CSS Selectors and Specificity

There are many different types of CSS selectors that allow you to target specific elements in HTML. So far, we've learned how to target by using elements names like <p> or <h1>.

We've also seen how we can target class and id names as well using #content for an id or .icon for a class, then it's more specific and instead of targeting every type of element like all <div>s or all <img>s we can be more *specific* in which ones to target. This is called specificity.

We saw a little bit of specificity when we talked about the order of CSS and how that matters. If the same exact selector had the same rule applied the one later in the CSS file would be applied and override the earlier one.

Think of specificity as a score or rank that determines which style declarations will be the one applied.

For example, if I declare a font-family for the whole page each child element inherits the font throughout the page. But if later I declare my h1 to have a different font family later in the CSS file, it is more specific, and all my h1s will have a different font. The body rule applies to all the text on the page, but the h1 rule is more specific so it is the one that will be applied.

Or I might have all my images set to a width of 100%, but I find that some of my images are just too big so I need to have a smaller percentage. So I might target just those images that I need smaller with their class name, for example .heat, to make just those images have a more specific rule set. Even though the width of 100% is still applied to all images, including them, the smaller width percentages wins out and will be applied because it is more specific. And here's where it gets interesting, even if that rule came before the img rule it would still apply because a class selector ranks higher in specificity than an element name selector.

Every selector will have a ranking in the specificity hierarchy. Inline styles as we saw earlier have a very high ranking. We are only going to be using an external CSS file so for us, the next more specific are id selectors like #logo-link, then classes and pseudo-classes like .card-img and .card-img:hover (which we will talk about in just a minute), then lastly the least specific are the element selectors like <p>, <h1>, <img>, etc. Keep this in mind when a rule seems to not work, you might have a more specific rule targeting that same element and over-riding your rule.

You may see some elements and classes used together in CSS. For example the .heat might be placed with the img tag using both element name and class, like this img.heat. This is just saying to target all img elements that have the class name of heat. Or I might want all the all a tags with the class of highlight a.highlight. This would come in handy if you've used the class highlight with different types of elements and want to specify which elements with that class name you want to target.

We can also list a number of element names, classes, ids etc and have the same declaration apply to all of them by separating them with commas. For example: h1, h4 {font-family: …} This would mean all h1s and all h4s would have that font-family and I wouldn't need a separate rule for each one.

We can get even more specific with descendant CSS selectors. For example, maybe we just want all the paragraphs inside our article to have line-height. So, we target just paragraphs that are children (or descendants) of an article element like this. article (space) p. The descendent would be the last element. So p's that are inside of an article element would be effected. Or maybe just the images inside our product gallery like this #prod-gallery img {…}

So, we could target images like we had before with a width of 40% but without giving them all a class name and just selecting them by either all the images in the main section or all the images inside the #img_container.

Visit w3schools at https://www.w3schools.com/cssref/css_selectors.asp for even more specific CSS selectors.

Let's also talk about pseudo-selectors and pseudo-classes.

One example of a pseudo selectors is the :nth-of-type() selector. If you have a number of the same type of element, it will let you select the first, second, third, or fourth etc of those elements. For example if I have a number of list items, I could select just the 2nd one without having to give it a class or id name by using the :nth-of-type like this. If I wanted the fourth one, I'd use a 4 instead of a 2.

Pseudo-classes allow you to change the appearance of elements when a user is interacting with them. For example if I want the a links in my nav to have a different background color or text color when the user hovers over them I can use :hover after the selector and give it the declarations I want to happen when the user hovers.

There are other pseudo classes like :active, :focus and :visited. See w3schools at https://www.w3schools.com/css/css_pseudo_classes.asp for more examples.