Day 9: Event Handling

ایونٹس کو سنبھالنا

> **Quote of the Day:** *"Events are the heartbeat of web applications. They connect what users DO to what your code DOES."*
>
> *"ویب ایپلیکیشنز کی دھڑکن ہیں۔ یہ صارف کے اعمال کو آپ کے کوڈ سے جوڑتے ہیں۔ Events"*

📋 Today's Learning Goals (آج کے اہداف)

By the end of today, you will:

- ☐ Understand what events are and how they work
- ☐ Add event listeners to make elements interactive
- ☐ Handle different event types (click, input, submit, keyboard)
- ☐ Access and use the event object
- ☐ Prevent default browser behaviors
- ☐ Build an Interactive Counter App with multiple features

Time Breakdown (کل وقت: 150 منٹ)

- 🕐 7:00-7:05 PM (5min): Standup - Show your Profile Card!
- 🕐 7:05-8:05 PM (60min): Understanding events (3× Pomodoro)
- 🕐 8:05-8:50 PM (45min): Practice with different event types
- 🕐 8:50-9:25 PM (35min): Build Interactive Counter App
- 🕐 9:25-9:30 PM (5min): Quiz & reflection

🎯 What We're Building Today

Today you'll build an **Interactive Counter App** - a dynamic application with buttons that respond to clicks, inputs that update values, color changes based on state, and multiple ways to manipulate the counter!

Why This Matters for Your Career:
Events are how EVERY interactive website works:

- **Facebook:** Click "Like" → event fires → count increases
- **Daraz:** Type in search → event fires → suggestions appear
- **YouTube:** Click play → event fires → video starts
- **WhatsApp Web:** Type message → event fires → "typing..." shows

Without events, websites are just static posters!

🧠 سمجھنا (Understanding): What Are Events?

The Real-World Analogy

Scenario: Doorbell System (دروازے کی گھنٹی)

Imagine your house doorbell:

```
Someone outside (User Action)
    ↓
Presses button (Event Trigger)
    ↓
Bell rings inside (Event Fires)
    ↓
You hear it (Event Listener)
    ↓
You open door (Event Handler / Your Response)
```

This is EXACTLY how web events work!

```
Button on webpage (Element)
    ↓
User clicks it (Event)
    ↓
JavaScript is "listening" (Event Listener)
    ↓
Your function runs (Event Handler)
    ↓
Something happens on page (Result)
```

Daily Life Event Examples

1. Traffic Signal (ٹریفک سگنل)

```
Event: Signal turns green
Listener: You watching the signal
Handler: You press accelerator
Result: Car moves forward
```

2. Mobile Phone (موبائل فون)

```
Event: SMS notification arrives
Listener: Phone monitoring for messages
Handler: Phone shows notification
Result: You read the message
```

3. Exam Bell (امتحان کی گھنٹی)

```
Event: Bell rings (time up)
Listener: Students listening for bell
Handler: Students stop writing
Result: Papers submitted
```

Why Events Matter

Without Events (Yesterday's Knowledge):

```
// Code runs ONCE when page loads
document.querySelector("#heading").textContent = "Welcome";
// User can't interact - it's done!
```

With Events (Today's Knowledge):

```
// Code WAITS for user to click
button.addEventListener("click", function() {
    // Runs EVERY TIME user clicks!
    count++;
    display.textContent = count;
});
```

The Mental Model

Think of events like a **waiter at a restaurant** (ویٹر):

```
Customer (User)
    ↓
Raises hand (Event: click, hover, type)
    ↓
Waiter notices (Event Listener: watching for signal)
    ↓
Waiter comes over (Event Handler: your function)
    ↓
Takes order (Execute code)
    ↓
Brings food (Update webpage)
```

Waiter doesn't constantly ask "Ready? Ready? Ready?"
Waiter WAITS and responds WHEN signaled!

That's event-driven programming!

📚 Building Block #1: Understanding Event Listeners

What is an Event Listener? (کیا ہے؟)

Urdu Analogy: Event listener is like a **chowkidar** (چوکیدار) guarding your house.

```
Chowkidar's Job:
1. WATCH the gate (element)
2. WAIT for someone to come (event)
3. When someone comes (event fires)
4. DO something (call your function)
```

The Basic Pattern

Every event listener has 3 parts:

```
element.addEventListener(eventType, handlerFunction);
//  ↑           ↑              ↑            ↑
// What to     Method      What event   What to do
// watch                   to watch for  when it happens
```

Your First Event Listener

```
// THINKING: Make a button that responds to clicks

// HTML: <button id="myButton">Click Me!</button>

// Step 1: SELECT the element
const button = document.querySelector("#myButton");

// Step 2: ADD event listener
button.addEventListener("click", function() {
    // Step 3: This code runs WHEN button is clicked
    alert("Button was clicked! بٹن دبایا گیا!");
});

// NOTE: This code WAITS. It doesn't run until user clicks!
```

Breaking Down addEventListener

```
// THINKING: Understanding each part

const button = document.querySelector("#myButton");
```

```
button.addEventListener("click", function() {
    console.log("Clicked!");
});

// Part 1: button
//   - Which element are we watching?

// Part 2: addEventListener
//   - The method that sets up the "watcher"

// Part 3: "click"
//   - What user action triggers this?
//   - Other options: "mouseover", "keydown", "submit"

// Part 4: function() { ... }
//   - The code that runs when event happens
//   - This is your "event handler"
```

Why Use Functions Here?

```
// THINKING: The function is the "action plan"

// ❌ WRONG - This runs IMMEDIATELY
button.addEventListener("click", alert("Hi"));
// alert runs RIGHT NOW, not on click!

// ✅ RIGHT - This WAITS for click
button.addEventListener("click", function() {
    alert("Hi");
});
// function wraps the code - runs ONLY on click!
```

Your First Practice

```
<!DOCTYPE html>
<html>
<body>
    <button id="greet-btn">Say السلام عليكم</button>
    <p id="message"></p>
</body>
<script>
    // TODO Step 1: Select button
    const btn = document.querySelector("_____");

    // TODO Step 2: Select message paragraph
    const msg = document.querySelector("_____");
    // TODO Step 3: Add click event listener
    btn.addEventListener("_____", function() {
        // TODO Step 4: Change message text
        msg.textContent = "_____";
    });

    // Test: Click the button. Does message appear?
</script>
</html>
```

Common Mistakes

❌ **Wrong:**

```
button.addEventListener(click, function() {  // No quotes!
    console.log("Clicked");
});
```

✅ **Right:**

```
button.addEventListener("click", function() {  // Event type in quotes
    console.log("Clicked");
});
```

❌ **Wrong:**

```
button.addEventListener("click", myFunction());  // Runs immediately!
```

✅ **Right:**

```
button.addEventListener("click", myFunction);  // No (), passes function
// OR
button.addEventListener("click", function() {  // Wrap in function
    myFunction();
});
```

Check Your Understanding

- ☐ What are the 3 parts of addEventListener?
- ☐ Why do we wrap code in a function?
- ☐ What happens if you forget quotes around event type?

- ☐ Can you explain addEventListener to someone in Urdu?

📚 Building Block #2: Common Event Types

Click Events (کلک ایونٹس)

Urdu Analogy: Like pressing a **light switch** (بجلی کا سوئچ)

```
// THINKING: Most common event - button clicks!

const button = document.querySelector("#myBtn");

button.addEventListener("click", function() {
    console.log("Button clicked!");
    // This runs EVERY time user clicks
});

// Works on any element, not just buttons:
const heading = document.querySelector("h1");
heading.addEventListener("click", function() {
    this.style.color = "red";  // 'this' refers to the heading
});
```

Input Events (ان پٹ ایونٹس)

Urdu Analogy: Like watching someone write on a **whiteboard** (سفید تختہ)

```
// THINKING: Fires while user is typing!

const inputBox = document.querySelector("#name-input");

inputBox.addEventListener("input", function() {
    // This runs with EVERY keystroke
    console.log("Current value:", inputBox.value);
    console.log("Length:", inputBox.value.length);
});

// Real-world use: Live search, character counter
```

Change Events (تبدیلی ایونٹس)

Urdu Analogy: Like selecting a shirt size at **ChenOne**

```
// THINKING: Fires when selection changes

const dropdown = document.querySelector("#city-select");
```

```
dropdown.addEventListener("change", function() {
    // Runs when user picks a different option
    console.log("Selected city:", dropdown.value);
});

// Also works on checkboxes and radio buttons
```

Submit Events (جمع کرانا ایونٹس)

Urdu Analogy: Like submitting application at **NADRA office**

```
// THINKING: Fires when form is submitted

const form = document.querySelector("#registration-form");

form.addEventListener("submit", function(event) {
    // IMPORTANT: Prevent page refresh!
    event.preventDefault();

    // Now handle form data
    console.log("Form submitted!");
});
```

Keyboard Events (کی بورڈ ایونٹس)

Urdu Analogy: Like playing **piano keys** (پیانو کی کلیدیں)

```
// THINKING: Fires when user presses keys

const input = document.querySelector("#search-box");

// When key is pressed down
input.addEventListener("keydown", function(event) {
    console.log("Key pressed:", event.key);

    // Check for specific key
    if (event.key === "Enter") {
        console.log("Enter key pressed!");
    }
});

// Also: "keyup" (when key released), "keypress" (deprecated)
```

Mouse Events (ماؤس ایونٹس)

Urdu Analogy: Like moving your hand over a **candle flame** (شمع کی لو)

```
// THINKING: Track mouse movements

const box = document.querySelector("#hover-box");

// Mouse enters element
box.addEventListener("mouseenter", function() {
    box.style.backgroundColor = "yellow";
});

// Mouse leaves element
box.addEventListener("mouseleave", function() {
    box.style.backgroundColor = "white";
});

// Also: "mouseover", "mouseout", "mousemove"
```

Event Type Cheat Sheet

Event	When It Fires	Common Use
`click`	Element clicked	Buttons, links
`input`	Text being typed	Live search, counters
`change`	Selection changes	Dropdowns, checkboxes
`submit`	Form submitted	Form handling
`keydown`	Key pressed	Keyboard shortcuts
`mouseenter`	Mouse enters element	Hover effects
`focus`	Element gets focus	Input validation
`blur`	Element loses focus	Save draft

Your Practice Exercise

```
<!DOCTYPE html>
<html>
<head>
    <style>
        .box { width: 200px; height: 200px; background: lightblue;
               display: flex; align-items: center; justify-content: center; }
        .active { background: lightgreen; transform: scale(1.1); }
    </style>
</head>
<body>
    <div id="interactive-box" class="box">
        Hover and Click Me!
    </div>
    <input type="text" id="text-input" placeholder="Type here...">
    <p id="char-count">Characters: 0</p>
</body>
<script>
    const box = document.querySelector("#interactive-box");
    const input = document.querySelector("#text-input");
    const counter = document.querySelector("#char-count");

    // TODO: Add mouseenter event - add 'active' class to box
    box.addEventListener("_____", function() {
        box.classList._____("active");
    });

    // TODO: Add mouseleave event - remove 'active' class
    box.addEventListener("_____", function() {
        box.classList._____("active");
    });

    // TODO: Add click event - show alert
    box.addEventListener("_____", function() {
        alert("Box clicked! باکس دبایا!");
    });

    // TODO: Add input event - update character count
    input.addEventListener("_____", function() {
        counter.textContent = "Characters: " + input.value._____;
    });
</script>
</html>
```

Check Your Understanding

- ☐ What's the difference between "click" and "mouseenter"?
- ☐ When does "input" event fire vs "change"?
- ☐ Why is "submit" event special?
- ☐ Can you name 5 different event types?

📚 Building Block #3: The Event Object

What is the Event Object? (کیا ہے؟)

Urdu Analogy: Think of event object like **parcel tracking information** (پارسل کی معلومات):

```
When parcel arrives (event happens):
- Who sent it? (event.target)
- When did it arrive? (event.timeStamp)
- What type? (event.type)
- What's inside? (event details)
```

Accessing the Event Object

```
// THINKING: Event object is automatically passed to handler

button.addEventListener("click", function(event) {
    //                                      ↑
    //                          This parameter receives event object

    console.log(event);  // See all event properties
    console.log(event.type);  // "click"
    console.log(event.target);  // The button element
});

// Common parameter names: event, e, evt (all work the same)
```

Important Event Properties

```
// THINKING: What information does event object contain?

input.addEventListener("keydown", function(e) {
    // Which key was pressed?
    console.log(e.key);  // "a", "Enter", "Escape", etc.
    console.log(e.code);  // "KeyA", "Enter", "Escape"

    // Which element triggered this?
    console.log(e.target);  // The input element

    // What type of event?
    console.log(e.type);  // "keydown"

    // When did it happen?
    console.log(e.timeStamp);  // Milliseconds since page load

    // Was Shift/Ctrl/Alt pressed?
    console.log(e.shiftKey);  // true/false
    console.log(e.ctrlKey);   // true/false
    console.log(e.altKey);    // true/false
});
```

Using event.target

```
// THINKING: event.target is the element that triggered event

// Example: Multiple buttons using ONE event handler

const buttons = document.querySelectorAll(".color-btn");
```

```
buttons.forEach(function(button) {
    button.addEventListener("click", function(e) {
        // Which button was clicked?
        const clickedButton = e.target;

        // Get data from button
        const color = clickedButton.dataset.color;
        document.body.style.backgroundColor = color;

        console.log("Button text:", e.target.textContent);
    });
});
```

Practical Example: Form Input

```
// THINKING: Get value from input that fired event

const inputs = document.querySelectorAll("input");

inputs.forEach(function(input) {
    input.addEventListener("input", function(e) {
        // Which input is being typed in?
        console.log("Input name:", e.target.name);
        console.log("Current value:", e.target.value);
        console.log("Input type:", e.target.type);
    });
});
```

Your First Example

```
<!DOCTYPE html>
<html>
<body>
    <button data-number="1">Button 1</button>
    <button data-number="2">Button 2</button>
    <button data-number="3">Button 3</button>
    <p id="result">Click any button...</p>
</body>
<script>
    const buttons = document.querySelectorAll("button");
    const result = document.querySelector("#result");

    buttons.forEach(function(btn) {
        btn.addEventListener("click", function(event) {
            // TODO: Get button number from dataset
            const num = event.target.dataset._____;

            // TODO: Get button text
            const text = event.target._____;

            // TODO: Display info
            result.textContent = `You clicked ${text} (Number: ${num})`;
        });
    });
</script>
</html>
```

Common Event Object Properties

Property	What It Contains	Example Use
`event.target`	Element that triggered event	Identify which button clicked
`event.type`	Type of event	Check if "click" or "submit"
`event.key`	Keyboard key pressed	Check for "Enter" or "Escape"
`event.preventDefault()`	Stop default action	Prevent form submission
`event.currentTarget`	Element listener is attached to	Different from target!
`event.timeStamp`	When event occurred	Measure response time

Check Your Understanding

- ☐ What is event.target?
- ☐ How do you access which key was pressed?
- ☐ What's automatically passed to event handler?
- ☐ Why is event object useful?

📚 Building Block #4: Preventing Default Behaviors

What are Default Behaviors? (کیا ہیں؟)

Urdu Analogy: Think of default behaviors like **automatic actions** (خودکار اعمال):

```
Link clicked → Browser navigates to URL (default)
Form submitted → Page refreshes (default)
Right-click → Context menu appears (default)

Sometimes you want to STOP these automatic actions!
```

Why Prevent Defaults?

```
// THINKING: Why would we stop default behavior?

// Example 1: Form submission
// Default: Page refreshes, lose all data
// We want: Validate first, then submit via JavaScript

// Example 2: Link click
// Default: Navigate away from page
// We want: Open modal/popup instead

// Example 3: Context menu
// Default: Browser menu appears
// We want: Custom menu instead
```

Using event.preventDefault()

```
// THINKING: How to prevent default action

const form = document.querySelector("#myForm");

form.addEventListener("submit", function(event) {
    // STOP the default form submission!
    event.preventDefault();

    // Now WE control what happens
    console.log("Form submitted, but page didn't refresh!");

    // Get form data
    const name = document.querySelector("#name").value;
    console.log("Name:", name);

    // You can now:
    // - Validate data
    // - Send to API
    // - Show success message
    // - NOT refresh the page!
});
```

Real Example: Link Prevention

```
// THINKING: Stop link from navigating

const link = document.querySelector("#special-link");

link.addEventListener("click", function(e) {
    // Stop default navigation
    e.preventDefault();
    // Do something else instead
    alert("Link clicked, but didn't navigate!");

    // Or maybe open a modal:
    document.querySelector("#modal").style.display = "block";
});
```

Form Validation Example

```
<!DOCTYPE html>
<html>
<body>
    <form id="login-form">
        <input type="text" id="username" placeholder="Username" required>
        <input type="password" id="password" placeholder="Password" required>
        <button type="submit">Login</button>
    </form>
    <p id="message"></p>
</body>
<script>
    const form = document.querySelector("#login-form");
    const message = document.querySelector("#message");

    form.addEventListener("submit", function(e) {
        // PREVENT default form submission
        e.preventDefault();

        // Get values
        const username = document.querySelector("#username").value;
        const password = document.querySelector("#password").value;

        // Validate
        if (username.length < 3) {
            message.textContent = "Username too short!";
            message.style.color = "red";
            return;  // Stop here
        }

        if (password.length < 6) {
            message.textContent = "Password must be 6+ characters!";
            message.style.color = "red";
            return;
        }

        // If we reach here, validation passed!
        message.textContent = "✅ Login successful!";
        message.style.color = "green";

        // Here you would normally send data to server
        console.log("Sending to server:", {username, password});
    });
</script>
</html>
```

Your Practice Exercise

```
<!DOCTYPE html>
<html>
<body>
    <a href="https://google.com" id="external-link">
        Don't Navigate - Show Alert Instead
    </a>

    <form id="search-form">
        <input type="text" id="search" placeholder="Search..." required>
        <button type="submit">Search</button>
    </form>
    <div id="search-results"></div>
</body>
<script>
    const link = document.querySelector("#external-link");
    const searchForm = document.querySelector("#search-form");
    const results = document.querySelector("#search-results");

    // TODO: Prevent link navigation
    link.addEventListener("_____", function(e) {
        e._____();  // Stop navigation
        alert("Link clicked! But didn't navigate.");
    });

    // TODO: Prevent form submission and show results instead
    searchForm.addEventListener("_____", function(e) {
        e._____();  // Stop page refresh

        const query = document.querySelector("#search").value;
        results.textContent = `Searching for: ${query}...`;

        // In real app, you'd fetch results from API here
    });
</script>
</html>
```

When to Use preventDefault()

✅ **Use it when:**

- Handling form submissions with JavaScript
- Creating single-page applications (no page refresh)

- Building custom behaviors for links
- Implementing keyboard shortcuts
- Creating drag-and-drop interfaces

❌ **Don't use it when:**

- You want normal form submission to backend
- Links should navigate normally
- Default behavior is what you need

Check Your Understanding

- ☐ What does preventDefault() do?
- ☐ Why is it useful for forms?
- ☐ When should you NOT use it?
- ☐ Can you give 3 examples of default behaviors?

💻 Practice Session: Event Handling Mastery

🎯 Practice Goal

By the end of this section, you'll handle multiple event types confidently!

Exercise 1: Color Changer (ہم ساتھ کریں)

Scenario: Create buttons that change page background color

HTML:

```
<!DOCTYPE html>
<html>
<head>
    <style>
        body { transition: background-color 0.3s; padding: 20px; }
        .color-btn { padding: 10px 20px; margin: 5px; cursor: pointer; }
    </style>
</head>
<body>
    <h1>Color Changer</h1>
    <button class="color-btn" data-color="#ff6b6b">Red</button>
    <button class="color-btn" data-color="#4ecdc4">Teal</button>
    <button class="color-btn" data-color="#ffe66d">Yellow</button>
    <button class="color-btn" data-color="#95e1d3">Mint</button>
</body>
<script>
    // TODO Step 1: Select all buttons
    const buttons = document.querySelectorAll("_____");

    // TODO Step 2: Add click event to each button
    buttons.forEach(function(button) {
        button.addEventListener("_____", function(event) {
            // TODO Step 3: Get color from data attribute
            const color = event.target.dataset._____;

            // TODO Step 4: Change body background
            document.body.style.backgroundColor = _____;
        });
    });
</script>
</html>
```

Exercise 2: Live Character Counter (اب آپ)

Problem: Create a textarea that shows remaining characters (like Twitter)

Requirements:

- ☐ Textarea with 280 character limit
- ☐ Shows "X / 280 characters" live
- ☐ Color changes: green (lots left), orange (close), red (limit reached)
- ☐ Prevent typing when limit reached

Starter Code:

```
<!DOCTYPE html>
<html>
<head>
    <style>
        .counter { font-size: 14px; margin-top: 5px; }
        .safe { color: green; }
        .warning { color: orange; }
        .danger { color: red; }
    </style>
</head>
<body>
    <h2>Tweet Composer</h2>
    <textarea id="tweet" rows="4" cols="50"
              placeholder="What's happening? (280 chars max)"></textarea>
    <p id="counter" class="counter">0 / 280</p>
</body>
<script>
    const textarea = document.querySelector("#tweet");
    const counter = document.querySelector("#counter");
    const MAX = 280;

    // TODO: Add input event listener
    textarea.addEventListener("_____", function() {
        // TODO: Get current length
        const length = textarea.value._____;
        // TODO: Update counter text
        counter.textContent = `${length} / ${MAX}`;

        // TODO: Remove all color classes first
        counter.classList.remove("safe", "warning", "danger");

        // TODO: Add appropriate color class
        if (length < MAX * 0.7) {  // Less than 70%
            counter.classList.add("_____");  // green
        } else if (length < MAX * 0.9) {  // Less than 90%
            counter.classList.add("_____");  // orange
        } else {
            counter.classList.add("_____");  // red
        }

        // TODO: Prevent typing if at limit
        if (length >= MAX) {
            // Cut off extra characters
            textarea.value = textarea.value.substring(0, _____);
        }
    });
</script>
</html>
```

Don't Look Below Until You Try! ⬇️

Hints (if stuck):

▶ Stuck on event type?

Use "input" event - it fires for every character typed/deleted.

▶ Stuck on color logic?

```
const length = textarea.value.length;
counter.classList.remove("safe", "warning", "danger");

if (length < 196) {
    counter.classList.add("safe");
} else if (length < 252) {
    counter.classList.add("warning");
} else {
    counter.classList.add("danger");
}
```

Exercise 3: Keyboard Shortcuts

Problem: Create keyboard shortcuts for common actions

```
<!DOCTYPE html>
<html>
<body>
    <h1>Keyboard Shortcuts Demo</h1>
    <p>Try these shortcuts:</p>
    <ul>
        <li>Ctrl + S = Save (shows alert)</li>
        <li>Ctrl + P = Print (shows alert)</li>
        <li>Escape = Close message</li>
    </ul>
    <p id="message" style="display:none; padding:10px; background:#ffe66d;">
</p>
</body>
<script>
    const message = document.querySelector("#message");

    // TODO: Add keydown event to entire document
    document.addEventListener("_____", function(e) {
        // TODO: Check for Ctrl+S
        if (e._____ && e.key === "_____") {
            e.preventDefault();  // Stop browser's save dialog
            message.textContent = "✅ Saved! (جمع ہو گیا)";
            message.style.display = "block";
        }

        // TODO: Check for Ctrl+P
        if (e._____ && e.key === "_____") {
            e.preventDefault();  // Stop browser's print dialog
            message.textContent = "🖨️ Print! (پرنٹ کریں)";
            message.style.display = "block";
        }

        // TODO: Check for Escape
        if (e.key === "_____") {
            message.style.display = "none";
        }
    });
</script>
</html>
```

🚀 اج کا چیلنج (Today's Challenge)

Project: Interactive Counter App

انٹرایکٹو کاؤنٹر ایپ

The Problem:
Build a complete counter application that responds to multiple user actions - clicking buttons, typing numbers, and changing colors based on state. This is similar to apps like step counters, score keepers, or inventory managers!

What You're Building:
A fully interactive counter with:

- Buttons to increment, decrement, and reset
- Input to jump to specific number
- Color changes based on value
- Smooth animations
- Multiple increment/decrement amounts

Success Criteria:

- [ ] Counter displays starting at 0
- [ ] Increment button adds 1
- [ ] Decrement button subtracts 1
- [ ] Reset button returns to 0
- [ ] Input allows jumping to any number
- [ ] Colors change: negative=red, zero=gray, positive=green
- [ ] All interactions work smoothly

Phase 1: Planning (سوچیں پہلے)

Before coding, answer:

1. What HTML elements do I need?

```
- Display for counter value (h1)
- Increment button
- Decrement button
- Reset button
- Input for jump to number
- Submit button for input
```

2. What events do I need?

```
- Click events for all buttons
- Submit event for form
- Input event for live feedback (optional)
```

3. What state do I need to track?

```
- Current counter value (number variable)
```

4. What's my logic flow?

```
Increment: value++, update display, update color
Decrement: value--, update display, update color
Reset: value = 0, update display, update color
Jump: value = input, update display, update color

Pattern: All actions modify value, then refresh UI!
```

Planning Checkpoint:

- ☐ I know what events to listen for
- ☐ I understand state management (counter variable)
- ☐ I know how to update display and colors
- ☐ I see the pattern in all actions

Phase 2: Foundation (بنیاد)

Starter Code:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Interactive Counter</title>
    <style>
        * { margin: 0; padding: 0; box-sizing: border-box; }

        body {
            font-family: 'Segoe UI', Arial, sans-serif;
            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
            min-height: 100vh;
            display: flex;
            justify-content: center;
            align-items: center;
            padding: 20px;
        }

        .container {
            background: white;
            padding: 40px;
            border-radius: 20px;
            box-shadow: 0 10px 40px rgba(0,0,0,0.2);
```

```css
            text-align: center;
            max-width: 400px;
            width: 100%;
        }

        h1 {
            color: #333;
            margin-bottom: 10px;
            font-size: 18px;
            text-transform: uppercase;
            letter-spacing: 2px;
        }

        #counter {
            font-size: 80px;
            font-weight: bold;
            margin: 20px 0;
            transition: all 0.3s ease;
        }

        .positive { color: #27ae60; }
        .negative { color: #e74c3c; }
        .zero { color: #95a5a6; }

        .buttons {
            display: flex;
            justify-content: center;
            gap: 10px;
            margin: 20px 0;
            flex-wrap: wrap;
        }

        button {
            padding: 12px 24px;
            font-size: 16px;
            border: none;
            border-radius: 8px;
            cursor: pointer;
            transition: all 0.3s ease;
            font-weight: bold;
        }

        .btn-increment {
            background: #27ae60;
            color: white;
        }
        .btn-increment:hover { background: #229954; transform: scale(1.05); }

        .btn-decrement {
            background: #e74c3c;
            color: white;
        }
        .btn-decrement:hover { background: #c0392b; transform: scale(1.05); }
        .btn-reset {
            background: #95a5a6;
            color: white;
        }
        .btn-reset:hover { background: #7f8c8d; transform: scale(1.05); }

        .jump-section {
            margin-top: 30px;
            padding-top: 20px;
            border-top: 2px solid #ecf0f1;
        }

        .jump-section h3 {
            font-size: 14px;
            color: #7f8c8d;
            margin-bottom: 10px;
        }

        #jump-form {
            display: flex;
            gap: 10px;
            justify-content: center;
        }

        #jump-input {
            padding: 10px;
            font-size: 16px;
            border: 2px solid #bdc3c7;
            border-radius: 8px;
            width: 120px;
            text-align: center;
        }

        .btn-jump {
            background: #3498db;
            color: white;
        }
        .btn-jump:hover { background: #2980b9; transform: scale(1.05); }

        @keyframes bounce {
            0%, 100% { transform: scale(1); }
            50% { transform: scale(1.1); }
        }

        .animate {
            animation: bounce 0.3s ease;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Interactive Counter</h1>
        <h2 id="counter" class="zero">0</h2>

        <div class="buttons">
            <button id="btn-decrement" class="btn-decrement">-
Decrease</button>
            <button id="btn-reset" class="btn-reset">↻ Reset</button>
            <button id="btn-increment" class="btn-increment">+
Increase</button>
        </div>

        <div class="jump-section">
            <h3>Jump to Number</h3>
            <form id="jump-form">
                <input type="number" id="jump-input" placeholder="Enter
number">
                <button type="submit" class="btn-jump">Go</button>
            </form>
        </div>

        <!-- BONUS SECTION: Uncomment to enable -->
        <!--
        <div class="buttons" style="margin-top: 20px;">
            <button id="btn-dec-5" class="btn-decrement">-5</button>
            <button id="btn-inc-5" class="btn-increment">+5</button>
        </div>
        -->
    </div>

    <script>
        // ===================================
        // INTERACTIVE COUNTER APP
        // By: [Your Name]
        // ===================================

        // ========== STATE MANAGEMENT ==========

        // TODO Step 1: Initialize counter value
        let counterValue = _____;  // Start at 0

        // ========== SELECT ELEMENTS ==========

        // TODO Step 2: Select all necessary elements
        const counterDisplay = document.querySelector("#_____");
        const btnIncrement = document.querySelector("#_____");
        const btnDecrement = document.querySelector("#_____");
        const btnReset = document.querySelector("#_____");
        const jumpForm = document.querySelector("#_____");
        const jumpInput = document.querySelector("#_____");

        // ========== HELPER FUNCTION: Update Display ==========

        function updateDisplay() {
            // TODO Step 3: Update counter text
            counterDisplay.textContent = _____;

            // TODO Step 4: Remove all color classes
            counterDisplay.classList.remove("_____", "_____", "_____");

            // TODO Step 5: Add appropriate color class based on value
            // HINT: Use if-else to check: positive, negative, or zero
            if (counterValue > 0) {
                counterDisplay.classList.add("_____");
            } else if (counterValue < 0) {
                counterDisplay.classList.add("_____");
            } else {
                counterDisplay.classList.add("_____");
            }

            // BONUS: Add animation class
            counterDisplay.classList.add("animate");
            // Remove animation class after animation completes
            setTimeout(function() {
                counterDisplay.classList.remove("animate");
            }, 300);
        }

        // ========== EVENT HANDLERS ==========

        // TODO Step 6: Increment button click
        btnIncrement.addEventListener("_____", function() {
            // TODO: Increase counter by 1
            counterValue_____1;

            // TODO: Update the display
            _____();

            console.log("Counter:", counterValue);
        });

        // TODO Step 7: Decrement button click
        btnDecrement.addEventListener("_____", function() {
            // TODO: Decrease counter by 1
            counterValue_____1;

            // TODO: Update the display
            _____();

            console.log("Counter:", counterValue);
        });

        // TODO Step 8: Reset button click
        btnReset.addEventListener("_____", function() {
            // TODO: Set counter back to 0
            counterValue = _____;

            // TODO: Update the display
```

```
        _____();

        console.log("Counter reset to 0");
    });

    // TODO Step 9: Jump form submission
    jumpForm.addEventListener("_____", function(event) {
        // TODO: Prevent page refresh
        event._____();

        // TODO: Get value from input
        const jumpValue = _____(_____);
        // HINT: Need to convert string to number!

        // TODO: Validate input (is it a number?)
        if (isNaN(jumpValue)) {
            alert("Please enter a valid number!");
            return;
        }

        // TODO: Update counter value
        counterValue = _____;

        // TODO: Update display
        _____();

        // TODO: Clear input field
        jumpInput.value = "";

        console.log("Jumped to:", counterValue);
    });

    // ========== BONUS: +5 and -5 Buttons ==========
    // Uncomment the HTML section and add these listeners

    /*
    const btnInc5 = document.querySelector("#btn-inc-5");
    const btnDec5 = document.querySelector("#btn-dec-5");

    btnInc5.addEventListener("click", function() {
        counterValue += 5;
        updateDisplay();
    });

    btnDec5.addEventListener("click", function() {
        counterValue -= 5;
        updateDisplay();
    });
    */

    // ========== INITIALIZE ==========
    // Update display when page loads
    updateDisplay();
```

```
        console.log("✅ Counter app loaded!");
    </script>
</body>
</html>
```

Phase 3: Milestones (سنگ میل)

Milestone 1: Basic Counter Works ✅

- [ ] Page loads with 0 displayed
- [ ] Increment button adds 1
- [ ] Decrement button subtracts 1
- [ ] Console logs show correct values
- Test: Click buttons 5 times each

Milestone 2: Reset Works ✅

- [ ] Reset button returns counter to 0
- [ ] Works from any value (positive or negative)
- Test: Go to 10, reset, go to -5, reset

Milestone 3: Colors Change ✅

- [ ] Positive numbers show green
- [ ] Negative numbers show red
- [ ] Zero shows gray
- Test: Increment to +3 (green), decrement to -2 (red), reset (gray)

Milestone 4: Jump Feature Works ✅

- [ ] Can type number in input
- [ ] Submit button sets counter to that number
- [ ] Invalid input shows alert
- [ ] Input clears after submit
- Test: Jump to 50, jump to -20, try typing "abc"

Milestone 5: Animations Work ✅

- [ ] Counter bounces on change
- [ ] Button hover effects work
- [ ] Smooth color transitions
- Test: Click rapidly - animations should be smooth

Debugging Guide (اگر پھنس جائیں)

Problem: Button click doesn't do anything

- ☐ Check: Did you add event listener?
- ☐ Check: Is querySelector spelling correct?
- ☐ Check: Did you call updateDisplay()?
- ☐ Add: console.log in click handler to test

Problem: Counter shows NaN

- ☐ Check: Did you initialize counterValue = 0?
- ☐ Check: Are you using ++ or -- correctly?
- ☐ Check: Did you convert input to number? (parseInt or Number())

Problem: Colors don't change

- ☐ Check: Are class names spelled correctly?
- ☐ Check: Did you remove old classes first?
- ☐ Check: Is your if-else logic correct?
- ☐ Inspect element (F12) - are classes being added?

Problem: Form refreshes page

- ☐ Check: Did you use event.preventDefault()?
- ☐ Check: Is it inside submit event handler?
- ☐ Check: Did you add to form, not button?

Problem: Jump input doesn't work

- ☐ Check: Did you get value from input?
- ☐ Check: Did you convert to number?
- ☐ Check: Is validation working?
- ☐ Console.log the jumpValue - is it a number?

Extension Challenges (بونس چیلنج)

If you finish early:

🌟 Level 1: Step Size Selector

```
// Add dropdown to choose step size (1, 5, 10)
const stepSelect = document.querySelector("#step-size");

btnIncrement.addEventListener("click", function() {
    const step = parseInt(stepSelect.value);
    counterValue += step;
    updateDisplay();
});
```

🌟🌟 Level 2: History Tracking

```
// Track all values in an array
const history = [];

function updateDisplay() {
    // ... existing code ...
    history.push(counterValue);

    // Show last 5 values
    console.log("History:", history.slice(-5));
}

// Add "Undo" button
function undo() {
    if (history.length > 1) {
        history.pop();  // Remove current
        counterValue = history[history.length - 1];
        updateDisplay();
    }
}
```

🌟🌟🌟 Level 3: Local Storage Persistence

```
// Save counter value to localStorage
function updateDisplay() {
    // ... existing code ...
    localStorage.setItem("counterValue", counterValue);
}

// Load saved value on page load
const saved = localStorage.getItem("counterValue");
if (saved !== null) {
    counterValue = parseInt(saved);
    updateDisplay();
}
```

📝 Daily Quiz (5 منٹ کا ٹیسٹ)

Instructions: Answer WITHOUT looking at notes!

1. What does addEventListener do?

- A) Creates a new element
- B) Waits for and responds to user actions
- C) Changes element style
- D) Deletes an element

▶ See Answer (Try first!)

Answer: B - addEventListener sets up a "watcher" that waits for a specific user action (event) and runs your code (handler function) when it happens. Like a chowkidar watching the gate!

---

2. What is the correct syntax for adding a click event?

- A) `element.onClick("click", function)`
- B) `element.addEventListener("click", function)`
- C) `element.addEventListener(click, function())`
- D) `element.addEvent("click")`

▶ See Answer (Try first!)

Answer: B - Correct syntax is `element.addEventListener("click", function)`. Event type must be in quotes. Function passed WITHOUT parentheses (or with anonymous function).

---

3. What does event.preventDefault() do?

- A) Stops the event from firing
- B) Deletes the element
- C) Stops the default browser action
- D) Prevents bugs

▶ See Answer (Try first!)

Answer: C - Stops default browser behavior. For forms, prevents page refresh. For links, prevents navigation. For context menu, prevents menu from showing. You take control of what happens!

---

4. Which event fires when user types in an input?

- A) click
- B) submit
- C) input
- D) change

▶ See Answer (Try first!)

Answer: C - "input" event fires with EVERY keystroke as user types. "change" only fires when input loses focus. For live character counting or search suggestions, use "input"!

---

5. What is event.target?

- A) The element you want to change
- B) The element that triggered the event
- C) The parent element
- D) A random element

▶ See Answer (Try first!)

Answer: B - event.target is the specific element that was interacted with. If you have 5 buttons with same handler, event.target tells you WHICH button was clicked. Super useful!

Scoring:

- **5/5:** 🎉 Event Master! You're ready for advanced interactions!
- **4/5:** 👏 Great! Review the one you missed
- **3/5:** 👍 Good! Practice more with events
- ❤️**/5:** 💪 Re-read event listeners section

🎓 Today's Homework (گھر کا کام)

Required (لازمی):

- ☐ Complete the Interactive Counter App
- ☐ Test all features thoroughly
- ☐ Add your own creative feature
- ☐ Show it to your team tomorrow!

Optional (اختیاری):

- ☐ Try the extension challenges
- ☐ Build a "Like Button" (click to toggle, show count)
- ☐ Create a "Color Picker" (buttons change page color)
- ☐ Make a "Simple Calculator" (2 inputs, 4 operation buttons)

For Tomorrow:

- ☐ Think: "How do websites validate forms before submitting?"
- ☐ Tomorrow: Forms & Validation - making sure data is correct!

💭 Daily Reflection (روزانہ کی سوچ)

آج میں نے کیا سیکھا (What I Learned Today):

مشکل کیا لگا (What I Found Difficult):

مزید کیا سیکھنا ہے (What I Want to Explore More):

My Confidence Level (1-10): _______

🔄 Tomorrow's Preview

Tomorrow we'll learn about **Forms & Validation** where you'll build a **Complete Registration Form**!

You'll learn how to:

- Capture user input from forms
- Validate data before submission
- Show error messages dynamically
- Handle different input types
- Build professional forms

Get Ready By:

- ☐ Making sure your Counter works perfectly
- ☐ Thinking: "What makes a good password?"
- ☐ Forms + validation = professional websites!

📚 Resources (اگر مزید پڑھنا ہو)

Free Resources (3G-Friendly):

📖 **MDN - Introduction to Events**

- Link: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Building_blocks/Events
- Best for: Deep understanding of events

📖 **MDN - addEventListener**

- Link: https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener
- Best for: Full syntax and options

🎥 **JavaScript Events in Urdu**

- Search: "JavaScript event handling Urdu tutorial"
- Best for: Visual demonstration

CodeSensei's Tip of the Day: 💡

"Events are what make websites ALIVE. Every interaction you've ever had with a website - clicking, typing, scrolling - is an event. Master events, and you can build anything interactive. Pro tip: Open any website, F12 → Elements tab → Click an element → Event Listeners tab. See all the events attached! Try it on YouTube or Facebook!"

"Events websites کو زندہ کرتے ہیں۔ ہر کلک، ہر ٹائپنگ event ہے۔ Master these, build anything!"

Day 9 Complete! Your pages now RESPOND to users! 🎯

اللہ حافظ! کل ملتے ہیں Forms کے ساتھ! 📝