Lesson 7: DOM Events and Event Handling

Objective

By the end of this lesson, students will understand how to work with DOM events, how to handle user interactions, and how to create dynamic, interactive web pages using event listeners in JavaScript.

1. Understanding DOM Events

- **What Are Events?**
  - Events are actions or occurrences that happen in the browser, such as clicking a button, hovering over an element, or pressing a key.
  - The browser generates events in response to user actions or browser activities (e.g., loading a page).
- **Event Flow**
  - **Event Capturing:** The event is first captured by the outermost element and propagated down to the target element.
  - **Event Bubbling:** After reaching the target element, the event bubbles up to the outermost element.
  - By default, most events in JavaScript use event bubbling.

2. Adding Event Listeners

- **Introduction to `addEventListener`**
  - `addEventListener` is a method used to attach an event handler to a specified element.
  - Syntax:

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

    - `event` : The type of event to listen for (e.g., 'click', 'mouseover').
    - `function` : The function to execute when the event occurs.
    - `useCapture` : Optional. A boolean value indicating whether the event should be captured during the capturing phase (`true`) or the bubbling phase (`false`).
- **Example: Click Event**

  ```
  let button = document.getElementById('myButton');
  button.addEventListener('click', function() {
      alert('Button was clicked!');
  });
  ```

- **Removing Event Listeners**
  - You can remove an event listener using `removeEventListener` .

    ```
    button.removeEventListener('click', function);
    ```

3. Common DOM Events

- **Mouse Events**
  - **`click`** : Triggered when an element is clicked.
  - **`dblclick`** : Triggered when an element is double-clicked.
  - **`mouseover`** : Triggered when the mouse pointer hovers over an element.
  - **`mouseout`** : Triggered when the mouse pointer leaves an element.

```
let box = document.getElementById('box');
box.addEventListener('mouseover', function() {
    box.style.backgroundColor = 'blue';
});
```

- **Keyboard Events**
  - **`keydown`** : Triggered when a key is pressed down.
  - **`keyup`** : Triggered when a key is released.

```
document.addEventListener('keydown', function(event) {
    console.log('Key pressed:', event.key);
});
```

- **Form Events**
  - **`submit`** : Triggered when a form is submitted.
  - **`focus`** : Triggered when an input field is focused.
  - **`blur`** : Triggered when an input field loses focus.

```
let form = document.getElementById('myForm');
form.addEventListener('submit', function(event) {
    event.preventDefault();  // Prevents the form from submitting
    alert('Form submitted!');
});
```

4. Event Delegation

- **What is Event Delegation?**
  - Event delegation is a technique where you add a single event listener to a parent element to manage events triggered by its child elements.
  - This is efficient, especially when dealing with many similar elements.
- **Example: Handling Clicks on Multiple List Items**

```
let list = document.getElementById('myList');
list.addEventListener('click', function(event) {
    if (event.target.tagName === 'LI') {
        event.target.style.color = 'red';
    }
});
```

- **Advantages of Event Delegation**
  - Reduces memory usage by avoiding multiple event listeners.
  - Simplifies the code when working with dynamically added elements.

5. Hands-On Practice

Exercise 1: Interactive Button

- Create a button that changes its text content when clicked.

Exercise 2: Keyboard Interaction

- Write a script that changes the background color of the page when certain keys are pressed (e.g., 'R' for red, 'G' for green).

Exercise 3: Form Validation

- Create a form with an input field for an email address. Write a script that checks if the input field is empty when the form is submitted. If it is, display an error message.

6. Homework/Assignment

Assignment 1: To-Do List

- Create a to-do list where you can add new tasks by typing in an input field and pressing a button. Each task should be clickable, allowing users to mark it as complete (e.g., change its color or add a strikethrough).

Assignment 2: Image Gallery

- Create an image gallery where clicking on a thumbnail changes the main displayed image. Use event delegation to manage the clicks on the thumbnails.

Assignment 3: Dynamic Form Fields

- Write a script that dynamically adds a new input field to a form every time a button is clicked.

7. Recommended Resources

Documentation and Tutorials

- MDN Web Docs: Event Reference
  - A comprehensive reference for all DOM events, including detailed descriptions and usage examples.
- JavaScript.info: Bubbling and Capturing
  - An explanation of event flow in the DOM, covering event capturing, bubbling, and how to manage event propagation.

Video Tutorials

- Traversy Media: JavaScript DOM Events
  - A tutorial that covers the basics of DOM events and how to use them to create interactive web pages.

- Programming with Mosh: JavaScript Event Handling
    - A video guide on handling events in JavaScript, including common event types and practical examples.