allTextContents() & all() Methods

1. innerText() vs textContent()

- **innerText()**:
  - Returns the *visible text* of an element.
  - Ignores hidden elements.
  - Removes extra whitespace and line breaks.
- **textContent()**:
  - Returns *all text* including from hidden elements.
  - Retains whitespaces, tabs, and line breaks.
  - Often needs trimming for cleaner output.

🔸 *Use innerText() when you need clean, visible-only text. Use textContent() when hidden content or exact raw text is required.*

2. allInnerTexts() vs allTextContents()

- **allInnerTexts()**:
  - Returns an array of visible text strings (from all matched elements).
  - Automatically cleans up whitespace.
- **allTextContents()**:
  - Returns an array of raw text strings (including from hidden elements).
  - Keeps extra whitespace and line breaks.
  - Often followed by .map(text => text.trim()) for cleanup.

🔸 *allInnerTexts() is simpler for most test output logs, while allTextContents() gives complete raw content.*

3. all() Method

- **locator.all()**:
    - Converts a group of elements (Locator) into an **array of individual locators**.
    - Allows you to interact with each element (e.g., get inner text, click, etc.) using array-style access or loops.

🔸 *Helpful for performing operations on specific elements or iterating over all matched elements.*

Summary Table

Method	Description	Trims Output	Includes Hidden
innerText()	Visible, cleaned-up text of a single element	✅	❌
textContent()	Raw text including hidden parts and formatting	❌	✅
allInnerTexts()	Array of visible, cleaned-up texts	✅	❌
allTextContents()	Array of raw texts, includes hidden content	❌	✅
all()	Converts Locator to array of locators (for iteration)	N/A	N/A

Handling Static Web Table

https://testautomationpractice.blogspot.com/

BookName	Author	Subject	Price
Learn Selenium	Amit	Selenium	300
Learn Java	Mukesh	Java	500
Learn JS	Animesh	Javascript	300
Master In Selenium	Mukesh	Selenium	3000
Master In Java	Amod	JAVA	2000
Master In JS	Amit	Javascript	1000

Key Actions Performed:

1. **Locating the Table:**

   ```
   const table = page.locator("table[name='BookTable'] tbody");
   ```

   - Uses a CSS selector to locate the table body.

2. **Counting Rows:**

   ```
   const rows = table.locator("tr");
   await expect(rows).toHaveCount(7);
   ```

   - Counts all rows (including header).
   - Two approaches are used: assertion with toHaveCount() and retrieving count via rows.count().

3. **Counting Columns (Headers):**

   ```
   const columns = rows.locator("th");
   await expect(columns).toHaveCount(4);
   ```

   - Only the header row contains <th> elements, which represent column headers.

4. **Reading Data from a Specific Row:**

```
const secondRowCells = rows.nth(2).locator('td');
const secondRowTexts = await secondRowCells.allInnerTexts();
```

   - Reads all cells from the 3rd row (index 2), excluding the header row.

5. **Reading All Data (excluding header):**

```
for (let row of allRowData.slice(1)) {
  const cols = await row.locator('td').allInnerTexts();
  console.log(cols.join('\t'));
}
```

   - Iterates over each data row and prints all cell values.

6. **Filtering Rows Based on Cell Value (Author = Mukesh):**

```
if (author === 'Mukesh') {
  console.log(`${author} \t ${book}`);
  mukeshBooks.push(book);
}
```

   - Finds and prints all books written by Mukesh.

7. **Aggregating Column Data (Total Price):**

```
totalPrice += parseInt(price);
```

   - Parses and sums the price of all books.