SVELTE • REFERENCE

Compiler warnings

Svelte warns you at compile time if it catches potential mistakes, such as writing inaccessible markup.

Some warnings may be incorrect in your concrete use case. You can disable such false positives by placing a `<!-- svelte-ignore <code> -->` comment above the line that causes the warning. Example:

```
<!-- svelte-ignore a11y_autofocus -->
<input autofocus />
```

You can list multiple rules in a single comment (separated by commas), and add an explanatory note (in parentheses) alongside them:

```
<!-- svelte-ignore a11y_click_events_have_key_events, a11y_no_static_element_interact
<div onclick>...</div>
```

a11y_accesskey

```
Avoid using accesskey
```

Enforce no `accesskey` on element. Access keys are HTML attributes that allow web developers to assign keyboard shortcuts to elements. Inconsistencies between keyboard shortcuts and keyboard commands used by screen reader and keyboard-only users create accessibility complications. To avoid complications, access keys should not be used.

```
<!-- A11y: Avoid using accesskey -->
<div accesskey="z"></div>
```

```
An element with an aria-activedescendant attribute should have a tabindex value
```

An element with `aria-activedescendant` must be tabbable, so it must either have an inherent `tabindex` or declare `tabindex` as an attribute.

```
<!-- A11y: Elements with attribute aria-activedescendant should have tabindex value -
<div aria-activedescendant="some-id"></div>
```

a11y_aria_attributes

```
`<%name%>` should not have aria-* attributes
```

Certain reserved DOM elements do not support ARIA roles, states and properties. This is often because they are not visible, for example `meta`, `html`, `script`, `style`. This rule enforces that these DOM elements do not contain the `aria-*` props.

```
<!-- A11y: <meta> should not have aria-* attributes -->
<meta aria-hidden="false" />
```

a11y_autocomplete_valid

```
'%value%' is an invalid value for 'autocomplete' on `<input type="%type%">`
```

a11y_autofocus

```
Avoid using autofocus
```

Enforce that `autofocus` is not used on elements. Autofocusing elements can cause usability issues for sighted and non-sighted users alike.

Docs

a11y_click_events_have_key_events

```
Visible, non-interactive elements with a click event must be accompanied by a keyboard ev
```

Enforce that visible, non-interactive elements with an `onclick` event are accompanied by a keyboard event handler.

Users should first consider whether an interactive element might be more appropriate such as a `<button type="button">` element for actions or `<a>` element for navigations. These elements are more semantically meaningful and will have built-in key handling. E.g. `Space` and `Enter` will trigger a `<button>` and `Enter` will trigger an `<a>` element.

If a non-interactive element is required then `onclick` should be accompanied by an `onkeyup` or `onkeydown` handler that enables the user to perform equivalent actions via the keyboard. In order for the user to be able to trigger a key press, the element will also need to be focusable by adding a `tabindex` . While an `onkeypress` handler will also silence this warning, it should be noted that the `keypress` event is deprecated.

```
<!-- A11y: visible, non-interactive elements with an onclick event must be accompanie
<div onclick={() => {}}></div>
```

Coding for the keyboard is important for users with physical disabilities who cannot use a mouse, AT compatibility, and screenreader users.

a11y_consider_explicit_label

```
Buttons and links should either contain text or have an `aria-label` or `aria-labelledby`
```

a11y_distracting_elements

Docs

Enforces that no distracting elements are used. Elements that can be visually distracting can cause accessibility issues with visually impaired users. Such elements are most likely deprecated, and should be avoided.

The following elements are visually distracting: `<marquee>` and `<blink>`.

```
<!-- A11y: Avoid <marquee> elements -->
<marquee></marquee>
```

a11y_figcaption_index

```
`<figcaption>` must be first or last child of `<figure>`
```

a11y_figcaption_parent

```
`<figcaption>` must be an immediate child of `<figure>`
```

Enforce that certain DOM elements have the correct structure.

```
<!-- A11y: <figcaption> must be an immediate child of <figure> -->
<div>
  <figcaption>Image caption</figcaption>
</div>
```

a11y_hidden

```
`<%name%>` element should not be hidden
```

Certain DOM elements are useful for screen reader navigation and should not be hidden.

Docs

a11y_img_redundant_alt

```
Screenreaders already announce `<img>` elements as an image
```

Enforce img alt attribute does not contain the word image, picture, or photo. Screen readers already announce `img` elements as an image. There is no need to use words such as *image*, *photo*, and/or *picture*.

```
<img src="foo" alt="Foo eating a sandwich." />

<!-- aria-hidden, won't be announced by screen reader -->
<img src="bar" aria-hidden="true" alt="Picture of me taking a photo of an image" />

<!-- A11y: Screen readers already announce <img> elements as an image. -->
<img src="foo" alt="Photo of foo being weird." />

<!-- A11y: Screen readers already announce <img> elements as an image. -->
<img src="bar" alt="Image of me at a bar!" />

<!-- A11y: Screen readers already announce <img> elements as an image. -->
<img src="foo" alt="Picture of baz fixing a bug." />
```

a11y_incorrect_aria_attribute_type

```
The value of '%attribute%' must be a %type%
```

Enforce that only the correct type of value is used for aria attributes. For example, `aria-hidden` should only receive a boolean.

```
<!-- A11y: The value of 'aria-hidden' must be exactly one of true or false -->
<div aria-hidden="yes"></div>
```

```
The value of '%attribute%' must be either 'true' or 'false'. It cannot be empty
```

a11y_incorrect_aria_attribute_type_id

```
The value of '%attribute%' must be a string that represents a DOM element ID
```

a11y_incorrect_aria_attribute_type_idlist

```
The value of '%attribute%' must be a space-separated list of strings that represent DOM e
```

a11y_incorrect_aria_attribute_type_integer

```
The value of '%attribute%' must be an integer
```

a11y_incorrect_aria_attribute_type_token

```
The value of '%attribute%' must be exactly one of %values%
```

a11y_incorrect_aria_attribute_type_tokenlist

```
The value of '%attribute%' must be a space-separated list of one or more of %values%
```

a11y_incorrect_aria_attribute_type_tristate

```
The value of '%attribute%' must be exactly one of true, false, or mixed
```

a11y_interactive_supports_focus

Enforce that elements with an interactive role and interactive handlers (mouse or key press) must be focusable or tabbable.

```
<!-- A11y: Elements with the 'button' interactive role must have a tabindex value. --
<div role="button" onkeypress={() => {}} />
```

a11y_invalid_attribute

```
'%href_value%' is not a valid %href_attribute% attribute
```

Enforce that attributes important for accessibility have a valid value. For example, `href` should not be empty, `'#'`, or `javascript:`.

```
<!-- A11y: '' is not a valid href attribute -->
<a href="">invalid</a>
```

a11y_label_has_associated_control

```
A form label must be associated with a control
```

Enforce that a label tag has a text label and an associated control.

There are two supported ways to associate a label with a control:

> Wrapping a control in a label tag.
>
> Adding `for` to a label and assigning it the ID of an input on the page.

```
<label for="id">B</label>

<label>C <input type="text" /></label>
```

a11y_media_has_caption

```
`<video>` elements must have a `<track kind="captions">`
```

Providing captions for media is essential for deaf users to follow along. Captions should be a transcription or translation of the dialogue, sound effects, relevant musical cues, and other relevant audio information. Not only is this important for accessibility, but can also be useful for all users in the case that the media is unavailable (similar to `alt` text on an image when an image is unable to load).

The captions should contain all important and relevant information to understand the corresponding media. This may mean that the captions are not a 1:1 mapping of the dialogue in the media content. However, captions are not necessary for video components with the `muted` attribute.

```
<video><track kind="captions" /></video>

<audio muted></audio>

<!-- A11y: Media elements must have a <track kind=\"captions\"> -->
<video></video>

<!-- A11y: Media elements must have a <track kind=\"captions\"> -->
<video><track /></video>
```

a11y_misplaced_role

```
`<%name%>` should not have role attribute
```

Certain reserved DOM elements do not support ARIA roles, states and properties. This is often because they are not visible, for example `meta`, `html`, `script`, `style`. This rule enforces that these DOM elements do not contain the `role` props.

Docs

a11y_misplaced_scope

```
The scope attribute should only be used with `<th>` elements
```

The scope attribute should only be used on `<th>` elements.

```
<!-- A11y: The scope attribute should only be used with <th> elements -->
<div scope="row" />
```

a11y_missing_attribute

```
`<%name%>` element should have %article% %sequence% attribute
```

Enforce that attributes required for accessibility are present on an element. This includes the following checks:

- `<a>` should have an href (unless it's a fragment-defining tag)
- `<area>` should have alt, aria-label, or aria-labelledby
- `<html>` should have lang
- `<iframe>` should have title
- `<img>` should have alt
- `<object>` should have title, aria-label, or aria-labelledby
- `<input type="image">` should have alt, aria-label, or aria-labelledby

```
<!-- A11y: <input type=\"image\"> element should have an alt, aria-label or aria-labe
<input type="image" />

<!-- A11y: <html> element should have a lang attribute -->
<html></html>
```

```
<a>text</a>
```

a11y_missing_content

```
`<%name%>` element should contain text
```

Enforce that heading elements (`h1` , `h2` , etc.) and anchors have content and that the content is accessible to screen readers

```
<!-- A11y: <a> element should have child content -->
<a href="/foo"></a>

<!-- A11y: <h1> element should have child content -->
<h1></h1>
```

a11y_mouse_events_have_key_events

```
'%event%' event must be accompanied by '%accompanied_by%' event
```

Enforce that `onmouseover` and `onmouseout` are accompanied by `onfocus` and `onblur` , respectively. This helps to ensure that any functionality triggered by these mouse events is also accessible to keyboard users.

```
<!-- A11y: onmouseover must be accompanied by onfocus -->
<div onmouseover={handleMouseover} />

<!-- A11y: onmouseout must be accompanied by onblur -->
<div onmouseout={handleMouseout} />
```

a11y_no_abstract_role

```
Abstract role '%role%' is forbidden
```

```
`<%element%>` cannot have role '%role%'
```

WAI-ARIA roles should not be used to convert an interactive element to a non-interactive element. Non-interactive ARIA roles include `article`, `banner`, `complementary`, `img`, `listitem`, `main`, `region` and `tooltip`.

```
<!-- A11y: <textarea> cannot have role 'listitem' -->
<textarea role="listitem"></textarea>
```

a11y_no_noninteractive_element_interactions

```
Non-interactive element `<%element%>` should not be assigned mouse or keyboard event liste
```

A non-interactive element does not support event handlers (mouse and key handlers). Non-interactive elements include `<main>`, `<area>`, `<h1>` (, `<h2>`, etc), `<p>`, `<img>`, `<li>`, `<ul>` and `<ol>`. Non-interactive WAI-ARIA roles include `article`, `banner`, `complementary`, `img`, `listitem`, `main`, `region` and `tooltip`.

```
<!-- `A11y: Non-interactive element <li> should not be assigned mouse or keyboard eve
<li onclick={() => {}}></li>

<!-- `A11y: Non-interactive element <div> should not be assigned mouse or keyboard event
<div role="listitem" onclick={() => {}}></div>
```

a11y_no_noninteractive_element_to_interactive_role

```
Non-interactive element `<%element%>` cannot have interactive role '%role%'
```

WAI-ARIA roles should not be used to convert a non-interactive element to an interactive element. Interactive ARIA roles include `button`, `link`, `checkbox`, `menuitem`, `menuitemcheckbox`, `menuitemradio`, `option`, `radio`, `searchbox`, `switch` and `textbox`.

Docs

a11y_no_noninteractive_tabindex

```
noninteractive element cannot have nonnegative tabIndex value
```

Tab key navigation should be limited to elements on the page that can be interacted with.

```
<!-- A11y: noninteractive element cannot have nonnegative tabIndex value -->
<div tabindex="0"></div>
```

a11y_no_redundant_roles

```
Redundant role '%role%'
```

Some HTML elements have default ARIA roles. Giving these elements an ARIA role that is already set by the browser has no effect and is redundant.

```
<!-- A11y: Redundant role 'button' -->
<button role="button">...</button>

<!-- A11y: Redundant role 'img' -->
<img role="img" src="foo.jpg" />
```

a11y_no_static_element_interactions

```
`<%element%>` with a %handler% handler must have an ARIA role
```

Elements like `<div>` with interactive handlers like `click` must have an ARIA role.

```
<!-- A11y: <div> with click handler must have an ARIA role -->
<div onclick={() => ''}></div>
```

Docs

```
Avoid tabindex values above zero
```

Avoid positive `tabindex` property values. This will move elements out of the expected tab order, creating a confusing experience for keyboard users.

```
<!-- A11y: avoid tabindex values above zero -->
<div tabindex="1"></div>
```

a11y_role_has_required_aria_props

```
Elements with the ARIA role "%role%" must have the following attributes defined: %props%
```

Elements with ARIA roles must have all required attributes for that role.

```
<!-- A11y: A11y: Elements with the ARIA role "checkbox" must have the following attri
<span role="checkbox" aria-labelledby="foo" tabindex="0"></span>
```

a11y_role_supports_aria_props

```
The attribute '%attribute%' is not supported by the role '%role%'
```

Elements with explicit or implicit roles defined contain only `aria-*` properties supported by that role.

```
<!-- A11y: The attribute 'aria-multiline' is not supported by the role 'link'. -->
<div role="link" aria-multiline></div>

<!-- A11y: The attribute 'aria-required' is not supported by the role 'listitem'. This ro
<li aria-required></li>
```

Docs

```
The attribute '%attribute%' is not supported by the role '%role%'. This role is implicit (
```

Elements with explicit or implicit roles defined contain only `aria-*` properties supported by that role.

```
<!-- A11y: The attribute 'aria-multiline' is not supported by the role 'link'. -->
<div role="link" aria-multiline></div>

<!-- A11y: The attribute 'aria-required' is not supported by the role 'listitem'. This ro
<li aria-required></li>
```

a11y_unknown_aria_attribute

```
Unknown aria attribute 'aria-%attribute%'
```

```
Unknown aria attribute 'aria-%attribute%'. Did you mean '%suggestion%'?
```

Enforce that only known ARIA attributes are used. This is based on the WAI-ARIA States and Properties spec.

```
<!-- A11y: Unknown aria attribute 'aria-labeledby' (did you mean 'labelledby'?) -->
<input type="image" aria-labeledby="foo" />
```

a11y_unknown_role

```
Unknown role '%role%'
```

```
Unknown role '%role%'. Did you mean '%suggestion%'?
```

```
<!-- A11y: Unknown role 'toooltip' (did you mean 'tooltip'?) -->
<div role="toooltip"></div>
```

attribute_avoid_is

```
The "is" attribute is not supported cross-browser and should be avoided
```

attribute_global_event_reference

```
You are referencing `globalThis.%name%`. Did you forget to declare a variable with that na
```

attribute_illegal_colon

```
Attributes should not contain ':' characters to prevent ambiguity with Svelte directives
```

attribute_invalid_property_name

```
'%wrong%' is not a valid HTML attribute. Did you mean '%right%'?
```

attribute_quoted

```
Quoted attributes on components and custom elements will be stringified in a future versi
```

bind_invalid_each_rest

```
The rest operator (...) will create a new object and binding '%name%' with the original ol
```

```
Empty block
```

component_name_lowercase

```
`<%name%>` will be treated as an HTML element unless it begins with a capital letter
```

css_unused_selector

```
Unused CSS selector "%name%"
```

element_invalid_self_closing_tag

```
Self-closing HTML tags for non-void elements are ambiguous — use `<%name% ...></%name%>` i
```

event_directive_deprecated

```
Using `on:%name%` to listen to the %name% event is deprecated. Use the event attribute `on
```

export_let_unused

```
Component has unused export property '%name%'. If it is for external reference only, pleas
```

legacy_code

```
`%code%` is no longer valid — please use `%suggestion%` instead
```

legacy_component_creation

node_invalid_placement_ssr

```
%thing% is invalid inside `<%parent%>`. When rendering this component on the server, the
```

HTML restricts where certain elements can appear. In case of a violation the browser will 'repair' the HTML in a way that breaks Svelte's assumptions about the structure of your components. Some examples:

> `<p>hello <div>world</div></p>` will result in `<p>hello </p><div>world</div><p></p>` for example (the `<div>` autoclosed the `<p>` because `<p>` cannot contain block-level elements)
>
> `<option><div>option a</div></option>` will result in `<option>option a</option>` (the `<div>` is removed)
>
> `<table><tr><td>cell</td></tr></table>` will result in `<table><tbody><tr><td>cell</td></tr></tbody></table>` (a `<tbody>` is auto-inserted)

This code will work when the component is rendered on the client (which is why this is a warning rather than an error), but if you use server rendering it will cause hydration to fail.

non_reactive_update

```
`%name%` is updated, but is not declared with `$state(...)`. Changing its value will not
```

options_deprecated_accessors

```
The `accessors` option has been deprecated. It will have no effect in runes mode
```

options_deprecated_immutable

options_missing_custom_element

```
The `customElement` option is used when generating a custom element. Did you forget the `
```

options_removed_enable_sourcemap

```
The `enableSourcemap` option has been removed. Source maps are always generated now, and
```

options_removed_hydratable

```
The `hydratable` option has been removed. Svelte components are always hydratable now
```

options_removed_loop_guard_timeout

```
The `loopGuardTimeout` option has been removed
```

options_renamed_ssr_dom

```
`generate: "dom"` and `generate: "ssr"` options have been renamed to "client" and "server
```

perf_avoid_inline_class

```
Avoid 'new class' — instead, declare the class at the top level scope
```

perf_avoid_nested_class

reactive_declaration_invalid_placement

```
Reactive declarations only exist at the top level of the instance script
```

reactive_declaration_module_script_dependency

```
Reassignments of module-level declarations will not cause reactive statements to update
```

reactive_declaration_non_reactive_property

```
Properties of objects and arrays are not reactive unless in runes mode. Changes to this p
```

script_context_deprecated

```
`context="module"` is deprecated, use the `module` attribute instead
```

script_unknown_attribute

```
Unrecognized attribute — should be one of `generics`, `lang` or `module`. If this exists
```

slot_element_deprecated

```
Using `<slot>` to render parent content is deprecated. Use `{@render ...}` tags instead
```

state_referenced_locally

store_rune_conflict

```
It looks like you're using the `$%name%` rune, but there is a local binding called `%name%`
```

svelte_component_deprecated

```
`<svelte:component>` is deprecated in runes mode — components are dynamic by default
```

In previous versions of Svelte, the component constructor was fixed when the component was rendered. In other words, if you wanted `<X>` to re-render when `X` changed, you would either have to use `<svelte:component this={X}>` or put the component inside a `{#key X}...{/key}` block.

In Svelte 5 this is no longer true — if `X` changes, `<X>` re-renders.

In some cases `<object.property>` syntax can be used as a replacement; a lowercased variable with property access is recognized as a component in Svelte 5.

For complex component resolution logic, an intermediary, capitalized variable may be necessary. E.g. in places where `@const` can be used:

```svelte
{#each items as item}
  <svelte:component this={item.condition ? Y : Z} />
  {@const Component = item.condition ? Y : Z}
  <Component />
{/each}
```

A derived value may be used in other contexts:

```svelte
<script>
  // ...
  let condition = $state(false);
  const Component = $derived(condition ? Y : Z);
```

```
<Component />
```

svelte_element_invalid_this

```
`this` should be an `{expression}`. Using a string attribute value will cause an error in
```

svelte_self_deprecated

```
`<svelte:self>` is deprecated — use self-imports (e.g. `import %name% from './%basename%'
```

unknown_code

```
`%code%` is not a recognised code
```

```
`%code%` is not a recognised code (did you mean `%suggestion%`?)
```

Edit this page on GitHub

PREVIOUS
Compiler errors

NEXT
Runtime errors