Tillgänglighetsrapport 2024-06-26

Genomfört av: Toby Morey, Webmaster (webmaster@opera.se)

Sidorna listade nedanför testades både med automatiserade verktyg samt manuella tester där testaren använde ett tangentbord för navigering, samt hade en talsyntes aktiverat under testet.

Verktyg använt i testet

- Lighthouse: https://developers.google.com/web/tools/lighthouse
- WAVE: https://wave.webaim.org/
- NVDA, en talsyntes av NV Access: https://www.nvaccess.org/
- Matomo, för analys av webbtrafik: https://matomo.org/

Sidor som testades

Listan över sidorna nedanför täcker samtliga sidmallar som finns på GöteborgsOperans publikwebbplats opera.se vid skrivande stund. Sidorna som har haft mest trafik under perioden sedan förra test valdes som exempel där mallen har fler exempel på webbplatsen, för att mäta påverkan på flest antal webbplatsbesökare. Antaganden är att sidor som bygger på samma mall beter sig på ett väldigt liknande sätt till exempelsidan.

1. **Startsidan** *(Mall: startsida)*
   https://www.opera.se/

2. **Biljetter > Abonnemang** *(Mall: landningssida)*
   https://www.opera.se/biljetter/abonnemang/

3. **Föreställningar** *(Mall: föreställningslistning)*
   https://www.opera.se/forestallningar/

4. **Kalender** *(Mall: informationssida med komplex kalenderlistning)*
   https://www.opera.se/forestallningar/kalender/

5. **Vault of heaven**, dans *(Mall: föreställningssida som presenterar en evenemang med fler verk)*
   https://www.opera.se/forestallningar/sasong-2023-2024/vaults-of-heaven/

6. **Wicked**, musikal *(Mall: föreställningssida med mycket information)*
   https://www.opera.se/forestallningar/sasong-2023-2024/wicked/

7. **Mat och dryck > I pausen** *(Mall: informationssida)*
   https://www.opera.se/mat-och-dryck/i-pausen/

8. **Feline Andersson** *(Mall: profilsida)*
   https://www.opera.se/opera-musikal-och-konsert/gaster/feline-andersson/

9. **Sök** *(Mall: Sökresultat som visar paginering)*
   https://www.opera.se/sok/?q=charlie&page=1

Utskrifter från testerna genomfört med verktyget Lighthouse bifogas på följande sidor.

https://www.opera.se/

⋮

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.

AUDIO AND VIDEO

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

These are opportunities to provide alternative content for audio and video. This may improve the experience for users with hearing or vision impairments.

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 (28)   Show

NOT APPLICABLE (28)   Show

■ Captured at Jun 27, 2024, 10:21 AM GMT+2

■ Initial page load

■ Emulated Desktop with Lighthouse 12.0.0

■ Custom throttling

■ Single page session

■ Using Chromium 126.0.0.0 with devtools

Generated by **Lighthouse** 12.0.0 | File an issue

https://www.opera.se/biljetter/abonnemang/ ⋮

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)   Show

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

PASSED AUDITS (27)   Show

NOT APPLICABLE (30)   Show

- Captured at Jun 27, 2024, 10:28 AM GMT+2
- Initial page load
- Emulated Desktop with Lighthouse 12.0.0
- Custom throttling
- Single page session
- Using Chromium 126.0.0.0 with devtools

Generated by **Lighthouse** 12.0.0 | [File an issue](#)

https://www.opera.se/forestallningar/ ⋮

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.

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.sc-iBPRYJ.emocBb

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 (26)   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](#).

● 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](#).

● `[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](#).

● 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]` 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](#).

●   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`](#).

●   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](#).

●   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](#).

●   Skip links are focusable.   ⌃

Including a skip link can help users skip to the main content to save time. [Learn more about skip links](#).

●   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](#).

●   Deprecated ARIA roles were not used   ⌃

Deprecated ARIA roles may not be processed correctly by assistive technology. [Learn more about deprecated 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 (30)   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](#).

○   `button`, `link`, and `menuitem` elements have accessible names   ⌃

https://www.opera.se/forestallningar/kalender/　　　　　　⋮

98

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.

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 |
> | --- |
> | h3.sc-iBPRYJ.cMqrMS |

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 (26)   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](#).

● 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](#).

● `[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](#).

● 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]` 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.

●   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`](#).

●   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](#).

●   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](#).

●   Skip links are focusable.   ⌃

Including a skip link can help users skip to the main content to save time. [Learn more about skip links](#).

●   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](#).

●   Deprecated ARIA roles were not used   ⌃

Deprecated ARIA roles may not be processed correctly by assistive technology. [Learn more about deprecated 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 (30)   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](#).

○   `button`, `link`, and `menuitem` elements have accessible names   ⌃

https://www.opera.se/forestallningar/sasong-2023-2024/vaults-of-heaven/   ⋮

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)   Show

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

PASSED AUDITS (28)   Show

NOT APPLICABLE (29)   Show

■ Captured at Jun 27, 2024, 1:41 PM GMT+2   ■ Emulated Desktop with Lighthouse 12.0.0   ■ Single page session

■ Initial page load   ■ Custom throttling   ■ Using Chromium 126.0.0.0 with devtools

Generated by **Lighthouse** 12.0.0 | [File an issue]()

https://www.opera.se/forestallningar/sasong-2023-2024/wicked/

⋮

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)   Show

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

PASSED AUDITS (28)   Show

NOT APPLICABLE (29)   Show

■ Captured at Jun 26, 2024, 11:14 AM GMT+2

■ Initial page load

■ Emulated Desktop with Lighthouse 12.0.0

■ Custom throttling

■ Single page session

■ Using Chromium 126.0.0.0 with devtools

Generated by **Lighthouse** 12.0.0 | [File an issue]()

https://www.opera.se/mat-och-dryck/i-pausen/ ⋮

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)   Show

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

PASSED AUDITS (28)   Show

NOT APPLICABLE (29)   Show

■ Captured at Jun 27, 2024, 1:43 PM GMT+2	■ Emulated Desktop with Lighthouse 12.0.0	■ Single page session
■ Initial page load	■ Custom throttling	■ Using Chromium 126.0.0.0 with devtools

Generated by **Lighthouse** 12.0.0 | [File an issue](#)

https://www.opera.se/opera-musikal-och-konsert/gaster/feline-andersson/

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) Show

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

PASSED AUDITS (27) Show

NOT APPLICABLE (30) Show

Captured at Jun 27, 2024, 1:44 PM GMT+2

Initial page load

Emulated Desktop with Lighthouse 12.0.0

Custom throttling

Single page session

Using Chromium 126.0.0.0 with devtools

Generated by **Lighthouse** 12.0.0 | File an issue

🗼 https://www.opera.se/sok/?q=charlie&page=1 ⋮

96

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.

BEST PRACTICES

🔺 **Touch targets do not have sufficient size or 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]().

Failing Elements
⊠   button.sc-Fyfyc.cXQJO
[img]   input#search.sc-jXktwP.eoJlhn

These items highlight common accessibility best practices.

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 (27)   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](#).

● 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](#).

- `[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](#).

- 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]` 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](#).

- 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 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](#).

●   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](#).

●   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](#).

●   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](#).

●   Skip links are focusable.   ^

Including a skip link can help users skip to the main content to save time. [Learn more about skip links](#).

●   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](#).

●   Deprecated ARIA roles were not used   ^

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