Intern Name: Fatima Wajid

Submitted to: Ali Hyder

Date: 21st July 2025

Internship Role: Web Development

Objective of the Day

To understand how **JavaScript handles events** and how different event-handling techniques (onclick, onchange, and addEventListener) can be used to make web pages interactive and responsive to user actions.

What Are JavaScript Events?

JavaScript events are actions or occurrences that happen in the browser, which can be detected and responded to using JavaScript. Common events include:

- Clicking a button
- Changing input in a text field
- Submitting a form
- Hovering over an element
- Loading a webpage

1. onclick Event

The onclick event is triggered when a user clicks on an HTML element. It can be defined **inline in HTML** or handled with JavaScript.

Example (Inline HTML):

```
<button onclick="alert('Button Clicked!')">Click Me</button>
```

Example (JavaScript):

```
<button id="btn">Click Me</button>

<script>
  document.getElementById("btn").onclick = function () {
    alert("Button was clicked!");
  };
</script>
```

Use Cases:

- Submitting forms
- Triggering animations
- Opening modals or pop-ups

2. onchange Event

The onchange event is triggered when the value of an input field, select box, or textarea changes **after focus is lost**.

Example:

```
<select id="dropdown">
  <option value="apple">Apple</option>
```

```
  <option value="banana">Banana</option>
</select>

<script>
  document.getElementById("dropdown").onchange = function () {
    alert("You selected: " + this.value);
  };
</script>
```

Use Cases:

- Form validations
- Updating data based on user input
- Auto-calculating values in forms

3. addEventListener() Method

This is the **most recommended** and flexible way to handle events in JavaScript. It allows attaching **multiple event handlers** to the same event without overwriting existing ones.

Syntax:

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

Example:

```
<button id="testBtn">Try Me</button>

<script>
  const btn = document.getElementById("testBtn");
  btn.addEventListener("click", function () {
    alert("You clicked the button!");
  });
```

</script>

Advanced Usage:

```
function handleClick() {
  console.log("Event Triggered!");
}

btn.addEventListener("click", handleClick);
btn.removeEventListener("click", handleClick); // Removes the event
```

Benefits:

- Separation of HTML and JavaScript code
- Ability to attach multiple listeners
- Better scalability in large applications

Comparison Table

Feature	onclick	onchange	addEventListener()
HTML Inline Support	✅ Yes	✅ Yes	❌ No
JavaScript Support	✅ Yes	✅ Yes	✅ Yes
Multiple Handlers	❌ No	❌ No	✅ Yes
Best Practice	❌ Not Recommended	❌ Not Recommended	✅ Recommended

What I Learned Today

- JavaScript events allow interaction with web pages.

- onclick and onchange are useful but limited in flexibility.
- addEventListener() is preferred for clean, scalable code.
- I practiced attaching event handlers and learned how to trigger alerts, update DOM elements, and respond to form inputs dynamically.

Conclusion

Today's session was crucial for mastering the **core concept of interactivity in JavaScript**. Understanding event handling prepares me for more complex front-end applications, like form validation, animations, and dynamic content manipulation. I now feel more confident in using JavaScript to respond to user actions effectively.