📌 JavaScript Event Handling: Inline vs. Scripted Approach

When handling events like **button clicks**, **form submissions**, or **mouse movements**, there are **two ways** to do it:
1️⃣ **Inline event handling** (older, less preferred)
2️⃣ **Scripted event handling** (better, modern approach)

📌 1️⃣ Inline Event Handling (Not Recommended)

Inline event handlers are directly added to the **HTML element**.

🔹 Example: Button Click (Inline Event Handling)

```
<input type="button" value="Click" onClick="sayHello();">
<script>
function sayHello() {
   alert("Hi there!");
}
</script>
```

❌ Why Avoid It?

- **Mixes JavaScript with HTML** (harder to manage and debug).
- **Not reusable** (each element needs its own onClick).

📌 2️⃣ Scripted Event Handling (Recommended)

Instead of writing event handlers inside HTML, we assign them **via JavaScript**.

🔹 Example: Button Click (Scripted Event Handling)

```
<input type="button" value="Click" id="button1">
<script>
var b1 = document.getElementById("button1");
b1.onclick = sayHello;

function sayHello() {
   alert("Hi there!");
}
</script>
```

✓ Why Use This Method?
✅ **Keeps JavaScript separate from HTML** (cleaner code).
✅ **More flexible** (can attach multiple events dynamically).
✅ **Easier debugging** (errors are in JavaScript, not in HTML).

📌 Event Handling Rules

🔹 1. Use lowercase event names in JavaScript (not camelCase).

✅ onclick, onmouseover, onsubmit
❌ onClick, onMouseOver, onSubmit

🔹 2. Do not use parentheses when assigning a function.

✅ b1.onclick = sayHello;
❌ b1.onclick = sayHello(); (This will call the function immediately instead of waiting for the event.)

📌 Examples of Event Handling

◆ 1. Changing Image on Mouse Hover

```
<img id="myImage" src="image1.jpg" width="200">
<script>
var img = document.getElementById("myImage");

img.onmouseover = function() {
   img.src = "image2.jpg"; // Change image on hover
};

img.onmouseout = function() {
   img.src = "image1.jpg"; // Change back on mouse out
};
</script>
```

◆ 2. Validating an Email on Form Submission

```
<form id="myForm">
   <input type="email" id="email" placeholder="Enter email">
   <input type="submit" value="Submit">
</form>
```

```
<script>
var form = document.getElementById("myForm");

form.onsubmit = function(event) {
    var email = document.getElementById("email").value;

    if (!email.includes("@")) {
        alert("Invalid email address!");
        event.preventDefault(); // Prevents form submission if invalid
    }
};
</script>
```

✓ How it Works:

1️⃣ If the email doesn't contain "@", it alerts **"Invalid email address!"**
2️⃣ event.preventDefault(); stops the form from being submitted.

📌 Best Practice: Using addEventListener()

For **modern JavaScript**, use addEventListener() instead of onclick, onmouseover, etc.

🔹 Example: Button Click (Using addEventListener)

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

✓ Advantages of addEventListener()

✅ Can attach **multiple events** to the same element.
✅ Can be **removed later** using removeEventListener().

📌 Summary

Method	Pros	Cons
Inline event handling	Simple, quick	Messy, hard to manage
Scripted event handling (element.onclick = function)	Clean, reusable	Only supports one event per type

Method	Pros	Cons
addEventListener() (Recommended)	Best practice, supports multiple events	Slightly longer code

---