CSS: Selectors

October 2023

Overview

- Pseudo Selectors
- Element Selectors
- Class selectors
- Id selectors
- Universal
- Attribute etc.....

Universal CSS Selector

The universal selector works like a wild card character, selecting all elements on a page. Every HTML page is built on content placed within HTML tags. Each set of tags represents an element on the page. Look at the following CSS example, which uses the universal selector:

```
* {
    color: green;
    font-size: 20px;
    line-height: 25px;
}
```

Universal Selector

The three lines of code inside the curly braces (color, font-size, and line-height) will apply to all elements on the HTML page. As seen here, the universal selector is declared using an asterisk. You can also use the universal selector in combination with other selectors.

Element Type CSS Selector

Also referred to simply as a “type selector,” this selector must match one or more HTML elements of the same name. Thus, a selector of nav would match all HTML nav elements, and a selector of <ul> would match all HTML unordered lists, or <ul> elements. The following example uses an element type selector to match all <ul> elements:

```
ul {
    list-style:
none;
    border: solid
1px #ccc;
}
```

ID CSS Selector

An ID selector is declared using a hash, or pound symbol (#) preceding a string of characters. The string of characters is defined by the developer. This selector matches any HTML element that has an ID attribute with the same value as that of the selector, but minus the hash symbol.

```
#container {
    width: 960px;
    margin: 0 auto;
}
```

This CSS uses an ID selector to match an HTML element such as

```
<div id="container"></div>
```

ID CSS Selector

In this case, the fact that this is a element doesn't matter—it could be any kind of HTML element. As long as it has an ID attribute with a value of container, the styles will apply.

An ID element on a web page should be unique. That is, there should only be a single element on any given page with an ID of container. This makes the ID selector quite inflexible, because the styles used in the ID selector rule set can be used only once per page.

```
#container {
    width: 960px;
    margin: 0 auto;
}
```

This CSS uses an ID selector to match an HTML element such as

```
<div id="container"></div>
```

Class Selectors

Class Selectors

```
.box {
    padding: 20px;
    margin: 10px;
    width: 240px;
}
```

```
<div
class="box"></div
>
```

The class selector is the most useful of all CSS selectors. It's declared with a dot preceding a string of one or more characters. Just as is the case with an ID selector, this string of characters is defined by the developer. The class selector also matches all elements on the page that have their class attribute set to the same value as the class, minus the dot.

Class Selectors

```
.box {
    padding: 20px;
    margin: 10px;
    width: 240px;
}
```

```
<div
class="box"></div
>
```

Another reason the class selector is a valuable ally is that HTML allows multiple classes to be added to a single element. This is done by separating the classes in the HTML class attribute using spaces. Here's an example:

```
<div class="box box-more
box-extended"></div>
```

Descendant Combinator

CSS selector combinators combine selectors for precision targeting. The descendant selector or, more accurately, the descendant combinator lets you combine two or more selectors so you can be more specific in your selection method. For example:

```
#container .box {
   float: left;
   padding-bottom:
15px;
}
```

```
<div
id="container">
  <div
class="box"></div>

  <div
class="box-2"></div
>
</div>

<div

class="box"></div>
```

This declaration block will apply to all elements that have a class of box that are inside an element with an ID of container. It's worth noting that the .box element doesn't have to be an immediate child: there could be another element wrapping .box, and the styles would still apply.

Look at the following HTML:

Child Combinator

A selector that uses the child combinator is similar to a selector that uses a descendant combinator, except it only targets immediate child elements:

```
#container > .box {
   float: left;
   padding-bottom:
15px;
}
```

This is the same code from the descendant combinator example, but instead of a space character, we're using the greater-than symbol (or right angle bracket.)

Child Combinator

```
<div
id="container">
  <div
class="box"></di
v>

  <div>
    <div
class="box"></di
v>
  </div>
</div>
```

In this example, the selector will match all elements that have a class of box and that are immediate children of the #container element. That means, unlike the descendant combinator, there can't be another element wrapping .box—it has to be a direct child element.

Here's an HTML example:

General Sibling Combinator

A selector that uses a general sibling combinator matches elements based on sibling relationships. That is to say, the selected elements are beside each other in the HTML.

```
h2 ~ p {
    margin-bottom: 20px;
}
```

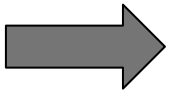
```
h2 ~ p {
   margin-bottom:
20px;
}
```

```
<h2>Title</h2>
<p>Paragraph example.</p>
<p>Paragraph example.</p>
<p>Paragraph example.</p>
<div class="box">
  <p>Paragraph
example.</p>
</div>
```

This type of selector is declared using the tilde character (~). In this example, all paragraph elements (<p>) will be styled with the specified rules, but only if they are siblings of <h2> elements. There could be other elements in between the <h2> and <p>, and the styles would still apply.

Attribute CSS Selector

The attribute selector targets elements based on the presence and/or value of HTML attributes, and is declared using square brackets:



The attribute selector targets elements based on the presence and/or value of HTML attributes, and is declared using square brackets:

```
input[type="text"] {
   background-color:
#444;
   width: 200px;
}
```

```
input[type] {
   background-color:
#444;
   width: 200px;
}
```

Pseudo-class CSS Selector

A pseudo-class uses a colon character to identify a pseudo-state that an element might be in—for example, the state of being hovered, or the state of being activated. Let's look at a common example:

```
a:hover {
    color: red;
}
```

In this case, the pseudo-class portion of the selector is the :hover part. Here we've attached this pseudo-class to all anchor elements (a elements). This means that when the user hovers their mouse over an a element, the color property for that element will change to red. This type of pseudo-class is a dynamic pseudo-class, because it occurs only in response to user interaction—in this case, the mouse moving over the targeted element.

Other popular pseudo-classes include:

- `:visited`: matches visited links
- `:target`: matches an element targeted by a document URL
- `:first-child`: targets the first child element
- `:nth-child`: selects specific child elements
- `:empty`: matches an element with no content or child elements
- `:checked`: matches a toggled-on checkbox or radio button
- `:blank`: styles an empty input field
- `:enabled`: matches an enabled input field
- `:disabled`: matches a disabled input field
- `:required`: targets a required input field
- `:valid`: matches a valid input field
- `:invalid`: matches an invalid input field
- `:playing`: targets a playing audio or video element
- `:is`: a native CSS solution for nesting styles
- `:has`: Identical to `:is` but differs with specificity.
- `:where`: similar syntax to `:is()` and `:where()`, but it targets an element which *contains* a set of