EcomLabs

STEP-BY-STEP GUIDE

An accessible modal with HTML dialog

The complete code for a confirmation modal built with <dialog>, in plain HTML and in React, with the ::backdrop styles, the scroll lock and an accessibility checklist to review it with the keyboard.

FOR	Stores and web apps in HTML, React or Next.js
TESTED WITH	Chromium 151 · Playwright 1.62 · React 19.0.0 · Node 22.22.3
UPDATED	November 3, 2026
DOCS	developer.mozilla.org/en-US/docs/Web/HTML/Reference/Elements/dialog
ARTICLE	ecomlabs.dev/g/dialog-en
AUTHOR	Luis Morales · @luismoralesdev

Contents

Before you start	What you will build
Step 01	The HTML example
Step 02	The React version
Step 03	Styles and scroll
Step 04	Test it with the keyboard
When you are done	Accessibility checklist

BEFORE YOU START

What you will build

A confirmation modal for removing an item from the cart, built with the `<dialog>` element and `showModal()`. The browser moves focus inside, makes the rest of the page inert, closes it on Escape and draws it in the top layer with its `::backdrop`. You add the accessible name, a visible close button, and where focus goes when the button that opened it disappears.

We tested it on November 2, 2026 with Playwright 1.62 in Chromium 151, on Windows 11 with Node 22.22.3. The React version, with React 19.0.0. `<dialog>` has been Baseline since March 2022, per MDN.

STEP 01

The HTML example

Save it as `cart.html` and open it in a browser. It has no dependencies:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Your cart</title>
  <style>
    dialog { max-width: min(28rem, calc(100% - 2rem)); padding: 1.5rem; border: 1px solid #d4d4d4; border-
radius: 12px; }
    dialog::backdrop { background: rgb(0 0 0 / 0.55); }
    html:has(dialog:modal) { overflow: hidden; }
  </style>
</head>
<body>
  <h1 id="cart-title" tabindex="-1">Your cart</h1>
  <ul id="cart">
    <li id="item-1">Runner sneakers, size 9
      <button type="button" class="remove" data-item="item-1">Remove</button>
    </li>
  </ul>

  <dialog id="confirm-remove" aria-labelledby="confirm-title">
    <form method="dialog">
      <h2 id="confirm-title">Remove this item from your cart?</h2>
      <p>You can add it back later.</p>
      <button value="cancel" autofocus>Cancel</button>
      <button value="remove">Remove</button>
    </form>
  </dialog>

  <script>
```

```
    const dialog = document.getElementById("confirm-remove");
    let pending = null;

    document.getElementById("cart").addEventListener("click", (event) => {
      const button = event.target.closest(".remove");
      if (!button) return;
      pending = document.getElementById(button.dataset.item);
      dialog.returnValue = "";
      dialog.showModal();
    });

    dialog.addEventListener("close", () => {
      if (dialog.returnValue === "remove") {
        pending.remove();
        document.getElementById("cart-title").focus();
      }
      pending = null;
    });
  </script>
</body>
</html>
```

- `aria-labelledby` names the dialog after the `h2`; with `method="dialog"`, either button closes it and leaves its `value` in `returnValue`.
- `autofocus` sits on Cancel, the safe choice.
- `dialog.returnValue = ""` before opening: per the standard, Escape does not change `returnValue`, so without the reset a previous `"remove"` could linger.
- Deleting the row removes the button that opened the modal, which is why the script moves focus to the `h1`.

STEP 02

The React version

A ref to the `<dialog>` and the browser's own methods, with no `isOpen` state. In Next.js the file needs `"use client"`:

```
"use client";

import { useId, useRef, useState } from "react";

function RemoveButton({ name, onRemove }: { name: string; onRemove: () => void }) {
  const dialogRef = useRef<HTMLDialogElement>(null);
  const titleId = useId();

  function open() {
    const dialog = dialogRef.current;
    if (!dialog) return;
    dialog.returnValue = "";
    dialog.showModal();
```

```
  }

  function handleClose() {
    if (dialogRef.current?.returnValue === "remove") onRemove();
  }

  return (
    <>
      <button type="button" onClick={open}>Remove</button>
      <dialog ref={dialogRef} aria-labelledby={titleId} onClose={handleClose}>
        <form method="dialog">
          <h2 id={titleId}>Remove {name} from your cart?</h2>
          <p>You can add it back later.</p>
          <button value="cancel">Cancel</button>
          <button value="remove">Remove</button>
        </form>
      </dialog>
    </>
  );
}

export function Cart() {
  const [items, setItems] = useState([{ id: "1", name: "Runner sneakers, size 9" }]);
  const headingRef = useRef<HTMLHeadingElement>(null);

  function remove(id: string) {
    setItems((list) => list.filter((item) => item.id !== id));
    headingRef.current?.focus();
  }

  return (
    <section>
      <h1 ref={headingRef} tabIndex={-1}>Your cart</h1>
      <ul>
        {items.map((item) => (
          <li key={item.id}>
            {item.name} <RemoveButton name={item.name} onRemove={() => remove(item.id)} />
          </li>
        ))}
      </ul>
    </section>
  );
}
```

- `onClose` is the browser's `close` event; that is where `returnValue` is read.
- `useId()` gives each row's title a unique id.
- Cancel comes first, with no `autoFocus`: React focuses on mount, while the dialog is closed, and in our test it did not write the `autofocus` attribute. On open, the browser focuses the first button.
- Do not use `<dialog open={isOpen}>`: the `open` attribute opens a non-modal dialog (in our test, no `:modal`), with no inert page and no Escape.

STEP 03

Styles and scroll

These are the three rules in the `<style>` block from Step 01:

- `::backdrop` only exists when you open with `showModal()`.
- The dialog does not stop page scroll. Without the third rule, with the modal open, the mouse wheel moved the page 800 px and Page Down another 525; with it, 0.
- Do not set `display` on the `dialog` selector: `dialog { display: flex; }` made a closed dialog visible. Use `dialog[open]`.

STEP 04

Test it with the keyboard

- Tab to Remove and press Enter: focus should land on Cancel.
- Press Tab a few times: focus moves through the dialog's buttons and never reaches the page behind.
- Escape: it closes, the row stays and focus is back on Remove. Open it again and choose Remove: focus lands on the heading.
- With a screen reader, opening it should announce the dialog's name, not just "dialog."

> **CAUTION**
>
> Focus does not return "to the button that opened it": it returns to the element that had focus when you called `showModal()`. Open the modal from code and it goes somewhere else; if that element no longer exists, it falls to `body`. And Safari, by design, does not focus a button when you click it (MDN). Decide where focus goes yourself whenever the origin can disappear.

WHEN YOU ARE DONE

Accessibility checklist

- [ ] You open with `showModal()`, not `show()` or the `open` attribute.
- [ ] The dialog has a name: `aria-labelledby` pointing at the visible title, or `aria-label`.
- [ ] There is a visible close button: with an `aria-label` if it is an icon, and `formnovalidate` if there are `required` fields.
- [ ] `autofocus` sits on the control the user needs first.
- [ ] You reset `returnValue` before every open.
- [ ] If the opening button can disappear, you move focus somewhere sensible.
- [ ] The background uses `::backdrop` and does not scroll while the modal is open.
- [ ] No `tabindex` on the `<dialog>` and no `display` on the `dialog` selector.
- [ ] Tested with the keyboard: Tab, Escape, and where focus lands.