Locating Web Elements in Playwright

Whenever we perform any action on a webpage, we need to locate the element. This means identifying the element before performing any action on it.

Throughout web automation, we mainly perform two tasks:

1. **Locating the element on the webpage**
2. **Performing an action on that element**

The action depends on the type of element. Normally, we see elements like input fields, buttons, checkboxes, dropdowns, and links. Before performing any action on these web elements, we need to locate them.

In Playwright, below are the supported methods to locate web elements:

Locating Elements Using Different Methods

Sample DOM Structure

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Login Page</title>
</head>
<body>
    <div class="container">
        <form id="loginForm">
            <label for="username">Username:</label>
            <input type="text" id="username" class="formUser" name="user">

            <label for="password">Password:</label>
            <input type="password" id="password" name="pass">

            <button id="submitButton">Login</button>
        </form>
    </div>
</body>
</html>
```

---

1. Property-Based Locator

Definition: Properties are attributes of an element in the DOM that can be used for identification.
Example: Locating the input field using the **ID** property

```
await page.locator("id='username'").fill('testuser');
```

---

2. CSS Selector Locators

Definition: CSS selectors target elements using their **attributes, class, ID, or structure**.

Examples:

- **ID Selector:**

```
await page.locator('#username').fill('testuser');
```

- **Class Selector:**

```
await page.locator('.formUser').fill('testuser');
```

- **Attribute Selector:**

```
await page.locator("input[id='username']").fill('testuser');
```

- **Hierarchy Selector:**

```
await page.locator("div.container >
input[type='text']").fill('testuser');
```

---

3. XPath Locators

Definition: XPath is used to traverse the DOM and locate elements based on their **position and attributes**.

Examples:

- **Locating an input by ID:**

```
await page.locator("//input[@id='username']").fill('testuser');
```

- **Locating a button by text:**

```
await page.locator("//button[text()='Login']").click();
```

- **Locating an input inside a specific div:**

```
await
page.locator("//div[@class='container']//input").fill('testuser');
```

Importing Required Packages

Before writing tests, import the required packages from `@playwright/test` module. We can import them in two ways:

```
const { test, expect } = require('@playwright/test');
// OR
import { test, expect } from '@playwright/test';
```

Locating Elements Using Different Methods for practice

Locating a Button Using a Property

A property is an attribute from the DOM (key-value pair), such as:

```
<button id='loginButton'>Login</button>
```

We can locate this element using two methods:

```
// Using locator function
await page.locator('id=loginButton').click();

// Using direct click function
await page.click('id=loginButton');
```

Note: Always use the `await` keyword when interacting with elements using `page`.

Locating an Input Field Using CSS Selectors

DOM:

```
<input type='text' id='username' class='formUser'>
<input type='text' id='password' class='formUser'>
```

CSS Selectors:

- `#username` (for ID)
- `.formUser` (for class)
- `input[id='username']` (using tag + attribute)
- `input.formUser` (using tag + class)

Locating elements in Playwright:

```
// Using locator function
await page.locator('#username').fill('Yogesh');

// Using direct fill function
await page.fill("input[id='password']", 'Yogesh');
```

Locating a Submit Button Using XPath

DOM:

```
<button type='submit' id='submitButton'>Submit</button>
```

Using XPath:

```
await page.locator("//button[@id='submitButton']").click();
await page.click("//button[@id='submitButton']");
```

Verifying Logout Link Visibility

```
const logoutLink = await page.locator("//a[text()='Logout']");
await expect(logoutLink).toBeVisible();
```

Note: We use `await` with `expect` because `logoutLink` is a Playwright locator.

Full Example Code

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

test('Locators', async ({ page }) => {
    await page.goto('https://example.com');
    await page.click('id=loginButton');
    await page.fill('#username', 'Yogesh');
    await page.fill("input[id='password']", 'testLife@123');
    await page.click("button[id='submitButton']");

    const loginLink = await page.locator("a[text()='loginLink']");
    await expect(loginLink).toBeVisible();

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

Running the Test

```
npx playwright test LocatorsLearning.spec.js --project=chromium --headed
```

To see the Playwright default report:

```
npx playwright show-report
```

Locating Multiple Web Elements in Playwright

If we need to locate multiple elements, such as all links on a webpage, we use the `$$` function to get an array of elements.

Example: Locating all links

```
const links = await page.$$('a');
for (const link of links) {
    const linkText = await link.textContent();
    console.log(linkText);
}
```

Example: Locating all products

```
const productList = await page.$$("div[id='content'] //div//h4/a");
for (const product of productList) {
    const productText = await product.textContent();
    console.log(productText);
}
```

DOM

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Example Page</title>
</head>
<body>
    <header>
        <nav>
            <a href="https://example.com/home">Home</a>
            <a href="https://example.com/about">About</a>
            <a href="https://example.com/contact">Contact</a>
        </nav>
    </header>

    <div id="content">
        <div class="product">
            <h4><a href="/product1">Product 1</a></h4>
        </div>
        <div class="product">
            <h4><a href="/product2">Product 2</a></h4>
        </div>
        <div class="product">
            <h4><a href="/product3">Product 3</a></h4>
        </div>
    </div>
</body>
</html>
```

Full Code for Locating Multiple Elements

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

test('Locating multiple elements', async ({ page }) => {
    await page.goto("https://example.com");

    // Locate multiple links
    const links = await page.$$('a');
    for (const link of links) {
        const linkText = await link.textContent();
        console.log(linkText);
    }

    // Locate multiple products
    const productList = await page.$$("div[id='content'] //div//h4/a");
    for (const product of productList) {
        const productText = await product.textContent();
        console.log(productText);
    }
});
```

Handling Page Load Delays with `waitForSelector`

Sometimes, elements take time to load. Use `waitForSelector` before interacting with them:

```
await page.waitForSelector("//div[@id='content']//h4/a");
const productList = await page.$$("div[id='content'] //div//h4/a");
```

Full Code

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

test('Locating multiple elements', async ({ page }) => {
    // Navigate to the website
    await page.goto("https://example.com");

    // Wait for the content to load
    await page.waitForSelector("//div[@id='content']//h4/a");

    // Locate multiple links on the page
    const links = await page.$$('a');
    console.log("All links on the page:");
    for (const link of links) {
        const linkText = await link.textContent();
        console.log(linkText);
    }

    // Locate multiple product elements inside #content
    const productList = await page.$$("div[id='content'] //div//h4/a");
    console.log("Product List:");
    for (const product of productList) {
        const productText = await product.textContent();
        console.log(productText);
    }

    // Assertion example: Ensure at least one product is available
    expect(productList.length).toBeGreaterThan(0);
});
```

Comparison Between Selenium and Playwright

Feature	Selenium	Playwright
Explicit Wait	`WebDriverWait`	`waitForSelector()`
Locator Strategy	`By.id()`, `By.xpath()`	`page.locator()`
Multiple Elements	`findElements()`	`page.$$()`
