Selectors

CSS3

CSS Selectors

Thus far, we've been using element names to select and style those elements in our page. This is one of the most common ways to make a selection, but not the only way! There are a wide array of CSS selectors available.

The basic selectors are as follows:

- Type Selector
- Universal Selector
- Attribute Selector
- Class Selector
- ID Selector

Type Selector

This is the selector we've been using thus far! You simply use the name of the element, and all elements with that tag name in the page will be affected by these styles.

For example...

```
p {
    font-family: Arial, Helvetica, sans-serif;
}
```

This code will make every paragraph in the web page use the specified font stack.

Universal Selector

The universal selector makes use of a wildcard character: the asterisk (*).

It is used to select all elements. For example, consider the following…

```
* {
    color: salmon;
}
```

This is telling the browser to salmon-colour literally every element in the web page's text. Everything from HTML, to BODY, to H2, to P, to A, to INPUT, and everything else!

It is for this reason, you'll want to use this sparingly, and only if you're sure it will have the desired effect.

Attribute Selector

We can target elements by their attributes and their values via the attribute selector. Take a look at the following HTML element…

```
<a
   id=”duckduckgo-link”
   class=”search-engine-link”
   href=”https://DuckDuckGo.com”
   title=”Click here to visit the DuckDuckGo search engine.”>
   DuckDuckGo
</a>
```

We could target this with the type selector by simply saying “a,” but that would also target every other anchor on the page. The attribute selector gives us more specific ways to target this element.

```
[title] {
   color: rgb(255, 0, 0);
}
```

The above would target any element that has a title attribute. That still might be a bit too broad if this were the only element we want to target. How about…

```
[href=”https://DuckDuckGo.com/”] {
   color: rgb(255, 0, 0);
}
```

This would target any element with an “href” attribute assigned specifically the value “https://DuckDuckGo.com”. This is often used to target form field inputs by their “name” attribute.

Attribute Selector Operators

```
<a
   id="duckduckgo-link"
   class="search-engine-link"
   href="https://DuckDuckGo.com"
   title="Click here to visit the DuckDuckGo search engine.">
   DuckDuckGo
</a>
```

By adding an *, $, or ~ after the attribute name you can modify the selector a bit…

```
[href*="Duck"] {
   color: rgb(255, 0, 0);
}
```

With an asterisk, it will look for any elements with that value in it at all. In this case, any element with "Duck" anywhere in the "href" will match.

```
[href$=".com"] {
   color: rgb(255, 0, 0);
}
```

With the dollar sign, it will check for an attribute value that ends with that text.

```
[href="https://duckduckgo.com" i] {
   color: rgb(255, 0, 0);
}
```

If you add an "i" at the end (still inside the square brackets) the check will no longer be capital-sensitive (case-insensitive.)

There are more operators available for the attribute selector that you'll want to check out as well!

Class Selector

```
<a
   id=”duckduckgo-link”
   class=”search-engine-link”
   href=”https://DuckDuckGo.com”
   title=”Click here to visit the DuckDuckGo search engine.”>
   DuckDuckGo
</a>
```

To target an element by class name, we are able to make use of the attribute selector…

```
[class=”search-engine-link”] {
   color: rgb(255, 0, 0);
}
```

CSS offers a separate CSS selector to offer a shorthand. Instead of typing out the attribute selector, you just place a period before the name of the class you’d like to target…

```
.search-engine-link {
   color: rgb(255, 0, 0);
}
```

ID Selector

```
<a
   id=”duckduckgo-link”
   class=”search-engine-link”
   href=”https://DuckDuckGo.com”
   title=”Click here to visit the DuckDuckGo search engine.”>
   DuckDuckGo
</a>
```

Much like with classes, we can target IDs via the attribute selector…

```
[id=”duckduckgo-link”] {
   color: rgb(255, 0, 0);
}
```

CSS offers a separate CSS selector to offer a shorthand. Instead of typing out the attribute selector, you just place an octothorpe before the name of the ID you’d like to target…

```
#duckduckgo-link {
   color: rgb(255, 0, 0);
}
```

Combinators

If you'd like to target an element somewhere inside of element, use a space between two selectors. This is the descendant combinator. The following selects any anchors that are anywhere inside of any paragraph.

```
p a { color: rgb(255, 0, 0 ); }
```

If you want to ensure you are targeting only elements immediately inside (only one level inside) of another, we use an angle bracket instead via the child combinator:

```
p > a { color: rgb(255, 0, 0 ); }
```

The adjacent sibling combinator is used to target an element that is directly beside (but not inside) another element. The following would target a paragraph element that immediately follows another (the first would not be targeted):

```
p + p { color: rgb(255, 0, 0); }
```

A bit less specific is the general sibling combinator. This targets an element following another, but it doesn't need to be directly beside it:

```
p ~ p { color: rgb(255, 0, 0); }
```

Selector List

You're able to apply a declaration block to more than one selection. Selections can simply be separated by commas in a selector list.

For example, the following could be used to apply rules to all heading sizes…

```
h1, h2, h3, h4, h5, h6 {
   font-family: Georgia, serif;
}
```

It is important to note that you can include combinators in items of a selector list. The following would apply underlines to any STRONG elements anywhere inside of any P, and any EM elements anywhere inside of any P…

```
p strong,
p em {
   text-decoration: underline;
}
```

Specificity

Remember when we mentioned the cascade? There is another metric to which rule wins out over others: specificity.

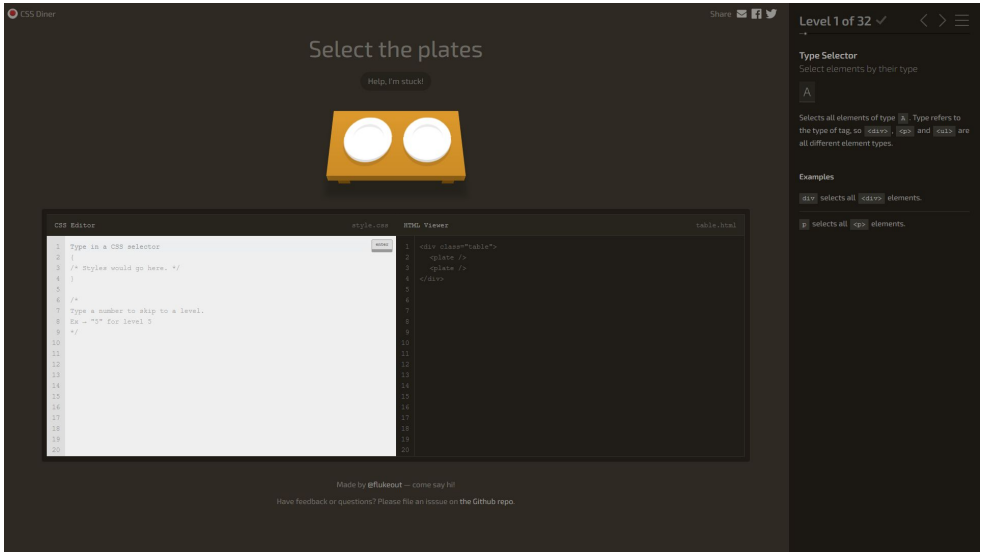
Specificity is the idea that if two rules affect the same element, the most specific one wins. If two are equally specific, then the cascade decides the last one will be the one that applies. The order that specificity takes precedence from most specific to least specific is as follows…

1. !important exceptions (refrain from using these except in emergency)
2. ID Selectors
3. Class Selectors, Attribute Selectors(, and Pseudo-Classes, which will come in the next lesson)
4. Type Selectors (and Pseudo-Elements, which will come in the next lesson)

Practicing Selectors

One of the leading free interactive CSS selector learning experiences is the CSS Diner.

This tool offers excellent exercises and practice for writing CSS code. Give it a try!



Recommended Reading

Level up your selector game by diving into these readings:

- Meyer, E. A; Weyl, E. (October 2017). CSS: The Definitive Guide, 4th Edition. O'Reilly Media, Inc.
  - Chapter 2. Selectors
    - Basic Style Rules
    - Grouping
    - Class and ID Selectors
    - Attribute Selectors
  - Chapter 3. Specificity and the Cascade
    - Specificity