CSS - Cascading Style Sheets

* It is a language used to Style the Html Documents.

Syntax:

```
h1 {
    Colour : blue ;
    font-size : 12px ;
}
```

Here h1 — Selector
{ } — Declaration
font-size, Colour — Property
12px, blue — Value

5 Types of Selectors

- Simple Selectors
  - Name, Id, Class
- Combinator Selectors
  - Descendent Selector (Space)
  - Child Selector (>)
  - Adjacent Sibling Selector (+)
  - General Sibling Selector (~)
- Pseudo - Class Selectors
  - :
- Pseudo - Element Selectors
  - ::
- Attribute Selectors
  - a[attribute]
  - a[attribute = "Value"]
  - a[attribute ~= "Value"]
  - a[attribute |= "Value"]
  - a[attribute ^= "Value"]
  - a[attribute $= "Value"]
  - a[attribute *= "Value"]

21/09/2023

How to Add CSS: 3 Ways

External CSS | Internal CSS | Inline CSS

---

① **External CSS:** Create seperate css file with `.css` extension

if External Stylesheet name is "mystyle.css"

Link it using link tag in Head Section.

```
<link rel = "stylesheet" href = "mystyle.css" >
```

---

② **Internal CSS:** Use style tag inside the Head Tag.

```
<head>
<style>
    - - - - - -
    - - - - - -
    - - - - - -
    - - - - - -
</style>
</head>
```

---

③ **Inline CSS:** Used to Style a Single Element inside Tag.

```
<Tag name  style = "property: value ; property: value;" > ---- </Tagname
```

Cascading Order:

① Inline CSS

② External & Internal CSS (Depends on Order Last Defind will be Overrid

③ Browser Default

Comments:

```
/* This is Single line Comment */

/* This is
   Multiline
   Comment */
```

CSS Colours :

RGB | HEX | HSL | RGBA | HSLA

rgb (red, green, blue)
rgb (0, 0, 0)
rgb (0, 255, 255, 255)

rgba (red, green, blue, alpha)
rgba (0, 0, 0, 0)
rgba (255, 255, 255, 1)

Hex Value : #rrggbb
#000000 to #ffffff
#ffcc99 = #fc9

hsl (hue, Saturation, lightness)
hsl (0, 0%, 0%)
hsl (359, 100%, 100%)

hsla (hue, Saturation, lightness, alpha)
hsla (0, 0%, 0%, 0)
hsla (359, 100%, 100%, 1)

CSS Backgrounds :

1. Background-color
2. Background-image
3. Background-repeat
4. Background-attachment
5. Background-position
6. Background (shorthand property)

① { background-color : AnyColor ; }

{ background-color : rgba (-,-,-,-) } If you need Opacity with Heading clarity use this.

② { background-image : url ("Image.format") ; }

③ { background-repeat : no-repeat ; (only once)
repeat-x ; (repeats Horizantally)
repeat-y : (repeats Vertically)
space ; (repeats with Spaces)
round ; (No gaps)

CSS Colours :

RGB | HEX | HSL | RGBA | HSLA

rgb (red, green, blue)
rgb (0, 0, 0)
rgb (0, 255, 255, 255)

rgba (red, green, blue, alpha)
rgba (0, 0, 0, 0)
rgba (255, 255, 255, 1)

Hex Value : #rrggbb
#000000 to #ffffff
#ffcc99 = #fc9

hsl (hue, Saturation, lightness)
hsl (0, 0%, 0%)
hsl (359, 100%, 100%)

hsla (hue, Saturation, lightness, alpha)
hsla (0, 0%, 0%, 0)
hsla (359, 100%, 100%, 1)

CSS Backgrounds :

1. Background-color
2. Background-image
3. Background-repeat
4. Background-attachment
5. Background-position
6. Background (shorthand property)

---

① { background-color : AnyColor ; }

{ background-color : rgba (-, -, -, -) } If you need Opacity with Heading clarity use this.

② { background-image : url ("Image.format") ; }

③ { background-repeat : no-repeat ; (only once)
repeat-x ; (repeats Horizantally)
repeat-y ; (repeats Vertically)
space ; (repeats with Spaces)
round ; (No gaps)

④ background-attachment : fixed; (will not scroll with page)
Scroll; (will scroll with page)
local; (will scroll with element's content)

---

⑤ background-position : left top;
left center;
left bottom;
right top;
right center;
right bottom;
center top;
center center;
center bottom;
x% y%; (Top left is 0% 0%
Right bottom is 100% 100%)
xpos ypos;

---

⑥ background-size : auto; (original size)
length; (width, height) in px
percentage; (width, height) in %
cover; (covers entire container)
contain; (Image is fully visible)

---

⑦ background-origin : padding-box; (background image starts from padding box)
border-box; (background image starts from border box)
content-box; (background image starts from content box)

---

⑧ background-clip : padding-box; (BG color spreads upto padding box)
border-box; (BG color spreads upto border box)
content-box; (BG color spreads upto content box)

---

⑨ background-blend-mode : normal; (normal)
multiply; (multiples BG color to Image color)
screen; (Looks like projection on BG screen)
overlay; (Looks like projection mixes with BG screen)
darken; (Img will be darken)
lighten; (Img will be brightened)
color-dodge; (Looks like film negative)
saturation; (shows color contrast)
color; (shows original colors)
Luminosity; (Looks like Black & white related to BG screen)

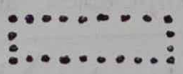
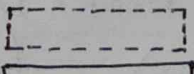
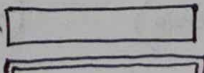
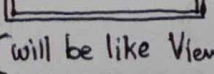
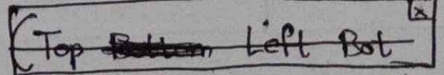
Background Shorthand — different BG properties in one Declaration

① Background-color (Specifies the BG colour & opacity with RGB)
② background-image (Specifies the BG image of mentioned URL)
③ background-position (In the given screen it indicates where the Img need to be placed)
④ background-size (Specifies the size of img)
⑤ background-repeat (Image by default repeats by this property Repetition managed)
⑥ background-origin (Specifies positioning area)
⑦ background-clip (Specifies painting area of BG img (or) colour)
⑧ background-attachment (Specifies BG img should be Fixed (or) Scrolled)
⑨ Initial (Sets this property to Default Value)
⑩ Inherit (Inherits this property from its parent element)

CSS Borders

CSS Borders mainly have 4 properties.

① Border-style
② Border-width
③ Border-color
④ Border-radius

① { border-style : dotted ; 
dashed ; 
solid ; 
double ; 
groove ; (will be like View of Paper boat from Top Upper side)
ridge ; (will be like View of Paper boat from Bottom lowside / like sand pond view)
inset ; (Theatre screen)
outset ; (Pyramid Top)
none ;
hidden ;
dotted dashed solid double ; 
(Top Right Bottom Left)

② border-width : thick;
medium;
thin;
5px;
10px;
100px; (All sides 100px)
5px 3px; (Top-bottom : 5px, Left-right : 3px)
5px 3px 2px; (Top: 5px, sides: 3px, bottom: 2px)
5px 3px 7px 10px; (Top: 5px, right : 3px, Bottom : 7px, Left : 10px)

---

③ border-color : red; (name)
#ff0000; (HEX value)
rgb(255,0,0); (rgb value)
hsl(0,100%, 50%) (hsl value)
red blue green yellow; (Top-red, right - blue, bottom-green, left-yellow)

---

④ border-radius : 5px;

Border Shorthand — Border properties in One Declaration

① border-width (Size/thickness of Border)
② border-style (Kind/type of Border)
③ border-color (Color of Border)

} border : 5px Solid Red; — Shorthand property

Combinations

border-top-width	border-bottom-width	border-left-width	border-right-width
border-top-style	border-bottom-style	border-left-style	border-right-style
border-top-color	border-bottom-color	border-left-color	border-right-color
border-top	border-bottom	border-left	border-right

④ border-radius (Results in Rounded Borders)
⑤ border-collapse: collapse;

CSS Margins

Margins create space around the Borders for an Element.

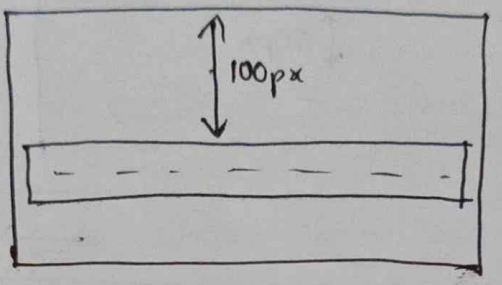
1. Margin-top
2. Margin-right
3. margin-bottom
4. margin-left

All the margin properties can have values like:

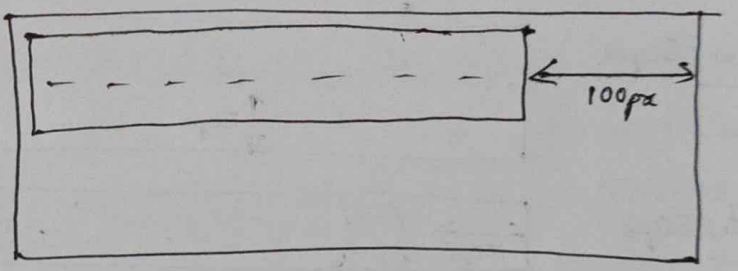
- Auto — (Browser calculates the margin)
- length — (Specifies a margin in px, pt, cm, etc)
- % — (Specifies a margin in (%) of width)
- Inherit — (Inherits from parent element)

---

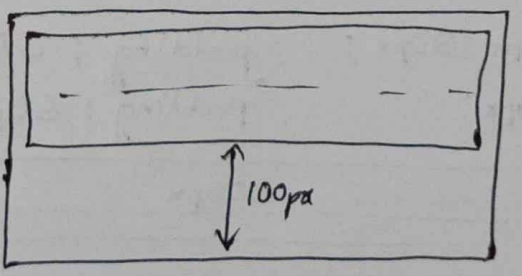
① Margin-top : 100px;



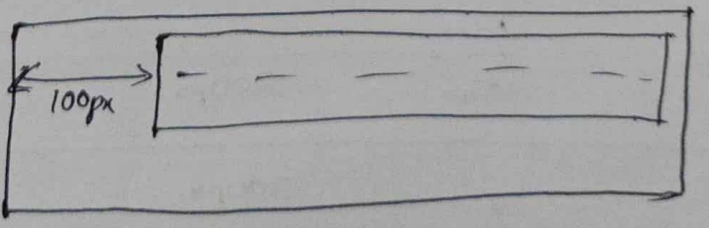
② margin-right : 100px;



③ margin-bottom: 100px;



④ margin-left : 100px



⑤ margin : 100px, 50px, 75px, 40px (Top, right, bottom, left)
margin : 100px, 50px, 40px (Top:-100px, Sides :-50px , bottom :-40px)
margin : 100px, 40px (Top & bottom :- 100px, Left & right :- 40px)
margin : 25px (Top: 25px, bottom : 25px, left: 25px, right : 25px)

⑥ margin collapse : margin : 0 0 25px 0; margin : 100px 0 0 0
margin : 0 0 0 50px; margin : 0 25px 0 0

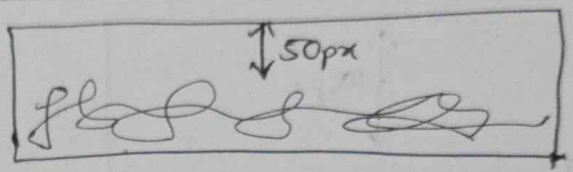
CSS Padding

Padding generates space around Content and below border
Simply between content and Borders.

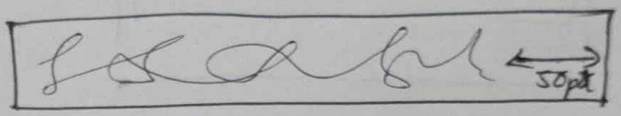
① padding-top
② padding-Right
③ padding-bottom
④ padding-left

All padding properties will have Values like:
length - (specifies a padding in px, pt, cm)
% - (specifies a padding in % of width in Element)
Inherit - (Inherit from parent element)

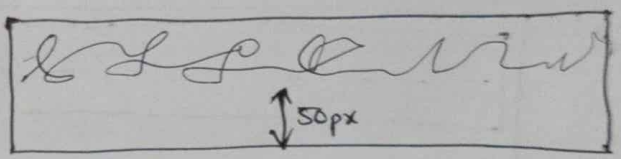
① Padding-top: 50px;



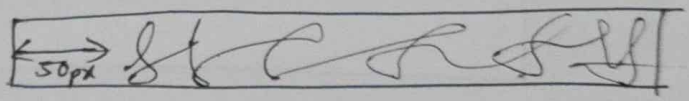
② padding-right: 50px;



③ padding-bottom: 50px



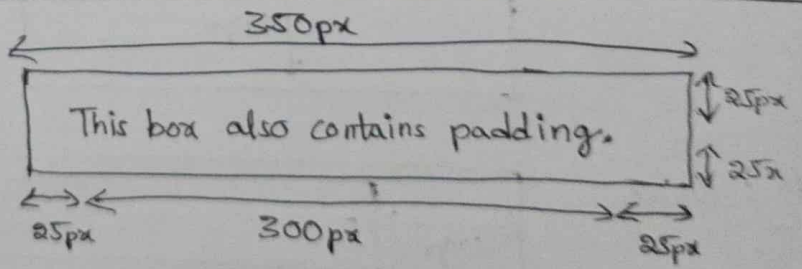
④ padding-right: 50px



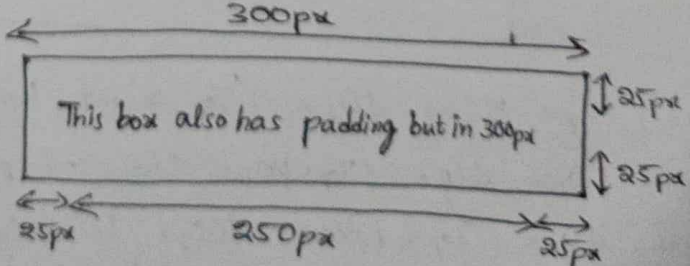
⑤ padding: 50px 25px 40px 30px;
padding: 50px 25px 40px;

padding: 50px 25px;
padding: 50px;

```
div {
  width: 300px;
  padding: 25px;
}
```




```
div {
  width: 300px;
  padding: 25px;
  box-sizing: border-box;
}
```



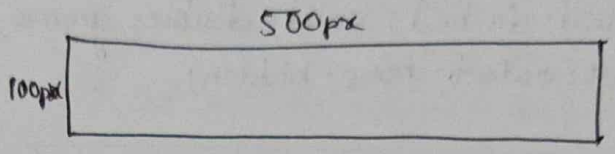
Note: If you need padding with in the Width mentioned then use `box-sizing: border-box;` or else give extra space for padding.

CSS Height & Width

CSS height & Width properties are used to set Height & width of elements

① height
② width
} properties — values {
- auto — (Default, Browser calculates Height & width)
- length — (defines the height/width in px, cm, etc)
- % — (defines height/width in % of Block)
- initial — (Default value)
- inherit — (Inherited from parent value)

```
div{
    height : 100px;
    width : 500px;
}
```



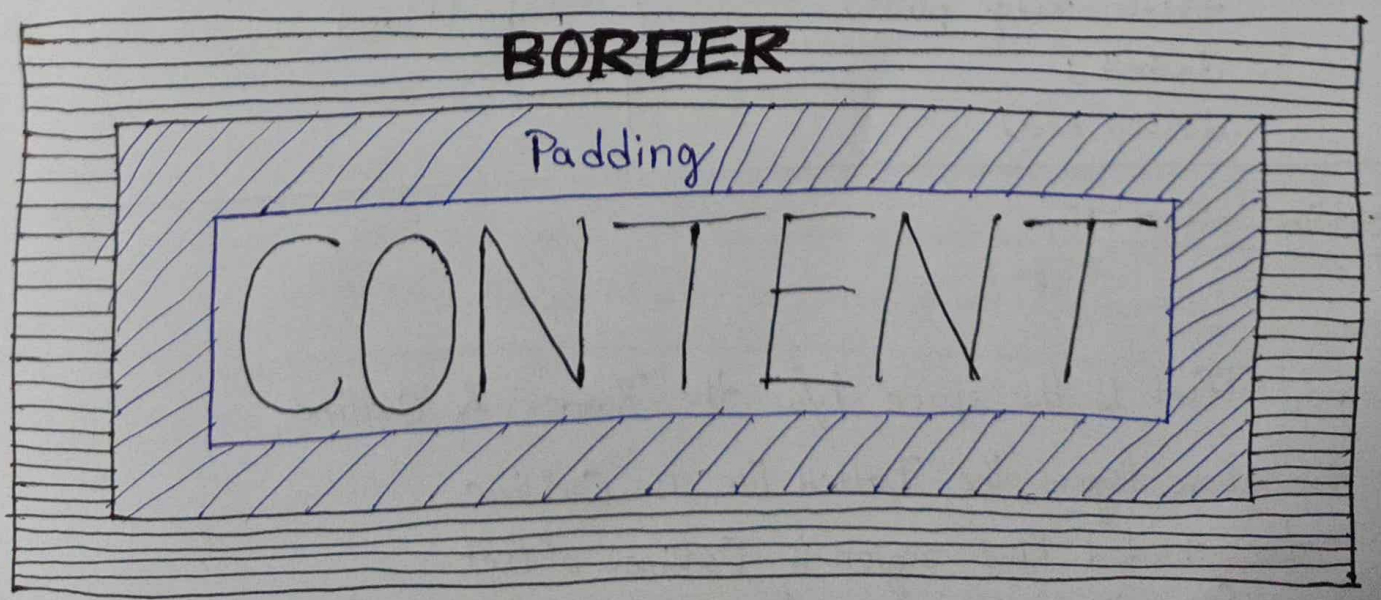
If we resize the browser if the width is less than 500px the content will get missed and horizontal scrollbar gets added.

In such Cases use Max-Min properties to Height & width

① max-height
② max-width
③ min-height
④ min-width

} with this properties even thought the browser gets resized the Content inside the width & Height will get Adjusted to view according to the page size Dimensions.

CSS Box-Model

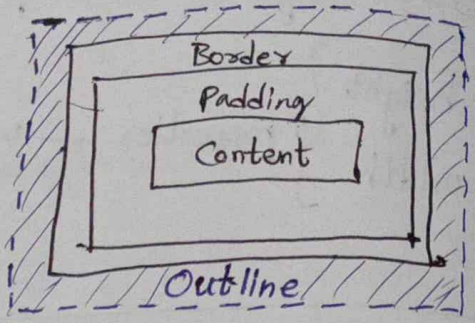


CSS Outline — Almost Similar to CSS Borders.

An Outline is a line drawn outside the element's Border

Outline is not a part of Element's Dimensions

The Elements Total Width & Height is not affected by the width of the outline,



---

① outline-style : dotted; dashed; solid; double; groove; ridge; inset; outset; none; hidden;

---

② outline-width : thin ; (typically 1 px)
medium ; (typically 3 px)
thick; (typically 5 px)
A Specific Size (in px, pt, cm, em, etc)

---

③ outline-color : name ; (red, blue, green, etc)
HEX ; (#ff0000)
RGB ; (rgb(255, 0, 0))
HSL ; (hsl(0, 100%, 50%))
invert ; (performs color inversion, which ensures that outline is visible, regardless of the color background)

---

④ outline : 5px ~~yellow~~ solid yellow;
thick ridge pink;
dashed ;
dotted red;
} Short Hand

---

⑤ outline-offset: 15px;
25 px;

☆ Outline-Offset is the space b/w the Border & Outline

In the above figure the Dotted line is Outline
the shaded Blue region is Outline-offset
(It is like padding space b/w the Border & outline)

CSS Text — So many properties for formatting Text

① Color : name ; (red)
#ff0000 ; (Hex values)
rgb(255,0,0); (rgb values)
hsl (0, 100%, 50%); (hsl values)

② text-align : center; (Text will be aligned to Center)
left; (Text will be aligned to left)
right; (Text will be aligned to right)
justify; (~~last it~~ Stretches the lines to equal width)

text-align-last : center; (last line will be aligned to center)
left; (last line will be left aligned)
right; (last line will be right aligned)
justify; (last line will be stretched to line width length)

text-direction :-

```
p {
    direction: rtl;
    unicode-bidi: bidi-override;
}
```

(Writes the paragraph from right to left
Entire line will be written rtol)

Vertical-align : baseline ; (default; Image will be placed on baseline)
text-top; (Image will be placed on text top □)
text-bottom; (Image will be placed at text-bottom □)
sub; (Image □ will be placed sub alignment)
sup; (Image □ will be placed super alignment)

③ text-decoration:-

text-decoration-line : overline ; (overline)
line-through ; (~~Line-through~~)
underline ; (underline)
Overline underline ; (overline underline)

text-decoration-color : name ; (red, blue, green, yellow)
#ff0000 ; (Hex values)
rgb(0,0,0) ; (rgb values)
hsl (0, 100%, 50%) ; (hsl values)

text-decoration-style : solid ; (——)
double ; (═══)
dotted ; (.....)
dashed ; (- - - -)
wavy ; (~~~~)

text-decoration-thickness : auto ; (Default)
5px ; (specified size)
25% ; (specified percentage)

text-decoration : none ; (removes underline which was default)
: underline ;
: underline red ;
: underline red solid ;
: underline red solid 5px ;

④ Text-Transformation :-

text-transform : uppercase ; (TURNED TO UPPERCASE)
lowercase ; (turned to lowercase)
capitalize ; (Turned To Capitalized Text)

---

⑤ Text Spacing :

(i) text-indent : 50px ; (first line of Text is started after 50px indentation)

(ii) letter-spacing : 5px ; (L e t t e r S p a c i n g)
-2px ; (Letter Spacing)

(iii) word-spacing : 10px ; (Its just a paragraph)
-2px ; (Itsjustaparagraph)

(iv) Line-height : 0.8 ; (line1 line2)
1.8 ; (line1 line2)
1.1 ; (Default, (Space b/w lines) to 1.2)

(v) white-space : nowrap ; (Text wrapping is disabled. Everything will be in single line)

---

⑥ Text Shadow : adds shadow to Text 2px (horizantal shadow) 2px (vertical shadow)

(i) text-shadow : 2px 2px ; (Horizantal shadow, vertical shadow)
2px 2px red ; (Horizantal shadow, vertical shadow, color of shadow)
2px 2px 5px blue ; (H-shadow, V-shadow, blureffect, color)
2px 2px 3px #0000ff ;
0 0 5px #ff0000 ;
0 0 3px #ff0000, 0 0 5px #00ff00 ;
0 0 2px #ff0000, 00 3px #00ff00, 0 0 5px #0000ff ;

CSS Fonts

★ If the font name is more than 1 word use " "

5 Generic Font Families

① serif — (fonts have small stroke at edges of each letter)
② Sans-serif — (fonts have clean lines (no small strokes))
③ Monospace — (fonts have same fixed width (Mechanical look))
④ Cursive — (fonts like Human handwriting)
⑤ Fantasy — (Decorative / Play Full fonts)

All the different font names belong to one of the generic families

F — sans-serif F — serif

---

① font-family: "Times new Roman", Times, serif ; (more than one word, so " ")
Arial, Helvetica, sans-serif;
"Lucida Console", "Courier New", monospace;

⇒ In case the first font is not found by browser the 2nd & 3rd will be used as backups this is fallback systems.

Web Safe fonts for Html & CSS:

(i) Arial (sans-serif)
(ii) Verdana (sans-serif)
(iii) Tahoma (sans-serif)
(iv) Trebuchet MS (sans-serif)
(v) Times New Roman (serif)
(vi) Georgia (serif)
(vii) Garamond (serif)
(viii) Courier New (monospace)
(ix) Brush Script MT (cursive)

Font Fall Backs:

"Times New Roman", Times, serif
Georgia, serif
Garamond, serif
Arial, Helvetica, sans-serif
Tahoma, Verdana, sans-serif
"Trebuchet MS", Helvetica, sans-serif
Geneva, verdana, sans-serif
"Courier New", Courier, monospace
"Brush Script MT", cursive
Copperplate, Papyrus, fantasy

② font-style : normal;
italic;
oblique;

③ font-weight : normal;
lighter;
bold;
900;

④ font-variant : normal; (Its a normal Font)
small-caps; (Its A Normal Font But Small-Caps)

⑤ font-size : 30px ; (in pixels)

font-size : 1.875em ; (Allows users to resize the Text in browser menu
1em (Default). (1em = 16px) (Pixels/16 = em))

font-size : 100% ; (in %)

Viewport width (vw) : the Text size will follow the size of the Browser window

~~<h1 style = "font-styl" "font-size" : 10vw"~~

<h1 style = "font-size : 10vw" > Helloworld </h1>

(Viewport is the Browser window size. 1vw = 1% of viewport width.)
(If the viewport is 50cm wide, 1vw = 0.5cm)

⑥ Google Fonts : add a special Style sheet in <head> section

< link rel = "stylesheet" href = "https://fonts.googleapis.com/css?family= name">

```
<head>
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Sofia">
<style>
body{ font-family: "Sofia", sans-serif; }
</style>
</head>
```

☆ For single Font selection

```
<head>
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Sofia|Audiowide|Trirong">
<style>
body{ font-family: "Audiowide", sans-serif;}
h1.a{ font-family: "Sofia", sans-serif;}
</style>
h1.b{ font-family: "Trirong", serif}
</style>
</head>
```

☆ For more than one fonts use "(|)" pipe character

☆ Enabling Font Effects (use &effect = effectname in link and mention class = "font-effect-effectname" in body)

```
<head>
<link rel="stylesheet"
      href="https://fonts.googleapis.com/css?family="Sofia"&effect=fire">
<style>
body{ font-family: "Sofia", sans-serif; }
</style>
</head>
<body>
<h1 class="font-effect-fire"> Sofia on Fire </h1>
</body>
```

☆ For Multiple effects use pipe character "(|)" in link and declare more classes in body for the effect enabling.

∴ <---- family = "font name" > — for using a font-family type

∴ <---- family = "fontname1 | fontname2 | fontname3" > — for Multiple types

∴ <---- family = "fontname & effect = effectname" > — for font-family & effect

- Dont forget to Declare class with class = font-effect-effectname

∴ <---- family = fn1 | fn2 | fn3 & effect = en1 | en2 | en3 | en4" > - for multiple font families & effects

---

Font ShortHand :

① font-style
② font-variant
③ font-weight
④ font-size/line-height
⑤ font-family

fontsize & font-family values are required or else Default values are used

font : 20px Arial, sans-serif;

font: italic small-caps bold 12px/30px Georgia, serif;

CSS Icons

① Font awesome Icons

Add script tag to Head section & Insert this

```
<script src="https://kit.fontawesome.com/a076d05399.js"
crossorigin="anonymous"></script>
```

You can extract the Icons using <i> tag in <body>

```
<i class="fas-fa-Iconname"></i>
```

if you want to style that Icon

```
<i class="fas-fa-Iconname" style="font-size: 60px; color: Red;"></i>
```

② Bootstrap Icons

To use Bootstrap Icons add this link inside Head tag

```
<link rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.
```

You can extract the Icons using <i> tag in <body>

```
<i class="glyphicon glyphicon-Iconname"></i>
```

You can style those Icons this way

```
<i class="glyphicon glyphicon-Iconname" style="font-size: 60px; color: Red;"></i
```

③ Google Icons

To use Google Icons add this line inside Head Section

```
<link rel="stylesheet"
      href="https://fonts.googleapis.com/icon?family=Material+Icons">
```

You can extract the Icons using <i> tag in <body>

```
<i class="material-icons">Iconname</i>
```

You can use styled Icons this way

```
<i class="material-icons" style="font-size: 60px; color: Red;">Iconname</i>
```

CSS Links

Links can be styled in Different ways.

There are 4 states of Link:

1. a:link — (a normal, unvisited link)
2. a:visited — (a link that is already visited by user)
3. a:hover — (a link when user mouses over it)
4. a:active — (a link at the moment it is clicked)

While setting a style for several link states, the order Rules are:

⇒ a:hover must come after a:link & a:visited

⇒ a:active must come after a:hover

---

i. color : Name; (Is used to color the link according to States)

ii. text-decoration : None; ((removes Underline from link), also underline, overline)

iii. background-color : name (Highlights the link according to states)

iv. padding : 2px 3px (Spreads the background color according to cordinates)

v. border : 2px solid blue (Gives the border color style width)

vi. text-align : center; (aligns the text to center, right, left or justifies)

vii. display : inline-block; (gives good appearence as button & text inside is set fine)

viii. font-size : 150%; (The link will set to the size mentioned)

ix. font-family : Arial; (The link will be set in given font)

Different types of cursors

Use this format to change cursor

```
<span style = "cursor : value" > value </span>
```

auto — I	e-resize — ↔	nw-resize — ↖	se-resize — ↘	text — I
crosshair — +	move — ✥	pointer — 👆	sw-resize — ↙	wait — ◎
default — ↖	n-resize — ↕	progress — ↖◎	w-resize — ↔	
help — ↖?	ne-resize — ↗	s-resize — ↕		

CSS Lists

 – unordered lists
 – ordered lists

---

① list-style-type : circle; (○)
square; (■)
upper-roman; (I)
lower-alpha; (a)
disc; (•)
armenian; (ա, բ, գ)
decimal; (1)
decimal-leading-zero; (01)
lower-alpha; (a)
lower-greek; (α, β, γ)
lower-latin; (a, b, c)
lower-roman; (i, ii, iii)
upper-alpha; (A, B, C)
upper-greek; (1, 2, 3)
upper-latin; (A, B, C)
upper-roman; (I, II, III)
cjk-ideographic; (一, 二, 三)
georgian; (ა, ბ, გ)
hebrew; (א, ב, ג)
hiragana; (あ, い, う)
hiragana-iroha (い, ろ, は)
katakana; (ア, イ, ウ)
katakana-iroha; (イ, ロ, ハ)
none;

---

② list-style-image : url ('sqpurple.gif'); (■ square1 ■ square2)
none;
initial;
inherit;

③ list-style-position : outside ; (default) • This is List

inside ; • This is List

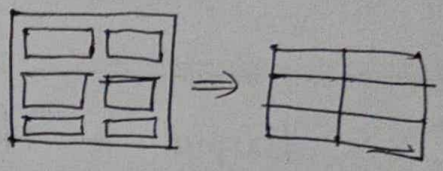
④ list-style-: square inside url("sqpurple.gif")

CSS Tables

table {

border-collapse : collapse ;
separate ; (default)

}

Will convert 

★ Table Borders are very useful for different types of Tables

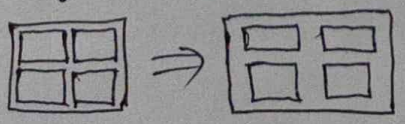
table Size — width : 100%; 50%; etc
height : 70px ; } Sets the Width & Height of cells in a row

table alignment — text-align : center/right/left ; (Horizantal Alignment)
vertical-align : top/bottom/center ; (Vertical Alignment)

table Style — padding : 15px ;
text-align : left ;
border-bottom : 1px solid black ;
tr : hover { background-color : orange ; } (Hoverable Table)
tr : nth-child (even) { background-color : #f2f2f2 } (Striped Table)
th { background-color : green ; color : white ; } (Header color)

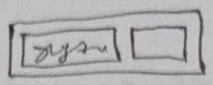
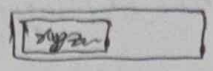
table Responsive — overflow-x : auto ; (creates a Horizantal Scroll bar if screen width is less than Table width)

border-spacing : 15px 50px ; (Horizantal, Vertical)



Caption-side : top; (caption above table)
bottom; (caption below table)

---

empty-cells : show; 
hide; 

---

table-layout : auto; (adjusts accordingly)
fixed; (fixed size based on heading row)

CSS Layout - The Display property

Block level Elements :

⇒ A Block level Elements always starts in a New line & takes full space width available

Egr
<h1> to <h6>
<p>
<form>
<header>
<footer>
<section>

Inline Elements :

⇒ Doesn't start on a New line only takes Necessary width

Egr , <a> , <img>

---

display : none; (as default)

---

visibility : hidden; (Hides the Element but space will be empty)
visible;
collapse; (only for <tr>, <tbody>, <col>, <colgroup>)

display: none ; (Default, completely removed)

inline ; (will set into inline, height & width doesn't work)

block ; (starts in a new line takes whole width)

contents ; (makes container disappear, making child elements children of the element the next level up in DOM)

flex ; (displays element as Block-level flex container)

grid ; (displays element as Block-level grid container)

inline-block ; (inline level Block container, inline element can have height & width)

inline-flex ; (inline level flex container)

inline-grid ; (inline level grid container)

inline-table ; (inline level table)

list-item ; (same as block but it will behave like .List)

run-in ; (Displays element as Block or inline, depending on context)

table ; (<table> element)

table-caption ; (<caption> element)

table-column-group ; (<colgroup> element)

table-header-group ; (<thead> element)

table-footer-group ; (<tfoot> element)

table-row-group ; (<tbody> element)

table-cell ; (<td> element)

table-column ; (<col> element)

table-row ; (<tr> element)

initial ;

inherit ;

Css Max-width: if width of browser window < width of element browser adds  horizontal scroll bar.

So to avoid this use max-width property then It will adjust according to Browsers width.

CSS Layout - The POSITION property

- there are 5 different position values:
  static, relative, fixed, absolute, sticky

- position: static; (default, not affected by top, bottom, left & right properties, positioned according to Normal flow)

 has position: static;

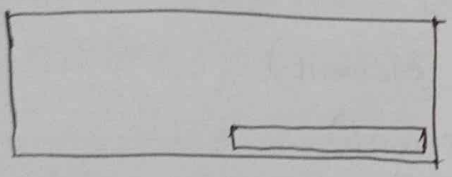
---

position: relative; (relative to normal position, setting top, bottom, right & left properties will cause adjusted away from normal elements, other content can't fill that gap)

 has position: relative;

---

position: fixed; (relative to Viewport, stays in same place even if page is scrolled, T, B, R & L properties are used. doesn't leave a gap at its original place)

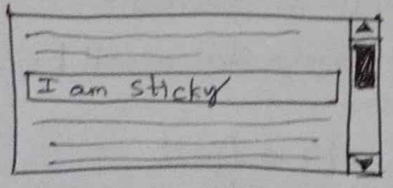


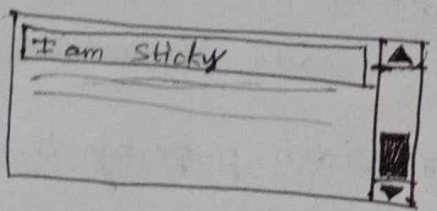
---

position: absolute; (scrolls along with content, positioned relative to nearest positioned ancestor (instead of positioned relative to viewport like fixed)) (content can be overlaped)

---

position: sticky; (based on user's scroll position, sticky element toggles between relative & fixed.)





top : 100px ; 50% ;
bottom : 100px ; 50% ;
left : 100px ; 50% ;
right : 100px ; 50% ;

} Helps in positioning margins to the given elements in relatively in the Divison.

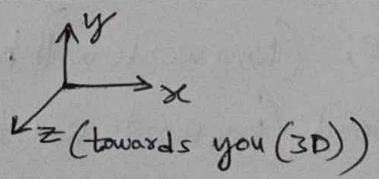
---

clip : rect (top, right, bottom, left) ; (shape)
auto ; (default, no clipping)

---

position : static ;
relative ;
fixed ;
absolute ;
sticky ;
initial ;
inherit ;

CSS Z-Index

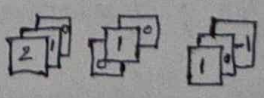


Z-Index property specifies which element should be placed infront of, or behind, the others. Element can have positive (or) negative stack order.

Z-Index only works on positioned elements & Flex elements

Sets the Stack order of an Element.

Z-index : -1 ; (behind the text)
1 ; (on the first element)
2 ; (on the second element)

} 

layer on layer.

CSS layout – Overflow

⇒ Controls what happens to the content that is too big to fit into an area.

Overflow property values:

① visible – (Default, overflow is not clipped, the content renders outside the element's box)

② hidden – (overflow is clipped, rest of the content is invisible)

③ scroll – (overflow is clipped, a scroll bar is added to see the content)

④ auto – (similar to scroll but adds scroll bar only when needed)

⇒ The Overflow property only works for block elements with a specified height

```
overflow-x : hidden;
overflow-y : scroll;
```

```
overflow-wrap : normal; (Default, long words will not Break)
                anywhere; (long words will break if they overflow container)
                break-word; (long words will break if they overflow container)
```

CSS layout - Float & Clear

⇒ Float property is used for positioning & formatting content.

```
float : left; (element floats to the left of its container)
        right; (element floats to the right of its container)
        none; (element doesnot float, displayed at its origin, Default)
        inherit;
```

⇒ simply float property can be used to wrap text around images

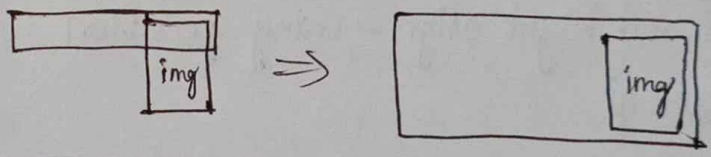
⇒ Clear property specifies what should happen with the element that is next to floating element.

```
clear : none; (Default, element is not pushed below left or right floated elements)
        left; (element is pushed below left floated elements)
        right; (element is pushed below right floated elements)
        both; (element is pushed below both left & right floated elements)
        inherit;
```

```
clearfix { overflow: auto; } (This may add scroll bars)

clearfix::after {
    content: " ";
    clear: both;
    display: table;
}
```

(used for most webpages
New, modern clearfix hack)

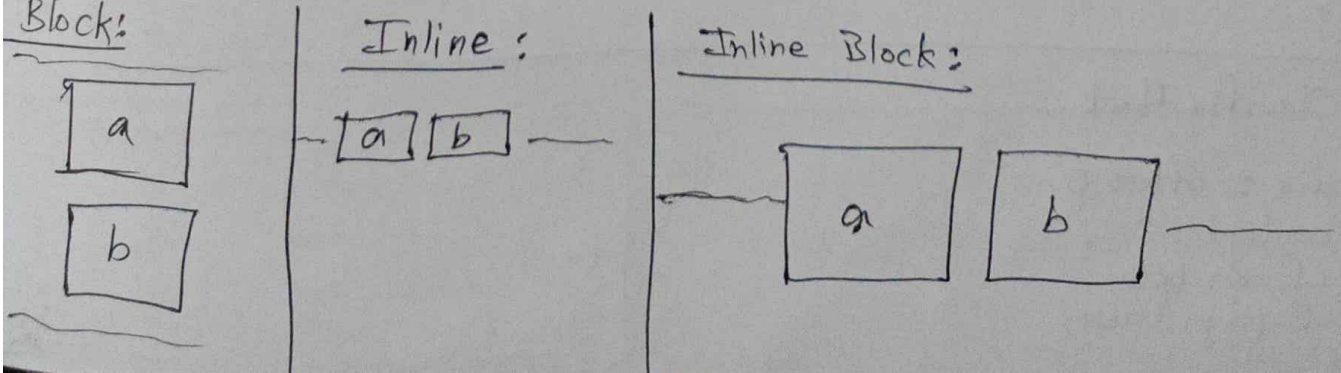


```
box-sizing: border-box;
```

CSS layout - Inline-Block (used to create Navigation Links)

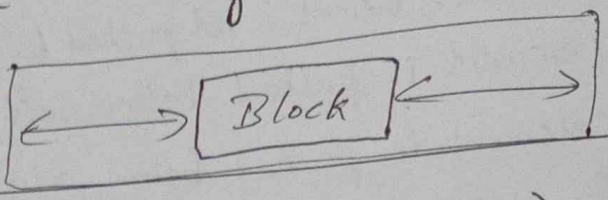
⇒ Compared to inline, the major difference is Inline-block allows to set a width & Height on the Element, Also top & bottom margins/paddings are respected.

⇒ Compared to Block, the major difference is that Inline Block doesn't add a Line break after the Element, so element can sit next to other elements.



CSS layout - Horizantal & Vertical Align

① margin : allto ; (Horizantally centers the Block)



② text-align : center; (centers the text)

③ img {
display: block;
margin-left: auto;
margin-right: auto;
width : 40%;
} (center an Image)

④ Left and Right align - using position

.right {
position : absolute;
right : 0px;
width : 300px ;
}

⑤ Left & Right align - using float

.right {
float : right ;
width : 300px;
}

⑥ The Clearfix Hack

.clearfix :: after {
content : " ";
clear : both;
display : table;
}

⑦ Center Vertically – using padding

```
.center {
    padding: 70px 0px;
    border: 3px solid green;
}
```

To center Horizantally add (text-align: center;) to the stylesheet

---

⑧ center vertically – using line-height

```
.center {
    line-height: 200px;
    height: 200px
    border: 3px solid green;
    text-align: center;
}
.center p {
    line-height: 1.5;
    display: inline-block;
    vertical-align: middle;
}
```

---

⑨ Center vertically – using Position & Transform

```
.center {
    position: relative;
    height: 200px;
    border: 3px solid green;
}
.center p {
    margin: 0;
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
}
```

---

⑩

```
.center {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 200px;
    border: 3px solid green; }
```

(Center vertically – Using Flexbox)

CSS Combinators

A Combinator explains the relationship between the selectors.

A CSS selector can have more than One simple selector.

Between the simple selectors, we can include a combinator.

There are 4 different combinators in CSS:

① descendant selector (space) (div p {——})
② Child selector (>) (div > p {----})
③ adjacent sibling selector (+) (div + p {——})
④ general sibling selector (~) (div ~ p {——})

```
<body>
<h2> css combinators </h2>
<p> Declare random paragraphs </p>
<div>
   <p> paragraph 1 </p>
   <p> paragraph 2 </p>
   <section> <p> paragraph 3 </p> </section>
</div>
   <p> pargraph 4 </p>
   <p> paragraph 5 </p>
<div>
   <p> paragraph 6 </p>
   <section> <p> paragraph 7 </p> </section>
</div>
   <p> paragraph 8 </p>
   <p> paragraph 9 </p>
   <p> paragraph 10 </p>
   <code> Some code </code>
   <p> paragraph 11 </p>
```

div p {—}	div > p {---}	div + p {----}	div ~ p {---}
P1, P2, P3 P6, P7	P1, P2, P6	P4, P8	P4, P5, P8, P9, P10, P11

CSS Pseudo Classes:

⇒ Pseudo class is used to define a special state of an element

Format:

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

Eg:

```
a:link { }
a:visited { }
a:hover { }
a:active { }
div:hover { }
div:hover p { }
```

Hover over the element to show <p> element:

```
p { display: none;
    background-color: yellow;
    padding: 20px;
  }
div:hover p {
    display: block;
  }
```

---

p:first-child { } (matches a specified element that is the first child of another element)

p i:first-child { } (matches first <i> element in all the <p> elements)

p:first-child i { } (matches all <i> elements in <p> elements that are first-childs)

```
q:lang(no) {
    quotes: "~" "~";
}
```

(:lang pseudo class allows you to define special rules for different languages.)

All CSS pseudo Classes:

a:link
a:visited
a:hover
a:active
:not(selector)
:root
:target
input:checked
input:disabled
input:enabled
input:focus
input:in-range
input:invalid
input:optional
input:out-of-range
input:read-only
input:read-write
input:required
input:valid
p:empty
p:first-child
p:first-of-type
p:lang(it)
p:last-child
p:last-of-type
p:nth-child(n)
p:nth-last-child(n)
p:nth-last-of-type(n)
p:nth-of-type(n)
p:only-of-type
p:only-child

CSS Pseudo Elements ::

Used to style specified parts of an Element.

Format: selector :: pseudo-element {
property: value;
}

:: first-line (can only applied to block level elements)

Applicable Properties:
- font properties
- color properties
- bg properties
- word-spacing
- letter-spacing
- text-decoration
- vertical-align
- text-transform
- line-height
- clear

:: first-letter (Only block elements)

Applicable Properties: font, color, bg, margin, border, padding properties
text-decoration, text-transform, vertical-align
line-height, float, clear (Only if float is none)

:: before (used to insert some content before the content of an element)

:: after (used to insert some content after the content of an element)

:: marker (selects the markers of the list items)

:: selection (matches the portion of element that is selected by user)

Applicable Properties: color, bg, cursor, outline

All CSS pseudo Elements:

:: first-line
:: first-letter
:: before
:: after
:: marker
:: selection

CSS Opacity / Transparency

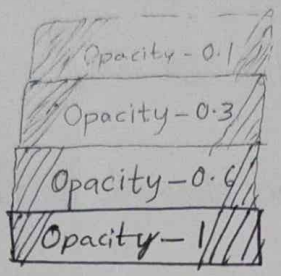
⇒ Opacity property can take value upto (0.0 – 1.0)

Low value – more Transparent

0 – more blurred | 1 – more clear Image

Transparent Box :

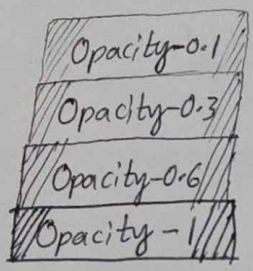
```
opacity : 0.1 ;
opacity : 0.3;
opacity : 0.6;
opacity : 1; (default)
```



Text also gets Transparent

Transparency using RGBA :

```
background : rgba(76,175, 80,0.1)
background : rgba(76, 175, 80, 0.3)
background : rgba(76, 175, 80, 0.6)
background : rgba(76, 175, 80, 1)
```



Here Text is visible with clarity

CSS Navigation Bar (Navigation Bar = List of Links)

create a list using <ul> or <ol> and <li> will have <a> for links

then by css `list-style-type : none;` (Removes bullets or markers of list)

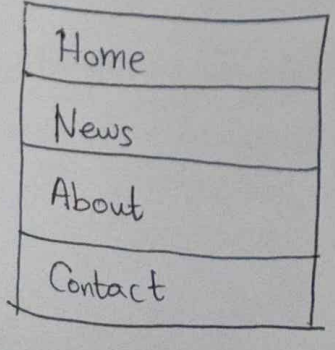
`margin = 0;`

`padding = 0;` (Removes browser default settings)

Vertical NavBar

`display : block;` (links as block elements makes whole link area clickable

`width : 60px;` (Block elements take up full width of ~~<ul>~~ available by default,



CSS Horizontal Navigation Bar

2 ways : Inline (or) Floating list items.

Inline list items : <li> elements as inline

```
li {
  display: inline;
}
```

(By default, <li> elements are Block elements
Here we remove the line breaks before ~~each~~ & after each list item to display them on the one line.

Floating list items : float the <li> elements

```
li {
  float: left;   (to get block elements to float next to each other)
}
a {
display: block;  (allows us to specify padding (height, width, margins))
padding: 8px;
background-color: #dddddd;
}
```

Responsive Sidenav Eg :

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial scale=1.0">
<style>
body {margin: 0;}
ul.sidenav {
    list-style-type: none;  (removes markers of list)
    margin: 0;
    padding: 0;   (Sets to Browser window)
    width: 25%;   (side 25% will be allocated)
    height: 100%;   (It will occupy entire Height)
    position: fixed;   (Unmovable)
    background-color: #f1f1f1;
    overflow: auto;   (If list Items are increased it adds scroll bar)
}
```

```
ul.sidenav li a {
    display: block;
    color: #000;
    padding: 8px 16px;  (Top-bottom, Right-left)
    text-decoration: none; (removes underline of a link)
}

ul.sidenav li a.active {
    background-color: #4CAF50;
    color: white;
}

ul.sidenav li a:hover:not(.active) {
    background-color: #555;
    color: white;
}

div.content {
    margin-left: 25%; (leaves the space of 25% width for navbar)
    padding: 1px 16px;
    height: 1000px;
}

@media screen and (max-width: 900px) {  (If screen size is below 900px)
    ul.sidenav {
        width: 100%;
        height: auto;
        position: relative;
    }
    ul.sidenav li a {
        text-align: center;   float: left;
        float: none;          padding: 15px;
    }
    div.content {margin-left: 0;}
}

@media screen and (max-width: 400px) {  (If screen size is below 400px)
    ul.sidenav li a {
        text-align: center;
        float: none;
    }
}
</style>
</head>
```

CSS Dropdowns

first declare a class = "dropdown"
then declare another class = "dropdown-content"

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333;
}
li {float: left;}
li a, .dropbtn {
    display: inline-block;
    color: white;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
  }
li a:hover, .dropdown:hover .dropbtn {background-color: red;}
li.dropdown {display: inline-block;}

.dropdown-content {
     display: none;
    position: absolute;
    background-color: #f9f9f9;
    min-width: 100px;
    box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
  z-index: 1;
}
.dropdown-content a {color: black;
                     padding: 18px 16px;
                     text-decoration: none;
                     display: block;
                     text-align: left;}
```

```
.dropdown-content a:hover { background-color: #f1f1f1; }
.dropdown:hover .dropdown-content { display: block; }
</style>
</head>
<body>
<ul>
  <li> <a href="#home"> Home </a> </li>
  <li> <a href="#News"> News </a> </li>
  <li class="dropdown">
    <a href="javascript:void(0)" class="dropbtn"> Dropdown </a>
    <div class="dropdown-content">
      <a href="#"> Link 1 </a>
      <a href="#"> Link 2 </a>
      <a href="#"> Link 3 </a>
    </div>
  </li>
</ul>
<h3> Dropdown Menu inside a Navigation Bar </h3>
<p> Hover over the "Dropdown" link to see the dropdown menu </p>
</body>
</html>
```

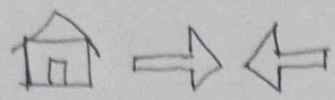
CSS Image Gallery (For Responsive Image Gallery)

```
.responsive { padding: 0 6px;
              float: left;
              width: 24.9999%; }
@media only screen and (max-width: 700px) {
    .responsive { width: 49.9999%;
                  margin: 6px 0; } }
@media only screen and (max-width: 500px) {
    .responsive { width: 100%; } }
.clearfix:after { content: "";
                  display: table;
                  clear: both; }
```

CSS Image Sprites

⇒ Collection of Images put into a single Image

⇒ We can reduce no. of server requests & save bandwidth



(only defines a small transparent img because src attribute cant be empty
The Displayed img will be the background img we specify in css)

width: 46px ; height: 44px; (Defines the portion of image we want to use)

background: url(img_navsprites.gif) 0 0;

(Defines the background img & its position (left 0px, top 0px))

CSS Attribute Selectors

⇒ the [Attribute] selectors is used to select elements with specified Attri

if a[target] is selected, it selects all the <a> elements with target Attribu

⇒ the [Attribute = "Value"] selects elements with both specified Attribute & v

⇒ [Attribute ~= "Value"] selects elements with specified word in value.

Eg: flower, selects sunflower, cauliflower etc. but not flowers, my

⇒ [Attribute |= "Value"] exact value or value followed by hyphen (-)

⇒ [Attribute ^= "Value"] starts with value

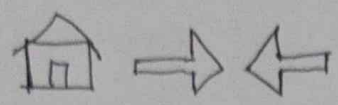
⇒ [Attribute $= "Value"] ends with value

⇒ [Attribute *= "Value"] selects any word which just contains value.

CSS Image Sprites

⇒ Collection of Images put into a single Image

⇒ We can reduce no. of server requests & save bandwidth




```
<img id="home" src="img_trans.gif" />
```

(only defines a small transparent img because src attribute cant be empty
The Displayed img will be the background img we specify in css

```
width: 46px ; height: 44px; (Defines the portion of image we want to use)
background: url(img_navsprites.gif) 0 0;
                (Defines the background img & its position (left 0px, top 0px))
```

CSS Attribute Selectors

⇒ the [Attribute] selectors is used to select elements with specified Attr

if a[target] is selected, it selects all the <a> elements with target Attrib

⇒ the [Attribute = "Value"] selects elements with both specified Attribute &

⇒ [Attribute ~= "Value"] selects elements with specified word in value.

Eg: flower, selects sunflower, cauliflower etc. but not flowers, ny

⇒ [Attribute |= "Value"] exact value or value followed by hyphen (-)

⇒ [Attribute ^= "Value"] starts with value

⇒ [Attribute $= "Value"] ends with value

⇒ [Attribute *= "Value"] selects any word which just contains value.

CSS Forms

input [type=text] only text fields

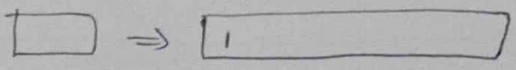
input [type=password] only password fields

input [type=number] only number fields.

```
transition: width 0.8s ease-in-out;

input[type=text]:focus { width: 100%; }
```

little Animation



CSS Counters (to track how many times they are used)

CSS Counters are like variables.

Properties:

counter-reset

counter-increment

content

counter() or counters()

```
<!DOCTYPE html>
<html>
<head>
<style>
body { counter-reset: section; }
h1 { counter-reset: subsection; }
h1::before { counter-increment: section;
             content: "section " counter(section) "."; }
h2::before { counter-increment: subsection;
             content: counter(section) "." counter(subsection) " "; }
</style></head><body>
<h1> Scripting Tutorials </h1>
<h2> JS </h2>
<h2> jQuery </h2>
<h2> React </h2>
<h1> Programming </h1>
<h2> Python </h2>
<h2> Java </h2>
<h2> C++ </h2></body></html>
```

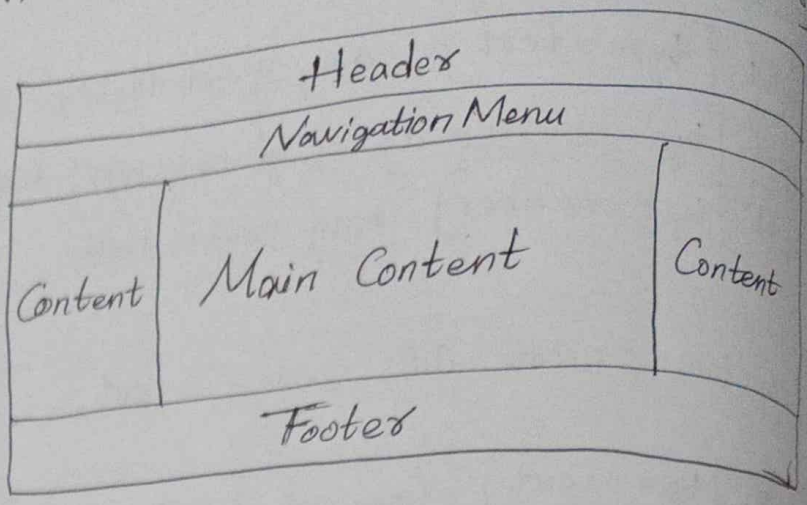
Section 1. Scripting Tutorials

1.1 JS
1.2 JQuery
1.3 React

Section 2. Programming

2.1 Python
2.2 Java
2.3 C++

CSS Website Layout (divided into headers, menus, content & a footer)



CSS Units

A whitespace can't appear between the number and the Unit
However if value is 0, the unit can be omitted.

2 types of lengths : Absolute & Relative

Absolute Lengths (units are fixed, length expressed will appear exact size)

- cm (centimeters)
- mm (millimeters)
- in (Inches (1in = 96px = 2.54cm))
- px* (Pixels (1px = 1/96 th of 1in)) (are relative to viewing device)
- pt (Points (1pt = 1/72 th of 1in))
- pc (picas (1pc = 12pt))

Relative lengths (units specify a length relative to another length property)

- em (Relative to the font-size of the element) (2em means 2 times the size of current f
- ex (Relative to the x-height of the current font) (rarely used)
- ch (Relative to the width of the "0" (zero))
- rem (Relative to the font-size of the root element)
- vw (Relative to the 1% of the width of the viewport)
- vh (Relative to the 1% of the height of the viewport)
- vmin (Relative to the 1% of viewport's smaller dimension)
- vmax (Relative to the 1% of viewport's Larger dimension)
- % (Relative to the Parent Element)

CSS Specificity (every css selector has its place in Specificity Hierarchy)

There are 4 categories which define the specificity level of Selector:

① Inline Styles (Eg: <h1 style = "color: pink;">) (value = 1000)

② IDs (#navbar) (value = 100)

③ Classes, pseudo-classes, attribute selectors (Eg: .test, :hover, [href]) (value = 10)

④ Elements & Pseudo-elements (Eg: h1, ::before) (value = 1)

The Universal Selector (*) & inherited values have specificity of "0"

CSS !important Rule (to add more importance to a property/value than normal)

Suppose in a code normal selector, class, id was declared

① (If we give some !important to the property value whole elements will follow if ~~dele~~ declared in normal selector)

② (If we give 3 elements as !important it performs individual as given)

```
#myid { color: blue; }
.myclass { color: gray; }
p { color: red !important; }
```

output: red, red, red

```
<body> <p> red </p>
       <p class = "myclass"> red </p>
       <p id = "myid"> red </p> </body>
```

CSS Math Functions

calc() – calc(expression) ⇒ expression operators (+, -, *, /) [Eg: ~~calc (~~ width: calc(100% - 100px)]

max() – max(value1, value2, ...) [Eg: width: max(50%, 300px);]

min() – min(value1, value2, ...) [Eg: width: min(50%, 300px);]