EcomLabs

STEP-BY-STEP GUIDE

Accessible forms under WCAG 2.2: six checks

A six-point checklist to get a checkout form to WCAG 2.2 level AA: labels, errors in text, messages screen readers announce, focus that is not obscured, 24 px targets and never asking for the same data twice. Each point with its criterion and tested example code.

FOR	Checkout, sign-up and contact forms in HTML, in any framework
TESTED WITH	Playwright 1.62 · Chromium 151 · axe-core 4.12.1 · Node 22.22.3
UPDATED	November 19, 2026
DOCS	w3.org/TR/WCAG22
ARTICLE	ecomlabs.dev/g/accesible-en
AUTHOR	Luis Morales · @luismoralesdev

Contents

Before you start	What this guide covers
Check 01	A visible label on every field
Check 02	Errors in text, not just red
Check 03	Messages a screen reader announces
Check 04	Focus visible and not obscured
Check 05	Targets of at least 24 by 24 px
Check 06	Never ask for the same thing twice
Last	How to test it

BEFORE YOU START

What this guide covers

Six checks from WCAG 2.2, the current W3C Recommendation (12 December 2024), each with its criterion and level. If you are asked for AA, you meet every A and every AA criterion; 2.4.11, 2.5.8 and 3.3.7 are new in 2.2. The code is taken from a test checkout that passed axe-core 4.12.1 with no violations in Chromium 151.

CHECK 01

A visible label on every field

Criterion: 1.3.1 Info and Relationships (A) and 3.3.2 Labels or Instructions (A). A placeholder is not a label: it is usually low contrast and disappears as soon as someone types.

```
<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>
```

Matching `for` and `id` tie label and field, and "required" is written in text. Check: click each label and confirm focus lands in its field. axe does not flag a placeholder-only field, because Chromium uses it as the accessible name: review labels by eye.

CHECK 02

Errors in text, not just red

Criterion: 1.4.1 Use of Color (A) and 3.3.1 Error Identification (A). Color can support the message, but the field is identified and the error described in text, with a valid example.

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

Three signals together: the text "Error: …", a thicker border and `aria-invalid="true"`. Check: in DevTools, Rendering, Emulate vision deficiencies, Achromatopsia; it should still be clear which field failed.

CHECK 03

Messages a screen reader announces

Criterion: 4.1.3 Status Messages (AA). The summary goes in a `role="status"` region (implicit `aria-live="polite"`) that exists, empty, from page load; each field's error arrives through `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>
```

The form carries `novalidate`. The error is emptied once fixed, not hidden: `aria-describedby` reads hidden text too. Focus goes to the first error. `role="alert"` interrupts; keep it for urgent messages.

CHECK 04

Focus visible and not obscured

Criterion: 2.4.7 Focus Visible (AA) and 2.4.11 Focus Not Obscured (Minimum) (AA). The focused field must not be entirely hidden under a fixed bar or a cookie banner. Leave room with `scroll-padding` (technique C43) or make the 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; }
```

Check: a 375 by 360 window, Tab through the form and look at every stop.

> **CAUTION**
> Measure the bar with its padding. In our test, without `box-sizing: border-box` it was 81px against 80px of `scroll-padding`, and the checkbox ended one pixel under it.

CHECK 05

Targets of at least 24 by 24 px

Criterion: 2.5.8 Target Size (Minimum) (AA). Exceptions: spacing, equivalent, inline, user agent control and essential. Do not lean on spacing: a new link next to the control breaks it.

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

To list small targets, paste this into the DevTools console:

```
[...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));
```

On the broken test form it listed the checkbox (13x13) and the Pay button (33x22). Links inside a sentence show up too and fall under the inline exception.

CHECK 06

Never ask for the same thing twice

Criterion: 3.3.7 Redundant Entry (A). Data already given in the same process is filled in or available to select. Browser autocomplete does not count: the site has to offer it.

```
<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>
```

While disabled, the `fieldset` is neither validated nor submitted. When unticked, each empty field takes its shipping twin's value. If a card is declined, do not wipe what was typed.

LAST

How to test it

Run axe-core with the WCAG 2.2 A and AA rules. On the test form with six mistakes it found one, the unlabeled checkbox. Test the rest by hand:

1. Go through the form with the keyboard only, in a short window, with the bar or banner showing.
2. Submit it empty: focus moves to the first error and the summary appears in text.
3. With a screen reader (NVDA on Windows, VoiceOver on macOS and iOS), listen for the label, the error and the summary.
4. Fix a field and untick "same as shipping": the error goes away and the data is already copied.