



Accessibility

These checks highlight opportunities to [improve the accessibility of your web app](#). Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so [manual testing](#) is also encouraged.

CONTRAST

▲ Background and foreground colors do not have a sufficient contrast ratio.

Low-contrast text is difficult or impossible for many users to read. [Learn how to provide sufficient color contrast.](#)

Failing Elements

button.px-3.py-1.5.rounded-full.text-sm.transition-all.hover:shadow-sm

div.bg-white.rounded-xl.p-6.shadow-xl.relative

button.px-3.py-1.rounded-full.text-sm.transition-all

div.bg-white.rounded-xl.p-6.shadow-xl.relative

button.px-3.py-1.rounded-full.text-sm.transition-all

div.bg-white.rounded-xl.p-6.shadow-xl.relative

These are opportunities to improve the legibility of your content.

ADDITIONAL ITEMS TO MANUALLY CHECK (10)

Hide

<input type="radio"/> Interactive controls are keyboard focusable	^
Custom interactive controls are keyboard focusable and display a focus indicator. Learn how to make custom controls focusable.	
<input type="radio"/> Interactive elements indicate their purpose and state	^
Interactive elements, such as links and buttons, should indicate their state and be distinguishable from non-interactive elements. Learn how to decorate interactive elements with affordance hints.	
<input type="radio"/> The page has a logical tab order	^
Tabbing through the page follows the visual layout. Users cannot focus elements that are offscreen. Learn more about logical tab ordering.	
<input type="radio"/> Visual order on the page follows DOM order	^
DOM order matches the visual order, improving navigation for assistive technology. Learn more about DOM and visual ordering.	
<input type="radio"/> User focus is not accidentally trapped in a region	^
A user can tab into and out of any control or region without accidentally trapping their focus. Learn how to avoid focus traps.	
<input type="radio"/> The user's focus is directed to new content added to the page	^
If new content, such as a dialog, is added to the page, the user's focus is directed to it. Learn how to direct focus to new content.	
<input type="radio"/> HTML5 landmark elements are used to improve navigation	^
Landmark elements (<main>, <nav>, etc.) are used to improve the keyboard navigation of the page for assistive technology. Learn more about landmark elements.	
<input type="radio"/> Offscreen content is hidden from assistive technology	^
Offscreen content is hidden with display: none or aria-hidden=true. Learn how to properly hide offscreen content.	
<input type="radio"/> Custom controls have associated labels	^
Custom interactive controls have associated labels, provided by aria-label or aria-labelledby. Learn more about custom controls and labels.	
<input type="radio"/> Custom controls have ARIA roles	^
Custom interactive controls have appropriate ARIA roles. Learn how to add roles to custom controls.	

These items address areas which an automated testing tool cannot cover. Learn more in our guide on [conducting an accessibility review.](#)

<code>[aria-hidden="true"]</code> is not present on the document <code><body></code>	^
Assistive technologies, like screen readers, work inconsistently when <code>aria-hidden="true"</code> is set on the document <code><body></code> . Learn how aria-hidden affects the document body.	
Buttons have an accessible name	^
When a button doesn't have an accessible name, screen readers announce it as "button", making it unusable for users who rely on screen readers. Learn how to make buttons more accessible.	
<code>[user-scalable="no"]</code> is not used in the <code><meta name="viewport"></code> element and the <code>[maximum-scale]</code> attribute is not less than 5.	^
Disabling zooming is problematic for users with low vision who rely on screen magnification to properly see the contents of a web page. Learn more about the viewport meta tag.	
Document has a <code><title></code> element	^
The title gives screen reader users an overview of the page, and search engine users rely on it heavily to determine if a page is relevant to their search. Learn more about document titles.	
<code><html></code> element has a <code>[lang]</code> attribute	^
If a page doesn't specify a <code>lang</code> attribute, a screen reader assumes that the page is in the default language that the user chose when setting up the screen reader. If the page isn't actually in the default language, then the screen reader might not announce the page's text correctly. Learn more about the lang attribute.	
<code><html></code> element has a valid value for its <code>[lang]</code> attribute	^
Specifying a valid BCP 47 language helps screen readers announce text properly. Learn how to use the lang attribute.	
Touch targets have sufficient size and spacing.	^
Touch targets with sufficient size and spacing help users who may have difficulty targeting small controls to activate the targets. Learn more about touch targets.	
Heading elements appear in a sequentially-descending order	^
Properly ordered headings that do not skip levels convey the semantic structure of the page, making it easier to navigate and understand when using assistive technologies. Learn more about heading order.	

<input type="radio"/> <code>[accesskey]</code> values are unique	^
Access keys let users quickly focus a part of the page. For proper navigation, each access key must be unique. Learn more about access keys.	

- ☐ `[aria-*)` attributes match their roles

Each ARIA role supports a specific subset of `aria-*` attributes. Mismatching these invalidates the `aria-*` attributes. [Learn how to match ARIA attributes to their roles.](#)

- ☐ Uses ARIA roles only on compatible elements

Many HTML elements can only be assigned certain ARIA roles. Using ARIA roles where they are not allowed can interfere with the accessibility of the web page. [Learn more about ARIA roles.](#)

- ☐ `button`, `link`, and `menuitem` elements have accessible names

When an element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. [Learn how to make command elements more accessible.](#)

- ☐ ARIA attributes are used as specified for the element's role

Some ARIA attributes are only allowed on an element under certain conditions. [Learn more about conditional ARIA attributes.](#)

- ☐ Deprecated ARIA roles were not used

Deprecated ARIA roles may not be processed correctly by assistive technology. [Learn more about deprecated ARIA roles.](#)

- ☐ Elements with `role="dialog"` or `role="alertdialog"` have accessible names.

ARIA dialog elements without accessible names may prevent screen readers users from discerning the purpose of these elements. [Learn how to make ARIA dialog elements more accessible.](#)

- ☐ `[aria-hidden="true"]` elements do not contain focusable descendents

Focusable descendents within an `[aria-hidden="true"]` element prevent those interactive elements from being available to users of assistive technologies like screen readers. [Learn how `aria-hidden` affects focusable elements.](#)

- ☐ ARIA input fields have accessible names

When an input field doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. [Learn more about input field labels.](#)

- ☐ ARIA `meter` elements have accessible names

When a meter element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. [Learn how to name meter elements.](#)

- ☐ ARIA `progressbar` elements have accessible names

When a progressbar element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. [Learn how to label progressbar elements.](#)

- ☐ Elements use only permitted ARIA attributes

Using ARIA attributes in roles where they are prohibited can mean that important information is not communicated to users of assistive technologies. [Learn more about prohibited ARIA roles.](#)

- ☐ [role]s have all required [aria-*] attributes

Some ARIA roles have required attributes that describe the state of the element to screen readers. [Learn more about roles and required attributes.](#)

- ☐ Elements with an ARIA [role] that require children to contain a specific [role] have all required children.

Some ARIA parent roles must contain specific child roles to perform their intended accessibility functions. [Learn more about roles and required children elements.](#)

- ☐ [role]s are contained by their required parent element

Some ARIA child roles must be contained by specific parent roles to properly perform their intended accessibility functions. [Learn more about ARIA roles and required parent element.](#)

- ☐ [role] values are valid

ARIA roles must have valid values in order to perform their intended accessibility functions. [Learn more about valid ARIA roles.](#)

- ☐ Elements with the role=text attribute do not have focusable descendents.

Adding role=text around a text node split by markup enables VoiceOver to treat it as one phrase, but the element's focusable descendents will not be announced. [Learn more about the role=text attribute.](#)

- ☐ ARIA toggle fields have accessible names

When a toggle field doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. [Learn more about toggle fields.](#)

- ☐ ARIA tooltip elements have accessible names

When a tooltip element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. [Learn how to name tooltip elements.](#)

- ☐ ARIA treeitem elements have accessible names

When a treeitem element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. [Learn more about labeling treeitem elements.](#)

- ☐ [aria-*] attributes have valid values


Assistive technologies, like screen readers, can't interpret ARIA attributes with invalid values. [Learn more about valid values for ARIA attributes.](#)


- ☐ [aria-*] attributes are valid and not misspelled

Assistive technologies, like screen readers, can't interpret ARIA attributes with invalid names. Learn more about valid ARIA attributes.	
<input type="radio"/> The page contains a heading, skip link, or landmark region	^
Adding ways to bypass repetitive content lets keyboard users navigate the page more efficiently. Learn more about bypass blocks.	
<input type="radio"/> <code><dl></code> 's contain only properly-ordered <code><dt></code> and <code><dd></code> groups, <code><script></code> , <code><template></code> or <code><div></code> elements.	^
When definition lists are not properly marked up, screen readers may produce confusing or inaccurate output. Learn how to structure definition lists correctly.	
<input type="radio"/> Definition list items are wrapped in <code><dl></code> elements	^
Definition list items (<code><dt></code> and <code><dd></code>) must be wrapped in a parent <code><dl></code> element to ensure that screen readers can properly announce them. Learn how to structure definition lists correctly.	
<input type="radio"/> ARIA IDs are unique	^
The value of an ARIA ID must be unique to prevent other instances from being overlooked by assistive technologies. Learn how to fix duplicate ARIA IDs.	
<input type="radio"/> No form fields have multiple labels	^
Form fields with multiple labels can be confusingly announced by assistive technologies like screen readers which use either the first, the last, or all of the labels. Learn how to use form labels.	
<input type="radio"/> <code><frame></code> or <code><iframe></code> elements have a title	^
Screen reader users rely on frame titles to describe the contents of frames. Learn more about frame titles.	
<input type="radio"/> <code><html></code> element has an <code>[xml:lang]</code> attribute with the same base language as the <code>[lang]</code> attribute.	^
If the webpage does not specify a consistent language, then the screen reader might not announce the page's text correctly. Learn more about the lang attribute.	
<input type="radio"/> Image elements have <code>[alt]</code> attributes	^
Informative elements should aim for short, descriptive alternate text. Decorative elements can be ignored with an empty alt attribute. Learn more about the alt attribute.	
<input type="radio"/> Image elements do not have <code>[alt]</code> attributes that are redundant text.	^
Informative elements should aim for short, descriptive alternative text. Alternative text that is exactly the same as the text adjacent to the link or image is potentially confusing for screen reader users, because the text will be read twice. Learn more about the alt attribute.	
<input type="radio"/> Input buttons have discernible text.	^

<p>Adding discernable and accessible text to input buttons may help screen reader users understand the purpose of the input button. Learn more about input buttons.</p>	
<p><input type="radio"/> <code><input type="image"></code> elements have <code>[alt]</code> text</p>	^
<p>When an image is being used as an <code><input></code> button, providing alternative text can help screen reader users understand the purpose of the button. Learn about input image alt text.</p>	
<p><input type="radio"/> Form elements have associated labels</p>	^
<p>Labels ensure that form controls are announced properly by assistive technologies, like screen readers. Learn more about form element labels.</p>	
<p><input type="radio"/> Links are distinguishable without relying on color.</p>	^
<p>Low-contrast text is difficult or impossible for many users to read. Link text that is discernible improves the experience for users with low vision. Learn how to make links distinguishable.</p>	
<p><input type="radio"/> Links have a discernible name</p>	^
<p>Link text (and alternate text for images, when used as links) that is discernible, unique, and focusable improves the navigation experience for screen reader users. Learn how to make links accessible.</p>	
<p><input type="radio"/> Lists contain only <code></code> elements and script supporting elements (<code><script></code> and <code><template></code>).</p>	^
<p>Screen readers have a specific way of announcing lists. Ensuring proper list structure aids screen reader output. Learn more about proper list structure.</p>	
<p><input type="radio"/> List items (<code></code>) are contained within <code></code>, <code></code> or <code><menu></code> parent elements</p>	^
<p>Screen readers require list items (<code></code>) to be contained within a parent <code></code>, <code></code> or <code><menu></code> to be announced properly. Learn more about proper list structure.</p>	
<p><input type="radio"/> The document does not use <code><meta http-equiv="refresh"></code></p>	^
<p>Users do not expect a page to refresh automatically, and doing so will move focus back to the top of the page. This may create a frustrating or confusing experience. Learn more about the refresh meta tag.</p>	
<p><input type="radio"/> <code><object></code> elements have alternate text</p>	^
<p>Screen readers cannot translate non-text content. Adding alternate text to <code><object></code> elements helps screen readers convey meaning to users. Learn more about alt text for object elements.</p>	
<p><input type="radio"/> Select elements have associated label elements.</p>	^
<p>Form elements without effective labels can create frustrating experiences for screen reader users. Learn more about the select element.</p>	
<p><input type="radio"/> Skip links are focusable.</p>	^
<p>Including a skip link can help users skip to the main content to save time. Learn more about skip links.</p>	

<input type="radio"/> No element has a <code>[tabindex]</code> value greater than 0	^
A value greater than 0 implies an explicit navigation ordering. Although technically valid, this often creates frustrating experiences for users who rely on assistive technologies. Learn more about the <code>tabindex</code> attribute.	
<input type="radio"/> Tables have different content in the summary attribute and <code><caption></code> .	^
The summary attribute should describe the table structure, while <code><caption></code> should have the onscreen title. Accurate table mark-up helps users of screen readers. Learn more about summary and caption.	
<input type="radio"/> Cells in a <code><table></code> element that use the <code>[headers]</code> attribute refer to table cells within the same table.	^
Screen readers have features to make navigating tables easier. Ensuring <code><td></code> cells using the <code>[headers]</code> attribute only refer to other cells in the same table may improve the experience for screen reader users. Learn more about the <code>headers</code> attribute.	
<input type="radio"/> <code><th></code> elements and elements with <code>[role="columnheader"/"rowheader"]</code> have data cells they describe.	^
Screen readers have features to make navigating tables easier. Ensuring table headers always refer to some set of cells may improve the experience for screen reader users. Learn more about table headers.	
<input type="radio"/> <code>[lang]</code> attributes have a valid value	^
Specifying a valid BCP 47 language on elements helps ensure that text is pronounced correctly by a screen reader. Learn how to use the <code>lang</code> attribute.	
<input type="radio"/> <code><video></code> elements contain a <code><track></code> element with <code>[kind="captions"]</code>	^
When a video provides a caption it is easier for deaf and hearing impaired users to access its information. Learn more about video captions.	

 Captured at Dec 3, 2024,
10:35 PM PST

 Initial page load

 Emulated Desktop with
Lighthouse 12.2.1

 Custom throttling

 Single page session

 Using Chromium 131.0.0.0
with devtools