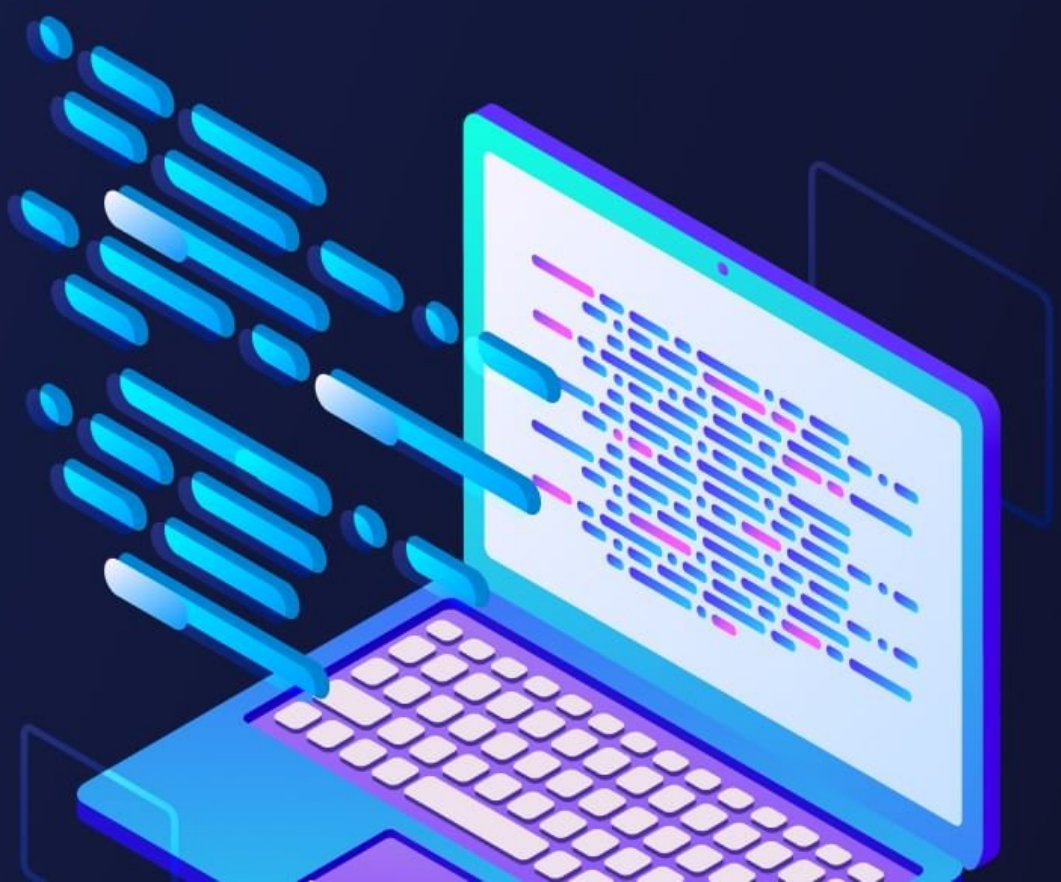
@LubnaDev • //lubna.dev

A Guide to CSS Specificity Rules



What is Specificity?

Specificity is the algorithm used by web browsers to calculate the weight of each CSS selector and identify which styles should be applied to every element.

How is it calculated?

There are 4 levels of selector categories. The higher level categories will carry more weight and override the lower level categories no matter the order the code is written in.

Level 1

Elements and pseudo-elements

```
/* Example */
h1 {
  color: #fa508f;
}
```

These selectors have the lowest specificity and can be overridden by most other types of selectors.

Level 2

Classes, attributes and pseudo-classes

```css
/* Example */
h1 {
  color: #fa508f;
}

.heading {
  color: #452d6d;
}
```

These selectors are more specific and will override any element selectors

Level 3

IDs

```css
/* Example */
#title {
  color: #1c3aa4;
}

.heading {
  color: #452d6d;
}
```

An ID selector will override a class or element selector even if it comes before them in the code.

Level 4

Inline styles

```css
/* CSS Example */
.heading {
  color: #452d6d;
}
```

```html
<!-- Example HTML -->
<h1 class="heading"
    style="color: #141318;">
  Hello World!
</h1>
```

Inline styles are the most specific and will always override any other type of selector.

!important

```css
/* Example */
#title {
  color: #1c3aa4;
}

h1 {
  color: #fa508f !important;
}
```

While not part of the specificity scale, using the !important rule will override any specific property. However, this is considered bad practice and usually discouraged.

Top Tip

CSS Specificity can be a real pain to debug, especially if you are working with others.

In order to prevent such bugs, it's best to stick to 1 type of selector like classes and nesting selectors too deeply.

Want more?

Follow
&
Leave a comment!

Alamin CodePapa

@CodePapa360

FOLLOW FOR MORE

Like Comment Repost