Selectors - CSS

Simple CSS Selectors are selectors that are most commonly used and cover a good amount of use cases. The simple CSS selectors include selecting by the HTML tag name, selecting by a specific ID, selecting with a class name, and selecting using various combinations of attributes. You can go pretty far with just these simple CSS selectors.

Pubudu Senanayake

Type Selector

Also known as the **Element** Selector, the **Type Selector** targets *all elements* that match the given node name. This allows you to select all **<p>** elements or all ** ** elements at one time.

```
element { css style properties }
```

Type Selector Example

```css
p {
    background-color: lightblue;
}
```

```html
<body>
    <div>
        <p>This is a paragraph with some text in it.</p>
        <div>I'm just one div in between two paragraph tags.</div>
        <p>This is a different paragraph with text.</p>
    </div>
</body>
```

This is a paragraph with some text in it.

I'm just one div in between two paragraph tags.

This is a different paragraph with text.

ID Selector

The **ID Selector** selects the HTML element based on the value contained in its **`id`** attribute. An ID Selector is used when you want to target one specific element on the web page.

```
#id_value { css style properties }
```

ID Selector Example

```css
#coffee {
    background-color: lightblue;
}
```

```html
<body>
    <div>
        <p>This paragraph doesn't have any id attribute.</p>
        <p id="coffee">This paragraph has an id of "coffee".</p>
    </div>
</body>
```

This paragraph doesn't have any id attribute.

This paragraph has an id of "coffee".

Class Selector

Some say the **Class Selector** is the *most useful* of the available CSS Selectors. In fact, almost all modern CSS Frameworks allow you to style elements by simply applying a class name. It is helpful to have many elements on the same page that share a given class. This helps to allow the reuse of styles leading to less repetition in the CSS code.

```
.class_name { css style properties }
```

Class Selector Example

```
.awesome {
    background-color: lightblue;
}
```

Markup Copy

```
<body>
    <div>
        <p>This paragraph doesn't have any class.</p>
        <p class="awesome">This paragraph has awesome class.</p>
        <div class="foo bar awesome">This div has 3 classes.  One of them is awesome.</<span class=
    </div></span></span></span></span></span></span>>
</body>
```

This paragraph doesn't have any class.

This paragraph has awesome class.

This div has 3 classes. One of them is awesome.

Attribute Selectors

Square brackets are used to create an **Attribute Selector** in CSS. This selector selects tags based on the existence or value of an HTML attribute. There are several types of attribute selectors in CSS.

[attribute]

The **[attribute] selector** is used to target elements on the web page with the given attribute.

```
[attribute] { css style properties }
```

[attribute] Selector Example

```
[title] {
    background-color: lightblue;
}
```

```
<body>
    <a href="#" title="Happy Link">This link has a title attribute.</a>
    <a href="#">This link has no title attribute.</a>
</body>
```

This link has a title attribute. This link has no title attribute.

[attr=value]

The **[attribute=value] selector** is used to target elements on the web page with the given attribute and specified value.

```
[attribute=value] { css style properties }
```

[attribute=value] Selector Example

```
[href="https://google.com"] {
    background-color: lightblue;
}
```

```
<body>
    <a href="https://google.com">Google</a>
    <a href="https://bing.com">Bing</a>
    <a href="https://duckduckgo.com">Duck Duck Go</a>
</body>
```

Google Bing Duck Duck Go

[attribute~=value]

The **[attribute~=value] selector** is used to target elements that have an attribute value containing a specified word.

```
[attribute~=value] { css style properties }
```

[attribute~=value] Selector Example

```css
[data~="hooray"] {
    background-color: lightblue;
}
```

```html
<body>
    <div data="hip">This is hip.</div>
    <div data="hip hop">This is hip hop.</div>
    <div data="hip hop hooray">Hip hop hooray.</div>
</body>
```

This is hip.
This is hip hop.
Hip hop hooray.

[attribute|=value]

The **[attribute|=value] selector** targets elements with an attribute value that can be exactly as given or can begin with the value immediately followed by a hyphen.

```
[attribute|=value] { css style properties }
```

[attribute|=value] Selector Example

```css
[class|="lg"] {
    background-color: lightblue;
}
```

```html
<body>
    <div class="btn">Button</div>
    <div class="lg-btn">Large Button</div>
    <div class="sm-btn">Small Button</div>
</body>
```

Button
Large Button
Small Button

[attribute^=value]

The [attribute^=value] has a caret (^) operator, which means “to start with”. This will target elements on the page where the given attribute starts with a specific value. For example you may be able to select all links on the page that are external vs internal, and make the background color light blue.

```
[attribute^=value] { css style properties }
```

[attribute^=value] Selector Example

```
a[href^="http://"] {
    background-color: lightblue;
}
```

```
<body>
    <a href="http://www.webdesignernews.com/">Web Design News (External Link)</a>
    <a href="/design/web.html">Design Category (Internal Link)</a>
</body>
```

Web Design News (External Link) Design Category (Internal Link)

[attribute$=value]

This attribute selector makes use of the dollar sign which is the opposite of the prior example. While the caret means to start with, the $ symbol means “ends with”. A use case for this might be to target links on the page which link to a PDF document.

```
[attribute$=value] { css style properties }
```

[attribute$=value] Selector Example

```css
a[href$=".pdf"] {
    background-color: lightblue;
}
```

```html
<body>
    <a href="http://www.webdesignernews.com/">Web Design News</a>
    <a href="/design/great.pdf">A Great PDF To Read</a>
</body>
```

A Great PDF To Read A Great PDF To Read

[attribute*=value]

The final attribute substring selector we'll look at makes use of the asterisk (*****) operator, which stands for "contains".

```
[attribute*=value] { css style properties }
```

[attribute*=value] Selector Example

```css
[class*="blockquote"] {
    background-color: lightblue;
}
```

```html
<body>
    <div class="blockquote-lead">Leading div element.</div>
    <div class="banana">Div in the middle.</div>
    <p class="blockquote-footer">Paragraph in the footer area.</p>
</body>
```

Leading div element.
Div in the middle.

Paragraph in the footer area.

Universal Selector

```
* { css style properties }
```

Universal Selector Example

The asterisk is often used as a wild card character of sorts, and when used as a CSS selector that is exactly what it is. The Universal Selector selects every element on the web page at once. It's not something you would use frequently but is good for resetting the box-sizing in layouts to render better widths and heights. You will often see a style sheet use the universal selector to set margin and padding both to 0. This gives the same starting point for a page layout no matter what type of browser, device, or operating system is being used.

```css
* {
    background-color: lightblue;
}
```

```html
<body>
    <div>One</div>
    <div>Two</div>
    <div>Three</div>
</body>
```

One
Two
Three

Combinator Selectors

Combinator selectors in CSS are the next level of granularity when selecting page elements in CSS. Combinators are used to target children, grandchildren, great-grandchildren, or later of a given element in the document tree. They symbols used for Combinators include the space character , greater than character **>**, the plus sign **+**, and the tilde **~**.

Descendant Combinator

The Descendant Combinator is actually an empty space ( ). The pattern of this combinator is simply two tags side by side with a space in between them. It selects elements that are descendants of the first element.

```
tag tag { css style properties }
```

Descendant Combinator Selector Example

In this example there are four <li> elements on the page. Only the two that are descendants of the are targeted however.

```
div li {
    background-color: lightblue;
}
```

```
<body>
    <ul>
        <li>First in the list.</li>
        <li>Second in the list.</li>
    </ul>
    <div>
        <li>First in the list.</li>
        <li>Second in the list.</li>
    </div>
</body>
```

- First in the list.
- Second in the list.

- First in the list.
- Second in the list.

Child Combinator

The child combinator uses the greater than (>) operator and targets elements that are immediate children of the parent. The difference between Descendant and Child selectors is that the Child Combinator is more specific. It only will select direct children elements of the first tag. The Descendant Combinator however is more liberal. Not only will it select children of the first tag, it will also select grand children, great grand children, and so on.

```
tag > tag { css style properties }
```

Child Combinator Selector Example

```css
div > p {
    background-color: lightblue;
}
```

```html
<body>
    <div>
        <p>Paragraph in a div.</p>
        <p>Another paragraph in a div.</p>
        <span>
            <p>A paragraph in a span.</p>
        </span>
    </div>
</body>
```

Paragraph in a div.

Another paragraph in a div.

A paragraph in a span.

Watch the difference now if we use the Descendant Combinator (**`div p`**) instead of the Child Combinator (**`div > p`**). Note now that the third paragraph which is a *grandchild* of the is still targeted.

```
div p {
    background-color: lightblue;
}
```

Paragraph in a div.

Another paragraph in a div.

A paragraph in a span.

Adjacent Sibling Combinator

The Adjacent Sibling Combinator uses the plus sign (**+**) and targets elements that are next to each other in the document tree which have the same parent. In other words the second tag directly follows the first, and they share the same parent element.

```
tag + tag { css style properties }
```

Adjacent Sibling Combinator Selector Example

```
h2 + p {
    background-color: lightblue;
}
```

```
<body>
    <div>
        <h1>Adjacent Sibling Combinator!</h1>
        <p>Paragraph.</p>
        <section>
            <h2>Section Title</h2>
            <p>Paragraph in the section.</p>
            <p>Next Paragraph in the section.</p>
        </section>
        <p>Paragraph after the section.</p>
    </div>
</body>
```

So in this example **h2 + p** targets every <p> element that is placed immediately after an <h2> element.

Output

Adjacent Sibling Combinator!

Paragraph.

Section Title

Paragraph in the section.

Next Paragraph in the section.

Paragraph after the section.

General Sibling Combinator

The General Sibling Combinator uses the tilde (~) operator. It differs from the Adjacent Sibling Combinator in that the second selector doesn't have to immediately follow the first. Both elements must both share the same parent.

```
tag ~ tag { css style properties }
```

```css
h1 ~ section {
    background-color: lightblue;
}
```

```html
<body>
    <div>
        <h1>General Sibling Combinator!</h1>
        <p>Paragraph.</p>
        <section>
            <h2>Section Title</h2>
            <p>Paragraph in the section.</p>
            <p>Next Paragraph in the section.</p>
        </section>
        <p>Paragraph after the section.</p>
    </div>
</body>
```

output

General Sibling Combinator!

Paragraph.

Section Title

Paragraph in the section.

Next Paragraph in the section.

Paragraph after the section.

Pseudo Classes

Pseudo-classes are used to target elements on the web page based on state information that is not part of the document tree. First we'll look at the **Structural pseudo-classes**.

Structural Pseudo Classes

Structural Pseudo Classes are a powerful way to target elements in the DOM which otherwise would not be able to be targeted without adding extra IDs or classes. The use of Structural Pseudo Classes is meant to minimize the addition of extraneous classes in your HTML markup thereby keeping your layouts cleaner and easier to deal with.

Search about Pseudo Class selectors

END