All selectors and their specificity

As you build a website, the complexity of the code might increase to such a point that more than one CSS rule is applied to the same element. Or, you might accidentally add more than one rule over the same element. This results in conflicts as only one rule can be applied to a specific property. For example, the color of a certain p tag can either be blue or white, but not both. CSS engines use something called specificity to resolve these conflicts. Specificity is the ranking or score that helps CSS determine the final rule that will be applied to a given element.

This reading will help you grasp how the element with the ‘highest’ specificity is selected by CSS. But before you read on, it is important to note that these rules only apply in cases where conflicts arise for the properties.

**Specificity hierarchy**

CSS has a set of rules that it uses to ‘score’ or assign a certain weight to selectors and this creates a specificity hierarchy. Based on the weights, there are four categories in this hierarchy:

* Inline styles
* IDs
* Classes, attributes, and pseudo-classes
* Elements and pseudo-elements

**Inline styles**

Inline styles are attached to the elements within your HTML code like with the ‘style’ attribute. inline styles have the highest specificity. That means that rules that are defined as inline styles will be applied irrespective of other rules.

For example, take these two rules that create a conflict in color styling for a p tag:

1

2

 <p style=“color: white;”>

p{color: blue}





The p tag will be colored white because it is declared inside the inline tag.

**IDs**

Next in the hierarchy are IDs and by now you know that they are represented by ‘#’.  For example:

#div

**Classes, attributes, and pseudo-classes**

Classes, and the attributes inside the selectors, come next with what is called the pseudo-classes that you will soon learn more about.

For example:

**.my-class**

**p[“attribute”]**

**div:hover**

**Elements and pseudo-elements**

Finally, elements and something you call pseudo-elements have the lowest position in the specificity hierarchy. You will learn more about pseudo-elements later in this lesson.

**Calculating scores**

But by now you might wonder how is specificity calculated?

CSS uses the hierarchical model internally to calculate the specificity of the selectors used on a web page. But often as the size of CSS code increases, developers unavoidably face rule conflicts. In these cases, developers use the specificity hierarchy to calculate the precedence of CSS rules and to control the outcome of their web pages.

Let’s explore a practical example of how to determine the score of a few selectors.

**#hello {} will be 0100**

**div {} will be 0001 and**

**div p.foo {} will be 0012**

In the order stated above, the four categories will be assigned values 1000, 100, 10 and 1 with the element selectors having the lowest value of 1. These scores will be calculated respectively for each element present inside the selector. The total score for these elements is then calculated and the one with the highest value has the highest specificity.

Let’s explore a couple of examples for clarity. Take note that the properties and values are not included in these examples to keep the focus on the selectors only.

**Example 1**

1

2

3

p {}

div p {}

div p.foo {}





p  => 1 element =>  0 0 0 1 => Score: 1

div p => 2 elements => 0 0 0 2 => Score: 2

div p.foo {} => 2 elements and 1 class selector => 0 0 1 2 => Score: 12

The third case has a total of 12 for the p tag and so has the highest specificity. The rules for the other two cases are then overridden and the rules inside the third case are applied.

**Example 2**

p#bar => 1 element & 1 ID =>  0 1 0 1 => Score: 101

p.foo => 1 element & 1 class => 0 0 1 1 => Score: 11

p.p.foo => 1 element & 2 class =>  0 0 2 1 => Score: 21

By now it should be clear that the case containing ID has a much higher score and the rules inside it will be applied.

Once you learn about the different pseudo-classes, pseudo-elements, and wide range of selectors later in this section, it will be easy to see why understanding specificity is important.

While the weights assigned from the hierarchical structure help in a systematic approach, there are a few more guidelines and rules that become important especially in cases when the score for the different selectors is the same. Some of these are:

* Every selector will have a score and place in the hierarchy
* In the case of selectors with equal specificity, the latest or last written rule is the one that will be applied
* In general, ID selector should be applied in cases where you need to be certain about a rule
* Universal selectors have zero specificity value

This reading only gave you an overview of specificity, but you should know that it is a much broader topic and also the underlying basis on which CSS engines work. That's what the 'Cascading' in CSS means: the way in which CSS engines evaluate and apply the specificity rules is called ‘cascade’. Cascade is a type of small waterfall that falls in stages down the rocks and that is exactly how CSS behaves.

Don’t be too worried about applying what you’ve learned now, there are CSS specificity calculators available that can help you with determining the styling outcomes of your pages.