**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).