Understanding `addEventListener()` : Where and How ?

The `addEventListener()` method is **one of the most important** functions in JavaScript events. It allows you to attach event listeners to elements dynamically without overwriting existing event handlers.

📌 Where Should We Explain `addEventListener()` ?

✔ Option 1: In DOM Manipulation Section

- **Why?** Because it is used to attach event listeners to dynamically created elements.
- **Example:** When adding a button dynamically using `createElement()` , we need `addEventListener()` to make it interactive.

✔ Option 2: In JavaScript Events Section ✅ (Best Place)

- **Why?** Because it is the **best practice** for handling JavaScript events.
- **Example:** Instead of using `onclick="myFunction()"` in HTML, we use `addEventListener()` in JavaScript.

Since `addEventListener()` is **directly related to event handling**, we should explain it in the **JavaScript Events section**.

📌 What Does `addEventListener()` Do?

- It **attaches an event** (like `click` , `mouseover` , etc.) to an element.
- It **allows multiple event listeners** on the same element.
- It **does not overwrite** existing event handlers.
- It **supports event delegation** and **advanced event propagation**.

📌 How to Use `addEventListener()` ?

1️⃣ Syntax

```
element.addEventListener(event, function, useCapture);
```

- `event` → The event type (`click` , `mouseover` , `keydown` , etc.).
- `function` → The function to execute when the event occurs.
- `useCapture` *(optional)* → Boolean (default `false`). Used for **event propagation** (bubbling or capturing).

📌 2. `addEventListener()` vs Inline Event Handling (`onclick`)

🚫 Bad Practice: Using Inline Event in HTML

```
<button onclick="showMessage()">Click Me</button>
<script>
    function showMessage() {
        alert("Button Clicked!");
    }
</script>
```

❌ Problem:

- Not reusable.
- Can't attach multiple event listeners.

✅ Best Practice: Using `addEventListener()`

```
<button id="myButton">Click Me</button>
<script>
    document.getElementById("myButton").addEventListener("click", function()
{
        alert("Button Clicked with addEventListener!");
    });
</script>
```

✔ Advantages:

- Separation of **HTML & JavaScript** (cleaner code).

- Can **add multiple event listeners**.
- Works with **dynamically created elements**.

📌 3. `addEventListener()` in Real-World Projects

🛠️ Real-World Project Example:

- **Dark Mode Toggle in a Website**

👨‍💻 HTML + JavaScript Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Dark Mode Toggle</title>
    <style>
        body.dark-mode {
            background-color: black;
            color: white;
        }
    </style>
</head>
<body>

    <button id="darkModeToggle">Toggle Dark Mode</button>

    <script>
        document.getElementById("darkModeToggle").addEventListener("click", 
function() {
            document.body.classList.toggle("dark-mode");
        });
    </script>

</body>
</html>
```

✔ Why Use `addEventListener()` ?

- We can **attach multiple events** if needed.
- Works on **multiple elements dynamically**.

📌 4. Adding Multiple Event Listeners

Example: Logging Button Clicks and Changing Text

```
<button id="multiEventBtn">Click Me</button>
<script>
    let button = document.getElementById("multiEventBtn");

    // First event: Log click
    button.addEventListener("click", function() {
        console.log("Button clicked!");
    });

    // Second event: Change text
    button.addEventListener("click", function() {
        button.innerText = "Clicked!";
    });
</script>
```

✔ Why Use `addEventListener()` ?

- Unlike `onclick`, this method **allows multiple event listeners**.

📌 5. Removing Event Listeners (`removeEventListener`)

To **remove an event listener**, you need to store the function in a variable.

```
<button id="removeEventBtn">Click Me</button>
<script>
    let button = document.getElementById("removeEventBtn");

    function sayHello() {
        alert("Hello!");
    }

    // Attach event
    button.addEventListener("click", sayHello);

    // Remove event after 5 seconds
    setTimeout(() => {
        button.removeEventListener("click", sayHello);
    }, 5000);
</script>
```

✔ **Why Use `removeEventListener()`?**

- **Prevents memory leaks** in large applications.
- **Stops unwanted behavior** after a certain condition.

🎯 Conclusion

Now you fully understand **why `addEventListener()` is important** and how to use it effectively!

💡 Key Takeaways:

✔ **Use `addEventListener()`** instead of `onclick` for flexibility.
✔ **Attach multiple events** to the same element.
✔ **Works with dynamically created elements**.
✔ **Can be removed with `removeEventListener()`** when necessary.
