1. **[Understanding em Units in CSS](https://www.impressivewebs.com/understanding-em-units-css/)** and rem

<https://snook.ca/archives/html_and_css/font-size-with-rem>

<https://www.impressivewebs.com/understanding-em-units-css/>  
  
html { font-size: 62.5%; }

body { font-size: 14px; font-size: 1.4rem; } /\* =14px \*/

h1 { font-size: 24px; font-size: 2.4rem; } /\* =24px \*/’

1. <https://www.freecodecamp.org/news/learn-flexbox-build-5-layouts/>

flex-basis

flex-gap:1

flex-grow

1. [**https://www.w3schools.com/cssref/css\_selectors.php**](https://www.w3schools.com/cssref/css_selectors.php)

[class^='icon-'] - classes starting with 'icon-' (eg. class='icon-blah blah')

[class$='-icon'] - classes ending with '-icon' (eg. class='blah blah-icon')

[class\*='icon'] - classes containing 'icon' (eg. class='blah xxx-icon-blah')

1. **Use VH with calculated height for any elem**
2. [**https://www.w3schools.com/cssref/playdemo.php?filename=playcss\_border-left**](https://www.w3schools.com/cssref/playdemo.php?filename=playcss_border-left)
3. **Use the resetting SCSS**

**html {**

**font-size: 62.5%;**

**}**

**\* {**

**margin: 0;**

**padding: 0;**

**box-sizing: border-box;**

**body {**

**width: 100%;**

**min-height: 100vh;**

**font-family: sans-serif;**

**font-size: 40px;**

**}**

**}**

**$gap: 4vh;**

**$padding: 4vh;**

**$color: #48cae4;**

**$bp: (**

**mobile: 480px,**

**tablet: 768px,**

**desktop: 1440px**

**);**

**@mixin query($display) {**

**@each $key, $value in $bp {**

**// defining max-width**

**@if ($display == $key) {**

**@media (max-width: $value) {**

**@content;**

**}**

**}**

**}**

**}**

**7. Specificity in CSS**

**i) div p{ // 1 + 1**

**}**

**ii) p{// 1**

**}**

**iii).className{ // 10**

**}**

**iv) #id{ // 100**

**}**

If elements is having same specificity whr then the styling written later will be applicable

1. If we want to create variables in CSS we can create them inside html tag to make then global so that we can use in other classes. We can also write the fallback CSS values for any variable

A black background with blue and yellow text

Description automatically generated

Yellow is the fallback css for –bgdiv. If no bgdiv var found yellow background gets applied.

1. Calc : Allows you to compute in CSS :

Calc(32px + 10vw)

Calc is useful when different units are involved and mainly for diff screen size we can calc wrt vw or vh.  
  


**For + and – we need to give space around the operator or else it will be an invalid expression.**

1. Pseudo selector:

These means, the particular element is nt really exists on the page, but either the state or some user interaction causes the element to get added.

Example:

li:hover

Input :focus + #banner{

}

Ul li:nth-child(3){ // selects 3rd child, 3 is an index and it is 1- based

Background: black;

Color: black

}

Ul li:nth-child(2n){ takes all children at the index 2,4,6 and so on

Background: black;

Color: black

}

Ul li:nth-child(2n-1 or 2n+1){ n value starts from 0 here

Background: black;

Color: black

}

Ul li: last-child{

}

Ul li: first-child{

}

1. Pseudo Element:

These are written with :: and actually these acts as element and gets added.

#banner::before{ // get appended when just the tag starts

Content: “Hello World”

}

#banner::after{// get appended when just the tag ends

Content: “Bye World”

}

If **we remove the content** then it will go away from the DOM and will not be visible on the web page, even if we give display: block

As these elemes are nt part of the actual DOM, we cannot access them from JS code

#banner::first-letter{// targets the first letter of the pseudo content

Text-transform: lowercase

}

#banner::first-line{// targets the first letter of the pseudo content

Text-transform: lowercase

}

We can apple pseudo selectors to pseudo elements as below:

#banner::before:hover{ // this doesn’t work as the before pseudo ele is nt present in the dom

Text-transform: lowercase

}

#banner::hover::first-letter{ // this works, here we r selecting the sate first

Text-transform: lowercase

}

They don’t work with self-closing tags <input> <hr></br>

1. Pseudo Element with attr content(can be any attribute)

A black background with text and a black rectangle

Description automatically generated

A black background with green text

Description automatically generated

Result:

A red and green square with white rectangles

Description automatically generated

We can have anything get append **to data-** to pass more data in the html.

A black screen with white text

Description automatically generated

1. We can’t apply margin or padding top or bottom to an inline HTML element, only margin left and right is applicable.
2. If we add padding to a block element it moves the next element to get those padding space applied to the block element. The padding decides the distance between content and border so it pushes the next content.

But

In case of inline element it doesn’t move the next element, it overlaps

We can add display: inline-block property to the elem and apply required margin and padding

Inline-block – the elem gets displayed as inline ele but we will be able to apply margin and padding.

1. Margin-collapse – it happens between two adjacent elements depending on how we have set the margin for both the elements. The larger value gets applied and the small margin value gets avoided. IT is applicable only for vertical margins and it happens with only block elements.

Why is this there because let’s say we have list of items and we have applied margin:32px, then we see that the vertical gap between any two list is 32px not (32+32)px; margin collapsing makes sanse

1. We have two types of inline elements – replaceable and non-replaceable inline elemets.

Non-replaceable inline elems - <a>

Replaceable inline elems - <img> as the element itself is getting replaced by the img

Most elements in HTML are non-replaced.

A non-replaced element is simply an element that is not a replaced element

Vertical margins will not have any effect on non-replaced inline elements but we can apply them to replaceable elems.

1. We also have attribute selector exa-
2. A screenshot of a computer

   Description automatically generated

1.HTML----

1. http-equiv meta tag
2. <input type=”search”>
3. <input type=”range”>
4. <input type=”date”>
5. <input type=”button”>
6. <input type=”datetime-local”>
7. HTML entities
8. Create same group of checkboxes by adding the same name attribute to them.
9. Logical and physical tag : The logical tags are useful for software which reads a screen or web page
   * 1. <b> and <strong> tags works similarly to make any text bold but <strong> tag is logical than b as it shows some seriousness of the content which is useful in case of screen readers
     2. <i> and <em> - makes text italic, <em> is more logical
     3. The semantic tags <header>, <footer> all r logical tags