# { Specificity. }

## Objectives:

By the end of this chapter, you should be able to:

- Define what specificity is
- Understand the difference between using element, class, id, inline and !important styling
- Compare and contrast ids and classes

## Specificity

So we've learned so far that CSS is built off of selecting HTML elements and their attributes and applying styling rules. But what happens if we have something like this (we are using a `style` tag just as an example)? One of the most important ideas around CSS is that of specificity, or how specific we are with our selectors. Imagine we have some CSS that looks like this:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        #main {
            background: green;
        }
        .container {
            background: blue;
        }
        div {
            background: red;
            height: 200px;
            width: 200px;
        }
    </style>
</head>
<body>
    <div class="container" id="main">
        I AM A DIV!
    </div>
</body>
</html>
```

`container` is a class on the single `div` element and `main` is an id on that `div` element... so we have 3 conflicting rules. Which one wins? The element styling? The class styling? The id styling?

Based on what you've learned, you might guess that the background of the `div` should be red, since CSS cascades and red is the last color to be set as the background color. However, if you open up this HTML page, you'll see that the background is green, not red! So what's going on?

To answer this question let's introduce a brief table for understanding. These weightings are not exact (10 element selectors does not equal a class selector), it is more an idea of showing you the magnitude of each kind of selector.

| Selector | Weighting |
|---|---|
| element | 1 |
| class | 10 |
| id | 100 |
| inline-style | 1000 |
| !important | 10000+ |

What this means is that a style rule on an id is weighed much more heavily than a style rule on a class. Similarly, a style rule on a class is weighed more heavily than a rule on an element. Put another way, id rules are more *specific* than class rules, and class rules are more *specific* than element rules.

This gives us another way to differentiate between classes and ids. Not only are ids meant to be unique, unlike classes, but also ids are more specific! This is also why the background color in our example is green: the id rule trumps the class rule, which in turn trumps the element rule. Cascading only applies when the rules have the same degree of specificity; in this case, rules about specificity determine the style, not rules about cascading.

Let's modify our earlier example a bit:

```
<!DOCTYPE html>
<html>
<head>
    <title>Specificity</title>
    <style>
        #main {
            background: green;
        }
        .container {
            background: blue;
        }
        div {
            background: red;
            height: 200px;
            width: 200px;
        }
        .winner {
```

```
            background: aqua !important;
        }
    </style>
</head>
<body>
    <div class="container winner" id="main" style="background:purple;">
        I AM A DIV!
    </div>
</body>
</html>
```

Here we see the two other types of selectors mentioned in the table above: an inline style rule and a style rule using the `!important` flag. Inline styles are styles declared using the `style` attribute on an HTML element. Both of these patterns should be avoided (we'll talk more about this down below), but for now, try to answer the following: In order of specificity (least to most specific), what is the background color?

## Another example

Let's try something a bit trickier. What happens if we have some code like this?

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        div.container {
            background: green;
        }
        body div {
            background: red;
        }
        div.container#main {
            background: purple;
        }
        body #main {
            background: magenta;
        }
        .container {
            background: blue;
        }
        div {
            background: yellow;
            height: 200px;
            width: 200px;
        }
```

```
        body div.container#main {
            background: brown;
        }
    </style>
</head>
<body>
    <div class="container" id="main">
        I AM A DIV!
    </div>
</body>
</html>
```

Which one wins? Try to calculate point values for each one! (1 for each element selector, 10 for each class selector and 100 for each id selector)

| Selector | Weighting |
|---|---|
| div | 1 |
| body div | 2 |
| .container | 10 |
| div.container | 11 |
| body #main | 101 |
| div.container#main | 111 |
| body div.container#main | 112 |

Therefore the order of colors (from least to most specific) is:

1. yellow
2. red
3. blue
4. green
5. magenta
6. purple
7. brown

We previously mentioned that using inline styling (a `<style></style>` tag) can have some unintended consequences. Strong specificity is one of them! Remember that inline styling is equal to 1000 points and is far more specific than any id, class or element! Even more, using `!important` will override ANY styling that you have (unless you have a more specific `!important`) so try your best to not use it. Not only does it make it more difficult to write future CSS, but it can get complicated quickly when working with other developers. If we catch you writing `!important` in your CSS, we'll ask you to refactor immediately!