1. What happens in the background.
2. Three pillars of writing good html and css.  
   Responsive design :   
   Writing maintainable and scalable code :   
   Web Performance :
3. Responsive web design:   
   one website that works beautifully on all screen sizes and all devices.

Here we need to think about responsive images, using correct units for font-sizes or element dimensions, even includes a desktop first or mobile first strategy.

1. Writing maintainable and scalable code:  
   clean, easy to understand, supports future growth and most importantly re-usable.  
   This also includes how to organize files, how to name classes, how to structure html.
2. Web Performance: less code , compress code, less http requests, use css preprocessor,mostly reduces the use of images by using the images which are really necessary for a website + compressing those images so that they consume less bandwidth for the user.
3. HOW CSS WORKS BEHIND THE SCENES.
4. What happens to our css code when we load a webpage.  
   - browser loads html  
   - browser parses html + loads the stylesheets and parses it which is a bit different than  
    how html is parsed.  
   - browser build the DOM like a family tree.
5. Two main steps in CSS parsing phase :   
   8.a : Conflicting css declarations are resolved in a process called cascade.  
   8.b : process final css values like converting a margin defined in percentage units to pixel   
    units.

After all of this is done final css is stored in a tree like structure called the CSS object model.

1. After the html and css are parsed and stored : these two together form the render tree.
2. Then the page is rendered. The browser uses the visual formatting model to render the page.  
     
    ---------\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*---------
3. CSS parsing phase.  
   step 1: cascade : it is the process of combining different stylesheets and resolving conflicts between different css rules and declarations – when more than one rule applies to a css element.  
   CSS can come from different sources.  
   **Developer written CSS**: they are called author declarations.

**User declarations**: CSS coming from the user i.e in the browser.  
**Default browser declarations**: user agent CSS as it is set by the browser.

1. But how does CSS resolve conflicts.

It looks at the IMPORTANCE > *selector* SPECIFICITY > SOURCE ORDER of conflicting declarations in order to determine which one takes precedence.

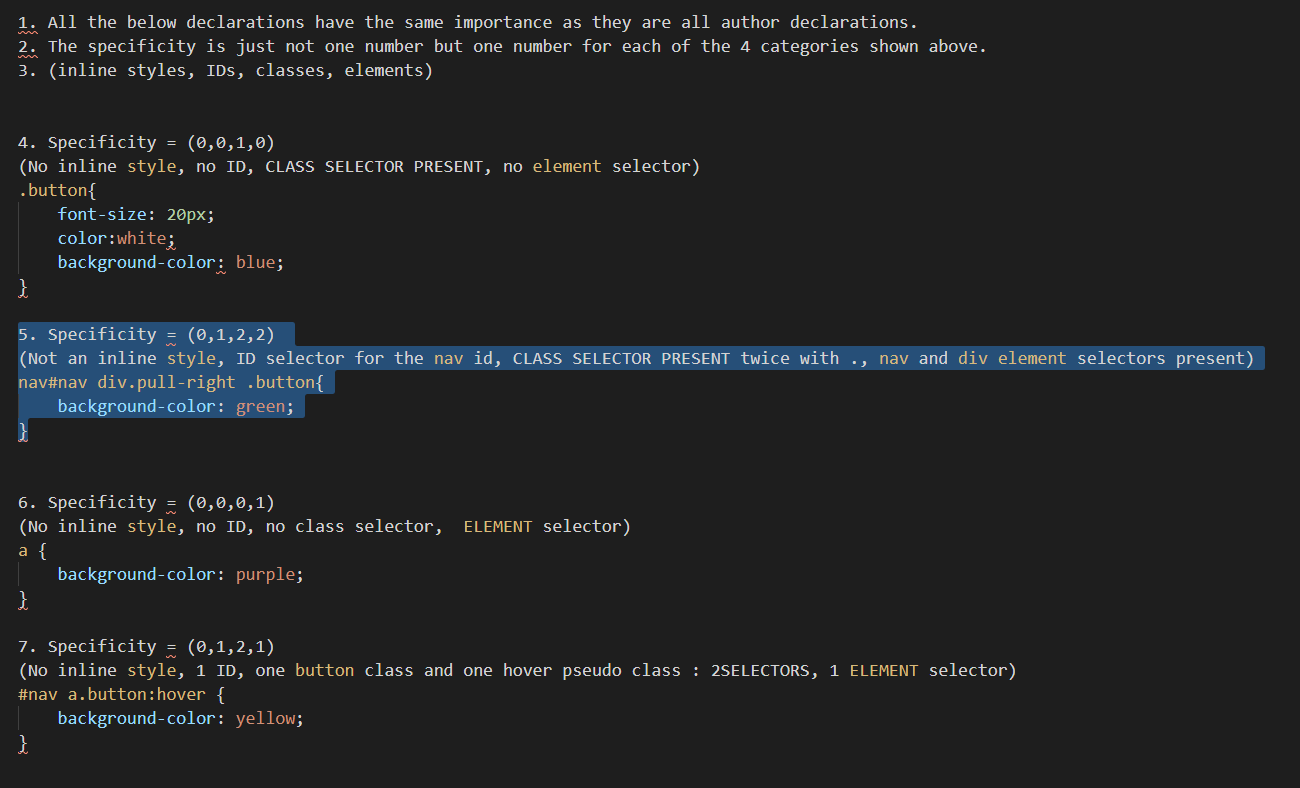
1. First the cascade starts by giving conflicting declarations different **importance** based on where they are declared / based on their source. The order is below

* Most important declarations are the user declarations marked with !important keyword.
* Author declarations marked with an !important keyword.
* Author declarations
* User declarations
* Browser declarations.

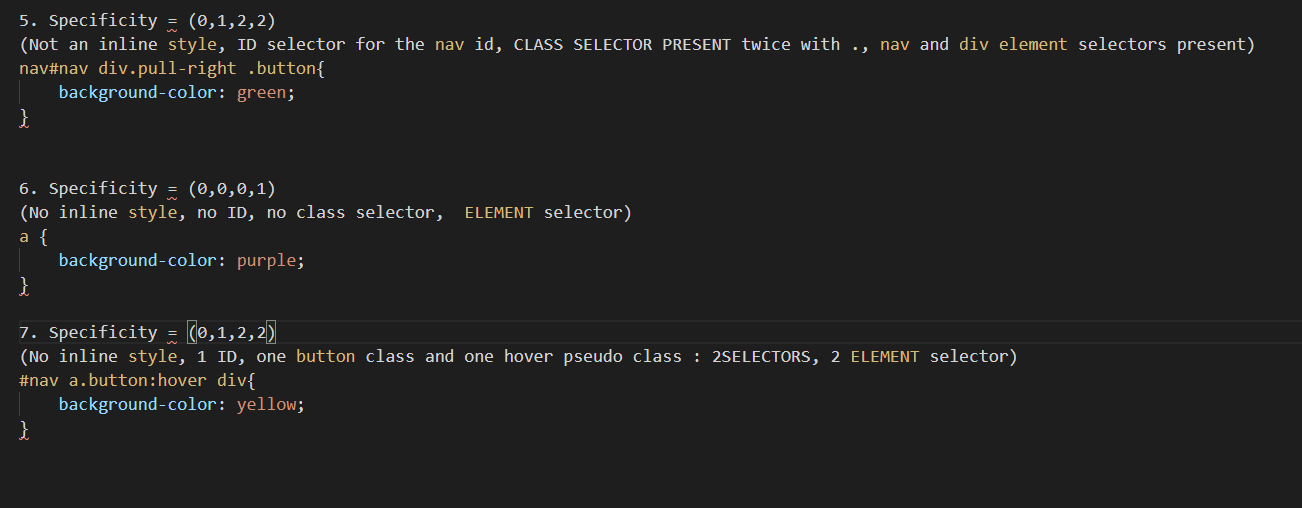
1. When we have conflicting declarations with the same importance we calculate the selector specificity based on the priorities shown below.

Now cascade calculates and compares the specificities of the declaration selectors.

* The specificity order is below.
* Inline styles : they have the highest specificity.
* IDs
* Classes , Pseudo classes, attribute selectors
* Elements, pseudo elements.

Example : How to calculate specificities.  


* Selector number 2 is the most specific selector of all.

1. Now let us say of if point 5 and 7 have the same specificity like below ..i.e if all the declarations have the same specificity.. then the last css declaration written in the code is the one that will apply.( SOURCE ORDER)  
     
   Universal Selector has zero specificity (0,0,0,0)  
   Put your stylesheets in the last of all the included stylesheets whichmay come from 3rd parties.   
    ---------\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*--------
2. How values are processed in the parsing phase + how units like em,rem,%,pixels are calculated.
3. How declared values are processed in six different steps starting from the declared values to the final actual value.  
   Lets analyze the ‘width’ ‘padding’ and ‘font-size’.
4. Lets begin with width-paragraph.  
   Declared Value : 140px via ‘p’ element selector  
    : 66% via ‘amazing’ class selector  
   Cascaded Value : As the class selector has more specificity 66% is picked.

Specified Value : It is the default value of a certain css property.

In this case its ir-relevant as we have a cascaded value already.

Computed value : In this step values with relative units are converted to pixels, So  
 that they can be inherited.  
 In this case we have a percentage value which is technically not a  
 unit.So nothing happens in this step.  
Used value : In this step the css engine uses the rendered layout to figure out some   
 of the remaining values.Ex percentage values that depend on the   
 layout. The 66% we specified is in relation to its parent element.  
 So the parser needs to know that width, in order to calculate the  
 paragraph width.  
 The parent element i.e ‘section’s ’ width is defined as 280px.

(66\*280)/100 = 184.8px – this is now our used value.

Actual Value : Browsers can not usually display 184.8 pixels and so they are rounded  
 of to another value 185pixels.This value will be used in the layout.

1. Padding property for the paragraph.  
   There is no declaration for the padding property for the paragraph .  
   Each and every CSS property should have a value even though it is not declared.And thus no cascade value.  
   Declared Value : X  
   Cascaded Value : X  
   Specified Value : Each CSS property has a initial value, which is used when there is no  
    cascaded value.i.e if we/browser/user don’t declare a value then   
    the initial value is used.[inheritance should be taken into   
    account].The initial value for padding is 0 pixels.  
   Computed + Used + Actual Value : There is not more calculation to do as its already   
    an absolute unit.  
     
   
2. Now the font size property of the root element.  
   Declared Value : We have not defined it anywhere. So no declared value.  
   Cascaded value : The browser has a default value of 16 pixels.We know that CSS   
    could come from different sources.In this case the font-size is a   
    user agent declaration.  
   Specified Value + further : no more calculations needed and 16 pixels is used.
3. Font-size of an element having section class.

Declared-value : 1.5 rem which is a relative unit.

Cascaded-value : 1.5rem  
Specified-value : 1.5 rem

Computed Value : 24pixel  
 1.5 \* 16pixels.  
 rem unit is always relative to the root font-size.  
 Used + Actual Value : 24 pixels.

1. Font size of the paragraph element.

Declared Value : X.  
Cascaded Value : X

Computed Value : Some properties like the ones related to text such as the font-size inherit the computed value from their parent elements.24 pixels.

1. How does the CSS engine convert relative units to pixels to calculate computed and used values.

Relative units are most important for building responsive websites.

How different units work in different situations.

* First lets start with percentages.
* There is a difference in using percentages for fonts and lengths/distance measurements.
* %+font : Now the header has a font-size of 150%. This means that the header will have a font-size 150% larger than its parent element which is the body element with a font-size of 16.
* %+Length : When we express a length measurement in percentages, like height,padding,margin – the reference is always the parent elements width. For Example consider the element which has this class .header-child, it has a padding of 10%.As its parent element has 1000px of width, 10% of 1000px is 100px.Remeber that the parents width is taken as a reference.
* Next are font based relative units em,rem. Its different to use em for fonts and for lengths.  
  Both ems and rems are font based.
* Em – uses the parent element or the current element as the reference.
* Rem – uses the root font size as the reference.   
    
    
  