CSS

* CSS (Cascading style sheets) → used to style a web page.

* There are three ~~types~~ ways to style a web page:-
  i) Inline styling / style attribute
  ii) Internal style sheet / style tag
  iii) External style sheet (CSS)

① Inline styling / style attribute:-
We have a universal style attribute which is style
It can be used with all the HTML tags.

- This universal style attribute is used as

```
<tagname style="property:value; property:value;">content</tagname>
```

Ex:-

```
<html>
    <head>
        <title> Grouping selectors </title>
    </head>
    <body>
        <p style = "color:red ; background-color :skyblue;">This para-
        graph has a background-color. </p>
    </body>
</html>
```

Output:-

This paragraph has a background-color

red

skyblue

2. Internal style sheet:-

This way is used to simplify when the ~~same~~ lines have same property values and same tagname.

For example:-

By Inline styling:-

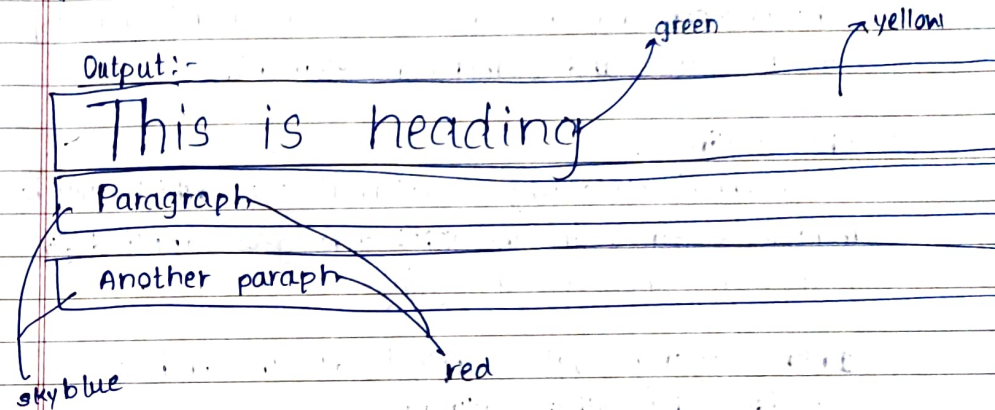
```
<html>
  <head>
    <title> CSS Page</title>
  </head>
  <body>
    <h1 style="color:green; background-color: yellow; font-size:24px"
                                        > This is heading </h1>
    <p style="color:red; background-color:skyblue; fontsize:14px"> Paragraph</p>
    <p  "  "  "  "  > Another
    paragraph </p>
  </body>
</html>
```

Converting from Inline styling to Internal style sheet:-

```
<html>
  <head>
    <title> CSS Page </title>
    <style>
      h1 {
        color: green;
        background-color: yellow;
        font-size: 24px;
      }
      p {
        color: red;
        background-color: skyblue;
        font-size: 14px;
      }
</head>
```

```
  <body>
    <h1> This is a heading </h1>
    <p> This is a paragraph </p>
    <p> Another paragh </p>
  </body>
</html>
```

Output:-



So, simply we can say that in this way we use style tag <style> to simplify the code.

3.) External style sheet:-

Helps in simplifying and introducing code in ~~6~~ 6w.

In before example we will be writting the style & property values in seperate file and add it through hyperlink in present html file.

```
<html>
  <head>
    <title> CSS Page </title>
    <link rel="stylesheet" href="project/style.csss"/>
  </head>
  <body>
    <h1> This is heading
    <p> Paragraph </p>
    <p> Another paragraph </p>
  </body>
</html>
```

(Where this style.css consists of property values.)

Note:-

> The browser renders the HTML document from top to bottom and element by element.

> The styling :-
  Internal style tag & external css. ; whichever is mentioned first in the HTML code gets applied first and then the second styling.

> Inline styling gets applied at the end & overides the other stylings.

To link a css file we have to
<link rel="stylesheet" href="css/style.css"/>
We can see the o/p in browser by typing ~~127.0.0.1/~~
127.0.0.1/project/your-filename.html

Note:-
```
selector {
        property-1 : value-1;
        property-2 : value-2;
}
```
* selector is used to select one or more elements to which we want to apply the style.
* Property means the CSS properties which we want to apply to the selector.
Ex:- font-color, background-color, etc.

* Values correspond to the specific style properties for the selected elements.
Ex:- red, 14px, etc.

Types of CSS Selectors :-

1. Element / Type Selectors :-
Used to style all the elements of a particular type at once.

In css file
```
Ex:- p {
        color : red ;
        background-color: skyblue ;
     }
```
In html file
```
<body>
    <p> This is paragraph </p>
    <p>  "  </p>
</body>
```

2. Id selectors :-
Used to style only a single element of a particular type.
Ex:- First we have to assign a unique Id to the element attribute.
* Conventions need to follow when
  1). Can have → alphanumeric characters or hyphen or underscore.
  2) Can't have any special characters.
  3) Can't start with a number.

Ex: In html file
```
<body>
    <p> Paragraph </p>
    <p> Paragraph </p>
    <p id=abc> Paragraph </p>
</body>
```
In css file
```
#abc {
    color : red;
    background-color: skyblue;
}
```
O/p :- Paragraph
Paragraph
Paragraph
red — skyblue

3. Class selectors :-

Used when we want to style more than one element.

Ex :- In html file :-

```
<body>
    <h1 class= "xyz"> This is heading </h1>
    <h2 class = "xyz"> Sub heading </h2>
    <h2> This is subheading </h2>
    <p class="xyz"> Paragraph </p>
</body>
```

In css file :-

```
.puji{
    color: red;
    background-color: skyblue;
    font size : 30px;
}
```

4. Combination of selectors

While using class selectors if we want to apply style more selectively like only one element.
We have to put element name in css before dot, like for example, (when xyz is class name)

for p :- p.xyz
for h1 :- h1.xyz
for h2 :- h2.xyz

Ex: html file is same as above given.

In css file :- If you want to apply style only for <h2> elements.

we write:

h2.puji { (In the starting that's it)

Child selectors :-

We use the concept of parent - child hierarchy to select any HTML element.

Ex :-

```
<body>
    <div>
     <ul>
      <li>
       <a href = "http://www.google.com" target="_blank"> Google </a>
      </li>
      <li>
       <a href= "http://www.facebook.com" target="blank"> Fb </a>
      </li>
     </ul>
    </div>
    <div>
     <p>
      <a href="http://www.internshala.com" target="_blank"> Internshala </a>
     </p>
    </div>
</body>
```

To style 3rd anchor tag :-

```
p>a {
    color : green;
}
```

O/p :-

- Google
- Fb

Internshala

For descendent selectors:
We just write
div a {
remaining is same like before.

Grouping Selectors :-
They help us to apply the same style to multiple sections of elements.

Ex:-
```
<body>
    <div>
        <h1> Heading </h1>
        <h2> Heading 2 </h2>
        <p> Paragraph 1 </p>
        <p> This is paragraph 2 </p>
    </div>
</body>
```
} To apply style for only these elements

CSS code:-
```
p, h2 {
    color: blue;
}
```

Attribute selectors

Ex:-
```
<body>
    <form>
        First name: <br/>
        <input type="text" name="firstname" value="Mickey"><br/>
        Lastname: <br/>
        <input type="text" name="lastname" value="Mouse">
        Password: <br/>
        <input type="password" name="password" value="Disney">
        <input type="submit" value="Submit">
    </form>
</body>
```

In CSS code:-
```
input [type="text"] {
    color: red;
}
```
O/P:-

Firstname:
Mickey ← red
Lastname:
Mouse
Password:
••••••

Attribute selectors help us to select a group of element which have an attribute with a specific value.

CSS pixel & Logical Resolution

Note:- For the laptops with pixel variation Browsers ~~css pixels~~ come up with a new idea. for laptops with 2560×1600 device pixels:-

2 device pixels = 1 CSS pixels (Both horizontally & vertically). So, technically 4 device pixels make one CSS pixel.

For device with 1366 x 768 px and 1920×1080 px
↔ 1 Device px = 1 CSS pixel.

Note:- For web development, we always deal with CSS pixels and not device pixels.

Logical Resolution - Term used to specify the number of CSS pixels a screen holds when dealing with web development.

In mobile & device px make up one CSS px.

Percentage :

* It is a relative length unit
* It gets converted to px as web page renders
* Percentage is evaluated relative to the property of its parent element

Ex: HTML CODE

```
<body>
   <div>
      <img src="img.jpeg" alt="image"/>
   </div>
</body>
```

CSS code

```
div{
   width:600px
}
img{
   width:50%;
}
```

Note:- Few more length units

em, rem } Not widely used.

red green blue

RGB values for colour

RGB values	Colour	HD value
rgb(0,0,0)	Black	#000000
rgb(255,0,0)	Red	#FF0000 (Highest no. is 255)
rgb(0,255,0)	Green	#00FF00
rgb(0,0,255)	Blue	#0000FF
rgb(255,255,0)	Yellow	#FFFF00
rgb(255,0,255)	Purple	#FF00FF
rgb(0,255,255)	Cyan	#00FFFF
rgb(255,255,255)	White	#FFFFFF

Note :- 256 * 256 = 16777216 (More than 16 Million colors)

These are also represented in hexadecimal system.

Note :-

1. A device pixel is the smallest dot on a device that can be given a definite color.
2. A CSS pixel is a web specific property and is defined as the smallest dot on web page
3. For a device with a high definition display, the logical resolution may be lower compared to its physical resolution. This is to make sure that the content does not appear too small on these devices.

CSS properties :-

For font style :-

If the class .abc is the one we selected & wanted to change font style. The code is css should be written as:

```
.abc {
    font-family : "comic sans MS", cursive, sans serif;
}
```

Note : Points to be noted while using font-family property

> Choose only those fonts that are supported by the browser.
> Different browsers support different font types so choose a font-family that is supported by all the browsers.
> So, use Google fonts

Font-size :-

```
.abc {
    font-size : 10px;
}
```

Font weight :- Boldness of font.

```
p {
  font-weight : bold;
}
```

Also you can rate from 100 - 900

```
p {
  font-weight : 200;
}
```

Also

font-weight : bolder; (can also be written.

Font style :-

italic text

```
p {
   font-style : Italic;
}
```

For Normal :-

```
p {
  font-style : normal;
}
```

text decoration :-

To add or remove underline

> Default value for most of elements is nothing but except <a>.

To underline :

```
p {
   text-decoration : underline;
}
```

To remove :

```
p {
  text-decoration : none;
}
```

Text-align :-

To allign the text.

Default is "left"

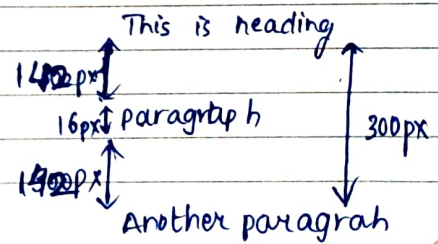
```
p {
    text-align : right;
}
```

line-height :

used to specify the height of text line.

Ex :

```
p {
   line-height : 300px
}
```

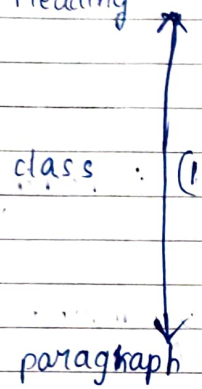


Note:-

```
.abc{
    font-size:16px;
    line-height:20;
}
```

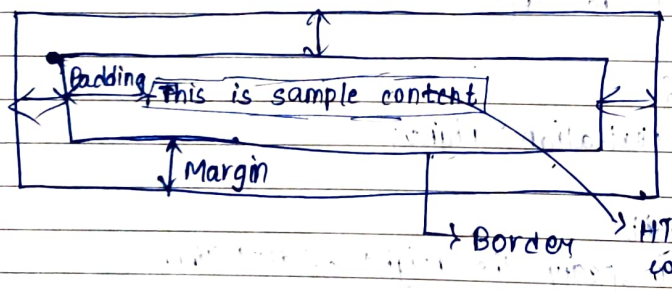
O/P:-

Heading



Box Model

In HTML every element is considered as a rectangular box which contains some more rectangular boxes.



Properties

Border style property:-

Specific a style to the border of an element.

i) solid
ii) Dotted
iii) Dashed

Ex:-

```
.a {
    border-style: solid;
}
```

Border-width:- Default: 3 pixels
Don't work if border-style is none

Ex:-

```
p {
    border-style:solid;
}
```
} specification imp

```
.abc{
    border-width: 6px;
}
```

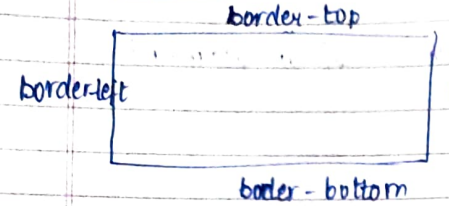
border-color:- Default → black, Don't work if border-style is non

Ex:-

```
p {
    border-style: solid;
}
```
} specification imp

```
.abc{
    border-color: [illegible];
}
```

subproperties of border :-



Ex:

```
.a {
    background color : light blue
    border-left : solid 5px blue;
}
.b {
    border-right : solid 6px red;
}
.c {
    border-top    : dashed 10px brown;
    border-bottom : dotted 8px grey ;
}
```

Always remember this order

border-style border-width border-color

Note: Padding can also be given on conditions.

```
p {
  border: solid 1px black;
}
.a {
   padding-left : 20px;
}
```

Padding :-

The space around the content of an element inside its its border

Default: 10px (Border must be their)

Ex:

```
p {
    border : solid 1 px black;
}
.a, .b {
    padding : 10px;
}
```

Note:-

```
p {
   border: solid 1px black;
}
.a {
    padding : 10px 40px 20px 5px;
}
```

Goes in clockwise manner (Top) (right) (bottom) (left)

Output :-

Note: Padding can also be given conditionally

For example :-

```
p {
  border : solid 1px black;
}
```

```
.a{
  padding-left:20px;
}
.b{
  padding-right:20px;
}
.c{
  padding-top:10px;
}
.d{
  padding-bottom:10px;
}
```

Margin:-
The space around the element outside its border and is added to maintain space between adjacent elements.

Ex:

```
div{
  border: solid 1px red;
}
.p{
  border: solid 1px black;
}
.a{
  margin: 10px, 20px 0px 5px;
}
```

O/p:-

This is paragraph with div tag. (default margin.

This paragraph has some margin.

Subproperties of margin:-

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

(clockwise order)

Way to write:

```
selector{
property: value;
}
```

here, property = (margin-top, margin-right, margin-bottom, margin-left)

Height & Width:-
Used for all elements. (Border should be their)

```
selector{
height:value;
}
```

```
selector{
width:value;
}
```

Note:- value = auto means the minimal space needed to fit the content

* Default value of height is auto
* Default value of width is different for different elements.

Actual width & height > defined height & width.

A-height = border-top + padding-top + defined height + padding-bottom + border-bottom.

Actual width = border-left + padding-left + defined width + padding-right + border-right

Box-sizing:-
Used to define height & border either in the content box or the border box.
> Values = border-box / content-box
> Default → content box.

HTML code:-

```
<div>
    <element1>
    <element2>
    <element3>
    <element4>
</div>
```

To apply box-sizing to these all elements.

CSS code:-

```
* {
  box-sizing: border-box;
}
```

Inspect Element:-
It is a browser integrated developers tool that is present in almost all widely used web browsers.

Content - Blue
Padding - Green
Border - Yellow
Margin - Orange

Display:-
It is used to specify how an element will allign horizontally with respect to other adjacent elements.
> property: value
(block, inline, inline-block, none)
Default is diff for diff elements:-
paragraph → block
anchor elements → inline.

Display: inline:
> An element starts beside the previous element, provided the previous element allows it to sit next to it & there is enough space.
> Width and height css properties have no effect.

```
a {
  border: solid 1px black;
  width: 200px;
  margin: 10px;
  padding: 10px;
}
```

Display block:-

> An element always starts from a new line even if there is enough space next to the previous element.
> Takes entire width available.

Ex:- a {
    border: solid 1px black;
    display: block;
    width: 200px;
}

Display: inline-block

> Elements acquire properties of inline and block elements.
> The elements sit next to each other and take only required space.

a {
    border: solid 1px black;
    display: inline-block;
}

Even if we give some width: 200px;
It increases but blocks sit next to eachother.

Display: none
Used to hide the HTML elements from the users.

	Block	Inline	Inline-block
Previous element	Always starts from a new line. won't allow to sit beside it	Starts next to the previous element	starts next to the previous element
Next element		Allows	It also allows
Default width	Takes entire width available	Takes only as much space as required	Same as Inline
Explicit height/ width	Can set height & width.	Can't set height & width but paddings & margins can be.	We can set height & width

Position:-

Helps to move HTML elements from their existing positions.
> This property has four values -> static, relative, fixed and absolute.
Note: -> positioned <-

1) position: static
elements cannot be moved from their position.

2) position: relative
It helps to move an HTML element with respect to its original position.

Ex: .abc {
    position: relative;
    top: 50px;
    left: 10px;
}

position: fixed;
> It moves the element with respect to the browser window.
> The position of the element remains fixed even if we scroll the web page
> The next element present after the positioned element takes the place of the positioned element leaving no gap.

position: absolute
It moves the element w.r.t the nearest positioned ancestor element.

	Movement w.r.t	Flowgap:-
Relative	Og position	No other element tries to fit & leaves a gap.
Fixed	Browser window	Next element takes its original place & leaves no gap
Absolute	Nearest positioned ancestor.	" " " " " " " "

z-index :-
> Used to position an element along z-axis.
> Used for overlapping.
> Higher z-index value of an element, the more close it is to us.
> It works only on positioned elements.
> Works only on positioned elements.

Ex: h1 {
position: relative;
z-index: 1;
}
p {
background-color: red;
position: relative;
z-index: 2;
}

Designing a Web Page

Wireframe

Diff ways to create a wireframe.
1. display: inline-block
2. float
3. flexbox

1. Display: inline-block:-

In this process we use tags and

border: solid 1px green;
display: inline-block
height:
width:

2 float:-

We use tags
border: solid 1px red;
height:
width:
float: right/left/centre;

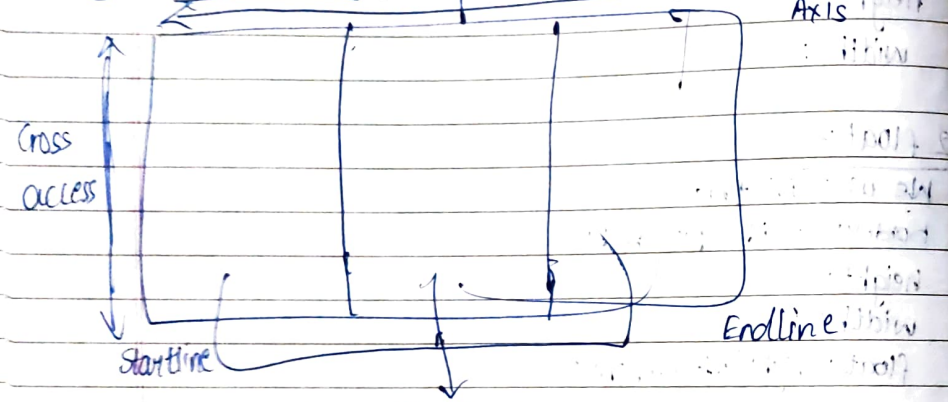
Note: We use this
* { box-sizing: border-box;
}
to ignore the spaces and left the boxes fit in frame exactly.

Note:- tags take up one entire row from left end to right end.
The parent element collapses to height 0px.

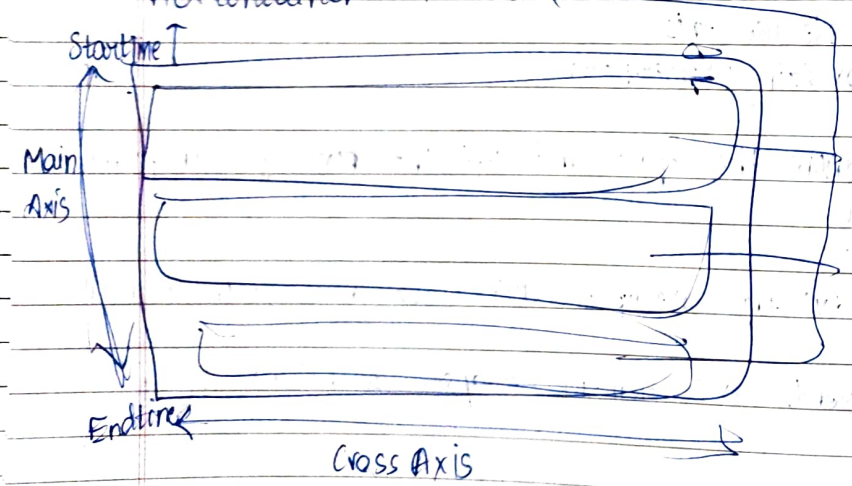
We can fix the problem where the parent element doesn't collapse to height 0px caused by float:left by adding the following code to the parent element:

```
.row::after {
    content: "";
    display: block;
    clear: both;
}
```

3. Flexbox (display:flex) → Flex container



Flex container Flex items



Properties:-

1. flex-direction: used to define direction for flex items.
Can only be applied to parent element.

flex-direction: (alignment)
- row (start to end)
- row-reverse (end to start)
- column (start to end)
- column-reverse (end to start)

flex basis:
Applied to flex-items.
To define the size of the item along the main axis.
Default: auto

Length value overrides the width property even if explicitly defined.

*. A CSS property used in flex boxes, that sets the initial size of a flex item before it grows or shrinks.

flex-grow:-
Applied to flex items.
Default value = 0
Only applicable if container has available space after placing all the flex items.
Its default value is 0.

Flex-shrink:-
App. to flex items (only if container has deficit space.
Default = 1.

These three properties of flex can be declared easily as

flex: flex-grow(value) flex-shrink(value) flex-basis(value)

* flex-wrap:

App to container. (deficit space).

* flex-wrap: wrap; → sends the last element to next row if there is deficit space.
* flex-wrap: nowrap; → shrinks in the container..

* flex-flow → combines both flex-direction & flex-wrap.

App to container.

Ex:-

flex-flow: row wrap;

justify-content:-

App to flex container. (Available space)

justify-content:-

- flex-start
- flex-end
- center
- space-around
- space-between
- space-evenly.

Align items:-

- App to flex container
- how items should laid out along the cross axis.
- four values: strech, flex-start, flex-end and center.
- Default is strech.

self-align:-

Individual flex items.

Same values and Default is also same.

Note:

* flex-item will use the width property instead, and not minimal space to fit in the content. Flex-basis overides the width property if given explicit value.
* When flex-wrap: wrap; → flex-shrink has no-effect on the items.
* Flex-shrink has significance only if the combined width of all the items is more than of the container.

Responsive Webpage:-

Viewport: View of screen (diff for diff types of devices)

Media queries:

The browser will include a block of css properties only if a certain condition is true

Ex:-

```
@ media (max-width: 600px) {
    body {
        font-size: 14px;
    }
    .logo {
        width: 200px;
    }
}
```

Note:- CSS written outside any media rule block is applicable for all viewports unless overridden later by some media rule.

Ex:-

```
.logo {
    width: 300px;          (for other devices)
}
@media(max-width:600px) {
    .logo {
    width: 200px;          (for mobilephones)
    }
}
```

Mobile - @media (max-width:600px) (360px * 640px)
Tablet - @media (" " :1000px) & (min-width:600px)
Laptop - @media(min-width:1000px)

If we want to make a device friendly web-page:-
<meta name="viewport" content="width=device-width, initial-scale=1">

Note:-
<meta name="viewport" content="width=device-width, initial-scale=1">
↳ This line makes some help to create device friendly webpage