Nieuws en werkinstructies

https://dev.kiss-demo.nl/

Accessibility

These checks highlight opportunities to improve the accessibility of your web app. Only a subset of accessibility issues can be automatically detected so manual testing is also encouraged.

ADDITIONAL ITEMS TO MANUALLY CHECK (10) Hide

- 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.
- 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 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.
- 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.
- 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.
- 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.

- 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.

- 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.

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.

- [role] values are valid

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

- [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.

- 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.

- 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.

- `[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-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.

- 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 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.

- ● 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.

- ● 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.

NOT APPLICABLE (35) 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

  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.

- 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`.

- 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.

- 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.

- 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.

- 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.

- 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.

- 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.

- 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 activate the targets. Learn more about touch targets.

Klantbeeld > Tab contactmomenten

https://dev.kiss-demo.nl/personen/1561a8f4-0d7d-48df-8bf1-e6cf23afc9e5

20/21

Accessibility

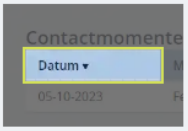
These checks highlight opportunities to improve the accessibility of your web app. Only a subset of accessibility issues can be automatically detected so manual testing is also encouraged.

ARIA

▲ ARIA IDs are not 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.

Failing Elements



Datum

```
<span id="datum-header" class="icon-after sort-descending">
```

Datum

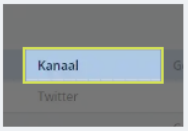
```
<span data-v-5c02104e="" id="datum-header" class="icon-after sort-descending">
```

Medewerker

```
<span id="medewerker-header">
```

Status

```
<span data-v-5c02104e="" id="medewerker-header">
```



Kanaal

```
<span id="kanaal-header">
```

Behandelaar

```
<span data-v-5c02104e="" id="kanaal-header">
```

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.

ADDITIONAL ITEMS TO MANUALLY CHECK (10) Hide

- 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.

- 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 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.

- 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.

- 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.

- 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.

- 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.

- 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.

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.

- 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.

- [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.

- 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.

- `[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.

- 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.

- `[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 `id`s.

- `<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.

- 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.

NOT APPLICABLE (35) 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

  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.

- 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 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.

- 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.

- 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.

- 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.

- 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.

- 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.

- 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.

- 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 activate the targets. Learn more about touch targets.

Personen zoeken > Zoekresultaten

https://dev.kiss-demo.nl/personen

23/23

Accessibility

These checks highlight opportunities to improve the accessibility of your web app. Only a subset of accessibility issues can be automatically detected so manual testing is also encouraged.

ADDITIONAL ITEMS TO MANUALLY CHECK (10) Hide

- 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.

- 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 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.

- 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.

- 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.

- 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.

- 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.

- 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.

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

PASSED AUDITS (23) 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.

- 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.

- [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.

- 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.

- 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.

- [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.

- 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.

- `[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`.

- `<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.

- 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.

- 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.

- 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.

NOT APPLICABLE (33) 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

  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.

- 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 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.

- 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.

- 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.

- 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.

- 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.

- 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.

- `<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.

- `<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 activate the targets. Learn more about touch targets.