EcomLabs

GUÍA PASO A PASO

# Formularios accesibles con WCAG 2.2: seis puntos

Una checklist de seis puntos para que un formulario de checkout cumpla WCAG 2.2 en nivel AA: etiquetas, errores en texto, mensajes que el lector anuncia, foco no tapado, objetivos de 24 px y no pedir dos veces el mismo dato. Cada punto con su criterio y código de ejemplo probado.

| | |
|---|---|
| PARA | Formularios de checkout, registro y contacto en HTML, en cualquier framework |
| PROBADO CON | Playwright 1.62 · Chromium 151 · axe-core 4.12.1 · Node 22.22.3 |
| ACTUALIZADA | 19 de noviembre de 2026 |
| DOCUMENTACIÓN | w3.org/TR/WCAG22 |
| ARTÍCULO | ecomlabs.dev/g/accesible |
| AUTOR | Luis Morales · @luismoralesdev |

# Índice


| | |
|---|---|
| Antes de empezar | Qué cubre esta guía |
| Punto 01 | Una etiqueta visible en cada campo |
| Punto 02 | El error se dice en texto, no solo en rojo |
| Punto 03 | Mensajes que el lector anuncia |
| Punto 04 | Foco visible y no tapado |
| Punto 05 | Objetivos de al menos 24 × 24 px |
| Punto 06 | No pedir lo mismo dos veces |
| Al final | Cómo probarlo |

ANTES DE EMPEZAR

## Qué cubre esta guía

Seis puntos de WCAG 2.2, la Recomendación vigente del W3C (12 de diciembre de 2024), cada uno con su criterio y su nivel. Si te piden AA, cumples los de nivel A y los de AA; 2.4.11, 2.5.8 y 3.3.7 son nuevos en 2.2. El código es un extracto de un checkout de prueba que pasó axe-core 4.12.1 sin violaciones en Chromium 151. Cambia los textos en inglés por los de tu sitio.

PUNTO 01

## Una etiqueta visible en cada campo

**Criterio:** 1.3.1 Info and Relationships (A) y 3.3.2 Labels or Instructions (A). El placeholder no es una etiqueta: suele tener poco contraste y desaparece al escribir.

```
<label for="email">Email (required)</label>
<input id="email" type="email" autocomplete="email" required
       aria-describedby="email-hint email-error"
       data-error="Error: enter an email like name@example.com.">
<p id="email-hint">We send the order confirmation here.</p>
<p id="email-error" class="error"></p>
```

`for` e `id` unen etiqueta y campo, y "obligatorio" va en texto. Comprueba: haz clic en cada etiqueta y mira que el foco vaya a su campo. axe no marca un campo con solo placeholder, porque Chromium lo usa como nombre accesible: revisa las etiquetas a ojo.

PUNTO 02

## El error se dice en texto, no solo en rojo

**Criterio:** 1.4.1 Use of Color (A) y 3.3.1 Error Identification (A). El color acompaña, pero el campo se identifica y el error se describe en texto, con un ejemplo válido.

```
.error { color: #b3261e; font-weight: 600; }
[aria-invalid="true"] { border: 2px solid #b3261e; }
```

Tres señales juntas: el texto "Error: …", el borde más grueso y `aria-invalid="true"`. Comprueba: en DevTools, Rendering, Emulate vision deficiencies, Achromatopsia; debe seguir claro qué campo falla.

PUNTO 03

## Mensajes que el lector anuncia

**Criterio:** 4.1.3 Status Messages (AA). El resumen va en una región `role="status"` (con `aria-live="polite"` implícito) que existe vacía desde que carga la página; el error de cada campo llega por `aria-describedby`.

```
<p id="form-status" role="status"></p>

<script>
const form = document.querySelector("#checkout");
const status = document.querySelector("#form-status");

function check(field) {
  const ok = field.checkValidity();
  field.setAttribute("aria-invalid", String(!ok));
  document.getElementById(field.id + "-error").textContent = ok ? "" : field.dataset.error;
  return ok;
}

form.addEventListener("change", (e) => {
  if (e.target.getAttribute("aria-invalid") === "true") check(e.target);
});

form.addEventListener("submit", (e) => {
  const fields = [...form.elements].filter((f) => f.dataset.error && f.willValidate);
  const bad = fields.filter((f) => !check(f));
  if (!bad.length) return;
  e.preventDefault();
  status.textContent = `${bad.length} field(s) need a fix. The first one has focus.`;
  bad[0].focus();
});
</script>
```

El formulario lleva `novalidate`. El error se vacía al corregirlo, no se oculta: `aria-describedby` lee también el texto oculto. El foco va al primer error. `role="alert"` interrumpe; resérvalo para lo urgente.

---

PUNTO 04

## Foco visible y no tapado

**Criterio:** 2.4.7 Focus Visible (AA) y 2.4.11 Focus Not Obscured (Minimum) (AA). El campo con foco no puede quedar oculto por completo bajo una barra fija o un banner de cookies. Deja espacio con `scroll-padding` (técnica C43) o haz el banner modal.

```
html { scroll-padding-bottom: 5rem; }   /* the fixed bar is 4rem tall */
body { padding-bottom: 5rem; }
:focus-visible { outline: 3px solid #135cfa; outline-offset: 2px; }
.total-bar { position: fixed; inset: auto 0 0; box-sizing: border-box; min-height: 4rem; }
```

Comprueba: ventana de 375 × 360, recorre con Tab y mira cada parada.

> **CUIDADO**
>
> Mide la barra con su relleno. En la prueba, sin `box-sizing: border-box` medía 81px contra 80px de `scroll-padding`, y la casilla quedó un píxel debajo.

PUNTO 05

## Objetivos de al menos 24 × 24 px

**Criterio:** 2.5.8 Target Size (Minimum) (AA). Excepciones: espaciado, equivalente, en línea, control del navegador y esencial. No dependas del espaciado: un enlace nuevo al lado lo rompe.

```
input, button { min-height: 44px; }
.check input { width: 24px; height: 24px; }
```

Para listar los objetivos chicos, pega esto en la consola de DevTools:

```
[...document.querySelectorAll("a, button, input, select, textarea, [role=button]")]
  .filter((e) => e.offsetParent)
  .map((e) => [e, e.getBoundingClientRect()])
  .filter(([, r]) => r.width < 24 || r.height < 24)
  .forEach(([e, r]) => console.log(Math.round(r.width) + "x" + Math.round(r.height), e));
```

En el formulario de prueba con errores listó la casilla (13x13) y el botón Pagar (33x22). Los enlaces dentro de un texto también salen y entran en la excepción "en línea".

PUNTO 06

## No pedir lo mismo dos veces

**Criterio:** 3.3.7 Redundant Entry (A). Lo ya escrito en el mismo proceso se rellena solo o se puede elegir. El autocompletado del navegador no cuenta: lo tiene que ofrecer el sitio.

```
<label class="check"><input id="same" type="checkbox" checked>
  Billing address is the same as shipping</label>

<fieldset id="billing" hidden disabled>
  <legend>Billing address</legend>
  <!-- label, input and error as in point 01, plus data-copy="ship-zip" -->
</fieldset>

<script>
const same = document.querySelector("#same");
const billing = document.querySelector("#billing");

same.addEventListener("change", () => {
  billing.hidden = billing.disabled = same.checked;
  // Unticked: start from the shipping address, not from empty fields.
  for (const f of billing.elements) f.value ||= form.elements[f.dataset.copy].value;
});
</script>
```

Deshabilitado, el `fieldset` no se valida ni se envía. Al desmarcar, cada campo vacío toma el valor de su par de envío. Si la tarjeta falla, no borres lo ya escrito.

AL FINAL

## Cómo probarlo

Pasa axe-core con las reglas de WCAG 2.2 A y AA. En el formulario de prueba con seis errores encontró uno, la casilla sin etiqueta. Lo demás, a mano:

1. Recorre el formulario solo con teclado, en una ventana chica, con la barra o el banner visibles.
2. Envíalo vacío: el foco va al primer error y el resumen aparece en texto.
3. Con un lector de pantalla (NVDA en Windows, VoiceOver en macOS e iOS), escucha la etiqueta, el error y el resumen.
4. Corrige un campo y desmarca "igual que el envío": el error se va y los datos ya están copiados.