Daily Task 4

Multi-Page Setup

Goal: Simulate a real-world app with multiple sections/pages.

#Implement the following-

- Add another page: about.html (app purpose, instructions, author info)
- Link about.html from navbar
- Ensure consistent layout (same navbar and footer)
- Highlight active page in the nav

#Explore the following-

- How Javascript is used to manipulate web application.
- Framework
- Javascript frameworks

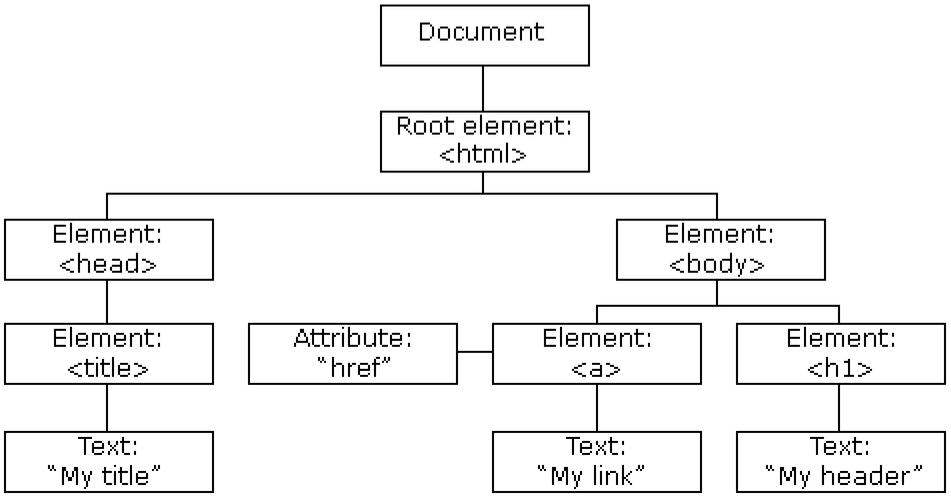
➢ **addEventListener() method :** This method attaches an event handler to the specified element. It attaches an event handler to an element without overwriting existing event handlers. You can add many event handlers to one element. You can add many event handlers of the same type to one element, i.e two "click" events. You can add event listeners to any DOM object not only HTML elements. i.e the window object.

Syntax: element.addEventListener(event, function, useCapture);

(use 'DOMContentLoaded' as Event to highlight the link)

DOM
(Document Object Model)

➢ The HTML DOM Tree of Objects :



➢ When an HTML document is loaded into a web browser, it becomes a **document object**. The **document object** is the root node of the HTML document.

The **document object** is accessed with:

window.document or just document

➢ To get an element :

1. **getElementById() :** Returns an element with a specified value. And returns null if the element does not exist. Ex –

```
const myElement = document.getElementById("idName");
myElement.style.color = "red";
```

2. **getElementsByClassName() :** returns a collection of elements with a specified class name(s). Ex –

document.getElementsByClassName(*classname*)[index];

3. **getElementsByTagName() :** returns a collection of all elements with a specified tag name. Ex –

   document.getElementsByTagName("tagName");

4. **querySelector() / querySelectorAll() :** Access elements with queries(CSS selector, class,id). Ex-

```
document.querySelector("p.example");

document.querySelector("#demo");

document.querySelector("div > p");
```

- **To get content of an HTML element**: **(innerHTML , innerText)**

  https://www.w3schools.com/jsrEF/tryit.asp?filename=tryjsref_node_textcontent_innerhtml_innertext

- **To create an Element :**

  const span = document.createElement('span');

  span.textContent = “taskText” ;   [textContent/ innerText / innerHTML]

  li.appendChild(span);

- **appendChild() :** It appends a node (element) as the last child of an element. Ex –

```
const node = document.createElement("li");
const textnode = document.createTextNode("Water");
node.appendChild(textnode);
document.getElementById("myList").appendChild(node);
```

  i. **The insertBefore() Method**
  ii. **The replaceChild() Method**

iii. The removeChild() Method
iv. The remove() Method
v. The childNodes Property
vi. The firstChild Property
vii. The lastChild Property
viii. The firstElementChild Property
ix. The lastElementChild Property

Daily Task - 5

DOM Manipulation & Event Handling

Goal : Learn how to interact with and dynamically update web page content using the Document Object Model (DOM) and handle user events using JavaScript.

#Implement the following-

1. **Create a new page:** todo.html
    - This page should simulate a basic to-do list app.
2. **Functionality to Implement using JavaScript:**
    - Add a task **[input + button]**
    - Display added tasks as a list. **[createElement, appendChild]**
    - Allow marking tasks as completed (e.g., strikethrough). **[addEventListener]**
    - Allow deleting individual tasks. **[addEventListener]**
    - Show a message if no tasks are present.