Pseudo-classes

A CSS ***pseudo-class*** is a keyword added to a selector that lets you style a specific state of the selected element(s). For example, the pseudo-class `:hover` can be used to select a button when a user's pointer hovers over the button and this selected button can then be styled.

CSS

```
/* Any button over which the user's pointer is hovering */
button:hover {
  color: blue;
}
```

A pseudo-class consists of a colon (`:`) followed by the pseudo-class name (e.g., `:hover`). A functional pseudo-class also contains a pair of parentheses to define the arguments (e.g., `:dir()`). The element that a pseudo-class is attached to is defined as an *anchor element* (e.g., `button` in case `button:hover`).

Pseudo-classes let you apply a style to an element not only in relation to the content of the document tree, but also in relation to external factors like the history of the navigator (`:visited` , for example), the status of its content (like `:checked` on certain form elements), or the position of the mouse (like `:hover` , which lets you know if the mouse is over an element or not).

> **Note:** In contrast to pseudo-classes, pseudo-elements can be used to style a *specific part* of an element.

Element display state pseudo-classes

These pseudo-classes enable the selection of elements based on their display states.

`:fullscreen`

Matches an element that is currently in fullscreen mode.

`:modal`

Matches an element that is in a state in which it excludes all interaction with elements outside it until the interaction has been dismissed.

`:picture-in-picture`

Matches an element that is currently in picture-in-picture mode.

Input pseudo-classes

These pseudo-classes relate to form elements, and enable selecting elements based on HTML attributes and the state that the field is in before and after interaction.

`:autofill`

Matches when an `<input>` has been autofilled by the browser.

`:enabled`

Represents a user interface element that is in an enabled state.

`:disabled`

Represents a user interface element that is in a disabled state.

`:read-only`

Represents any element that cannot be changed by the user.

`:read-write`

Represents any element that is user-editable.

`:placeholder-shown`

Matches an input element that is displaying placeholder text. For example, it will match the `placeholder` attribute in the `<input>` and `<textarea>` elements.

`:default`

Matches one or more UI elements that are the default among a set of elements.

`:checked`

Matches when elements such as checkboxes and radio buttons are toggled on.

`:indeterminate`

Matches UI elements when they are in an indeterminate state.

`:blank`

Matches a user-input element which is empty, containing an empty string or other null input.

`:valid`

Matches an element with valid contents. For example, an input element with the type 'email' that contains a validly formed email address or an empty value if the control is not required.

`:invalid`

Matches an element with invalid contents. For example, an input element with type 'email' with a name entered.

`:in-range`

Applies to elements with range limitations. For example, a slider control when the selected value is in the allowed range.

`:out-of-range`

Applies to elements with range limitations. For example, a slider control when the selected value is outside the allowed range.

`:required`

Matches when a form element is required.

`:optional`

Matches when a form element is optional.

`:user-valid`

Represents an element with correct input, but only when the user has interacted with it.

`:user-invalid`

Represents an element with incorrect input, but only when the user has interacted with it.

Linguistic pseudo-classes

These pseudo-classes reflect the document language and enable the selection of elements based on language or script direction.

`:dir()`

The directionality pseudo-class selects an element based on its directionality as determined by the document language.

`:lang()`

Select an element based on its content language.

Location pseudo-classes

These pseudo-classes relate to links, and to targeted elements within the current document.

`:any-link`

Matches an element if the element would match either `:link` or `:visited`.

`:link`

Matches links that have not yet been visited.

`:visited`

Matches links that have been visited.

`:local-link`

Matches links whose absolute URL is the same as the target URL. For example, anchor links to the same page.

`:target`

Matches the element which is the target of the document URL.

`:target-within`

Matches elements which are the target of the document URL, but also elements which have a descendant which is the target of the document URL.

`:scope`

Represents elements that are a reference point for selectors to match against.

Resource state pseudo-classes

These pseudo-classes apply to media that is capable of being in a state where it would be described as playing, such as a video.

`:playing`

Represents a media element that is capable of playing when that element is playing.

`:paused`

Represents a media element that is capable of playing when that element is paused.

Time-dimensional pseudo-classes

These pseudo-classes apply when viewing something which has timing, such as a WebVTT caption track.

`:current`

Represents the element or ancestor of the element that is being displayed.

`:past`

Represents an element that occurs entirely before the `:current` element.

`:future`

Represents an element that occurs entirely after the `:current` element.

Tree-structural pseudo-classes

These pseudo-classes relate to the location of an element within the document tree.

`:root`

Represents an element that is the root of the document. In HTML this is usually the `<html>` element.

`:empty`

Represents an element with no children other than white-space characters.

`:nth-child`

Uses `An+B` notation to select elements from a list of sibling elements.

`:nth-last-child`

Uses `An+B` notation to select elements from a list of sibling elements, counting backwards from the end of the list.

`:first-child`

Matches an element that is the first of its siblings.

`:last-child`

Matches an element that is the last of its siblings.

`:only-child`

Matches an element that has no siblings. For example, a list item with no other list items in that list.

`:nth-of-type`

Uses `An+B` notation to select elements from a list of sibling elements that match a certain type from a list of sibling elements.

`:nth-last-of-type`

Uses `An+B` notation to select elements from a list of sibling elements that match a certain type from a list of sibling elements counting backwards from the end of the list.

`:first-of-type`

Matches an element that is the first of its siblings, and also matches a certain type selector.

`:last-of-type`

Matches an element that is the last of its siblings, and also matches a certain type selector.

`:only-of-type`

Matches an element that has no siblings of the chosen type selector.

User action pseudo-classes

These pseudo-classes require some interaction by the user in order for them to apply, such as holding a mouse pointer over an element.

`:hover`

Matches when a user designates an item with a pointing device, such as holding the mouse pointer over the item.

`:active`

Matches when an item is being activated by the user. For example, when the item is clicked on.

`:focus`

Matches when an element has focus.

`:focus-visible`

Matches when an element has focus and the user agent identifies that the element should be visibly focused.

`:focus-within`

Matches an element to which `:focus` applies, plus any element that has a descendant to which `:focus` applies.

Functional pseudo-classes

These pseudo-classes accept a selector list or forgiving selector list as a parameter.

`:is()`

The matches-any pseudo-class matches any element that matches any of the selectors in the list provided. The list is forgiving.

`:not()`

The negation, or matches-none, pseudo-class represents any element that is not represented by its argument.

`:where()`

The specificity-adjustment pseudo-class matches any element that matches any of the selectors in the list provided without adding any specificity weight.

The list is forgiving.

`:has()`

The relational pseudo-class represents an element if any of the relative selectors match when anchored against the attached element.

Syntax

CSS

```
selector:pseudo-class {
  property: value;
}
```

Like regular classes, you can chain together as many pseudo-classes as you want in a selector.

Alphabetical index

Pseudo-classes defined by a set of CSS specifications include the following:

A

- `:active`
- `:any-link`
- `:autofill`

B

- `:blank`

C

- `:checked`
- `:current`

D

- `:default`
- `:defined`
- `:dir()`
- `:disabled`

E

- `:empty`
- `:enabled`

F

- `:first`
- `:first-child`
- `:first-of-type`
- `:focus`
- `:focus-visible`
- `:focus-within`

- `:future`

H

- `:has()`
- `:host`
- `:host()`
- `:host-context()`
- `:hover`

I

- `:indeterminate`

- :in-range
- :invalid
- :is()

L

- :lang()
- :last-child
- :last-of-type
- :left
- :link
- :local-link

M

- :modal

N

- :not()
- :nth-child()
- :nth-last-child()
- :nth-last-of-type()
- :nth-of-type()

O

- :only-child
- :only-of-type
- :optional
- :out-of-range

P

- :past
- :paused
- :picture-in-picture
- :placeholder-shown
- :playing
- :popover-open

R

- :read-only
- :read-write
- :required
- :right
- :root

S

- :scope
- :state()

T

- :target
- :target-within

U

- :user-invalid

V

- `:valid`
- `:visited`

W

- `:where()`

Specifications

Specification
HTML # pseudo-classes
Selectors Level 4
CSS Basic User Interface Module Level 4

See also

- Pseudo-elements

Help improve MDN

Was this page helpful to you?

Yes No

Learn how to contribute.

This page was last modified on Dec 18, 2024 by MDN contributors.