1. Mouse Hover – hover()

Purpose: To simulate a mouse hover over an element (useful for revealing dropdowns or tooltips).

Syntax:

```
await page.locator('selector').hover();
```

Example:

```
await page.locator('#menu').hover();
```

2. Right Click – click({ button: 'right' })

Purpose: To simulate a right-click on an element.

Syntax:

```
await page.locator('selector').click({ button: 'right' });
```

Example:

```
await page.locator('#file-icon').click({ button: 'right' });
```

3. Double Click – dblclick()

Purpose: To simulate a double-click on an element.

Syntax:

```
await page.locator('selector').dblclick();
```

Example:

```
await page.locator('#editable-text').dblclick();
```

4. Drag and Drop – dragTo()

Purpose: To drag an element and drop it to another target.

Syntax:

```
await page.locator('source-selector').dragTo(page.locator('target-selector'));
```

Example:

```
const source = page.locator('#drag-item');
const target = page.locator('#drop-zone');
await source.dragTo(target);
```

5. Mouse Down – page.mouse.down()

Purpose: To press and hold the mouse button down (without releasing it).

Syntax:

```
await page.mouse.down();
```

Example:

```
await page.mouse.move(100, 200);  // move to a position
await page.mouse.down();          // press mouse button
```

6. Mouse Up – page.mouse.up()

Purpose: To release the mouse button (after a mouse down).

Syntax:

```
await page.mouse.up();
```

Example:

```
await page.mouse.up();  // usually follows a mouse.down()
```

7. Mouse Move – page.mouse.move(x, y)

Purpose: To move the mouse to a specific position on the screen.

Syntax:

```
await page.mouse.move(x, y);
```

Example:

```
await page.mouse.move(150, 300);
```

Summary Table:

Action	Method/Command	Use Case
Hover	hover()	Show tooltips/menus
Right Click	click({ button: 'right' })	Context menu
Double Click	dblclick()	Edit input fields
Drag & Drop	dragTo()	Move items on UI
Mouse Down	page.mouse.down()	Custom drag actions
Mouse Up	page.mouse.up()	Release drag
Mouse Move	page.mouse.move(x, y)	Move to specific position

Playwright Scrolling

Automatic Scrolling (Default Behavior):

Playwright is smart enough to **automatically scroll** elements into view before interacting with them. So in most cases, **you don't need to scroll manually**.

When Manual Scrolling Is Needed?

In some rare scenarios, like:

- Loading more items in an **infinite scroll list**
- Triggering lazy-loaded content (images, text)

you may need to **scroll manually** using JavaScript.

How to Manually Scroll in Playwright

To manually scroll, you can use the **page.evaluate()** function.
This allows you to **run JavaScript code directly in the browser page**.

Syntax: page.evaluate()

This function lets you **execute JavaScript in the browser context**, and optionally interact with the DOM.

Example 1: Scroll to the bottom of the page

```
await page.evaluate(() => {
   window.scrollTo(0, document.body.scrollHeight);
});
```

Example 2: Get the current page height (Total scrollable height of the page)

```
const currentHeight = await page.evaluate(() => {
   return document.body.scrollHeight;
});
```