CSS Selectors

In this tutorial you will learn how to use CSS selectors for applying style rules to elements.

What is Selector?

A CSS selector is a pattern to match the elements on a web page. The style rules associated with that selector will be applied to the elements that match the selector pattern.

Selectors are one of the most important aspects of CSS as they allow you to target specific elements on your web page in various ways so that they can be styled.

Several types of selectors are available in CSS, let's take a closer look at them:

Universal Selector

The universal selector, denoted by an asterisk (*), matches every single element on the page.

The universal selector may be omitted if other conditions exist on the element. This selector is often used to remove the default margins and paddings from the elements for quick testing purpose.

Let's try out the following example to understand how it basically works:

Example **Try this code »**

```
* {
    margin: 0;
    padding: 0;
}
```

The style rules inside the `*` selector will be applied to every element in a document.

> **Note:** It is recommended *not to use* the universal selector (*) too often in a production environment, since this selector matches every element on a web page that puts too much of unnecessary pressure on the browsers. Use element type or class selector instead.

Element Type Selectors

An element type selector matches all instance of the element in the document with the corresponding element type name. Let's try out an example to see how it actually works:

Example **Try this code »**

```
p {
    color: blue;
}
```

The style rules inside the `p` selector will be applied on every `<p>` element (or paragraph) in the document and color it blue, regardless of their position in the document tree.

Id Selectors

The id selector is used to define style rules for a *single* or *unique* element.

The id selector is defined with a hash sign (`#`) immediately followed by the id value.

Example **Try this code »**

```
#error {
    color: red;
}
```

This style rule renders the text of an element in red, whose `id` attribute is set to `error`.

> **Note:** The value of an id attribute must be unique within a given document — meaning no two elements in your HTML document can share the same id value.

Class Selectors

The class selectors can be used to select any HTML element that has a `class` attribute. All the elements having that class will be formatted according to the defined rule.

The class selector is defined with a period sign (`.`) immediately followed by the class value.

Example **Try this code »**

```
.blue {
    color: blue;
}
```

The above style rules renders the text in blue of every element in the document that has `class` attribute set to `blue`. You can make it a bit more particular. For example:

Example **Try this code »**

```
p.blue {
    color: blue;
}
```

The style rule inside the selector `p.blue` renders the text in blue of only those `<p>` elements that has `class` attribute set to `blue`, and has no effect on other paragraphs.

Descendant Selectors

You can use these selectors when you need to select an element that is the descendant of another element, for example, if you want to target only those anchors that are contained within an unordered list, rather than targeting all anchor elements. Let's see how it works:

Example **Try this code »**

```
ul.menu li a {
    text-decoration: none;
}
h1 em {
    color: green;
}
```

The style rules inside the selector `ul.menu li a` applied to only those `<a>` elements that contained inside an `<ul>` element having the class `.menu`, and has no effect on other links inside the document.

Similarly, the style rules inside the `h1 em` selector will be applied to only those `<em>` elements that contained inside the `<h1>` element and has not effect on other `<em>` elements.

Child Selectors

A child selector is used to select only those elements that are the direct children of some element.

A child selector is made up of two or more selectors separated by a greater than symbol (`>`). You can use this selector, for instance, to select the first level of list elements inside a nested list that has more than one level. Let's check out an example to understand how it works:

Example **Try this code »**

```
ul > li {
    list-style: square;
}
ul > li ol {
    list-style: none;
}
```

The style rule inside the selector `ul > li` applied to only those `<li>` elements that are direct children of the `<ul>` elements, and has no effect on other list elements.

Adjacent Sibling Selectors

The adjacent sibling selectors can be used to select sibling elements (i.e. elements at the same level). This selector has the syntax like: E1 + E2, where E2 is the target of the selector.

The selector `h1 + p` in the following example will select the `<p>` elements only if both the `<h1>` and `<p>` elements share the same parent in the document tree and `<h1>` is immediately precedes the `<p>` element. That means only those paragraphs that come immediately after each `<h1>` heading will have the associated style rules. Let's see how this selector actually works:

Example **Try this code »**

```
h1 + p {
    color: blue;
    font-size: 18px;
}
ul.task + p {
    color: #f0f;
    text-indent: 30px;
}
```

General Sibling Selectors

The general sibling selector is similar to the adjacent sibling selector (E1 + E2), but it is less strict. A general sibling selector is made up of two simple selectors separated by the tilde (~) character. It can be written like: E1 ~ E2, where E2 is the target of the selector.

The selector `h1 ~ p` in the example below will select all the `<p>` elements that preceded by the `<h1>` element, where all the elements share the same parent in the document tree.

Example **Try this code »**

```
h1 ~ p {
    color: blue;
    font-size: 18px;
}
ul.task ~ p {
    color: #f0f;
    text-indent: 30px;
}
```

There are more sophisticated selectors like attribute selectors, pseudo-classes, pseudo-elements. We will discuss about these selectors in detail in the upcoming chapters.

Grouping Selectors

Often several selectors in a style sheet share the same style rules declarations. You can group them into a comma-separated list to minimize the code in your style sheet. It also prevents you from repeating the same style rules over and over again. Let's take a look:

Example **Try this code »**

```
h1 {
    font-size: 36px;
    font-weight: normal;
}
h2 {
    font-size: 28px;
    font-weight: normal;
}
h3 {
    font-size: 22px;
    font-weight: normal;
}
```

As you can see in the above example, the same style rule `font-weight: normal;` is shared by the selectors `h1`, `h2` and `h3`, so it can be grouped in a comma-separated list, like this:

Example **Try this code »**

```
h1, h2, h3 {
    font-weight: normal;
}
h1 {
    font-size: 36px;
}
h2 {
    font-size: 28px;
}
h3 {
    font-size: 22px;
}
```

PREVIOUS PAGE NEXT PAGE