Day 21 of Training

JavaScript: DOM Manipulation, Events, and Event Handling

Today's session was a crucial step into making our web pages truly interactive by introducing the **Document Object Model (DOM)** and how to manipulate it using JavaScript, along with handling user events. This marks the transition from static web pages to dynamic web applications.

Understanding the Document Object Model (DOM)

We started by understanding the DOM as a programming interface for web documents. It represents the page so that programs can change the document structure, style, and content. Essentially, it's a tree-like representation of all the elements on a web page, allowing JavaScript to "see" and "interact" with HTML elements.

Key concepts of the DOM:

- **Tree Structure:** The DOM organizes all elements of an HTML document in a hierarchical tree structure, where each HTML tag, text, and attribute is represented as a "node." The document object is the root node.
- **Nodes:** Everything in an HTML document is a node: element nodes (HTML tags), text nodes (text content), and attribute nodes (attributes like id or class).
- **Accessing Elements:** We learned various methods to select and access specific HTML elements from the DOM:
    - document.getElementById('idName'): Selects an element by its unique id attribute.
    - document.getElementsByClassName('className'): Selects all elements with a specific class name (returns an HTMLCollection).
    - document.getElementsByTagName('tagName'): Selects all elements with a given HTML tag name (e.g., 'p', 'div') (returns an HTMLCollection).
    - document.querySelector('cssSelector'): A versatile method that returns the *first* element that matches a specified CSS selector.

- document.querySelectorAll('cssSelector'): Returns *all* elements that match a specified CSS selector (returns a NodeList).

Manipulating the DOM

Once we can access elements, we can manipulate their properties and content:

- **Changing Text Content:**
  - element.innerText: Gets or sets the text content of an element, ignoring HTML tags.
  - element.innerHTML: Gets or sets the HTML content of an element, allowing us to inject or modify HTML structures.
- **Changing Attributes:**
  - element.getAttribute('attributeName'): Gets the value of a specified attribute.
  - element.setAttribute('attributeName', 'newValue'): Sets or changes the value of an attribute.
  - element.style.propertyName: Directly manipulates inline CSS styles of an element (e.g., element.style.color = 'red';).
  - element.classList.add('className'), element.classList.remove('className'), element.classList.toggle('className'): Methods for easily adding, removing, or toggling CSS classes on an element, enabling dynamic styling.