**Easy Questions:**

1. **Basic Click Event Listener:**
   * Create a button that, when clicked, changes the text inside a <p> tag.

Example:

html

Copy code

<button id="myButton">Click Me</button>

<p id="text">Original Text</p>

Task: Write JavaScript to change the <p> content to "Text Changed!" on button click.

1. **Mouseover Event Listener:**
   * Create a <div> that changes its background color when the mouse hovers over it and resets when the mouse leaves.
2. **Keyboard Event Listener:**
   * Write a function that logs every key the user presses on their keyboard in the console.
3. **Adding Event Listeners to Multiple Elements:**
   * Given a list of items, attach a click event listener to each item. When clicked, log the item text in the console.

**Medium Questions:**

1. **Toggle Event Listener:**
   * Create a button that toggles the visibility of a <div>. When clicked, it hides the <div>, and when clicked again, it shows the <div> back.
2. **Event Listener Removal:**
   * Write a function to add an event listener to a button that logs "Clicked!" when clicked. Then, after the first click, remove the event listener so it doesn't log on subsequent clicks.
3. **Prevent Default Action:**
   * Create a form with a text input and submit button. Attach a submit event listener that prevents the default form submission and logs "Form Submission Prevented" instead.
4. **Event Delegation:**
   * Create a parent <div> that contains multiple child <button> elements. Attach a click event listener to the parent <div> to handle click events on any button without adding individual listeners to each button.

**Hard Questions:**

1. **Custom Event Listener:**
   * Create a custom event called customClick that fires when a user clicks a button. When the custom event is triggered, log "Custom Event Triggered!" in the console.
2. **Event Propagation (Bubbling vs. Capturing):**
   * Create a nested structure with two <div> elements. Attach a click event listener to both the parent and child <div>. Demonstrate event propagation by logging messages for both capturing and bubbling phases when clicking the child <div>.
3. **Debouncing with Event Listeners:**
   * Implement a search input box that shows a list of suggestions based on what the user types. Use event debouncing to limit how often the suggestions are updated as the user types.
4. **Throttling a Scroll Event Listener:**
   * Attach a scroll event listener to the window that logs the current scroll position. Use throttling to limit how often the scroll position is logged, even when the user scrolls rapidly.

These exercises will help you practice event listener handling at various levels of complexity.

**Easy Level**

1. **Basic Event Listeners:**
   * Create a button that changes its text to "Clicked!" when clicked.
   * Add a mouseover event to a paragraph that changes its background color to yellow.
2. **Remove Event Listeners:**
   * Add an event listener to a button that logs "Clicked" to the console when clicked. After three clicks, remove the event listener.
3. **Listening for Keyboard Events:**
   * Add an event listener to the document that logs the key pressed by the user.

**Medium Level**

1. **Capturing vs. Bubbling:**
   * Create nested div elements with different background colors.
   * Add event listeners to each div and explore the difference between event capturing and bubbling.
   * Log the order of execution for both phases.
2. **Preventing Default Behavior:**
   * Create a form with a submit button.
   * Add an event listener that prevents the form from submitting and shows an alert instead.
3. **Delegation of Events:**
   * Create a list with 5 items. Instead of adding an event listener to each list item, attach a single event listener to the parent ul and log the clicked item's text.

**Hard Level**

1. **Event Propagation with Stop Propagation:**
   * Create a setup with a button inside a div. Attach click event listeners to both the button and the div.
   * Use event.stopPropagation() to stop the event from propagating to the div when the button is clicked.
2. **Multiple Event Types:**
   * Create an input field that changes its border color to green when focused and back to red when blurred.
3. **Throttling or Debouncing an Event:**
   * Implement a search box that logs input only after the user has stopped typing for 300ms (debouncing effect).

**JavaScript Events Assignment**

**Part 1: Easy**

**1. Simple Click Event**

* **Objective**: Write a function that changes the text of an HTML <p> element when a button is clicked.
* **Instructions**:
  1. Create a button with the text “Click Me” and a <p> element with some text.
  2. When the button is clicked, the text inside the <p> element should change to “Button Clicked!”

html

Copy code

<button id="myButton">Click Me</button>

<p id="text">This is some text</p>

<script>

// Write the JavaScript code here

</script>

**2. Mouseover Event**

* **Objective**: Change the background color of a <div> when the mouse hovers over it.
* **Instructions**:
  1. Create a <div> with some text inside.
  2. When the mouse enters the <div>, change its background color to blue.
  3. When the mouse leaves, change it back to the original color.

html

Copy code

<div id="hoverDiv">Hover over this div</div>

<script>

// Write the JavaScript code here

</script>

**3. Input Event**

* **Objective**: Capture user input from a text field and display it in real-time.
* **Instructions**:
  1. Create an input field and a <p> element.
  2. As the user types into the input field, display the value inside the <p> element in real-time.

html

Copy code

<input type="text" id="inputField" placeholder="Type something here">

<p id="displayText"></p>

<script>

// Write the JavaScript code here

</script>

**Part 2: Medium**

**4. Keydown Event**

* **Objective**: Detect key presses and display the key that was pressed.
* **Instructions**:
  1. Add an event listener that listens for any key press.
  2. Display the key pressed in a <p> element.

html

Copy code

<p>Press any key:</p>

<p id="keyPressed">Key Pressed: </p>

<script>

// Write the JavaScript code here

</script>

**5. Form Submit Event**

* **Objective**: Prevent the default form submission and display the form data without reloading the page.
* **Instructions**:
  1. Create a form with an input field for the user's name and a submit button.
  2. On form submission, prevent the default action and display the entered name below the form without refreshing the page.

html

Copy code

<form id="nameForm">

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

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

</form>

<p id="nameDisplay"></p>

<script>

// Write the JavaScript code here

</script>

**6. Toggle Button Event**

* **Objective**: Create a button that toggles the visibility of a <div>.
* **Instructions**:
  1. Create a <div> with some text and a button labeled “Show/Hide”.
  2. When the button is clicked, toggle the visibility of the <div> (i.e., show or hide it).

html

Copy code

<button id="toggleButton">Show/Hide</button>

<div id="contentDiv">This content will be toggled</div>

<script>

// Write the JavaScript code here

</script>

**Part 3: Hard**

**7. Event Delegation**

* **Objective**: Use event delegation to handle multiple buttons within a parent container.
* **Instructions**:
  1. Create a <div> that contains 3 buttons.
  2. Instead of adding a click event to each button individually, add an event listener to the <div>, and handle the clicks based on the button clicked (use event.target).
  3. When a button is clicked, display which button was clicked.

html

Copy code

<div id="buttonContainer">

<button class="myButton">Button 1</button>

<button class="myButton">Button 2</button>

<button class="myButton">Button 3</button>

</div>

<p id="buttonClicked"></p>

<script>

// Write the JavaScript code here

</script>

**8. Drag and Drop Event**

* **Objective**: Implement a drag-and-drop functionality.
* **Instructions**:
  1. Create two <div> elements: one for the draggable item and another as the drop zone.
  2. Allow the user to drag the item from one <div> to another using drag-and-drop events (dragstart, dragover, drop).

html

Copy code

<div id="dragContainer">

<div id="dragItem" draggable="true">Drag Me!</div>

</div>

<div id="dropZone">Drop Here</div>

<script>

// Write the JavaScript code here

</script>

**9. Throttle Scroll Event**

* **Objective**: Optimize a scroll event using throttling.
* **Instructions**:
  1. Create a page with some content and a scrollable area.
  2. Add an event listener for the scroll event but throttle the event so it doesn't fire continuously. Display the scroll position every time the event fires.
  3. Use a throttling mechanism (either manually or using a utility like lodash.throttle).

html

Copy code

<div id="scrollableContent" style="height: 400px; overflow: scroll;">

<!-- Add a large block of text or content here to make the area scrollable -->

</div>

<p id="scrollPos">Scroll Position: 0</p>

<script>

// Write the JavaScript code here (include throttling mechanism)

</script>

**Bonus Challenge (Optional)**

**10. Custom Event**

* **Objective**: Create and dispatch a custom event when a specific action occurs (e.g., a user clicks a button three times).
* **Instructions**:
  1. Create a button that triggers a custom event after being clicked three times.
  2. When the custom event is fired, display a message below the button.

html

Copy code

<button id="customEventButton">Click Me</button>

<p id="customEventMessage"></p>

<script>

// Write the JavaScript code here (handle custom event creation and dispatch)

</script>