Chapter 7

Event Handling

**addEventListener in JavaScript**

**Syntax**

element.addEventListener(event, listener, useCapture);

**Example:**

**mouse-events.html**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Mouse Events Example</title>

  <style>

    .box {

      width: 200px;

      height: 200px;

      background-color: lightblue;

      border: 2px solid darkblue;

      display: flex;

      justify-content: center;

      align-items: center;

      margin: 20px auto;

      text-align: center;

    }

    .highlight {

      background-color: lightgreen;

    }

  </style>

</head>

<body>

  <div class="box" id="mouseBox">Hover or Click Me!</div>

  <p id="eventLog"><center><b style="font-family:cambria">Event log will appear here.</b></center></p>

  <script src="mouse-script.js"></script>

</body>

</html>

**mouse-script.js**

// Get references to DOM elements

const box = document.getElementById('mouseBox');

const eventLog = document.getElementById('eventLog');

// Helper function to log events

function logEvent(message) {

eventLog.textContent = message;

}

// Mouse click event

box.addEventListener('click', () => {

logEvent('Box clicked!');

box.style.backgroundColor = 'lightcoral'; // Change background color

});

// Mouse double-click event

box.addEventListener('dblclick', () => {

logEvent('Box double-clicked!');

box.style.backgroundColor = 'orange';

});

// Mouse over event

box.addEventListener('mouseover', () => {

logEvent('Mouse over the box');

box.classList.add('highlight'); // Add a highlight class

});

// Mouse out event

box.addEventListener('mouseout', () => {

logEvent('Mouse left the box');

box.classList.remove('highlight'); // Remove the highlight class

});

// Mouse down event

box.addEventListener('mousedown', () => {

logEvent('Mouse button pressed down');

box.style.border = '4px solid red';

});

// Mouse up event

box.addEventListener('mouseup', () => {

logEvent('Mouse button released');

box.style.border = '2px solid darkblue';

});

**Example:**

**keyboard-events.html**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Keyboard Events Example</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      text-align: center;

      margin-top: 50px;

    }

    .output {

      margin-top: 20px;

      font-size: 1.5rem;

      color: darkblue;

    }

    .input-box {

      padding: 10px;

      font-size: 1.2rem;

      width: 300px;

    }

  </style>

</head>

<body>

  <h1 style="color:blue"><b>Keyboard Events Demo</b></h1>

  <input type="text" id="textInput" class="input-box" placeholder="Type something here..." />

  <div class="output" id="output"></div>

  <script src="keyboard-script.js"></script>

</body>

</html>

**keyboard-script.js**

// Get references to DOM elements

const textInput = document.getElementById('textInput');

const output = document.getElementById('output');

// Helper function to display the event details

function displayEventDetails(event, eventType) {

  output.innerHTML = `

    <p><strong>Event Type:</strong> ${eventType}</p>

    <p><strong>Key:</strong> ${event.key}</p>

    <p><strong>Key Code:</strong> ${event.keyCode}</p>

    <p><strong>Alt Key Pressed:</strong> ${event.altKey}</p>

    <p><strong>Ctrl Key Pressed:</strong> ${event.ctrlKey}</p>

    <p><strong>Shift Key Pressed:</strong> ${event.shiftKey}</p>

  `;

}

// Keydown event

textInput.addEventListener('keydown', (event) => {

  displayEventDetails(event, 'keydown');

});

// Keyup event

textInput.addEventListener('keyup', (event) => {

  displayEventDetails(event, 'keyup');

});

// Keypress event (deprecated but included for demonstration)

textInput.addEventListener('keypress', (event) => {

  displayEventDetails(event, 'keypress');

});

**Event Propagation Example**

**HTML Structure**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Event Propagation Example</title>

<style>

.grandparent, .parent, .child {

padding: 20px;

margin: 10px;

border: 1px solid black;

}

.grandparent { background-color: lightblue; }

.parent { background-color: lightgreen; }

.child { background-color: lightcoral; }

</style>

</head>

<body>

<div class="grandparent">Grandparent

<div class="parent">Parent

<div class="child">Child</div>

</div>

</div>

<script src="script.js"></script>

</body>

</html>

**JavaScript Code**

// Get references to the DOM elements

const grandparent = document.querySelector('.grandparent');

const parent = document.querySelector('.parent');

const child = document.querySelector('.child');

// Event Handlers

function logEvent(event) {

console.log(`${event.currentTarget.className} (Phase: ${event.eventPhase})`);

}

// Add event listeners

grandparent.addEventListener('click', logEvent, true); // Capturing phase

parent.addEventListener('click', logEvent); // Bubbling phase

child.addEventListener('click', logEvent); // Bubbling phase

**Example**

child.addEventListener('click', (event) => {

event.stopPropagation();

console.log('Child event stopped propagation.');

});

**Why Use event.preventDefault()?**

**Syntax**

event.preventDefault();

**Examples**

**1. Preventing Link Navigation**

<a href="https://example.com" id="myLink">Go to Example</a>

<script>

const link = document.getElementById('myLink');

link.addEventListener('click', (event) => {

event.preventDefault(); // Prevent navigation

console.log('Link clicked, but navigation prevented.');

});

</script>

**2. Preventing Form Submission**

<form id="myForm">

<input type="text" id="name" placeholder="Enter your name" />

<button type="submit">Submit</button>

</form>

<script>

const form = document.getElementById('myForm');

form.addEventListener('submit', (event) => {

const nameInput = document.getElementById('name').value;

if (!nameInput) {

event.preventDefault(); // Prevent form submission

alert('Name is required!');

}

});

</script>

**3. Preventing Default for Keyboard Events**

<input type="text" id="textBox" placeholder="Type here" />

<script>

const textBox = document.getElementById('textBox');

document.addEventListener('keydown', (event) => {

if (event.key === 'Backspace') {

event.preventDefault(); // Prevent backspace from navigating

console.log('Backspace pressed but navigation prevented.');

}

});

</script>

**4. Preventing Context Menu**

<div id="container" style="width: 200px; height: 100px; background: lightgray;">

Right-click on this box.

</div>

<script>

const container = document.getElementById('container');

container.addEventListener('contextmenu', (event) => {

event.preventDefault(); // Prevent the context menu

alert('Custom context menu action!');

});

</script>

**Form Submission Events**

form.addEventListener('submit', (event) => {

event.preventDefault(); // Prevent default form submission

console.log('Form submitted!');

});

**Key Methods for Handling Forms**

1. **Accessing Form Elements:**
   * **By ID:**

const input = document.getElementById('name');

console.log(input.value);

* + **Using the forms Collection:**

const form = document.forms['myForm'];

console.log(form['email'].value);

1. **Modifying Input Values:**

const input = document.getElementById('name');

input.value = 'John Doe';

1. **Validating Inputs:**

if (!input.value) {

alert('Name is required!');

}

1. **Preventing Default Submission:**

form.addEventListener('submit', (event) => {

event.preventDefault();

console.log('Custom submission logic here.');

});

**Example: Form Handling**

**HTML**

<form id="myForm">

<label for="name">Name:</label>

<input type="text" id="name" name="name" />

<label for="email">Email:</label>

<input type="email" id="email" name="email" />

<button type="submit">Submit</button>

</form>

**JavaScript**

const form = document.getElementById('myForm');

form.addEventListener('submit', (event) => {

event.preventDefault(); // Prevent default submission

const name = document.getElementById('name').value;

const email = document.getElementById('email').value;

if (!name || !email) {

alert('All fields are required!');

} else {

console.log(`Name: ${name}, Email: ${email}`);

// Custom submission logic, such as sending data via fetch()

}

});

**Dynamic Form Validation**

const emailInput = document.getElementById('email');

emailInput.addEventListener('input', () => {

const email = emailInput.value;

if (!email.includes('@')) {

emailInput.style.borderColor = 'red';

} else {

emailInput.style.borderColor = 'green';

}

});

**Submitting Form Data to a Server**

form.addEventListener('submit', (event) => {

event.preventDefault();

const formData = new FormData(form);

fetch('/submit', {

method: 'POST',

body: formData,

})

.then((response) => response.json())

.then((data) => {

console.log('Form submitted successfully:', data);

})

.catch((error) => {

console.error('Error submitting form:', error);

});

});

**Additional Concepts**

**1. Disabling Submit Button**

const submitButton = form.querySelector('button[type="submit"]');

form.addEventListener('submit', (event) => {

event.preventDefault();

submitButton.disabled = true;

// Submit form logic

setTimeout(() => (submitButton.disabled = false), 2000);

});

**2. Resetting Form Values**

form.reset();

**3. Custom Validation Messages**

const input = document.getElementById('email');

input.setCustomValidity('Please enter a valid email address.');

form.addEventListener('submit', (event) => {

if (!input.checkValidity()) {

event.preventDefault();

console.log(input.validationMessage);

}

});

**Comprehensive Example: Full Form Handling**

**HTML**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Registration Form</title>

    <link rel="stylesheet" href="ass2-cfile.css">

</head>

<body>

    <div class="container">

    <h1>Registration Form</h1>

    <form id="registrationForm">

        <!-- Full Name -->

        <div class="form-group">

        <label for="fullName">Full Name:</label>

        <input type="text" id="fullName" name="fullName" class="form-control" required>

        </div>

        <div class="form-group">

        <!-- Email -->

        <label for="email">Email:</label>

        <input type="email" id="email" name="email" class="form-control" required>

        <br>

    </div>

    <div class="form-group">

        <!-- Phone -->

        <label for="phone">Phone:</label>

        <input type="tel" id="phone" name="phone" pattern="[0-9]{10}" class="form-control" required>

        <small>Format: 1234567890</small>

        <br>

</div>

        <!-- Password -->

        <div class="form-group">

        <label for="password">Password:</label>

        <input type="password" id="password" name="password" class="form-control" required>

        <br>

        </div>

        <!-- Confirm Password -->

        <div class="form-group">

        <label for="confirmPassword">Confirm Password:</label>

        <input type="password" id="confirmPassword" name="confirmPassword" class="form-control" required>

        <br>

        </div>

        <!-- Gender -->

        <div class="form-group">

            <label>Gender:</label>

            <div class="gender-options">

              <input type="radio" id="male" name="gender" value="male" class="form-control" required>

              <label for="male">Male</label>

              <input type="radio" id="female" name="gender" value="female" class="form-control" required>

              <label for="female">Female</label>

              <input type="radio" id="other" name="gender" value="other" class="form-control" required>

              <label for="other">Other</label>

            </div>

          </div>

        <!-- Country -->

        <div class="form-group">

        <label for="country">Country:</label>

        <select id="country" name="country"  class="form-control" required>

            <option value="">Select Country</option>

            <option value="India">India</option>

            <option value="usa">United States</option>

            <option value="Canada">Canada</option>

            <option value="uk">United Kingdom</option>

            <option value="Japan">Japan</option>

            <option value="china">China</option>

            <!-- Add more countries as needed -->

        </select>

        <br>

        </div>

        <!-- Date of Birth -->

        <div class="form-group">

        <label for="dob">Date of Birth:</label>

        <input type="date" id="dob" name="dob" class="form-control" required>

        <br>

</div>

<div class="form-group">

<!-- Profile Picture -->

        <label for="profilePicture">Profile Picture:</label>

        <input type="file" id="profilePicture" name="profilePicture" accept="image/\*" class="form-control">

        <br>

</div>

        <!-- Image Preview -->

        <div class="form-group">

        <img id="imagePreview" src="" alt="Image Preview" style="display:none; max-width: 200px;" class="form-control">

</div>

<div class="form-group">

<!-- Age Range -->

        <label for="ageRange">Age Range:</label>

        <input type="range" id="ageRange" name="ageRange" min="0" max="100" value="25" class="form-control">

        <span id="ageValue">25</span>

        <br>

</div>

        <button type="submit">Register</button>

    </form>

</div>

    <!-- Display Information Section -->

    <h2>Submitted Information:</h2>

    <div id="infoDisplay"></div>

    <script src="ass2-jfile.js"></script>

</body>

</html>

**CSS**

body {

    font-family: cambria;

    margin: 20px;

}

h1

{

    text-align: center;

}

form {

    max-width: 1000px;

    margin: 0 auto;

}

label {

    display: block;

    margin: 10px 0 5px;

}

input, select {

    width: 100%;

    padding: 8px;

    margin-bottom: 10px;

    box-sizing: border-box;

}

button {

    padding: 10px 20px;

    background-color: #008080;

    color: white;

    border: none;

    border-radius: 4px;

    cursor: pointer;

    font-size: 16px;

    display: flex;

    justify-content: center;

    align-items: center;

}

button:hover {

    background-color: #436f44;

}

small {

    color: #888;

}

#infoDisplay {

    max-width: 600px;

    margin: 20px auto;

    padding: 10px;

    border: 1px solid #ddd;

    border-radius: 5px;

    background-color: #FFEFD5;

}

.container {

    max-width: 500px;

    margin: 50px auto;

    padding: 20px;

    border: 1px solid #ccc;

    border-radius: 5px;

    background-color: #f9f9f9;

  }

  .form-group {

    margin-bottom: 15px;

  }

  .form-control {

    width: 100%;

    padding: 8px;

    font-size: 16px;

    border: 1px solid #ccc;

    border-radius: 4px;

  }

  .gender-options {

    display: flex;

    align-items: center;

  }

  /\* To add some spacing between the radio buttons and labels \*/

  .gender-options input[type="radio"] {

    margin-right: 5px;

  }

  /\* Optional: Adjust the label spacing \*/

  .gender-options label {

    margin-right: 15px;

  }

**JavaScript**

document.getElementById('registrationForm').addEventListener('submit', function(event) {

    event.preventDefault(); // Prevent the form from submitting normally

    // Get form values

    const fullName = document.getElementById('fullName').value;

    const email = document.getElementById('email').value;

    const phone = document.getElementById('phone').value;

    var password = document.getElementById('password').value;

    const confirmPassword = document.getElementById('confirmPassword').value;

    const gender = document.querySelector('input[name="gender"]:checked')?.value;

    const country = document.getElementById('country').value;

    const dob = document.getElementById('dob').value;

    const profilePicture = document.getElementById('profilePicture').files[0];

    const ageRange = document.getElementById('ageRange').value;

    // Validation

    if (password !== confirmPassword) {

        alert('Passwords do not match!');

        return;

    }

    password = '\*\*\*\*\*';

    //confirmPassword='#####';

    // Create the information display

    let infoHTML = `<p><strong>Full Name:</strong> ${fullName}</p>`;

    infoHTML += `<p><strong>Email:</strong> ${email}</p>`;

    infoHTML += `<p><strong>Phone:</strong> ${phone}</p>`;

    infoHTML += `<p><strong>Password:</strong> ${password}</p>`;

   // infoHTML += `<p><strong>Confirm Password:</strong> ${confirmPassword}</p>`;

    infoHTML += `<p><strong>Gender:</strong> ${gender || 'Not selected'}</p>`;

    infoHTML += `<p><strong>Country:</strong> ${country}</p>`;

    infoHTML += `<p><strong>Date of Birth:</strong> ${dob}</p>`;

    infoHTML += `<p><strong>Age Range:</strong> ${ageRange}</p>`;

    if (profilePicture) {

        const reader = new FileReader();

        reader.onload = function(e) {

            infoHTML += `<p><strong>Profile Picture:</strong><br><img src="${e.target.result}" alt="Profile Picture" style="max-width: 200px;"></p>`;

            document.getElementById('infoDisplay').innerHTML = infoHTML;

        };

        reader.readAsDataURL(profilePicture);

    } else {

        document.getElementById('infoDisplay').innerHTML = infoHTML;

    }

});

// Update the range value display

document.getElementById('ageRange').addEventListener('input', function() {

    document.getElementById('ageValue').textContent = this.value;

});

**Example of Event Delegation**

**HTML**

<ul id="item-list">

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

</ul>

<button id="add-item">Add Item</button>

**JavaScript**

// Select the parent element (ul)

const itemList = document.getElementById('item-list');

// Delegate the click event to the parent

itemList.addEventListener('click', (event) => {

// Check if the clicked element is an li

if (event.target && event.target.nodeName === 'LI') {

alert('Item clicked: ' + event.target.textContent);

}

});

// Dynamically add a new item

document.getElementById('add-item').addEventListener('click', () => {

const newItem = document.createElement('li');

newItem.textContent = 'New Item';

itemList.appendChild(newItem);

});

**Example: Delegating Multiple Event Types**

**HTML**

<ul id="item-list">

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

</ul>

**JavaScript**

// Select the parent element (ul)

const itemList = document.getElementById('item-list');

// Delegate both click and mouseenter events

itemList.addEventListener('click', (event) => {

if (event.target && event.target.nodeName === 'LI') {

alert('Item clicked: ' + event.target.textContent);

}

});

itemList.addEventListener('mouseenter', (event) => {

if (event.target && event.target.nodeName === 'LI') {

event.target.style.backgroundColor = 'lightgrey'; // Highlight on hover

}

});

itemList.addEventListener('mouseleave', (event) => {

if (event.target && event.target.nodeName === 'LI') {

event.target.style.backgroundColor = ''; // Remove highlight when hover ends

}

});

**Event Delegation for Form Elements**

**HTML**

<form id="user-form">

<input type="text" name="name" placeholder="Enter your name">

<input type="email" name="email" placeholder="Enter your email">

<button type="submit">Submit</button>

</form>

**JavaScript**

const form = document.getElementById('user-form');

// Delegate the input change event

form.addEventListener('input', (event) => {

if (event.target && event.target.nodeName === 'INPUT') {

console.log(`${event.target.name}: ${event.target.value}`);

}

});

**Creating and Dispatching Custom Events**

**1. Creating a Custom Event**

**Example with Event Constructor (No additional data):**

// Create a custom event named 'myCustomEvent'

let event = new Event('myCustomEvent');

// Add an event listener for the custom event

document.addEventListener('myCustomEvent', function() {

console.log('Custom event triggered!');

});

// Dispatch the custom event

document.dispatchEvent(event);

**Example with CustomEvent Constructor (With additional data):**

// Create a custom event with additional data

let eventWithData = new CustomEvent('userLoggedIn', {

detail: { username: 'johnDoe', time: '2024-12-24' }

});

// Add an event listener for the custom event

document.addEventListener('userLoggedIn', function(event) {

console.log('User logged in: ' + event.detail.username);

console.log('Login time: ' + event.detail.time);

});

// Dispatch the custom event

document.dispatchEvent(eventWithData);

**Understanding the CustomEvent Constructor**

**Syntax:**

new CustomEvent(type, options);

**Dispatching Custom Events**

**Example:**

// Create a custom event

let event = new CustomEvent('dataLoaded', {

detail: { data: 'Sample Data' }

});

// Select an element to dispatch the event on

let element = document.getElementById('myElement');

// Add an event listener for the custom event

element.addEventListener('dataLoaded', function(event) {

console.log('Data loaded:', event.detail.data);

});

// Dispatch the event

element.dispatchEvent(event);

**Example with Event Bubbling:**

// Create a custom event that bubbles

let event = new CustomEvent('customEventWithBubble', {

bubbles: true,

detail: { message: 'Event bubbling test' }

});

// Add event listener on the parent element

document.body.addEventListener('customEventWithBubble', function(event) {

console.log('Custom event received on body:', event.detail.message);

});

// Dispatch the event on a child element

let childElement = document.getElementById('childElement');

childElement.dispatchEvent(event);

**Removing Event Listeners for Custom Events**

**Example:**

// Create a custom event

let event = new CustomEvent('userLoggedIn', {

detail: { username: 'johnDoe' }

});

// Define the event listener

function onUserLoggedIn(event) {

console.log('User logged in:', event.detail.username);

}

// Attach the event listener

document.addEventListener('userLoggedIn', onUserLoggedIn);

// Dispatch the custom event

document.dispatchEvent(event);

// Remove the event listener

document.removeEventListener('userLoggedIn', onUserLoggedIn);

// Dispatch again (no output because the listener was removed)

document.dispatchEvent(event);

**1. Event Delegation**

**Example:**

<ul id="parent">

<li class="child">Item 1</li>

<li class="child">Item 2</li>

<li class="child">Item 3</li>

</ul>

<script>

// Attach a single event listener to the parent element

document.getElementById('parent').addEventListener('click', function(event) {

// Check if the clicked element is an <li> with class 'child'

if (event.target && event.target.matches('li.child')) {

alert('Clicked on: ' + event.target.textContent);

}

});

</script>

**Removing an Event Listener:**

let button = document.getElementById('myButton');

function handleClick() {

alert('Button clicked');

}

// Attach event listener

button.addEventListener('click', handleClick);

// Remove event listener

button.removeEventListener('click', handleClick);

**Example of Throttling:**

function throttle(fn, wait) {

let lastTime = 0;

return function() {

let now = new Date().getTime();

if (now - lastTime >= wait) {

fn();

lastTime = now;

}

};

}

window.addEventListener('scroll', throttle(function() {

console.log('Scroll event handled');

}, 1000));

**Example of Debouncing:**

function debounce(fn, delay) {

let timeoutId;

return function() {

clearTimeout(timeoutId);

timeoutId = setTimeout(fn, delay);

};

}

let input = document.getElementById('searchInput');

input.addEventListener('keyup', debounce(function() {

console.log('User stopped typing');

}, 500));

**5. Batching and Debouncing Multiple Events**

**Example:**

function handleMultipleEvents(event) {

console.log('Event triggered:', event.type);

}

let button = document.getElementById('myButton');

let input = document.getElementById('myInput');

// Attach the same handler to different events

button.addEventListener('click', handleMultipleEvents);

input.addEventListener('focus', handleMultipleEvents);