JavaScript - Mouse Events

JavaScript mouse events allow users to control and interact with web pages using their mouse. These events trigger specific functions or actions in response to user clicks, scrolls, drags, and other mouse movements.

To handle mouse events in JavaScript, you can use the addEventListener() method. The addEventListener() method takes two arguments: the event type and the event handler function. The event type is the name of the event that you want to handle, and the event handler function is the function that will be called when the event occurs.

In web development, JavaScript provides a powerful mechanism to respond to user interactions with the mouse through a set of events. These events enable developers to create dynamic and interactive web applications by capturing and handling various mouse-related actions.

Common Mouse Events

Following are the most common JavaScript mouse events:

Mouse Event	**Description**
Click	When an element experiences the press of a mouse button, it triggers the click event.
Double Click	The dblclick event fires upon the rapid double-clicking of a mouse button.
Mouse Down and Mouse Up	The initiation of a mouse click triggers the 'mousedown' event, while the completion of that click causes the 'mouseup' event to occur.
Mouse Move	When the mouse pointer moves over an element, it triggers the 'mousemove' event; this event supplies developers with positional information about the mouse. This data empowers them to devise responsive interfaces that are rooted in dynamic mouse movements.
Context Menu	When the user attempts to open the context menu, typically by right-clicking, they trigger the contextmenu event. This event allows developers to customize or inhibit default behaviour of the context menu.

Wheel	When the mouse wheel rotates, it fires the 'wheel event'; this particular event commonly manifests in implementing features, notably zooming or scrolling.
Drag and Drop	Events like dragstart, dragend, dragover, dragenter, dragleave, and drop are associated with drag-and-drop functionality. They allow developers to create interactive interfaces for dragging elements within a web page.

Example : Click Event

In this example we demonstrate the click event. When the button is clicked, it prints an appropriate message to the console message i.e. “Clicked!”. This event is often used while submitting forms.

Open Compiler

```
<!DOCTYPE html>
<html>
<head>
   <title>Click Event Example</title>
</head>
<body>
   <button id="clickButton">Click me!</button>
   <p id = "output"></p>
   <script>
      const clickButton = document.getElementById('clickButton');
      const outputDiv = document.getElementById("output");
      clickButton.addEventListener('click', function(event) {
         outputDiv.innerHTML += 'Clicked!'+ JSON.stringify(event) + "<br>";
      });
   </script>
</body>
</html>
```

Learn **JavaScript** in-depth with real-world projects through our **JavaScript certification course**. Enroll and become a certified expert to boost your career.

Example: Double Click Event

The dblclick event operates in this example, triggering upon a double-click of the designated button. We attach the event listener to an element with "doubleClickButton" as its id. A user's double-click on the button prompts a function that logs a console message, confirming their interaction with it.

Open Compiler

```
<!DOCTYPE html>
<html>
<head>
   <title>Double Click Event Example</title>
</head>
<body>
   <button id="doubleClickButton">Double-click me!</button>
   <p id = "output"></p>
   <script>
      const doubleClickButton = document.getElementById('doubleClickButton');
      const outputDiv = document.getElementById("output");
      doubleClickButton.addEventListener('dblclick', function(event) {
         outputDiv.innerHTML += 'Double-clicked!' + JSON.stringify(event) + "<b
      });
   </script>
</body>
</html>
```

Example: Mouse Down and Mouse Up Events

The use of the mousedown and mouseup events is exemplified in this scenario: both events apply to a <div> element identified as "mouseUpDownDiv." Two distinct event listeners are established; one responds to the down action of the mouse button, while another reacts upon release or up motion of said button. Upon pressing over the designated div (mousedown), a message indicating that the user has depressed their mouse button appears within your console log. When the user releases the mouse button (mouseup), we also log another message to indicate that the mouse button is up.

Open Compiler

```
<!DOCTYPE html>
<html>
<head>
   <title>Mouse Down and Mouse Up Events Example</title>
</head>
```

```
<body>
   <div id="mouseUpDownDiv"
   style="width: 600px; height: 100px; background-color: lightblue;">
   Please perform mouse down and up event any where in this DIV.
   </div>
   <p id = "output"></p>
   <script>
      const mouseUpDownDiv = document.getElementById('mouseUpDownDiv');
      const outputDiv = document.getElementById("output");
      mouseUpDownDiv.addEventListener('mousedown', function(event) {
         outputDiv.innerHTML += 'Mouse button down!' + JSON.stringify(event) +
      });

      mouseUpDownDiv.addEventListener('mouseup', function(event) {
         outputDiv.innerHTML += 'Mouse button up!' + JSON.stringify(event) + "<
      });
   </script>
</body>
</html>
```

Example: Mouse Move Event

In this instance, we employ the mousemove event to monitor the mouse pointer's movement over a specific <div> element identified as "mouseMoveDiv." The handler function extracts clientX and clientY properties from an event object that represents X-Y coordinates of said pointer. Subsequently, these are logged into console; thus offering real-time feedback on where exactly within our designated div area is the user positioning their cursor.

Open Compiler

```
<!DOCTYPE html>
<html>
<head>
   <title>Mouse Move Event Example</title>
</head>
<body>
   <div id="mouseMoveDiv"
   style="width: 600px; height: 200px; background-color: lightgreen;">
   Please move you mouse inside this DIV.</div>
   <p id = "output"></p>
   <script>
```

```
      const mouseMoveDiv = document.getElementById('mouseMoveDiv');
      const outputDiv = document.getElementById("output");
      mouseMoveDiv.addEventListener('mousemove', function(event) {
         const x = event.clientX;
         const y = event.clientY;
         outputDiv.innerHTML += `Mouse moved to (${x}, ${y})` + JSON.stringify(
      });
   </script>
</body>
</html>
```

Example: Wheel Event

This example showcases the wheel event, activated when the mouse wheel is rotated. The event listener is attached to a <div> element with the id "wheelDiv." When the user rotates the mouse wheel over this div, the associated function logs a message to the console, indicating that the mouse wheel has been rotated.

Open Compiler

```
<!DOCTYPE html>
<html>
<head>
   <title>Wheel Event Example</title>
</head>
<body>
   <div id="wheelDiv"
   style="width: 600px; height: 200px; background-color: palevioletred;">
   Please bring the curser inside this DIV and rotate the wheel of mouse.</div>
   <p id = "output"></p>
   <script>
      const wheelDiv = document.getElementById('wheelDiv');
      const outputDiv = document.getElementById("output");
      wheelDiv.addEventListener('wheel', function(event) {
         outputDiv.innerHTML += 'Mouse wheel rotated!'+ event + "<br>";
      });
   </script>
</body>
</html>
```