RESEARCH INNOVATION DISCOVERY (RID BHARAT)
A place for coder, developer & Researcher
RUN BY TWKSAA WELFARE FOUNDATION & ISO: 9001:2015

Reg.No: 048884
Foundation Day
05-09-2024

CSS (Cascading Style Sheets)
Shorts Notes

Er. Rajesh Prasad(B.E, M.E)
Founder: TWF & RID Org.

- **RID ORGANIZATION** यानि **Research, Innovation and Discovery** संस्था जिसका मुख्य उदेश्य हैं आने वाले समय में सबसे पहले **NEW (RID, PMS & TLR)** की खोज, प्रकाशन एवं उपयोग भारत की इस पावन धरती से भारतीय संस्कृति, सभ्यता एवं भाषा में ही हो |
- देश, समाज, एवं लोगों की समस्याओं का समाधान **NEW (RID, PMS & TLR)** के माध्यम से किया जाये इसके लिए ही मैं राजेश प्रसाद **इस RID संस्था** की स्थपना किया हूँ|
- Research, Innovation & Discovery में रूचि रखने वाले आप सभी विधार्थियों, शिक्षकों एवं बुधीजिवियो से मैं आवाहनं करता हूँ की आप सभी **इस RID संस्था** से जुड़ें एवं अपने बुध्दि, विवेक एवं प्रतिभा से दुनियां को कुछ नई **(RID, PMS & TLR)** की खोजकर, बनाकर एवं अपनाकर लोगों की समस्याओं का समाधान करें |

त्वक्सा CSS के इस ई-पुस्तक में आप CSS से जुड़ी सभी बुनियादी अवधारणाएँ सीखेंगे। मुझे आशा है कि इस ई-पुस्तक को पढ़ने के बाद आपके ज्ञान में वृद्धि होगी और आपको कंप्यूटर विज्ञान के बारे में और अधिक जानने में रुचि होगी

"In this E-Book of TWKSAA CSS you will learn all the basic concepts related to CSS. I hope after reading this E- Book your knowledge will be improve and you will get more interest to know more thing about computer Science".

RID हमें क्यों करना चाहिए ?

(Research)

अनुसंधान हमें क्यों करना चाहिए ?

Why should we do research?

1. नई ज्ञान की प्राप्ति (Acquisition of new knowledge)
2. समस्याओं का समाधान(To Solving problems)
3. सामाजिक प्रगति (To Social progress)
4. विकास को बढ़ावा देने(To promote development)
5. तकनीकी और व्यापार में उन्नति(To advances in technology & business)
6. देश विज्ञान और प्रौद्योगिकी के विकास(To develop the country's science & technology)

(Innovation)

नवीनीकरण हमें क्यों करना चाहिए ?

Why should we do Innovation?

1. प्रगति के लिए(To progress)
2. परिवर्तन के लिए(For change)
3. उत्पादन में सुधार(To Improvement in production)
4. समाज को लाभ(To Benefit to society)
5. प्रतिस्पर्धा में अग्रणी (To be ahead of competition)
6. देश विज्ञान और प्रौद्योगिकी के विकास। (To develop the country's science & technology)

(Discovery)

खोज हमें क्यों करना चाहिए?

Why should we do Discovery?

1. नए ज्ञान की प्राप्ति(Acquisition of new knowledge)
2. अविष्कारों की खोज(To Discovery of inventions)
3. समस्याओं का समाधान(To Solving problems)
4. ज्ञान के विकास में योगदान(Contribution to development of knowledge)
5. समाज के उन्नति के लिए (for progress of society)
6. देश विज्ञान और तकनीक के विकास (To develop the country's science & technology)

❖ CSS

- **CSS (Cascading Style Sheets)** is used to **style web pages** (HTML/XML).
- It controls **colors, fonts, layout, spacing, backgrounds, animations** etc.
- Used with **HTML and JavaScript** for web designing. Helps designers **apply styles to HTML.**

❖ History of CSS

- **Early Web (1990s):** Only text, no design; HTML handled both content & style.**1996:** CSS1 introduced by W3C (separate content & design).**1998:** CSS2 added positioning & typography.**2000s:** Browser issues but later improved.**2003:** *CSS Zen Garden* showed CSS power. **CSS3:** Brought gradients, animations, responsive design.**Now:** Standardized browsers, frameworks (Bootstrap), preprocessors (SASS).
- **Still evolving** with new features. **Father of CSS:** *Håkon Wium Lie* (with Bert Bos, 1996).

❖ Advantages of CSS

- Separates **design from content**. **Consistent look** across pages.**Faster loading** (cached files, less HTML code). **Responsive design** for all devices.**Reusable styles** (one file, many pages).**SEO friendly** & supports **accessibility**.Easy **maintenance & updates**.Can make **print-friendly pages**.

❖ How to Apply CSS in HTML

➢ Basic CSS Rule

- **Selector** → Which HTML element to style (e.g., p, .class, #id).
- **Property** → What to change (e.g., color, font-size).
- **Value** → The setting for the property (e.g., red, 20px).
- **Syntax:**

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

Example:

```
h3 {
  color: blue;
  font-size: 30px;
  font-weight: bold;
}
```

❖ Ways to Add CSS

1. Inline CSS

- Added inside HTML tag using style attribute.
- Affects only that element.
- **Example:** This is blue text.

2. Internal CSS

- Written inside <style> in <head>.
- Affects the whole page.

```
<!DOCTYPE html>
<html>
<head>
 <style>
  p { color: green; font-size: 18px; }
 </style>
</head>
<body>
 <p>This is green text.</p>
</body></html>
```

3. External CSS

- CSS stored in a separate file (.css).
- Linked using <link> tag in <head>.
- Can be reused in many pages.

raj.css

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

HTML file:

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="css/raj.css">
</head>
<body>
        <p>This is red text.</p>
</body> </html>
```

4. CSS Frameworks

- Pre-made CSS libraries (e.g., **Bootstrap**).
- Just link & use ready classes

Ex: <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

Ex: This is blue text from Bootstrap.

Priority of CSS Styles in HTML

→ When multiple styles apply, CSS follows a **priority order**:

1. Inline CSS (Highest priority)

- Written inside the element using style. **Example:** This text is red.

2. Internal CSS (Second highest.)

- Written inside <style> in <head>.

```
<style>
  p { color: blue; }
</style>
```

3. External CSS (Third priority.)

- Written in a separate .css file and linked with <link>.

  Example: styles.css

```
p { color: green; }
<link rel="stylesheet" href="styles.css">
```

4. Specificity

- More **specific selectors** override general ones.

  p.my-class { color: purple; } → higher priority

  p { color: yellow; } → lower priority

5. Order of Styles

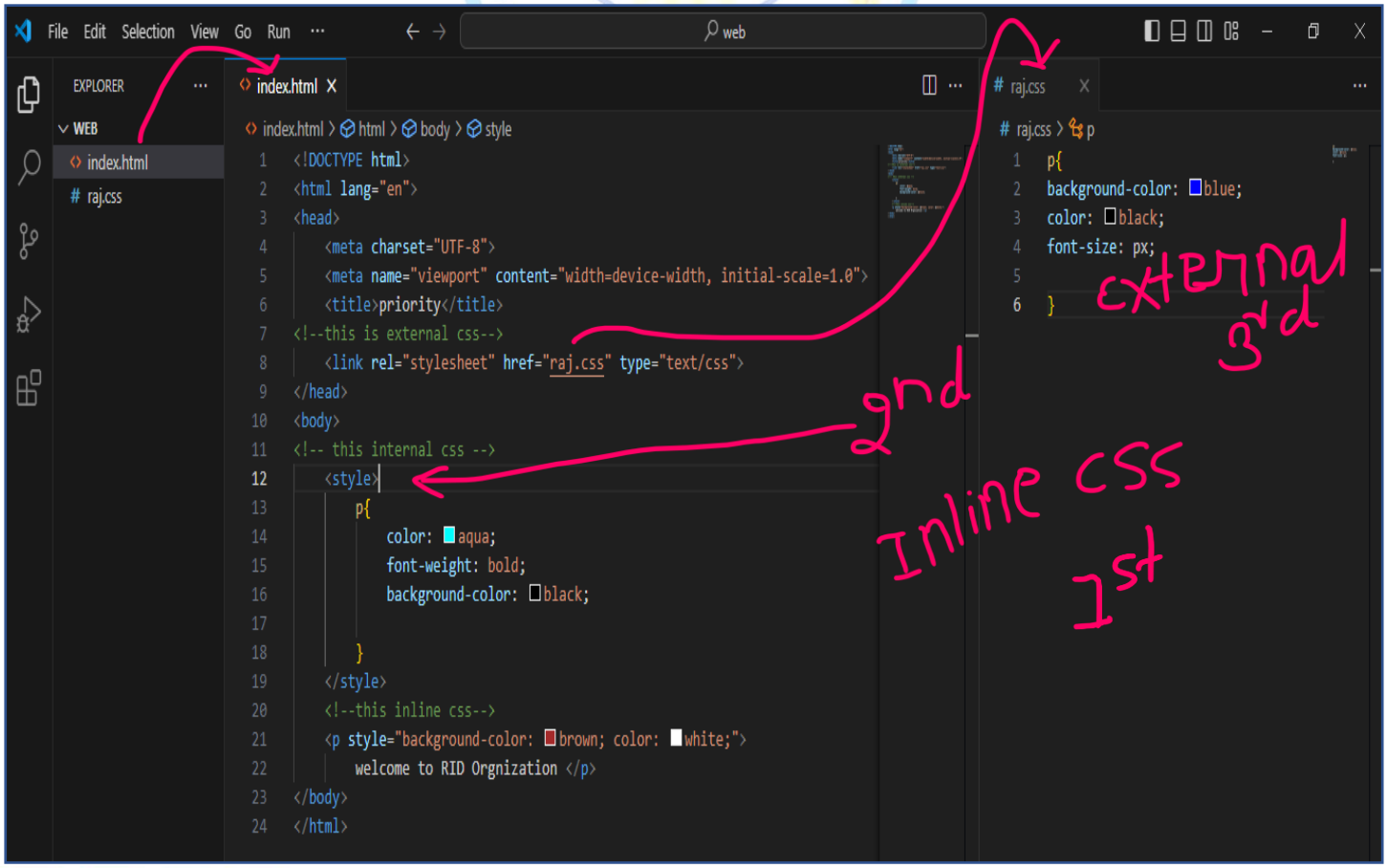
- If rules are same, **last one wins**.

  p { color: orange; }

  p { color: pink; } → This will apply

6.! Important :- Overrides everything, even inline styles. **Example: -** p { color: brown !important; }

Final Priority Order: → !important > Inline > Internal > External > Specificity > Order



CSS Selector with Priority

Low → High "Universal → Type → Class/Attr/Pseudo → ID → Inline →! important"

1) **Universal Selector (*)**
2) **Type Selector (p, h1)**
3) **Class / Attribute / Pseudo-class (.class_name, [attr], :hover)**
4) **ID Selector (#id)**
5) **Inline CSS (style="")**
6) **! important**

Selector	Example	Meaning	Priority Level
Universal	* {margin:0;}	Selects all elements	Lowest
Type (Element)	p {color:blue;}	Selects all tags	Low
Grouping	h1, h2, h3 {font-family:Arial;}	Groups multiple selectors	Same as their type/class/ID
Class	.highlight {background:yellow;}	Selects elements with a class	Medium
Attribute (Exact)	[data-type="btn"] {color:red;}	Match attribute value	Medium
Attribute (Prefix)	[href^="https"] {color:blue;}	Attribute starts with value	Medium
Attribute (Substring)	[class*="btn"] {background:gray;}	Attribute contains value	Medium
Attribute (Suffix)	[src$=".jpg"] {border:1px solid black;}	Attribute ends with value	Medium
Pseudo-class	a:hover {text-decoration:underline;}	State/position (hover, first-child etc.)	Medium
Pseudo-element	p::before {content:"Note:";}	Selects part of element	Medium
Descendant	nav ul {list-style:square;}	All inside another	Medium
Child	.menu > li {font-weight:bold;}	Direct child only	Medium
Adjacent Sibling	h2 + p {margin-top:10px;}	Next sibling	Medium
General Sibling	h3 ~ p {color:gray;}	All following siblings	Medium
ID	#header {font-size:24px;}	Unique element	High
Not Selector	p:not(.special) {font-style:italic;}	Exclude elements	Same as class
Inline CSS	Text	Style inside element	Very High
!important	p {color:brown !important;}	Overrides everything	Highest

Example: input[type="text"] { border:2px solid red; color:red; padding:5px; }
Example: input[type="password"] { border:2px solid blue; color:blue; padding:5px; }
Example: input[type="email"] { border:2px solid green; color:green; padding:5px; }
Example: input[type="number"] { border:2px solid orange; color:orange; padding:5px; }
Example: input[type="button"] { border:2px solid purple; color:purple; padding:5px; }
Example: input[type="submit"] { border:2px solid brown; color:brown; padding:5px; }
Example: input[type="radio"] { accent-color:red; }

CSS Text Properties

1. text-align: Used to set horizontal alignment of text.

- **Syntax:** text-align: left | right | center | justify;
- left → Align left
- right → Align right
- center → Center text
- justify → Spread evenly
- **Example:** Centered text

2. text-decoration: Used to add/remove lines on text

- none → No line
- underline → Line below
- overline → Line above
- line-through → Line in middle
- **Syntax:** text-decoration: none | underline | overline | line-through;
- **Example-1:** Underlined

*** skills is very important *

3. text-transform: Used to change text capitalization style

- capitalize → First letter capital
- uppercase → ALL CAPITALS
- lowercase → all small
- **Syntax:** text-transform: none | capitalize | uppercase | lowercase;
- **Example:** BIG TEXT

4. text-overflow: Used to handle overflowed text inside container

- clip → Cut text
- ellipsis → Show ...
- **Syntax:** text-overflow: clip | ellipsis;
- **Example:**

```
<p style="width:100px; white-space:nowrap; overflow:hidden; text-overflow:ellipsis;">
  This is a long text
</p>
```

overflow → Controls **whether content is shown, hidden, or scrollable**.
text-overflow → Decides **how hidden text looks** (just cut or with ...).

```
selector { overflow: visible;  /* Default → content goes outside */
  overflow: hidden;   /* Extra content is cut off */
  overflow: scroll;   /* Always show scrollbars */
  overflow: auto;     /* Show scrollbars only if needed */ }
```

5. text-shadow: Used to add shadow effect to text

- Adds shadow to text.
- **Syntax:** text-shadow: h-shadow v-shadow blur-radius color;
- **Example:** Shadow text

6. text-indent: Used to indent (shift) first line of a paragraph

- Indent first line of text.
- **Syntax:** text-indent: length
- **Example:** Indented text

7. text-wrap (white-space): Used to control line wrapping of text

- **Syntax: white-space: normal | nowrap;**
- normal → Wrap normally
- nowrap → No wrapping
- **Syntax:** white-space: value;
- **Example:** This text will not wrap

TEXT FORMATTING

1. font-family: Used to set font style. Example: Arial, Verdana, Times New Roman

• **Syntax:** font-family: font-name, fallback; **Example:** Hello

2. font-size: Used to set text size. Example: px, em, %

• **Syntax:** font-size: value; **Example:** Big Text

3. font-weight: Used to set thickness of text. Example: normal, bold

• **Syntax:** font-weight: value; **Example:** Bold Text

4. font-style: Used to set text italic or normal. Example: normal, italic

• **Syntax:** font-style: value; **Example:** Italic Text

5. font-variant: Used to display text in small-caps. Example: small-caps

• **Syntax:** font-variant: value; **Example:** Hello

6. color: Used to set text color. Example: red, blue, #FF0000

• **Syntax:** color: value; Example: Red Text

7. text-align: Used to align text horizontally. Example: left, right, center, justify

• **Syntax:** text-align: value; Example: Centered Text

8. text-decoration: Used to add or remove text decoration. Example: underline, overline, line-through

• **Syntax**: text-decoration: value; Example: Underlined

9. text-decoration-line: Used to define type of decoration. Example: underline, overline, line-through

• **Syntax:** text-decoration-line: value; Example: Overlined

10. text-decoration-color: Used to set decoration color. Example: blue, green

• **Syntax:** text-decoration-color: value;

Example: Blue Underline

11. text-decoration-style: Used to set decoration style. Example: solid, dotted, dashed

• **Syntax:** text-decoration-style: value;

• Example: Dashed Underline

12. text-transform: Used to change text case. Example: uppercase, lowercase, capitalize

• **Syntax:** text-transform: value; Example: hello

13. line-height: Used to set space between lines. Example: 1.5, 2

• **Syntax:** line-height: value; Example: Line Spacing

14. letter-spacing: Used to set space between letters. Example: 2px, 5px

• **Syntax:** letter-spacing: value; Example: S P A C E

15. word-spacing: Used to set space between words. Example: 4px, 10px

• **Syntax:** word-spacing: value; Example: Hello World

16. text-indent: Used to indent first line of text. Example: 20px, 30px

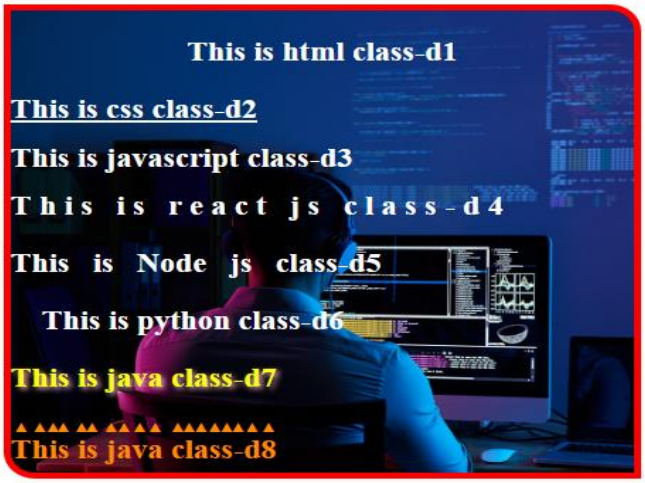
• **Syntax:** text-indent: value; Example: Indented Line

17. text-shadow: Used to add shadow to text. Example: 2px 2px 5px blue

• **Syntax:** text-shadow: h-shadow v-shadow blur color; Example: Shadow Text

```
<style>  .c1{
          width: 400px; height: 57vh; border: 5px solid red;
          border-radius: 0PX 20PX; background-image: url('img.jpg');
         background-size: cover;color: white; font-weight: bold font-size: 20px;}
       #d1{ text-align: center;}
       #d2{ text-decoration: underline;}
       #d3{ line-height: 10px;}
       #d4{ letter-spacing: 5px;}
       #d5{ word-spacing: 10px;}
       #d6{ text-indent: 20px;}
       #d7{ text-shadow:  2px 2px 4px pink; color: yellow;}
       #d8{ text-emphasis: triangle; color: darkorange;}
       h2{ margin-left: 70px;
          line-height: 0px;
          color: darkblue;}</style>
```

CSS TEXT FORMATTING



Background Properties in css

1. background-color: Used to set background color.

- **Syntax:** background-color: color;
- **Example:** .box { background-color: lightblue; }

2. background-image: Used to set an image as background.

- **Syntax:** background-image: url('image.jpg');
- **Example:** .box { background-image: url('bg.jpg'); }

3. background-repeat: Controls if/how background image repeats.

- **Values:** repeat | repeat-x | repeat-y | no-repeat | space | round
- **Example:** .box { background-repeat: no-repeat; }

4. background-position: Sets starting position of background image.

- **Values:** left, right, top, bottom, center OR x y (e.g., 50px 100px)
- **Example:** .box { background-position: center top; }

5. background-size: Sets size of background image.

- **Values:** auto | cover | contain | width height
- **Example:** .box { background-size: cover; }

6. background-attachment: Controls if background scrolls or stays fixed.

- **Values:** scroll | fixed | local
- **Example:** .box { background-attachment: fixed; }

7. background-origin: Defines where background image starts.

- **Values:** padding-box | border-box | content-box
- **Example:** .box { background-origin: padding-box; }

8. background-clip: Defines how far background color extends.

- **Values:** border-box | padding-box | content-box | text
- **Example:** .box { background-clip: content-box; }

9. background-blend-mode: Blends multiple backgrounds (image + color).

- **Values:** normal | multiply | screen | overlay | darken | lighten ...
- **Ex:** .box { background-image: url('bg.jpg');
  background-color: lightblue;
  background-blend-mode: multiply;}
- **Example**

```
.box { width: 300px;height: 150px;
 background-image: linear-gradient(to right, red, yellow),url("bg.jpg");
  background-blend-mode: overlay; /* Blend gradient with image */}
```

10.-webkit-background-clip:

❖ **webkit-background-clip:** text is a CSS property that allows you to control the clipping of background images or colors to the text of an element. When applied, it makes the background of the text transparent

Why Use -webkit-background-clip: text? Visual Effects, **Typography Enhancement** and **Flexibility**:

❖ **How to Use -webkit-background-clip: text**

1. **Basic Syntax**: this is typically used alongside background-image, background-color, or background-gradient and requires the text color to be transparent for the effect to be visible.

Example: .text {

```
color: transparent;  /* Hide the text color */
background-image: linear-gradient(to right, #FF5733, #33FF57); /* Apply a gradient */
-webkit-background-clip: text; /* Clip the background to the text */
background-clip: text;  /* Standard property (not widely supported) */}
```

CSS Colors

1.Ways to define color:

- **Named Colors** → color: red;
- **Hex Code** → color: #FF5733;
- **RGB** → color: rgb(255, 0, 0);
- **RGBA (with transparency)** → color: rgba(255, 0, 0, 0.5);
- **HSL** → color: hsl(120, 100%, 50%);
- **HSLA (with transparency)** → color: hsla(120, 100%, 50%, 0.5);

2.Where colors are used:

- color: → Text color
- background-color: → Background
- border-color: → Border
- box-shadow: → Shadow
- text-shadow: → Text shadow
- outline-color: → Outline

3. **Opacity & Transparency:**
   - opacity: 0.5; → 50% transparent element
   - rgba() / hsla() → Transparency in colors
4. **Gradients:**
   - Linear gradient → background: linear-gradient(red, blue);
   - Radial gradient → background: radial-gradient(circle, red, blue);

Examples

```
p { color: blue; }                      /* Text color */
div { background-color: #ff5733; }      /* Background */
.box { border: 2px solid green; }       /* Border color */
h1 { text-shadow: 2px 2px 5px gray; }   /* Text shadow */
```

❖ CSS Color Examples

1. Text Color (Named Color) **Example** `<p style="color: crimson;">This is crimson text.</p>`

2. Background Color (Named Color)

Example: `<div style="background-color: gold; color:black;">Gold background</div>`

3. Border Color Example: `<div style="border:2px solid seagreen;">Seagreen border</div>`

4. Box Shadow Color

```
<div style="width:200px;height:100px;box-shadow:5px 5px 10px darkgray;"> Shadow with darkgray</div>
```

5. Outline Color Example: `<div style="outline:2px dashed royalblue;">Royalblue outline</div>`

❖ Gradients

6. Linear Gradient

```
<div style="width:200px;height:100px;
background:linear-gradient(to bottom,#3498db,#e74c3c);">Linear Gradient</div>
```

7. Radial Gradient

```
<div style="width:200px;height:200px;
background:radial-gradient(circle,#2ecc71,#e74c3c);">Radial Gradient</div>
```

8. Multiple Color Stops

```
<div style="width:200px;height:100px;
background:linear-gradient(to right,#3498db,#e74c3c,#9b59b6);">Multi-Stop Gradient
</div>
```

❖ Color Functions

9. RGB `<div style="background-color:rgb(255,0,0);color:white;">RGB Red</div>`

10. RGBA (Transparency)

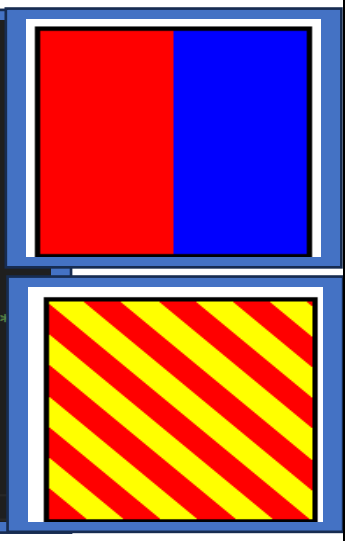
```
<div style="background-color:rgba(0,128,0,0.5);color:white;">Semi-Transparent Green</div>
```

11. HSL Example: `<p style="color:hsl(210,100%,50%);">HSL Blue Text</p>`

12. HSLA (Transparency) Example: `<p style="color:hsla(45,100%,50%,0.6);">HSLA Orange Text</p>`

Example-1: How to apply multiple color in background.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
    #d1{width: 100px;
        height: 100px;
        border: 2px solid black;
    /* background: linear-gradient(to bottom, red 50%, yellow 50%); */
     /* background:  linear-gradient(to right, red 50%, blue 50%),
                     linear-gradient(to bottom, yellow 50%, green 50%); */
       background-image: repeating-linear-gradient(
        45deg, red, red 10px, yellow 10px, yellow 20px);}
    </style>
</head>
<body>
    <div id="d1"></div>
</body></html>
```



Example-2: How to apply multiple image in background.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Multiple Images in One Div</title>
  <style>
    #d1 {
      width: 300px;
      height: 200px;
      border: 2px solid black;
      /* Multiple images + gradient */
      background:
        url(img1.png) no-repeat left top,
        url(rpsir.jpg) no-repeat right bottom,
        linear-gradient(to bottom, yellow, green);
      background-size: 80px 80px, 100px 100px, cover;
    }
  </style>
</head>
<body>
  <div id="d1"></div>
</body>
</html>
```

Example-3: How to apply multiple images in background.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Multiple Colors in One Div</title>
  <style>
    #d1 {
      width: 300px;
      height: 200px;
      border: 2px solid black;
      color: white;
      text-align: center;
      /* Multiple backgrounds color image :  */
      background: linear-gradient(to right, red, yellow),
                  linear-gradient(to bottom, blue, lightgreen),
                  url("rpsir.jpg");
                  background-blend-mode: multiply;
                  background-size: cover;
    }
    p{position: relative;top: 170px;}
  </style>
</head>
<body>
  <div id="d1"><p>Er.Rajesh Prasad</p></div>
</body></html>
```

Example-4: How to apply multiple color in background of each letter.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Gradient Letters</title>
  <style>
    h1 {
      font-size: 90px;
      font-family: 'Times New Roman', Times, serif;
      font-weight: bold;
      margin: 0;
      display: inline-block;
      background: linear-gradient(to right, red, orange, yellow,
       green, blue, indigo, violet);
      -webkit-background-clip: text;
      color: transparent;
    }
  </style>
</head>
<body>
  <h1>RID BHARAT</h1>
</body>
```

- **-webkit-background-clip: text**
  It makes the **background (color, gradient, image)** show **inside the text** instead of behind it.
  You must also set color: transparent; to hide normal text color.
- **Syntax:** selector {
  background: linear-gradient(to right, red, blue);
  -webkit-background-clip: text; color: transparent;}

RID BHARAT

Ex-5: How to apply multiple color in background of each r individual letter.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>One Color Per Letter</title>
  <style>
    span {
      font-size: 90px;
      font-family: 'Times New Roman', Times, serif;
      font-weight: bold;
      margin: 0 2px;
    }
  </style>
</head>
<body>
  <div>
    <span style="color: red;">R</span>
    <span style="color: orange;">I</span>
    <span style="color: yellow;">D</span>
    <span style="color: green;">B</span>
    <span style="color: blue;">H</span>
    <span style="color: indigo;">A</span>
    <span style="color: violet;">R</span>
    <span style="color: pink;">A</span>
    <span style="color: cyan;">T</span>
  </div>
</body>
</html>
```

-webkit-background-clip: text; color: transparent; only makes sense when you use a gradient or image as a background.
For solid colors per letter, you don't even need background-clip. Just set color directly.

RIDBHARAT

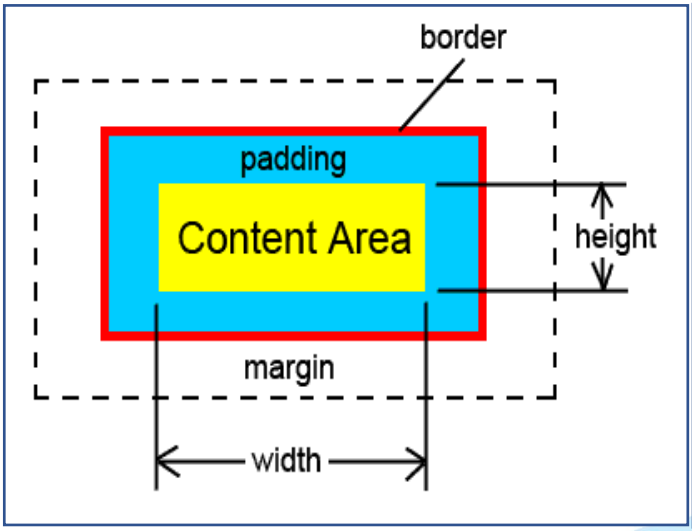
Ex-6: How to apply multiple image in background of each letter in word.

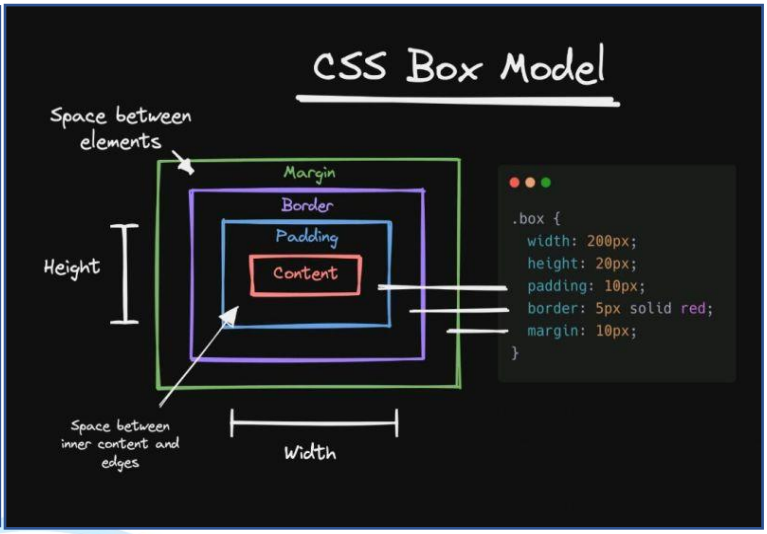
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Letters with Background Images</title>
    <style>
        h1, span { display: inline;
            background-image: url(image.png);
            background-size: cover;
            -webkit-background-clip: text;
            color: transparent; /* Hide the text color */
            font-size: 70px;
            font-family: 'Times New Roman', Times, serif;
            font-weight: bold;
        }
        .spacer {  margin-left: 30px; /* Adjust the space
    </style></head> <body>
    <h1>R</h1>
    <h1>I</h1>
    <h1>D</h1>
    <span class="spacer"></span> <!-- Spacer between RID a
    <span>B</span>
    <span>H</span>
    <span>A</span>
    <span>R</span>
    <span>A</span>
    <span>T</span>
</body></html>
```

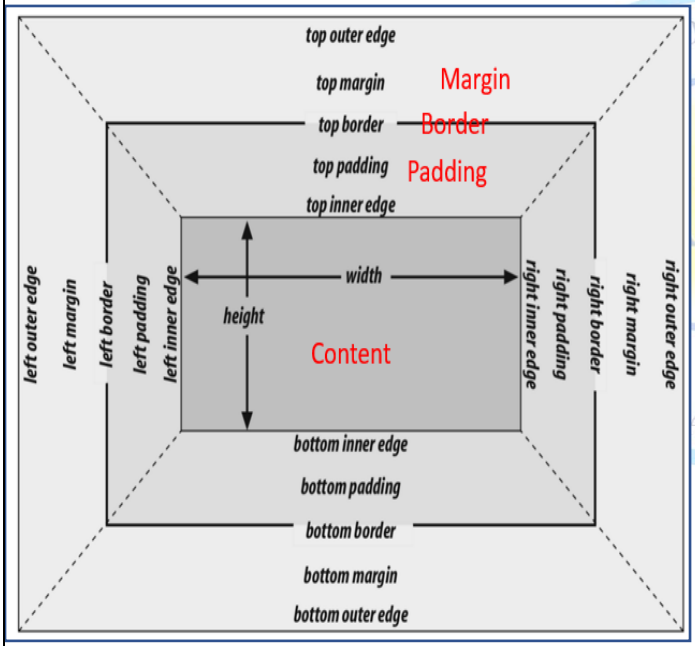
RID BHARAT

CSS BOX MODEL

- The CSS Box Model is a fundamental concept in web design and layout. It describes how elements on a web page are structured in terms of content, padding, borders, and margins.







Content Area

- The innermost part of an element.
- Contains actual content such as text, images, or other ele.
- Size is defined by the element's width and height.

Padding

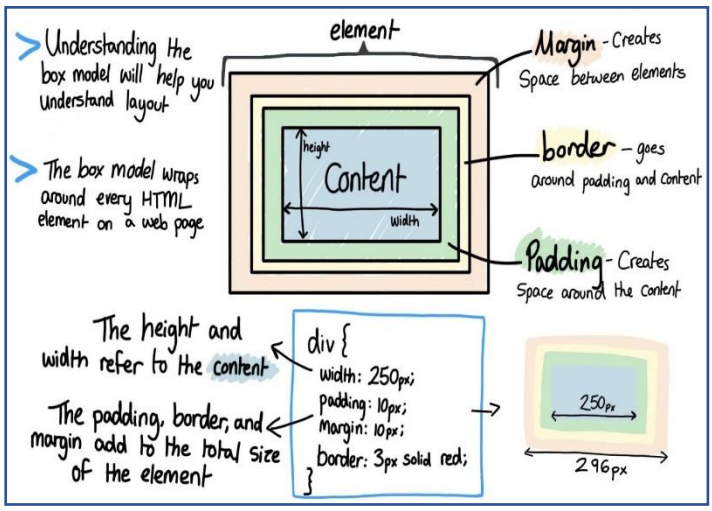
- The space between the content area and the border.
- Controlled by the padding property.
- Used to create spacing inside the element, making content look clear and readable.

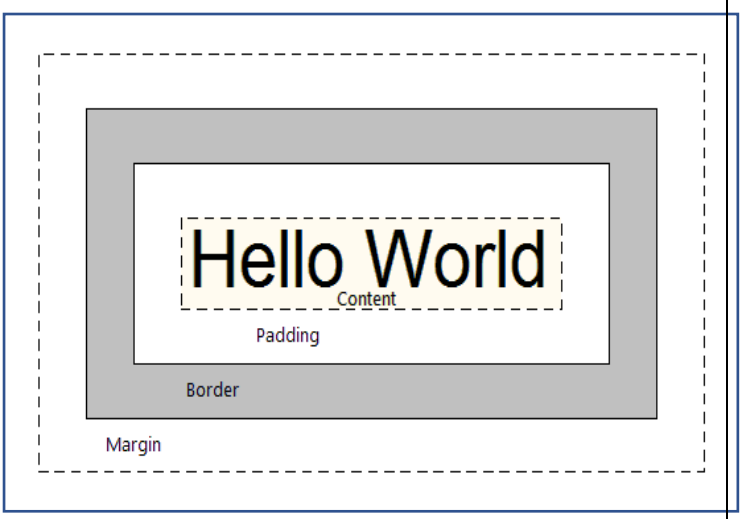
Border

- A line that wraps around the content and padding.
- Controlled by the border property (style, color, thickness).
- Often used to separate or highlight elements.

Margin

- The space outside the border of an element.
- Controlled by the margin property.
- Provides spacing between elements on a webpage.





CSS Padding

Padding = space between content & border.

- **Shorthand**
  - padding: 10px; → **all sides**
  - padding: 10px 20px; → **top/bottom = 10, left/right = 20**
  - padding: 10px 20px 30px; → **top = 10, left/right = 20, bottom = 30**
  - padding: 10px 20px 30px 40px; → **top, right, bottom, left**
- **Individual: -** padding-top: 20px; padding-right: 15px; padding-bottom: 25px; padding-left: 30px;
- **Units (**px = fixed**,** % = relative to parent em/rem = relative to font size)

Example: Padding Example

Unit Use CaseRecommendation

- px→ Fixed, exact spacingGood for small elements (buttons, cards)
- % --> Relative to parent sizeGood for responsive layouts (containers, sections)
- Rem→ Relative to root font sizeBest for consistent & scalable padding across the whole site

CSS Border

- **Border** = line around element's **content + padding**.
- **Shorthand**: border: width style color; **Example:** border: 2px solid black;

Border Width:- Sets thickness: px, em, rem, or keywords → thin, medium, thick.

Example:

- border-width: 5px;
- border-width: 5px 10px; **/* top-bottom | left-right */**
- border-width: 5px 10px 15px; **/* top | left-right | bottom */**
- border-width: 5px 10px 15px 20px; **/* top | right | bottom | left */**

Border Style Values: none, dotted, dashed, solid, double, groove, ridge, inset, outset.

- **Example:** border-style: dashed;

Border Color Set with names, HEX, RGB, etc.

- **Ex:** border-color: red; border-color: red green blue yellow; /* top | right | bottom | left */

Individual Borders → border-top: 2px solid red; border-right: 3px dotted green;
border-bottom: 4px dashed blue; border-left: 5px double orange;

CSS MARGIN

- Margin is the **space outside the border** of an element. It creates **gap/spacing between elements**. Controlled with the **margin property**. **Syntax** selector { margin: value; }

Types of Margin Values

1. **Single Value** – All four sides same. Example: margin: 20px; /* 20px on all sides */
2. **Two Values** – First: top/bottom, Second: left/right. margin: 10px 20px; 10px top-bottom, 20px left-right
3. **Three Values** – Top, left/right, bottom. margin: 5px 10px 15px;
4. **Four Values** – Top, right, bottom, left. margin: 5px 10px 15px 20px;

Special Cases

- **Negative Margin** → Pulls elements closer/overlaps.
- **Auto Margin** → Used for centering.
- **Collapsing Margins** → Adjacent margins may merge into one.

Box Model Reminder Total Size = Content + Padding + Border + Margin Example for Width:

Total Width = width + left padding + right padding + left border + right border + left margin + right margin

- Margins = outer space.
- Padding = inner space.
- Borders = between them.

CSS BOX-SIZING

- The box-sizing property controls **how width and height are calculated**.
- It decides whether **padding and border** are included in the total size of an element or not.

Values of box-sizing

1. **content-box (default)**
   - Width/height apply **only to content**.
   - Padding and border are added outside. **Example:** .box {box-sizing: content-box; }
2. **border-box**
   - Width/height include **content + padding + border**.
   - Easier to manage layouts. **Example:** box {box-sizing: border-box;}

Example. box { width: 200px; height: 100px; padding: 20px; border: 5px solid red;
box-sizing: border-box; /* Total size = 200x100 */}

- With content-box, total size would be **more than 200x100** (extra padding + border).
- With border-box, total size stays **200x100** (everything included).

CSS BORDER-RADIUS

- The border-radius property makes **rounded corners** of an element (box, button, image, etc.).

Corner Properties

1. border-top-left-radius → Top-left corner
2. border-top-right-radius → Top-right corner
3. border-bottom-right-radius → Bottom-right corner
4. border-bottom-left-radius → Bottom-left corner

Syntax border-radius: value1 value2 value3 value4;

- **value1** = top-left
- **value2** = top-right
- **value3** = bottom-right
- **value4** = bottom-left

Example: border-radius: 10px 20px 30px 40px;

- You can also use **/** to set different horizontal and vertical radii.
- border-radius: 10px 20px 30px 40px / 5px 15px 25px 35px;

Examples

```
.box1 {border-radius: 20px;   /* all corners same */}
.box2 {border-radius: 0px 40px 0px 40px; /* alternate corners rounded */}
.box3 {border-radius: 50%;   /* perfect circle if width=height */}
```

Note: border-radius makes corners round.

- You can set **one value** (all sides same) or **up to four values** (each corner separately).

CSS LAYOUT – OVERFLOW

The **overflow** property controls what happens when content is **too large** to fit inside an element.

Values of overflow

1. **visible (default)** → Content is not clipped, it overflows outside the box.
2. **hidden** → Extra content is clipped (not visible).
3. **scroll** → Extra content is clipped, and **scrollbars always appear**.
4. **auto** → Scrollbars appear **only when needed**.

Example
```
div { width: 200px;  height: 100px; border: 1px solid black;
  overflow: auto; /* Change to visible, hidden, scroll, auto */
}
```

Note: Works only on block elements with a fixed height/width

POSITION PROPERTIES

By default, all html tag is static so chaining the static to dynamic we are using the position properties Used to **control placement of elements** on a web page.

1. position: - Specifies **how an element is positioned**.

Values:

- **static** → Default, normal flow, ignores top/right/bottom/left.
- **relative** → Positioned **relative to normal position**. Can use top/right/bottom/left.
- **absolute** → Positioned **relative to nearest positioned ancestor**.
- **fixed** → Positioned **relative to viewport**, stays when scrolling.
- **sticky** → Acts like relative until **scroll reaches offset**, then behaves like fixed.

2. top, right, bottom, left

- Used to **offset positioned elements**. Values: px, %, em, etc.

3. z-index

- Controls **stacking order**. Higher value → appears **on top**.

4. float

- Aligns elements **horizontally** inside a container. Values: left, right, none.

Table

Position	Moves with scroll?	Relative to	Offsets work?
static	Yes	Normal flow	No
relative	Yes	Normal position	Yes
absolute	No	Nearest ancestor	Yes
fixed	No	Viewport	Yes
sticky	Yes	Normal flow until offset	Yes

CSS DISPLAY PROPERTY

The **display** property controls **how an element is shown** and its **layout behavior**.

Common Values

Value	Description
block	Starts on a new line, full width (e.g., , <p>).
inline	Stays in line, width = content only (e.g., , <a>).
inline-block	Inline but allows block styling (width, height, padding).
none	Hidden, takes **no space**.
flex	Makes element a **flex container** for arranging children.
inline-flex	Same as flex, but behaves inline.
grid	Creates a **grid container** for layout.
inline-grid	Same as grid, but behaves inline.
table	Behaves like a table element.
table-row	Behaves like a table row.
table-cell	Behaves like a table cell.
list-item	Behaves like a list item ( ).
initial	Sets default value.
inherit	Inherits from parent element.

Examples

Hides element	.none { display: none; }
Inline elements	.inline { display: inline; }
Block elements	.block { display: block; }
Inline-block elements	.inline-block { display: inline-block; }

Flex container

```
.flex-container {
  display: flex;
  flex-direction: row;          row or column
  justify-content: center;      align along main axis
  align-items: center;          align along cross axis
  gap: 10px;                    spacing between items
}
.flex-item {
  flex: 1;                      grow to fill space
  order: 2;                     change order
}
```

Flexbox Key Points

- **Flex Container** → Parent with display: flex.
- **Flex Items** → Children inside flex container.
- Properties for container: flex-direction, justify-content, align-items, flex-wrap, gap.
- Properties for items: order, flex-grow, flex-shrink, flex-basis, align-self.

Note:

- display decides **how element behaves** in layout.
- flex and grid are modern layout models for **responsive design**.
- none hides elements without space.
- block and inline are basic layout types.

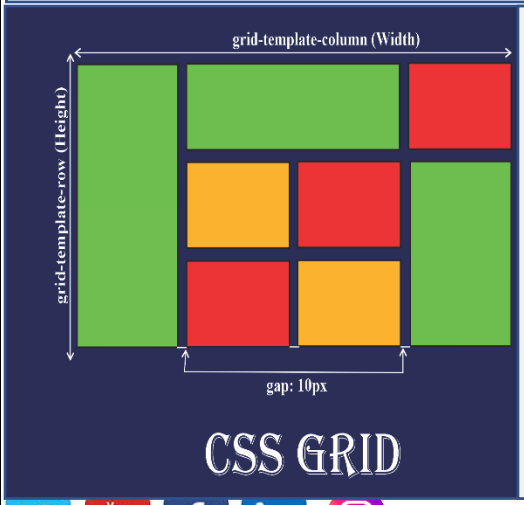
DISPLAY GRID

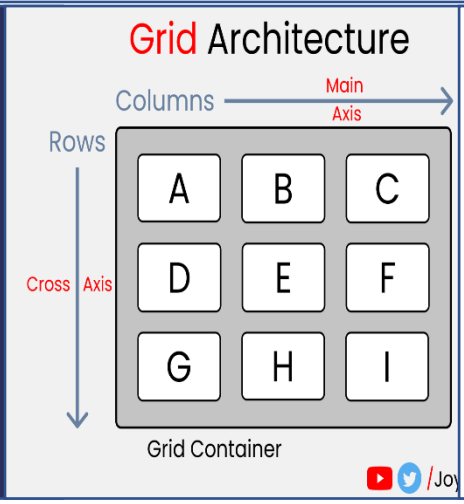
- CSS Grid, is a CSS module that provides a two-dimensional grid-based layout system, optimized for responsive design. It enables the creation of web layouts that are flexible, easy to manage, and visually consistent.

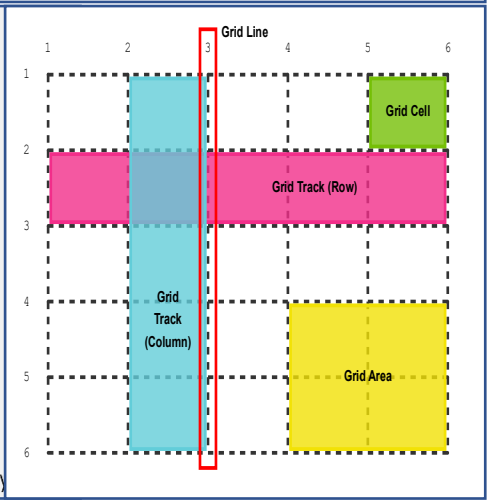
❖ Key Features:

- **Two-dimensional Layouts:** Unlike CSS Flexbox which is primarily one-dimensional, CSS Grid can handle both rows and columns simultaneously.
- **Grid Container and Grid Items:** A grid is made up of a grid container and its child elements, called grid items.
- **Tracks and Gaps:** Grid tracks (rows and columns) and gaps (gutters) can be defined to create the grid structure.
- **Flexible Sizing:** Supports flexible sizing with units such as fractions (fr), pixels (px), percentages (%), and more.
- **Placement and Alignment:** Grid items can be placed and aligned with precision using grid lines, spans, and areas.
- **Responsive Design:** It makes creating responsive layouts simpler through media queries and intrinsic sizing.

1) **grid:** It is shorthand for setting all the grid-related properties (grid-template-rows, grid-template-columns, grid-template-areas, grid-auto-rows, grid-auto-columns, grid-auto-flow, grid-gap, and grid).
2) **grid-template-rows:** Defines the size of the rows in a grid container.
3) **grid-template-columns:** Defines the size of the columns in a grid container.
4) **grid-template-areas:** Defines named grid areas.
5) **grid-template:** Shorthand for grid-template-rows, grid-template-columns, and grid-template-areas.
6) **grid-auto-rows:** Sets the size of automatically generated rows in a grid container.
7) **grid-auto-columns:** Sets the size of automatically generated columns in a grid container.
8) **grid-auto-flow:** Specifies how auto-placed items are positioned in the grid container.
9) **grid-row-gap:** Specifies the size of the gap between rows in a grid container.
10) **grid-column-gap:** Specifies the size of the gap between columns in a grid container.
11) **grid-gap:** Shorthand for grid-row-gap and grid-column-gap.
12) **justify-items:** Aligns grid items along the inline (row) axis.
13) **align-items:** Aligns grid items along the block (column) axis.
14) **justify-content:** Aligns grid items along the inline (row) axis of the grid container.
15) **align-content:** Aligns grid items along the block (column) axis of the grid container.
16) **grid-area:** Either specifies a name for the grid item, or it specifies the area within the grid to place the item.







Example-1

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>grid</title>
   <style>
      .container{
width: 800px;
height: 80vh;
background-color: blue;
display: grid;
gap: 10px;
grid-template-columns: 100px 100px 100px;
grid-template-rows: 100px 100px 100px; */
/* grid-template-columns: repeat(3, 1fr);
 grid-template-rows: repeat(3, 1fr); */
grid-template-columns: 1fr 1fr 1fr;
grid-template-rows: 1fr 1fr 1fr;
row-gap: 10px;column-gap: 30px;
      }
      .c{
         /* width: 100px;
         height: 100px; */
         background-color: brown;
      }
      .c1{
         grid-column-start: 1;
         grid-column-end: 3; }
      .c3{
         grid-row-start: 2;
         grid-row-end: 3;
         background-color: black;
         color: white;
         text-align: center; }
   </style>
</head>
<body>
   <div class="container">
      <div class="c c1">items1</div>
      <div class="c c2">items2</div>
      <div class="c c3">items3</div>
      <div class="c c4">items4</div>
      <div class="c c5">items5</div>
      <div class="c c6">items6</div>
      <div class="c c7">items7</div>
   </div>
</body>
</html
```

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
   <style>
      .m{
         display: grid;
         grid-template-columns: 200px 200px 200px 200px;
         /* grid-template-columns: repeat(4, 300px);
         grid-template-columns: 1fr 1fr 1fr 1fr ; */
         /* grid-template-columns: repeat(auto-fill,
minmax(400px, 1fr)); */
         /* grid-template-rows: auto; */
         grid-template-rows: 200px 200px 200px 200px;
         /* grid-template-rows: repeat(5, 300px); */
         row-gap: 50px;
         column-gap: 30px;
         /* grid-auto-columns: 200px; */
         grid-auto-rows: minmax(100px, auto);
         grid-auto-columns: minmax(300px, auto);
         grid-template-areas:  "box1 box1 box1"
                               "box2 box2 box3";
         /* gap: 10px; */ }
      .c{  border: 5px solid red;
         border-radius: 10px; }
      .c1{ /* grid-column-start: 1;
         grid-column-end: 5;
         grid-auto-rows: 3; */
         grid-area: box1; }
      .c2{ /* grid-column-start: 2;
         grid-column-end: 4;
         grid-row-start: 1;
         grid-row-end: 4; */
         grid-area: box2; }
      .c3{ grid-area: box3;}
      .c4{ grid-area: box4; }
      .c5{grid-area: box5;}
      .c6{ grid-area: box6;} </style> </head> <body>
   <div class="m">
   <div class="c c1">box 1
      <img src="img1.jpg"> </div>
   <div class="c c2">box 2  lorem100 </div>
   <div class="c c3">box 3</div>
   <div class="c c4">box 4</div>
   <div class="c c5">box 5</div>
   <div class="c c6">box 6</div>
   </div>
</body> </html>
```

CSS CURSOR PROPERTIES

- These are some of the most common and important cursor properties you can use in CSS. Each property changes the cursor's appearance when it hovers over an element to indicate different kinds of actions or states.

Icon	Values	Examples
	default	style="cursor: default;"
	hand	style="cursor: hand;"
	pointer	style="cursor: pointer;"
	crosshair	style="cursor: crosshair;"
	text	style="cursor: text;"
	wait	style="cursor: wait;"
	help	style="cursor: help;"
	move	style="cursor: move;"
	e-resize	style="cursor: e-resize;"
	ne-resize	style="cursor: ne-resize;"
	nw-resize	style="cursor: nw-resize;"
	n-resize	style="cursor: n-resize;"
	se-resize	style="cursor: se-resize;"
	sw-resize	style="cursor: sw-resize;"
	s-resize	style="cursor: s-resize;"
	w-resize	style="cursor: w-resize;"
	progress	style="cursor: progress;"
	all-scroll	style="cursor: all-scroll;"
	col-resize	style="cursor: col-resize;"
	no-drop	style="cursor: no-drop;"
	not-allowed	style="cursor: not-allowed;"
	row-resize	style="cursor: row-resize;"
	vertical-text	style="cursor: vertical-text;"

Name values

1. default: cursor: default;
2. hand: cursor: hand;
3. pointer: cursor: pointer;
4. crosshair: cursor: crosshair;
5. text: cursor: text;
6. wait: cursor: wait;
7. help: cursor: help;
8. move: cursor: move;
9. e-resize: cursor: e-resize;
10. ne-resize: cursor: ne-resize;
11. nw-resize: cursor: nw-resize;
12. n-resize: cursor: n-resize;
13. se-resize: cursor: se-resize;
14. sw-resize: cursor: sw-resize;
15. s-resize: cursor: s-resize;
16. w-resize: cursor: w-resize;
17. progress: cursor: progress;
18. all-scroll: cursor: all-scroll;
19. col-resize: cursor: col-resize;
20. no-drop: cursor: no-drop;
21. not-allowed: cursor: not-allowed;
22. row-resize: cursor: row-resize;
23. vertical-text: cursor: vertical-text;

CSS HYPERLINK STYLES

You can style **four link states** using CSS:

1. **Link (unvisited)** – default style for links not clicked yet.
2. **Visited** – links that have been clicked.
3. **Hover** – when mouse is over the link.
4. **Active** – when the link is clicked.

Example CSS

Unvisited link

```
a {
  text-decoration: none;          Remove underline
  color: #333;                    Default link color
}

Visited link  a:visited {  color: #666;          Visited link color }
```

Hover effect

```
a:hover {
  text-decoration: underline;     Show underline on hover
  color: #FF5733;                 Hover color
}
```

Active link

```
a:active {
  color: #FF0000;          Color when clicked
}
```

In short:

- **a** → unvisited
- **a:visited** → visited
- **a:hover** → mouse over
- **a:active** → clicked

Drop Box

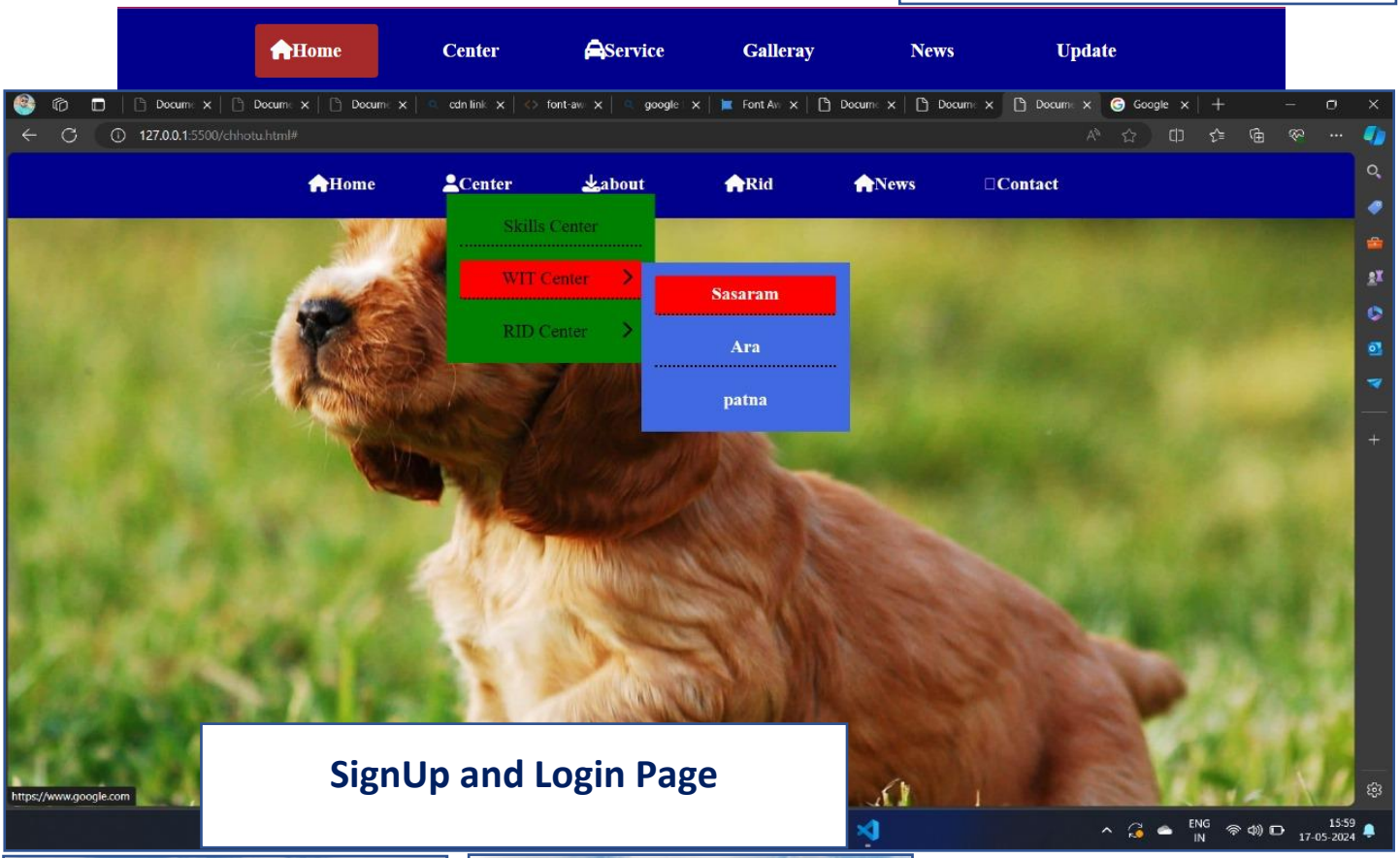
Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <link rel="stylesheet" href="dropbox.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ " />
</head>
<body>
  <div class="menu-bar">
    <ul>
<li id="act"><a href="#"><i class="fa-solid fa-house"></i>Home </a></li>
      <li><a href="#">Center</a>
        <div class="sub-menu-1">
          <ul>
            <li>Skills Center</li>
<li class="raj">WIT Center <i class="fa-solid fa-greater-than"></i>
            <div class="sub-menu-2">
              <ul>
                <li>Bhopal</li>
                <li>Patna</li>
                <li>Delhi</li>
              </ul>
            </div>
            </li>
<li class="raj" >RID Center <i class="fa-solid fa-greater-than"></i>
              <div class="sub-menu-2">
                <ul>
                  <li>Researcch Center</li>
                  <li>Innovation  Center</li>
                  <li>Discovery Center</li>
                </ul>
              </div></li></ul></div></li>
      <li><a href="#"><i class="fa-solid fa-taxi"></i>Service</a> </li>
      <li><a href="#">Galleray</a> </li>
      <li><a href="#">News </a>
        <div class="sub-menu-1">
          <ul>
            <li>Skills Center</li>
            <li>WIT Center <i class="fa-solid fa-greater-than"></i></li>
            <li>RID Center <i class="fa-solid fa-greater-than"></i></li>
          </ul>
        </div>
      </li>
      <li id="up"><a href="#">Update</a> </li>
    </ul> </div>
</body>
</html>
```

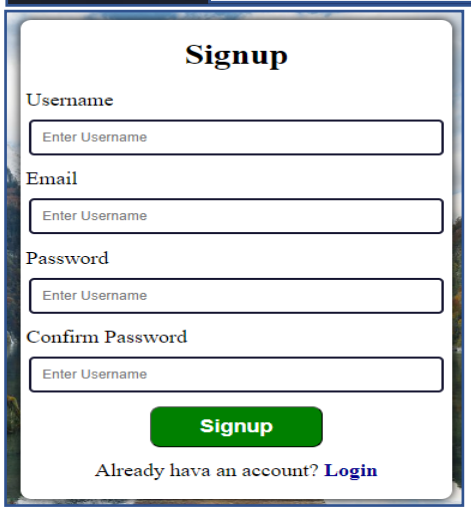
Dropboc.css

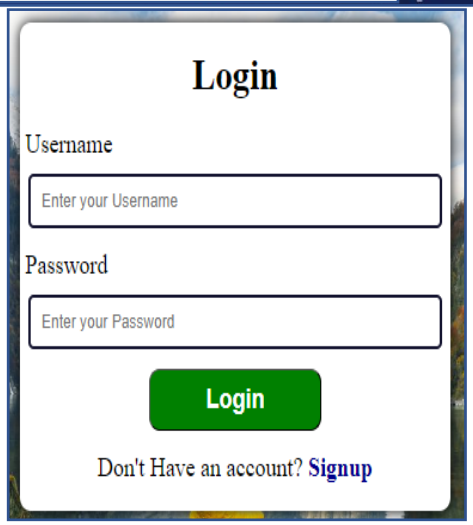
```
* { margin: 0;
   padding: 0;
   box-sizing: border-box }
body{
   background-image: url(nature1.jpg);
   background-size: cover;
   background-repeat: no-repeat; }
.menu-bar{
   background-color: darkblue;
   color: white;
   text-align: center; }
.menu-bar ul{
   display: inline-flex;
   list-style: none;}
.menu-bar ul li{
   padding: 20px;
   margin: 20px;
   font-size: 25px;
   position: relative;
   width: 160px; }
.menu-bar ul li a{
   text-decoration: none;
   color: white;
   font-weight: bold; }
#act, .menu-bar ul li:hover{
   background-color: brown;
   border-radius: 5px;
   text-align: left; }
.sub-menu-1{
   display: none; }
.menu-bar ul li:hover .sub-menu-1{
   display: block;
   margin-top: 20px;
   position: absolute;
  background-color: black;
   margin-left: -15PX; }
.menu-bar ul li:hover .sub-menu-1 ul{
   display: block; }
.sub-menu-1 ul li{
   font-size: 15px;
   border-bottom: solid red; }
.sub-menu-1 ul li:last-child{
   border-bottom: none; }
.fa-greater-than{
   float: right;
   color: white; }
.sub-menu-2{
   display: none; }
```

```
.raj:hover .sub-menu-2{
   display: block;
   margin-top: 0px;
   margin-left: 140px;
   position: absolute;
   background-color: black;
   /* background-image:
url(nature1.jpg); */
   background-size: cover;
   border: 5px solid red;
   border-radius: 5px;
   color: white;
   font-weight: bold;font-size: 30px;
}
```



SignUp and Login Page





Signup.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Signup</title>
  <link  rel="stylesheet" href="Style.css">
</head>
<body>
  <div class="container">
    <h1>Signup</h1>
    <form>
      <label for="username">Username</label>
      <input type="text" required placeholder="Enter Username">
      <label for="email">Email</label>
      <input type="email" required placeholder="Enter Username">
      <label for="password">Password</label>
      <input type="password" required placeholder="Enter Username">
      <label for="Re-Password">Confirm Password</label>
      <input type="password" required placeholder="Enter Username">
      <button type="submit">Signup</button>
    </form>
 <p>Already hava an account?  <a href="Login.html"> Login</a></p>
  </div>
</body> </html>
```

Login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Login</title>
  <link  rel="stylesheet" href="signup.css">
</head>
<body>
  <div class="container">
    <h1>Login</h1>
    <form>
      <label for="username">Username</label>
     <input type="text" required placeholder="Enter your Username">
      <label for="Password">Password</label>
      <input type="text" required placeholder="Enter your Password">
      <button type="submit">Login</button>
    </form>
    <p>Don't Have an account? <a href="Sigup.html">Signup</a></p>
  </div> </body> </html>
```

Style.css

```
body{
 /* background-color: aqua; */
   background-image: url(nature1.jpg);
   background-size: cover;
   margin: 0;
   padding: 0;
   display: flex;
   justify-content: center;
   align-items: center;
   height: 100vh; }
.container{
background-color: white;
   border-radius: 10px;
   box-shadow: 0 0 10px black;
   width: 400px;
   text-align: center; }
h1{margin-top: 20px; }
label{
display: block;
   text-align: left;
   margin-bottom: 10px;
   font-size: 20px;
   padding-left: 5px;}
input[type="text"],
input[type="email"],
input[type="password"]{
   width: 90%;
   padding: 10px;
   margin-bottom: 15px;
   border-radius: 5px;
   border: 3px solid rgb(18, 18, 46); }
button{ width: 40%;
   padding: 10px;
   background-color: green;
   color: white;
   font-weight: bold;
   border-radius: 10px;
   font-size: 20px;
   cursor: pointer; }
button:hover{ background-color: brown;}
p{ margin-top: 15px; font-size: 20px; }
a{
   text-decoration: none;
   /* color: green; */
   color: darkblue;
   font-weight: bold;
}
a:hover{
   text-decoration: underline;
}
```

Table style in css

1. border
- **Definition:** Adds a border around table or cells. **Syntax:** border: 2px solid black;
- **Example:** table, th, td { border: 2px solid black; }

2. border-collapse
- **Definition:** Combines or separates cell borders.
  Syntax: border-collapse: collapse; or border-collapse: separate;
- **Example:** table {border-collapse: collapse;}

3. border-spacing
- **Definition:** Space between cells (only with separate). **Syntax:** border-spacing: 10px 15px;
- **Example:** table {border-spacing: 10px 15px;}

4. Width & Height (table)
- **Definition:** Sets table size. **Syntax:** width: 80%; height: 300px;
- **Example:** table { width: 80%; height: 300px; }

5. Width & Height (th, td)
- **Definition:** Sets cell size. **Syntax:** width: 150px; height: 50px;
- **Example:** th, td { width: 150px; height: 50px; }

6. table-layout
- **Def:** Fixes or auto-adjusts table columns. **Syntax:** table-layout: fixed; or table-layout: auto;
- **Example:** table {table-layout: fixed;}

7. padding
- **Definition:** Space inside table cells. **Syntax:** padding: 10px;
- **Example:** th, td {padding: 10px;}

8. text-align
- **Definition:** Horizontal alignment inside cells. **Syntax:** text-align: left | center | right;
- **Example:** td { text-align: center;}

9. vertical-align
- **Definition:** Vertical alignment inside cells. **Syntax:** vertical-align: top | middle | bottom;
- **Example:** td { vertical-align: middle; }

10. caption-side
- **Definition:** Position of table caption. **Syntax:** caption-side: top | bottom;
- **Example:** caption { caption-side: top; }

11. :nth-child()
- **Definition:** Style specific rows or columns. **Syntax:** tr:nth-child(even) { ... }
- **Example:** tr:nth-child(even) { background-color: #f2f2f2; }

12. :hover
- **Definition:** Style on mouse hover.
- **Syntax:** tr:hover { ... }
- **Example:** tr:hover { background-color: #d1c4e9;}

13. :empty
- **Definition:** Style empty cells.
- **Syntax:** td:empty { ... }
- **Example:** td:empty { background-color: #ffc;}

CSS Animation

- CSS **animations** let you move, resize, or change colors of elements smoothly over time.
- They use **@keyframes** (to define steps) + **animation properties** (to control speed, delay, etc.).
- Shorthand order:
  animation: name duration timing-function delay iteration-count direction fill-mode play-state;

1. @keyframes

- Defines steps of animation.
- **Syntax:**

```
@keyframes mymove {
  from { background: red; }
  to { background: yellow; }
}
```

2. animation-name

- Name of the animation (linked to @keyframes).
- **Syntax:** div { animation-name: mymove; }

3. animation-duration

- How long the animation runs (s/ms).
- **Syntax:** div { animation-duration: 3s; }

4. animation-delay

- Delay before animation starts.
- **Syntax:** div { animation-delay: 2s; }

5. animation-iteration-count

- Number of times animation repeats (1, 3, infinite).
- **Syntax:** div { animation-iteration-count: infinite; }

6. animation-direction

- Direction of animation cycle.
- Values: normal | reverse | alternate | alternate-reverse.
- **Syntax:** div { animation-direction: alternate; }

7. animation-timing-function

- Speed curve of animation.
- Values: linear | ease | ease-in | ease-out | ease-in-out | cubic-bezier().
- **Syntax:** div { animation-timing-function: ease-in-out; }

8. animation-fill-mode

- Defines style before/after animation.
- Values: none | forwards | backwards | both.
- **Syntax:** div { animation-fill-mode: forwards; }

9. animation-play-state

- Controls play or pause.
- Values: running | paused.
- **Syntax:** div { animation-play-state: paused; }

10. animation (shorthand)

- Set all animation properties in **one line**.
- **Syntax:**div {animation: mymove 5s linear 2s infinite alternate forwards;}

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background: red;
  position: relative;
  animation: moveBox 4s ease-in-out 1s infinite alternate;
}
@keyframes moveBox {
  0%   { left: 0px; top: 0px; background: red; }
  50%  { left: 200px; top: 50px; background: blue; }
  100% { left: 0px; top: 100px; background: green; }
}
</style>
</head>
<body>
<h2>CSS Animation Example</h2>
<div></div>
</body>
</html>
```

How to Add Google Font Family

1. **Go to Google Fonts** → fonts.google.com
2. **Choose a font** (e.g., Poppins).
3. Copy the <link> code given. Example:
   <link href="https://fonts.googleapis.com/css2?family=Poppins&display=swap" rel="stylesheet">
4. Paste it inside the <head> of your HTML file.
5. Use it in CSS:

➔ body {font-family: 'Poppins', sans-serif; }

CSS Transformations

Used to **move, rotate, scale, skew** elements in 2D or 3D.

- **Property**: transform

Common Types:

1. **Translate** → Moves element transform: translate(50px, 20px); /* 50px right, 20px down */
2. **Scale** → Resizes element transform: scale(1.5); /* 1.5 times bigger */
3. **Rotate** → Rotates element transform: rotate(45deg); /* 45° clockwise */
4. **Skew** → Slants element transform: skew(20deg, 10deg); /* Tilt */
5. **Perspective** → Adds 3D depth transform: perspective(500px) rotateY(45deg);

Note: You can **combine multiple**: transform: translate(20px, 20px) scale(1.2) rotate(30deg);

CSS Transitions Used to make **smooth animations** when property values change (like hover, click).

- **Property**: transition

Syntax: transition: property duration timing-function delay;

Example: Smooth Hover Effect

```
.box {
  width: 100px;
  height: 100px;
  background: red;
  transition: transform 0.5s ease; /* Smooth 0.5s animation */}
.box:hover {transform: scale(1.3) rotate(15deg);}
```

Note:

- **Transform** = Changes the **shape/position** of an element.
- **Transition** = Makes the **change smooth** over time.

CSS Transitions

- Transitions make property changes (like color, size, position) happen smoothly over time.
- **Use**: Commonly used for hover effects (e.g., button color changes smoothly).
- **Example**: transition: background-color 0.3s ease-in-out;

Key Concepts

1. **Transition Properties**

- **transition-property** → which CSS property changes.
  - Example: transition-property: background-color;
  - Or use all → transition-property: all;
- **transition-duration** → how long it takes.
  - Example: transition-duration: 0.5s;
- **transition-timing-function** → speed style.
  - ease, linear, ease-in, ease-out, ease-in-out, or custom (cubic-bezier).
  - Example: transition-timing-function: ease-in;
- **transition-delay** → wait before starting.
  - Example: transition-delay: 0.2s;

Example of CSS transitions: IMAGE TRANSFORMATIONS

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Image Transformation Example</title>
  <style>
    .image-container {
      width: 300px;
      margin: 20px auto;
    }
    .image {
      width: 100%;
      max-width: 100%;
      height: auto;
      transition: transform 0.3s ease;
    }
    .image:hover {
      transform: scale(1.2);
```

```
    }
  </style>
</head>
<body>
  <h1>Image Transformation Example</h1>
  <div class="image-container">
    <img class="image" src="image/img4.jpg" alt="Sample Image">
    <p>image size will increase after hover</p>
  </div>
</body>
</html>
```

Image Transformation Example

image size will increase after hover

Open VS Code Live Server on Mobile

Requirements:

- VS Code installed with Live Server extension.
- Computer and mobile on the same Wi-Fi.

Steps:

1. **Open Project in VS Code** → Load your folder.
2. **Start Live Server** → Right-click index.html → *Open with Live Server* OR click **Go Live**.
   - URL opens like http://127.0.0.1:5500.
3. **Find Computer IP**
   - **Windows**: Run ipconfig → note IPv4 (e.g., 192.168.1.x).
   - **Mac**: Check System Preferences > Network → find IP.
4. **Replace URL**
   - Change localhost to your IP: http://192.168.1.x:5500.
   - Test on your PC browser first.
5. **Open on Mobile**
   - On mobile browser, enter the same IP URL.
   - Your site opens!
6. **Live Changes**
   - Edits in VS Code auto-refresh on both devices.

Tips:

- Ensure firewall allows port 5500.
- Both devices must be on the same Wi-Fi.
- If issues → check Live Server settings in VS Code.

min-width in CSS

- **Definition**: Sets the minimum width of an element.
- **Purpose**:
    1. Prevents elements (like buttons, boxes) from becoming too small.
    2. Keeps layout consistent on different screen sizes.

Example:

```
.container {
  min-width: 300px;
  background: #f2f2f2;
  padding: 20px;}
```

This ensures .container is **never smaller than 300px**, even if the screen is small.

max-width in CSS

- Sets the **maximum width** of an element. The element won't grow wider than this value.
- **Syntax**: .container { max-width: 800px;}

Why We Use max-width:

1. **Responsive Design** – Stops content from stretching too wide on big screens.
2. **Better Layout** – Keeps text and elements easy to read and visually balanced.

Example: .container {

```
  max-width: 600px;
  background: #f2f2f2;
  padding: 20px;
  margin: 0 auto;   /* center */ }
```

Media Query

CSS Media Queries

- **Definition**: Media queries let you apply CSS styles based on device conditions (like screen size, orientation, or resolution).
- **Use**: They make websites **responsive**, so they look good on mobiles, tablets, and desktops.

Basic Syntax

```
@media (condition) {   /* CSS rules */ }
```

Example:

```
@media (max-width: 600px) {
  body { background: lightblue; } }
```

Applies when screen width ≤ 600px.

Common Features

1. **Width & Height**
- max-width → style for smaller screens
- min-width → style for larger screens
2. **Orientation**
- portrait → vertical mode
- landscape → horizontal mode
3. **Resolution** → Target high DPI screens (e.g., Retina).

Operators

- and → both conditions must be true.
- , (comma) → OR (any condition true).
- not → exclude condition.
- only → target specific media types.

Media Types

- screen → for devices (mobiles, desktops).
- print → for printers.
- all → applies everywhere.

Breakpoints (common ranges)

- Mobile → max-width: 600px
- Tablet → 601px – 900px
- Desktop → 901px+

Example with Flex/Grid

```
.container { display: flex; }
@media (max-width: 768px) {
  .container { flex-direction: column; }
}
```

Tips

✓ Use **mobile-first** (start small, then add larger).

✓ Keep queries simple & organized.

✓ Always test on real devices or browser tools.

How we you can Develop a responsive website

1. **Flexible Layouts (Fluid Grids)**

- Use %, em, or rem instead of fixed pixels.
  .container { width: 80%; padding: 1rem; }

2. **Media Queries**

- Apply different styles for mobiles, tablets, and desktops.
  @media (max-width: 600px) {
  .container { width: 100%; }
  }

3. **Responsive Units**

- Use vw (viewport width) and vh (viewport height).

.header { height: 10vh; }

4. **Responsive Typography**

- Scale text with em, rem, or vw. body { font-size: 1.2vw; }

5. **Flexible Images & Media**

- Make images scale automatically. img { max-width: 100%; height: auto; }

6. **Flexbox & Grid**

- Use modern layouts that adjust easily.
  Flexbox
  .container { display: flex; flex-wrap: wrap; }
  Grid

```
.grid-container {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
  gap: 20px;
}
```

Note: **Use flexible units + media queries + Flexbox/Grid + scalable images** → your site will

CSS Units

Units define **sizes** in CSS (like width, height, font size, margin).
They are of **two types**:

1. Absolute Units (Fixed size – don't change with screen/parent)

- **px (pixels)** → Most common, fixed point on screen.
  .box { width: 200px; height: 150px; }
- **cm, mm (centimeters, millimeters)** → For print.
- **in (inches)** → Used in print.
- **pt (points)** → 1pt = 1/72 inch, typography in print.
- **pc (picas)** → 1pc = 12pt, rarely used.

2. Relative Units (Flexible – change with screen/parent)

- **% (percentage)** → Relative to parent size.

.container { width: 80%; }

- **em** → Relative to parent font size.

.text { font-size: 1.5em; } /* 1.5 × parent font */

- **rem** → Relative to root (html) font size.

.heading { font-size: 2rem; }

- **vw / vh** → Relative to viewport (screen).

.banner { width: 50vw; } /* 50% of screen width */
.full { height: 100vh; } /* full screen height */

- **vmin / vmax** → Based on smaller/larger side of screen.
- **ch** → Width of "0" character, good for text layouts.
- **ex** → Height of lowercase "x", rarely used.
- **lh** → Based on line-height of element.

Practical Tips

✓ Use **px** for small fixed elements.
✓ Use **%** for flexible layouts.
✓ Use **em/rem** for fonts & spacing.
✓ Use **vw/vh** for full-screen sections.
✓ Use **ch** for text blocks.

CSS Interview Question and Answer

1. What is CSS?

Answer: CSS (Cascading Style Sheets) is a stylesheet language used to describe the presentation of a document written in HTML or XML.

2. What is the box model in CSS?

Answer: The CSS box model describes the rectangular boxes generated for elements, consisting of margins, borders, padding, and the content area.

3. What is the difference between class and id selectors?

Answer: class selectors can be used multiple times within a document, while id selectors are unique to a single element.

4. What is the difference between absolute, relative, fixed, and static positioning?

Answer:

absolute: Positioned relative to the nearest positioned ancestor.
relative: Positioned relative to its normal position.
fixed: Positioned relative to the viewport.

static: Default position, element follows the normal document flow.

5. What are CSS pseudo-classes?

Answer: Pseudo-classes are used to define the special state of an element, e.g., :hover, :first-child.

6. What are CSS pseudo-elements?

Answer: Pseudo-elements are used to style specific parts of an element, e.g., ::before, ::after.

7. How can you include CSS in an HTML document?

Answer: Through inline styles, internal stylesheets within the <style> tag, and external stylesheets using the <link> tag.

8. What is a CSS preprocessor?

Answer: A CSS preprocessor is a scripting language that extends CSS and compiles it into regular CSS. Examples include Sass, LESS, and Stylus.

9. What is the use of the z-index property?

Answer: The z-index property specifies the stack order of elements; elements with a higher z-index appear in front of those with a lower z-index.

10. How can you center a block element horizontally in CSS?

Answer: By setting margin: auto; and a width for the element.

11. What is Flexbox?

Answer: Flexbox (Flexible Box Layout) is a layout model that allows items within a container to be automatically arranged in a predictable way to accommodate different screen sizes.

12. How do you create a CSS grid layout?

Answer: By using the display: grid; property on a container and defining rows and columns with grid-template-rows and grid-template-columns.

13. What is the difference between margin and padding?

Answer: Margin is the space outside an element's border, while padding is the space inside the element's border.

14. How can you make a website responsive?

Answer: By using media queries, flexible grid layouts, flexible images, and employing techniques like responsive units (%, em, rem, vw, vh).

15. What are media queries?

Answer: Media queries are a feature of CSS that allows content to adapt to different screen sizes and resolutions using the @media rule.

16. How do you apply styles for a specific element when it is being hovered over?

Answer: By using the :hover pseudo-class, e.g., element:hover { /* styles */ }.

17. What is a CSS transition?

Answer: CSS transitions allow changes in CSS property values to occur smoothly over a specified duration.

18. How can you use CSS to hide an element?

Answer: By using display: none; or visibility: hidden;.

19. What are CSS animations?

Answer: CSS animations enable the gradual change of CSS properties using keyframes, defined with the @keyframes rule.

20. What does the inherit value do in CSS?

Answer: The inherit value causes a property to take the same computed value as the property for the element's parent.

21. How can you make a list horizontal using CSS?

Answer: By setting the list items to display: inline; or using display: flex; on the parent container.

22. What is the purpose of the calc() function in CSS?

Answer: calc() function allows you to perform calculations to determine CSS property values dynamically.

23. How do you apply a style to multiple classes?

Answer: By separating the classes with a comma in the selector, e.g., .class1, .class2 { /* styles */ }.

24. What is the difference between inline and block elements?

Answer: inline elements do not start on a new line and only take up as much width as necessary. block elements start on a new line and take up the full width available.

25. How can you create a dropdown menu using CSS?

Answer: By using position: relative; on the parent and position: absolute; on the child, along with :hover to toggle visibility.

26. What is the ::before pseudo-element used for?

Answer: The ::before pseudo-element is used to insert content before the content of an element.

27. How can you make text italic in CSS?

Answer: By using the font-style: italic; property.

28. What does box-sizing: border-box; do?

Answer: It makes the width and height properties include content, padding, and border, but not the margin.

29. What is the em unit in CSS?

Answer: The em unit is a relative unit that is based on the font size of the element. 1em is equal to the current font size.

30. How can you clear floated elements?

Answer: By using the clear property on a following element or using the clearfix hack.