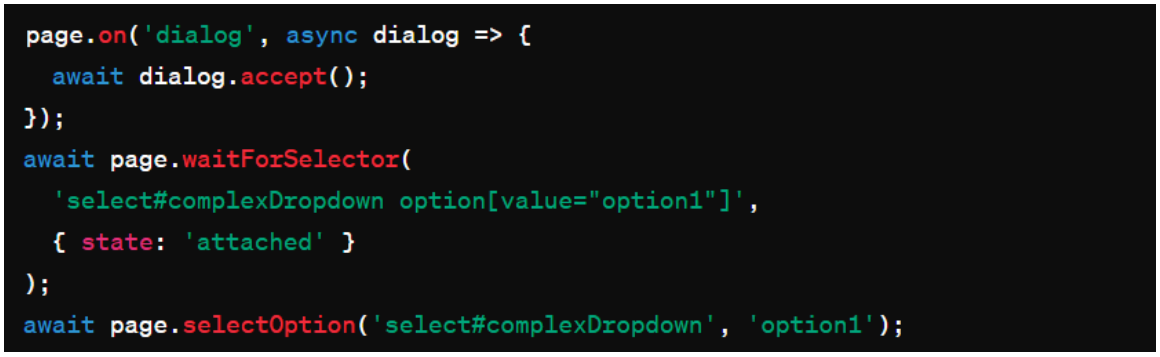
APPROACH:

Combine various techniques to handle a multi-faceted test scenario.

ANSWER:

For a test involving a dropdown that dynamically loads options, triggers an alert, and requires waiting for elements to appear: set up a dialog handler, use `waitForSelector` to ensure options are loaded, select an option, and handle the alert.

EXAMPLE:



```
page.on('dialog', async dialog => {
  await dialog.accept();
});
await page.waitForSelector(
  'select#complexDropdown option[value="option1"]',
  { state: 'attached' }
);
await page.selectOption('select#complexDropdown', 'option1');
```