ScopeTreasure Courses

JAVASCRIPT COURSES · LESSON 3.14 · CODE SCOPE SERIES

Events: click, input, submit

Pages that listen: addEventListener's three parts, click counters, one function for many buttons, live input counters and submit with preventDefault.

Introduction

In the last lesson, your colour buttons only worked because something connected the click to the code. That connection is an **event listener**. An **event** is something that happens on the page — a click, a keystroke, a form being sent — and a listener is an ear you attach to an element, waiting for one of those moments so it can run your function. *(Roman Urdu: Event matlab page par koi amal — click, typing ya form bhejna — aur listener us amal ka intezaar karta hai.)*

Key points

- `addEventListener` **has THREE parts:** on **what** element, for **what** event, **do what** (a function). Example: `button.addEventListener("click", greet);` — the listener is an ear on the button.
- **The golden rule: pass the NAME, do not CALL it.** Write `greet`, not `greet()`. With brackets, the function runs **right now**, once, and the listener ends up listening to its return value instead — the classic beginner bug.

  (Roman Urdu: Sirf function ka naam do, brackets lagaye to woh foran chal jayega aur click par kuch nahi hoga.)
- **The** `click` **event** fires when the user clicks the element — buttons, headings, images, anything can listen for it.
- **The event object tells you who:** when your function runs, JavaScript hands it an event object (often named `e`). `e.target` is the exact element that was clicked — so **one function can serve many buttons**.
- **The** `input` **event fires on every keystroke** in a text field — perfect for live character counters and live validation. Read the current text any time with `input.value`.
- **The** `submit` **event belongs to the form:** a form's normal behaviour is to **reload the page** and send the data away. `e.preventDefault()` stops that reload, so your code can keep the data and use it on the page.

  (Roman Urdu: preventDefault page ko reload honay se rok deta hai, taakeh data aap ke code ke paas reh jaye.)

Code examples

Example 1 — Your first live click: a counter that climbs

```
<!DOCTYPE html>
<html>
<head>
  <title>Click Counter</title>
</head>
<body>
  <h1>Click Counter</h1>
  <p>Count: <span id="count">0</span></p>
  <button id="btn">Click me</button>

  <script>
    let count = 0;
    const btn = document.getElementById("btn");
    const countText = document.getElementById("count");

    function increase() {
      count = count + 1;
      countText.textContent = count;
    }

    btn.addEventListener("click", increase);
  </script>
</body>
</html>
```

What it does: The listener is attached to the button with the function's **name only** — `increase`, no brackets. Every click fires the event, the function runs, `count` climbs by one, and the new number is written into the page with `textContent`. Nothing reloads; the page simply listens and reacts.

What you will see: the page starts at `Count: 0`. Click the button three times and it reads `Count: 3` — each click adds exactly one, as many times as you click.

Example 2 — One function serves three colour buttons

```
<!DOCTYPE html>
<html>
<head>
  <title>One Function, Three Buttons</title>
</head>
<body>
  <h1 id="title">Pick a colour</h1>
  <button class="colourBtn" data-colour="red">Red</button>
  <button class="colourBtn" data-colour="green">Green</button>
  <button class="colourBtn" data-colour="blue">Blue</button>

  <script>
    function paint(e) {
      const colour = e.target.dataset.colour;
      document.body.style.backgroundColor = colour;
      document.getElementById("title").textContent = "Background is now " + colour;
    }

    const buttons = document.querySelectorAll(".colourBtn");
    buttons.forEach((button) => {
      button.addEventListener("click", paint);
    });
  </script>
</body>
</html>
```

What it does: All three buttons share the **same** listener function. When any button is clicked, `e.target` is that exact button, and `e.target.dataset.colour` reads the colour stored in its `data-colour` attribute — so one small function does the work of three. Writing `addEventListener("click", paint())` here would be the classic mistake: `paint` would run once at load time (and crash, because there is no event yet), and clicks would do nothing.

What you will see: click `Green` and the background turns green with the heading `Background is now green`. Click `Red` or `Blue` — the same function repaints the page from whichever button `e.target` reports.

Example 3 — A live character counter and a submit that stays

```
<!DOCTYPE html>
<html>
<head>
  <title>Live Counter and Form</title>
</head>
<body>
  <form id="myForm">
    <input type="text" id="nameBox" placeholder="Your name">
    <p><span id="counter">0</span> / 10</p>
    <button type="submit">Save</button>
  </form>
  <p id="result"></p>

  <script>
    const form = document.getElementById("myForm");
    const nameBox = document.getElementById("nameBox");
    const counter = document.getElementById("counter");
    const result = document.getElementById("result");

    nameBox.addEventListener("input", function () {
      const length = nameBox.value.length;
      counter.textContent = length;
      if (length >= 10) {
        counter.style.color = "red";
      } else {
        counter.style.color = "black";
      }
    });

    form.addEventListener("submit", function (e) {
      e.preventDefault();
      result.textContent = "Saved name: " + nameBox.value;
    });
  </script>
</body>
</html>
```

What it does: Two listeners run this page. The `input` listener fires on **every keystroke**: it counts the characters in the box, updates the counter live, and turns the counter red when the count reaches the limit of 10. The `submit` listener calls `e.preventDefault()` first — the page does **not** reload — and then uses the typed name right there on the page.

What you will see: type `Ahmed` and the counter climbs `1`, `2`, `3`, `4`, `5` as you type. Type ten characters and the counter turns red. Press Save and the line `Saved name:` appears with your typed name — and the

page never reloads.

Quick recap

- `addEventListener` = element + event name + function — and you pass the function's **name**, never `name()`.
- `e.target` tells one shared function exactly which element was used.
- `input` fires per keystroke for live counters; `submit` plus `e.preventDefault()` keeps form data on the page.

Common mistakes

1. **Calling the function instead of passing it** — `addEventListener("click", greet())` runs `greet` once immediately and registers its return value (usually `undefined`) as the listener, so clicks do nothing. The fix: pass the name only — `addEventListener("click", greet)`.
2. **Putting the event name in the wrong form** — listeners take plain names like `"click"`, `"input"` and `"submit"`, not `"onclick"` (that spelling belongs to old HTML attributes). The fix: drop the `on` when you use `addEventListener`.
3. **Forgetting preventDefault on submit** — without it, the form reloads the page the moment it is sent, and everything your code built (messages, results, state) disappears. The fix: make `e.preventDefault()` the first line of every submit handler you write.

Practice

1. Build a page with two buttons, `+` and `-`, and a number shown between them. One shared counter variable, two listeners: the plus button adds 1, the minus button subtracts 1, and the number on the page updates after every click.
2. Build a small form with an email box and a live counter under it that shows how many characters are typed. On submit, stop the reload with `preventDefault` and show `You typed:` plus the email, on the page.

Practice quiz

1. How do you hand a function to addEventListener?

- button.addEventListener("click", greet());
- button.addEventListener("click", greet);, pass the name, do not call it
- button.addEventListener("click", "greet");
- button.addEventListener(greet, "click");

Answer: button.addEventListener("click", greet);, pass the name, do not call it

2. What does e.target tell a shared event handler?

- The page title
- The exact element that was clicked, so one function can serve many buttons
- The mouse position only
- Nothing, it is always null

Answer: The exact element that was clicked, so one function can serve many buttons

3. What happens on form submit if you forget e.preventDefault()?

- Nothing changes
- The form reloads the page and sends the data away, and everything your code built disappears
- The form is validated twice
- The submit button disappears

Answer: The form reloads the page and sends the data away, and everything your code built disappears

---

Watch the video lesson: https://youtu.be/gk8Q30xZJ8c

Read online: https://scopetreasure.com/notes/js-events · More free tools and notes at https://scopetreasure.com