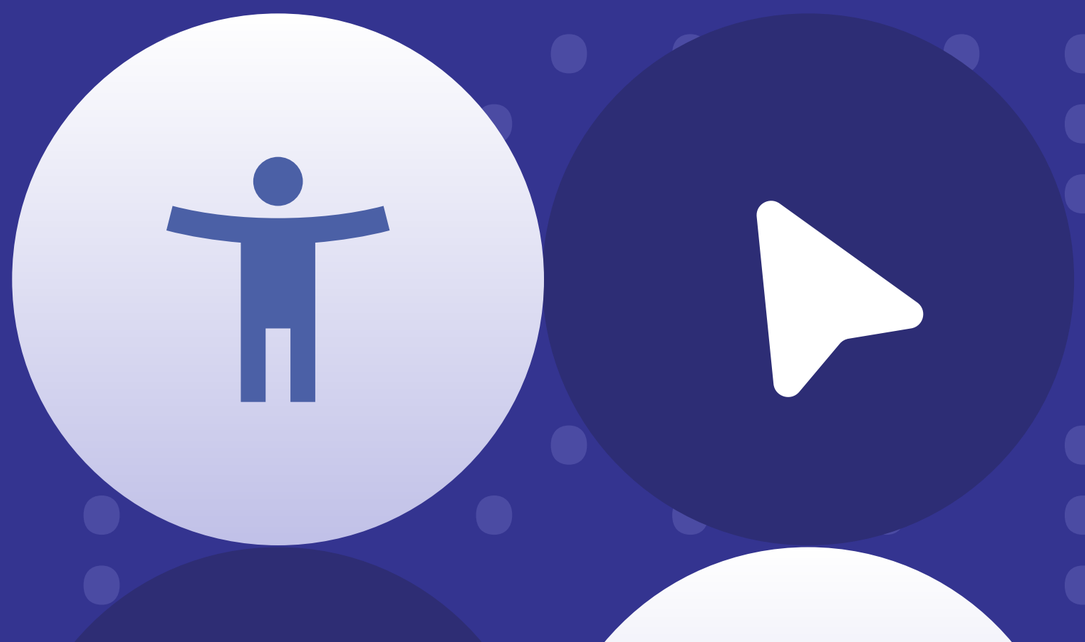


Accessibility Checklist

CRAFT INNOVATIONS

craftinnovations.global

Contents

Intro

About this checklist

This checklist was created by Craft Innovations to help designers, developers, and content teams build more accessible digital products.

It brings together 55 accessibility rules based on the Web Content Accessibility Guidelines (WCAG) 2.1 and 2.2 — Levels A and AA. These rules are presented as practical recommendations and are complemented by industry best practices. Each rule includes links to the original WCAG source and supplementary resources used to create this checklist.

Whether auditing an existing website, reviewing a design system, or testing a new feature, this checklist will help build a more inclusive interface.

Important notice

The information in this checklist is presented in a simplified and condensed format to make accessibility concepts easier to understand and apply. For detailed explanations, examples, and technical specifications, please refer to the official **WCAG Guidelines**↗ and the **WCAG Understanding Docs**↗.

While following this checklist will help improve accessibility, it does not guarantee full WCAG conformance or ensure complete accessibility for all users.

Terminology and key definitions

This glossary explains common terms and abbreviations used throughout the checklist, helping you understand accessibility concepts in plain language.

Term	Definition
WCAG	The Web Content Accessibility Guidelines are an international set of rules and standards designed to ensure access to web content for people with disabilities.
Conformance Levels (A, AA, AAA)	The three WCAG levels that indicate how accessible a product is. A – basic requirements, AA – enhanced accessibility and usability, AAA – the highest standard, often optional.
Success Criteria (SC)	A measurable accessibility requirement defined in the WCAG standard. Each success criterion describes a specific condition that must be met to achieve a certain conformance level (A, AA, or AAA).
Screen reader	A tool that reads content aloud for users who are blind or have low vision. Popular screen readers: NVDA, JAWS (Windows), VoiceOver (macOS/iOS), TalkBack (Android)
Assistive technology (AT)	Hardware or software that supports users with disabilities (e.g., screen readers, voice input, switches).
Keyboard navigation	A method of interacting with web content using only the keyboard instead of a mouse. This is a key requirement for individuals who cannot hold a mouse or rely on screen readers.
Accessible name	The programmatically determined name of a user interface element (such as a button, link, or form field) that is announced by a screen reader.
ARIA	Accessible Rich Internet Applications are special attributes used in HTML to provide assistive technologies (such as screen readers) with additional information about complex or interactive elements (like widgets or pop-up windows). For example, the aria-expanded="true" attribute informs the screen reader that a certain block of content (such as a menu) is open.

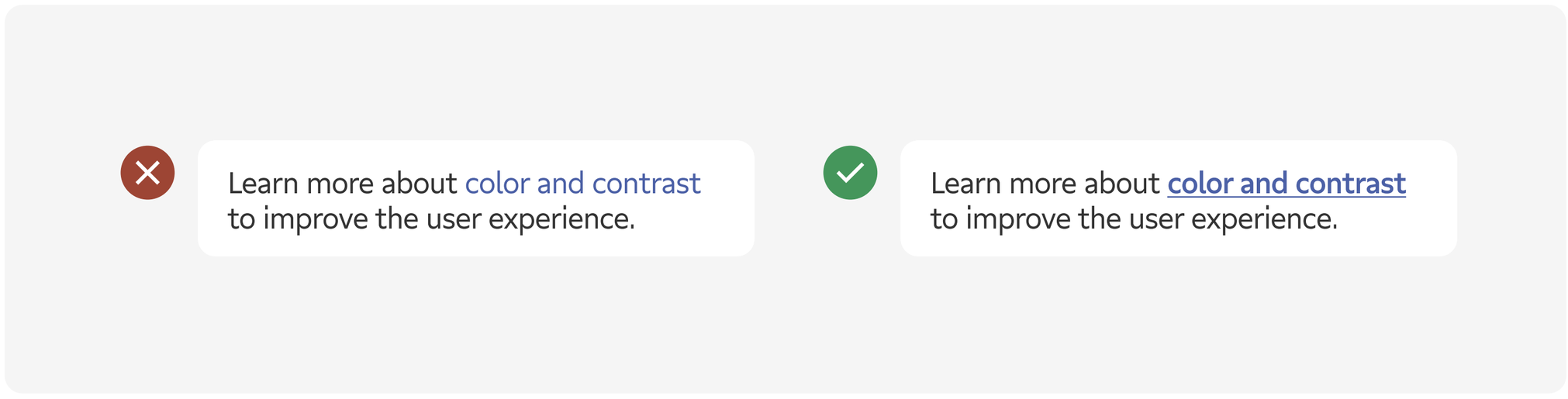
Visual Design and Perception

1.1 Visual Cues and Contrast

- [ ] **Do not use only color to convey meaning** Level A

  - Color should never be the only way to communicate information or status.
  - Combine color with other visual cues such as text, icons, patterns, or shapes to make meaning clear for all users — including those with color vision deficiencies.

WCAG — 1.4.1 Use of Color ↗

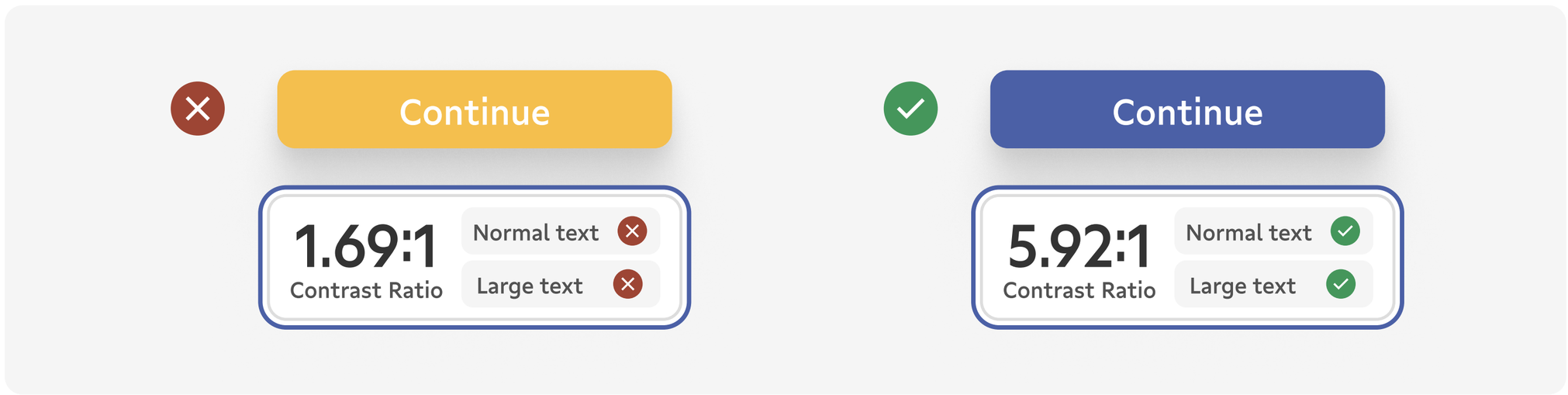


The example shows that links should not be identified by color alone. Use additional cues, such as an underline, so that users who cannot distinguish colors can easily differentiate links from plain text.

- [ ] **Ensure sufficient color contrast between text and background** Level AA

  - Use at least a 4.5:1 contrast ratio for regular-sized text.
  - Use at least 3:1 contrast ratio for large or bold text. This refers to text that is 18pt (24px) or larger, or 14pt (18px) and bold.
  - You can check contrast using free online tools like WebAIM Contrast Checker↗, Stark plugin for Figma↗ or browser extensions.

WCAG — 1.4.3 Contrast (Minimum) ↗



The example shows that the button with a yellow background has insufficient contrast between the text and background, as the ratio of 1.69:1 is below the minimum required 4.5:1. In contrast, the button on the right with a 5.92:1 ratio meets WCAG requirements.

- [ ] **Meaningful non-text content must achieve 3:1 color contrast** Level AA

This rule applies to graphical objects (like icons, charts or graphs) and active user interface elements (like buttons, checkboxes, form fields, and their states).

WCAG — 1.4.11 Non-text Contrast ↗

- [ ] **Don’t rely on color, shape, or position alone to convey meaning** Level A

- Information, instructions, or feedback should not depend only on visual or sensory characteristics — such as color, shape, sound, size, or position.
- Always offer an alternative method to understand the meaning through text labels or accessible names.

WCAG — 1.3.3 Sensory Characteristics ↗

✕

Tasks	Status
Wash dishes	●
Write a book	▲
Go to the gym	■

✓

Tasks	Status
Wash dishes	● completed
Write a book	▲ in progress
Go to the gym	■ postponed

The example shows that using color and shape for status is unclear, while adding text labels makes it accessible.

1.2 Typography and Readability

- [ ] **Text can be resized up to 200% without loss of content or functionality** Level AA

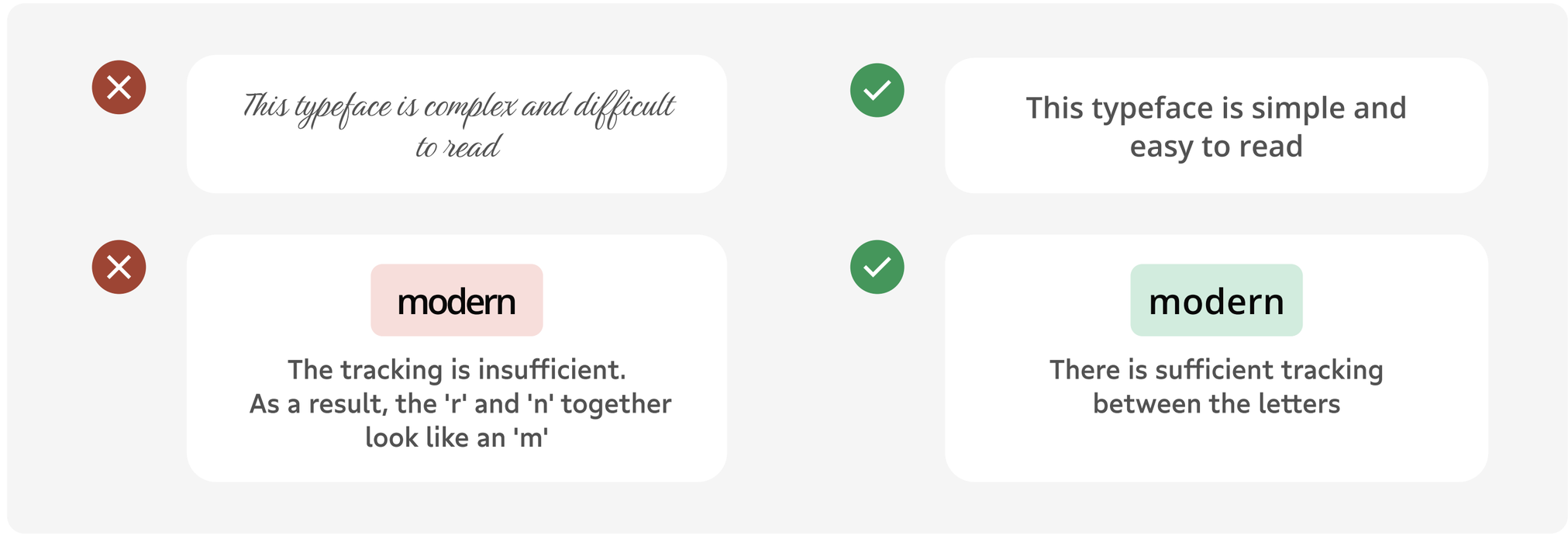
- The primary requirement is that all text must be able to increase in size up to 200% using the user's software/browser settings (not relying on specialized Assistive Technology) so content doesn't overlap, get cut off, or disappear.
- Use proper CSS settings (e.g., relative units like em or rem) to ensure the layout remains functional and readable.
- **Best Practice Tip:** While WCAG does not mandate a specific font size, it is highly recommended that body text is at least 16px and large text at least 24px for optimal readability.

WCAG — 1.4.4 Resize Text ↗ A11Y Collective — How to Pick the Perfect Font Size ↗

- [ ] **Ensure that the text is readable by using an appropriate typeface & styling** Level AA
  - Use simple, familiar, and easily-parsed fonts.
  - Line height to at least 1.5 times the font size;
  - Spacing following paragraphs to at least 2 times the font size;
  - Letter spacing (tracking) to at least 0.12 times the font size and word spacing to at least 0.16 times the font size.

  Note: While content isn't required to use these text spacing values, it should work correctly if users change text spacing. This requires using adaptable CSS.

  WCAG — 1.4.12 Text Spacing ↗ WebAIM — Typefaces and Fonts ↗



1.3 Animation and Motion

- [ ] **Moving or scrolling content that lasts over 5 sec. must allow users to pause, stop, or hide it** Level A

  Provide controls (pause buttons, arrows) for content that starts automatically (this includes carousels, image sliders, animation, videos etc.).

  WCAG — 2.2.2 Pause, Stop, Hide ↗

- [ ] **Make sure no content flashes more than 3 times in a second** Level A

  This accessibility rule is concerned with visual effects that may trigger photosensitivity or seizures in some individuals.

  Examples of violating the rule:

  - Animated GIFs: If a GIF flashes or changes at too high a frequency, it can cause discomfort for users.

  WCAG — 2.3.1 Three Flashes or Below Threshold ↗

Multimedia and Content

2.1 Images and Graphics

- [ ] **All non-text content that conveys important information needs alternative text (alt text)** Level A
  - Add descriptive alternative text to all meaningful images, illustrations, charts, etc. You can refer to the guide NN/g – Alt Text: What to Write ↗ for practical examples and best practices.
  - For decorative images, use an empty alt attribute (alt="") so screen readers ignore them. For decorative icons, apply aria-hidden="true".

WCAG — 1.1.1 Non-text Content ↗

Cocker Spaniels are known for their cheerful nature and playfulness.

alt="Cocker Spaniel runs across a green lawn with a red ball in its mouth"

- [ ] **Provide comprehensive alternatives for complex images** Level A

  For images containing substantial information (e.g., graphs, charts, diagrams, detailed maps), provide two-part text alternative:

  - Part 1: Use short alt text to identify the image and, if applicable, indicate where the detailed information is located (e.g., data available in the table below.).
  - Part 2: Long Description: Provide a complete textual explanation of all essential information conveyed by the graphic. Use a linked or expandable detailed section, an accessible data table, or text next to the graphic.

  WCAG — 1.1.1 Non-text Content ↗ WCAG. Complex images ↗

- [ ] **Avoid embedding meaningful text in an image** Level AA
  - Make sure you only have absolutely necessary images of text.
  - For images containing text, include alt text or a caption/description.

  WCAG — 1.4.5 Images of Text ↗

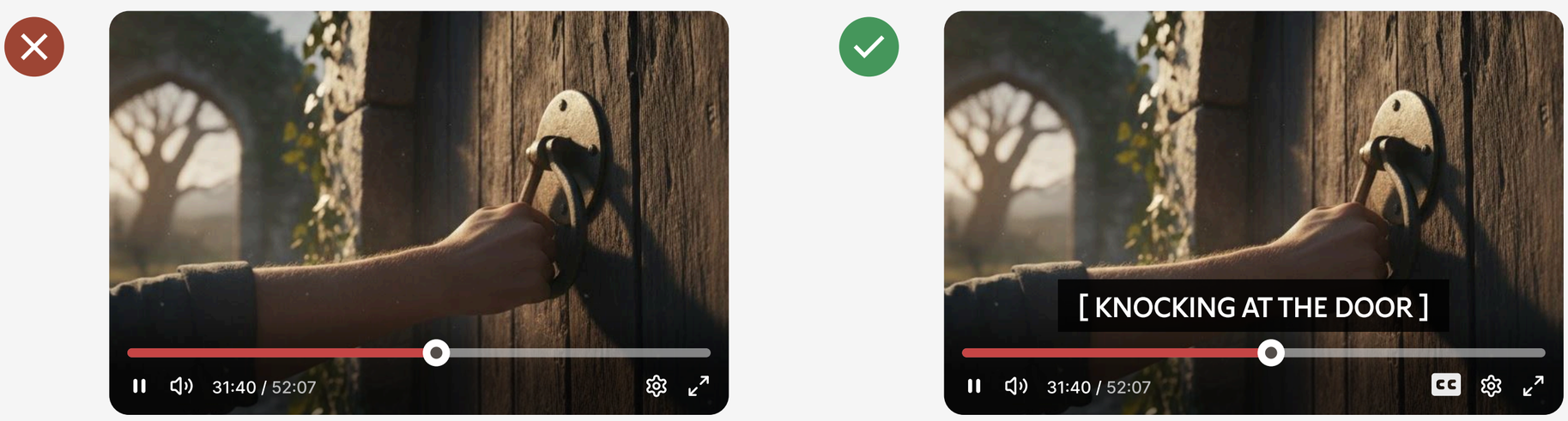
2.2 Audio and Video Content

- [ ] **Add alternatives to audio-only and video-only content** Level A
  - For prerecorded audio-only content (like a podcast), provide a text transcript.
  - For prerecorded video-only content (like a silent video), provide an audio description or a text transcript.

  WCAG — 1.2.1 Audio-only and Video-only (Prerecorded) ↗

- [ ] **Provide synchronized captions for all prerecorded video with audio** Level A
  - Ensure all dialogue and important auditory information (e.g., sound effects, background music, speaker identification) is presented as synchronized captions.
  - This makes audio content accessible to users who are deaf or hard of hearing, allowing them to follow the conversation and context.

  WCAG — 1.2.2 Captions (Prerecorded) ↗



- [ ] **Provide a description of the visual content in videos** Level A
  - Ensure that all important visual information (actions, graphics, scene changes, etc.) is verbally described, as this content is not available to users who are blind or have low vision.
  - This description must be provided either with a separate narration track (audio description) or through a detailed text transcript that includes the visual details. The description should be synchronized with the video and fill pauses in the existing dialogue.

  WCAG — 1.2.3 Audio Description or Media Alternative (Prerecorded) ↗

- [ ] **Provide synchronized text for audio content in real-time videos** Level AA

  If you are presenting a live broadcast, you need to provide real-time captions.

  WCAG — 1.2.4 Captions (Live) ↗

- [ ] **Provide a synchronized audio description of the visual content in videos** Level AA

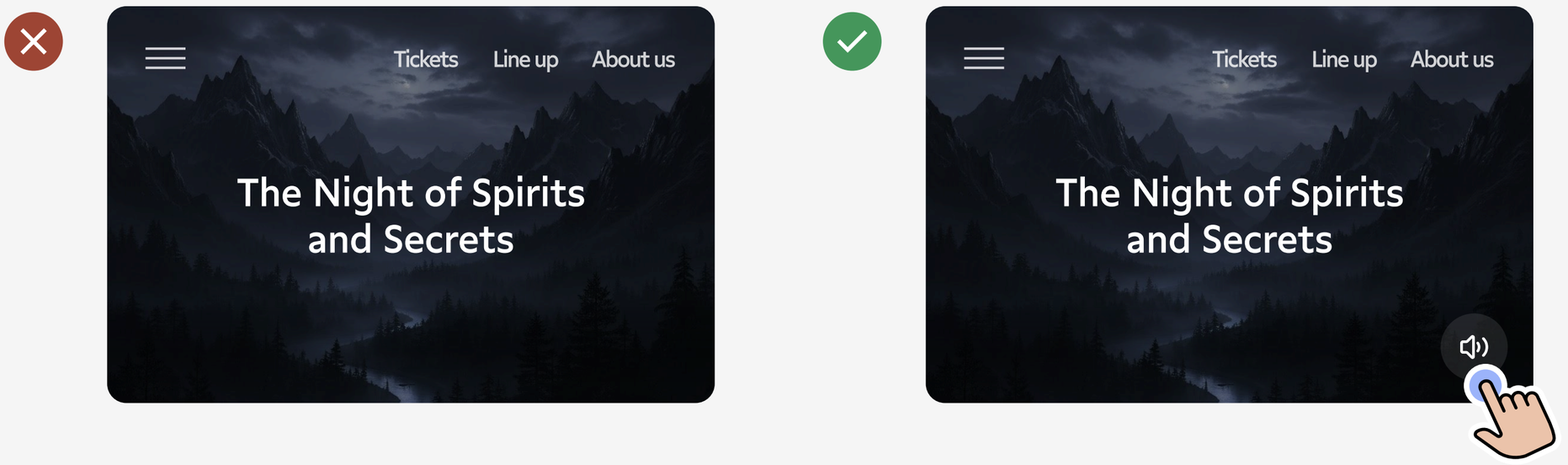
  - Mandatory Requirement: Provide a synchronized Audio Description for all meaningful visual information in prerecorded video content.
  - Audio Description is only required when crucial information is conveyed visually but not audibly in the main soundtrack. Example: In an instructional video demonstrating how to use software the on-screen actions must be verbally described.
  - Difference from SC 1.2.3: This criterion specifically requires an Audio Description and does not allow substituting it with only a detailed text transcript (which is sufficient for 1.2.3).

  WCAG — 1.2.5 Audio Description (Prerecorded) ↗

- [ ] **Provide a way to control or stop audio that plays automatically** Level A

  - If audio or video with sound starts automatically and lasts longer than three seconds, users must be able to pause, stop, or mute it.
  - Autoplaying audio can interfere with screen readers and create confusion or discomfort.
  - Avoid starting audio automatically. Instead, provide a clear play button or a dedicated button to mute the sound.

  WCAG — 1.4.2 Audio Control ↗



2.3 Language Settings

- [ ] **Set the correct language for each page** Level A
  - The main language of each page must be defined in the HTML using the lang attribute (for example, <html lang="en">).
  - This ensures that screen readers pronounce the content correctly and that translation tools work as expected.

  WCAG — 3.1.1 Language of Page ↗

- [ ] **Mark parts of content written in a different language** Level AA
  - When a section, phrase, or word is written in a language different from the page's main language, it must be marked with the correct lang attribute in HTML (for example, Bonjour).
  - This helps screen readers switch pronunciation rules and read the text accurately.

  WCAG — 3.1.2 Language of Parts ↗

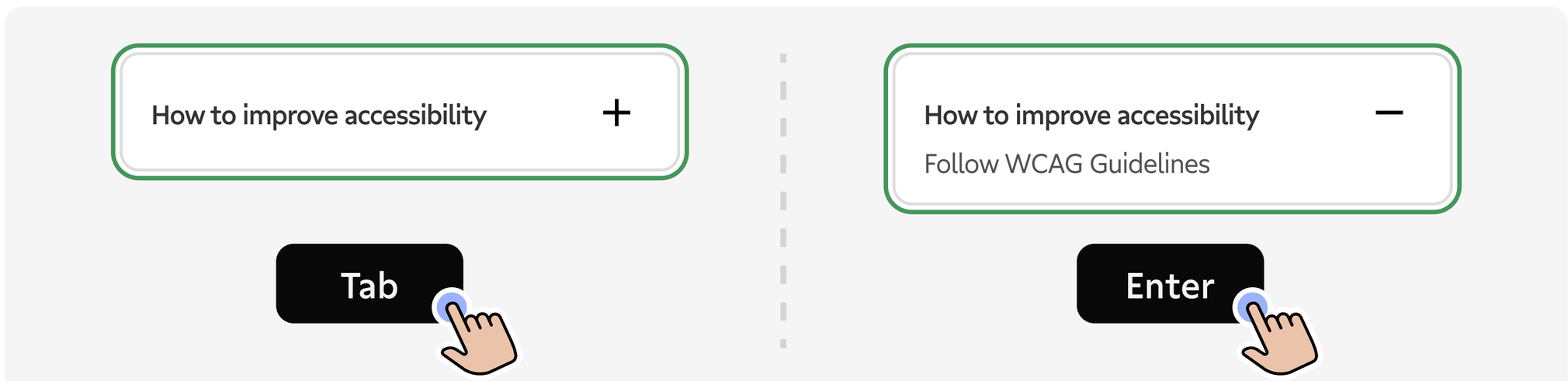
Interactive Elements

3.1 Keyboard and Focus Navigation

- [ ] **All functionality must be operable using only a keyboard** Level A

  - Ensure you can access and operate every feature/interactive element (e.g., buttons, menus, sliders, links) only using a keyboard.
  - For a detailed guide on common keyboard interactions, refer to the article WebAIM – Keyboard Accessibility Guide ↗

  WCAG — 2.1.1 Keyboard ↗

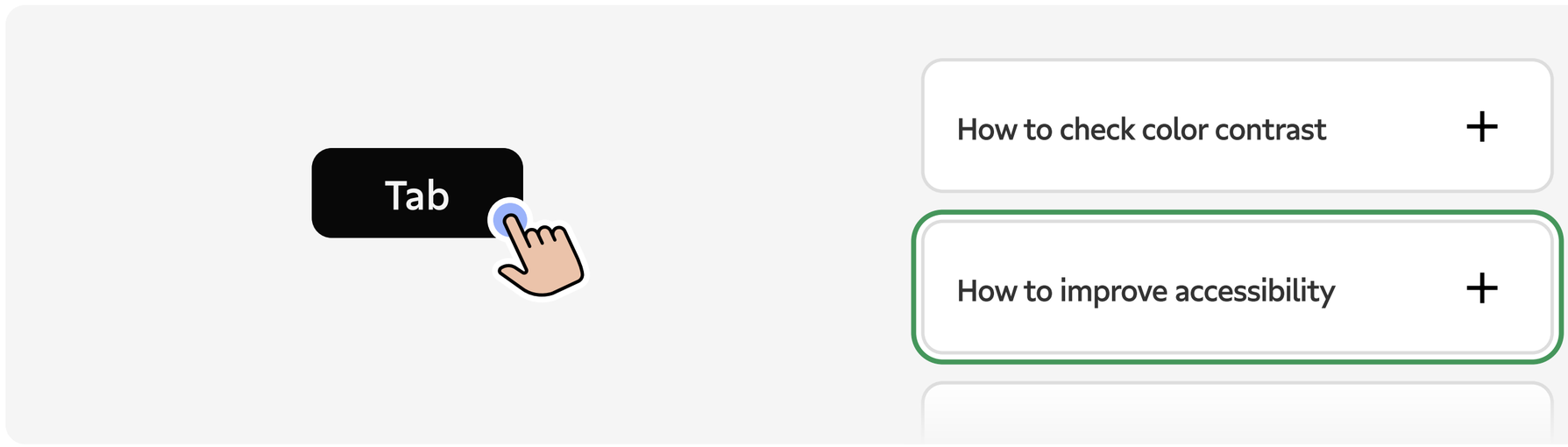


Visual representation of how to navigate and expand accordion content using the keyboard — press Tab to focus the element and Enter or Space to open it.

- [ ] **Any interactive element must have a clearly visible focus indicator** Level AA

  When navigating with the keyboard, the interactive element that receives focus must be visibly distinct. Use a clear outline or box-shadow (at least 2 px thick is recommended) with a minimum 3:1 contrast ratio.

  WCAG — 2.4.7 Focus Visible ↗



The example of an interactive element in a focus state outlined with a green border.

- [ ] **Ensure the focused element is visible and not hidden behind other content** Level AA

  - When an element receives keyboard focus, it must not be hidden behind pop-ups, sticky headers, or other overlapping content.

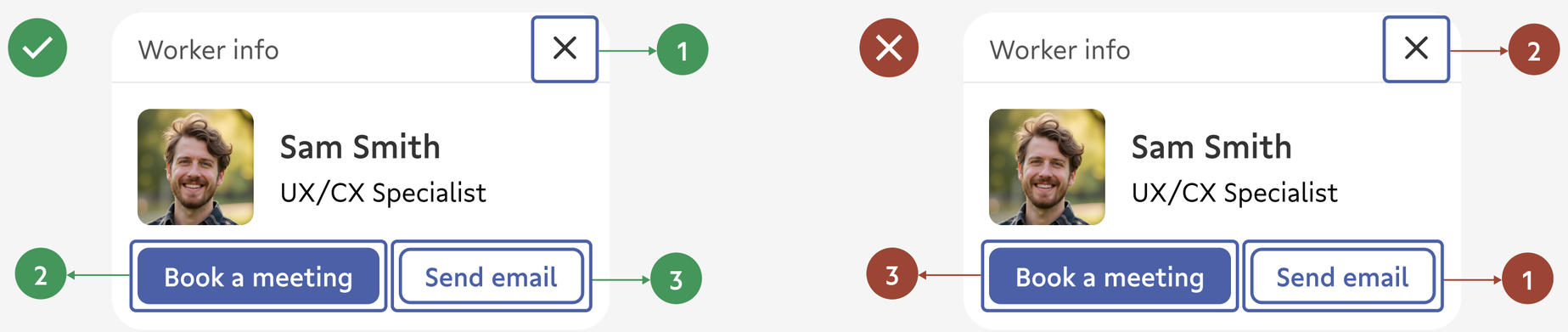
- Users should always be able to see where the focus is and interact with it without obstruction.

WCAG — 2.4.11 Focus Not Obscured (Minimum) ↗

- [ ] **Keyboard focus must follow a logical and predictable order** Level A

- The focus order should follow the visual and logical structure of the page. Typically, focus moves from top to bottom and left to right, following the intended reading and interaction flow.
- To test the order, navigate using the Tab key.

WCAG — 2.4.3 Focus Order ↗



- [ ] **Ensure keyboard focus can move away from any element** Level A

- Keyboard focus should never become trapped inside an element or component. Users must be able to move focus away using standard keys like Tab (navigate forward), Shift + Tab (navigate backward), or Esc.
- If focus gets stuck in a modal, menu, or other component without a way to leave, interaction with the page is blocked, that does not meet accessibility standards.

WCAG — 2.1.2 No Keyboard Trap ↗

- [ ] **Character key shortcuts must not cause accidental actions** Level A

- Single-character shortcuts (like pressing S to open search) can lead to unintended actions, particularly for users relying on voice input or while typing text.

- Check all keyboard shortcuts. Ensure they can be turned off, modified in settings, or work only within a specific active component.

WCAG — 2.1.4 Character Key Shortcuts ↗

☐ **Provide enough time to interact with content** Level A

- If your website or app includes a time limit (e.g., countdown, session timeout, or quiz timer), users must be able to turn it off, extend, or adjust it. At least one of the following must be available:
  - The option to disable the time limit entirely.
  - The ability to extend or increase the limit to at least 10 times the default duration.
  - A warning before the time expires, giving at least 20 seconds to extend it by 10x or more.
- Some exceptions apply — such as real-time events (e.g., live auctions), essential time limits (like purchasing tickets where extending time would affect the outcome), or cases where the time limit exceeds 20 hours.

WCAG — 2.2.1 Timing Adjustable ↗

3.2 Controls: Links, Buttons, and Widgets

- [ ] **Provide descriptive names or context for all links** Level A

  - Each link must have clear, descriptive name that communicates its destination or function. Avoid using vague links like "Click here", "Learn more", and using long URLs as link text.
  - Check if the link name makes sense on its own. If space or design constraints limit the visible label, consider using methods like visually hidden text (CSS) or ARIA labels to provide sufficient context for screen reader users.

WCAG — 2.4.4 Link Purpose (In Context) ↗ A11Y Collective — Link Accessibility ↗

✗ Click here to learn more about New WCAG Guidelines

✓ Learn more about New WCAG Guidelines

- [ ] **Inform users that a link will open in a new tab or window** Level AA

  - Unexpectedly opening new windows can disorient users, especially those with visual or cognitive difficulties.
  - Add a short note next to the link text such as "(opens in a new tab)", or include an external-link icon with accessible text that is visually hidden for sighted users.

WCAG Technique G201 ↗

You will find more information in the article Link Accessibility

```
<span class="visually-hidden">Opens in a new tab</span>
```

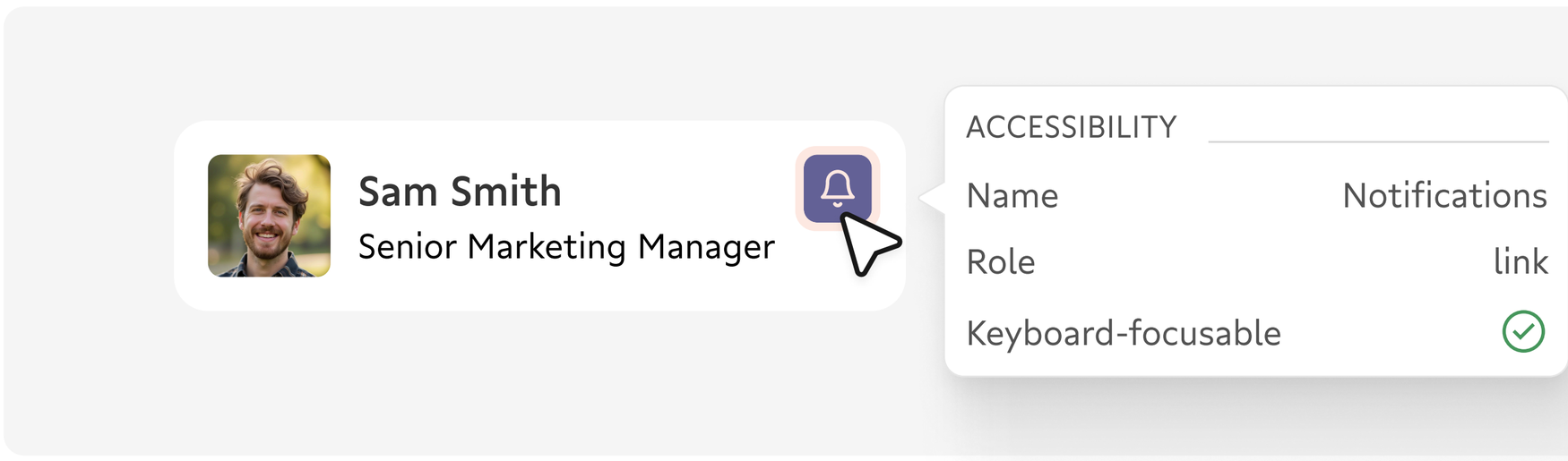
The example demonstrates using an external link icon along with hidden text that informs screen reader users about the link opening in a new tab.

Define name, role, and state for all controls

Level A

- All interactive elements (e.g., buttons, links, checkboxes) must have a programmatically identifiable name, role, and state/value or ARIA labels.
- This allows assistive technologies (like screen readers) to announce what the element is, what it does, and its current status:
  - **Name** — a text label describing how the control is called (e.g. “Notifications”).
  - **Role** — defines the type of element (for example, button, link, slider).
  - **State / Value** — communicates the current condition or value of the element (e.g., checked, expanded).

WCAG — 4.1.2 Name, Role, Value ↗ Video – Accessibility Annotations ↗



Example of how accessibility properties are displayed in the browser's Developer Tools when inspecting an interactive element. The "Name" (Accessible Name) and "Role" are the properties a screen reader will announce.

Match visible labels with accessible names

Level A

The visible label should match or be part of the accessible name (announced by assistive technologies). For example, a navigation link label is “View gifts,” and the same text is used as the accessible name.

WCAG — 2.5.3 Label in Name ↗

Use consistent identification for repeated elements

Level AA

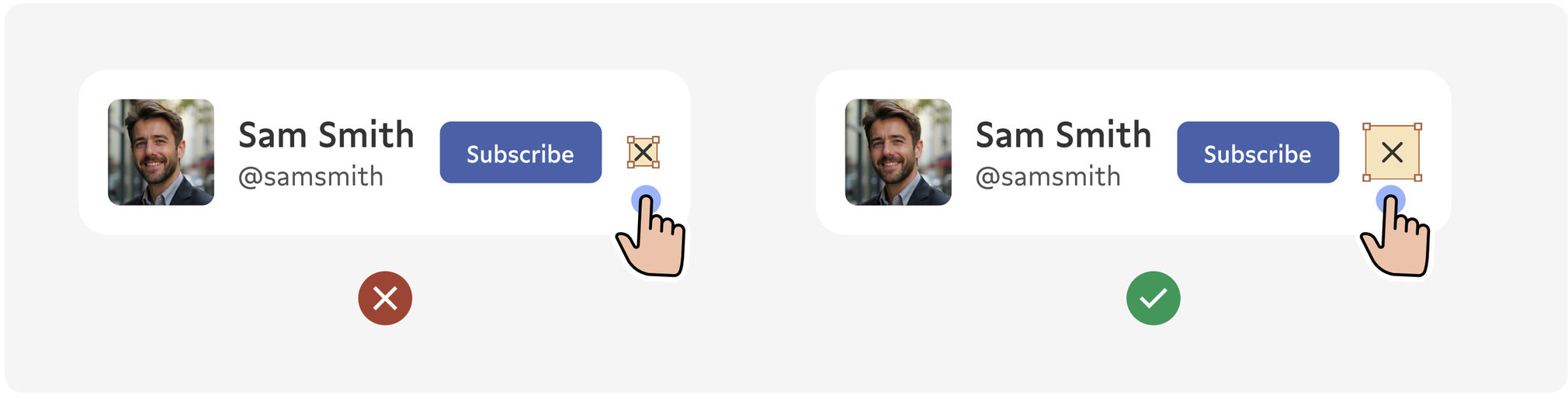
- When elements perform the same action, they should look and be labeled in the same way. For example, if one search icon is labeled “Search”, the same label should be used for all other search icons.
- Consistent naming and design help users — especially those using assistive technologies — recognize patterns and predict interactions.

WCAG — 3.2.4 Consistent Identification ↗

- [ ] **Make interactive elements large enough to tap or click easily** Level AA

  - Interactive elements such as buttons, links, and icons must be large enough to prevent accidental taps or clicks.
  - Minimum target size is 24 × 24 CSS px, but for touch interfaces, the tap area should ideally follow platform guidelines:
    - Google Material Design: at least 48 × 48 dp
    - Apple Human Interface Guidelines: at least 44 × 44 pt

  WCAG — 2.5.8 Target Size (Minimum) ↗



- [ ] **Prevent unexpected changes on focus or input** Level A

  - Gaining focus or entering data in a control must not automatically cause navigation, submission, or layout changes unless clearly indicated beforehand.
  - For example, avoid automatic page navigation after selecting an option in a dropdown without explicit confirmation.

  WCAG — On Focus 3.2.1 ↗ WCAG — On Input 3.2.2 ↗

- [ ] **Keep hover and focus content visible and easy to interact with** Level AA

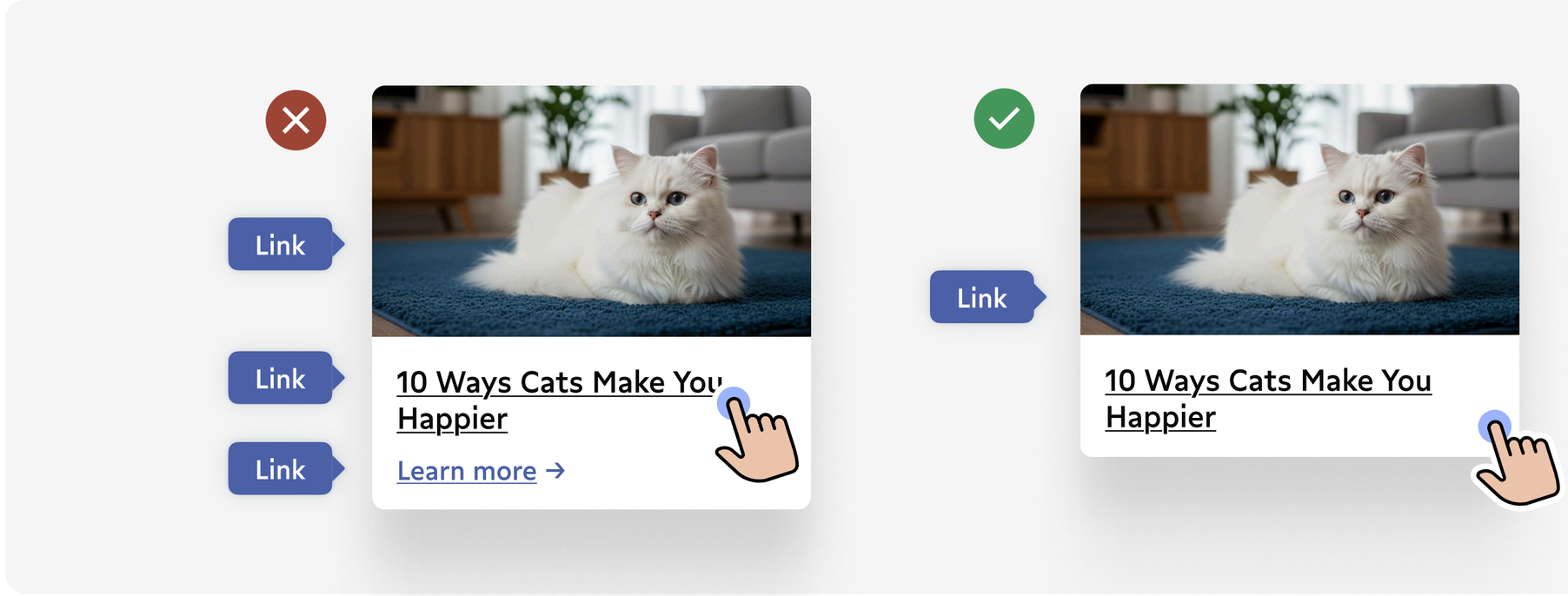
  - When additional content appears after hovering or focusing on an element (for example, a tooltip or dropdown menu) it must be easy to read, access, and close. The additional content should be:
    - **Persistent.** Stay visible until the hover or focus trigger is removed by the user or the information is no longer relevant.
    - **Hoverable.** Be easy to move to without disappearing when the pointer moves over it. A simple example: When the tooltip appears, you can move the cursor from the initial element to the tooltip itself to read it or click on the link within it.

- **Dismissible.** Have a clear way to dismiss, without moving pointer or focus, such as pressing Esc or close button.

WCAG — 1.4.13 Content on Hover or Focus ↗

- [ ] **Avoid multiple links to the same destination within a single card**
  - When a card contains multiple links to the same page, users navigating with screen readers or a keyboard encounter duplicate interactive elements with identical purpose, causing confusion and extra navigation steps.
  - Each card should contain only one interactive link leading to a given destination. Avoid including several elements (for example — the title, image, and link) that all link to the same URL.
  - Instead, choose one approach:
    - Option 1: Make the entire card clickable (the image, title, and area all act as one link).
    - Option 2: Keep only a single, clear link such as "Explore our services".

Graceful Web Studio: Accessible card linking best practices ↗



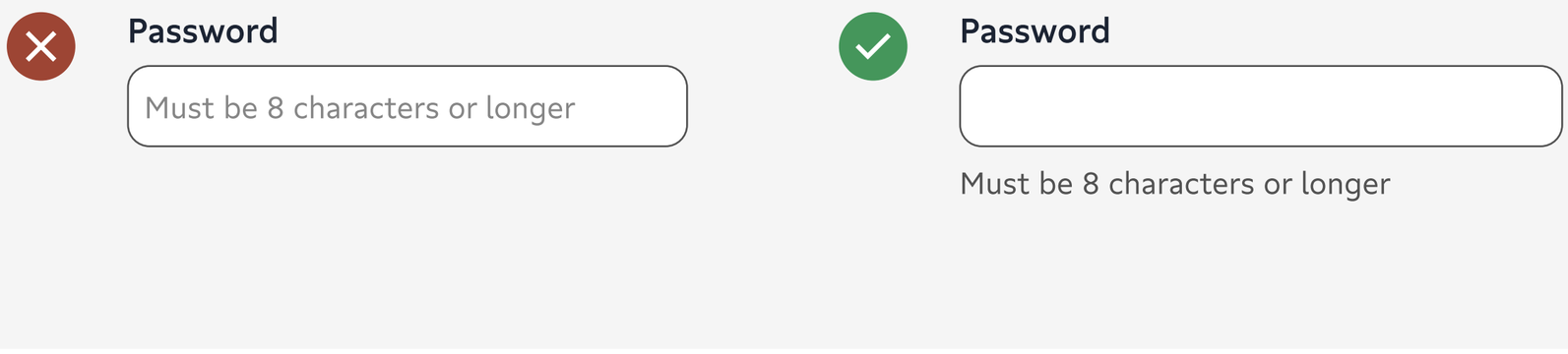
This illustration represents one of the accessible design options — linking the whole card as a single interactive element.

3.3 Forms and Input

- [ ] **Use clear labels and instructions for all form fields** Level A

  - Every input field must have a visible label that clearly describes what information is expected and is programmatically associated with the input field, to allow assistive technologies to announce it.
  - Place labels close to their fields and avoid relying only on placeholder text, as placeholders disappear once typing begins and are not always read by assistive technologies.
  - Provide additional instructions when needed (for example, "Password must be at least 8 characters long")

WCAG — 3.3.2 Labels or Instructions ↗ WebAIM – Creating Accessible Forms ↗



The illustration compares two password fields: the left one uses a placeholder for instructions, while the right one includes a visible helper text below, making it accessible.

- [ ] **Group related elements logically** Level A

  - When form inputs or other controls belong to the same category, they should be grouped using semantic HTML or ARIA attributes. Grouping helps assistive technologies announce the relationship between elements, improving navigation and understanding.
  - For instance, a set of radio buttons for shipping options should be wrapped inside a fieldset labelled "Shipping Options", so screen readers announce the group heading before each choice.

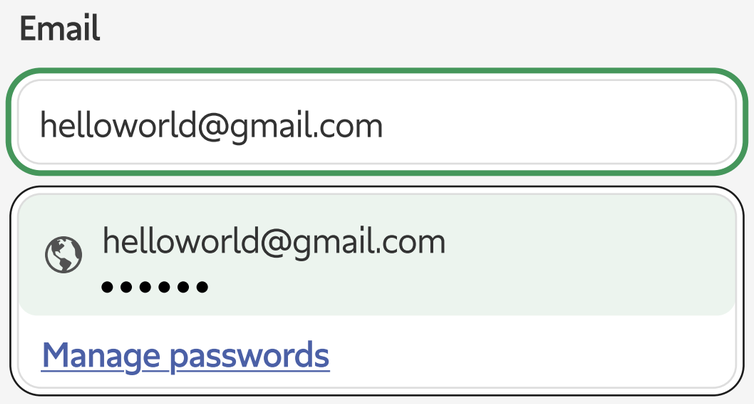
WCAG — 1.3.1 Info and Relationships ↗ WebAIM – Creating Accessible Forms ↗

Make form fields recognizable for autofill and assistive tools

Level AA

- Form fields that collect common user information — such as name, email, address or phone number — should have a programmatically defined purpose. Use semantic identifiers like autocomplete="email" or autocomplete="name" to enable autofill and assistive features.
- This helps browsers, password managers, and assistive technologies recognize the field type and offer features like autofill or voice input, making forms faster and easier to complete for everyone.

WCAG — 1.3.5 Identify Input Purpose ↗



Avoid redundant data entry

Level A

- Do not require users to re-enter information that has already been provided. For example, if the user enters a billing address, do not require them to re-enter the same information for the shipping address; instead, offer a 'Same as Billing Address' checkbox.
- Preserve input after errors when possible.
- This reduces cognitive load and helps users with memory or motor difficulties.

WCAG — 3.3.7 Redundant Entry ↗

Ensure authentication does not require complex cognitive tasks

Level AA

- Provide users with an easy way to log in that doesn’t make them remember, solve, or transcribe any information.
- Provide accessible alternatives such as “magic link” login, or copy-paste support for one-time codes.

WCAG — 3.3.8 Accessible Authentication (Minimum) ↗

Identify and describe errors clearly

Level A

- When a form field contains an error, the error must be clearly visible and programmatically connected to the specific field where it occurred (for example via <aria-describedby> or similar).
- The error message should explain what exactly is wrong and ideally how to fix it (for example, “Password must include at least one number”).
- Provide additional visual indicators to make error detection easier, such as highlighting the field border in red and displaying an error icon.

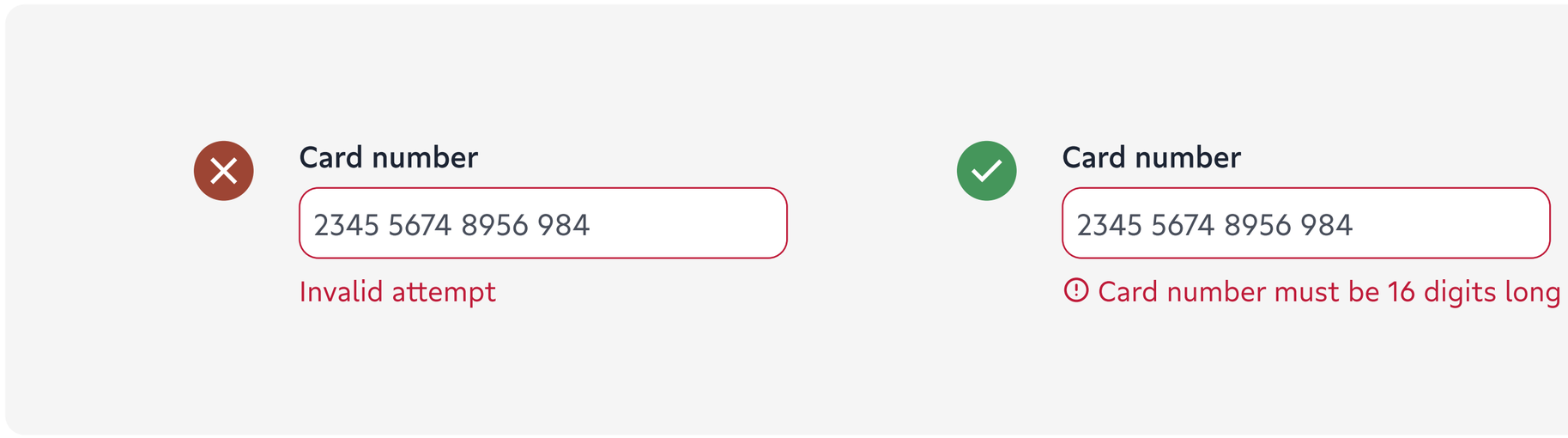
WCAG — 3.3.1 Error Identification ↗

Suggest ways to correct input errors

Level AA

- Provide clear instructions on how to fix errors, unless doing so would compromise the security or intended purpose of the content.
- For example, “Card number must be 16 digits long”.

WCAG — 3.3.3 Error Suggestion ↗



Prevent critical errors in important forms

Level AA

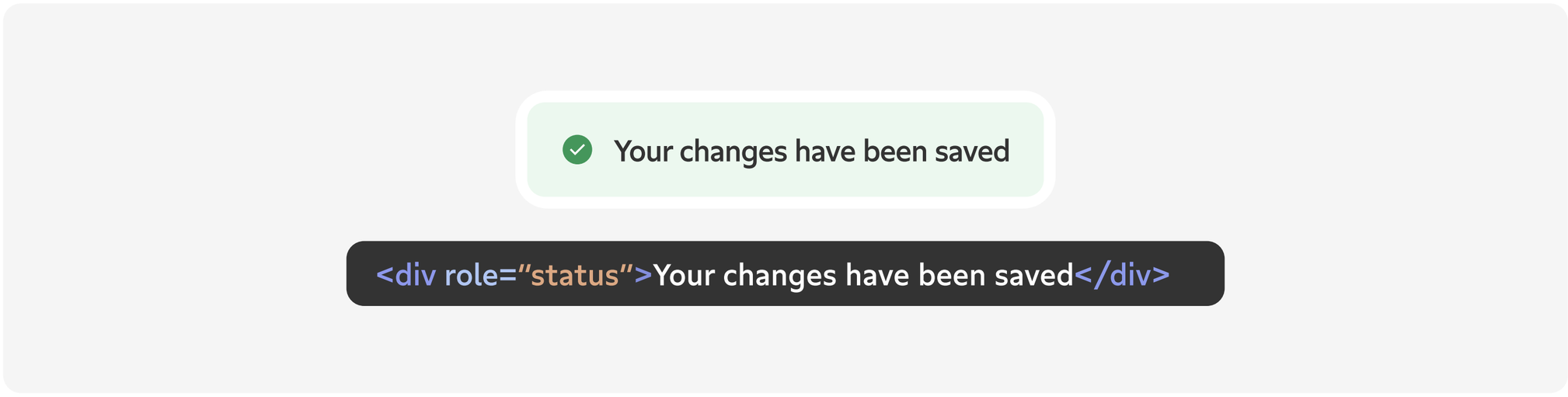
- Forms involving financial, legal, or personal data must include mechanisms to prevent irreversible errors.
- Before final submission, the process should allow one of the following:
  - Error check — the system automatically checks the data for mistakes and lets the person fix them before submitting.
  - Confirmation — there’s a clear review step to confirm or correct information before it’s final.
  - Undo — the action can be reversed after it’s submitted.

WCAG — 3.3.4 Error Prevention (Legal, Financial, Data) ↗

- [ ] **Make sure assistive technology can detect and announce status messages** Level AA

  - Status messages (such as form validation errors, success confirmations, or loading indicators) must be automatically announced by assistive technologies without changing the user's focus.
  - Use ARIA roles such as role="status", role="alert", or aria-live regions to ensure that status messages are announced by screen readers.
  - The choice between role="alert", role="status", and aria-live depends on the urgency and context of the message:
    - use role="alert" for important messages (e.g., form errors or failed submissions) that will be announced immediately.
    - use role="status" for non-critical information (e.g., content saved).
    - use aria-live regions for dynamic updates that change over time (e.g., live notifications, data loading indicators).

WCAG — 4.1.3 Status Messages ↗ A11y Playground — Status messages ↗ MDN — aria-live ↗



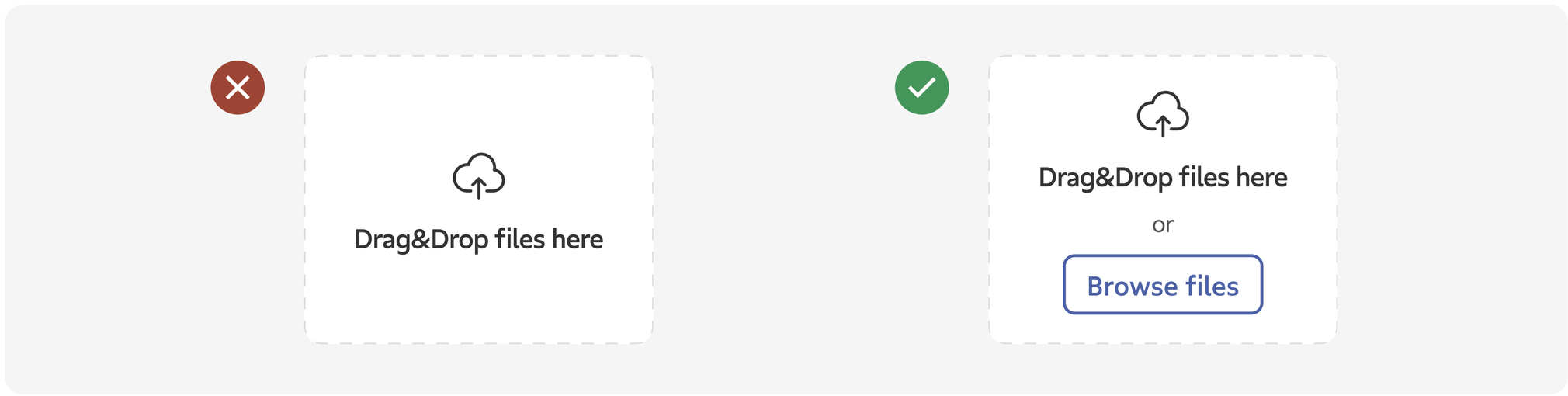
The example shows how using role="status" allows non-critical messages, like "Your changes have been saved," to be automatically announced by screen readers without shifting focus.

3.4 Gestures and Motion

- [ ] **Avoid complex gestures or provide simple alternatives** Level A

  - Actions that require multiple fingers or complex gestures (like swipe, pinch, drag and drop, zoom) must also be available through simple, single-point input such as a tap, click or keyboard command.
  - This ensures users who cannot perform precise gestures — for example, due to motor impairments — can still access the functionality.

WCAG — 2.5.1 Pointer Gestures ↗ WCAG — 2.5.7 Dragging Movements ↗



The illustration shows that the correct example provides an alternative way to upload a file — using the “Browse files” button instead of relying only on the “Drag and Drop” gesture.

- [ ] **Allow disabling motion-based interactions** Level A

  - If device movement (like shaking or tilting) triggers an action, an alternative way to perform that action must be available without motion.
  - Users should also be able to turn off motion controls completely.

WCAG — 2.5.4 Motion Actuation ↗

- [ ] **Prevent accidental actions from unintended touches** Level A

  - An action should only be executed after the interaction is completed, specifically when the user releases the mouse button (on mouseup), rather than when the touch or click begins (on mousedown).
  - For example, a “Submit” button sends the form only after the click is released (on mouseup).
  - This helps prevent accidental activations and provides better control, especially for people using assistive technologies or those with limited motor precision.

WCAG — 2.5.2 Pointer Cancellation ↗

Navigation and Structure

4.1 Page structure and hierarchy

Give each page a clear and unique title

Level A

- Every page must have a descriptive and unique title that clearly indicates its purpose or content.
- Check that the <title> element in the HTML reflects the main topic of the page and make sense when read out of context.
- The title should appear in the browser tab and help users distinguish between open pages or identify the page in search results.

WCAG — 2.4.2 Page Titled ↗ WCAG — Providing descriptive titles for web pages ↗

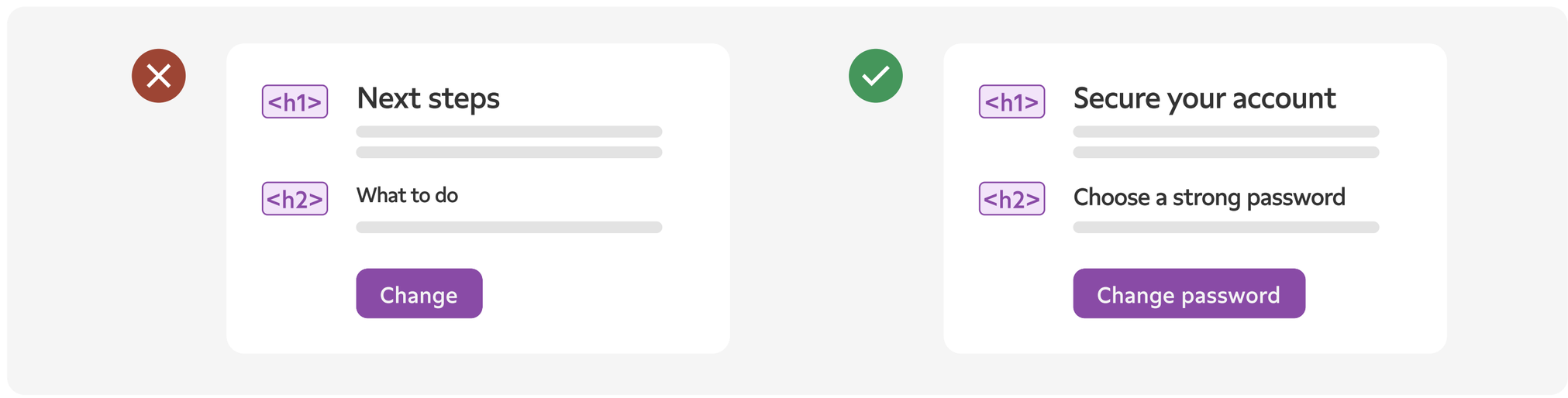
Use clear and descriptive headings and labels

Level AA

Headings and labels help users understand the purpose of content and controls. They should be clear, descriptive, and explain either what information follows (for sections and content) or what action can be taken (for interactive elements):

- Use meaningful and concise headings that follow a logical hierarchy (from <h1> to <h6>, without skipping levels). Each page must include one main <h1> heading.
- Provide descriptive labels for all controls that clearly describe their function or destination — for example, “View latest articles” or “Add to cart.”

WCAG — 2.4.6 Headings and Labels ↗ WAI — Headings Tutorial ↗



The example demonstrates the use of descriptive and non-descriptive headings and labels.

Use structure and relationships to organize content

Level A

- Use proper HTML elements — like headings, lists, tables, and landmarks — to represent structure and relationships between content.

- This helps assistive technologies understand how information is grouped and how sections relate to each other.

WCAG — 1.3.1 Info and Relationships ↗ A11y Playground – Info and relationships ↗

Structure	Relationship
Lists	Implement all lists using semantic list elements (, , and)
Tables	Use <table> for tabular data, ensuring clear headers with <th> and data cells with <td>
Landmarks and regions	Use semantic HTML5 landmark tags (<header>, <nav>, <main>, <footer>, etc.) and corresponding ARIA roles to define major regions of the page for easier navigation.

The illustration provides examples of how to present information and relationships correctly by using semantic HTML.

Maintain a logical reading and navigation order

Level A

- Use semantic HTML and a logical structure to keep the reading and navigation order consistent with how content is presented visually.
- This ensures that information makes sense when read by screen readers or navigated with a keyboard, even if the visual layout differs from the underlying code.
- Use the Tab key and a screen reader to navigate through content. Confirm that the reading and focus order follows a logical flow.

WCAG — 1.3.2 Meaningful Sequence ↗ A11y Playground – Meaningful content order ↗

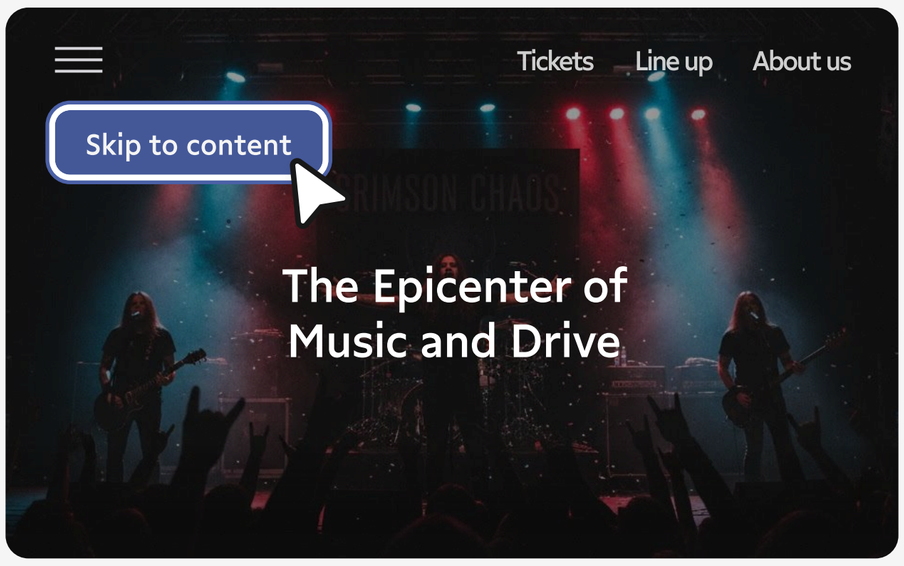
4.2 Navigation and wayfinding

Add a skip link to each page of your website

Level A

- Users must be able to quickly skip navigation menus or headers and jump straight to the main content of the page. This improves efficiency, especially for keyboard and screen reader users, who otherwise need to move through the same links repeatedly.
- Check that a “Skip to main content” link appears as the first focusable element and works correctly.

WCAG — 2.4.1 Bypass Blocks ↗



A visual example illustrating how the "Skip to main content" might appear on the page.

- [ ] **Provide multiple ways to find content** — Level AA

  - Users should be able to find pages or information in more than one way.
  - Provide at least one additional navigation method besides the main menu (e.g., search, sitemap, a list of related pages). Check that all methods lead to the same content consistently.

  WCAG — 2.4.5 Multiple Ways ↗

- [ ] **Keep navigation consistent across pages** — Level AA

  - Navigation elements (such as menus, headers, search, or footers) that appear on multiple pages must maintain consistent positioning and order.
  - Consistent navigation helps users quickly understand where they are and how to move around the site.

  WCAG — 3.2.3 Consistent Navigation ↗

- [ ] **Provide consistent access to help options** — Level A

  - If your website or app includes help features — such as chat support, contact links, or help/FAQ pages — they must be presented in the same place and manner on all pages where they appear.
  - This ensures users can always find assistance when they need it, without confusion.

  WCAG — 3.2.6 Consistent Help ↗

4.3 Adaptability

- [ ] **Support both portrait and landscape orientation** Level AA
  - Content must work in both portrait and landscape modes unless a specific orientation is essential for its use — for example, a piano app or a game that depends on horizontal movement.
  - Make sure that all content remains visible and functional in both orientations. Verify that interactive elements are still reachable and not cut off.
  - Users should not be forced to rotate their device to view or interact with content.

WCAG — 1.3.4 Orientation ↗

- [ ] **Ensure content reflows without loss of information** Level AA
  - Content must be usable and readable when zoomed up to 400% or viewed on smaller screens (320px) — it should not require scrolling in more than one direction (usually vertically).
  - Avoid fixed-width containers or elements that force horizontal scrolling when zoomed or viewed on narrow screens. Also ensure that content is not overlapped, hidden, or cut off.
  - Allow exceptions only when two-direction scrolling is essential (e.g., maps, diagrams).

WCAG — 1.4.10 Reflow ↗

Testing and Validation

Testing and Validation

Congratulations on reaching the last chapter! This is where all your design and development work comes together — testing and validation ensure that your interface truly works for everyone.

Why accessibility testing matters

Accessibility testing isn't just about checking boxes — it's about seeing how real people interact with your product. It helps identify whether users can find information, complete tasks, and navigate your interface without barriers.

Types of accessibility testing

1. Automated testing tools

Automated tools (browser extensions, scanners) help quickly identify technical issues like missing alt text, low contrast, or incorrect heading structure.

However, as W3C notes, these tools should be only one part of your testing process — they can't evaluate content clarity or interaction logic.

Action steps:

- Choose tools that match your tech stack (for example: Axe or WAVE) and integrate them into your process. A comprehensive list of tools and references is available in the Web Accessibility Evaluation Tools List ↗.
- Review reports regularly and fix recurring issues.

2. Manual testing

Manual accessibility testing is typically performed by designers, developers, and QA specialists as part of regular product testing. In some cases, dedicated accessibility experts may be involved to conduct more advanced or formal evaluations.

This process is used to ensure the product or website is accessible. It involves simulating common user journeys—navigating between pages, using interactive elements, filling out forms, and more. This helps uncover accessibility issues that automated tools often miss.

Action steps:

- Identify the key user flows, and then test their accessibility using the keyboard alone and a screen reader.
- Document your findings using a checklist based on WCAG criteria.

3. Usability testing with people with disabilities

This is the most valuable testing phase — involving real users who rely on assistive technologies such as screen readers, voice input, or keyboard navigation.

According to the Nielsen Norman Group, observing these users provides unique insights into real accessibility barriers and how design decisions affect usability.

Action steps:

- Recruit 5–8 participants with diverse accessibility needs.
- Prepare realistic task scenarios (e.g., "order a product" or "find and change your account settings").
- Observe how participants interact, take notes, and document obstacles and improvement opportunities.

Final thoughts

Thank you for reaching this final section! 🎉

Keep testing, keep learning, and make sure your interface is truly inclusive — for everyone. Your effort directly impacts the lives of millions, ensuring equal access to information and services.

About Craft Innovations

Craft Innovations is a global customer research and UX design firm dedicated to helping financial industry leaders innovate faster and deliver exceptional digital and physical customer experiences.

Research services:

- Discovery and product-market fit research
- Customer Experience and UX audit
- Benchmarking, CVP testing and validation
- Usability testing

Design services:

- UX/UI design: mobile, web banking, websites
- Product Value Proposition design
- Gamification design
- AI UX design

Segments:

- Retail & Corporate banking
- Landing / BNPL / Wallets / PFM
- Investech & Insuretech
- Crypto

Let's start with a friendly talk!

Book an intro meeting ↗

CRAFT INNOVATIONS

Visit website ↗

bizdev@craftinnovations.global

+1437 4214992

REVIEWED ON Clutch 20 REVIEWS