More Selectors

CSS3

Pseudo-Classes

Elements in a web page have an implicit state based on their position on the page or an attribute they possess. They can even experience changes in state based on user action or interaction.

In CSS we use pseudo-classes to target and style elements only when they are in a specific state.

There are a variety of elements that have different types of style-able state, but we'll go over some of the most common.

You can tell that a CSS selector is targeting the state of an element if you see a single colon (two adjoined colons mean something else we'll cover shortly.)

User Action Pseudo-Classes

Some of the easiest to understand and most fun to try out include the user action pseudo-classes!

Some common ones include:

- :hover
  These styles activate when a cursor enters with the element.
- :active
  These styles activate when the element is “activated” by a user (usually clicked on, or the enter key is pressed after the element is highlighted.)
- :focus
  These styles activate when the element is in focus by the user (“highlighted” and ready to activate, or type into in the case of form field inputs.)

To help illustrate these, try the following on a form field in your page…

index.html snippet:

```
<form>
  <label>Try interacting with this button::
    <input type=”button” name=”my-button” value=”Interact with me!”>
  </label>
</form>
```

main.css snippet:

```
[name=”my-button”]:hover { border-color: blue; }
[name=”my-button”]:focus { border-color: green; }
[name=”my-button”]:active { border-color: red; }
```

One of the easiest ways to focus interactive (form or anchor) elements one-by-one is to use the tab key. See if you can get each of these states to show!

Input Pseudo-Classes

There are form field input-specific states and attributes for use in styling forms and their fields. To target these we use input pseudo-classes.

Here are a number of them that you are likely to use or come across:

- :valid
  Styles apply when the form field input passes the HTML5 validations applied to it.
- :invalid
  Styles apply when the form field input fails to pass the HTML5 validations applied to it.
- :required
  Targets elements if they have the “required” HTML5 validation attribute.
- :optional
  Targets elements if they do not have the “required” HTML5 validation attribute.
- :enabled
  An available form field ready to accept user input.
- :disabled
  An unavailable form field; a field that cannot or can no longer accept user input.
- :checked
  In cases where a checkbox, radio button, or select option is selected, that activated option can be targeted.

Tree-Structural Pseudo-Classes

There are an array of pseudo-class selectors available that allow for targeting of elements based on their position in the HTML document compared to other elements. These are called tree-structural pseudo-classes.

These can be tricky to get the hang of. Much like the rest of CSS it is important to practice and increase understanding through experience with these features.

Child Pseudo-Class Selector

Consider the following HTML list:

```
<ul>
   <li>List Item #1</li>
   <li>List Item #2</li>
   <li>List Item #3</li>
   <li>List Item #4</li>
   <li>List Item #5</li>
</ul>
```

How might you target just the first item in the list? You may think to yourself, based on what we've covered, that adding an ID or class to the element would make it easier to target, and you'd be absolutely right. Through pseudo-classes, however, there is an alternative approach we can take!

Try the following, and note how they target the elements in the page…

```
li:first-child { color: red; } /* Selects an LI if it is the first child in its parent */
li:last-child { color: blue; } /* Selects an LI if it is the last child in its parent */
li:nth-child(3) { color: green; } /* An LI that is the 3rd child in its parent. */
li:nth-child(even) { color: yellow; } /* Every LI that is an even child. */
```

The child pseudo-class selectors open a lot of doors for selections, without the need for having to modify your HTML.

You can even match patterns within the parentheses if you need to match a selection more specific than even or odd children. See CSS Tricks' recipe book for some fantastic examples.

Type Pseudo-Class Selector

Consider, again, the following HTML list:

```
<ul>
   <li>List Item #1</li>
   <li>List Item #2</li>
   <li>List Item #3</li>
   <li>List Item #4</li>
   <li>List Item #5</li>
</ul>
```

If we want to make selections similar to the child-based ones we've been experimenting with count by number of elements of that tag / element name, we can do so with the type pseudo-class selectors.

Try the following, and note how they target the elements in the page…

```
li:first-of-type { color: red; } /* Selects an LI if it is the first LI in its parent */
li:last-of-type { color: blue; } /* Selects an LI if it is the last LI in its parent */
li:nth-of-type(3) { color: green; } /* An LI that is the 3rd LI in its parent. */
li:nth-of-type(even) { color: yellow; } /* Every even LI in its parent. */
```

Like the child pseudo-class selectors, you can match patterns within the parentheses.

See CSS Tricks' article on differences between the nth-child and nth-of-type selectors.

Pseudo-Elements

Pseudo-elements allow for targeting of specific parts within elements. This type of selector is marked by use of two colons (in contrast to pseudo-class' single colon syntax.)

Text-Based Pseudo Elements

The first-letter selector allows you to target and style the first text character that appears in the element.

This is great for achieving what one might normally see in a novel or article: an enlarged and stylized first character of a paragraph.

```
p::first-letter {
   font-size: 2em;
   font-weight: bold;
}
```

If instead the first line should stand out, you can use the first-line selector.

Note that when applying both first letter and first-line to the same element(s) there will exist an overlap of styles.

```
p::first-line {
   color: gray;
   font-family: Georgia, “Times New Roman”, serif;
}
```

To choose the styling of user-highlighted text, you can use the selection selector.

```
p::selection {
   background-color: lightgreen;
}
```

Before and After

Before and after are a bit special in that they essentially create a new first or last child element, respectively, within the target.

These elements are not reflected in the HTML and therefore not read by most any screen reader or search engine bot. Ensure no essential content is included in such a pseudo-element, as it will be overlooked by these tools.

Observe the following example to get a better idea of how these pseudo-elements work.

Notice how the “content” property populates new text inside of the anchor!

HTML:

```
<a href=”#”>
   50.00 (Click Here to Buy)
</a>
```

CSS:

```
a::before{
   content: “$”;
}
a::after {
   content: “⇛”;
   color: green;
}
```

$ 50.00 (Click Here to Buy) ⇛

Before and after are exceptionally powerful tools. See CSS Tricks’ article on what they can do!

Recommended Reading

If you'd like to understand pseudo classes and pseudo elements more deeply, read the following:

- Meyer, E. A; Weyl, E. (October 2017). CSS: The Definitive Guide, 4th Edition. O'Reilly Media, Inc.
  - Chapter 2. Selectors
    - Pseudo-Class Selectors
    - Pseudo-Element Selectors