Handling Multi-Select Dropdowns in Playwright

In Playwright, we can interact with dropdowns (select elements) to select multiple options. Below is a guide to handling **multi-select dropdowns**

→checking the number of options

→printing the dropdown values and

→validating the presence or absence of certain options.

Example DOM:

Here's an example of the HTML for a multi-select dropdown:

```
<select id="colors" class="form-control" multiple>
  <option value="red">RED</option>
  <option value="blue">BLUE</option>
  <option value="white">WHITE</option>
</select>
```

The `<select>` element has the `multiple` attribute, which allows selecting multiple options.

---

1. Select Multiple Options from the Dropdown

To select multiple options from a dropdown, use the `selectOptions` function. You need to pass an **array** of the options you want to select.

Example: Select Red and White Options

```
import { test, expect } from '@playwright/test';

test('Select multiple elements in dropdown', async ({ page }) => {
  await page.goto('https://example.com'); // Replace with the actual URL
  await page.locator('#colors').selectOptions(['red', 'white']); // Select
red and white options
});
```

---

2. Check Number of Options in the Dropdown

To check the number of options present in a dropdown, locate all the `<option>` elements and use `toHaveCount` assertion.

Example: Check Total Options in the Dropdown

```
import { test, expect } from '@playwright/test';

test('Check number of options in the dropdown', async ({ page }) => {
  await page.goto('https://example.com'); // Replace with the actual URL
  const options = await page.locator('#colors option');
  await expect(options).toHaveCount(3); // Assert that there are 3 options
});
```

---

3. Check Number of Options Using JavaScript Array

You can also use JavaScript to handle the options as an array and check the length using the `length` property.

Example: Check Total Options Using JavaScript Array

```
import { test, expect } from '@playwright/test';

test('Check number of options using JS array', async ({ page }) => {
  await page.goto('https://example.com'); // Replace with the actual URL
  const options = await page.$$('#colors option'); // Using $$
  console.log(options.length); // Log the number of options
  await expect(options.length).toBe(3); // Assert that there are 3 options
});
```

---

4. Print All Values in the Dropdown

To print all the values in a dropdown, you can use the `textContent` method to get the text for each option and then iterate over them.

Example: Print All Dropdown Values

```
import { test, expect } from '@playwright/test';

test('Print all values in dropdown', async ({ page }) => {
  await page.goto('https://example.com'); // Replace with the actual URL
  const optionsText = await page.locator('#colors').textContent();
  const optionList = optionsText.split('\n'); // Split the text content
into individual option values

  for (const optionText of optionList) {
    console.log(optionText); // Log each option's value
  }
});
```

---

5. Check if a Particular Value (e.g., BLUE) is Included in the Dropdown

To check if a particular value is present in the dropdown, you can use the `includes` method after extracting the text content.

Example: Check if "BLUE" is Included in the Dropdown

```
import { test, expect } from '@playwright/test';

test('Check if BLUE is present in the dropdown', async ({ page }) => {
  await page.goto('https://example.com'); // Replace with the actual URL
  const optionsText = await page.locator('#colors').textContent();
  await expect(optionsText.includes('BLUE')).toBeTruthy(); // Assert that
"BLUE" is included
});
```

---

6. Check if a Value (e.g., BLACK) is Not Included in the Dropdown

Similarly, you can check if a value is not present in the dropdown by using `includes` and asserting the result to be `false`.

Example: Check if "BLACK" is Not Included in the Dropdown

```
import { test, expect } from '@playwright/test';

test('Check if BLACK is not present in the dropdown', async ({ page }) => {
  await page.goto('https://example.com'); // Replace with the actual URL
  const optionsText = await page.locator('#colors').textContent();
  await expect(optionsText.includes('BLACK')).toBeFalsy(); // Assert that
"BLACK" is not included
});
```
