JS-6

Agenda

- Understanding document
- CRUD on our webpage
- working with user Interaction
- Todo App

Understanding document

Introduction

- **Document**: Represents the HTML page loaded in the browser.
- **Document Object Model (DOM)**: A tree-like structure that models the HTML document.

The DOM Tree Structure

The DOM is a hierarchical representation of the HTML document. Each element in the HTML becomes a node in the DOM tree. Nodes can have child nodes, creating a nested structure.

Example HTML Structure

Consider the following HTML code:

```
<div>
    <p>Hi</p>
    <div>
        <h2>Hi All</h2>
        <p>
            Lorem ipsum dolor sit amet consectetur adipisicing elit.
Corporis,
            impedit.
        </p>
```

```
        <ul>
            <li>lorem</li>
            <li>lorem</li>
        </ul>
    </div>
</div>
```

Visual Representation of the DOM Tree

```
<div>
  ├── <p>Hi</p>
  └── <div>
        ├── <h2>Hi All</h2>
        ├── <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
Corporis, impedit.</p>
        └── <ul>
              ├── <li>lorem</li>
              └── <li>lorem</li>
```

Notes for Learners: CRUD Operations with the DOM

Introduction

The DOM (Document Object Model) represents the structure of an HTML document as a tree of nodes. Understanding how to manipulate this structure is key to dynamic web development.

Example HTML Structure

```
<h1>Document Lecture</h1>
<p class="first">hello I am a p</p>
<p id="unique">I am another p tag</p>
<ul class="list"></ul>
<ul class="list"></ul>

<script src="1_crud_dom.js"></script>
```

CRUD Operations with the DOM

1. ## Select Nodes

   Nodes are elements in the DOM tree. To manipulate them, you first need to select them.

```
const idPtTag = document.querySelector("#unique");
```

   - `querySelector`: Selects the first matching element.
   - `querySelectorAll`: Selects all matching elements.

2. ## Text Content

   ### Reading Content:

```
const pTag = document.querySelector("p");
console.log("Content inside p:", pTag.textContent);
```

   ### Difference between `textContent` and `innerText`:

   - `textContent`: Returns all text, including hidden elements.
   - `innerText`: Returns only the visible text, considering CSS styling.

   ### Reading HTML Content:

```
const body = document.querySelector("body");
console.log("innerHTML:", body.innerHTML);
```

3. ## Updating Content

   ### Using `textContent`:

```
idPtTag.textContent = "I was updated by JS";
```

   ### Using `innerHTML`:

```
idPtTag.innerHTML = "<strong>Updated content</strong>";
```

   ### Updating Styles:

```
idPtTag.style.backgroundColor = "blue";
idPtTag.style.color = "white";
```

4. ### Adding Elements

Creating a New Element:

```
const liElem = document.createElement("li");
liElem.innerText = "I am a task";
```

Appending to a Parent Node:

```
const ulArr = document.querySelectorAll(".list");
ulArr[1].appendChild(liElem);
```

Adding Classes and Styling:

```
liElem.classList.add("task-item");
liElem.style.color = "lightblue";
```

5. ### Deleting Elements

Using `remove`:

```
idPtTag.remove();
```

Using `removeChild`:

```
const parent = document.querySelector("ul");
const child = parent.querySelector("li");
parent.removeChild(child);
```

Visual Representation of Append Child

Consider the following structure:

```
<ul class="list">
    <li>Existing task</li>
</ul>
```

After appending:

```
<ul class="list">
    <li>Existing task</li>
    <li>I am a task</li>
</ul>
```

Events and Event Listeners in JavaScript

Introduction

Events are actions that occur in the browser, such as clicks, hovers, key presses, and more. Event listeners allow you to run JavaScript code in response to these events.

Key Concepts

1. **Events**:
    - Actions that happen on the web page (e.g., mouse clicks, keyboard inputs, form submissions).
2. **Event Listeners**:
    - Functions that wait for a specified event to occur and execute code in response.

Example HTML Structure

```
<h1>Document Lecture</h1>
<p class="first">hello I am a p</p>
<p id="unique">I am another p tag</p>
<ul class="list">
    <!-- <li>I am task</li> -->
</ul>
<ul class="list"></ul>
```

```
<button>Click Me</button>
<div class="box" style="width:100px; height:100px; border:1px solid
black;"></div>

<script src="events.js"></script>
```

Example JavaScript Code

```
const button = document.querySelector("button");
const box = document.querySelector(".box");

const colors = ["lightblue", "lightgreen", "cyan", "gray", "red"];
let i = 0;

button.addEventListener("click", function () {
    i = i % colors.length;
    box.style.backgroundColor = colors[i];
    i++;
});
```

Detailed Explanation

1. **Selecting Elements**:
    - Use `querySelector` to select HTML elements that you want to attach event listeners to.

    ```
    const button = document.querySelector("button");
    const box = document.querySelector(".box");
    ```

2. **Events**:
    - **Click Event**: Triggered when the user clicks on an element.
    - **Hover Event**: Triggered when the user moves the mouse over an element.
3. **Adding Event Listeners**:
    - **addEventListener**: A method to attach an event handler to a specified event on a selected element.
    - Syntax: `element.addEventListener(event, function)`

4. **Example: Click Event**:
    - We change the background color of a box every time the button is clicked.

```
button.addEventListener("click", function () {
    i = i % colors.length;
    box.style.backgroundColor = colors[i];
    i++;
});
```

5. **Hover Event**:
    - Similar to the click event, you can also listen for hover events using `mouseover` and `mouseout`.

```
box.addEventListener("mouseover", function () {
    box.style.border = "2px solid blue";
});

box.addEventListener("mouseout", function () {
    box.style.border = "1px solid black";
});
```

Visual Representation

Consider the following visual representation of event listeners:

1. **Button Click**:
    - The user clicks the button.
    - The event listener attached to the button is triggered.
    - The background color of the box changes.
2. **Box Hover**:
    - The user hovers over the box.
    - The event listener attached to the box is triggered.
    - The border of the box changes.