## 🧩 Task 5: DOM Manipulation.

---

## ✅ Task 5: DOM Manipulation (Deep Dive)

### 🎯 Goals:

- Understand what the DOM is and how it's structured
- Learn how to select, read, and update HTML elements using JavaScript
- Create, append, and remove elements
- Modify classes and styles dynamically
- Build interactions like button clicks and form submissions

---

### 📘 What is the DOM?

**DOM (Document Object Model)** is a tree-like representation of your HTML document. JavaScript uses the DOM to:

- Access elements ( , <p>, etc.)
- Read/modify content
- Handle user interactions (like clicks or form inputs)

---

### 🔍 Key Concepts & Code Examples

---

#### 🔸 1. Selecting DOM Elements

**getElementById**

- Selects an element with a specific id.

```
const title = document.getElementById("main-title");
```

**querySelector / querySelectorAll**

- Selects the first element that matches a CSS selector (class, id, tag, etc.)

```
const firstBtn = document.querySelector(".btn");
```

```
const allButtons = document.querySelectorAll(".btn"); // returns NodeList
```

---

### 🔸 2. Reading & Changing Content

**.textContent vs .innerHTML**

```
element.textContent = "Hello";    // Only sets text
element.innerHTML = "<strong>Hi</strong>";  // Can include HTML
```

---

### 🔸 3. Modifying Attributes

```
const link = document.querySelector("a");
link.href = "https://example.com";
link.setAttribute("target", "_blank");
```

---

### 🔸 4. Changing Styles

```
const box = document.getElementById("box");
box.style.backgroundColor = "lightblue";
box.style.padding = "20px";
```

✅ Tip: Use classList to manage classes (preferred over .className)

```
box.classList.add("active");
box.classList.remove("hidden");
box.classList.toggle("dark-mode");
```

---

### 🔸 5. Creating, Appending, and Removing Elements

**Create an Element:**

```
const newItem = document.createElement("li");
newItem.textContent = "New Task";
```

**Append it to a parent:**

```
const list = document.querySelector("ul");
list.appendChild(newItem);
```

**Remove an Element:**

```
list.removeChild(newItem); // requires a reference
```

Or use:

```
newItem.remove(); // modern and cleaner
```

---

### 🔸 6. Event Handling

**addEventListener**

- Reacts to user actions like clicks, typing, submitting forms, etc.

```
const btn = document.querySelector("#submitBtn");


btn.addEventListener("click", function () {
  alert("Button clicked!");
});
```

**Common Events:**

| Event Type | Trigger |
|---|---|
| click | User clicks an element |
| input | User types in input |
| submit | Form is submitted |
| mouseover | Mouse hovers over element |

---

### 🔸 7. Form Input Handling

```
<form id="contactForm">
```

```
  <input id="username" />
  <button type="submit">Submit</button>
</form>
const form = document.getElementById("contactForm");

form.addEventListener("submit", function (e) {
  e.preventDefault(); // prevent reload
  const name = document.getElementById("username").value;
  console.log("Submitted:", name);
});
```

---

### 💉 Practice Exercises

### ✅ Exercise 1: Content Changer

- Create a paragraph and a button
- When the button is clicked, change the paragraph's text to "You clicked the button!"

---

### ✅ Exercise 2: Task List

- Create a list (<ul>) and an input field with a button
- When user types a task and clicks the button:
    - Add a new <li> to the list
    - Clear the input field

---

### ✅ Exercise 3: Toggle Theme

- Create a dark mode toggle button
- When clicked, use .classList.toggle('dark') on the body
- Add CSS for .dark { background-color: black; color: white; }

---

💬 **Reflection Questions**

1. What is the difference between textContent and innerHTML?
2. Why is querySelector preferred over older methods like getElementsByTagName?
3. What does e.preventDefault() do in form submissions?
4. What's the difference between appendChild() and innerHTML +=?