

Session 5

JavaScript HTML DOM Events and Event Listener

In Session 4, you learned how to access and manipulate HTML elements using the DOM. Session 5 builds upon this foundation by introducing DOM Events, which enable web pages to react dynamically to user actions and browser occurrences. Events are the bridge between static HTML and interactive JavaScript, allowing scripts to execute in response to triggers like mouse clicks, page loads, keyboard input, or form submissions.

This session shifts focus from inline event attributes (e.g., `onClick="func()"`) to the modern, standardized Event Listener API. You will learn how to attach, manage, and remove event handlers using `addEventListener()` and `removeEventListener()`, understand event propagation (bubbling vs. capturing), and implement multiple independent listeners on a single element. By the end of this session, you will be able to design responsive, event-driven web interfaces while maintaining clean separation between HTML structure and JavaScript behavior.

Objectives

✖ Do not copy. Read for understanding and the viva

- Understand common HTML DOM events and their triggers
- Attach event handlers using `addEventListener()` without overwriting existing ones
- Control event propagation using the `useCapture` parameter (bubbling vs. capturing)
- Remove event listeners dynamically using `removeEventListener()`
- Implement practical event-driven interactions through hands-on exercises

DOM Events

DOM events are signals that something has occurred in the browser or with a specific HTML element. JavaScript can listen for these signals and execute code when they fire.

Event Category	Common Examples	Trigger
Mouse	<code>click</code> , <code>dblclick</code> , <code>mouseover</code> , <code>mouseout</code> , <code>mousedown</code> , <code>mouseup</code>	User interacts with pointing device
Keyboard	<code>keydown</code> , <code>keyup</code> , <code>keypress</code>	User presses/releases keys
Form/Input	<code>change</code> , <code>input</code> , <code>submit</code> , <code>focus</code> , <code>blur</code>	User modifies or submits form data
Window/Document	<code>load</code> , <code>DOMContentLoaded</code> , <code>resize</code> , <code>scroll</code>	Page lifecycle or viewport changes
Media/Image	<code>load</code> (on <code></code>), <code>play</code> , <code>pause</code>	Resource state changes

Event Handling Evolution

Approach	Syntax	Pros	Cons
Inline HTML	<code><button onclick="myFunc()"></code>	Quick to write	Mixes HTML & JS, hard to maintain, overwrites if duplicated
DOM Property	<code>btn.onclick = myFunc;</code>	Simple separation	Only allows one handler per event type (overwrites previous)
Event Listener	<code>btn.addEventListener("click", myFunc);</code>	Recommended: supports multiple handlers, clean separation, modern standard	Slightly more verbose

Syntax & Parameters

addEventListener()

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

JS

Parameter	Description
<code>event</code>	String name of the event (e.g., <code>"click"</code> , <code>"load"</code> , <code>"change"</code>)
<code>function</code>	Callback to execute when event fires (named function or anonymous)
<code>useCapture</code>	Optional Boolean. <code>false</code> = Bubbling (default), <code>true</code> = Capturing

Key Advantage: Unlike `.onclick =`, `addEventListener` does not overwrite existing handlers. You can attach multiple listeners to the same element for the same event.

```
1 // Multiple listeners on one button
2 btn.addEventListener("click", firstAction);
3 btn.addEventListener("click", secondAction); // Both will run!
```

JS

removeEventListener()

Dynamically detaches a previously attached listener. Crucial: The function reference passed to `removeEventListener` must be the exact same function used in `addEventListener`. Anonymous functions cannot be removed directly.

```
1 // ✅ Correct: Named function
2 function showAlert() { alert("Hello"); }
3 btn.addEventListener("click", showAlert);
4 btn.removeEventListener("click", showAlert);
5
6 // ❌ Incorrect: Anonymous functions are unique instances
7 btn.addEventListener("click", function() { alert("Hello"); });
8 // btn.removeEventListener("click", function() { alert("Hello"); }); // Won't work!
```

JS

Event Propagation

When nested elements have event listeners, the browser must decide the execution order. This is called **event propagation**.

Mode	Order of Execution	useCapture	Value
Bubbling (Default)	Innermost element → Outermost parent	false	
Capturing	Outermost parent → Innermost element	true	

Example

✖ Do not copy. Read for understanding and the viva

```
1 <div id="outer">
2   <p id="inner">Click me</p>
3 </div>
```

HTML

- If `<p>` is clicked with bubbling: `<p>` handler runs first, then `<div>` handler.
- If `<p>` is clicked with capturing: `<div>` handler runs first, then `<p>` handler.

Best Practices & Common Pitfalls

✖ Do not copy. Read for understanding and the viva

Pitfall	Best Practice
Using inline <code>onClick</code> in modern projects	Use <code>addEventListener()</code> for separation of concerns
Attaching listeners inside loops without caching	Cache elements: <code>const btn = document.getElementById("x")</code> ; then attach once
Forgetting event type strings must be lowercase	Use <code>"click"</code> not <code>"Click"</code> or <code>"onClick"</code>
Passing anonymous functions to <code>removeEventListener</code>	Always use named functions or store references if removal is needed
Ignoring <code>preventDefault()</code> on form/button clicks	Use <code>event.preventDefault()</code> to stop default browser actions when needed

Quick Guide

✖ Do not copy. Read for understanding and the viva

```
1 // 📌 Add Listener
2 el.addEventListener("click", myFunction);
3
4 // 🗑 Remove Listener
5 el.removeEventListener("click", myFunction);
6
7 // 🛑 Control Propagation
8 el.addEventListener("click", myFunction, false); // Bubbling (default)
9 el.addEventListener("click", myFunction, true); // Capturing
10
11 // 📦 Anonymous function with parameters
12 el.addEventListener("click", function() { myCustomFunc(a, b); });
13
14 // 🛑 Stop default behavior (e.g., form submission, link navigation)
15 el.addEventListener("submit", function(e) { e.preventDefault(); });
```

JS

💡 Suggestion

See Examples in Lab manual. It covers all the main methods mentioned above.

⚠ Alert

If you see a alert message on screen after a scroll to question 2 please ignore that. It's part of it.

Question 1

Problem Statement

■ Write in lab record

Design an HTML page as shown in figure(refer to lab manual). Implement the event "on button click display data" using HTML DOM events.

Before click on submit button:

Figure 1: *Text with button but no time text*

After click on submit button:

Figure 2: *Show Date & Time below submit button*

Preview (Working)

✖ Do not copy. Read for understanding and the viva

Exercise 1: Display Date on Click

Show Date

Click the button to display the date

Code

■ Write in lab record

date.html

```
1 <!doctype html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <title>Ex1: Display Date</title>
6   </head>
7   <body style="font-family: sans-serif; padding: 20px">
8     <h2>Exercise 1: Display Date on Click</h2>
9     <button id="showDateBtn">Show Date</button>
10    <p id="dateOutput" style="margin-top: 10px; font-weight: bold"></p>
11
12    <script>
13      // Attach event listener using addEventListener (Session 5 concept)
14      document
15        .getElementById("showDateBtn")
16        .addEventListener("click", () => {
17          const today = new Date().toLocaleDateString();
18          document.getElementById("dateOutput").textContent =
19            `Today's Date: ${today}`;
20        });
21    </script>
22  </body>
23 </html>
```

Question 2

Problem Statement

■ Write in lab record

Design an HTML page to shown that, when this page is loaded it must check the cookies status of the user's browser and display a message accordingly in the popup box i.e if cookies are enabled show: "Cookies are enabled" and if not then show: "Cookies are disabled". Implement the event "on load" using HTML DOM events

Preview (Working)

✕ Do not copy. Read for understanding and the viva

Exercise 2: Cookie Status

Initializing...

Alert shown when scrolled to this section, Only once.

Code

■ Write in lab record

color-change.html

```
1 <!doctype html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <title>Ex2: Cookie Status Check</title>
6   </head>
7   <body style="font-family: sans-serif; padding: 20px">
8     <h2>Exercise 2: Cookie Status on Load</h2>
9     <p>Reload the page to trigger the load event.</p>
10
11     <script>
12       // Using window.onload as specified in the lab
13       window.addEventListener("load", () => {
14         if (navigator.cookieEnabled) {
15           alert("Cookies are enabled");
16         } else {
17           alert("Cookies are disabled");
18         }
19       });
20     </script>
21   </body>
22 </html>
```

Question 3

Problem Statement

■ Write in lab record

Whenever user input a text in the text box in any case (i.e. upper or lower case), automatically it should be converted to uppercase. Design an HTML page to implement above functionality using HTML DOM event “onchange”.

Preview (Working)

✖ Do not copy. Read for understanding and the viva

Exercise 3: Auto Uppercase

Converted: Waiting for input...

Code

■ Write in lab record

onchange.html

```
1 <!doctype html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <title>Ex3: Auto Uppercase</title>
6   </head>
7   <body style="font-family: sans-serif; padding: 20px">
8     <h2>Exercise 3: Auto Uppercase on Change</h2>
9     <input
10       type="text"
11       id="textInput"
12       placeholder="Type something (press Enter or click outside)"
13       style="padding: 8px; width: 300px"
14     />
15     <p id="result" style="margin-top: 10px; font-weight: bold"></p>
16
17     <script>
18       document
19         .getElementById("textInput")
20         .addEventListener("change", function () {
21           // Convert to uppercase and update value
22           this.value = this.value.toUpperCase();
23           document.getElementById("result").textContent =
24             `Converted: ${this.value}`;
25         });
26     </script>
27   </body>
28 </html>
```

Question 4

Problem Statement

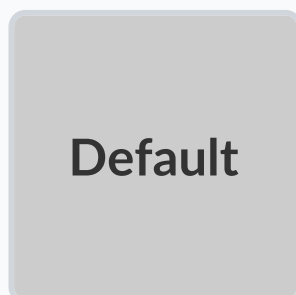
■ Write in lab record

Design HTML page to implement HTML DOM events by changing an image on the following events: onmouseover, onmouseout, onmousedown, and onmouseup Events.

Preview (Working)

✖ Do not copy. Read for understanding and the viva

Exercise 4: Image Mouse Events



Hover or click the image

Code

[Write in lab record](#)

image-mouse-event.html

```
1 <!doctype html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <title>Ex4: Mouse Events on Image</title>
6   </head>
7   <body style="font-family: sans-serif; padding: 20px; text-align: center">
8     <h2>Exercise 4: Image Mouse Events</h2>
9     
15     <p id="eventLog" style="font-style: italic; color: #555">
16       Hover or click the image
17     </p>
18
19     <script>
20       const img = document.getElementById("dynamicImg");
21       const log = document.getElementById("eventLog");
22
23       img.addEventListener("mouseover", () => {
24         img.src =
25           "https://placeholder.co/200x200/3498db/fff?text=Mouse+Over";
26         log.textContent = "Event: mouseover";
27       });
28       img.addEventListener("mouseout", () => {
29         img.src =
30           "https://placeholder.co/200x200/95a5a6/fff?text=Mouse+Out";
31         log.textContent = "Event: mouseout";
32       });
33       img.addEventListener("mousedown", () => {
34         img.src =
35           "https://placeholder.co/200x200/e74c3c/fff?text=Mouse+Down";
36         log.textContent = "Event: mousedown";
37       });
38       img.addEventListener("mouseup", () => {
39         img.src =
40           "https://placeholder.co/200x200/2ecc71/fff?text=Mouse+Up";
41         log.textContent = "Event: mouseup";
42       });
43     </script>
44   </body>
45 </html>
```

[Edit this page](#)[< Previous](#)
Session 4**Session 6** [Next >](#)