CSS SUPREME

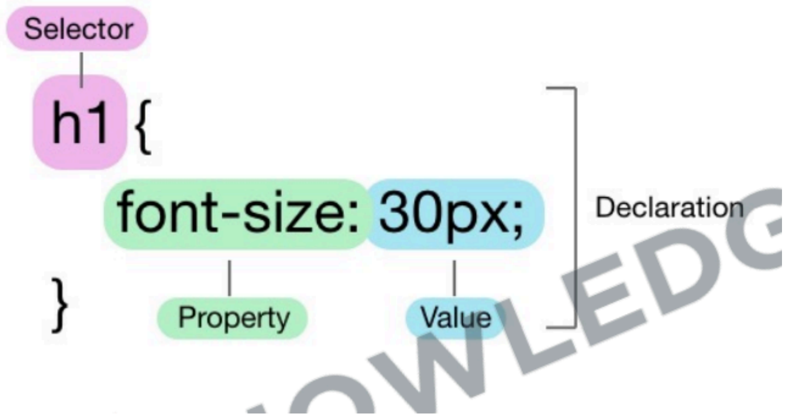
Chapter-1

https://github.com/Prakash9596/CSS_Supreme
https://prakash9596.github.io/CSS_Supreme/

Importance of CSS:

Basic Syntax:

• 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.).



Comment in HTML: <!-- Comment -->

Comment in CSS: /* Comment */

Color Property:

• Definition: The CSS color property defines the text color or foreground color in an HTML element.

• Enhancement: Use it to emphasize sections and elevate webpage aesthetics.

```
<head>
  <title>Color</title>
  <style>
    h1 {
      color: green;
    }
    p {
      color: red;
    }
  </style>
</head>
```

Including Styles (Inline Styling)

• Direct Application: Apply styles directly to HTML elements using the style attribute.

• One-off Changes: Ideal for single, unique style alterations.

• Can Be Cluttered: May lead to cluttered HTML if used extensively.

• Limited Reusability: Reduces the reusability of CSS rules in larger projects.

```
<body>
  <h1 style="color: red;">Inline Styling</h1>
  <p style="color: green;">Paragraph is green</p>
</body>
```

Including Styles (Internal Styling)

• Embedded CSS: Styles are placed within <style> tags in the HTML head section.

• Cleaner than Inline: More organized compared to inline styles.

• Reusable Styles: Allows for some reuse of styles across the Page.

```
<head>
  <title>Internal Styling</title>
  <style>
    h1 {
      color: green;
    }
    p {
      color: red;
    }
  </style>
</head>
```

Including Styles (External Styling)

• Separate CSS File: Stores styles in a separate .css file, linked to HTML.

• Reusable: Enables style reuse across multiple webpages.

• Link in HTML: Use the tag within the section to link the CSS.

• Relative or Absolute Path: The href attribute can contain a relative or absolute path to the CSS file.

```
<head>
  <title>External Styling</title>
  <link rel="stylesheet" href="/Course Code/css/level 1/4.external styling.css">
</head>
```

Priority: (Inline Styling > Internal Styling > External Styling)

Selectors (Element selector)

• Targets Elements: Selects HTML elements based on their tag name.

• Syntax: Simply use the element's name

• Uniform Styling: Helps in applying consistent styles to all instances.

• Ease of Use: Straightforward and easy to implement for basic styling.

```
 <style>
    h1 {
      color: red;
    }
  </style>
```

Selectors (Universal selector)

• Matches All: Targets and styles all elements on a webpage.

• Syntax: Utilized as an asterisk (*).

• Resets Styles: Commonly used to reset margins and paddings globally.

• Broad Styling: Useful for setting universal attributes like font or color.

• Usage Caution: Can cause style conflicts due to its wide-reaching effects.

```
<head>
  <title>Universal Selector</title>
  <style>
    * {
      color: red;
    }
  </style>
</head>
```

Selectors (id & class property)

• ID Property: Assigns a unique identifier to a single HTML element.

• Class Property: Allows grouping of multiple HTML elements to style them collectively.

• Reusable Classes: Class properties can be reused across different elements for consistent styling.

• Specificity and Targeting: Both properties assist in targeting specific elements or groups of elements for precise styling.

```
<body>
  <h1 id="top_heading">Id and Class</h1>
  <p class="article">Lorem ipsum dolor sit amet consectetur adipisicing elit. Nulla, nisi.</p>
</body>
```

Selectors (Id selector)

• Unique Identifier: Targets a specific element with a unique ID attribute.

• Syntax: Uses the hash (#) symbol

• Single Use: Each ID should be used once per page for uniqueness.

• Specific Targeting: Ideal for styling individual, distinct elements.

```
<head>
  <title>Id selector</title>
  <style>
    #first { color: red; }
    #second { color: green; }
  </style>
</head>
<body>
  <div id="first">First Div</div>
  <div id="second">Second Div</div>
</body>
```

Selectors (Class selector)

• Group Styling: Allows styling of multiple elements grouped under a class.
• Syntax: Utilizes the dot (.) symbol.
• Reusable: Can be used on multiple elements for consistent styling.
• Versatility: Ideal for applying styles to a category of elements.

```
<head>
  <title>Class selector</title>
  <style>
    #first { color: red; }
    .second { color: green; }
  </style>
</head>
<body>
  <div id="first">First Div</div>
  <div class="second">Second Div</div>
  <div class="second">Third Div</div>
</body>
```

Selectors (Group selector)

• Multiple Elements: Styles multiple elements simultaneously.
• Syntax: Separates selectors with commas.
• Efficiency: Reduces code redundancy and saves time

```
<head>
    <title>Group selector</title>
    <style>
        h1, h2, h3 {
            color: red
        }
    </style>
</head>
<body>
    <!-- Kisi bhi tarah ke selector ko combine kiya ja skta h.. element ho, id ho, class ho-->
    <h1>Heading 1</h1>
    <h2>Heading 2</h2>
    <h3>Heading 3</h3>
    <h4>Heading 4</h4>
</body>
```

Selectors (Descendant selector)

• Nested Targeting: Styles elements nested within a specified element.

• Syntax: Separate selectors with spaces.

• Hierarchy-Based: Allows styling based on the hierarchical structure of HTML.

• Specific Styling: Facilitates more targeted and specific styling of elements.

```
<head>
 <style>
        #first p { color: red; }
        div h1 { color: aqua;}
        p { color: blue; }
    </style>
</head>
<body>
    <div id="first"> <p>Lorem ipsum </p> </div>
    <div> <h1>Lorem</h1>  </div>
    <p>Lorem ipsum</p>
</body>
```

Priority: (Element/ID/Class/Group/Descendant selector > Universal selector)

Chapter-2

Background Color

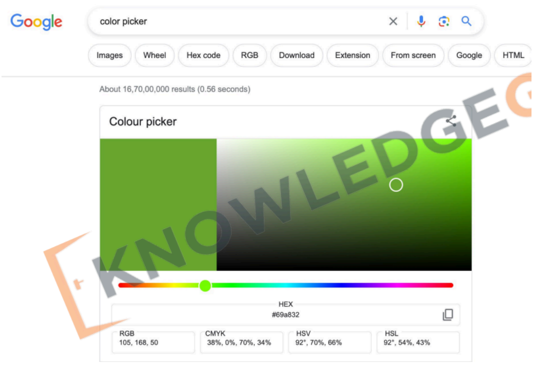
- Definition: Sets the background color of an element.
- Syntax: Utilized as background-color: color;
- Visual Appeal: Enhances the visual appeal and contrast of webpage elements.

```
<head>
   <title>Background Color</title>
   <style>
      #first { color: black; background-color: green;  }
      #second { color: black; background-color: rgba(216, 49, 133, 0.3); }
      button { color: black; background-color: #7ed6ef; }
   </style>
</head>
<body>
   <div id="first">First</div>
   <div id="second">Second</div>
   <button>Click Me</button>
</body>
```

Color System (Color Theory)

- RGB Model: Creates colors by mixing Red (R), Green (G), and Blue (B) light sources.
- Additive Model: More light means increased brightness.
- Primary Colors: R, G, and B are the foundational colors.
- White & Black: All combined yield white; absence equals black.
- Color Depth: Allows for millions of color variations.

Color System (Search color picker To get the code of any colour.)



About 16,70,00,000 results (0.56 seconds)

Colour picker

HEX
#69a832

RGB
105, 168, 50

CMYK
38%, 0%, 70%, 34%

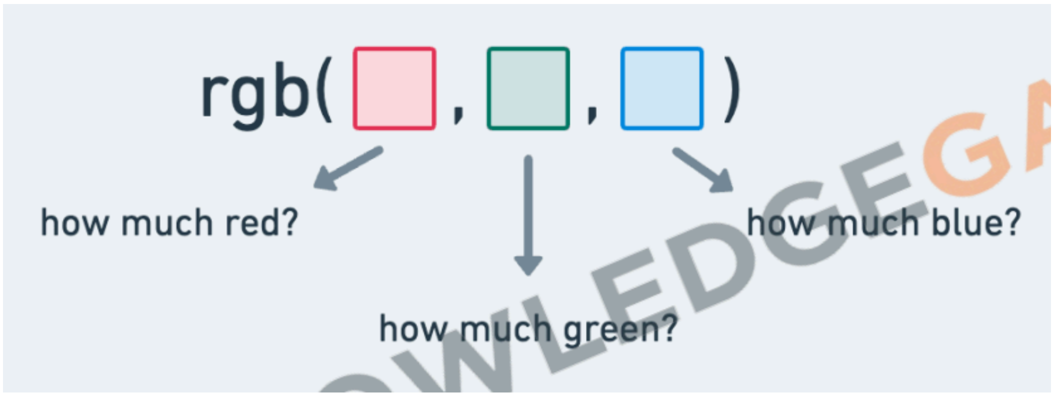
HSV
92°, 70%, 66%

HSL
92°, 54%, 43%

Color System (RGB Color Model)

• Three Channels: Consists of Red (R), Green (G), and Blue (B) channels to create a variety of colors.

• Syntax: Utilized as rgb(r, g, b) where r, g, and b are values between 0 and 255.



Color System (RGB Color Model)

```
<body>
   <div style="background-color: rgb(255,0,0);">First</div>
   <div style="background-color: rgb(0,255,0);">Second</div>
   <div style="background-color: rgb(0,0,255);">Third</div>
   <div style="background-color: rgb(28, 244, 244);">Fourth</div>
</body>
```

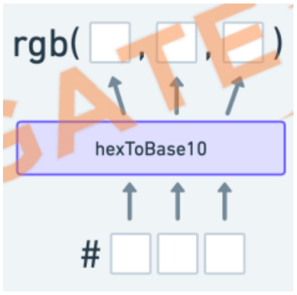
Color System (HEX Color Model)

• Hexadecimal Codes: Represents colors using hexadecimal values, consisting of 6 digits combined from numbers and letters (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



Color System (HEX Color Model)

```html
<body>
   <div style="background-color: #ff0000;">First</div>
   <div style="background-color: #1fc51f;">Second</div>
   <div style="background-color: #0000ff;">Third</div>
   <div style="background-color: #ed35ce;">Fourth</div>
</body>
```

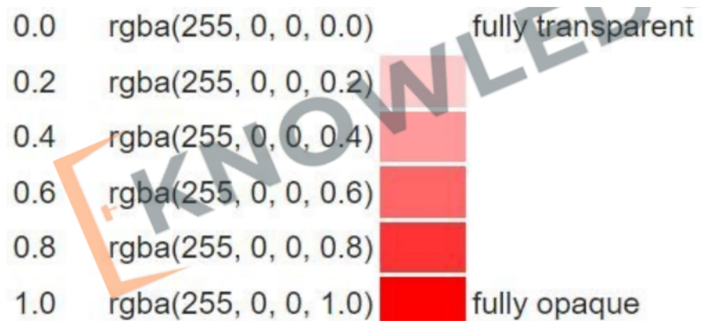
Color System (Alpha Channel)

• RGBA: RGB's extension, includes alpha for opacity control (0-1 range).

• Transparency Control: Facilitates the adjustment of transparency levels in colors.

• Visual Effects: Enables the creation of visual effects like shadows and overlays.

• Layering: Assists in layering elements with varying degrees of visibility.



0.0	rgba(255, 0, 0, 0.0)		fully transparent
0.2	rgba(255, 0, 0, 0.2)	🟥	
0.4	rgba(255, 0, 0, 0.4)	🟥	
0.6	rgba(255, 0, 0, 0.6)	🟥	
0.8	rgba(255, 0, 0, 0.8)	🟥	
1.0	rgba(255, 0, 0, 1.0)	🟥	fully opaque

```html
<body>
   <!-- rgba: red, green, blue, alpha (visibility: 0-1)-->
   <h1 style="color: rgba(255,0,0,0.1);">First</div>
   <h1 style="color: rgba(255,0,0,0.25);">First</div>
   <h1 style="color: rgba(255,0,0,0.5);">First</div>
   <h1 style="color: rgba(255,0,0,0.75);">First</div>
   <h1 style="color: rgba(255,0,0,1.0);">First</div>
</body>
```

Absolute Units

• Definition: Pixels (px) are fixed-size units, representing a dot on a computer screen.

• Precision: Allows for precise control over element dimensions.

• Graphics & Web Design: Commonly used in graphics and web design for setting font 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

• Dimensions Control: Used to specify the height and width of elements.

• Unit Variability: Can use units like pixels (px)

• Box Model Component: Influences padding, border, and margin.

• Min and Max Values: Can utilize min-height, max-height, min-width, and max-width to set restrictions on dimensions.

```
<head>
  <style>
    .box { height: 40px; width: 40px;}
    #box1 {background-color: red;}
    #box2 {background-color: blue;}
    #box3 {background-color: green;}
  </style> </head>
<body>
  <div id="box1" class="box">Box 1</div> <br>
  <div id="box2" class="box">Box 2</div> <br>
  <div id="box3" class="box">Box 3</div>
</body>
```

Background image Property

• Repetition: Control image repetition using background-repeat.

• Positioning: Adjust image position using background position.

• Size Control: Manipulate image size using background-size.

```
<!-- Need to know more:
• Usage: Adds an image as a background to elements.
• Syntax: Defined using background-image: url('path/to/image');.
• background-repeat: no-repeat; se image box ke andar repeat na ho agar
box ka size bada h to.
• background-position: right; se apne image ko box ke andar left right
kahi v position kr skte hain.
• background-size: contain; se image kitna bhi bada/chhota ho wo box size
ke andar hi aayega.
• background-size: cover; se no empty space.
• Background-Attachment: Sets whether the background image scrolls with
the element or remains fixed.
• Shorthand (color, image, repeat, attachment, position)
background: green url('jeans.webp') no-repeat center;  -->
```

```
<style>
        #box1 {
            background-image: url(../../images/OIP.jfif);
            height: 296px;
            width: 474px;
        }
        #box2 {
            background-image: url(../../images/css.png);
            height: 512px;
            width: 512px;
        }
        #box3 {
            background-image: url(../../images/OIP.jfif);
            height: 700px;
            width: 700px;
            background-size: contain;
        }
        #box4 {
            background-image: url(../../images/OIP.jfif);
            height: 700px; width: 700px;
            background-size: contain;
            background-repeat: no-repeat;
            background-position: right;
        }
    </style>
```

Visibility Property

• Usage: Controls the visibility of elements without changing the layout.
• Values: Can take visible, hidden, or collapse as values.
• Space Occupancy: Even when hidden, the element occupies space.
• Interactivity: Hidden elements are not accessible to user interactions.

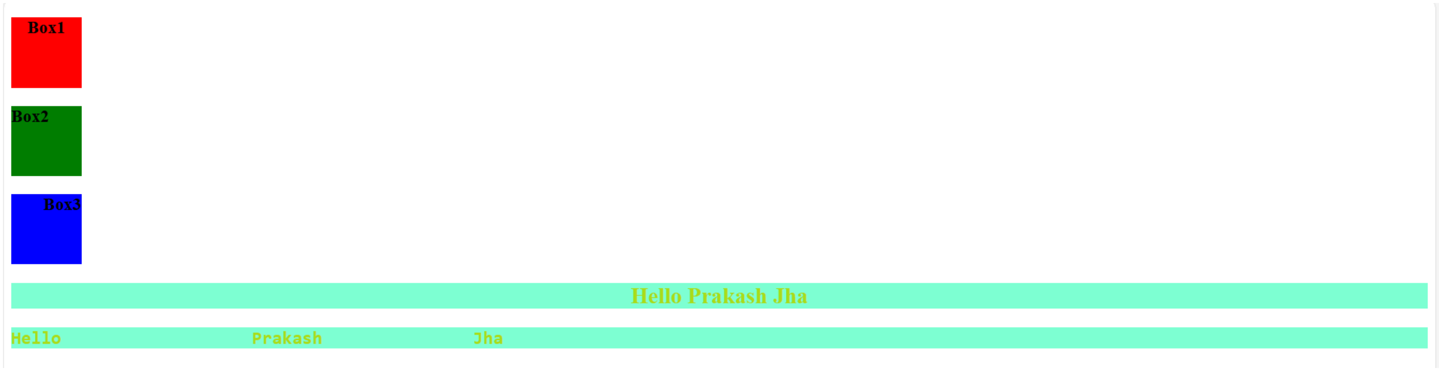
```
<style>
  .box { height: 40px; width: 40px;}
   #box1 {background-color: red; visibility: hidden;}
   #box2 {background-color: green; visibility: visible;}
</style>
```

Chapter-3

Text-Align Property

• Usage: Controls the horizontal alignment of text within an element.
• Values: Can take values like left, right, center, and justify.
• Visual Appeal: Enhances readability and visual appeal by organizing text neatly.

```
<head>
  <title>Text Align</title>
  <style>
    .box {height: 75px; width: 75px;}
    #box1 {background-color: red; text-align: center;}
    #box2 {background-color: green; text-align: left;}
    #box3 {background-color: blue; text-align: right;}
    #a1 { background-color: aquamarine; color: rgb(173, 222, 27); text-align: center;}
    #a2 { background-color: aquamarine; color: rgb(173, 222, 27);}
  </style> </head>
<body>
  <h3 id="box1" class="box">Box1</h3>
  <h3 id="box2" class="box">Box2</h3>
  <h3 id="box3" class="box">Box3</h3>
  <h2 id="a1"> Hello Prakash Jha</h2>
  <pre id="a2"><h2>Hello            Prakash          Jha</h2></pre>
</body>
```

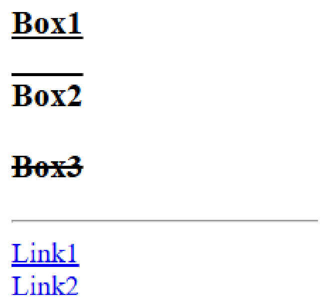


```
Hello Prakash Jha
Hello              Prakash             Jha
```

Text-Decoration Property

• Usage: Modifies the appearance of inline text.
• Values: Options include none, underline, overline, and line-through.
• Hyperlinks: Commonly used to remove underlines from hyperlinks for aesthetic purposes.

```
<head>
    <title>Text Decoration</title>
    <style>
        #box1 {text-decoration: underline;}
        #box2 {text-decoration: overline;}
        #box3 {text-decoration: line-through;}
    </style>
</head>
<body>
    <h3 id="box1" class="box">Box1</h3>
    <h3 id="box2" class="box">Box2</h3>
    <h3 id="box3" class="box">Box3</h3> <hr>
    <a href="#">Link1</a> <br>

    <a href="#" style="text-decoration: none;">Link2</a>
</body>
```



Box1

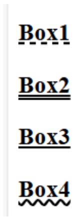
Box2

~~**Box3**~~

Link1
Link2

Text-Decoration Property (style)

```
<style>
        .box {text-decoration: underline;}
        #box1 {text-decoration-style: dashed;}
        #box2 {text-decoration-style: double;}
        #box3 {text-decoration-style: solid;}
        #box4 {text-decoration-style: wavy;}
    </style>
```



Box1

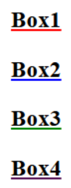
Box2

Box3

Box4

Text-Decoration Property (color)

```
<head>
  <title>Text Decoration Style</title>
  <style>
    .box {text-decoration: underline;}
    #box1 {text-decoration-color: red;}
    #box2 {text-decoration-color: blue;}
    #box3 {text-decoration-color: green;}
    #box4 {text-decoration-color: rgb(86, 20, 86);}
  </style>
</head>
<body>
  <h3 id="box1" class="box">Box1</h3>
  <h3 id="box2" class="box">Box2</h3>
  <h3 id="box3" class="box">Box3</h3>
  <h3 id="box4" class="box">Box4</h3>
</body>
```



```
Box1

Box2

Box3

Box4
```

Text-Transform Property

• Usage: Controls the capitalization of text.

• Common Values: Can be uppercase, lowercase, or capitalize.

• None Value: none value disables text transformations.

• Typography: Useful for setting text style and improving typography in web design.

```
<head>
  <title>Text Transform</title>
  <style>
    #box1 {text-transform: uppercase;}
    #box2 {text-transform: lowercase;}
    #box3 {text-transform: capitalize;}
    #box4 {text-transform: none;}
  </style></head>
```

```
<body>
   <h3 id="box1" class="box">First bOx</h3>
   <h3 id="box2" class="box">Second bOx</h3>
   <h3 id="box3" class="box">Third bOx</h3>
   <h3 id="bo43" class="box">Fourth bOx</h3>
</body>
```

Line Height

• Usage: Adjusts the amount of space above and below inline elements.
• Readability: Enhances text readability by preventing overcrowding.
• Vertical Spacing: Useful for controlling vertical spacing between lines of text.

```
<head>
   <title>Line Height</title>
   <style>
      #first { line-height: 6px; }
      #second { line-height: 30px; }
   </style>
</head>
<body>
   <p id="first">Lorem ipsum dolor, sit amet consectetur adipisicing elit. Quis odit blanditiis
nobis </p> <hr>
   <p id="second">Lorem ipsum dolor sit amet consectetur adipisicing elit. Esse, harum sit
ex sapiente </p>
</body>
```

Font Property (font-size)

• Usage: Sets the size of the font in web content.
• Responsiveness: Helps in creating responsive designs adaptable to various screen sizes.
• Readability: Crucial for ensuring the readability of text on websites.

```
<head>  <title>Font Size</title>  <style>
      #first {font-size: 5px;} #second {font-size: 15px;} #third {font-size: 25px;}
      #fourth {font-size: 35px;}  </style>  </head>
<body>
  <p id="first">P1</p> <p id="second">P2</p> <p id="third">P3</p>
   <p id="fourth">P4</p>
</body>
```

Font Property (font-weight)

• Text Emphasis: Utilized to emphasize text or create contrast between text elements.

```
<head>
   <title>Font Weight</title>
   <style>
      #first {font-weight: 100;}
      #second {font-weight: 400;}
      #third {font-weight: 600;}
      #fourth {font-weight: 900;}
   </style>
</head>
<body>
   <!-- Usage: Defines the thickness of characters in a font. (Normal=400, Bold=700)
        Values: Can take values like normal, bold, bolder, or numeric values (100 to 900). - ->
   <p id="first">Lorem ipsum dolor sit amet</p>
   <p id="second">Lorem ipsum dolor sit amet</p>
   <p id="third">Lorem ipsum dolor sit amet</p>
   <p id="fourth">Lorem ipsum dolor sit amet</p>
</body>
```

Font Property (font-style)

• Usage: Controls the style of the font, mainly affecting its inclination.

• Values: Common values are normal, italic, and oblique.

• Text Formatting: Useful for highlighting or distinguishing certain text segments.

```
<head>
   <style>
      #first {font-style: italic;}
      #second {font-style: oblique;}
      #third {font-style: normal;}
   </style>
</head>
<body>
   <p id="first">Lorem ipsum dolor sit amet</p>
   <p id="second">Lorem ipsum dolor sit amet</p>
   <p id="third">Lorem ipsum dolor sit amet</p>
</body>
```

Font Family

• Usage: Defines which font should be used for text within an element.
• Specific Fonts: Common choices include Arial, Segoe UI, Times New Roman, and others.
• Fallback 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 and operating systems.
• Generic Family: Always end the font family list with a generic family like serif or sans-serif as a last resort option.

```
<head>
  <title>Font Family</title>
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link
href="https://fonts.googleapis.com/css2?family=Dancing+Script:wght@400..700&display=s
wap" rel="stylesheet">
  <style>
    #first {font-family: Arial, Helvetica, sans-serif;}
    #second {font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;}
    #third {font-family: "Dancing Script", cursive;}
  </style>
</head>
<body>
  <p id="first">Lorem</p> <p id="second">Lorem</p> <p id="third">Lorem </p>
</body>
```

Note: font-family can be downloaded from google fonts. (Copy link put it in head and copy font-family put it in style)

Icons using Fonts Using https://fontawesome.com

```
(Copy script put it in head and copy <i></i> put it in body)
<head>
    <script src="https://kit.fontawesome.com/43290fa92d.js"
crossorigin="anonymous"></script> </head>
<body> <i class="fa-solid fa-house"></i>
<i class="fa-brandsfa-facebook"></i><i class="fa-brands fa-tiktok"></i>
<i class="fa-solid fa-poo" style="font-size: 40px;"></i>
</body>
```

Chapter-4

What is Box Model

• Core Concept: Central concept in CSS that outlines the design and layout of elements on the web page.
• Components: Consists of four main components - margin, border, padding, and content.
• Margin: The space outside the border, separating the element from others.
• Border: The outline that encapsulates the padding and content.
• Padding: The space between the border and the actual content, providing a buffer.
• Content: The innermost layer where text, images, or other media are housed.

Padding Property

• Usage: Defines the space between the content of an element and its border.
• Individual Sides: Allows setting padding for individual sides using padding-top, padding-right, padding-bottom, and padding-left.
• Shorthand: Can use shorthand property padding to set all sides at once, e.g., padding: 10px 20px 10px 20px.

```
<head>
  <title>Padding</title>
  <style>
    * { margin: 0; padding: 0; }
    #button1 {
      padding: 5px 10px 15px 20px; /*(Top Right Bottom Left): C.W*/
      background-color: aquamarine;
      height: 80px;
      width: 80px;
    }
  </style>
</head>
<body>
  <button id="button1">Click Me</button>
</body>
<!--
padding: 5px; means it's the same for all four sides.
padding: 5px 10px; means (top and bottom 5px) & (Right and left 10px)
-->
```

Margin Property

• Functionality: Sets the space around elements, separating them from others.

• Individual Sides: Customizable for top, right, bottom, and left sides.

• Shorthand: Allows quick setup, e.g., margin: 10px 20px. (clockwise)

• Auto Value: Can be used for central alignment with auto value.

```
<head>
  <title>margin</title>
  <style>
    * { margin: 0; padding: 0; }
    .pad {
      height: 50px;
      width: 50px;
      padding: 2px;
    }
    #first {
      margin: 15px 10px 25px 20px; /*(Top Right Bottom Left): C.W*/
      background-color: rgb(225, 30, 76);
    }
    #second {
      margin: 50px 10px 5px 20px;
      background-color: rgb(35, 214, 92);
    }
  </style>
</head>
<body>
  <div id="first" class="pad">First</div>
  <div id="second" class="pad">Second</div>
</body>
```

Border Property

• Usage: Creates an outline around HTML elements.

• Components: Defined by width, style, and color attributes.

• Styles: Includes options like solid, dashed, and dotted.

• Shorthand: Can set attributes at once, e.g., border: 2px solid black.

```
<head>
    <title>Border</title>
    <style>
        * { margin: 0; padding: 0; }
        .pad {
            height: 50px;
            width: 50px;
         }
        #first {
            border: 2px solid;
            background-color: rgb(225, 30, 76);
        }
        #second {
            border: 3px dotted red;
            background-color: rgb(35, 214, 92);
        }
    </style>
</head>
<body>
    <!-- (content + padding + border)= element size, Note: margin will not be counted in
element size  --> <!-- border: 2px solid red; (border: weight type colour)-->
    <div id="first" class="pad">First</div>
    <div id="second" class="pad">Second</div>
</body>
```

Border Property (border radius)

• Usage: Used to create rounded corners for elements.

• Individual Corners: Allows setting different radii for each corner.

• Shorthand: e.g., border-radius: 10px 20px.

```
border-radius: 10px;
```

Border Property (box sizing)

```
<head>
  <title>Box Sizing</title>
  <style>
    * { margin: 0; padding: 0; }
    .pad {
      height: 50px;
      width: 50px;
      padding: 2px;
      margin: 2px;
    }
    #first {
      border: 2px solid;
      border-radius: 10px;
      box-sizing: content-box;
      background-color: rgb(225, 30, 76);
    }
    #second {
      border: 2px dotted red;
      border-radius: 10px;
      box-sizing: border-box;
      background-color: rgb(35, 214, 92);
    }
  </style>
</head>
<body>
  <!-- Div ka height: 50px; width: 50px;
     content-box: (57.2 * 57.2) = 50(height/width)+{2+2}(padding)+{1.6+1.6}(border),
     ye by-default hota h (border 2px= 1.6 me badal gya thoda km ho jata h)

     border-box: (50 * 50) = 42.8(height/width)+{2+2}(padding)+{1.6+1.6}(border),
     (height, width, padding and border usi me aayega)  -->
  <div id="first" class="pad">First</div>
  <div id="second" class="pad">Second</div>
</body>
```

--> **CPBM Model(Content Padding Border Margin)**
--> padding: 5px 10px 15px 20px; (Top Right Bottom Left): C.W
    padding: 5px; means it's the same for all four sides.
    padding: 5px 10px; means (top and bottom =5px) & (Right and lefy =10px)

--> margin: 15px 10px 25px 20px; (Top Right Bottom Left): C.W
    Margin between 2 boxes will be the one which is greater.
    ex:(box1: bottom margin=10px, box2: top margin=20px), then margin between both the boxes=20px

--> border: 2px solid red; (border: weight type color)
    border-radius: 5px 15px 5px 5px; (Top-left Top-right Bottom-right Bottom-left)

--> Element size=content + padding + border, Note: margin will not be counted in element size

Browser Tools:
--> Browser sources will contain all the sources which are required for that page.
--> Browser network will show from where it downloaded all the required files.
--> Browser performance will show the performance of the page.
--> Browser console shows the output.
--> Browser elements show the code.
--> Browser application shows different types of storage.
--> Browser memory shows memory details i.e heap memory, run time memory, memory allocation.

Chapter-5

Display Property (Block / Inline Elements)

Block Elements

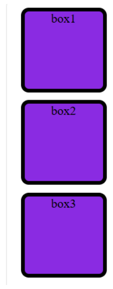
- New Line: Start on a new line.
- Full Width: Take up all horizontal space.
- Styling: Can have margins and padding.
- Size: Width and height can be set.
- Examples: <div>, <p>, <h1>, <ul>, <li>

Inline Elements

- Flow: Stay in line with text.
- Width: Just as wide as the content.
- No Break: No new line between elements.
- Limited Styling: Can't set size easily.
- Examples: <span>, <a>, <strong>, <em>

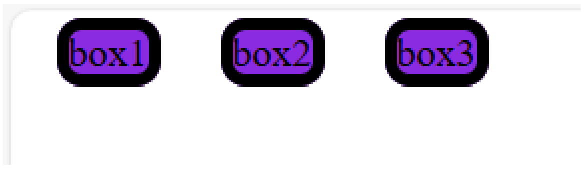
Display Property (Block)

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

   display: block; /* By-default hota hai jiska mtlb ye naye line se shuru hoga as its a block
element*/
}
```



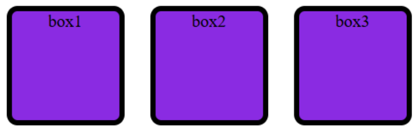
Display Property (Inline)

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

    display: inline;
    /* display: inline; hone ke wajah se div ab ek line me aa rha hai but size jo hm
100px * 100px set kiye hain wo pick nhi kr rha h why ?
    becoz its inline element now and its Styling is Limited: Can't set size easily.
    Iska size content ke size ke according hoga, irrespective ki aap koi height and
width de rhe hain
    Iska solution hai display: inline-block;*/
}
```



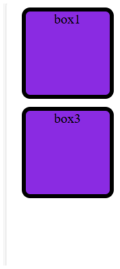
Display Property (Inline-Block)

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

    display: inline-block;
    /* This is the solution of display_inline */
}
```



Display Property (None)

```
.box {
    height: 100px;
    width: 100px;
    background-color: blueviolet;
    margin: 10px;
    text-align: center;
    border: 5px solid black;
    border-radius: 10px;
}
#div2 {
    /* visibility: hidden; */
    display: none;
    /* display: none; Isse invisible to hoga hi hoga saath hi saath apna height aur width
v leke chala jayega */
    /* visibility: hidden; Isse invisible to hoga pr height aur width rhega  */
}
```



--> display: block; By-default hota hai jiska mtlb ye naye line se shuru hoga as its a block element.

--> display: inline; hone ke wajah se div ab ek line me aa rha hai but size jo hm 100px*100px set kiye hain wo pick nhi kr rha h why ?

becoz its inline element now and its Styling is Limited: Can't set size easily.

Iska size content ke size ke according hoga, irrespective ki aap koi height and width de rhe hain. Iska solution hai display: inline-block;

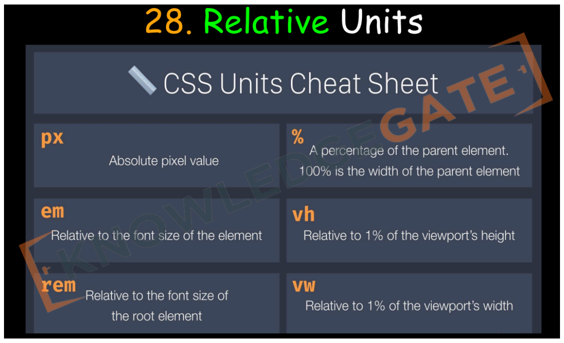
--> display: inline-block; This is the solution of display_inline.

--> display: none; Isse invisible to hoga, saath hi apna height & width v leke chala jayega.

--> visibility: hidden; Isse invisible to hoga pr height aur width rhega.

Responsive Website

1. Adapts layout for different screen sizes
2. Flexible layouts
3. Optimizes images and assets
4. Enhances user experience on mobile and desktop



28. Relative Units

CSS Units Cheat Sheet

px Absolute pixel value	**%** A percentage of the parent element. 100% is the width of the parent element
em Relative to the font size of the element	**vh** Relative to 1% of the viewport's height
rem Relative to the font size of the root element	**vw** Relative to 1% of the viewport's width

Relative Units (Percentage)

• Relative Sizing: Facilitates dynamic sizing relative to parents.

• Adaptability: Ensures responsiveness across various screens.

• Dimensions: Quickly set width and height as a percentage.

```
#first {
   height: 200px;
   width: 200px;
   background-color: aqua;
   font-size: 25px;
}
#second {
   background-color: blueviolet;
   width: 50%;
   height: 30%;
}
<!-- Need to know more: Responsive Website: Relative Units:
1. px: Static unit in pexels
2. %: It is relative to its parent. (If parent is not there then it will treat page size as parent
size) Ex: 40% (40% of its parent)
3. em: Ex: 2.5em (2.5 times of its parent)
4. rem: Ex: 2rem (2 times of its root parent)
   em & rem are only for font size.
5. vh: viewport height, Ex:50vh (50% of page)
6. vw: viewport width, Ex: 70vw (70% of page) - ->
```

Relative Units (EM)

• Relative Unit: Sized relative to the parent element's font size.

• Scalability: Facilitates easy scaling of elements for responsive design.

• Font Sizing: Commonly used for setting font sizes adaptively.

```
body { font-size: 100px;}
#first { height: 200px; width: 200px; background-color: aqua;
font-size: 25px;}
#second {background-color: blueviolet; width: 70%; height: 50%;
    font-size: 2em; /* Twice of my parent i.e 50px */}
```

Relative Units (REM)

• Relative Sizing: Facilitates dynamic sizing relative to the root element.

• Adaptability: Ensures responsiveness across various screens.

• Dimensions: Quickly set width and height as a percentage.

```
* {font-size: 50px;}
#first {
   height: 200px; width: 200px;
   background-color: aqua; font-size: 10px;
}
#second {
   background-color: blueviolet; width: 70%; height: 50%;
   font-size: 2rem; /* mtlb 2 times of root parent i.e 50*2= 100px */
}
```

Relative Units (VW/VH)

• Viewport Relative Units: Units based on viewport's width (vw) or height (vh) for responsive design.

• Responsive Layouts: Essential for creating adaptive layouts; e.g., height: 100vh for full-screen sections.

• Element Sizing: Useful for defining heights and widths that scale with the viewport.

```
   <style>
      #first {
         height: 50vh;
         width: 90vw;
         background-color: red;
      }
   </style>
<body> <div id="first">First</div> </body>
```

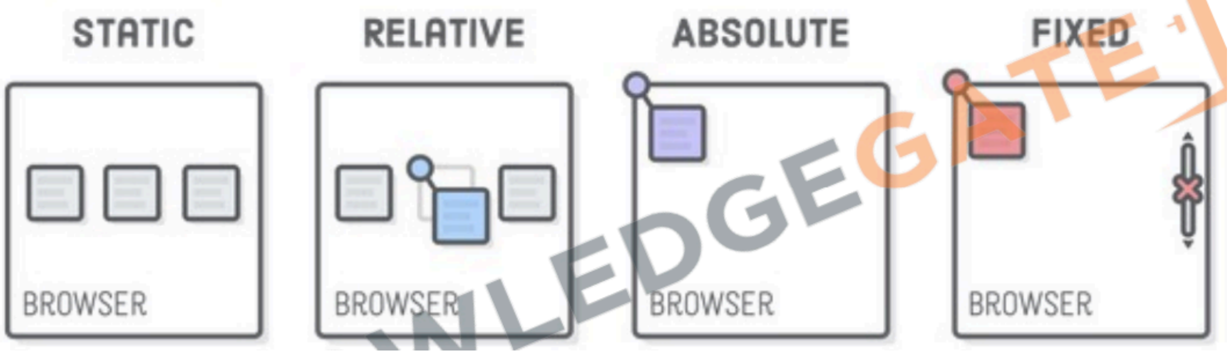
Position Property

• Static (default) : Elements follow the normal document flow. (top, right, bottom, left, z-index would not work)

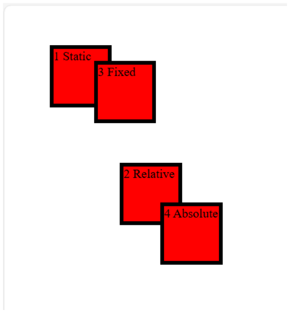
• Relative: Element's position adjusted from its normal position.

• Absolute: Positions element relative to the nearest positioned ancestor.

• Fixed: Element positioned relative to the viewport, does not move on scroll.



```
<head>
    <link rel="stylesheet" href="../../css/level 5/9.positions.css">
</head>
<body>
    <div id="div1">1 Static</div>
    <div id="div2">2 Relative</div>
    <div id="div3">3 Fixed</div>
    <div id="div4">4 Absolute</div>
</body>
<!--Need to know more:
Position Property:
• Static (default) : It's static.
• Relative: Element's relative position from its normal position.
• Absolute: Element's relative position from page or its nearest
positioned ancestor. (ek ke upar ek chadh jate hain)
• Fixed: Element's relative position from viewport(Page), does not move on
scroll. (ek ke upar ek chadh jate hain)-->

div {
    height: 70px; width: 70px; background-color: red;
    border: 5px solid black; margin: 50px;}
#div1 { position: static;}
#div2 { position: relative; top: 20px; left: 90px;}
#div3 { position: fixed; top: 20px; left: 65px;}
#div4 { position: absolute; top: 200px; left: 150px; }
```



Position Property (z index)

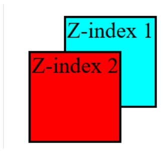
• Stacking Order: Determines the stacking order of elements along the Z-axis.

• Position Context: Only applies to elements with position set to relative, absolute, fixed, or sticky.

• Integer Values: Accepts integer values, including negative numbers.

• Higher Values: An element with a higher z-index value appears above others.

```
<head>
    <title>Z-Index</title>
    <link rel="stylesheet" href="../../css/level 5/10.z-index.css">
</head>
<body>
    <div class="container">
        <div class="box1">Z-index 2</div>
        <div class="box2">Z-index 1</div>
    </div>
</body>

.container { position: relative;}
.box1, .box2 { position: absolute; border: 3px solid black;
   width: 100px; height: 100px; text-align: center; font-size: 25px; }
.box1 { background-color: red;
    left: 20px; top: 60px; z-index: 2; }
.box2 { background-color: aqua;
    left: 60px; top: 20px; z-index: 1; }
/* z-index: mtlb z-axis pe ek ke upar ek, jiska z-index jitna jyada hoga
wo utna upar hoga */
```



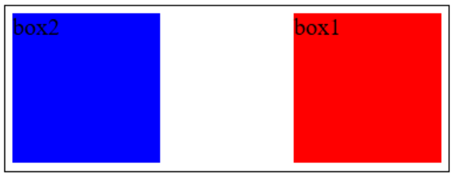
Chapter-6

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.

```
<head>
    <title>Float Property</title>
    <link rel="stylesheet" href="../../css/level 6/1.float.css">
</head>
<body>
    <div class="container">
        <div id="box1" class="box">box1</div>
        <div id="box2" class="box">box2</div>
    </div>
</body>
```

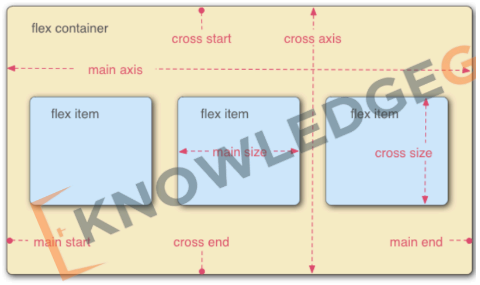
```
.container { height: 110px; width: 300px; border: 1px solid #000; }
.box { width: 100px;  height: 100px;  margin: 5px; }
#box1 { background-color: red; float: right; }
#box2 { background-color: blue; float: left; }
```



What is Flexbox?

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

Flex Model:



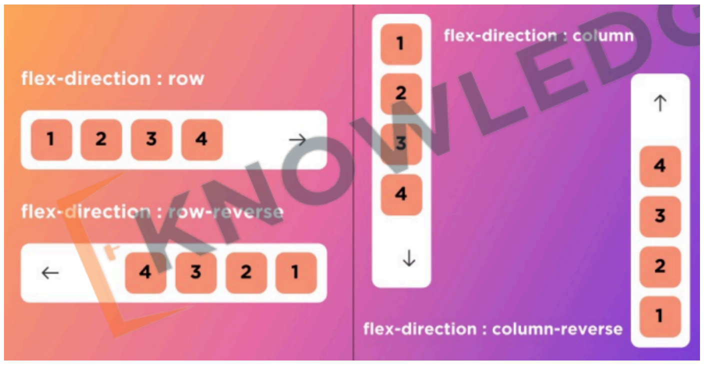
Flexbox Direction

• Property Name: flex-direction is the property used to define the direction in a flex container.

• Row Layout: row value aligns the flex items horizontally, in a left-to-right fashion.

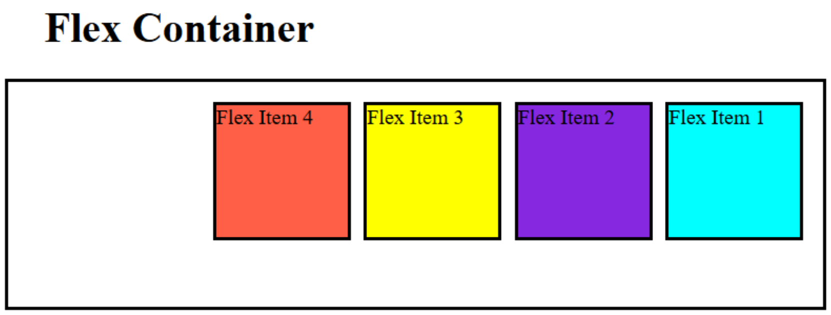
• Column Layout: column value stacks the flex items vertically, from top to bottom.

• Reverse Direction: Adding -reverse to row or column (as in row-reverse or column-reverse) reverses the order of the items.



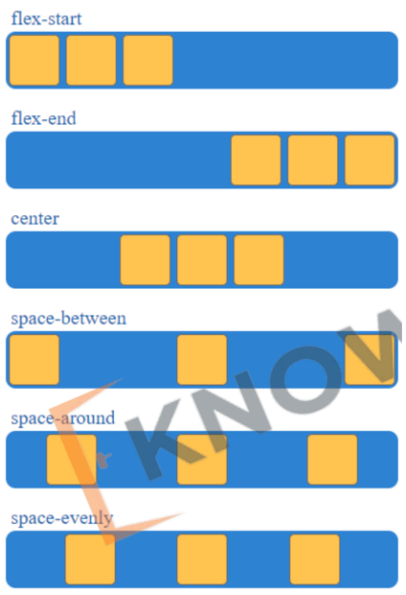
```
<head>
    <title>Flexbox Direction</title>
    <link rel="stylesheet" href="../../css/level
6/2.flexbox_direction.css">
</head>
<body>
    <h1 id="heading">Flex Container</h1>
    <div id="container">
        <div id="box1" class="box">Flex Item 1</div>
        <div id="box2" class="box">Flex Item 2</div>
        <div id="box3" class="box">Flex Item 3</div>
        <div id="box4" class="box">Flex Item 4</div>
    </div>
</body>
<!-- Flexbox: It is a one-dimensional layout method for arranging items in
rows or columns. Flex Items(expand) to fill additional space or shrink to
fit into smaller spaces.
Flex container >> Flex Items (Main axis: Left to right, Cross Axis: Top to
bottom)
flex-direction:row; Left to right
flex-direction:row-reverse; Right to Left
flex-direction:column; Top to bottom
flex-direction:column-reverse; Bottom to top -->
```

```
* {margin: 0; padding: 0;}
.box { height: 100px; width: 100px;
    border: 3px solid black; margin: 5px;}
#heading {margin-left: 50px;}
#container { height: 150px; width: 600px;
  padding: 10px; margin: 20px; border: 3px solid black;
  display: flex; flex-direction: row-reverse;}
#box1 { background-color: aqua;}
#box2 { background-color: blueviolet;}
#box3 { background-color: yellow;}
#box4 { background-color: tomato;}
```



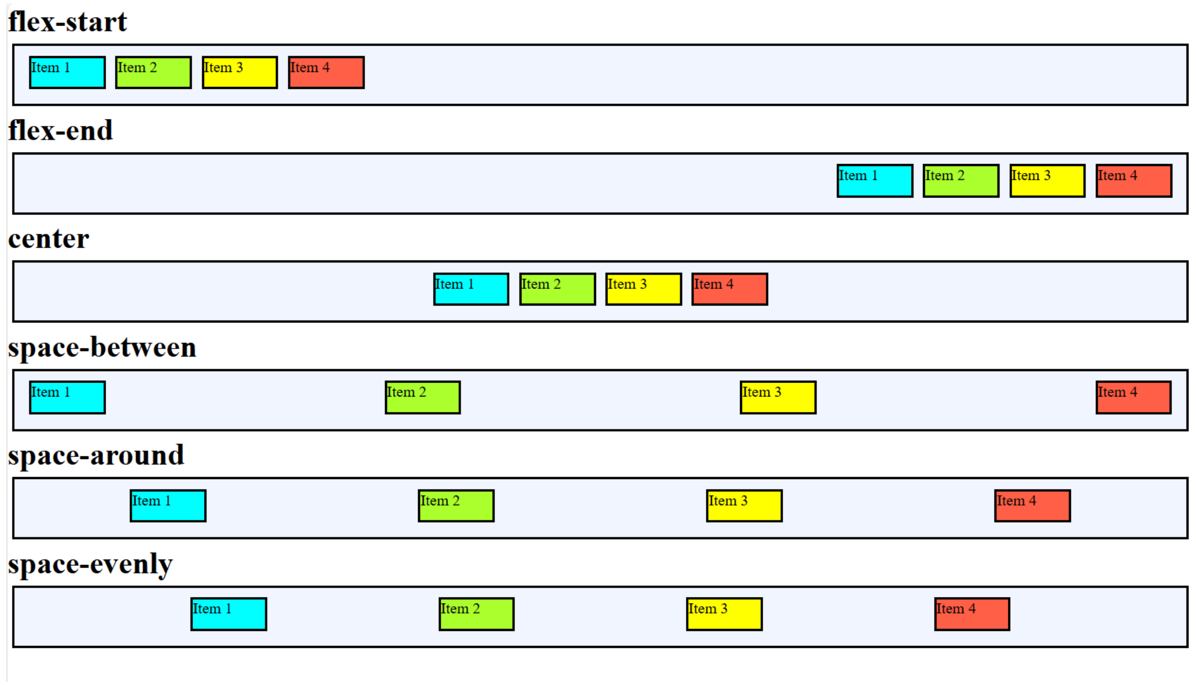
Flexbox container (Justify Content)

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



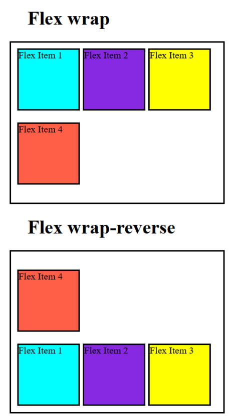
```
 <body>
    <h1 class="heading">flex-start</h1>
    <div id="container_1"  class="container">
<div id="box1" class="box">Item 1</div> <div id="box2" class="box">Item 2
</div>
<div id="box3" class="box">Item 3 </div> <div id="box4" class="box">Item 4
</div></div>
    <h1 class="heading">flex-end</h1>
    <div id="container_2" class="container">
        <div id="box1" class="box">Item 1</div> <div id="box2"
class="box">Item 2</div>
        <div id="box3" class="box">Item 3</div> <div id="box4"
class="box">Item 4</div></div>
        <h1 class="heading">center</h1>
    <div id="container_3" class="container">
        <div id="box1" class="box">Item 1</div> <div id="box2"
class="box">Item 2</div>
        <div id="box3" class="box">Item 3</div> <div id="box4"
class="box">Item 4</div></div>
        <h1 class="heading">space-between</h1>
    <div id="container_4" class="container">
        <div id="box1" class="box">Item 1</div> <div id="box2"
class="box">Item 2</div>
        <div id="box3" class="box">Item 3</div> <div id="box4"
class="box">Item 4</div>
    </div>
    <h1 class="heading">space-around</h1>
    <div id="container_5" class="container">
        <div id="box1" class="box">Item 1</div> <div id="box2"
class="box">Item 2</div>
        <div id="box3" class="box">Item 3</div> <div id="box4"
class="box">Item 4</div></div>
        <h1 class="heading">space-evenly</h1>
    <div id="container_6" class="container">
        <div id="box1" class="box">Item 1</div> <div id="box2"
class="box">Item 2</div>
        <div id="box3" class="box">Item 3</div> <div id="box4"
class="box">Item 4</div>
    </div>
</body>
```

```css
* {margin: 0; padding: 0;}
.box {height: 75px; width: 75px; border: 3px solid black;
    margin-right: 5px; margin-left: 5px;}
.container { height: 100px; width: 1200px; padding: 10px; margin: 5px;
    border: 3px solid black; display: flex; background-color: aliceblue;}
#container_1{ justify-content: flex-start;}
#container_2{ justify-content: flex-end; }
#container_3{ justify-content: center;}
#container_4{ justify-content: space-between; }
#container_5{ justify-content: space-around;}
#container_6{ justify-content: space-evenly; }

#box1 { background-color: aqua;}
#box2 { background-color: greenyellow;}
#box3 { background-color: yellow;}
#box4 { background-color: tomato;}
```



Flexbox container (Flex Wrap)

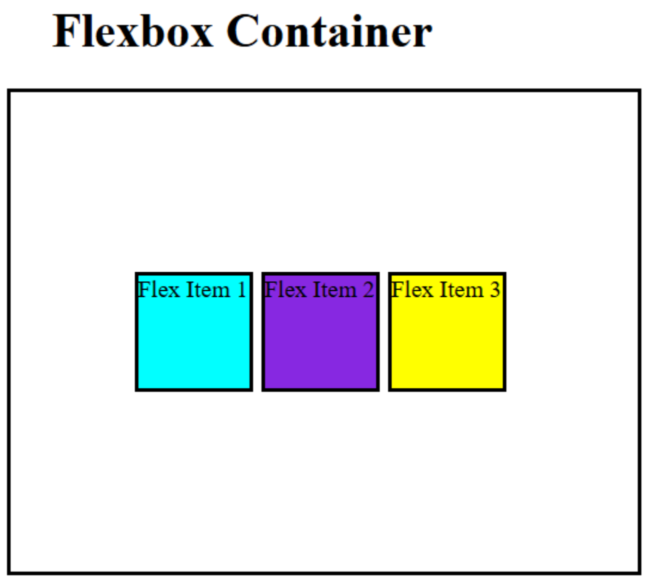
```
<body>
    <h1 id="heading">Flex wrap</h1>
    <div id="hello1" class="container">
        <div id="box1" class="box">Flex Item 1</div>
        <div id="box2" class="box">Flex Item 2</div>
        <div id="box3" class="box">Flex Item 3</div>
        <div id="box4" class="box">Flex Item 4</div>
    </div>
    <h1 id="heading">Flex wrap-reverse</h1>
    <div id="hello2" class="container">
        <div id="box1" class="box">Flex Item 1</div>
        <div id="box2" class="box">Flex Item 2</div>
        <div id="box3" class="box">Flex Item 3</div>
        <div id="box4" class="box">Flex Item 4</div>
    </div>
</body>

* { margin: 0; padding: 0;}
.box { height: 100px; width: 100px; border:
      3px solid black; margin-right: 5px;}
#heading { margin-top: 10px; margin-left: 50px;}
.container { height: 250px; width: 335px; padding: 10px;
    margin: 20px; border: 3px solid black; display: flex;}
#hello1{flex-wrap: wrap;}
#hello2{flex-wrap: wrap-reverse;}
#box1 { background-color: aqua;}
#box2 { background-color: blueviolet;}
#box3 { background-color: yellow;}
#box4 { background-color: tomato;}
```



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.

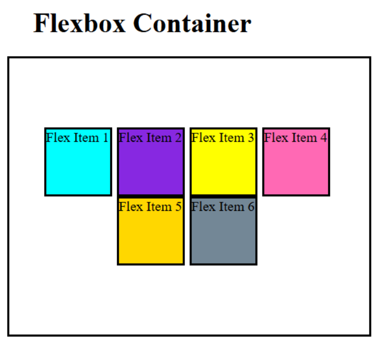
```
<body>
    <h1 id="heading">Flexbox Container</h1>
    <div id="container">
        <div id="box1" class="box">Flex Item 1</div>
        <div id="box2" class="box">Flex Item 2</div>
        <div id="box3" class="box">Flex Item 3</div>
    </div>
</body>
<!-- Flex align items: isse box ko hm cross axis ke along adjust kr skte
hain. -->
* { margin: 0; padding: 0; }
.box { height: 75px; width: 75px;
    border: 3px solid black; margin-right: 5px;}
#heading {margin-left: 50px;}
#container { height: 300px; width: 400px;
    padding: 10px; margin: 20px; border: 3px solid black;
    display: flex; flex-direction: row;
    justify-content: center; align-items: center;}
#box1 { background-color: aqua;}
#box2 { background-color: blueviolet;}
#box3 { background-color: yellow;}
```



Properties: Flexbox container (Align Content)

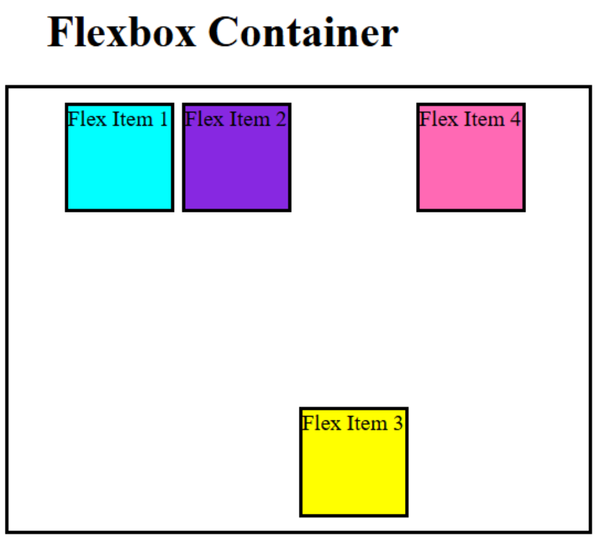
It is utilized to adjust the spacing between flex lines within a flex container, particularly when there is extra space along the cross axis.

```
<body>
    <h1 id="heading">Flexbox Container</h1>
    <div id="container">
        <div id="box1" class="box">Flex Item 1</div>
        <div id="box2" class="box">Flex Item 2</div>
        <div id="box3" class="box">Flex Item 3</div>
        <div id="box4" class="box">Flex Item 4</div>
        <div id="box5" class="box">Flex Item 5</div>
        <div id="box6" class="box">Flex Item 6</div>
    </div>
</body>
<!--Need to know more:
Flex Align Content: Isse cross axis me box ke bich ka space manage kiya
jata hai. - ->
* {margin: 0; padding: 0;}
.box {height: 75px; width: 75px;
   border: 3px solid black; margin-right: 5px; }
#heading {margin-left: 50px;}
#container {height: 300px; width: 400px; padding: 10px;
    margin: 20px; border: 3px solid black;
    display: flex; flex-direction: row; justify-content: center;
    flex-wrap: wrap; align-content: center;}
#box1 { background-color: aqua;}
#box2 { background-color: blueviolet;}
#box3 { background-color: yellow;}
#box4 { background-color: hotpink;}
#box5 { background-color: gold}
#box6 { background-color: lightslategray;}
```



Properties: Flex Items (Align Self)
Allows individual flex items to override the container's align-items property, aligning them differently along the cross-axis.

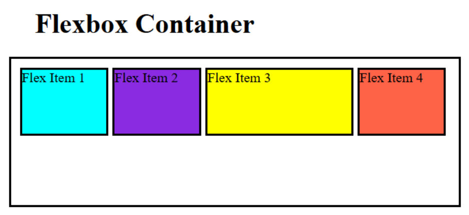
```
<body>
    <h1 id="heading">Flexbox Container</h1>
    <div id="container">
        <div id="box1" class="box">Flex Item 1</div>
        <div id="box2" class="box">Flex Item 2</div>
        <div id="box3" class="box">Flex Item 3</div>
        <div id="box4" class="box">Flex Item 4</div>
    </div>
</body>
* { margin: 0; padding: 0; }
.box { height: 75px; width: 75px; border: 3px solid black;
       margin-right: 5px;}
#heading {margin-left: 50px;}
#container { height: 300px; width: 400px; padding: 10px;
    margin: 20px; border: 3px solid black;
    display: flex; flex-direction: row; justify-content: center;
    flex-wrap: wrap; align-items: start; }
#box1 { background-color: aqua;}
#box2 { background-color: blueviolet;}
#box4 { background-color: hotpink;}
#box3 {
    background-color: yellow;
    align-self: end;
}
```



Flex Items (Flex Shrink) (Flex Grow)

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

```
<body>
    <h1 id="heading">Flexbox Container</h1>
    <div id="container">
        <div id="box1" class="box">Flex Item 1</div>
        <div id="box2" class="box">Flex Item 2</div>
        <div id="box3" class="box">Flex Item 3</div>
        <div id="box4" class="box">Flex Item 4</div>
    </div>
</body>
</html>
<!-- Flex Shrink/Grow: Isse kisi particular flex item ko shrink/ grow krwa
skte hain ..jb flex wrap property use na ho rha ho.
wrna flex wrap krenge to flex iteam shrink/grow hoga hi nhi na.
    flex-shrink: 1; normally sbke jaisa shrink hoga
    flex-shrink: 0.5; sb jitna shrink ho rha h usse aadha hoga.
    flex-shrink: 0; ye to shrink hoga hi nhi.
    flex-grow: 4; sb jitna grow ho rha h usse 4 times jyada hoga. →
* { margin: 0; padding: 0; }
.box { height: 75px; width: 75px; border: 3px solid black;
    margin-right: 5px; flex-grow: 1;}
#heading {margin-left: 50px;}
#container { height: 150px; width: 500px; padding: 10px;
    margin: 20px; border: 3px solid black; display: flex;
    justify-content: space-between; }
#box1 { background-color: aqua;}
#box2 { background-color: blueviolet;}
#box3 {
    background-color: yellow;
    flex-grow: 4;
}
#box4 { background-color: tomato;}
```



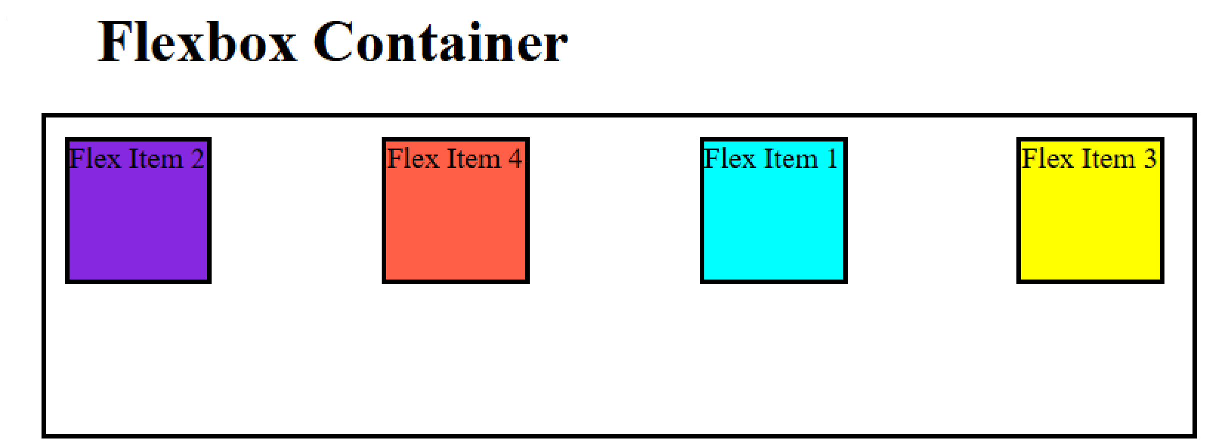
Flexbox Container

Flex Item 1 Flex Item 2 Flex Item 3 Flex Item 4

Properties: Flex Items (Order)

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.

```
<body>
    <h1 id="heading">Flexbox Container</h1>
    <div id="container">
        <div id="box1" class="box">Flex Item 1</div>
        <div id="box2" class="box">Flex Item 2</div>
        <div id="box3" class="box">Flex Item 3</div>
        <div id="box4" class="box">Flex Item 4</div>
    </div>
</body>
<!-- Flex Items (Order)
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. →
* { margin: 0; padding: 0;}
.box { height: 75px; width: 75px; border: 3px solid black;
    margin-right: 5px; }
#heading {margin-left: 50px;}
#container { height: 150px; width: 600px; padding: 10px;
    margin: 20px; border: 3px solid black; display: flex;
    justify-content: space-between; }
#box1 { background-color: aqua; order: 3; }
#box2 { background-color: blueviolet; order: 1; }
#box3 { background-color: yellow; order: 4; }
#box4 { background-color: tomato;  order: 2; }
```



Grid

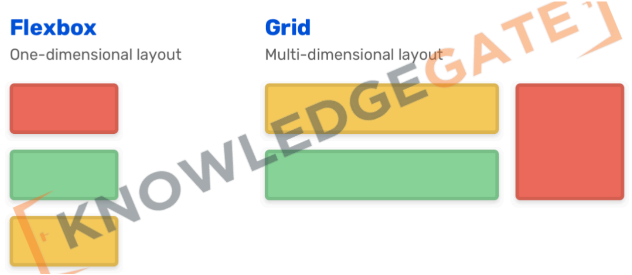
2D layout system for rows & columns.

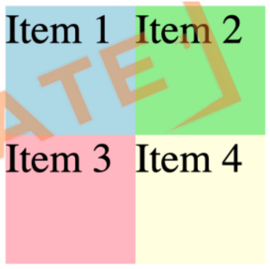
• Activate with display: grid;.

• Children become grid items.

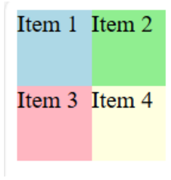
• Define structure with grid-template properties.

• Individual units called grid cells.



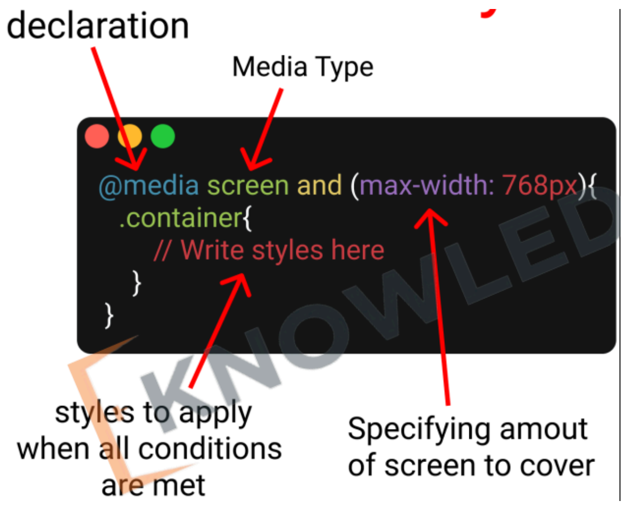


```
<body>
<div class="container">
  <div class="item1">Item 1</div>
  <div class="item2">Item 2</div>
  <div class="item3">Item 3</div>
  <div class="item4">Item 4</div>
</div> </body>
<!-- GRID: 2D layout system for rows & columns. Individual units called
grid cells.
.container {display: grid; grid-template-columns: 50px 50px;
grid-template-rows: 50px 50px;}
.item1 {grid-column: 1 / 2; grid-row: 1 / 2;} -->
.container { display: grid; grid-template-columns: 50px 50px;
    grid-template-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; }
```



Media Queries

- Tailor styles for specific device characteristics.
- Used to create responsive web designs.
- Apply styles based on conditions like screen size.
- Syntax: @media (condition) { CSS rules }.
- Can combine multiple conditions using and, or.



```
@media screen and (max-width: 768px){
  .container{
    // Write styles here
  }
}
```

declaration
Media Type
styles to apply when all conditions are met
Specifying amout of screen to cover

Media Queries (width)

```
<style>
        .box {
            height: 100px;
            width: 100px;
            background-color: rgb(122, 255, 131);
        }
        @media screen and (width: 800px) {
            .box {
                border: 5px solid red;
                background-color: aqua;
            }
        }
    </style>
<body>
    <div class="box"></div>
</body>
```

Media Queries (min-width)

```
<style>
        .box {
            height: 100px;
            width: 100px;
            background-color: lightsalmon;
        }
        @media screen and (min-width: 800px) {
            .box {
                height: 150px;
                width: 150px;
                background-color: aqua;
            }
        }
    </style>
<body>
    <div class="box"></div>
</body>
```

Media Queries (max-width)

```
<style>
        .box {
            height: 100px;
            width: 100px;
            background-color: lightsalmon;
        }
        @media screen and (max-width: 800px) {
            .box {
                height: 50px;
                width: 50px;
                background-color: blue;
            }
        }
    </style>
<body>
    <div class="box"></div>
</body>
```

Media Queries (combination)

```
<style>
        .box { height: 100px; width: 100px; background-color: lightsalmon;}
         @media screen and (min-width: 250px)
          and (max-width: 700px) {
              .box { border-radius: 50%;} }
    </style>
<body>
    <div class="box"></div>
</body>
```

```
--> Flexbox: It is a one-dimensional layout method for arranging items in rows or columns.
    Flex Items(expand) to fill additional space or shrink to fit into smaller spaces.
    Flex container >> Flex Items --> Main axis: Left to right, Cross Axis: Top to bottom
--> ye sara property flex container pe lag rha hai.
     flex-direction:row; Left to right (main axis ke along)
     flex-direction:row-reverse; Right to Left (main axis ke along)
     flex-direction:column; Top to bottom (main axis ke along)
     flex-direction:column-reverse; Bottom to top (main axis ke along)

--> ye sara property flex container pe lag rha hai.
    Flex Justify Content: tb use hota hai jb hmara box container ke size se chhota hota
hai. (main axis ke along)
    Flex wrap: isse box apna size chhota nhi krega wo pure container ke andar wrap ho
jayega. (main axis ke along)
    Flex align items: isse box ko hm cross axis ke along adjust kr skte hain. (cross axis)
    Flex align Content: Isse cross axis me box ke bich ka space manage kiya jata hai.
(cross axis)

--> ye sara property flex items pe lag rha hai flex container pe nhi.
    Flex align self: isse kisi particular flex item ka cross axis me position change kr
skte hain.
    Flex Shrink/Grow: Isse kisi particular flex item ko shrink/ grow krwa skte hain ..jb
flex wrap property use na ho rha ho.
    wrna flex wrap krenge to flex iteam shrink/grow hoga hi nhi na.

--> Flex Items (Order)
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.  This is also flex item propety.

--> GRID: 2D layout system for rows & columns. Individual units called grid cells.

--> @media screen and (width: 800px) {
            .box { border: 5px solid red; background-color: aqua; }
     }@media screen and (min-width: 250px) and (max-width: 700px) {
            .box { border-radius: 50%; } }
```

Chapter-7

Pseudo Classes

• 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.

```
<style>
        .btn {
            height: 20px;
            width: 70px;
            border: 2px solid rgb(98, 255, 0);
            border-radius: 5px;
            background-color: lightcoral;
        }
        .btn:hover {
            height: 25px;
            width: 80px;
            border: 3px solid rgb(10, 0, 0);
        }
        .btn:active {
            height: 25px;
            width: 80px;
            border: 3px solid rgb(228, 105, 142);
            background-color: rgb(92, 205, 147);
        }
    </style>
<body>
    <button class="btn">Click Me</button>
</body>
```

Transitions

CSS transition is a property that enables smooth animation between changes in CSS property values

• transition-property: Defines which CSS properties will transition.

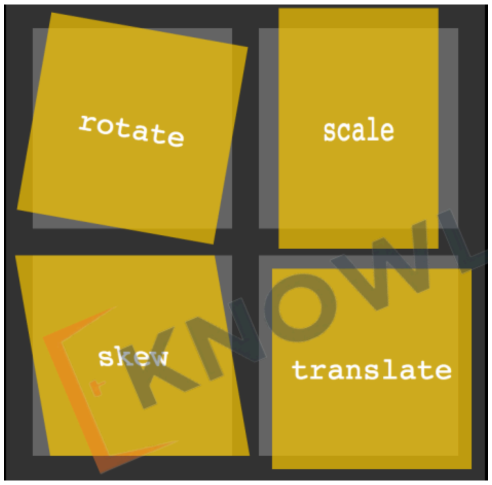
• transition-duration: Sets how long the transition lasts.

• transition-timing-function: Controls the speed curve of the transition. • transition-delay: Specifies a delay before the transition starts.

```
<style>
        .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: 0.2s;
            /*transition: all 1s ease-in-out 1s;*/
            /* transition-timing-function: steps(3); */
        }
        .btn:hover {
            height: 25px;
            width: 80px;
            border: 1px solid red;
        }
        .btn:active {
            height: 25px;
            width: 80px;
            border: 1px solid red;
            background-color: indianred;
        }
    </style>
<body>
    <button class="btn">Click Me</button>
</body>
```

CSS Transform

- Allows modification of an element's shape and position.
- Can perform operations like rotate, scale, and translate.
- Does not affect the layout of surrounding elements.
- Used to create visual effects like 3D space transformations.
- Implemented with functions like rotate(), scale(), and translate().



CSS Transform (Rotate)

- Rotates an element around a fixed point.
- Defined using the rotate() function within the transform property.
- Default rotation point is the element's center.

```
  <style>
  .box { height: 60px; width: 60px; padding: 5px; 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: red;} #box2:hover { transform: rotate(180deg);}
#box3 { background-color: blue;} #box3:hover { transform: rotatex(45deg);}
#box4 { background-color: violet;}#box4:hover {transform: rotatey(45deg);}
#box5 { background-color: pink);}#box5:hover { transform: rotatez(45deg);}
</style>
<body>
    <div id="box1" class="box">45 <sup>deg</sup> rotate</div>
    <div id="box2" class="box">180 <sup>deg</sup>rotate</div>
    <div id="box3" class="box">x 45 <sup>deg</sup>rotate</div>
    <div id="box4" class="box">y 45 <sup>deg</sup>rotate</div>
    <div id="box5" class="box">z 45 <sup>deg</sup>rotate</div>
</body>
```

CSS Transform (Scale)

```
<style>
        .box {
            height: 50px;
            width: 50px;
            padding: 5px;
            margin: 50px;
            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: scale(2); }
        #box2 { background-color: blueviolet;}
        #box2:hover { transform: scale(4); }
        #box3 { background-color: indianred;}
        #box3:hover { transform: scalex(2); }
        #box4 { background-color: violet;}
        #box4:hover { transform: scaley(3); }
        #box5 { background-color: rgb(130, 216, 238);}
        #box5:hover { transform: scaleZ(2) }
    </style>
</head>
<body>
    <div id="box1" class="box">Scale Twice</div>
    <div id="box2" class="box">Scale 4 times</div>
    <div id="box3" class="box">Scale X Twice</div>
    <div id="box4" class="box">Scale Y Twice</div>
    <div id="box5" class="box">Scale Z Twice</div>
</body>
```

CSS Transform (Translate)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Transform Translate</title>
    <style>
        .box {
            height: 75px;
            width: 75px;
            padding: 5px;
            margin: 50px;
            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: translate(50px); }
        #box2 { background-color: blueviolet;}
        #box2:hover { transform: translate(50px, 50px); }
        #box3 { background-color: indianred;}
        #box3:hover { transform: translatex(50px); }
        #box4 { background-color: violet;}
        #box4:hover { transform: translatey(50px); }
    </style>
</head>
<body>
    <div id="box1" class="box">Translate 50px</div>
    <div id="box2" class="box">Translate 50px, 50px</div>
    <div id="box3" class="box">Translate X 50px</div>
    <div id="box4" class="box">Translate Y 50px</div>
</body>
</html>
```

CSS Transform (Skew)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Transform Skew</title>
    <style>
        .box {
            display: inline-block;
            height: 75px;
            width: 75px;
            padding: 5px;
            margin: 50px;
            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: skew(45deg); }
        #box2 { background-color: blueviolet;}
        #box2:hover { transform: skew(90deg); }
    </style>
</head>
<body>
    <div id="box1" class="box">Translate 50px</div>
    <div id="box2" class="box">Translate 50px, 50px</div>
</body>
</html>
--> transition-property: all;
    transition-duration: 1s;
    transition-timing-function: ease-in-out/steps(3);
    transition-delay: 0.2s;
--> transition: all 1s ease-in-out 0.2s; [Shorthand]

transition-property: all; (jo-jo property iske liye defined h sb use kro)
transition-duration: 1s; (Ye 1sec me complete ho jana chahiye)
transition-timing-function: ease-in-out/steps(3);(hone ka tarika kya hoga)
transition-delay: 0.2s; (starting delay kitna hoga)
```

Animation

Animation Properties

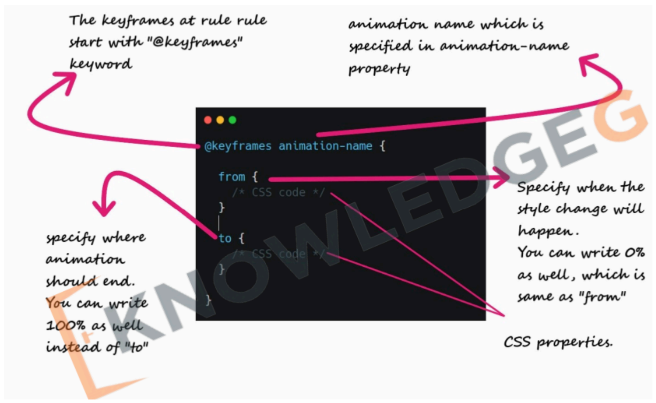
• animation-name: Specifies the name of the @keyframes defined animation. • animation-duration: Defines the total time the animation takes to complete one cycle.

• 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 number of times the animation should repeat.

• animation-direction: Specifies the direction of the animation, allowing for reverse or alternate cycles.



```
@keyframes animation-name {
  from {
    /* CSS code */
  }
  to {
    /* CSS code */
  }
}
```

The keyframes at rule rule start with "@keyframes" keyword
animation name which is specified in animation-name property
Specify when the style change will happen. You can write 0% as well, which is same as "from"
specify where animation should end. You can write 100% as well instead of "to"
CSS properties.
KNOWLEDGEG

```
--> animation-name: ghumakkad;
    animation-duration: 4s;
    animation-timing-function: ease-in-out;
    animation-delay: 0s;
    animation-iteration-count: 4;
    animation-direction: normal/alternate/reverse/alternate-reverse;
--> animation: ghumakkad 4s ease-in-out 0s 4 alternate; [Shorthand]

    animation-name: ghumakkad; (naam rakh do kuchh bhi)
    animation-duration: 4s; (kitna der me animation complete hoga)
    animation-timing-function: ease-in-out; (kis tarah se animate krega)
    animation-delay: 0s; (starting delay kitna hoga)
    animation-iteration-count: 4; (total kitna baar same chij hoga)
    animation-direction:(direction kya hoga)

    @keyframes ghumakkad { from {left: 10px}  to {left: 300px} }
    @keyframes ghumakkad { 0%{left: 10px; top: 0px}
     50%{ left: 150px; top:100px } 100%{left: 300px; top: 0px} }
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Animation</title>
    <style>
        .box {
            height: 75px;
            width: 75px;
            border: 1px solid black;
            border-radius: 5px;
            position: absolute;
            left: 10;
            background-color: lime;

            animation-name: ghumakkad;
            animation-duration: 5s;
            animation-timing-function: ease-in-out;
            animation-delay: 0s;
            animation-iteration-count: 5;

            animation-direction: alternate;
            /* normal/alternate/reverse/alternate-reverse */
            /* animation: ghumakkad 4s ease-in-out 0s 4 alternate; */
        }

        @keyframes ghumakkad {
            from {left: 10px}
            to {left: 300px}
        }
    </style>
</head>
<body>
    <div class="box">Animation</div>
</body>
</html>
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Animation</title>
    <style>
        .box {
            height: 75px;
            width: 75px;
            border: 1px solid black;
            border-radius: 5px;
            position: absolute;
            left: 10;
            background-color: lime;

            animation-name: ghumakkad;
            animation-duration: 4s;
            animation-timing-function: ease-in-out;
            animation-delay: 0s;
            animation-iteration-count: 6;

            animation-direction: alternate;

            /* animation: ghumakkad 4s ease-in-out 0s 4 alternate; */
        }

        @keyframes ghumakkad {
            0% {left: 10px; top: 0px}
            50% { left: 150px; top: 100px }
            100% {left: 300px; top: 0px}
        }
    </style>
</head>
<body>
    <div class="box">Animation</div>
</body>
</html>
```