CURB CUT PROJECT

Website accessibility report

Automated WCAG 2.2 Level A and AA evaluation with a manual testing plan

Organization	Sample Organization (Curb Cut Project test page)
Page evaluated	https://curbcutproject.org/test-fixtures/broken.html
Assessment date	November 17, 2026
Prepared by	The Curb Cut Project, a free accessibility review service for nonprofits

Executive summary

The automated test found 11 failed rules affecting 14 elements on this page, including 6 critical and 5 serious issues that can block people from using it.

Results at a glance

Pages evaluated	1
Rules run (wcag2a, wcag2aa, wcag21a, wcag21aa, wcag22aa)	69
Rules passed	12
Rules that did not apply to this page	43
Rules violated (distinct problems)	11
Affected elements (individual places on the page)	14
Rules needing human review	3

Violations by severity

Severity	Rules violated	Affected elements
Critical	6	8
Serious	5	6
Moderate	0	0
Minor	0	0

A rule is one kind of problem (for example, images without a text description). Affected elements counts each place on the page where that problem occurs. Severity comes from the test engine: critical and serious issues can block people from using the page; moderate and minor issues make it harder.

How this page was tested

- Test engine: axe-core 4.14.1, running the WCAG 2.0, 2.1 and 2.2 Level A and AA rule sets (wcag2a, wcag2aa, wcag21a, wcag21aa, wcag22aa).
- Browser: Chromium 152.0.7977.64, 1280×800 viewport.
- Scope: one page, without logging in, typing or submitting forms. After loading it, the scanner pressed Tab to move through the page and opened collapsed menus and sections (only buttons marked as expandable and <details> elements, never links, inputs or anything in a form).

- Curb Cut Project checks, run in addition to axe-core:
  - Phone-size re-test: Re-ran axe-core at 390×844 with phone and touch emulation.
  - Keyboard walk: Pressed Tab 10 times; 8 elements received focus; focus kept cycling through 2 elements.
  - Collapsed menus and sections: Pressed 1 collapsed menu or section button (never links, inputs or anything in a form), then re-ran axe-core on what appeared.
  - Reflow at 320px: At 320px wide the page is 650px wide and scrolls 330px sideways.

Limitations

> Automated testing finds only some accessibility problems; many WCAG success criteria cannot be checked by software at all. This report is not a complete accessibility audit, and it is not a certification of conformance with WCAG, the ADA, Section 508 or any other law or standard. Treat it as a starting point: the manual testing checklist below covers what a person still needs to check.

> The Curb Cut Project checks (rule ids starting with ccp-) are automated too. The reflow and keyboard-trap checks measure the page directly; the focus-visibility and unreachable-element checks are heuristics, so they are listed under human review. Passing them does not mean the page works for keyboard users or at 400% zoom: a person still needs to check both.

Detailed findings

11 rules failed, sorted by severity. The website platform is not on record, so fix instructions are general.

Critical Keyboard focus gets stuck

Rule	`ccp-keyboard-trap` (Curb Cut Project check, not an axe-core rule)
Page	https://curbcutproject.org/test-fixtures/broken.html
Affected elements	2
Found at	Desktop (1280×800)
WCAG criteria	2.1.2 No Keyboard Trap (Level A)

What this means

When the Curb Cut Project moved through the page with the Tab key, focus got caught on a small group of elements and could not move on to the rest of the page. If Shift+Tab and Escape did not help either, the page traps keyboard users. This is a Curb Cut Project check, not an axe-core rule.

Who it affects

People who use a keyboard, switch device or screen reader instead of a mouse. Once trapped they cannot reach the rest of the page and often have to reload it or give up.

Recommended fix

Find the script that handles the Tab key on the listed elements and let Tab and Shift+Tab move focus out as usual. Widgets that keep focus on purpose (an open dialog) must close with Escape and with a visible close button, then return focus to the page.

Example for a site built with custom HTML:

Before

```
el.addEventListener("keydown", (e) => { if (e.key === "Tab") { e.preventDefault();
other.focus(); } });
```

After

```
// Let the browser move focus with Tab; only manage focus inside an open dialog, and close it on
Escape.
```

Technical reference: W3C Understanding page for 2.1.2 No Keyboard Trap

Evidence

1. **Element:** `#trap-a`

   HTML:

   ```
   <button type="button" id="trap-a">Option A</button>
   ```

   Evidence:

   ```
   Tab kept cycling through 2 elements and never reached the rest of the page. Shift+Tab and Escape did not move focus out either.
   ```

2. **Element:** `#trap-b`

   HTML:

   ```
   <button type="button" id="trap-b">Option B</button>
   ```

   Evidence:

   ```
   Tab kept cycling through 2 elements and never reached the rest of the page. Shift+Tab and Escape did not move focus out either.
   ```

Critical Images without a text description

Rule	`image-alt`
Page	https://curbcutproject.org/test-fixtures/broken.html
Affected elements	2
Found at	Desktop (1280×800) and Phone (390×844, touch)
WCAG criteria	1.1.1 Non-text Content (Level A)

What this means

An image has no alt text, so assistive technology has nothing to say about it, or reads out the file name instead.

Who it affects

Blind and low-vision visitors using screen readers, and anyone whose images fail to load.

Recommended fix

Give every meaningful image alt text describing what it shows or does, in a short phrase. Give purely decorative images empty alt text (alt="") so screen readers skip them.

Example for a site built with custom HTML:

Before

```
<img src="volunteers.jpg">
```

After

```
<img src="volunteers.jpg" alt="Volunteers sorting cans at the Saturday food drive">
```

Technical reference: axe-core rule image-alt

Evidence

1. **Element:** `img[width="120"]`

   HTML:

   ```
   <img src="data:image/svg+xml,%3Csvg xmlns='http://www.w3.org/2000/svg' width='120' height='60'%3E%3Crect width='120' height='60' fill='%23e0c060'/%3E%3C/svg%3E" width="120" height="60">
   ```

   Evidence:

   ```
   Fix any of the following:
     Element does not have an alt attribute
     aria-label attribute does not exist or is empty
     aria-labelledby attribute does not exist, references elements that do not exist or references elements that are empty
     Element has no title attribute
     Element's default semantics were not overridden with role="none" or role="presentation"
   ```

   Found at: Desktop (1280×800) and Phone (390×844, touch)

2. **Element:** `img[width="80"]`

   HTML:

   ```
   <img src="data:image/svg+xml,%3Csvg xmlns='http://www.w3.org/2000/svg' width='80' height='40'%3E%3Crect width='80' height='40' fill='%2360a0e0'/%3E%3C/svg%3E" width="80" height="40">
   ```

   Evidence:

   ```
   Found after expanding a collapsed menu or section.
   Fix any of the following:
     Element does not have an alt attribute
     aria-label attribute does not exist or is empty
     aria-labelledby attribute does not exist, references elements that do not exist or references elements that are empty
     Element has no title attribute
     Element's default semantics were not overridden with role="none" or role="presentation"
   ```

   Found at: Desktop (1280×800) and Phone (390×844, touch)

Critical Invalid ARIA attribute values

Rule	`aria-valid-attr-value`
Page	https://curbcutproject.org/test-fixtures/broken.html
Affected elements	1
Found at	Desktop (1280×800) and Phone (390×844, touch)
WCAG criteria	4.1.2 Name, Role, Value (Level A)

What this means

An element uses an ARIA attribute with a value that is not allowed, such as aria-hidden="maybe" or an aria-labelledby pointing to an id that does not exist.

Who it affects

Screen reader users receive broken or contradictory information about the element.

Recommended fix

Correct the value to one the attribute allows (for example aria-hidden="true" or "false"), or make sure referenced ids exist on the page.

Example for a site built with custom HTML:

Before

```
<div aria-hidden="maybe">Decorative divider</div>
```

After

```
<div aria-hidden="true">Decorative divider</div>
```

Technical reference: axe-core rule aria-valid-attr-value

Evidence

1. **Element:** `div[aria-hidden="maybe"]`

   HTML:

   ```
   <div aria-hidden="maybe">Decorative divider</div>
   ```

   Evidence:

   ```
   Fix all of the following:
     Invalid ARIA attribute value: aria-hidden="maybe"
   ```

   Found at: Desktop (1280×800) and Phone (390×844, touch)

Critical Buttons with no readable name

Rule	`button-name`
Page	https://curbcutproject.org/test-fixtures/broken.html
Affected elements	1
Found at	Desktop (1280×800) and Phone (390×844, touch)
WCAG criteria	4.1.2 Name, Role, Value (Level A)

What this means

A button has no text that assistive technology can read, usually an icon-only button such as a search, close or menu button.

Who it affects

Screen reader users hear only "button" and cannot tell what it does; voice-control users cannot activate it by name.

Recommended fix

Give the button visible text, or an aria-label describing its action (for example "Search", "Close", "Open menu").

Example for a site built with custom HTML:

Before

```
<button type="submit"><svg aria-hidden="true">…</svg></button>
```

After

```
<button type="submit" aria-label="Search"><svg aria-hidden="true">…</svg></button>
```

Technical reference: axe-core rule button-name

Evidence

1. **Element:** `.icon-btn`

   HTML:

   ```
   <button type="submit" class="icon-btn"><svg width="16" height="16" aria-hidden="true">
   <circle cx="8" cy="8" r="6"></circle></svg></button>
   ```

   Evidence:

   ```
   Fix any of the following:
     Element does not have inner text that is visible to screen readers
     aria-label attribute does not exist or is empty
     aria-labelledby attribute does not exist, references elements that do not
   exist or references elements that are empty
     Element has no title attribute
     Element does not have an implicit (wrapped) <label>
     Element does not have an explicit <label>
     Element's default semantics were not overridden with role="none" or
   role="presentation"
   ```

   Found at: Desktop (1280×800) and Phone (390×844, touch)

Critical Form fields without a label

Rule	`label`
Page	https://curbcutproject.org/test-fixtures/broken.html
Affected elements	1
Found at	Desktop (1280×800) and Phone (390×844, touch)
WCAG criteria	4.1.2 Name, Role, Value (Level A)

What this means

A text box, checkbox or other form field has no label that assistive technology can connect to it.

Who it affects

Screen reader and voice-control users cannot tell what the field is for, and everyone loses the larger click target a label provides.

Recommended fix

Give every field a visible label that is programmatically tied to it. Placeholder text alone is not a substitute.

Example for a site built with custom HTML:

Before

```
<input type="text" name="volunteer-name">
```

After

```
<label for="vname">Your name</label>
<input type="text" id="vname" name="volunteer-name">
```

Technical reference: axe-core rule label

Evidence

1. **Element:** `input`

   HTML:

   ```
   <input type="text" name="volunteer-name">
   ```

   Evidence:

   ```
   Fix any of the following:
     Element does not have an implicit (wrapped) <label>
     Element does not have an explicit <label>
     aria-label attribute does not exist or is empty
     aria-labelledby attribute does not exist, references elements that do not
   exist or references elements that are empty
     Element has no title attribute
     Element has no placeholder attribute
     Element's default semantics were not overridden with role="none" or
   role="presentation"
   ```

   Found at: Desktop (1280×800) and Phone (390×844, touch)

Critical Dropdowns without a label

Rule	`select-name`
Page	https://curbcutproject.org/test-fixtures/broken.html
Affected elements	1
Found at	Desktop (1280×800) and Phone (390×844, touch)
WCAG criteria	4.1.2 Name, Role, Value (Level A)

What this means

A dropdown menu (select) has no label that assistive technology can connect to it.

Who it affects

Screen reader users hear the selected option but not what the choice is for.

Recommended fix

Give the dropdown a visible label tied to it, the same way as other form fields.

Example for a site built with custom HTML:

Before

```
<select name="shift">…</select>
```

After

```
<label for="shift">Preferred shift</label>
<select id="shift" name="shift">…</select>
```

Technical reference: axe-core rule select-name

Evidence

1. **Element:** `select`

   HTML:

   ```
   <select name="shift">
         <option>Tuesday morning</option>
         <option>Saturday morning</option>
       </select>
   ```

   Evidence:

   ```
   Fix any of the following:
     Element does not have an implicit (wrapped) <label>
     Element does not have an explicit <label>
     aria-label attribute does not exist or is empty
     aria-labelledby attribute does not exist, references elements that do not
   exist or references elements that are empty
     Element has no title attribute
     Element's default semantics were not overridden with role="none" or
   role="presentation"
   ```

   Found at: Desktop (1280×800) and Phone (390×844, touch)

Serious Touch targets that are too small

Rule	`target-size`
Page	https://curbcutproject.org/test-fixtures/broken.html
Affected elements	2
Found at	Phone (390×844, touch)
WCAG criteria	2.5.8 Target Size (Minimum) (Level AA)

What this means

A link or button is smaller than 24 by 24 pixels and too close to other targets.

Who it affects

People with tremors, limited dexterity or large fingers, especially on phones, tap the wrong thing or miss it.

Recommended fix

Make the control at least 24 by 24 pixels (44 by 44 is better), or add space around it so neighboring targets do not overlap that area.

Example for a site built with custom HTML:

Before

```
.social a { width: 16px; height: 16px; }
```

After

```
.social a { min-width: 24px; min-height: 24px; padding: 4px; }
```

Technical reference: axe-core rule target-size

Evidence

1. **Element:** `button[aria-label="Share by email"]`

   HTML:

   ```
   <button type="button" aria-label="Share by email"><svg width="8" height="8" aria-hidden="true"><circle cx="4" cy="4" r="3"></circle></svg></button>
   ```

   Evidence:

   ```
   Fix any of the following:
     Target has insufficient size (16px by 16px, should be at least 24px by 24px)
     Target has insufficient space to its closest neighbors. Safe clickable space has a diameter of 8px instead of at least 24px.
   ```

2. **Element:** `button[aria-label="Share by text message"]`

   HTML:

   ```
   <button type="button" aria-label="Share by text message"><svg width="8" height="8" aria-hidden="true"><rect width="8" height="8"></rect></svg></button>
   ```

   Evidence:

   ```
   Fix any of the following:
     Target has insufficient size (16px by 16px, should be at least 24px by 24px)
     Target has insufficient space to its closest neighbors. Safe clickable space has a diameter of 8px instead of at least 24px.
   ```

Serious Page scrolls sideways on a narrow screen

Rule	`ccp-reflow` (Curb Cut Project check, not an axe-core rule)
Page	https://curbcutproject.org/test-fixtures/broken.html
Affected elements	1
Found at	320px wide (like 400% zoom)
WCAG criteria	1.4.10 Reflow (Level AA)

What this means

At 320 pixels wide (a small phone, or a desktop browser zoomed to 400%) part of the page is wider than the screen, so people have to scroll left and right to read it. The Curb Cut Project measured this directly; it is not an axe-core rule.

Who it affects

People with low vision who zoom in, and anyone on a small phone. Scrolling in two directions to read each line is slow and easy to get lost in.

Recommended fix

Find the element listed below and let it shrink with the screen: replace fixed widths with max-width: 100%, let rows of boxes wrap onto new lines, and make images and embeds scale. Data tables, maps and code may scroll sideways inside their own box instead of making the whole page scroll.

Example for a site built with custom HTML:

Before

```
<div style="width:600px">Holiday hours: closed November 27 and 28.</div>
```

After

```
<div style="max-width:600px">Holiday hours: closed November 27 and 28.</div>
```

Technical reference: W3C Understanding page for 1.4.10 Reflow

Evidence

1. **Element:** `body > div:nth-of-type(2)`

   HTML:

   ```
   <div class="wide-banner">Holiday hours: closed November 27 and 28.</div>
   ```

   Evidence:

   ```
   At 320px wide the page is 650px wide and scrolls 330px sideways. This element is 618px wide and reaches 650px from the left edge.
   ```

Serious Text with too little color contrast

Rule	`color-contrast`
Page	https://curbcutproject.org/test-fixtures/broken.html
Affected elements	1
Found at	Desktop (1280×800) and Phone (390×844, touch)
WCAG criteria	1.4.3 Contrast (Minimum) (Level AA)

What this means

Some text is too close in color to its background to read comfortably. Normal text needs a contrast ratio of at least 4.5:1, and large text at least 3:1.

Who it affects

People with low vision or color blindness, older visitors, and anyone reading a phone in bright sunlight.

Recommended fix

Darken the text or lighten the background (or the reverse) until the pair passes 4.5:1. Check with a contrast checker before changing your brand colors site-wide.

Example for a site built with custom HTML:

Before

```
<p style="color:#b5b5b5">Open Tuesdays, 9am to noon.</p>
```

After

```
<p style="color:#595959">Open Tuesdays, 9am to noon.</p>
```

Technical reference: axe-core rule color-contrast

Evidence

1. **Element:** `.faint`

   HTML:

   ```
   <p class="faint">Open Tuesdays and Saturdays, 9am to noon.</p>
   ```

   Evidence:

   ```
   Fix any of the following:
     Element has insufficient color contrast of 2.05 (foreground color: #b5b5b5,
   background color: #ffffff, font size: 12.0pt (16px), font weight: normal).
   Expected contrast ratio of 4.5:1
   ```

   Found at: Desktop (1280×800) and Phone (390×844, touch)

Serious Page language not declared

Rule	`html-has-lang`
Page	https://curbcutproject.org/test-fixtures/broken.html
Affected elements	1
Found at	Desktop (1280×800) and Phone (390×844, touch)
WCAG criteria	3.1.1 Language of Page (Level A)

What this means

The page does not say what language it is written in (the lang attribute on the <html> tag is missing).

Who it affects

Screen reader users may hear the page read with the wrong voice and pronunciation.

Recommended fix

Add a lang attribute to the <html> tag with the page's main language, such as lang="en" or lang="es".

Example for a site built with custom HTML:

Before

```
<html>
```

After

```
<html lang="en">
```

Technical reference: axe-core rule html-has-lang

Evidence

1. **Element:** `html`

   HTML:

   ```
   <html>
   ```

   Evidence:

   ```
   Fix any of the following:
     The <html> element does not have a lang attribute
   ```

   Found at: Desktop (1280×800) and Phone (390×844, touch)

Serious Links with no readable name

Rule	`link-name`
Page	https://curbcutproject.org/test-fixtures/broken.html
Affected elements	1
Found at	Desktop (1280×800) and Phone (390×844, touch)
WCAG criteria	2.4.4 Link Purpose (In Context) (Level A); 4.1.2 Name, Role, Value (Level A)

What this means

A link has no text that assistive technology can read, usually because it is an icon or image with no label.

Who it affects

Screen reader users hear only "link" and cannot tell where it goes; voice-control users cannot say its name to click it.

Recommended fix

Give the link visible text, or if it is an icon, give the icon alt text or the link an aria-label that says where it goes (for example "Donate" or "Facebook").

Example for a site built with custom HTML:

Before

```
<a href="/donate"><svg aria-hidden="true">…</svg></a>
```

After

```
<a href="/donate" aria-label="Donate"><svg aria-hidden="true">…</svg></a>
```

Technical reference: axe-core rule link-name

Evidence

1. **Element:** a

   HTML:

   ```
   <a href="https://example.org/donate"><svg width="16" height="16" aria-hidden="true"><rect width="16" height="16"></rect></svg></a>
   ```

   Evidence:

   ```
   Fix all of the following:
     Element is in tab order and does not have accessible text

   Fix any of the following:
     Element does not have text that is visible to screen readers
     aria-label attribute does not exist or is empty
     aria-labelledby attribute does not exist, references elements that do not exist or references elements that are empty
     Element has no title attribute
   ```

   Found at: Desktop (1280×800) and Phone (390×844, touch)

Needs human review

The test engine could not decide these on its own (for example, text over a background image whose contrast it cannot measure), or a Curb Cut Project heuristic flagged them (for example, no visible change when an element receives keyboard focus). They are not counted as failures. A person should check each one.

- ### Keyboard focus may not be visible

  `ccp-focus-visible` (Curb Cut Project check), 1 element to check. WCAG: 2.4.7 Focus Visible (Level AA)

  When this element received keyboard focus it looked exactly the same as before: no outline, shadow, border, color or underline appeared. Usually that means the focus outline was removed in the site's styles. This is a Curb Cut Project check, and a heuristic: focus shown some other way (an icon, a pseudo-element) is not detected, so a person should confirm it.

1. **Element:** `body > p:nth-of-type(3) > button`

   HTML:

   ```
   <button type="button" class="no-ring">Print this page</button>
   ```

   Evidence:

   ```
   With keyboard focus this element looked the same as without it: no outline, shadow, border, color or underline change was detected. Tab to it and check you can see where focus is.
   ```

- ## Clickable element the keyboard cannot reach

  `ccp-keyboard-unreachable` (Curb Cut Project check), 1 element to check. WCAG: 2.1.1 Keyboard (Level A)

  This element looks clickable (it has a click handler, a pointer cursor or a button-like role) but it is a plain element such as a div or span, so Tab never reaches it and Enter cannot activate it. This is a Curb Cut Project check, and a heuristic: some of these are decorative, so a person should confirm each one.

  1. **Element:** `body > div:nth-of-type(3)`

     HTML:

     ```
     <div class="card" onclick="this.textContent = 'Menu: soup, bread, apples'">See this week's menu</div>
     ```

     Evidence:

     ```
     This <div> looks clickable (it has a click handler) but is not a link, button or form control and has no tabindex, so Tab never reaches it.
     ```

- ## Text with too little color contrast

  `color-contrast`, 1 element to check. WCAG: 1.4.3 Contrast (Minimum) (Level AA)

  Some text is too close in color to its background to read comfortably. Normal text needs a contrast ratio of at least 4.5:1, and large text at least 3:1.

1. **Element:** `.wide-banner`

   HTML:

   ```
   <div class="wide-banner">Holiday hours: closed November 27 and 28.</div>
   ```

   Evidence:

   ```
   Fix any of the following:
     Element's background color could not be determined because it's partially
   obscured by another element
   ```

Manual testing checklist

These checks need a person. They cover the WCAG criteria where automated testing helps least and nonprofit sites most often have problems. Each step can be done by a volunteer with a laptop, a phone and a free screen reader (NVDA on Windows, or VoiceOver, built into Macs and iPhones).

Keyboard-only navigation

Many people cannot use a mouse.

2.1.1 Keyboard (Level A)

- [ ] Put the mouse away. Press Tab to move forward and Shift+Tab to move back through the whole page.
- [ ] Check that you can reach and use every link, button, menu, form field, slider and video control; use Enter, Space and the arrow keys to operate them.
- [ ] Open dropdown menus and pop-ups with the keyboard and check that you can reach the items inside.

2.1.2 No Keyboard Trap (Level A)

- [ ] Tab through the whole page, including embedded maps, videos, chat widgets and pop-ups.
- [ ] Check that you can always Tab or Shift+Tab out of each one, or press Escape to close it. If you get stuck, note where.

2.4.3 Focus Order (Level A)

- [ ] Press Tab repeatedly from the top of the page and watch where focus goes.
- [ ] Check that it follows the visual reading order (left to right, top to bottom) and does not jump around unexpectedly.

- [ ] Open a pop-up or menu and check that focus moves into it, and returns to a sensible place when it closes.

2.4.1 Bypass Blocks (Level A)

- [ ] Load the page and press Tab once. A "Skip to content" link should appear; press Enter and check that focus jumps past the menu.
- [ ] If there is no skip link, use a screen reader's landmarks list (NVDA: Insert+F7, then Landmarks) and check that a main landmark exists.

Visible focus

Keyboard users need to see where they are.

2.4.7 Focus Visible (Level AA)

- [ ] Press Tab through the page and watch the screen.
- [ ] Check that you can always see which link, button or field is focused (an outline, underline or color change). If focus seems to disappear, note which element.

2.4.11 Focus Not Obscured (Minimum) (Level AA)

- [ ] With any cookie banner left open, press Tab through the whole page.
- [ ] Check that the focused element is never completely covered by a sticky header, footer, cookie banner or chat widget.

Meaningful alt text

Software can tell when alt text is missing, not whether it is accurate.

1.1.1 Non-text Content (Level A)

- [ ] For each image, ask: if it disappeared, what would a reader miss? Check that its alt text says that (in Chrome, right-click the image, choose Inspect, and read the alt attribute).
- [ ] Check that decorative images (borders, background flourishes) have empty alt text (alt="") so screen readers skip them.
- [ ] Turn on a screen reader (NVDA on Windows, VoiceOver on Mac with Cmd+F5) and move through the page; every image and icon button should be announced with a meaningful name, never a file name like IMG_2041.jpg.

Screen reader checks

Names, roles and structure must make sense when heard, not seen.

4.1.2 Name, Role, Value (Level A)

- [ ] Turn on a screen reader and Tab through every button, link and form control.
- [ ] Check that each is announced with a name and a role ("Donate, button"; "Email, edit text").
- [ ] Open menus, accordions and tabs and check that the state is announced ("expanded", "collapsed", "selected").

1.3.1 Info and Relationships (Level A)

- [ ] Install the free HeadingsMap browser extension, or use a screen reader's headings list (NVDA: Insert+F7; VoiceOver: Ctrl+Option+U), and check that every visual heading appears as a real heading at a sensible level.
- [ ] Check that things that look like lists are announced as lists, and that data tables announce their column headers as you move between cells.
- [ ] Check that groups of related form controls (like radio buttons) have a group label that is read out.

2.4.4 Link Purpose (In Context) (Level A)

- [ ] Open a screen reader's links list (NVDA: Insert+F7; VoiceOver: Ctrl+Option+U, then the Links menu).
- [ ] Check that each link makes sense, and flag vague ones like "click here" or "read more" that are not clarified by the sentence around them.
- [ ] Check that icon-only links (social media, logo) are announced with a name.

Form usability

Contact, donation and volunteer forms must be usable by everyone.

3.3.2 Labels or Instructions (Level A)

- [ ] Look at each form field and check that a visible label stays on screen while you type (placeholder text that disappears is not enough).
- [ ] Check that required fields and formats (such as a date format) are explained before the visitor submits.

3.3.1 Error Identification (Level A)

- [ ] Submit each form empty, then with an invalid email address.
- [ ] Check that each error says in words which field is wrong and why, not only a red outline.
- [ ] With a screen reader on, check that the error is announced or easy to find.

3.3.3 Error Suggestion (Level AA)

- [ ] Enter clearly wrong values, such as an email without @ or a phone number with letters, and submit.
- [ ] Check that the error message says how to fix it, such as "Enter an email address like name@example.org".

1.3.5 Identify Input Purpose (Level AA)

- [ ] Open each form that asks about the visitor (contact, donation, volunteer sign-up).
- [ ] Click into the name, email, phone and address fields. The browser should offer to autofill them with your saved details.
- [ ] If a field does not offer autofill, note it; the fix is an autocomplete attribute such as autocomplete="email".

Reading order and headings

Content must make sense in the order assistive technology reads it.

1.3.2 Meaningful Sequence (Level A)

- [ ] Turn on a screen reader and read the page from the top with the down arrow (NVDA) or Ctrl+Option+Right Arrow (VoiceOver).
- [ ] Check that content is read in the same order you would read it on screen, especially multi-column layouts and side-by-side cards.

2.4.6 Headings and Labels (Level AA)

- [ ] Read only the headings on the page (use a headings list). Check that they give a clear outline of the content.
- [ ] Read each form label on its own and check that it says what to enter, such as "Email address" rather than "Field 2".

Mobile and 400% zoom

People with low vision zoom in; most visitors use phones.

1.4.10 Reflow (Level AA)

- [ ] On a desktop browser window about 1280 pixels wide, zoom to 400% (press Ctrl and + repeatedly).
- [ ] Read down the page. You should only need to scroll up and down, not sideways, except for things like maps, data tables or diagrams.

- [ ] Check that nothing is cut off and the menu can still be opened and used.

1.4.4 Resize Text (Level AA)

- [ ] On a desktop browser, press Ctrl and + (Cmd and + on Mac) until the zoom shows 200%.
- [ ] Check that no text is cut off, overlapping or hidden, and that menus and forms still work.
- [ ] On a phone, pinch to zoom; the page should let you zoom in.

2.5.8 Target Size (Minimum) (Level AA)

- [ ] On a phone, try tapping small icons, social media links, close buttons and pagination numbers.
- [ ] Check that you can hit each one reliably without accidentally tapping its neighbor.

1.3.4 Orientation (Level AA)

- [ ] Open the page on a phone or tablet with rotation lock off.
- [ ] Rotate between portrait and landscape and check that the content rotates with the device and stays usable both ways.

Recommended next steps

1. **Fix the critical and serious issues first.** These can stop people from using the page at all:
    1. Keyboard focus gets stuck (critical, 2 elements)
    2. Images without a text description (critical, 2 elements)
    3. Invalid ARIA attribute values (critical, 1 element)
    4. Buttons with no readable name (critical, 1 element)
    5. Form fields without a label (critical, 1 element)
    6. Dropdowns without a label (critical, 1 element)
    7. Touch targets that are too small (serious, 2 elements)
    8. Page scrolls sideways on a narrow screen (serious, 1 element)
    9. Text with too little color contrast (serious, 1 element)
    10. Page language not declared (serious, 1 element)
    11. Links with no readable name (serious, 1 element)
2. **Review the 3 items that need human judgment** and treat any confirmed problem as a fix to schedule.
3. **Work through the manual testing checklist**, starting with keyboard-only navigation and screen reader checks, and note anything that fails.
4. **Check fixes in templates, not just this page.** Problems in a header, footer, menu or theme repeat on every page; fixing them once fixes them everywhere.

Follow-up evaluation plan

- After fixes are made, the Curb Cut Project re-scans this page with the same rule set and test engine so the before and after results are comparable.
- A fix counts as resolved only after a person confirms it by hand. A later scan no longer detecting an issue is not enough on its own.
- This report covers one page. Evaluate the other pages visitors rely on most, such as donate, contact, volunteer, events and any forms.

Appendix: WCAG 2.2 coverage

WCAG 2.2 has 55 success criteria at Levels A and AA (4.1.1 Parsing was removed in WCAG 2.2). The automated test can partially check 23 of them; the other 32 can only be checked by a person. The Curb Cut Project's own checks also partially looked at 1.4.10, 2.1.1, 2.1.2, 2.4.3, 2.4.7.

Success criteria and what this evaluation found

Success criterion	Level	Result from this scan
1.1.1 Non-text Content	A	Automated failures found
1.2.1 Audio-only and Video-only (Prerecorded)	A	No automated failures found; manual check still needed
1.2.2 Captions (Prerecorded)	A	No automated failures found; manual check still needed
1.2.3 Audio Description or Media Alternative (Prerecorded)	A	Not tested; manual check needed
1.2.4 Captions (Live)	AA	Not tested; manual check needed
1.2.5 Audio Description (Prerecorded)	AA	Not tested; manual check needed
1.3.1 Info and Relationships	A	No automated failures found; manual check still needed
1.3.2 Meaningful Sequence	A	Not tested; manual check needed
1.3.3 Sensory Characteristics	A	Not tested; manual check needed
1.3.4 Orientation	AA	No automated failures found; manual check still needed
1.3.5 Identify Input Purpose	AA	No automated failures found; manual check still needed

Success criterion	Level	Result from this scan
1.4.1 Use of Color	A	No automated failures found; manual check still needed
1.4.2 Audio Control	A	No automated failures found; manual check still needed
1.4.3 Contrast (Minimum)	AA	Automated failures found
1.4.4 Resize Text	AA	No automated failures found; manual check still needed
1.4.5 Images of Text	AA	Not tested; manual check needed
1.4.10 Reflow	AA	Automated failures found
1.4.11 Non-text Contrast	AA	Not tested; manual check needed
1.4.12 Text Spacing	AA	No automated failures found; manual check still needed
1.4.13 Content on Hover or Focus	AA	Not tested; manual check needed
2.1.1 Keyboard	A	Needs human review
2.1.2 No Keyboard Trap	A	Automated failures found
2.1.4 Character Key Shortcuts	A	Not tested; manual check needed
2.2.1 Timing Adjustable	A	No automated failures found; manual check still needed
2.2.2 Pause, Stop, Hide	A	No automated failures found; manual check still needed
2.3.1 Three Flashes or Below Threshold	A	Not tested; manual check needed
2.4.1 Bypass Blocks	A	No automated failures found; manual check still needed
2.4.2 Page Titled	A	No automated failures found; manual check still needed
2.4.3 Focus Order	A	No automated failures found; manual check still needed
2.4.4 Link Purpose (In Context)	A	Automated failures found

Success criterion	Level	Result from this scan
2.4.5 Multiple Ways	AA	Not tested; manual check needed
2.4.6 Headings and Labels	AA	Not tested; manual check needed
2.4.7 Focus Visible	AA	Needs human review
2.4.11 Focus Not Obscured (Minimum)	AA	Not tested; manual check needed
2.5.1 Pointer Gestures	A	Not tested; manual check needed
2.5.2 Pointer Cancellation	A	Not tested; manual check needed
2.5.3 Label in Name	A	No automated failures found; manual check still needed
2.5.4 Motion Actuation	A	Not tested; manual check needed
2.5.7 Dragging Movements	AA	Not tested; manual check needed
2.5.8 Target Size (Minimum)	AA	Automated failures found
3.1.1 Language of Page	A	Automated failures found
3.1.2 Language of Parts	AA	No automated failures found; manual check still needed
3.2.1 On Focus	A	Not tested; manual check needed
3.2.2 On Input	A	Not tested; manual check needed
3.2.3 Consistent Navigation	AA	Not tested; manual check needed
3.2.4 Consistent Identification	AA	Not tested; manual check needed
3.2.6 Consistent Help	A	Not tested; manual check needed
3.3.1 Error Identification	A	Not tested; manual check needed
3.3.2 Labels or Instructions	A	No automated failures found; manual check still needed
3.3.3 Error Suggestion	AA	Not tested; manual check needed
3.3.4 Error Prevention (Legal, Financial, Data)	AA	Not tested; manual check needed
3.3.7 Redundant Entry	A	Not tested; manual check needed
3.3.8 Accessible Authentication (Minimum)	AA	Not tested; manual check needed

Success criterion	Level	Result from this scan
4.1.2 Name, Role, Value	A	Automated failures found
4.1.3 Status Messages	AA	Not tested; manual check needed