Git / Github

1. What is version control?
2. Need of version control.
3. principle of version control.
4. Evaluation of version control.
5. Emergency of Git.
6. understanding Git
7. Need of Git Hub.
8. GIT deep dive
9. GIT hands-on.

GIT → 2005

CSS cashcad Styling Sheet.

IDE → Integrated development environment :-

Level ①: Css Basic
level ②: Color System and Background.
level ③: Text properties
level ④ : Box model CSS
level ⑤: Display & position
level ⑥: flexbox, Grid & media Queries
level ⑦: Animation, Transition & Transform
Project : myntra Clone.

1. Basic dyntax :-

```
Sector
h1 { font-size : 30px ;
     ↓properties  ↓value
}
```

→ Declaration.

→ Selector:- The HTML element that you want to style.

→ Property: The attribute you want to change (like font, color, etc)

→ value: The specific style you want to apply to the property (like red, bold, etc).

Lorem → to create rand

Ex:
```
<!DOCTYPE html>
<html>
<head>
<title> Color property </title>
<style>
h1 {
color: green
}
p {
color: red
}
</style>
</head>
<body>
<h1> This is the heading </h1>
<p> <b> i am priyanka </b> </p>
</body>
</html>
```

* Inline Styling :→
- Direct Application : Apply styles directly to HTML elements using the style attribute.
- One-off Changes : Ideal for single, unique style alteration.
- Can be cluttered : May lead to cluttered HTML if used extensively.
- Limited Reusability : Reduce the reusability of CSS rules in layer project.

Ex: <h1 style = "color : green"> this is the heading </h1>

2. Including Styling (External Styling)
- separate css file : stores styles in in separate.cs

[8. selections (Descendant selector)]

```
<title> Descendant selector </title>
  <style>
     div p { color: red }
      p { Color : blue }
  </style>
  <head>
  <body>
    <div>
      <p>
        Loren ipsum dolor sit amet
      </p>
    </div>
    <p>
      loren ipsum dolor sit amet
    </p>
  </body>
```

(AI) → generative pre-trained transformer

- Nested Targeting : Styles elements nested within a specified element.
- Syntax: separate selectors with spaces.
- Hierarchy-based: Allows styling based on the hierarchal structure of H.
- Specific styling : facilitates more targeted & specific styling of elements.

* Practice Set Level 1

- Create a heading and set the text color 'red'.
- Create a div with id # heading, include css using all 3 ways Inline, style tag and external and observe property.
- Add comments to your css class
- ~~Class~~ create a div, paragraph and headings & use id selector, element selector & class selector for them.
- Create two divs with id first and 2nd and define color for both using group selector.

* <!--- comment -->
or, /* comment */

* <!doctype>
<html>
<head>
<title> practice set 1 </title>
<style>
p {
color: green;
}

```
#second {
        color : violet; }
/* this is my comment */
#heading {
        Color : blueviolet;
</style>
</head>
<body>
<h1 style = "color : red"> <center> this is my heading
                                <center></h1><hr>
<center><div id = "heading" Style = "Color : gold">
                heading </div> <hr/>
<div id = "div-22" style = "Color : green"> div </div>

<p> lorem ipsum dolor amet consecutor adipiscating elit.
Recusande, priyanshu Roy ? </p></hr>
<div id = "first"> < first </div>
<div id = "second"> second </div>
<div id = "third" style = "Color : yellow" >third
        </div></center>
</body>
</html>
```

Level-2

Color System & Background

9. Background-color property
10. Color system
11. Absolute units
12. Height & width property
13. Background-image property
14. Visibility property.

(9) Background color:-

Syntax:→ background-color : color ;

(10) Color-system:→

(color theory):→

- RGB model : creates colors by mixing Red, Green, blue light sources.
- Additive model:- more light means increased brightness.
- Primary colors:- R, G, & B.
- Color depth:→

Ex.: first

```
<!doctype>
<html>
<head>
<title> Background-Color </title>
<style>
#first {
    background-color : orangered
    }
#second {
    background-color : white
        }
#third {
    background-color : green
            }
</style>
</head>
<body><b><center>
<div id = "first"> first </div>
<div id = "second"> Second </div>
<div id = "third"> Third </div></b></center>
</center><div style = "background-color : rgb (0, 0, 255);">
        <center> forth </center></div>
    <body>
    <html>.
```

* * *** Hex Color model * * * *

- Hexadecimal codes :⇒ Represents colors using hexadecimal values, consisting of 6 digits combined from numbers & letter (A-F).

- Syntax :- written as #RRGGBB.

- Easy color matching:- facilitates easy color matching with graphic design tools and branding colors.
- Web Standards: widely supported and a common standard for defining colors in web design.

Ex:→

```
<div style = "background-color : #ffcd2">
                                first</div>
```

Absolute units

- Definition: Pixels(Px), are fixed-size units, presenting a dot on a computer screen.
- Precision: Allows for precise control over element dimension.
- Graphics & web design:-
  Commonly used in graphics & web design for setting fonts sizes, margins, and more.
- Cross-Browser Consistency: provides consistency across different browsers.
- High-DPI Displays: Can vary in appearance on high-DPI (dots per inch) displays.

Height & width property

```
<head>
  <title > Height </title>
  <style>
    . box { hight : 40px ; width : 40 px ; }
    # box1 { background-color : red ; }
    # box2 { background-color : blue ; }
    # box3 { background-color : green ; }
```

</style>
<head>
<body>
 Box 1
 Box 2
 Box 3
</body>

*** Background image Property ***

- usage: Adds an image as a background to elements.
- syntax: Define using background-image:
  url ('path/to/image');
- Repetition: Control image repetition using background-size.
- Background-size.
- Background-Attachment: sets whether the background image scrolls with the elemet or remain fixed.
- shorthand (color, image, repeat, attachmet, position).

W-3 L-23 Podded Engine:

Why put Engines in pods on wing?

- better access to work on engines
- Panenger safety during accidents

Ex-: <title> Background image </title>

```
<style>
    #icon {
        height : 50 px;
        width : 50 px;
        background-image : url (mynta.png);
    }
</style>
</head>
<body>
    <div id = "icon"></div>
</body></html>
```

color: "red";
background color: "red"

Ex: <!DOCTYPE html>
<head>
<title> visibility </title>
<style>
#box1{
height: 50px;
width: 50px;
}
</style>
</head>
<body>
 box1
 box2
</body>

Ex:->

#box2{
hight: 50px;
width: 50px;
visibility: hidden;
}

Ex: <!Doctype html>
<html lang = "en>

<head>
<title> Practice Set 2 </title>
<style>

}
# bar {
background-color : aquamarine;
Color : rgb(103, 18, 18, 0.8);
}
main { height : 500px;
background-color : blanchedalmond; }

#product {
background-image : url(learn.webp);
width : 800;
height : 500;
backgroundposition : center;
background-repeat : 400px;

</style>
</head>
<body>
<header>
 this is my nav bar
</header>
<main>
 ~~this is my nav bar~~
</main>

<footer>

</footer>
</body> </html>

Level : 3 (Text Properties)

- Text-Align Property
- Text-Decoration property
- Text-transform property
- line-Height property.
- font properties
- font family
- Icon using fonts.

Ex: (1)

```
<head>
  <title> Text Align </title>
  <style>
    .box { height : 100 px ; width : 100 px ; }
  #box1 { background-color : red; text-align : center; }
  # box 2 { background-color : green ; text-align : left; }
  # box 3 { background-color: blue; text-align : right ; }
  </ style>
 </head>
 <body>
<h3 id = "box1" class = "box" > Box 1 </h3>
<h3 id = "box2" class = "box" > Box 2 </h3>
<h3 id = "box3" class = "box" > Box 3 </h3> .
 < body .
```

text-align

Syntax: text-alignment = center/last/first

(2) Text decoration :-

```
# box 1 {
```

Syntax: text-decoration : underline ;
" " : overline ;
" " : sign through ;

}

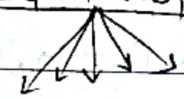
. → class ko define karne ke liye

→ id ko define karne ke liye

** Syntax :→ #box1 { text-decoration-style : dashed;
" " " : double;
" " " : solid;
" " " : wavy;
}

*** syntax :→ #box1 { text-decoration-color : red; }

17 : Text-Transform Property



#box1 {
text-transform : uppercase
" " : lowercase;
" " : Capitalize;
" " : none; }

18. line height

```
<!DOCtype html>
<html lang="en">
<head>
<title> line Height </title>
<style>
#article {
line-height : 15px ;
}
</style>
</head>
<body>
<p class="article"> myname is priyanshee Rai d.ngu.
</body>
```

19: font Property

#first1 { font size : 5px; }

#second2 { font-weight : 600; }

third { font-style: normal / oblique / italic }

20: font family

- usage :- Defines which font should be used for text within an element.
- Specific fonts: Common choice include Arial, Segoe UI, Times New Roman, and others.
- fall back mechanism : Incorporate a fallback font family in case the primary font is unavailable; helps in maintaining the site aesthetics.

- web safe fonts: employ web-safe fonts to ensure consistency across different browsers & operating systems.
- Generic family : Always end the font family list with a generic family like serif or sans-serif as a last resort option.

21: Icons using fonts

```
<head>
<title>ICONS</title>
<script src="https://kit-fontawesome.com/
43290fa92d.js" crossorigin="anonymous">
<script>
</head>
<body>
<i class="fa-solid fa-house"></i>
<i class="fa-brands fa-facebook"></i>
<i class="fa-brands fa-tiktok"></i>
<i class="fa-solid fa-poo" style="font-size:
40px;"></i>
<body>..
```

Practice Set level: 3

- Create an heading at the centre & make capitalize.
- use font family for the whole page to Timenew Roman.
- create one div inside another div.
  set id and text outer to outer div,
  set id and text inner to inner div.
  set id & text outer to inner div.
  text size 25px.
  set inner to 10px.
  set inner to 10px.
- use icons from fontawesone.com & use icons of linkedIn and GitHub.

Level-4 : (Box model):

22.* What is Box model :→

- Core concept :
  Central concept in CSS at outlines the design and layout of elemets on the web page.

- Components : Consists of four main components - margin, border, padding, and content.
- margin : The space outside the border, separating the elemet from others.
- Border : The outline that encapsulates the padding and content.
- padding : The space b/w the border & the actual content, providing a buffer.
- Content : The innermost layer where text, images, or other media are housed.

Ex-: <head>
<title> padding </title>
<style>
* {
margin : 5px 10px 15px 20px;
background-color : aquamarine;
}
<style>
</head>
<body>
<button id = " button1 "> Click me </button>
</body>.

* What is Box-model:
* Padding property
* Margin property
* Border property

Ex.: <!. doctype html>
<html lang = "en">
<head>
<title> Color and visibility </title>
<style>
* {
margin:0;
padding : 0;
}

.pad {
height : 100px;
width : 100px;
padding 2 cap;
box-sizing : border-box;
}

```
#first {
    color: rgb(16, 235, 235);
    background-color: blue;
    border: 10px solid rgb(226, 30, 8);
    border-radius: 30px 30px 0 0;
    margin: 5px;
    text-align: center;
    text-transform: uppercase;
}
#second {
    color: purple;
    background-color: rgb(226, 30, 8);
    border-radius: 0 0 30px 30px;
    margin: 5px;
    text-align: center;
    text-transform: uppercase;
}
</style>
</head>
<body>
<div id="first" class="pad">first</div>
<div id="second" class="pad">second</div>
</body>
</html>
```

Practice set level →4

Level→05 (Display & Position)

* Display property
* Responsive website
* Relative units
* Position property
* Semantic Tags.

Block-Elements

- New line : Start on a new line.
- full width : Take up all horizontal space.
- Styling : can have margins & padding.
- Size : width & height can be set.
- ex.: <div>, <p>, <h1>, <ul>,

Inline-Elements

- flow
- width
- No break
- Limited styling
- Ex: <span>.

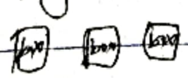
① Display property :-

```
.box {
    height : 100px;
    width : 100px;
    background-color : blueviolet;
    margin : 10px;
    text-align : center;
    border : 5px solid black;
    display : block;
}
```

display : block :-

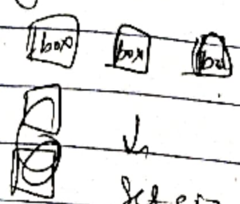
↓
set size

display : inline



↓
not set size

display : inline-block



↓
set size

display : None.

→ not fix size

(87) Responsive design :→

- Adapts layout for different screen sizes
- flexible layouts
- optimizes images & assets
- enhances user experience on mobile & desktop

(88) Relative unit (for responsive design) :-

```
ex: #first {
        height : 200px;
        width : 200px;
        bgcolor : aqua;
        font-size : 25px;
    }
    #second {
        bgcolor : blueviolet;
        width : 50%;
        height : 30%;
    }
```

* Relative units (VW/VH) :-

```
<head>
  <title> unit VH & VW </title>
  <style>
    #first {
        height : 50vh;
        width : 90 vw;
        bgcolor : red;
    }
  </style>
</head>
<body>
    <div id = "first"></div>
</body>
```

29 (Position Property):

Static Relative absolute fixed

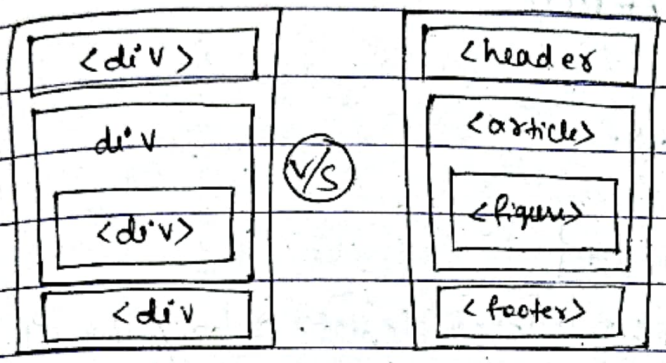
- Static :→ Elements follow the normal document flow. (top, right, bottom, left, z-index would not work).
- Relative :→ Element's position adjusted from its normal position.
- Absolute : Positions element relative to the nearest positioned ancestor.
- Fixed :→ Element positioned relative to the viewport, does not move on scroll.

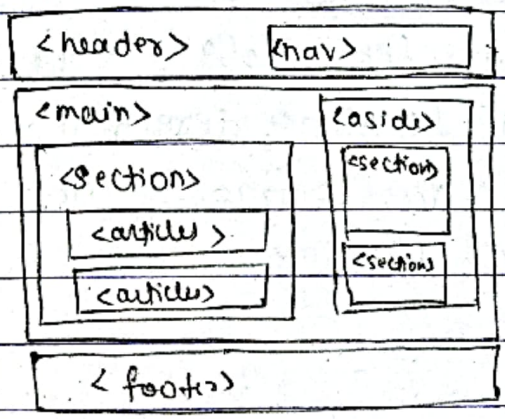
- Position property (z-index)

```
.container {
    position : relative;
    }
  . box1, .box2 {
    position : absolute;
    border : 3px solid black;
    width : 100px;
    height : 100px;
    text-align : center;
    font-size : 25px;
    }
  .box1 {
    bgcolor : red;
    left : 20px;
    top : 60px;
    z-index : 2;
    }
  .box2 {
    bgcolor : aqua;
    left : 60px;
    top : 20px;
    z-index : 1;
```

- stacking order: determines the stacking order of elements along the z-axis.

Semantic Tags





Semantic Tag	Non-Semantic Tags :→
• Meaningful: Describe content	• Generic: No specific meaning
• SEO: Good for search engines	• For styling: used for layout
• Accessibility: useful for screen readers	• No SEO: Not SEO-friendly
	Ex: , <b> ..

Ex: <header>, <footer>, <article>, <section>, <nav>.

Practice Set level 5

Level-6

31. float property
32. What is flexbox?
33. flex model
34. flexbox direction.
35. properties : flexbox Container.
36. properties : flex items.
37. Grid layout
38. Media Queries.

31. float property

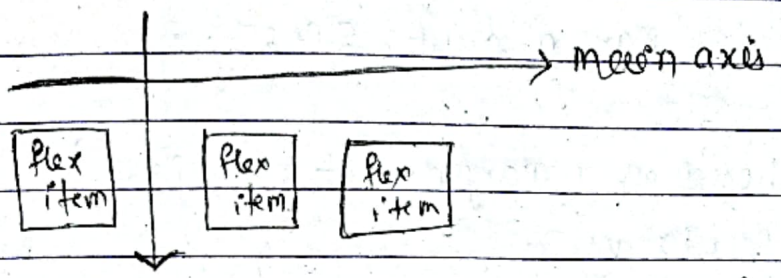
- Element Alignment :→ Allows elements to be aligned to the left or right within their containing element.
- Values :→ Can take values like 'left', "right", or "none" to determine the floating direction.
- Old layout Technique :→ less commonly used with the advent of flexbox :-

Ex:
```
.container {
    height : 110 px;
    width : 300 px;
    border : 1px solid
}
.box {
    width : 100 px;
    height : 100 px;
    margin : 5px;
}
.box1 {
    background-color : red;
    float : right;
}
.box 2 {
    background-color : blue;
    float : left;
}
```

(32) flexbox ?

:→ flexbox is a one-dimensional layout layout method for arranging items in rows or columns.
items flex (expand) to fill additional space or shrink to fit into smaller space.

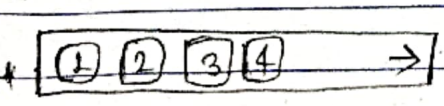
(33) * flex model :→



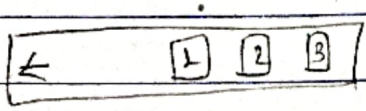
display: flex.

(34) flexbox Direction :→

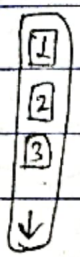
- property Name :→ flex-direction is the propery used to define the direction in a flex container.
- Row layout :- Row value aligns the flex items horizontally in a left-to-right fashion.
- Column layout: column value stacks the flex items vertically from top to bottom.
- Reverse Direction : Adding - reverse to ~~flex~~ row or column (as in row-reverse or column-reverse) reverse the order of the items.



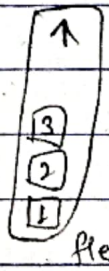
flex-direction : row



flex-direction : row-reverse



flex-direction : column



flex-direction: Colum-rever

Ex:- * {
margin: 0;
padding: 0;
}

.box { height: 100px;
width: 100px;
border: 3px solid black;
margin-right: 5px;
}

heading { margin-left: 200px; }

container {
height: 150px;
width: 600px;
padding: 10px;
margin: 20px;
border: 3px solid black;
display: flex;
flex-direction: row-reverse;
}

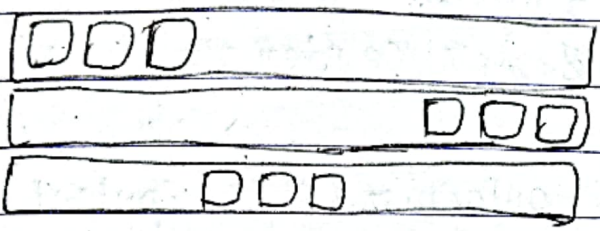
box1 { background-color: aqua }
box2 {
box3 {
box4 {

display: flex;

d :- flex-direction :- row/row-reverse
or column/colun-reverse.

(35) properties : flexbox Container (justify Content) :->

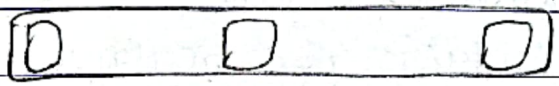
- Alignment : Aligns flex items along the main axis.
- flex-start : items align to the start of the flex.
- flex-end : items align to the end of the flex Containe
- center : items are centered within the flex container
- space-between / space-around / space-evenly.
  Distributes space between items evenly.



flex-start

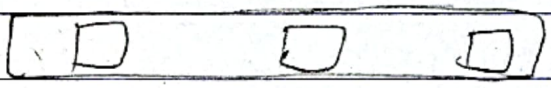
flex-end.

flex-center



space-between

Space-around.



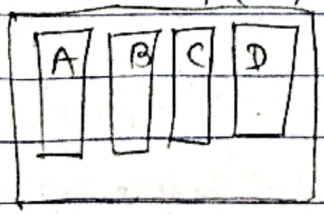
Space-evenly.

space-between.

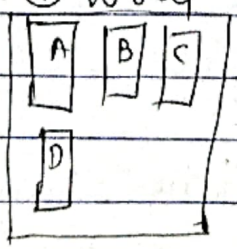
syntax : justify-content: ~~space~~ flex-end / flex-start.

(11) properties : flexbox Container (flex-wrap) :-

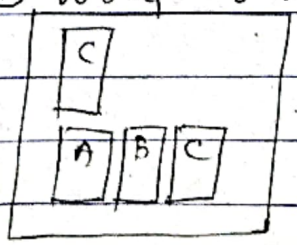
① nowrap (default)



② wrap



③ wrap-reverse



* properties: flexbox container (Align items):→
This property is used to align the flex container's items along the cross-axis, which is perpendicular to the main axis.

Ex:
```
display: flex;
flex-direction: row;
justify-content: center;
align-items: center;
```

* properties: flexbox Container (Align content):-
It is utilized to adjust the spacing between flex lines within a flex container, particularly when there is extra space along the cross-axis.

```
display: flex;
flex-direction: row;
justify-content: center;
flex-wrap: wrap;
align-content: center;
```

* (30) properties: flex items (flex shrink):-

```
#box 3 {
    background-color: yellow;
    flex-shrink: 4;
}
```

→ The "flex-shrink" property in CSS determines how much a flex item will shrink relative to other items in the flex container of there is insufficient space.

36. Properties : flex item (flex Grow) :

```
# box3 {
    background-color : yellow;
    flex-grow : 1;
}
```

→ The "flex-grow" property in CSS specifies how much a flex item will grow relative items in the flex container when additional space is available.

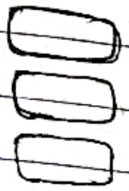
36. Properties : flex items (order)

Ex:
```
# box1 {
    background-color : aqua;
    order : 3
}
# box2 {
    background-color : blueviolet;
    order : 1;
}
# box3 {
    background-color : yellow;
    order : 4;
}
# box4 {
    background-color : tomato;
    order : 2;
}
```

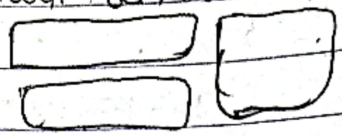
→ The "order" property in CSS allows you to define the sequence in which flex items appear within the flex container, overriding their original order in the HTML.

37. Grid

flexbox :→
one-dimensional layout



Grid :→
Multi-dimensional layout



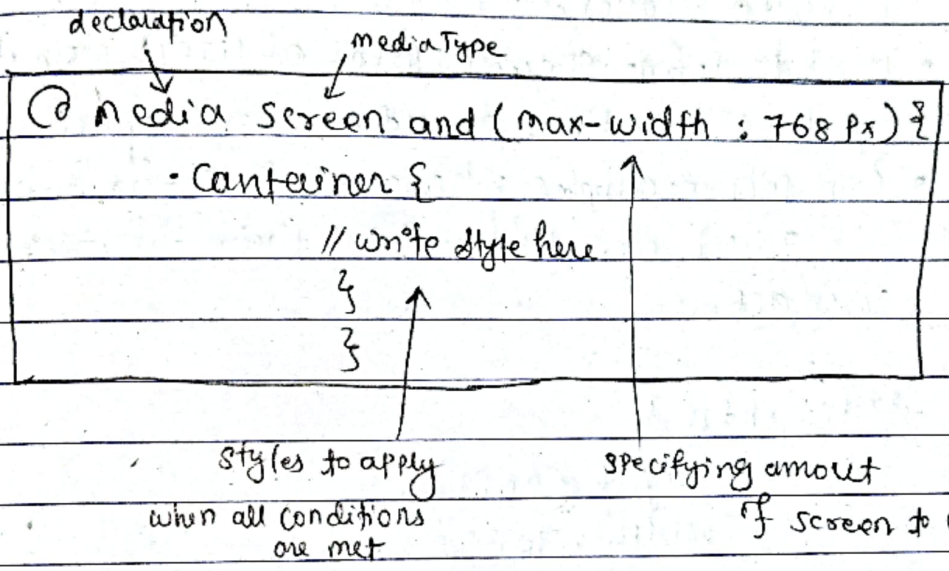
→ 2D layout system for rows & columns.
→ Activate with display: grid;
→ Children become grid items.
→ Define structure with grid-template properties.
→ Individual units called grid cells.

ex:-

```
.Container {
    display: grid;
    grid-template-columns: 50px 50px;
    grid-templet-rows: 50px 50px.
}
.item1 {
    grid-column: 1/2;
    grid-row: 1/2;
    background-color: lightblue;
}
.item2 {
    grid-column: 2/2;
    grid-row: 1/2;
    background-color: lightgreen;
}
.item3 {
    grid-column: 1/2;
    grid-row: 2/2;
    background-color: lightpink;}
.item4 {
    grid-column: 2/2;
    grid-row: 2/2;
    background-color: lightyellow;
}
```

38. (Media Queries)

- Tailor styles for specific device characteristics.
- use to create responsive web designs.
- Apply styles based on conditions like screen size.
- Syntax: @media (condition) {css rules}.
- Can combine mult



Ex:- .box {
height : 100px;
width : 100px;
background-color : lightsalmon
}
@media screen and (width : 250 px) {
border : 5 px solid red;
}}

38. Media Queries (min-width)

Ex: .box {
height : 100 px;
width : 100px;
background-color : lightsalmon;
}
@media screen and (min-width : 300px) {
(max-width : 300 px)
.box {
height : 150 px;
width : 150 px;
}}

Level-7 (Animation, Transition & Transfer)

39. Pseudo Classes
40. Transitions
41. CSS Transform
42. Animation.

⊛ Pseudo Clanes :→

- used to define special states of HTML elements.
- syntax: selector : pseudo-class { styles }.
- Common examples : hover, active, first-child.
  Target elements based on their position or user action.

Ex-:

```
.btn {
    height : 20px;
    width : 70px;
    border : 1px solid blue;
    border-radius : 5px;
    background-color : aliceblue;
}
.btn:hover {
    height : 25 px;
    width : 80 px;
    border : 1px solid red;
}
.btn : active {
    height : 25px;
    width : 80 px;
    border : 1px solid red;
    background-color : indianred;
}
```

* 40. Transitions :→

CSS transition is a property that enables smooth animation b/w changes in CSS property values

- Transition-property : Defines which CSS properties will transition changes in CSS property values
- Transition-duration : Sets how long the transition lasts.
- Transition-timing-function : Controls the speed curve of the transition.
- Transition-delay : Specifies a delay before the transition starts.

Ex:

```
.btn {
    height : 20px;
    width : 70px;
    border : 1px solid blue;
    border-radius : 5px;
    background-color : aliceblue;
    transition-property : all;
    transition-duration : 1s;
    transition-timing-function : ease-in-out;
    transition-delay : 1s;
}
.btn : hover {
    height : 25px;
    width : 80px;
    border : 1px solid red;
}
.btn : active {
    height : 25 px;
    width : 80 px;
    border : 1 px solid red;
    background-color : indianred;
}
```

* 41 (CSS Transform):-

- Allows modification of an elemet's shape and position.
- Can perform operations like rotate, scale & translate.
- Does not affect the layout of sarrounding elemets.
- Used to create visual effect.
- rotate()
- Scale()
- Skew() & translate.

Ex-:
```
.box {
    height : 50 px ;
    width : 50px ;
    padding : 5 px ;
    margin : 20px ;
    border : 1px solid black ;
    border-radius : 5px ;
    transition-property : all ;
    transition-duration : 1s ;
    transition-timing-function : ease-in-out ;
    }
#box1 { background-color : lime ; }
#box1 : hover { transform : rotate (45deg) ; }
#box2 { background-color : blueviolet ; }
#box2 : hover { transform : rotate (180deg) ; }
```

* Scale Property

```
#box1 { background-color : lime ;
#box1 : hover { transform : scale (2) ; }
```

* Translate :—

```
# box1 { background-color : lime ; }
# box1 : hover { transform : translate (50 px) ; }
```

* Skew :—

```
# box1 { bgcolor : lime; }
# box1 . hover { transform : skew (45deg) ; }
```

** 42 (Animation) :→

- Animation-name : Specifies the name of the @keyframes defined animation.
- animation-duration : Defines the total time the animation takes to complete one cycle.
- animation-timing-function : Controls the pacing of the animation (e.g., linear, ease-in).
- animation-delay : sets a delay before the animation starts, allowing for a pause before initiation.
- Animation-iteration-count : Indicates the no. of times the animation should repeat.
- Animation-direction : specifies the dir^n of the animation, allowing for reverse or alternate cycles.

Ex:→

```
.box {
    height = 10 px;
    width = 10 px;
    border = 2 px solid black;
    border-radius : 5 px;
    bgcolor : plum;
    position : relative;
    top ; 50 px;
    left : 50 px;
```

animation-name: ghumakkad;
animation-duration: 2sec;
Animation-timing-function: ease-in-out;
animation-delay: 1sec;
animation-iteration-count: 4;
animation-direction: alternate;

or,

animation: ghumakkad 2s ease-in-out
1sec infinite alternate;
}

@keyframe ghumakkad {
0% from { left: 10px }
50% to { left: 200px }
60% left { 160 px }
100px { left: 200 px }
}

↓

Project

(Myntra clone)