The Importance of Inheritance and Specificity in CSS

For styling HTML with CSS, the importance of inheritance and specificity is paramount. Inheritance is the process of child elements obtaining styles applied to their parents. It gives you more possibilities when coding in CSS for styling elements. Specificity is also very important when targeting elements with styles. It is imperative to know how to properly target elements using specificity as opposed to using id’s all throughout the markup.

The purpose of inheritance is to make CSS make more sense to the user. It does this by passing on attributes that would make sense to pass on to the child, such as font-family. In opposition, it also holds back attributes that would make no sense to pass on, like colors passing from a p element to an em element. I like to think of inheritance as a bouncer for a nightclub. He will stand out front of each child element and decide whether or not a style belongs in it. If the style makes sense to be inherited, it’s allowed in the nightclub, but if not, it gets bounced and isn’t allowed inside.

When styling a document, it is always poor practice to rely too much on id’s to target everything. Specificity is the best way to avoid making this mistake. It is basically targeting based on uniqueness. Say I have to style the bullet points for all the unordered lists in an HTML document. Without affecting the ordered lists in the document, I would have to target it like so:

Ul li { list-style-type: none; }

This makes it so the selector will only target list items inside of unordered lists. The more specific the selector, the more exclusive the style will become. Now how do you know how specific a selector is?

When looking at a CSS rule, you can determine the specificity value very easily. You simply add 1000 for every inline attribute, 100 for every ID, 10 for every class, pseudo-class, and attribute, and 1 for every element and pseudo-element. For example, the result of this CSS rule…

#sidebar ul li a:active { color: aqua; }

would be 113. 100 for #sidebar, 10 for :active, and 3 for ul li a. The resultant number can be compared to that of other CSS rules to find the most specific way of targeting a certain element. If there ever is a situation where it isn’t possible to target an element more specifically, it is best to use an id on the element to target it directly.

Often times when dealing with CSS, one element will be given many different iterations of the same style, such as its color. But in the end, how can you tell which will be applied to the element? Problems like this are solved by a system called source order. It takes over when an element has many styles of the same caliber applied to it. It’s very simple to understand, because the last style applied in order is the one that finally gets displayed.

Another way styles might be affected is due to origin style sheets. These are all of the style sheets that are applied to the final web page by the web browser, user, and of course the author. The web browser uses its own style sheets to help load basic HTML sheets. The author style sheets are the most common and are usually the ones that have the most effect on the page. Finally, the user style sheets trump every other styling that happens on the page, provided the user even has one.