Cascading in CSS:

1. .my-class{

color: blue;

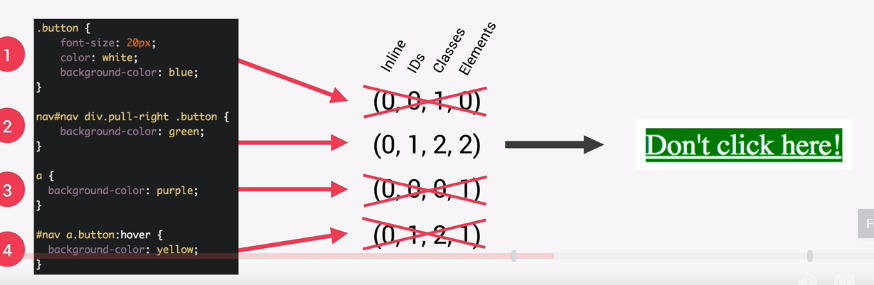
text-align: center;

font-size: 20f;

}

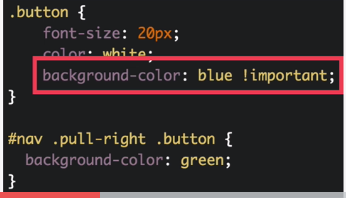
In the CSS code above, the ‘.my-class’ would be the selector, the curly braces would be called the declaration block, and each of the lines in the braces (e.g. color: blue;) would be the declarations.

1. In a Cascading Styles Sheet, the most specific selector is the one that is chosen, which means in this case the second selector would apply and the background-color would be green.

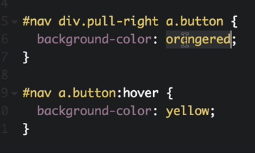


1. If two or more of the selectors would have the same amount of specificity (e.g. if two or more of the selectors were (0, 1, 2, 2)), then the last selector in the CSS would be applied.
2. The CSS declarations marked with !important will have the highest priority. It should only be used as a last resource.

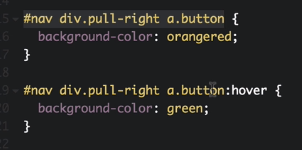
e.g.:



1. It is wise to rely more on specificity than the order of selectors. Orders are more important if you’re using 3rd party external style sheets as the author stylesheet will need to be last.
2. In the case below, the hover won’t work for the button because it has low specificity:

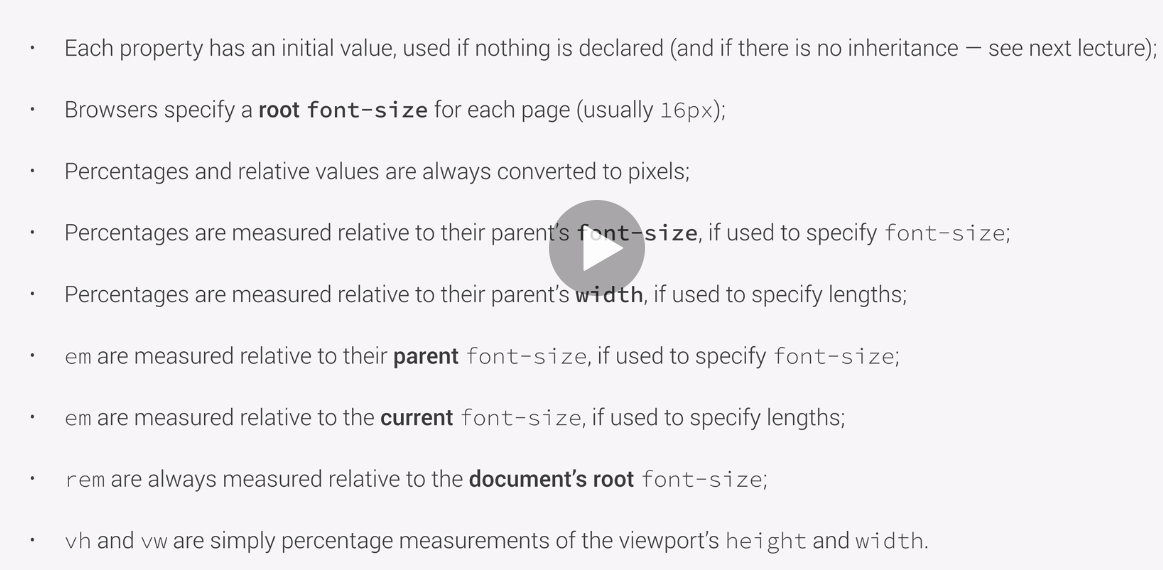


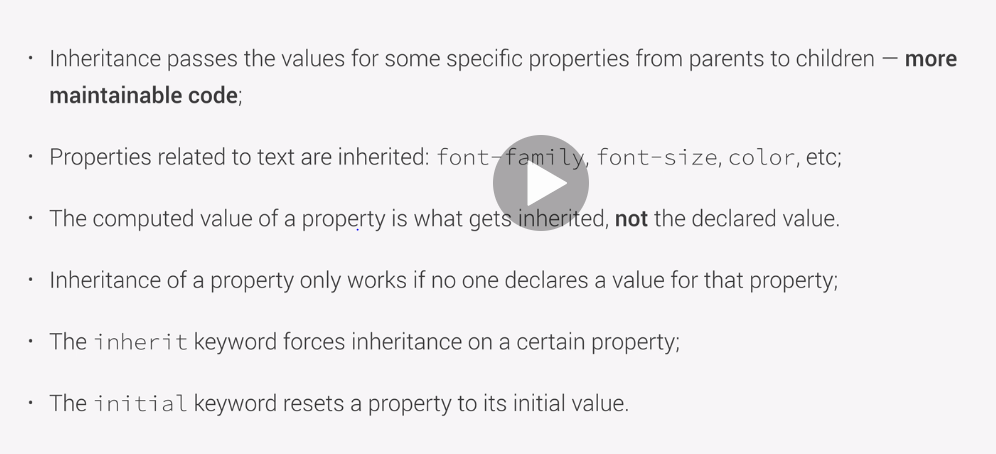
In order to make the hover work, you have to have a selector with a higher specificity, like the following:



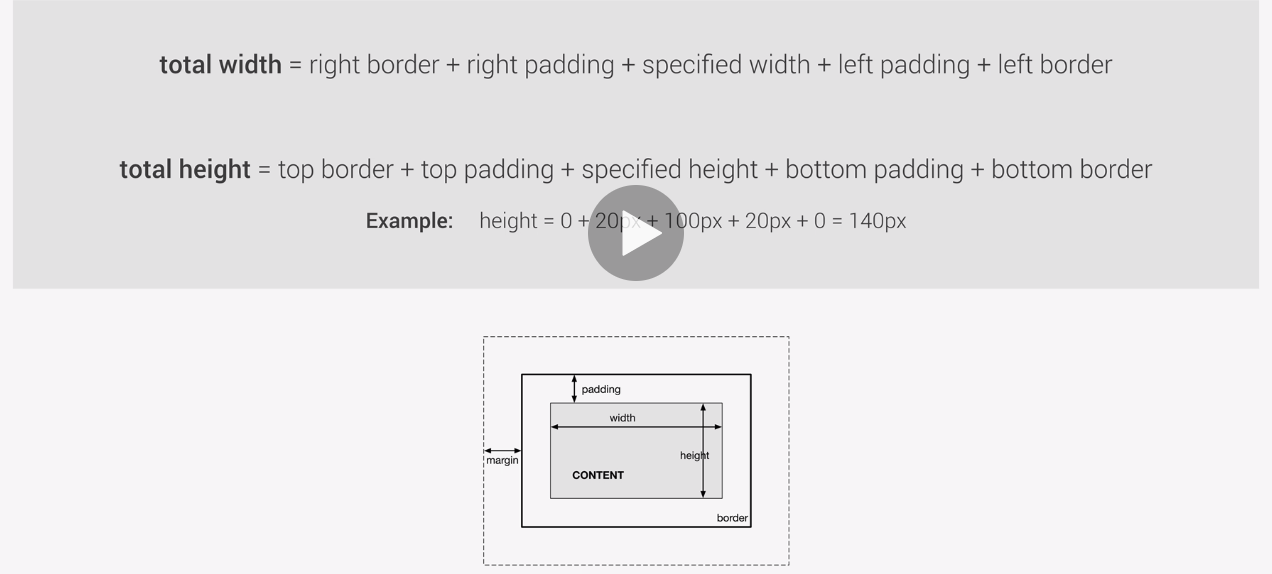
1. The fontsize will be 150% larger than the parent element(In this case, the html, body{font-size}):



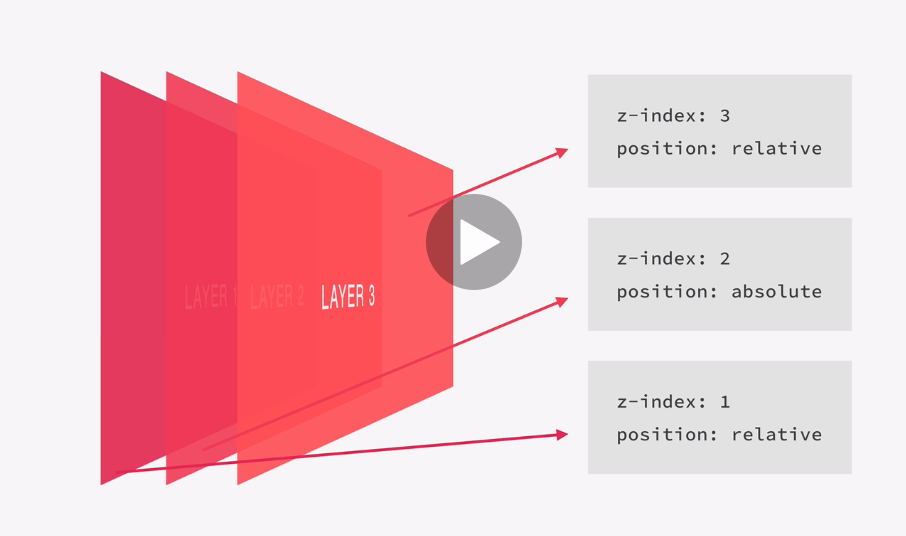
1. Some important points to take in : 
2. Some important points to take related to inheritance:



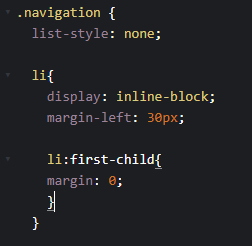
1. Border, Padding and margin for a box.



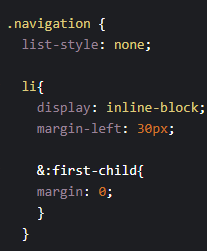
1. Z-Index Layout:



1. In SCSS, you can define a variable name using the dollar sign ($).
2. Here is an example of a nested SCSS:



1. For the li:first-child, to reduce redundancy, we can change it to:



1. For the .btn-main:hover, the 15% next to the color declaration means the color is going to be 15% more darker than the color itself because we added darken in front of the properties:



1. We’ve added compile:sass to our package.json:
2. {
3. "name": "c",
4. "version": "1.0.0",
5. "description": "",
6. "main": "index.js",
7. "scripts": {
8. "compile:sass": "node-sass sass/main.scss css/style.css -w`"
9. },
10. "author": "",
11. "license": "ISC",
12. "devDependencies": {
13. "node-sass": "^4.13.1"
14. },
15. "dependencies": {
16. "sass": "^1.26.3"
17. }
18. }
19. To run live server we input live-server in bash while on root directory, and to run the compile:sass we type in npm run compile:sass.
20. Fluid grids and layouts use % instead of px for all layout-related lengths.
21. There are three layout types:
    1. Floats: where we put a bunch of boxes, side by side, using floats
    2. Flexbox: Laysout elements in a one dimensional row.
    3. CSS grid: great for creating an overall layout of a page in a fully-fledged 2d grid.

However, in this course of SASS, we focus more on the use of float layouts only due to support reasons during the time of the course, but also touch bases on flexbox and CSS grid.

1. Difference between width and max-width: max-width means if we have enough space, than we will have enough width that we’ve specified, or else it will fill 100% of the available space.
2. The shortcut for writing a
3. BEM -> Block Element Modifier
4. &nbsp; in html will create an empty space
5.  -> This will create a parent element with a class called row, and 2 children classes called col-1-of-2
6. Ease in and out using cubic-berzier example:

 transition: transform .8s cubic-bezier(0.83, 0, 0.17, 1);

More on <https://easings.net>

1. The checkbox and button are siblings and the icon is the button’s sibling.

&\_\_checkbox:checked + &\_\_button &\_\_icon {

    }

1. Transform-origin makes allows an element to rotate on the specified side. (e.g. for a rotating horizontal line with transform-origin: left, the left point will be chosen as the center point of rotation and for a rotating horizontal line with transform-origin: right, the right point will be chosen)
2. 1em = 16px;