http://localhost:3000/dashboard

94 Accessibility

91 SEO

▲ 0–49 ■ 50–89 ● 90–100

94

Accessibility

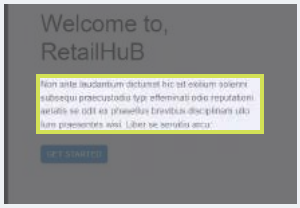
These checks highlight opportunities to improve the accessibility of your web app. Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so manual testing is also encouraged.

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.

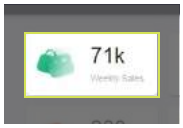
Failing Elements



p.MuiTypography-root.MuiTypography-body1.MuiTypography-alignLeft.css-zpphrz-MuiTypography-root

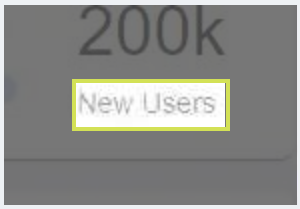
main.MuiBox-root.css-1yep2xe

h6.MuiTypography-root.MuiTypography-subtitle2.css-1kipjet-MuiTypography-root

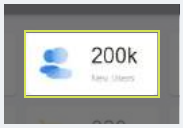


div.MuiStack-root.MuiPaper-root.MuiPaper-elevation.MuiPaper-rounded.MuiPaper-elevation1.paper2.css-mj7erh-MuiStack-root-MuiPaper-root

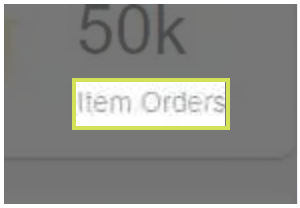
Failing Elements



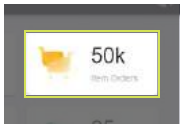
h6.MuiTypography-root.MuiTypography-subtitle2.css-1kipjet-MuiTypography-root



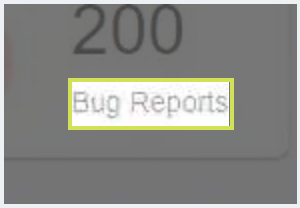
div.MuiStack-root.MuiPaper-root.MuiPaper-elevation.MuiPaper-rounded.MuiPaper-elevation1.paper2.css-mj7erh-MuiStack-root-MuiPaper-root



h6.MuiTypography-root.MuiTypography-subtitle2.css-1kipjet-MuiTypography-root



div.MuiStack-root.MuiPaper-root.MuiPaper-elevation.MuiPaper-rounded.MuiPaper-elevation1.paper2.css-mj7erh-MuiStack-root-MuiPaper-root

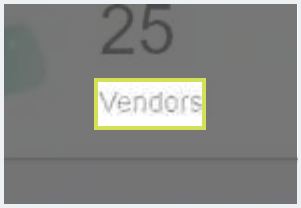


h6.MuiTypography-root.MuiTypography-subtitle2.css-1kipjet-MuiTypography-root

div.MuiStack-root.MuiPaper-root.MuiPaper-elevation.MuiPaper-rounded.MuiPaper-elevation1.paper2.css-mj7erh-MuiStack-root-MuiPaper-root

h6.MuiTypography-root.MuiTypography-subtitle2.css-1kipjet-MuiTypography-root

div.MuiStack-root.MuiPaper-root.MuiPaper-elevation.MuiPaper-rounded.MuiPaper-elevation1.paper2.css-mj7erh-MuiStack-root-MuiPaper-root



h6.MuiTypography-root.MuiTypography-subtitle2.css-1kipjet-MuiTypography-root

div.MuiStack-root.MuiPaper-root.MuiPaper-elevation.MuiPaper-rounded.MuiPaper-elevation1.paper2.css-mj7erh-MuiStack-root-MuiPaper-root

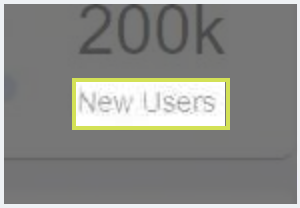
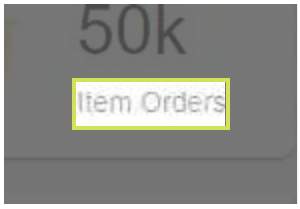
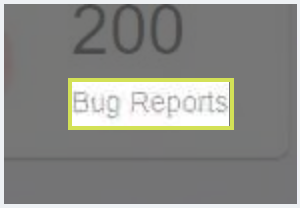
Failing Elements

These are opportunities to improve the legibility of your content.

NAVIGATION

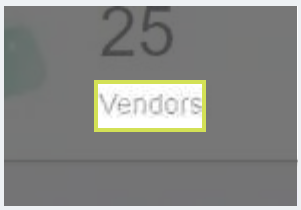
▲ Heading elements are not in a sequentially-descending order

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.

Failing Elements	
	h4.MuiTypography-root.MuiTypography-h4.css-5lbw0b-MuiTypography-root
	h6.MuiTypography-root.MuiTypography-subtitle2.css-1kipjet-MuiTypography-root
	h6.MuiTypography-root.MuiTypography-subtitle2.css-1kipjet-MuiTypography-root
	h6.MuiTypography-root.MuiTypography-subtitle2.css-1kipjet-MuiTypography-root
	h6.MuiTypography-root.MuiTypography-subtitle2.css-1kipjet-MuiTypography-root

Failing Elements

h6.MuiTypography-root.MuiTypography-subtitle2.css-1kipjet-MuiTypography-root



h6.MuiTypography-root.MuiTypography-subtitle2.css-1kipjet-MuiTypography-root

These are opportunities to improve keyboard navigation in your application.

ADDITIONAL ITEMS TO MANUALLY CHECK (10) Hide

- Interactive controls are keyboard focusable

  Custom interactive controls are keyboard focusable and display a focus indicator. Learn how to make custom controls focusable.

- 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.

- 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.

- 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.

- 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.

- 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.

- 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.

- 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.

- 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.

- 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.

PASSED AUDITS (20) Hide

- `[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.

- `[aria-hidden="true"]` is not present on the document `<body>`

  Assistive technologies, like screen readers, work inconsistently when `aria-hidden="true"` is set on the document `<body>`. Learn how `aria-hidden` affects the document body.

- `[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.

- `[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.

- 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.

- Image elements have `[alt]` 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.

- `[user-scalable="no"]` is not used in the `<meta name="viewport">` element and the `[maximum-scale]` 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.

- `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-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.

- `[role]` values are valid

ARIA roles must have valid values in order to perform their intended accessibility functions. Learn more about valid ARIA roles.

- Document has a `<title>` 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.

- `<html>` element has a `[lang]` attribute

If a page doesn't specify a `lang` 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.

- `<html>` element has a valid value for its `[lang]` attribute

Specifying a valid BCP 47 language helps screen readers announce text properly. Learn how to use the `lang` attribute.

- Links have a discernible name

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.

- Lists contain only `<li>` elements and script supporting elements (`<script>` and `<template>`).

Screen readers have a specific way of announcing lists. Ensuring proper list structure aids screen reader output. Learn more about proper list structure.

- List items (`<li>`) are contained within `<ul>`, `<ol>` or `<menu>` parent elements

Screen readers require list items (`<li>`) to be contained within a parent `<ul>`, `<ol>` or `<menu>` to be announced properly. Learn more about proper list structure.

- No element has a `[tabindex]` 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 `tabindex` attribute.

- Values assigned to `role=""` are valid ARIA roles.

ARIA `roles` enable assistive technologies to know the role of each element on the web page. If the `role` values are misspelled, not existing ARIA `role` values, or abstract roles, then the purpose of the element will not be communicated to users of assistive technologies. Learn more about ARIA roles.

- Image elements do not have `[alt]` 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.

NOT APPLICABLE (39) Hide

- `[accesskey]` 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.

- 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 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 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.

- 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.

- 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.

- `<dl>`'s contain only properly-ordered `<dt>` and `<dd>` groups, `<script>`, `<template>` or `<div>` elements.

  When definition lists are not properly marked up, screen readers may produce confusing or inaccurate output. Learn how to structure definition lists correctly.

- Definition list items are wrapped in `<dl>` elements

  Definition list items (`<dt>` and `<dd>`) must be wrapped in a parent `<dl>` element to ensure that screen readers can properly announce them. Learn how to structure definition lists correctly.

- `[id]` attributes on active, focusable elements are unique

  All focusable elements must have a unique `id` to ensure that they're visible to assistive technologies. Learn how to fix duplicate `ids`.

- 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.

- 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.

- `<frame>` or `<iframe>` elements have a title

  Screen reader users rely on frame titles to describe the contents of frames. Learn more about frame titles.

- `<html>` element has an `[xml:lang]` attribute with the same base language as the `[lang]` 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 buttons have discernible text.

  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.

- `<input type="image">` elements have `[alt]` text

  When an image is being used as an `<input>` button, providing alternative text can help screen reader users understand the purpose of the button. Learn about input image alt text.

- Form elements have associated labels

  Labels ensure that form controls are announced properly by assistive technologies, like screen readers. Learn more about form element labels.

- Links are distinguishable without relying on color.

  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.

- The document does not use `<meta http-equiv="refresh">`

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.

- `<object>` elements have alternate text

Screen readers cannot translate non-text content. Adding alternate text to `<object>` elements helps screen readers convey meaning to users. Learn more about alt text for `object` elements.

- Select elements have associated label elements.

Form elements without effective labels can create frustrating experiences for screen reader users. Learn more about the `select` element.

- Skip links are focusable.

Including a skip link can help users skip to the main content to save time. Learn more about skip links.

- Tables have different content in the summary attribute and `<caption>`.

The summary attribute should describe the table structure, while `<caption>` should have the onscreen title. Accurate table mark-up helps users of screen readers. Learn more about summary and caption.

- Cells in a `<table>` element that use the `[headers]` attribute refer to table cells within the same table.

Screen readers have features to make navigating tables easier. Ensuring `<td>` cells using the `[headers]` attribute only refer to other cells in the same table may improve the experience for screen reader users. Learn more about the `headers` attribute.

- `<th>` elements and elements with `[role="columnheader"/"rowheader"]` 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.

- `[lang]` 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 `lang` attribute.

- `<video>` elements contain a `<track>` element with `[kind="captions"]`

When a video provides a caption it is easier for deaf and hearing impaired users to access its information. Learn more about video captions.

- All heading elements contain content.

A heading with no content or inaccessible text prevent screen reader users from accessing information on the page's structure. Learn more about headings.

- Identical links have the same purpose.

Links with the same destination should have the same description, to help users understand the link's purpose and decide whether to follow it. Learn more about identical links.

- Document has a main landmark.

One main landmark helps screen reader users navigate a web page. Learn more about landmarks.

- 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.

- Elements with visible text labels have matching accessible names.

Visible text labels that do not match the accessible name can result in a confusing experience for screen reader users. Learn more about accessible names.

- Tables use `<caption>` instead of cells with the `[colspan]` attribute to indicate a caption.

Screen readers have features to make navigating tables easier. Ensuring that tables use the actual caption element instead of cells with the `[colspan]` attribute may improve the experience for screen reader users. Learn more about captions.

- `<td>` elements in a large `<table>` have one or more table headers.

Screen readers have features to make navigating tables easier. Ensuring that `<td>` elements in a large table (3 or more cells in width and height) have an associated table header may improve the experience for screen reader users. Learn more about table headers.

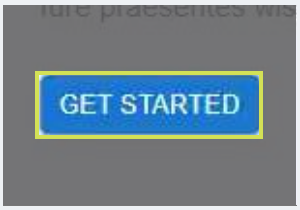
SEO

These checks ensure that your page is following basic search engine optimization advice. There are many additional factors Lighthouse does not score here that may affect your search ranking, including performance on Core Web Vitals. Learn more about Google Search Essentials.

CRAWLING AND INDEXING

Links are not crawlable

Search engines may use `href` attributes on links to crawl websites. Ensure that the `href` attribute of anchor elements links to an appropriate destination, so more pages of the site can be discovered. Learn how to make links crawlable

Uncrawlable Link	
	a.MuiButtonBase-root.MuiButton-root.MuiButton-contained.MuiButton-containedPrimary.MuiButton-sizeSmall.MuiButton-containedSizeSmall.MuiButton-colorPrimary.css-11qr2p8-MuiButtonBase-root-MuiButton-root

To appear in search results, crawlers need access to your app.

ADDITIONAL ITEMS TO MANUALLY CHECK (1) Hide

Structured data is valid

Run the Structured Data Testing Tool and the Structured Data Linter to validate structured data. Learn more about Structured Data.

Run these additional validators on your site to check additional SEO best practices.

PASSED AUDITS (10) Hide

Has a `<meta name="viewport">` tag with `width` or `initial-scale`

A `<meta name="viewport">` not only optimizes your app for mobile screen sizes, but also prevents a 300 millisecond delay to user input. Learn more about using the viewport meta tag.

Document has a `<title>` 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.

- Document has a meta description

Meta descriptions may be included in search results to concisely summarize page content. Learn more about the meta description.

- Page has successful HTTP status code

Pages with unsuccessful HTTP status codes may not be indexed properly. Learn more about HTTP status codes.

- Links have descriptive text

Descriptive link text helps search engines understand your content. Learn how to make links more accessible.

- Page isn’t blocked from indexing

Search engines are unable to include your pages in search results if they don't have permission to crawl them. Learn more about crawler directives.

- robots.txt is valid

If your robots.txt file is malformed, crawlers may not be able to understand how you want your website to be crawled or indexed. Learn more about robots.txt.

- Image elements have `[alt]` 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.

- Document has a valid `hreflang`

hreflang links tell search engines what version of a page they should list in search results for a given language or region. Learn more about `hreflang`.

- Document avoids plugins

Search engines can't index plugin content, and many devices restrict plugins or don't support them. Learn more about avoiding plugins.

NOT APPLICABLE (3) Hide

- Document has a valid `rel=canonical`

Canonical links suggest which URL to show in search results. Learn more about canonical links.

- Document uses legible font sizes

Font sizes less than 12px are too small to be legible and require mobile visitors to “pinch to zoom” in order to read. Strive to have >60% of page text ≥12px. Learn more about legible font sizes.

- Tap targets are sized appropriately

Interactive elements like buttons and links should be large enough (48x48px), or have enough space around them, to be easy enough to tap without overlapping onto other elements. Learn more about tap targets.

Captured at Apr 21, 2024, 6:39 PM EDT
Initial page load
Emulated Desktop with Lighthouse 11.5.0
Custom throttling
Single page session
Using Chromium 123.0.0.0 with devtools

Generated by **Lighthouse** 11.5.0 | File an issue

http://localhost:3000/inventory

100

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.

ADDITIONAL ITEMS TO MANUALLY CHECK (10)

Hide

- Interactive controls are keyboard focusable

  Custom interactive controls are keyboard focusable and display a focus indicator. Learn how to make custom controls focusable.

- 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.

- 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.

- 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.

- 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.

- 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.

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.

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.

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.

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.

PASSED AUDITS (21) Hide

`[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.

`[aria-hidden="true"]` is not present on the document `<body>`

Assistive technologies, like screen readers, work inconsistently when `aria-hidden="true"` is set on the document `<body>`. Learn how `aria-hidden` affects the document body.

`[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.

`[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.

- 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.

- Image elements have `[alt]` 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.

- `[user-scalable="no"]` is not used in the `<meta name="viewport">` element and the `[maximum-scale]` 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.

- `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-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.

- `[role]` values are valid

ARIA roles must have valid values in order to perform their intended accessibility functions. Learn more about valid ARIA roles.

- Background and foreground colors have a sufficient contrast ratio

Low-contrast text is difficult or impossible for many users to read. Learn how to provide sufficient color contrast.

- Document has a `<title>` 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.

- `<html>` element has a `[lang]` attribute

If a page doesn't specify a `lang` 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.

- `<html>` element has a valid value for its `[lang]` attribute

Specifying a valid BCP 47 language helps screen readers announce text properly. Learn how to use the `lang` attribute.

- Links have a discernible name

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.

- Lists contain only `<li>` elements and script supporting elements (`<script>` and `<template>`).

Screen readers have a specific way of announcing lists. Ensuring proper list structure aids screen reader output. Learn more about proper list structure.

- List items (`<li>`) are contained within `<ul>`, `<ol>` or `<menu>` parent elements

Screen readers require list items (`<li>`) to be contained within a parent `<ul>`, `<ol>` or `<menu>` to be announced properly. Learn more about proper list structure.

- No element has a `[tabindex]` 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 `tabindex` attribute.

- Values assigned to `role=""` are valid ARIA roles.

ARIA `roles` enable assistive technologies to know the role of each element on the web page. If the `role` values are misspelled, not existing ARIA `role` values, or abstract roles, then the purpose of the element will not be communicated to users of assistive technologies. Learn more about ARIA roles.

- Image elements do not have `[alt]` 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.

NOT APPLICABLE (40) Hide

- `[accesskey]` 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.

- 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 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 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.

- 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.

- 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.

- `<dl>`'s contain only properly-ordered `<dt>` and `<dd>` groups, `<script>`, `<template>` or `<div>` elements.

  When definition lists are not properly marked up, screen readers may produce confusing or inaccurate output. Learn how to structure definition lists correctly.

- Definition list items are wrapped in `<dl>` elements

  Definition list items (`<dt>` and `<dd>`) must be wrapped in a parent `<dl>` element to ensure that screen readers can properly announce them. Learn how to structure definition lists correctly.

- `[id]` attributes on active, focusable elements are unique

  All focusable elements must have a unique `id` to ensure that they're visible to assistive technologies. Learn how to fix duplicate `ids`.

- 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.

- 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.

- `<frame>` or `<iframe>` elements have a title

Screen reader users rely on frame titles to describe the contents of frames. Learn more about frame titles.

- 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.

- `<html>` element has an `[xml:lang]` attribute with the same base language as the `[lang]` 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 buttons have discernible text.

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.

- `<input type="image">` elements have `[alt]` text

When an image is being used as an `<input>` button, providing alternative text can help screen reader users understand the purpose of the button. Learn about input image alt text.

- Form elements have associated labels

Labels ensure that form controls are announced properly by assistive technologies, like screen readers. Learn more about form element labels.

- Links are distinguishable without relying on color.

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.

- The document does not use `<meta http-equiv="refresh">`

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.

- `<object>` elements have alternate text

Screen readers cannot translate non-text content. Adding alternate text to `<object>` elements helps screen readers convey meaning to users. Learn more about alt text for `object` elements.

- Select elements have associated label elements.

Form elements without effective labels can create frustrating experiences for screen reader users. Learn more about the `select` element.

- Skip links are focusable.

Including a skip link can help users skip to the main content to save time. Learn more about skip links.

- Tables have different content in the summary attribute and `<caption>`.

The summary attribute should describe the table structure, while `<caption>` should have the onscreen title. Accurate table mark-up helps users of screen readers. Learn more about summary and caption.

- Cells in a `<table>` element that use the `[headers]` attribute refer to table cells within the same table.

Screen readers have features to make navigating tables easier. Ensuring `<td>` cells using the `[headers]` attribute only refer to other cells in the same table may improve the experience for screen reader users. Learn more about the `headers` attribute.

- `<th>` elements and elements with `[role="columnheader"/"rowheader"]` 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.

- `[lang]` 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 `lang` attribute.

- `<video>` elements contain a `<track>` element with `[kind="captions"]`

When a video provides a caption it is easier for deaf and hearing impaired users to access its information. Learn more about video captions.

- All heading elements contain content.

A heading with no content or inaccessible text prevent screen reader users from accessing information on the page's structure. Learn more about headings.

- Identical links have the same purpose.

Links with the same destination should have the same description, to help users understand the link's purpose and decide whether to follow it. Learn more about identical links.

- Document has a main landmark.

One main landmark helps screen reader users navigate a web page. Learn more about landmarks.

- 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.

- Elements with visible text labels have matching accessible names.

Visible text labels that do not match the accessible name can result in a confusing experience for screen reader users. Learn more about accessible names.

- Tables use `<caption>` instead of cells with the `[colspan]` attribute to indicate a caption.

Screen readers have features to make navigating tables easier. Ensuring that tables use the actual caption element instead of cells with the `[colspan]` attribute may improve the experience for screen reader users. Learn more about captions.

- `<td>` elements in a large `<table>` have one or more table headers.

Screen readers have features to make navigating tables easier. Ensuring that `<td>` elements in a large table (3 or more cells in width and height) have an associated table header may improve the experience for screen reader users. Learn more about table headers.

Captured at Apr 21, 2024, 6:46 PM EDT | Emulated Desktop with Lighthouse 11.5.0 | Single page session

Initial page load | Custom throttling | Using Chromium 123.0.0.0 with devtools

Generated by **Lighthouse** 11.5.0 | File an issue

http://localhost:3000/vendors

86

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.

ARIA

- Elements with an ARIA `[role]` that require children to contain a specific `[role]` are missing some or all of those required children.
- ARIA `progressbar` elements do not have accessible names.

These are opportunities to improve the usage of ARIA in your application which may enhance the experience for users of assistive technology, like a screen reader.

CONTRAST

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

These are opportunities to improve the legibility of your content.

NAVIGATION

- Heading elements are not in a sequentially-descending order

These are opportunities to improve keyboard navigation in your application.

ADDITIONAL ITEMS TO MANUALLY CHECK (10)

Show

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

PASSED AUDITS (22)

Show

NOT APPLICABLE (35) Show

Captured at Apr 21, 2024, 6:47 PM EDT
Initial page load
Emulated Desktop with Lighthouse 11.5.0
Custom throttling
Single page session
Using Chromium 123.0.0.0 with devtools

Generated by **Lighthouse** 11.5.0 | File an issue