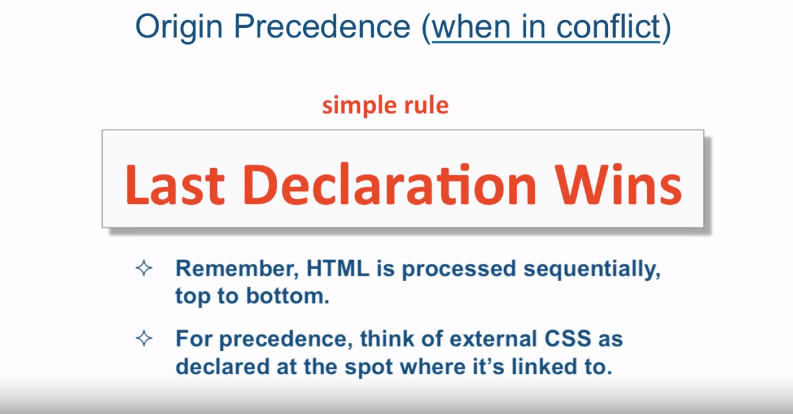
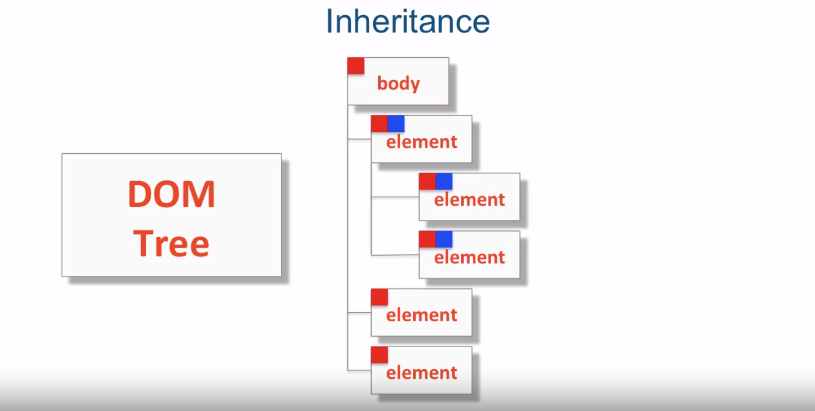
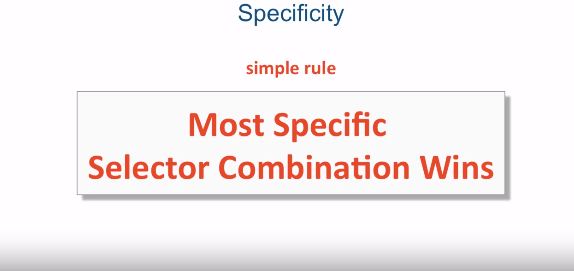


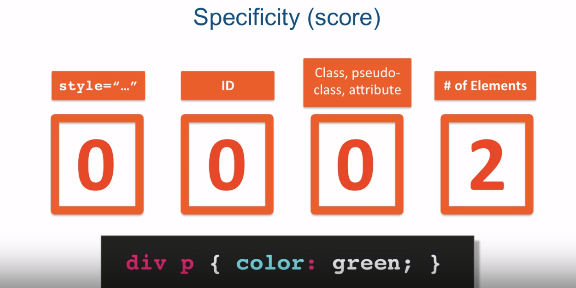
Origin: last declaration wins. Remember HTML is processed sequentially, top to bottom, the lower in the page they are the more precedents they have. For precedence, think of external CSS as declared at the spot where it’s linked to.

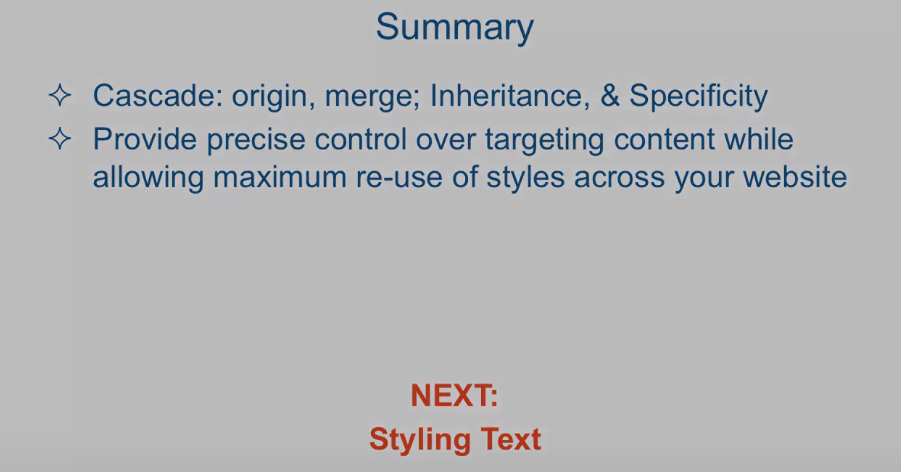
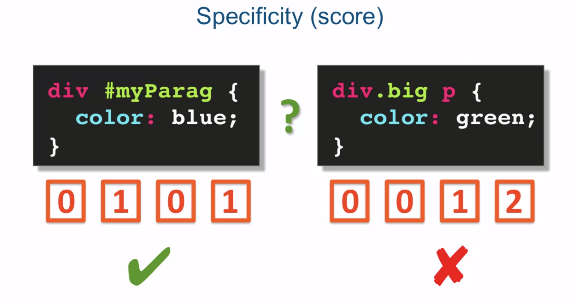












[MUSIC] Cascading is a fundamental feature of CSS. It's an algorithm defining how to combine

properties values originating from different sources. As the name itself suggests, cascading

style sheets, in other words the cascade algorithm, is at the core of

understanding and using CSS. The cascade combine the importance,

origin, specificity and source order of the applicable

style declarations to determine exactly which declaration

should be applied to any given element. And if there's a conflict,

how to resolve that conflict. In other words,

how to tell which CSS rule wins. There are a lot of terminology and concepts surrounding

the cascading algorithm. However, I believe to have a working

knowledge of the cascading algorithm, you should understand these four concepts. And these are origin,

or origin precedence, merge, as well as inheritance and

specificity. So let's tackle the first two,

origin and merge. When two declarations are in conflict, in other words they specify

the same property for the same target, origin precedence rule

kicks in, and it's a very simple rule. And the rule is,

the last declaration wins. Now, when trying to figure out

what the last declaration is, you have to remember that HTML

is processed sequentially. That means top to bottom. So as you see the declarations happen, the lower on the page they are,

the more precedence they have. And also for precedence, think of external CSS as declared

at the spot where it's linked to. So usually it's declared

in the head somewhere, but there could be other styles

that I declared in the head. And to figure out which

one was declared last, imagine that the entire contents of

an external CSS file were cut and pasted straight into the head portion,

where that external CSS is declared. When different CSS declarations

do not conflict, that is, they still target the same element, but

the CSS properties with which they target that element are different,

there's even a simpler rule. And that is that declarations merge. So a declaration for, for example, font

size, and a declaration for color, since they're two different properties, when

they're targeted to the same element, even if they're targeted from two different

origins, they will merge into one. And the element will get both

the font size and the color. Let's take a quick look at

an example of that in code. Okay, so here I am in Sublime Text and

I'm looking at origin.html. And it's located in

the examples Lecture17 folder. Let me go ahead and hide the sidebar so

we could see the file a little bit better. So to go over the structure of our HTML,

in the head we have a link that links our external.css, which is also located

in the same folder as this file, and you could see that file is right here. And it specifies that every p, every paragraph tag,

should have font size increased to 130%, the background will be gray, and

the color of the text will be white. Now, right after that,

we have a style tag, again, still in the head,

that overwrites the p element and says that the color, that is,

the text color, should be maroon. Now, when we get to

the actual viewable content, we have a couple of paragraphs here. The first one is just a simple paragraph

and the second one is a paragraph where we've declared an attribute style and

specified color of the text to be black. So let's take a look at

the first paragraph. You could see that the background

of the paragraph is gray and the text color is maroon. Now, how do we get this gray and maroon? Well, let's take a look. We first declared external.css. That's the first thing that

comes in the HTML file. So remember, I told you to think about it

as if we basically took these contents and pasted them straight into this spot. So basically we've declared we want

the font size to be increased, the background color should be gray,

and the text color should be white. However, right after that,

we turned around and overwritten that with another color declaration and said

that the color in fact should be maroon. And that's really the last thing that we

could see as far as the text color of our paragraph text. And that is why this paragraph right here

does in fact have its text color maroon. That is because of the last

declaration of that color, one. The second paragraph, however, has its

text color black because the very last declaration of the text color is black,

so therefore the text color is black. So that's our origin

precedence rule in action. However, unlike the color property,

we have other properties here, which are the background color and

the font size, that are still being applied to both paragraphs and

that's our example of a merge. In fact, if we right-click on

this paragraph and go ahead and inspect the element, we can see that

the first paragraph does in fact have the color: maroon and

it's coming from origin.html, line 9. That's our style tag

inside our head section. But you could see that it's showing

you that other things apply as well, that background color is gray and

the font size is 130. And the Chrome developer tools are clearly

showing you here that we've overwritten the color that was specified in

external.css with our color: maroon. And that's why you see

the color crossed out here. The next concept we're going to

take a look at is inheritance. And it's a pretty simple concept. The basic idea is that you have

the document object model tree. And if you specify some CSS property

on some element, all the children and grandchildren and so on and so

on of that element will also inherit that property without you having to specify

the property for each and every element. So for example,

if I specify one property on the body tag, every element that is

a child of a body tag, or even a child of a child of a body tag and

so on, will inherit that property. Similarly, if I specify a property on

some element within my HTML page, every child and grandchild and so on of that

element will also inherit that property. But obviously, no parents of that

element will inherit that property and therefore will be completely

unaffected by it. So let's take a look at what

inheritance looks like in code. While inheritance in CSS is a pretty

simple concept to understand, it happens to be a very

powerful technique as well. It allows me to specify one rule and have all the children of the target

element inherit that rule. Let's take a look at inheritence.html, which is also located in the same

folder as the previous file. It's a very common practice to specify

some declarations for the body tag and the reason is obvious. Body is the top element in our HTML

as far as the view port is concerned. In it, I can specify some rules for

the entire HTML document and then using our cascade origin precedence,

overwrite some specific rules a little bit lower down the line, while keeping

everything else very consistent. So here in our example, as you can see, I specified that any text appearing

in our document should be color: red. And I also specified that everything

should be align: center and, which is very common,

to specify one font that will be consistent throughout the entire HTML

document, in this case, Helvetica. Then what I did was I specified

a particular color of the text just for paragraphs. And even though each paragraph

will inherit color: red, we're here to override

that with color: blue. Last, in the H2 tag here a little

bit lower to the bottom, I'm using again the origin precedence rules to

over write the text color to green for this particular subheading too. And as you can see, our H1 here in

the browser is red and so is the first H2. And they're red even though I never

specified H1 or H2 to be red. I just said that anything inside the body,

color should be red. And you notice that every single block

level element here is centered, and the reason it's centered is because

I specified again in the body that the text-align should be centered. Same goes for font-family. It's Helvetica throughout

the entire document. Again, I never specified text-align or

font-family for every single element in the page, but I did specify

it for the parent element, the body tag. So as you can see,

this is pretty powerful. Okay, so since this lecture

is getting a little too long, I'm going to split it up and

we'll finish specificity in part two.