Cascading Style Sheets (CSS)

Anatomy of CSS Rule

```
<Style>
  Selector => Tag to which style is to apply,
    ↓          it is paragraph, headline
  P {
      color      : blue;
      font-size  : 200 px;
    }
```

Declarations which property and value separated by Semicolon (;)

properties ↑ ↑ values

```
  h1 {
      color : green;
      text-align: center;
    }
</Style>
```

This whole is Style Sheet.

Selectors

It used to html element to which we want to style.

1) Element Selector

Selector which uses tag name as a Selector for Styling.

```
P{
   Color : Blue;
  }
```

2) Class Selector

we define Selector as a class with `.class name` Selector and this class is assigned to required tags many classes can assigned one element as

`Class = "Blue Red"`

```
.Blue {
      color : Blue;
    }
```

```
<p class = "Blue">.... </p>
<p class = "Blue">.... </p>
<p> ..... </p>
```

↑ unaffected

↑ Blue Text

3) id Selector

```
# name {
      Color : blue;
    }
```

```
<p>- - - - </p>
<div id = "name"> - - - </div>
```

Here "id" is given to div element from that id div element to which Style is to be applied is identified and inside <Style> we use `id with #` and assigned style to it.

Combining Selectors

1) Element with class Selector.

```
p.big {
    font_size = 20px;
}
```

```
<p class="big"> ... </p>
<div class="big"> ... </div>
```

only p elements with given class are affected others are unaffected.

2) Child Selector

Every p of article which is direct child gets the style.

```
article > p {
    color : blue;
}
```

```
<article>
  <p> ----- </p>
  <p> ----- </p>
</article>
```

But p should be direct child.

3) Descendant Selectors

```
article p{
    color = blue;
}
```

```
<article>
  <p> ---- </p>
  <div> <p> ... </p> </div>
</article>
```

Every p element inside article element irrespective of whether it is direct element & not gets that style.

Various Selector combinations

```
.colored p{
        color: blue;
    }
```

Every p element inside elements with class colored gets style.

```
article > .colored {
        color: blue;
    }
```

Every element having class colored and inside article and direct child gets style.

Adjacent Sibling Selector

```
div +p {
        background-color: yellow;
    }
```

adjacent Sibling is the element next to get element at same.

```
<div>
    <p> ... </p>
    <p> ... </p>
</div>
<p> this element gets it </p>
```

General Sibling Selector

```
div~p{
        background-color: yellow;
    }
```

All the Siblings given element.

```
<div>
    <p> ... </p>
    <p> ..... </p>
</div>
<p> This gets </p>
<p> This also gets </p>
```

pseudo - class selectors

```
Select : pseudo-class {
                property : value ;
            }
```

Selector are choosen according to rule studied and pseudo-classes. classes can be as follows:

1) link - If element is a link
2) visited - If element visited
3) hover - If mouse hovers over
4) active - If mouse click on it but not released.
5) nth_child (α) - particular level of child's of element is selected.

Eg:

1) a : hover {
        color : blue ;
    }

Hovering over link in a given blue colour to it.

2) p : hover, a : active {
        color : green ;
    }

Hovering over paragraph and clicking on link changes them to green.

CSS pseudo - Elements

CSS pseudo - element is used to style specified parts of an element.

Eg:
1) Style the first letter, & letter of an element.
2) Insert content before or after the content element.

```
Selector :: pseudo - element {
                property : value;
            }
```

1) first - line : It styles first line of element only.

```
P:: first - line {
        }
```

2) first - letter : It styles first letter of elements.

```
P:: first - letter {
        }
```

3) before : It is used to insert something before like emoji with content property.

```
P:: before {
        content : url (smiley.gif);
    }
```

4) after : used to insert after content of elements.

5) selection : when element content is selected by user.

```
P:: selection {
        color : red;
        back-ground : blue;
    }
```

Attribute Selectors

It is used to style elements having specific attribute.

Syntax

```
Element [attribute = "value"] {

                                }
```

Eg:-

```
a [target = "_blank"] {
                        background-color : red;
                    }
```

It styles link having `_blank` as target attributes with background color red.

Other formats

1) element [attribute = "value"]
   Attribute with Specific value.

2) element [attribute = "value"]
   [title ~= "flower"]
   when flower is one word inside title.

3) element [attribute | = "value"]
   [title |= "flower"]
   when title start with Space Separated | (-) Separated flower word.

4) element [attribute ^ = "value"]
   [title ^ = "flower"]

when title start with flower word flower may not be Separated like `FlowerOfLotus`

5) element [title * = "flower"]

[title * = "flower"]

when flower coord occured some where like in [w=] but may not be separated by space & (-) like myflowerisis is title.

⇒ Attribute selector is most useful for using CSS for forms..

Style placement

1) Style can placed in elements called "Inline Style".

<element style = "property: value;">----</P>

Eg:-

<p style = "text-align: centre;">----</P>

This method is least used because it is least reusable.

2) External Style

<head>

<link rel = "stylesheet" href = "style.css">

</head>

style.css

```
body {
    background-color: green;
    font-size: 130%;
}
```

In this method external Styling Sheet is used and its reference is given with link.

This method is used in real world application. It is useful when there are lots of pages to style as particular way.

3) Head Styles - Style with <Style> tag (Internal)

```
<head>
  <Style>
      P{
        Color: red;
       }
  </Style>
</head>
```

Head Style is used while overriding the external styles in real world application.

Back ground

1) Background color

i) background-color: a) "color name";
b) rgb(r,g,b);
c) rgba (r,g,b,a);
d) hsl (h,s,l);
e) hsla(h,s,l,α);
f) #000000;

ii) opacity : (0-1);

2) Background image

i) background-image : url ("url format");

ii) background-repeat : repeat;
no-repeat;
repeat-x;
repeat-y;

(iii) background-position : right top; right bottom;
inherit; left top; left bottom;
(x%, y%); right center; left center;
(xpx, ypx); top center; bottom center;

3) Background Attachment

i) background-attachment : Scroll; background image scroll.
fixed; background image is fixed.

4) Background Shorthand

This property is used to specify a background property in single line.

i) background: rgb (R, G, B)
url (" . ")
repeat
Scroll
50% 50%;

Any property of shorthand sequence can be missing but sequence should be same.

a) color
b) image
c) repeat
d) attachment
e) position

opacity property can be used with images, text, etc. so that image transperancy can be controlled. It is also set with the help of background-color property with opacity attribute(α)

5. Background Shadow.

i) text-Shadow : 2px 2px 2px rgba(R,G,B,α);

- 1st 2px → Horizontal
- 2nd 2px → Vertical
- 3rd 2px → Blur
- rgba(R,G,B,α) → Color

Horizontal and vertical can be negative for upward or left side.

ii) box-Shadow : 2px 2px 2px rgba(R,G,B,α)

used for Shadow of whole box containing element.

CSS Colors

Properties and values

1) property

i) color

ii) background-color

iii) border

iv) opacity

2) value

RGB & RGBA

i) rgb(R,g,b) — R,g,b ranges {0-255}

ii) rgba(R,g,b,α) — α ranges {0-1}

HEXADECIMAL - HEX

000000 to # ffffff — first, second and third "00" are for R, G, B respectively.

HSL - HUE, SATURATION, LIGHT

1) hsl (h, s%, l%)

h - hue - Colorwheel 0 is red, 120 is green, 240 is blue.

Saturation - Degree of grey shade 0% is grey 100% is full color.

light - Degree of light from 0% to 100%

2) hsla (h, s%, l%, α) - α ranges [0-1]

CSS Borders

Border Style

i) border-style : dotted; groove; inset; Solid; dashed; ridge; outset; double;

ii) border-style : dotted (upper) Solid (Right) ridge (bottom) dashed; (left)

iii) border-style: dotted (upper) Solid (left Right) ridge; (bottom)

iv) border-style: dotted (upper bottom) Solid; (Right left.)

Border width

i) border-width : medium;
thick;
xpx;

ii) border-width : xpx ypx zpx medium/thick;
top Right bottom left.

(iii) border-width : xpx ypx;
top bottom

Border color

i) border-color : color; hsl(h, S%, l%);
rgb(R, G, B); rgba(R, G, B, A);
#ffffff;

Border Sides

border-top-style : Styles

border-right-style : Styles

border-bottom-style : Styles

border-left-style : Styles

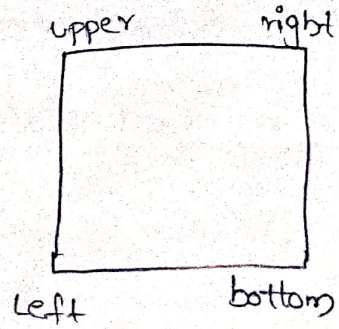
Border Radius

i) border-radius : 15px 15px 15px 15px;
upper Right Bottom left

ii) border-radius : 15px 15px 15px;
upper Right left Bottom

iii) border-radius : 15px 15px;
upper bottom Right left

iv) border-radius : 15px;
All



(v) border - top - left - radius : 2em;

(vi) border - top - Right - radius : 1;

(vii) border - bottom - left - radius : 50%;

(viii) border - bottom -right - radius: 5px;

Border Shorthand property

border : 6px Solid red;

order of values

i) width

ii) Style (compulsory)

iii) color

CSS outlines

Outline is used to making element standout and it outside borders with same spacing.

* It has all the properties same as borders.
* only one different property as follows:

outline - offset : xpx;

CSS Margins / Padding & Height, width

Margins

```
margin : auto;    // browser calculates
         xpx;     // length
         xem;     // length
         x%;      // % width

margin-top    : auto;
margin-bottom : xpx;
margin-right  : xem;
margin-left   : x%;

margin : xpx, ypx, zpx, apx;
         upper  right  bottom  left

margin : xpx, ypx, zpx;
         upper  right left  bottom

margin : xpx, ypx;
         upper bottom  left right

margin : xpx
         All
```

Padding :- Same as that of margin.

Height & width

i) Height : auto; xpx; xem; x%; inherit; initial.

ii) width : auto; xpx; xem; x%; inherit; initial.

auto - default value, browser calculate height width

xpx - length

xem - length

x% - % of box Containg it.

inherit - Sets value to default.

initial - parent's value.

i) min-width : xpx; xem; x%; auto; inherit; initial.

ii) max-width : xpx; xem; x%; auto; inherit; initial.

iii) min-height : xpx; xem; x%; auto; inherit; initial.

iv) max-width : xpx; xem; x%; auto; inherit; initial.

max-min property is used when resizing the concern browsers.

CSS Text

Text have properties like -

Text Color

color : "color"; rgb(R,G,B); rgba(R,G,B,∝); #000000, hsl()

background-color : "color"; rgb(RGB), rgba(R,G,B,A). #000000, hsl()

Text Alignment

```
text-align : justify;          // every line Stretched to same width.
             right; left; center
Vertical-align: top;
                middle;
                bottom;
```

Text decoration

text-decoration : overline; underline; line-through;

text which is not link should not be underline it creates confusion.

Text transform

```
text-transform : uppercase;
                 lowercase;
                 capitalize;
```

Text Spacing

```
text-spacing
letter-spacing
word-spacing
line-height
```

Text Shadow

```
text-shadow : xpx  ypx  zpx  "color";
```

horizontal vertical blur colour

CSS Fonts

Font family

It is used for multiple fonts when first font is not supported then next font is selected as per compatibility of browser.

i) If font name is more than one word then " " commas are used to quote them.

ii) To start with font we want we make it a class name.

Eg:-

```
.serif {
        font-family : "Times New Roman", Times,
        }
```

Font Style

```
font-style : normal;
             italic;
             oblique;
```

font weight

```
font-weight : normal;
              bold;
              900;
```

font Size

```
font-size : xpx;   // xpx
            xem;   // x is relative to parent
            x vw;  // x is default and vw for x.
```

Font Google

The link of google fonts inserted and then font family is used.

```
<link rel = "stylesheet" href = "https://fonts.googleapis.
            com/css? family = Sofia" >
font-family : "Sofia";
```

Font property Shorthand

```
font : italic;
       Small caps;
       bold;
       12px;
       Georgia, Serif;
```

Sequence

font - style
font - variant
font - weight
font - size (Required)
font - family (Required)

CSS links

CSS properties for links are same as we applied text & blocks but otherthan this we need the properties according to activities of mouse.

```
a:link { // unvisited link
    Style for event
}

a:hover { // hover over
    Style for event
}

a:visited { // visited link
    Style for event
    text-decoration: none;
}

a:active { // clicked link
    Style for event
}
```

CSS Lists

Unordered list

```
ul {
    list-style-type : circle;
                      square;
}
list-style-image : url ('name');
```

Ordered list

```
ol {
    list-style-type : upper-roman;
                      lower-roman;
                      upper-alpha;
                      lower-alpha;
```

Style position

```
li {
    list-style-position : outside;
                          inside;
}
```

list have certain margin / paddings.
we can remove bullet points by setting style to nam

```
list-style-type : none;
margin : 0px;
padding : 0px;
```

CSS Tables

we can define borders, background colors, text colours events of mouse etc. to table and tr, th, td etc.

```
table, th, td {
        border-collapse : collapse.
        border: 1px, Solid black;
      }
```

Properties for table, th, td, tr

i) border : 1px, Solid, black;
ii) border-bottom: 1px, ridge, red;
iii) border-collapse: collapse ; no-collapse;
iv) width : xpx;
v) height: xpx;
vi) text-align : left; Right;
vii) padding : xpx;
viii) vertical align : upper; bottom;
ix) color: "color";
x) background-color : "color";

Class Layout - overflow

It decides how content that unfits into box is handled.

i) overflow : visible - overflow not clipped

hidden - overflow hidden

Scroll - overflow handled with Scrollbar Compulsory

auto - Scroll bar only when necessary.

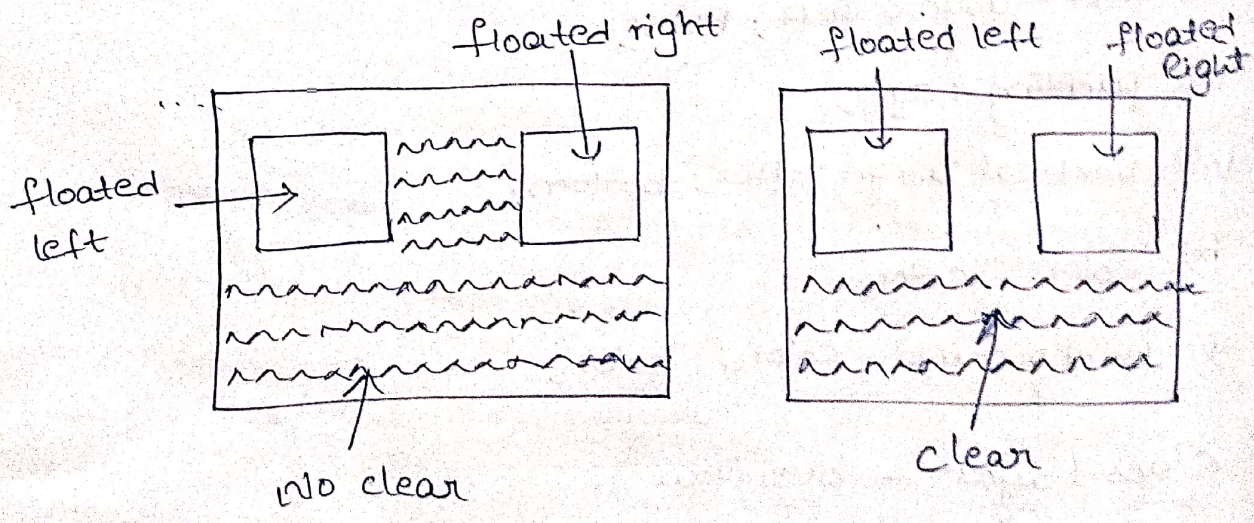
ii) overflow-x

iii) overflow-y // To handle overflow in x & y directions only.

CSS Layout - float & clear

float - It Specify how element should float, i.e, the property for positioning and formatting Content inside of container.

clear - It is used after the float property to decide wheather element should be on next to initially floated element or go below it.



i) float: right;

left;

none;

inherit;

ii) clear: right; // No floating element on right side.
left; // No floating element on left side.
both; // No floating element on either sides.
none; // floating element on either side allowed
inherit; // Inherit;

CSS Display

override elements inline & block property.

Display: inline;
block;
inline-block;

Visibility: hidden; // element affects layout block is hidden.

CSS position.

The position property specifies the type of positio method used for an element.
(Static, relative, fixed, absolute & Sticky)

Conflict Resolution

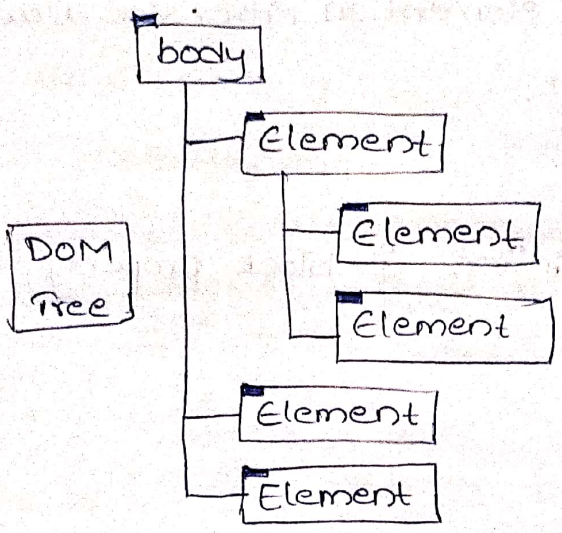
) when there is conflict between two declaratio we follow the rule - "Last Declaration wins" because Html follows top to bottom approach

when external Style declarations have conflict they follow the same rule as per the position of declaration as most of time <link> declared at head (top).

2) When there is no conflict it follows rule –

"Both declaration merged"

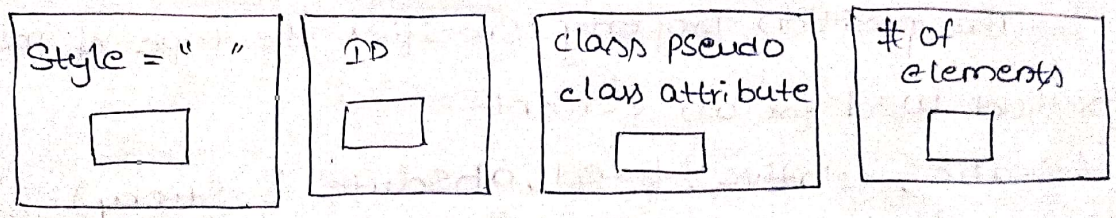
Inheritance



–Any property given to parent element is also inherited by child element

3) Specificity – Most specific selector combination win.

To find the Specificity we use Specificity as




```
div #my para {
    color: blue;
}
```

Style = " "	ID	class	# Element
0	1	0	1

Score = 101

```
div.bigP {
    color: green;
}
```

Style = " "	ID	class	# Element
0	0	1	2

Score = 12

According to Scores paragraph gets blue color.

Overriding the Specificity rules

When we declare the style with `!important` tag it overrides all of the rules of cascading and use the same property declared with `!important` tag.

Eg:- In the above example will show green color irrespective of their specificity score. if we add important tag after property as

```
Color : green!important
```

CSS Text Styling

Font - family

It gives different font choices for user's browser from which browser selected font which it support

Eg:- font - family : Ariel, Helvetica, Sans-Serif;

Color

It allows to set colors either by name & by RGB formats in hexadecimal number followed after # , & `rgba (0,0,0,0)`

a for tranferancy

Eg:- # 0000ff;
Red green Blue

font - Style

It gives options like

i) Normal
ii) Oblique
iii) italic
iv) inherit

Eg:- font - style : italic;

font-weight

It gives boldness by assigning the word value bold or numbers from 100, 200, 300 ..... 900.

Eg: font-weight : 900;

font-Size

The default is `16px` for most of browsers we can set it with `px` declaration & `%` of default

Eg: font-size : 24px;

But font sizes changes according to user make it zoom in & zoom out so keep everything relative to each other. we style font size with relative styling method as follow.

```
body {
    font-size : 120%;
}
```

Now when we want to change the style of font size. we will redeclare without overriding original but relative to it as.

```
<p style = "font-size = 2em"> ..... </p>
```

2em, 3em, 4em or 0.1em, 0.2em gives font size of double, triple ..... respectively to parent font size which we set to `120%` initially.

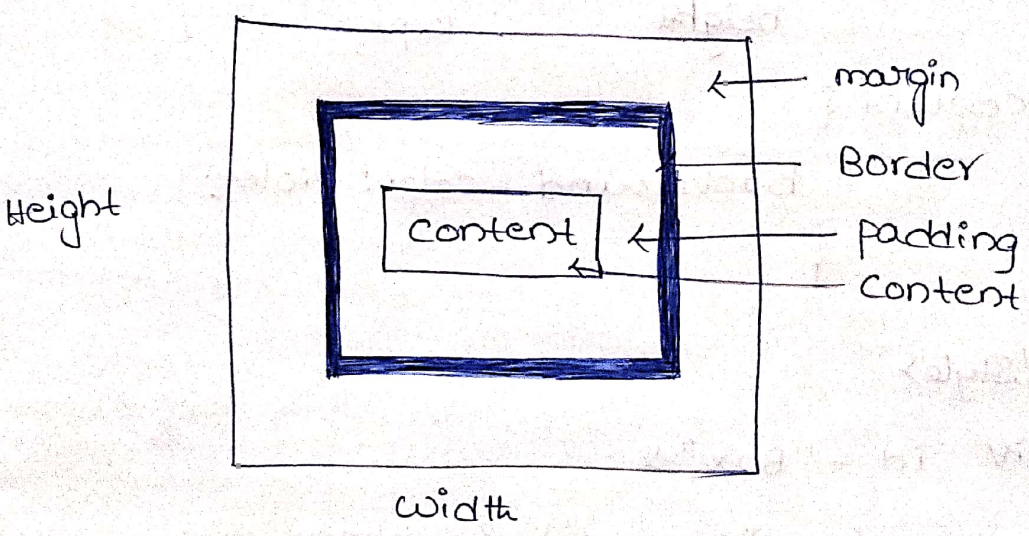
As body is direct parent of `<p>`

Text Effects

```
P{
    text-overflow: clip;
                   ellipsis;
    word-break : keep-all;      // break only at space
                 break-all;     // break at any channe
    word-wrap  : break-word;    // It break word but no
                                   fitted in box
    writing-mode : horizontal-tb; // write horizontal
                   vertical-rl;   // write upside
```

Box Model

The Box is the wrapper that wraps every html elements

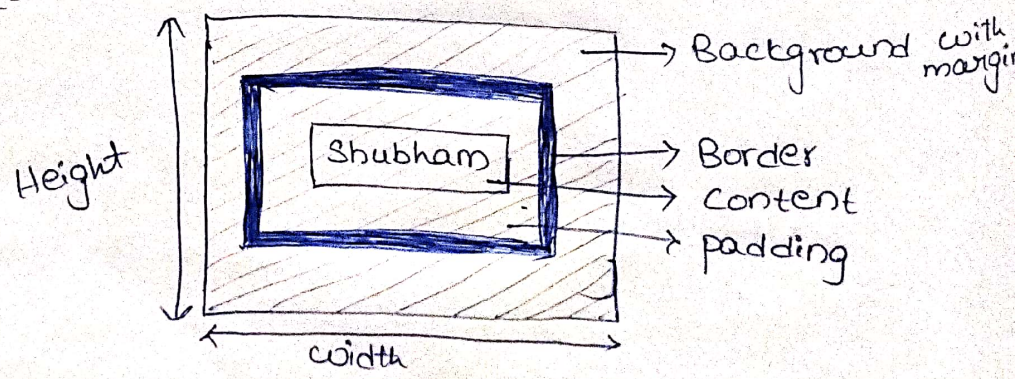


width = width + 2x Border + 2 x padding

height = height + 2x Border + 2 x padding

```
<style>
  body {
        background-color : grey;
        margin : 0;
        padding : 0;
      }

  #box {
        background-color : green;
              padding    : 10px, 10px, 10px, 10px;
              border     : 5px, solid black;
              margin     : 40 px;
              width      : 30px;
              height     : 10 px;

  #content {
              background-color: violet;
            }

</style>

<div id = "Box">
   <div id = "content"> Shubham </div>
</div>
```

Output :-



Width = 30 + 5 + 5 + 10 + 10 = 60 Px

Height = 10 + 5 + 5 + 10 + 10 = 40 Px

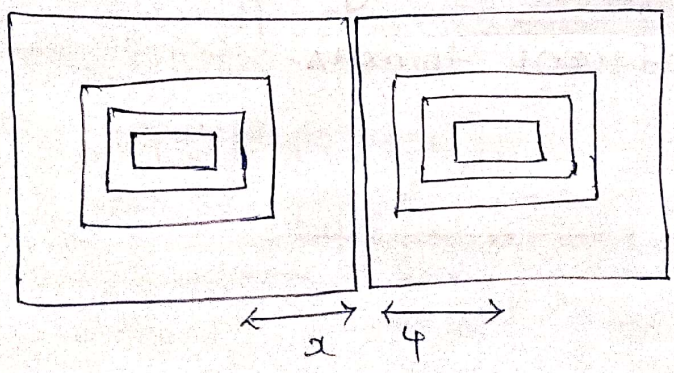
The width is always addition of all element inside element content, borders and pad as box is defined to content but it can be changed using property

```
box-sizing: border-box;
```

Inside that element Styling to parent body & Select for all the element.

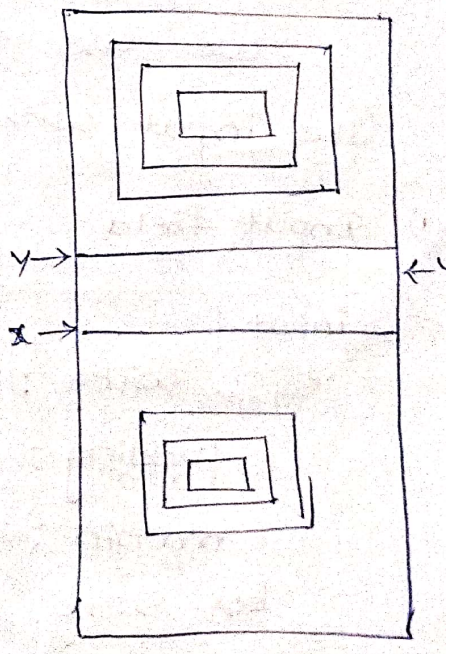
```
{
  box-sizing: border-box;    // for all elements
}
body {
      box-sizing: border-box; // for child elements
    }
  P{
      box-sizing: border-box; // for specific elements
    }
```

Commulative and Collapsing margins



margins on same line get commulated hence

Net margin = x+y



margin on top and belo boxes are merged with biggest margin: y>x

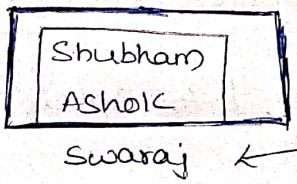
margin = y

Overflow property

overflow

when box size is set to very low with height and width and texts inside are overflowing outside the boxes then we use this command overflow to handle the situation.

overflow : visible - Default it cause text to overflow out.
hidden - It hides the text that don't fit inside.
auto - It provide Scroll bar whenever neccessary
Scroll - It always provides two Scroll bar even when not required.



CSS Forms

we use CSS forms to style the form element like input with different formats.

1) Input field

```
input {
        width : 50%;
      padding : 12px 20px;
      margin : 8px 0;
   box-sizing : border-box;
   border : 2px solid red;
   border-radius : 4px;
   color : black;
   background color: white;
```

2) Input field with Specific type

To Style Specific type of input.

```
input [type = text] {      }
input [type = password {   }
input [type : number {     }
~~input~~ [type ⌀
```

also button, Submit, Reset etc.

3) Focused inputs

To change the style when it gets focused (clicked) we use `Focus` Selector for doing so.

```
input [type =text]: focus {
                    background-color: blue;
                }
```

4) Input with icon/image

If we want icon inside the input use the `background` properties.

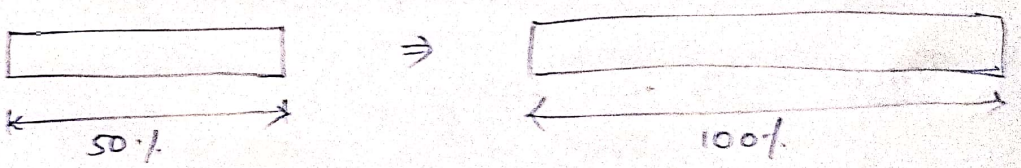
```
input [type =text]{
        background - color : white;
        background - image: url ('searchicon.png');
        background - position: 10px  10px;
        background - repeat : no - repeat;
        padding - left  : 40 px;
    }
```

5) Animated Search input

use transition focused properties to animate input when focused.

```
input[type=text]{
        transition: width 0.45 ease-in-out;
        width : 50%;
        padding: 4px;
        background-color: white;
input[type=text]:focus{
                width : 100%;
            }
```

output



in 0.45

6) Text areas input:-

It is used to resize property to define whether text areas should be resizable & not.

```
text area{
        resize: none;
        width : 100%;
        height : 50%;
    }
```

7) Select menus

```
Select{
        width : 100%;
        height : 20px;
        border : none;
    }
```

CSS Animation

1) CSS @ key frames Rule

@ key frames defines animation from changing one CSS Style to other CSS Style gradually with defined **%. selectors** or **from to** Selectors as follows -

0%. Start
10%
50%
100%. end

From (Same as 0%)
to (Same as 100%)

Syntax

```
@key frames animationname{
                keyframe { CSS-Style}
                }
```

2) animation - name

It is used to apply animation with @keyframe to CSS block in which its specified.

```
animation-name: name of animation;
```

3) animation-duration

Duration in Seconds.

```
animation-duration : 4s;
```

4) animation - delay

Delay to Start animation in Seconds.

```
animation-delay : 2s;
```

5) animation - iteration - count

Number of times it repeats.

```
animation - iteration - count : 3;
```

we can set to infinite for forever run of

6) animation - direction

The direction in which animation to place.

```
animation - direction : normal
                      : reverse
                      : alternate
                      : alternate-reverse
```

7) animation timing function

It defines animation speed curve.

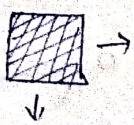
```
animation - timing - function : ease // Start & end slow
                              : linear // Same constant slow
                              : ease-in // slow start
                              : ease-out // slow end
                              : ease-in-out //
                              : cubic-Bezier(n,n,n,n)
                                // defined
```

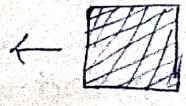
8) animation - fill - mode :- Style of element when an element when the animation is not playing.

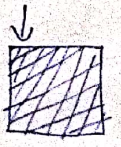
```
animation - fill - mode : none
                        : Forwards
                        : Backwards
                        : both
```

```
div{
    width : 100 px;
    height : 100 px;
background-color: red;
    position : relative;
animation-name: example;
animation- duration: 4S;
animation- delay : 2s
animation - iteration-count : 3;
animation - timing-function: ease;
animation-fill-mode: forwards;
}
@ keyframes example {
  0% {background-color: red ; left : 0px; top : 0px;}
 50% {background- color: blue; left : 200 px; top : 0 px;}
 100% {background-color: green; left: 0px; top : 0px;}
}
```

output

1) 0% 

2) 50% 

3) 100% 

3 times

CSS Navigator bar

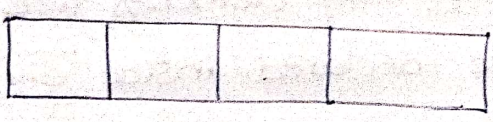
Navigation bar is a kind of list of Url's. It has base of html list elements and there are two kinds of Navigation bars.

1) Vertical Navigation bar

vertical navigation bar

```
ul {
    list-style-type: none; // order style of list
    width: 50%;    // width of navigation bar
    background-color: #f1f1f1; // background color.
    height: 50%;   // Height of navigation bar
    position: fixed;
    overflow: auto;
  }

  li a {
      display: block; // to make block elements
      Color: #00000; // color of text of tab
    }

  li {
    text-align: center; // text aligns in tab
    }

  li a:hover {
          background-color: #555;
          Color: white;
        }

  li a.active {
          background-color: #1CAE50;
          Color: white;
        }
```

2) Horizontal Navigation bar



use `display: block` to make whole area clickable

`Float: left` to float blocks

`position: fixed` to fix navigation bar on page.

```
li {
    float: left;
    position: fixed;
}

li a {
    display: block;
    color : white;
    text-align: center;
    padding : 14px 16 px;
text- decoration: none;
}

li a:hover {
        background-color: #111;
    }

a.active {
        background-color: 4CAF50;
    }
```

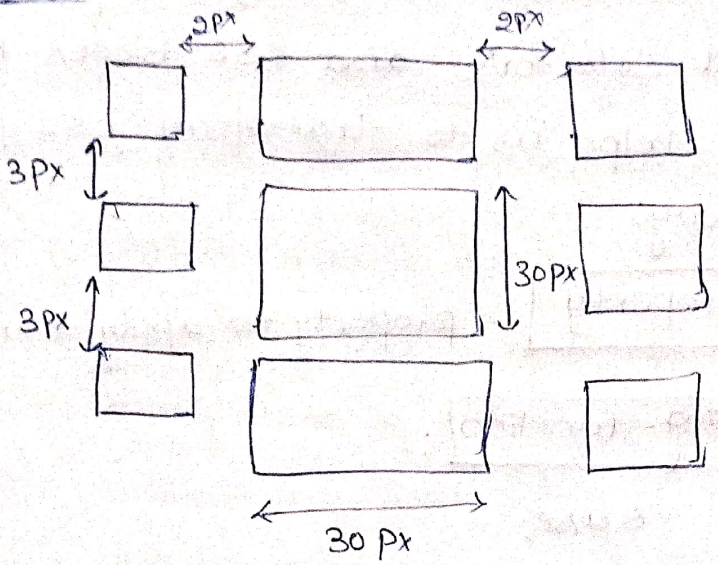
To make tabs divider in navigation bar use box property border-right, border bottom etc.

CSS Grid

CSS grid used for defining elements like keypad. when `Div` elements are declared inside `div` element then outer `div` is grid - container and inside `div` are grid elements/items.

```
.grid-container{

    display: grid;
    grid-colum-gap: 2px;
    grid-row-gap: 3px;
    grid-gap: 2px 3px;
    grid-template-column: auto 30px auto;
    grid-template-rows: auto 30px auto;

// No.of elements defined defines column, row numbers auto is undefined size and size can be given. here 3x3 created.

    {
        background color: green;
        text-align: center;
        font-size: 20px;
    }

<div class = "grid-container
    <div> 1 </div>
    <div> 2 </div>
    <div> 3 </div>
    <div> 4 </div>
    <div> 5 </div>
    <div> 6 </div>
    <div> 7 </div>
    <div> 8 </div>
    <div> 9 </div>
```

Output



CSS Transitions

CSS transition allows you to change property value smoothly over given time duration.

We define transitions with two selector blocks CSS one have property `transition` and one have pseudo-selector and properties to change as mentioned in `transition-name`

```
div {
    width: 100 px;
    height: 100 px;
background-color: red;
transition-property: width height;
transition-delay: 2s;
transition-duration: 4s;
transition-timing-function: ease;
}
div:hover {
        width: 200 px;
        height: 200 px;
```

-As here hover over div may change the css property as per selectors and css blocks but transition only helps us to transform css properties with defined ways.

transition - property - property to apply transitions.

transition - timing - function :-

ease;
ease-in;
ease-out;
ease-in-out;
linear;
cubic-bezier (n n m)

CSS Units

Absolute lengths

cm - centimeter
mm - millimeter
in - inch
Px - pexels
Pt - points
Pc - picas

Eg:- 20px, 20cm, 20mm

pixels are relative to viewing device Screen for low-dpi-device. one pixel is 1 device pixels for high-dpi-device. one pixel is multiple pixels.

Absolute lengths are not recommended to use.

Relative lengths

It is to specify length relative to another length property. It is recommended method.

em - (2em means 2 times current font)

ex - relative to x height of current font.

ch - relative to 0 width (zero's width).

rem - relative to font size of root element.

vw - relative to 1% of width of viewport.

vh - relative to 1% of height of viewport.

vmin - relative to 1% of viewports smaller dimension

vmax - relative to 1% of viewports larger dimension

% - relative to parent element.