Training Day 9

1st July 2025

TOPICS COVERED

- **innerText, textContent, and innerHTML**

These are used to access or modify the textual content or HTML structure of DOM elements.

1. innerText- Returns visible text as rendered in the browser. It respects CSS styling like display: none or visibility: hidden.

Example:

```
let element = document.getElementById("demo");

console.log(element.innerText);

element.innerText = "Updated with innerText";
```

2. textContent- Returns all text, including hidden content. It's faster and more suitable when you just need raw text.

Example:

```
let element = document.getElementById("demo");

console.log(element.textContent);

element.textContent = "Updated with textContent";
```

3. innerHTML- Returns or sets the HTML markup inside an element.

Example:

```
let element = document.getElementById("demo");

console.log(element.innerHTML);

element.innerHTML = "<strong>Bold Text</strong>";
```

- **getAttribute() and setAttribute()-** Used to read or modify the attributes of an HTML element.

1. getAttribute(attrName)- Retrieves the value of the specified attribute.

Example:

```
let link = document.querySelector("a");
```

```
console.log(link.getAttribute("href"));
```

2. setAttribute(attrName, value)- Sets or updates the value of an attribute

Example:

```
link.setAttribute("href", "https://example.com");
link.setAttribute("target", "_blank");
```

- **Adding Elements to the DOM**

You can create and append new elements dynamically using JavaScript.

createElement() – creates a new element.

appendChild() – adds it to the DOM (usually inside another element like body or a div).

Example:

```
let new = document.createElement("p");
new.textContent = "This is a new paragraph!";
document.body.appendChild(new);
```

- **Removing Elements from the DOM-** To delete an element from the document.

Example:

```
<ul id="myList">
  <li>Item 1</li>
  <li>Item 2</li>
</ul>
let list = document.getElementById("myList");
let firstItem = list.querySelector("li");
list.removeChild(firstItem);
```

- **DOM Events-** DOM events let you run JavaScript in response to user actions like clicks, mouseovers, etc.

Common Events:

click

mouseover

keydown

submit

Adding an Event Listener:

```
let btn = document.getElementById("clickMe");
btn.addEventListener("click", function() {
  alert("Button was clicked!");
});
```

addEventListener() allows attaching multiple event types and handlers to a single element.

TOOLS USED

Visual Studio Code (VS Code)

Chrome Browser (JavaScript Console)