EcomLabs

GUÍA PASO A PASO

Modal accesible con dialog de HTML

El código completo de un modal de confirmación con <dialog>, en HTML y en React, con los estilos de ::backdrop, el bloqueo del scroll y una checklist de accesibilidad para revisarlo con teclado.

PARA	Tiendas y apps web en HTML, React o Next.js
PROBADO CON	Chromium 151 · Playwright 1.62 · React 19.0.0 · Node 22.22.3
ACTUALIZADA	3 de noviembre de 2026
DOCUMENTACIÓN	developer.mozilla.org/en-US/docs/Web/HTML/Reference/Elements/dialog
ARTÍCULO	ecomlabs.dev/g/dialog
AUTOR	Luis Morales · @luismoralesdev

Índice

Antes de empezar	Qué vas a montar
Paso 01	El ejemplo en HTML
Paso 02	La versión React
Paso 03	Estilos y scroll
Paso 04	Pruébalo con teclado
Al terminar	Checklist de accesibilidad

ANTES DE EMPEZAR

Qué vas a montar

Un modal de confirmación para quitar un producto del carrito, hecho con el elemento `<dialog>` y `showModal()`. El navegador pone el foco dentro, deja inerte el resto de la página, lo cierra con Escape y lo dibuja en la capa superior con su `::backdrop`. Tú pones el nombre accesible, un botón visible para cerrar y adónde va el foco cuando el botón que lo abrió desaparece.

Lo probamos el 2 de noviembre de 2026 con Playwright 1.62 en Chromium 151, en Windows 11 con Node 22.22.3. La versión React, con React 19.0.0. `<dialog>` es Baseline desde marzo de 2022, según MDN.

PASO 01

El ejemplo en HTML

Guárdalo como `carrito.html` y ábrelo en el navegador. No tiene dependencias:

```
<!doctype html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Tu carrito</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">Tu carrito</h1>
  <ul id="cart">
    <li id="item-1">Tenis Runner, talla 27
      <button type="button" class="remove" data-item="item-1">Quitar</button>
    </li>
  </ul>

  <dialog id="confirm-remove" aria-labelledby="confirm-title">
    <form method="dialog">
      <h2 id="confirm-title">¿Quitar este producto del carrito?</h2>
      <p>Puedes volver a agregarlo después.</p>
      <button value="cancel" autofocus>Cancelar</button>
      <button value="remove">Quitar</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` le da al diálogo el nombre del `h2`; con `method="dialog"`, cada botón lo cierra y deja su `value` en `returnValue`.
- `autofocus` va en Cancelar, la opción segura.
- `dialog.returnValue = ""` antes de abrir: Escape no cambia `returnValue`, según el estándar, y sin el reinicio podría quedar el `"remove"` anterior.
- Al borrar la fila, el botón que abrió el modal desaparece; por eso el script mueve el foco al `h1`.

PASO 02

La versión React

Un ref al `<dialog>` y los métodos del navegador, sin estado `isOpen`. En Next.js, el archivo necesita `"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}>Quitar</button>
      <dialog ref={dialogRef} aria-labelledby={titleId} onClose={handleClose}>
        <form method="dialog">
          <h2 id={titleId}>¿Quitar {name} del carrito?</h2>
          <p>Puedes volver a agregarlo después.</p>
          <button value="cancel">Cancelar</button>
          <button value="remove">Quitar</button>
        </form>
      </dialog>
    </>
  );
}

export function Cart() {
  const [items, setItems] = useState([{ id: "1", name: "Tenis Runner, talla 27" }]);
  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}>Tu carrito</h1>
      <ul>
        {items.map((item) => (
          <li key={item.id}>
            {item.name} <RemoveButton name={item.name} onRemove={() => remove(item.id)} />
          </li>
        ))}
      </ul>
    </section>
  );
}
```

- `onClose` es el evento `close` del navegador; ahí se lee `returnValue`.
- `useId()` da un id único al título de cada fila.
- Cancelar va primero y sin `autoFocus`: React enfoca al montar, con el diálogo cerrado, y en nuestra prueba no escribió el atributo `autofocus`. Al abrir, el navegador enfoca el primer botón.
- No uses `<dialog open={isOpen}>`: el atributo `open` abre un diálogo no modal (en nuestra prueba, sin `:modal`), sin página inerte ni Escape.

PASO 03

Estilos y scroll

Son las tres reglas del `<style>` del Paso 01:

- `::backdrop` solo existe cuando abres con `showModal()`.
- El diálogo no bloquea el scroll. Sin la tercera regla, con el modal abierto, la rueda del mouse movió la página 800 px y Av Pág otros 525; con ella, 0.
- No pongas `display` en el selector `dialog`: `dialog { display: flex; }` dejó visible un diálogo cerrado. Usa `dialog[open]`.

PASO 04

Pruébalo con teclado

- Tab hasta Quitar y pulsa Enter: el foco debe quedar en Cancelar.
- Pulsa Tab varias veces: el foco recorre los botones del diálogo y nunca llega a la página de atrás.
- Escape: se cierra, la fila sigue y el foco vuelve a Quitar. Abre otra vez y elige Quitar: el foco queda en el título.
- Con un lector de pantalla, al abrir debe anunciarse el nombre del diálogo, no solo «diálogo».

> **CUIDADO**
>
> El foco no vuelve «al botón que lo abrió»: vuelve al elemento que tenía el foco cuando llamaste `showModal()`. Si abres el modal por código, vuelve a otro lado; si ese elemento ya no existe, cae en `body`. Y Safari, por diseño, no enfoca un botón al hacerle clic (MDN). Decide tú adónde va el foco cuando el origen puede desaparecer.

AL TERMINAR

Checklist de accesibilidad

- [ ] Abres con `showModal()`, no con `show()` ni con el atributo `open`.
- [ ] El diálogo tiene nombre: `aria-labelledby` al título visible o `aria-label`.
- [ ] Hay un botón visible para cerrar: con `aria-label` si es un icono y `formnovalidate` si hay campos `required`.
- [ ] `autofocus` está en el control que el usuario necesita primero.
- [ ] Reinicias `returnValue` antes de cada apertura.
- [ ] Si el botón de origen puede desaparecer, mueves el foco a un lugar con sentido.
- [ ] El fondo usa `::backdrop` y no se desplaza con el modal abierto.
- [ ] Sin `tabindex` en el `<dialog>` y sin `display` en el selector `dialog`.
- [ ] Probado con teclado: Tab, Escape y adónde vuelve el foco.