CSS: Pseudo Selectors

OVERVIEW

CSS is an important web technology to understand and learn. This module will take you through from the basics to more advanced CSS. It will teach you how to apply your skills in web development to build any web page you'd like. It's crucial to build your portfolio, this module is one of the first steps to your portfolio, and web development as a whole.

WHAT DO YOU NEED FOR THIS MODULE?

This module covers HTML, and CSS. We'll be working through practical examples as we go along. Feel free to do your own research as well as practice on your own. Also have a text editor installed on your laptop for practical sessions.

The following content is going to be covered in this document:

- Pseudo Classes
- Pseudo Elements

Pseudo-classes

Pseudo-classes are keywords which allow selection based on information that lies outside of the document tree or that cannot be expressed by other selectors or combinators. This information can be associated to a certain state (state and dynamic pseudo-classes), to locations (structural and target pseudo-classes), to negations of the former (negation pseudo-class) or to languages (lang pseudo-class). Examples include whether or not a link has been followed (`:visited`), the mouse is over an element (`:hover`), a checkbox is checked (`:checked`), etc.

Syntax

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

List of pseudo-classes:

Name	Description
`:active`	Applies to any element being activated (i.e. clicked) by the user.
`:any`	Allows you to build sets of related selectors by creating groups that the included items will match. This is an alternative to repeating an entire selector.
`:target`	Selects the current active #news element (clicked on a URL containing that anchor name)
`:checked`	Applies to radio, checkbox, or option elements that are checked or toggled into an "on" state.
`:default`	Represents any user interface element that is the default among a group of similar elements.
`:disabled`	Applies to any UI element which is in a disabled state.
`:empty`	Applies to any element which has no children.
`:enabled`	Applies to any UI element which is in an enabled state.
`:first`	Used in conjunction with the `@page` rule, this selects the first page in a printed document.
`:first-child`	Represents any element that is the first child element

of its parent.

`:first-of-type` Applies when an element is the first of the selected element type inside its parent. This may or may not be the first-child.

`:focus` Applies to any element which has the user's focus.
This can be given by the user's keyboard, mouse events, or other forms of input.

`:focus-within` Can be used to highlight a whole section when one
element inside it is focused. It matches any element that the `:focus` pseudo-class matches or that has a descendant focused.

`:full-screen` Applies to any element displayed in full-screen mode. It selects the whole stack of elements and not just the top level element.

`:hover` Applies to any element being hovered by the user's
pointing device, but not activated.

`:indeterminate` Applies radio or checkbox UI elements which are
neither checked nor unchecked, but are in an indeterminate state. This can be due to an element's attribute or DOM manipulation.

`:in-range` The `:in-range` CSS pseudo-class matches when
an element has its value attribute inside the specified range limitations for this element. It allows the page to give a feedback that the value currently defined using the element is inside the range limits.

`:invalid` Applies to `<input>` elements whose values are
invalid according to the type specified in the `type= attribute`.

`:lang` Applies to any element who's wrapping <body>
element has a properly designated `lang= attribute`. For the pseudo-class to be valid, it must contain a valid two or three letter language code.

`:last-child` Represents any element that is the last child element of its parent.

`:last-of-type` Applies when an element is the last of the selected element type inside its parent. This may or may not be the last-child.

`:left` Used in conjunction with the `@page` rule, this selects all the left pages in a printed document.

`:link` Applies to any links which haven't been visited by the user.

`:not()` Applies to all elements which do not match the value passed to `:not(p)` or `:not(.class-name)` for example. It must have a value to be valid and it can only contain one selector. However, you can chain multiple `:not` selectors together.

`:nth-child` Applies when an element is the **n -th** element of its parent, where **n** can be an integer, a mathematical expression (e.g **n+3**) or the keywords **odd** or **even** .

`:nth-of-type` Applies when an element is the **n -th** element of its parent of the same element type, where **n** can be an integer, a mathematical expression (e.g **n+3**) or the keywords **odd** or **even** .

`:only-child` The `:only-child` CSS pseudo-class represents any element which is the only child of its parent. This is the same as `:first-child:last-child` or `:nth-child(1):nth-last-child(1)`, but with a lower specificity.

`:optional` The `:optional` CSS pseudo-class represents any element that does not have the required attribute set on it. This allows forms to easily indicate optional fields and to style them accordingly.

`:out-of-range` The `:out-of-range` CSS pseudo-class matches when an element has its value attribute outside the specified range limitations for this element. It allows the page to give a feedback that the value currently defined using the element is outside the range limits. A value can be outside of a range if it is either smaller or larger than maximum and minimum set values.

`:placeholder-shown` **Experimental.** Applies to any form element currently displaying placeholder text.

`:read-only` Applies to any element which is not editable by the user.

`:read-write` Applies to any element that is editable by a user, such as `<input>` elements.

`:right` Used in conjunction with the `@page` rule, this selects all the right pages in a printed document.

`:root` Matches the root element of a tree representing the document.

`:scope` CSS pseudo-class matches the elements that are a reference point for selectors to match against.

`:visited` Applies to any links which have has been visited by the user.

The `:visited` pseudoclass can't be used for most styling in a lot of modern browsers anymore because it's a security hole.

Child Pseudo Class

"The :nth-child(an+b) CSS pseudo-class matches an element that has an+b-1 siblings before it in the document tree, for a given positive or zero value for n" - MDN :nth-child

pseudo-selector 1 2 3 4 5 6 7 8 9 10

`:first-child` ✔

`:nth-child(3)` ✔

`:nth-child(n+3)` ✔ ✔ ✔ ✔ ✔ ✔ ✔ ✔

`:nth-child(3n)` ✔ ✔ ✔

`:nth-child(3n+1)` ✔ ✔ ✔ ✔

`:nth-child(-n+3)` ✔ ✔ ✔

`:nth-child(odd)` ✔ ✔ ✔ ✔ ✔

`:nth-child(even)` ✔ ✔ ✔ ✔ ✔

`:last-child` ✔

`:nth-last-child(3)` ✔

Pseudo-Elements

pseudo-element	Description
`::after`	Insert content after the content of an element.
`::before`	Insert content before the content of an element.
`::first-letter`	Selects the first letter of each element.
`::first-line`	Selects the first line of each element.
`::selection`	Matches the portion of an element that is selected by a user.
`::backdrop`	Used to create a backdrop that hides the underlying document for an element in the top layer's stack.
`::placeholder`	Allows you to style the placeholder text of a form element **(Experimental)**
`::marker`	For applying list-style attributes on a given element **(Experimental)**
`::spelling-error`	Represents a text segment which the browser has flagged as incorrectly spelled **(Experimental)**
`::grammar-error`	Represents a text segment which the browser has flagged as grammatically incorrect **(Experimental)**

Pseudo-elements, just like pseudo-classes, are added to a CSS selectors but instead of describing a special state, they allow you to scope and style certain parts of an html element. For example, the `::first-letter` pseudo-element targets only the first letter of a block element specified by the selector.

Pseudo-Elements

Pseudo-elements are added to selectors but instead of describing a special state, they allow you to style certain parts of a document.\

The content attribute is required for pseudo-elements to render; however, the attribute can have an empty value (e.g. `content: ""`).

```
div::after {
content: 'after'; color:
red; border: 1px solid red;
} div { color: black;
border: 1px solid black;
padding: 1px; } div::before {
content: 'before'; color:
green; border: 1px solid
green; }
```