CSS

* Cascading Stylesheet.
* xyz
  This is inline css.
* ~~<head>~~ Below is internal css.

```
<style type = "text/css">
    p {
        color : blue;
    }
</style>
```

put this in head or atleast.

* External css :- Need to create file with .css extension. example xyz.css.

```
p {
    color : blue;
    font-size : 20px;
}
```

and link it in head tag like below.

```
<link rel = "stylesheet" type = "text/css" href = "xyz.css">
```

(path)

* Preferaces Rule

Inline css > [Internal CSS = External CSS]

Last Rule will applied.

* External Css :-
  - It helps in seperating content related code and presentation related code of website.
  - It may or may not have lesser priority than internal css.
  - It helps in writting less repeated code hence the loading time of page decreases.
  - It helps in easy debugging and maintenance of code.
  - It is preffered over inline and internal css. for styling purposes, as you can not reuse inline or internal CSS with other HTML code files.

* Type Selector.

```
tag {
}
```

* Class Selector :- add 'class' attribute to the html tags for which you want to apply same style multiple & separated by space in html & by comma in css. class names can be

.classname {
    propertyname : value ;
}

* class & tag selectors are used to specify a group of elements.

* Id selector :- Id should unique to each element.
- add id attribute &

idname {
    propertyname : value ;
}

* Preferenance or specificity of selectors.
Inline css > id selector > class selector > Type selector

* Super class :- Apply specific style when condition satisfying.
like (:hover)

* Colors :- Names, Hexadecimal codes (#), rgb(,,), rgba(,,,), hsl(hue saturation light)
0-255 opacity 0-1
h s l
0 120 240 0-100 0-100

* CSS Units :-
Absolute - exact
1 cm
1 inch
1 px = 1/96 inch.

Relative :- percentage of to parent
% relative to parent
Relative Browser - vh - viewport height
vw - viewport width

Relative to font size - rem - 1 rem - fontsize of root element (<html>)
em - fontsize of parent

* Border :-
Border-width, border-style, border-color, border-radius
top right bottom left
solid / dashed / doubled
Shorthand : border : 4px solid red;

* Text Styling :-

font-size : 30 px ;
color : blue ;
font-weight : bold ; /* 100 - 900 */
text-align : justify ;
line-height : 30 px ;
word-spacing
text-transform : uppercase ;
text-decoration : underline ;

* Background :-

background-image : url (path) ;
background-repeat : no-repeat ;
background-position : center ;
background-size : contain ;
background-attachment : fixed ;

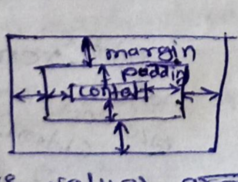
- We can set more than one background image.

* Margin :- Space around element.

margin : 4 px ; — all ~~sides~~
margin : 4px 8px :- Top-botto . left right
margin : 4px 8px 6px ; To left right botton
margin : 4px 8px 6px px ; top right bottom left
margin-top : px ;
margin : auto ; — center of ~~screen~~ parent

- Vertical margins collapse.
- Horizontal margins don't collapse, they add up.
- we can use negative values in margin ~~property~~

* Padding :- Space within content & ~~element~~ border of element



- No negative values ~~are allowed~~.

* Display Property :

Inline :- should not have height & width and vertical margin + padding
Block :-
Inline-block : there is some spacing.

* Position Property.
(i) Static (ii) relative (iii) absolute (iv) fixed
(v) sticky

(i) Static :- Normal position.

(ii) relative :- relative to normal position.
position : relative;
left : 20px;
space normal position also get blocked.

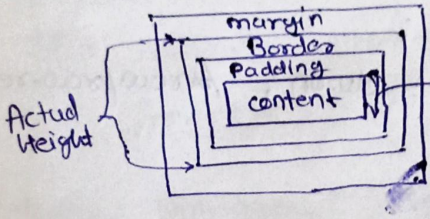
(iii) absolute :- with respect to first position parent.
→ No space get reserved.

(iv) fixed :- stick to viewport ; not take place.

(v) sticky :- fixed till container. start sticking from some point (depends on top)

* Box Model.

(1) Content Box Model :- ~~marg~~ Default



Actual Height = content + padding + Border

box-sizing: content-box;

margins dont add up.

(2) Border Box model.

box-sizing : border-box.

Then, actual Height = Height.

* Min Max Width :-
min-width
max-width.
overflow : scroll ; scroll / hidden / auto

* *{} to apply style to every element of HTML page.

Resume Project.

Home → About → Skills → Experience → Education,
Contact ← Portfolio

Select <x> that is immediately preceded by <y>
y + x {...}

* tag with specific attribute.
input[disabled]{---}

* Target all the images whose extension is "jpg".
img[src$="jpg"]{---}

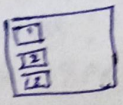
* box-shadow property :- Used to produce shadow like effect for an element.

syntax box-shadow: none | h-offset v-offset blur-rad spread-radius color.

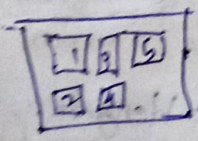
h-offset & v-offset are required.
values can be negative.

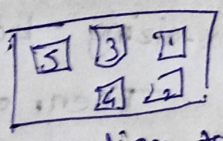
* Flex :- arrange multiple element vertically & horizontally smoothly.

display: flex;
flex-direction: column; /* row/row-reverse/column-reverse */



order: 1; Higher the order last placed.
flex-wrap: wrap; } wrapping in multiple line.
wrap-reverse / nowrap





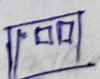
flex-grow: 1; /* grows according to flexgrow" Total of flex grow of sibling.

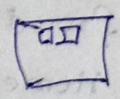
flex-shrink: 4; /* shrink by 4 pixels.

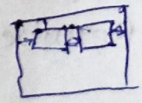
justify-content: flex-start; /* align items across for row main axis

default.

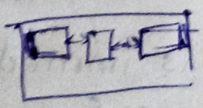
flex-end, center, space-around







space-between;

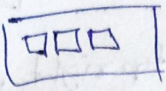


* align-items - Align items across cross axis. for row

align-item: flex-start; flex-end / center

stretch < height

baseline



* nth-child

section: nth-child (n) { - - - }

* .class > div - only to direct div child

.class div - all div childs

* flex-basis: ; initial length to the flex item.

- shorthands

* flex-flow : flex-direction flex-wrap (initial | inherit;

initial - represents flex flow property as (row nowrap)

inherit - inherits this property from parent element.

* flex : flex-grow flex-shrink flex-basis (auto | initial)

- auto - 1 1 auto

initial - 0 1 auto

none - 0 0 auto

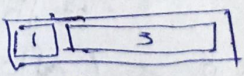
inherit :- property from its parents.

None | inherit.

* Grid :- To create 2-dimensional layouts.

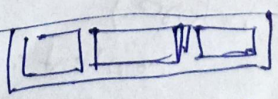
display: grid ; grid or inline-grid;

grid-template-rows: 1fr 3fr;



we can use px, %, rem also

grid-template-columns: repeat(3, 1fr);



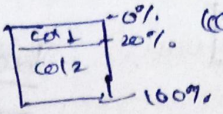
* Overflow-wrap: break-word; (word overflow) word-wrap

normal

* Gradients:

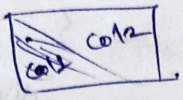
background-image: linear-gradient(color1, color2, ...);

can use 20PX;



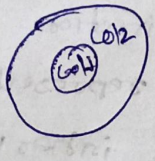
linear-gradient(to top right, col1, col2);

can use deg also



background: repeating-linear-gradient(col1, col2)

radial-gradient(col1, col2)



* Pseudo Classes:— tag: link / active / visited

link
active - session is active
visited - vi
target = selected / clicked

* transition: all 2s ease-in-out;

* Pseudo elements: tag::element{ ... }

p::first-element{ }

::first-line

* before / after pseudo element -- add virtual element before & after.

p::before{

content: "xyz";

...

}

required

* z-index :- higher the z-index value it will be visible over element with lower z-index.

* Advance CSS

* Responsive website.

Three ways to make responsive.

① Responsive web design - (CSS Rules).

② Dynamic serving :- Use Html/CSS/JS/etc code based on the device we are using. Amazon is using it.

③ Seperate mobile website :- Different urls used. generally starts from 'm' facebook.

* Responsive Web Design :-

* Measurement units :- ① Absolute ⓘⓘ Relative.

Absolute :- px, in, cm

Relative :- %, vh, vw, rem, em
(rem: font size of HTML | em: font size of parent)

* fluid layouts : content of the website adjust according to the browser window. Relative units are used.

* fixed layout :- Dimension are fixed. Absolute units are used.

* Elastic layout :- content of the website adjust according to the font size. Relative typographic unit used. like em.

* Hybrid layout :- mix of any of the above.

* Viewport Meta tag :- (It can be seen in Google)

<meta name="viewport" content="width=device-width, initial-scale=1">

* Media Query :- ① media type :- screen, print, speech, all

② media-properties :- ⓘ width ⓘⓘ height ⓘⓘⓘ Resolution

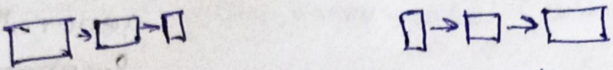
@media screen and (max-width: 600px) { - - }

ex. (only use for use)
@media screen and (max-width:600px){ - - - -
- - - - -
- - - -
}

~~High condition to~~ lower condition:
critical at last.

max-width : 900 px { width:80%; }
↓
max-width : 600 px { width: 60%; }

* Design approaches :
(i) Desktop first (ii) Mobile first



min-width : 600 px{80%}
min-width : 900 px{60%}

* media features : width, orientation, resolution etc

* @media { max-width: 600px}{ p{- - - }} .

* Transitions & Transform

transform : rotate(xodeg); (+deg clock wise)
: scale(0.5, 0.4);
: translate(100px, 100px); — moves

transition : 1s ease-in-out; default "all"
: height 1s ease-in-out 0.5s ;
on what — for How much — How ease-in, ease-out, linear — after How much.

use hover after

* Animation :
animation : changeShape 1s ease-in-out { no of times for animation 2, alternate, infinite } (animename)

* create animation
@keyframes changeShape {
0% from { border-radius: 0%; }
100% to { border-radius: 50%; } to animation happen smooth

We can add as many stages between 'from' & 'to' in form of %.

ex from { - - - } — from can be 0%
10% { - - - }
20% { - - - }
to { - - - } ——— to can be 100%.

We can create multiple animations and apply them to same element by adding ',' comma.

animate.css
copy paste CDN and give class to to element which ever animation you wanted to use.