CSS

CSS is the language we use to style an HTML document.

CSS describes how HTML elements should be displayed.

CSS is the language we use to style a Web page.

## **What is CSS?**

* CSS stands for Cascading Style Sheets
* CSS describes how HTML elements are to be displayed on screen, paper, or in other media
* CSS saves a lot of work. It can control the layout of multiple web pages all at once
* External stylesheets are stored in CSS files

## **Why Use CSS?**

CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

HTML: What to Display

CSS: How to Display

<!DOCTYPE html>

<html>

<head>

<style>

body {

background-color: lightblue;

}

h1 {

color: white;

text-align: center;

}

p {

font-family: verdana;

font-size: 20px;

}

</style>

</head>

<body>

<h1>My First CSS Example</h1>

<p>This is a paragraph.</p>

</body>

</html>

Css describe how HTML elements are to be displayed on screen,paper or in other media

## **CSS Saves a Lot of Work!**

The style definitions are normally saved in external .css files.

With an external stylesheet file, you can change the look of an entire website by changing just one file!

# CSS Syntax

A CSS rule consists of a selector and a declaration block.

## **CSS Syntax**



The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

For example

<!DOCTYPE html>

<html>

<head>

<style>

p {

color: red;

text-align: center;

}

</style>

</head>

<body>

<p>Hello World!</p>

<p>These paragraphs are styled with CSS.</p>

</body>

</html>

#### Example Explained

* p is a selector in CSS (it points to the HTML element you want to style: <p>).
* color is a property, and red is the property value
* text-align is a property, and center is the property value

# How to Add CSS

When a browser reads a style sheet, it will format the HTML document according to the information in the style sheet.

## **Three Ways to Insert CSS**

There are three ways of inserting a style sheet:

* External CSS
* Internal CSS
* Inline CSS

## **External CSS**

With an external style sheet, you can change the look of an entire website by changing just one file!

Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

### Example

External styles are defined within the <link> element, inside the <head> section of an HTML page:

### Example

External styles are defined within the <link> element, inside the <head> section of an HTML page:

<!DOCTYPE html>  
<html>  
<head>  
<link rel="stylesheet" href="mystyle.css">  
</head>  
<body>  
  
<h1>This is a heading</h1>  
<p>This is a paragraph.</p>  
  
</body>  
</html>

An external style sheet can be written in any text editor, and must be saved with a .css extension.

The external .css file should not contain any HTML tags.

Here is how the "mystyle.css" file looks:

### "mystyle.css"

body {  
  background-color: lightblue;  
}  
  
h1 {  
  color: navy;  
}

**Note:** Do not add a space between the property value and the unit:  
Incorrect (space): margin-left: 20 px;  
Correct (nospace): margin-left: 20px;

We can add as many style sheets as we can, Last css may get maximum priority

## **Internal CSS**

An internal style sheet may be used if one single HTML page has a unique style.

The internal style is defined inside the <style> element, inside the head section.

### Example

Internal styles are defined within the <style> element, inside the <head> section of an HTML page:

<!DOCTYPE html>  
<html>  
<head>  
<style>  
body {  
  background-color: linen;  
}  
  
h1 {  
  color: maroon;  
}  
</style>  
</head>  
<body>  
  
<h1>This is a heading</h1>  
<p>This is a paragraph.</p>  
  
</body>  
</html>

## **Inline CSS**

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

### Example

Inline styles are defined within the "style" attribute of the relevant element:

<!DOCTYPE html>  
<html>  
<body>  
  
<h1 style="color:blue;text-align:center;">This is a heading</h1>  
<p style="color:red;">This is a paragraph.</p>  
  
</body>  
</html>

# CSS Selectors

A CSS selector selects the HTML element(s) you want to style.

## **Universal Selector**

\***{}**

The \* selector selects all elements.

**Html page**

<!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="style.css" />

</head>

<body>

<h1>This is the heading</h1>

<p>This is simple and normal paragraph</p>

<button>Click Me!</button>

</body>

</html>

**Style.css**

**\*{**

**color:blue;**

**}**

It will set all the element foreground color to blue

# What is the difference between \* and body tag in css

body is an element selector (selects an element body) while \* is a universal selector (selects all elements). The body selector has higher priority, but the \* selector applies more broadly

\* - Is called universal selector - Selects everything - every element in the document - Can come in handy while applying a common style to every element. body - selects only the body element

**Html file**

<!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="style.css" />

</head>

<body>

This is inside the body tag

<h1>This is the heading</h1>

<p>This is simple and normal paragraph</p>

<button>Click Me!</button>

</body>

</html>

**Style.css**

\*{

    color:blue;

}

body{

    color:green;

}

## **The CSS element Selector**

The element selector selects HTML elements based on the element name.

### Example

Here, all <p> elements on the page will be center-aligned, with a red text color:

p {  
  text-align: center;  
  color: red;  
}

Example

Html page

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

</head>

<body>

    <p>Paragraph 1</p>

    <p>Paragraph 2</p>

    <p>Paragraph 3</p>

    <p>Paragraph 4</p>

    <p>Paragraph 5</p>

</body>

</html>

Mystyle.css

p{

  color:red;

}

We can use multiple selector at same time

   <head>

        <title>Document</title>

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

    </head>

    <body>

        <h1>Heading</h1>

        <p>Paragraph 1</p>

        <p>Paragraph 2</p>

        <p>Paragraph 3</p>

        <p>Paragraph 4</p>

        <p>Paragraph 5</p>

    </body>

style.css

p,h1{

    color:blue;

}

## **The CSS id Selector**

The id selector uses the id attribute of an HTML element to select a specific element.

The id of an element is unique within a page, so the id selector is used to select one unique element!

To select an element with a specific id, 27 a hash (#) character, followed by the id of the element.

### Example

The CSS rule below will be applied to the HTML element with id="para1":

#para1 {  
  text-align: center;  
  color: red;  
}

<!DOCTYPE html>

<html>

<head>

<style>

#para1 {

text-align: center;

color: red;

}

</style>

</head>

<body>

<p id="para1">Hello World!</p>

<p>This paragraph is not affected by the style.</p>

</body>

</html>

**Note:** An id name cannot start with a number!

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

</head>

<body>

    <p>Paragraph 1</p>

    <p>Paragraph 2</p>

    <p id="p3">Paragraph 3</p>

    <p>Paragraph 4</p>

    <p>Paragraph 5</p>

</body>

</html>

Mystyle.css

p{

  color:red;

}

#p3{

  color:blue;

}

## **The CSS class Selector**

The class selector selects HTML elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the class name.

### Example

In this example all HTML elements with class="center" will be red and center-aligned:

.center {  
  text-align: center;  
  color: red;  
}

**Example**

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

</head>

<body>

    <h1 class="green-text">Heading</h1>

    <p>Paragraph 1</p>

    <p class="green-text">Paragraph 2</p>

    <p id="p3">Paragraph 3</p>

    <p>Paragraph 4</p>

    <p class="green-text">Paragraph 5</p>

</body>

</html>

Mystyle.css

p{

  color:red;

}

#p3{

  color:blue;

}

.green-text{

  color:green;

}

**We can use id and css selector at same time**

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

</head>

<body>

    <h1 id="h" class="green-text">Heading</h1>

    <p>Paragraph 1</p>

    <p class="green-text">Paragraph 2</p>

    <p id="p3">Paragraph 3</p>

    <p>Paragraph 4</p>

    <p class="green-text">Paragraph 5</p>

</body>

</html>

**Mystyle.css**

p{

  color:red;

}

#p3{

  color:blue;

}

.green-text{

  color:green;

}

#h{

  font-size:50px;

}

**We can use more than one class in Html file**

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

</head>

<body>

    <h1 id="h" class="green-text underline-h">Heading</h1>

    <p>Paragraph 1</p>

    <p class="green-text">Paragraph 2</p>

    <p id="p3">Paragraph 3</p>

    <p>Paragraph 4</p>

    <p class="green-text">Paragraph 5</p>

</body>

</html>

**Mystyle.css**

p{

  color:red;

}

#p3{

  color:blue;

}

.green-text{

  color:green;

}

#h{

  font-size:50px;

}

.underline-h{

  text-decoration: underline;

}

**An element having all four selector element,class and id and all the selector point to color , then priority order will be**

* **Id**
* **Class**
* **Element**
* **Universal**

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

</head>

<body>

    <h1 id="h" class="green-text">Heading</h1>

    <p>Paragraph 1</p>

    <p class="green-text">Paragraph 2</p>

    <p>Paragraph 3</p>

    <p>Paragraph 4</p>

    <p class="green-text">Paragraph 5</p>

</body>

</html>

**Mystyle.css**

h1{

  color:red;

}

.green-text{

  color:green;

}

#h{

 color:blue;

}

**If we use same selector then order will matter**

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

</head>

<body>

    <h1 id="h" class="green-text">Heading</h1>

    <p>Paragraph 1</p>

    <p class="green-text">Paragraph 2</p>

    <p>Paragraph 3</p>

    <p>Paragraph 4</p>

    <p class="green-text">Paragraph 5</p>

</body>

</html>

**Mystyle.css**

h1{

  color:red;

}

.green-text{

  color:green;

}

#h{

 color:blue;

}

#h{

  color:yellow;

 }

**Now heading will print in yellow colour not in blue colour.**

# CSS Comments

CSS comments are not displayed in the browser, but they can help document your source code.

## **CSS Comments**

Comments are used to explain the code, and may help when you edit the source code at a later date.

Comments are ignored by browsers.

A CSS comment is placed inside the <style> element, and starts with /\* and ends with \*/:

/\* This is a single-line comment \*/  
p {  
  color: red;  
}

You can add comments wherever you want in the code:

### Example

p {  
  color: red;  /\* Set text color to red \*/  
}

Comments can also span multiple lines:

### Example

/\* This is  
a multi-line  
comment \*/  
  
p {  
  color: red;  
}

## **HTML and CSS Comments**

From the HTML ,we learned that we can add comments to your HTML source by using the <!--...--> syntax.

In the following example, we use a combination of HTML and CSS comments:

### Example

<!DOCTYPE html>  
<html>  
<head>  
<style>  
p {  
  color: red; /\* Set text color to red \*/  
}  
</style>  
</head>  
<body>  
  
<h2>My Heading</h2>  
  
<!-- These paragraphs will be red -->  
<p>Hello World!</p>  
<p>This paragraph is styled with CSS.</p>  
<p>CSS comments are not shown in the output.</p>  
  
</body>  
</html>

# Colors in CSS

In CSS Colors can be specified using

* Predefined color names
* **RGB**
* **HEX**
* HSL
* HSLA
* RGBA

Predefined Color Names

* Modern browser support 140 named colors

(Name of these color can be find on below link)

Go to google and type color picker

For color palette visit site

https://colorhunt.co/

# CSS RGB Colors

* rgb(red,green,blue)
* Color value between 0 and 255
* Black:rgb(0,0,0)
* White:rgb(255,255,255)
* Red:rgb(255,0,0)
* Green:rgb(0,255,0)
* Blue: rgb(0,0,2555)
* In CSS, a color can be specified as an RGB value, using this formula:
* Each parameter (red, green, and blue) defines the intensity of the color between 0 and 255.
* For example, rgb(255, 0, 0) is displayed as red, because red is set to its highest value (255) and the others are set to 0.
* To display black, set all color parameters to 0, like this: rgb(0, 0, 0).
* To display white, set all color parameters to 255, like this: rgb(255, 255, 255).

<!DOCTYPE html>

<html>

<body>

<h1>Specify colors using RGB values</h1>

<h2 style="background-color:rgb(255, 0, 0);">rgb(255, 0, 0)</h2>

<h2 style="background-color:rgb(0, 0, 255);">rgb(0, 0, 255)</h2>

<h2 style="background-color:rgb(60, 179, 113);">rgb(60, 179, 113)</h2>

<h2 style="background-color:rgb(238, 130, 238);">rgb(238, 130, 238)</h2>

<h2 style="background-color:rgb(255, 165, 0);">rgb(255, 165, 0)</h2>

<h2 style="background-color:rgb(106, 90, 205);">rgb(106, 90, 205)</h2>

</body>

</html>

# CSS HEX Colors

A hexadecimal color is specified with: #RRGGBB, where the RR (red), GG (green) and BB (blue) hexadecimal integers specify the components of the color.

## **HEX Value**

In CSS, a color can be specified using a hexadecimal value in the form:

**#rrggbb**

Where rr (red), gg (green) and bb (blue) are hexadecimal values between 00 and ff (same as decimal 0-255).

For example, #ff0000 is displayed as red, because red is set to its highest value (ff) and the others are set to the lowest value (00).

To display black, set all values to 00, like this: #000000.

To display white, set all values to ff, like this: #ffffff.

**It is similar to rgb where 0-255 are there value but in HEXCode the value is given between 00 and ff**

<!DOCTYPE html>

<html>

<body>

<h1>Specify colors using HEX values</h1>

<h2 style="background-color:#ff0000;">#ff0000</h2>

<h2 style="background-color:#0000ff;">#0000ff</h2>

<h2 style="background-color:#3cb371;">#3cb371</h2>

<h2 style="background-color:#ee82ee;">#ee82ee</h2>

<h2 style="background-color:#ffa500;">#ffa500</h2>

<h2 style="background-color:#6a5acd;">#6a5acd</h2>

</body>

</html>

## **RGBA Value**

Rgba(reg,green,blue,alpha)

Alpha parameter is a number between 0.0 (fully transparent) and 1.0

Rgb(255,99,71,0.5)

RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.

An RGBA color value is specified with:

**rgba(*red,* *green*, *blue, alpha*)**

The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

Example:

<!DOCTYPE html>

<html>

<body>

<h1>Make transparent colors with RGBA</h1>

<h2 style="background-color:rgba(255, 99, 71, 0);">rgba(255, 99, 71, 0)</h2>

<h2 style="background-color:rgba(255, 99, 71, 0.2);">rgba(255, 99, 71, 0.2)</h2>

<h2 style="background-color:rgba(255, 99, 71, 0.4);">rgba(255, 99, 71, 0.4)</h2>

<h2 style="background-color:rgba(255, 99, 71, 0.6);">rgba(255, 99, 71, 0.6)</h2>

<h2 style="background-color:rgba(255, 99, 71, 0.8);">rgba(255, 99, 71, 0.8)</h2>

<h2 style="background-color:rgba(255, 99, 71, 1);">rgba(255, 99, 71, 1)</h2>

</body>

</html>

# 

**HSL**

* **Hsl(hue,saturation,lightness)**
* **Hue,saturation and lightness(HSL)**
* **Hue is a degree on the color wheel from 0 to 360. 0 is red,120 is green and 240 is blue**
* **Saturation is a percentage value,0% means a shade of gray and 100% is the full color.**
* **Lightness is also a percentage , 0% is black , 50% is neither light or dark ,100% is white**
* **RED: hsl(0,100%,50%)**

### Example

<!DOCTYPE html>

<html>

<body>

<h1>Specify colors using HSL values</h1>

<h2 style="background-color:hsl(0, 100%, 50%);">hsl(0, 100%, 50%)</h2>

<h2 style="background-color:hsl(240, 100%, 50%);">hsl(240, 100%, 50%)</h2>

<h2 style="background-color:hsl(147, 50%, 47%);">hsl(147, 50%, 47%)</h2>

<h2 style="background-color:hsl(300, 76%, 72%);">hsl(300, 76%, 72%)</h2>

<h2 style="background-color:hsl(39, 100%, 50%);">hsl(39, 100%, 50%)</h2>

<h2 style="background-color:hsl(248, 53%, 58%);">hsl(248, 53%, 58%)</h2>

</body>

</html>

### Saturation

Saturation can be described as the intensity of a color.

100% is pure color, no shades of gray.

50% is 50% gray, but you can still see the color.

0% is completely gray; you can no longer see the color.

### Example

<!DOCTYPE html>

<html>

<body>

<h1>HSL Saturation</h1>

<p>The second parameter of hsl() defines the saturation. Less saturation mean less color. 0% is completely gray:</p>

<h2 style="background-color:hsl(0, 100%, 50%);">hsl(0, 100%, 50%)</h2>

<h2 style="background-color:hsl(0, 80%, 50%);">hsl(0, 80%, 50%)</h2>

<h2 style="background-color:hsl(0, 60%, 50%);">hsl(0, 60%, 50%)</h2>

<h2 style="background-color:hsl(0, 40%, 50%);">hsl(0, 40%, 50%)</h2>

<h2 style="background-color:hsl(0, 20%, 50%);">hsl(0, 20%, 50%)</h2>

<h2 style="background-color:hsl(0, 0%, 50%);">hsl(0, 0%, 50%)</h2>

</body>

</html>

### Lightness

The lightness of a color can be described as how much light you want to give the color, where 0% means no light (black), 50% means 50% light (neither dark nor light) and 100% means full lightness (white).

<!DOCTYPE html>

<html>

<body>

<h1>HSL Lightness</h1>

<p>The third parameter of hsl() defines the lightness. 0% means black, and 100% means white:</p>

<h2 style="background-color:hsl(0, 100%, 0%);">hsl(0, 100%, 0%)</h2>

<h2 style="background-color:hsl(0, 100%, 25%);">hsl(0, 100%, 25%)</h2>

<h2 style="background-color:hsl(0, 100%, 50%);">hsl(0, 100%, 50%)</h2>

<h2 style="background-color:hsl(0, 100%, 75%);">hsl(0, 100%, 75%)</h2>

<h2 style="background-color:hsl(0, 100%, 90%);">hsl(0, 100%, 90%)</h2>

<h2 style="background-color:hsl(0, 100%, 100%);">hsl(0, 100%, 100%)</h2>

</body>

</html>

HSLA

* Hsla(hue,saturation,lightness,alpha)
* Alpha parameter is number between 0.0 (fully transparent) and 1.0 (not transparent at all)

<!DOCTYPE html>

<html>

<head>

<style>

#p1 {background-color:hsla(120,100%,50%,0.3);}

#p2 {background-color:hsla(120,100%,75%,0.3);}

#p3 {background-color:hsla(120,100%,25%,0.3);}

#p4 {background-color:hsla(120,60%,70%,0.3);}

#p5 {background-color:hsla(290,100%,50%,0.3);}

#p6 {background-color:hsla(290,60%,70%,0.3);}

</style>

</head>

<body>

<h1>The hsla() Function</h1>

<p>HSL colors with opacity:</p>

<p id="p1">Green</p>

<p id="p2">Light green</p>

<p id="p3">Dark green</p>

<p id="p4">Pastel green</p>

<p id="p5">Violet</p>

<p id="p6">Pastel violet</p>

</body>

</html>

**Practice Set**

1. Create a simple div with an id "box" . Add some text content inside the div .

Set its background color to blue and foreground color to white

2. Create 3 heading with h1,h2,h3 Give them all a class "heading" & set color

of "heading" & set color of "heading" to red , also use element selector

3. Create a button & set its background color to :

green using external stylesheet

blue using <style> tag

pink using inline style

**Text Properties**

text-align

text-align: left/right/center

It change alignment according to our parent tag for example

html page

<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="style.css" />

</head>

<body>

<h1>Heading</h1>

<p>Lorem ipsum dolor, sit amet consectetur adipisicing elit. Eaque voluptatum rerum aspernatur. Pariatur odio libero tenetur! Illo, quidem? Tenetur architecto minima labore molestiae, atque id quasi quam dolorem qui consectetur. Lorem ipsum dolor sit amet consectetur adipisicing elit. Distinctio labore, molestias sapiente repudiandae, sequi nam nulla illum animi sed in quae odio quasi atque. Tempora voluptatem repellendus mollitia eaque officia.

Quas laboriosam maiores veniam et vitae, fugit non suscipit itaque natus asperiores sequi quibusdam! Impedit perspiciatis nemo voluptates accusamus ut nisi illum, tempore temporibus, vero deleniti quaerat, possimus labore numquam.

Officiis accusamus ratione velit eos ex at. Quibusdam illo in deserunt hic quia laborum iure quis id magni nulla tempora excepturi debitis reiciendis, aliquam officiis ea doloribus nemo corporis accusantium.

Cumque ducimus nobis dolores quo rerum assumenda deleniti neque velit officia distinctio ipsam rem modi quod unde ut laboriosam inventore tempore dolorum, numquam asperiores esse animi! Exercitationem iste quidem recusandae?

Maxime nulla fugit ut neque at provident aspernatur vitae, quas facere. Molestias eveniet velit a sapiente necessitatibus ex porro labore, exercitationem quaerat dolor? Delectus voluptatum accusamus sit perferendis sequi nesciunt.

Repellendus impedit officia fugit consequatur perferendis ipsum tempore. Quasi voluptates asperiores pariatur harum ut repellendus tempore esse suscipit quia molestiae. Ad minima ab ullam optio expedita asperiores non qui recusandae.

Enim voluptas consectetur, veniam minus assumenda architecto commodi eligendi voluptatum quas impedit repudiandae amet, modi molestiae nostrum corrupti dolorem! Hic voluptatum molestiae aut, optio commodi esse quibusdam illum totam dolor.

Veniam soluta tempora excepturi ducimus ratione cumque repudiandae, nobis voluptates, eos quaerat, error ipsa velit consequuntur ex hic aliquid pariatur laborum delectus harum. Nam inventore soluta in. Numquam, deleniti eos.

Cupiditate similique doloribus iure exercitationem in vero corporis ipsum ab veritatis minus inventore neque cumque quaerat eius sapiente facilis quo accusantium, quae laborum saepe tenetur nisi laboriosam ullam expedita! Earum!

Aliquid numquam, praesentium expedita soluta iste repudiandae modi mollitia dolorem. Perspiciatis laudantium odio voluptas

tenetur nemo deleniti deserunt omnis? Perspiciatis sed facilis autem labore dolor excepturi ea. Harum, iusto asperiores?</p>

</body>

</html>

Style.css

p{

    text-align:right;

}

h1{

    text-align:center

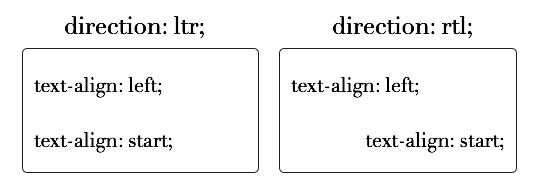
}

Here <h1> (child)is inside the <body> (parent) tag. So text align according to its parent tag i.e. according to body

We can change property to left and center also.

There are two new values in CSS3 as well, start and end. These values make multiple language support easier For example, English is a left-to-right (ltr) language and Arabic is a right-to-left (rtl) language. Using “right” and “left” for values is too rigid and doesn’t adapt as the direction changes. These new values do adapt:

* start – Same as “left” in ltr, same as “right” in rtl.
* end – Same as “right” in ltr, same as “left” in rtl.



# text-decoration

The text-decoration property specifies the decoration added to text

text-decoration:underline/overline/line-through or none

**text-decoration-style**

The text-decoration-style property sets the style of the underline on links and the underline, overline, or line-through on any text with text-decoration applied.

### Values

* solid: the default. Decoration is a single solid line.
* double: Decoration is a pair of solid lines.
* dotted: Decoration is a dotted line.
* dashed: Decoration is a dashed line.
* wavy: Decoration is a wavy line.

### text-decoration as a Shorthand Property

text-decoration can be used in combination with text-decoration-style and text-decoration-color as a shorthand property:

h1 {

text-decoration-line: underline;

text-decoration-style: wavy;

text-decoration-color: red;

/\* can be shortened to \*/

text-decoration: underline wavy red;

}

**Html Page**

<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="style.css" />

</head>

<body>

<h1>Heading</h1>

<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Autem, reprehenderit!</p>

<br/>

<br/>

<a href="https://www.google.com/">Click Me</a>

</body>

</html>

**Style.css**

    p{

        text-align:center;

    }

    h1{

        text-align:center;

       /\* text-decoration: underline; \*/

       /\* text-decoration: overline; \*/

       /\* text-decoration:line-through; \*/

       /\* text-decoration: red underline; \*/

       /\* text-decoration: red wavy underline; \*/

       /\* can be break into two lines:; \*/

       /\* text-decoration-line: underline;

       text-decoration-color: red;

       text-decoration-style:wavy; \*/

        text-decoration: underline wavy red;

    }

    a{

        text-decoration: none;

    }

**Text Properties**

**font-weight**

The font-weight property sets how thick or thin characters in text should be displayed.

**font-weight:**normal/bold/bolder/lighter

**font-weight:**100-900

(lightest shade value is 100 and darkest shade value is 900 and bold is set to value 700 )

**Html file**

<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="style.css" />

</head>

<body>

<h1 id="h1">Heading 1</h1>

<h1 id="h2">Heading 2</h1>

<h1 id="h3">Heading 3</h1>

<h1 id="h4">Heading 4</h1>

<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Autem, reprehenderit!</p>

<br/>

<br/>

<a href="https://www.google.com/">Click Me</a>

</body>

</html>

**Style.css**

    p{

        text-align:center;

    }

    #h1{

        /\* font-weight: normal; \*/

        font-weight:100;

    }

    #h2{

        /\* by default heading are bold \*/

        /\* font-weight: bold; \*/

        font-weight:250;

    }

    #h3{

        /\* font-weight: lighter; \*/

        /\* font-weight:400; \*/

        font-weight:700;

    }

    #h4{

        /\* font-weight: lighter; \*/

        font-weight:900;

    }

    a{

        text-decoration:none;

    }

# Font Family

CSS font-family property is used to set the font face of the text on the webpage

For example

h1 {

font-family: Courier, monospace;

}

***Now inspect the page select the element to check the font of the element***

Html file

<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="style.css" />

</head>

<body>

<h1 id="h1">Heading 1</h1>

<p id="paraone">Lorem ipsum dolor sit amet consectetur adipisicing elit. Autem, reprehenderit! Lorem ipsum dolor sit, amet consectetur adipisicing elit. Iusto eaque eveniet quasi corporis veritatis eius rem ab vel. Esse, ipsum atque nihil reiciendis quidem assumenda quos! Totam praesentium aspernatur enim!

Enim minus necessitatibus rem illo repellat praesentium quibusdam aliquid, nesciunt molestias pariatur nam odio laudantium, error blanditiis! Quo praesentium, autem consequuntur alias voluptates ullam sunt magni qui! Voluptatem, autem impedit.

At doloremque quaerat perferendis blanditiis, dolor ducimus voluptatem, ratione optio aperiam, nesciunt tempora corporis animi vero accusamus placeat ipsum eaque quae delectus eius reprehenderit dicta! Vel ad libero error perferendis?

Exercitationem minus tempora nostrum aperiam consectetur sunt vitae ad voluptatum, autem possimus fugiat error, eius, ipsa iste ipsam. Accusamus quam molestiae laboriosam perspiciatis dolore? Placeat illo alias id sunt hic?

Voluptatum recusandae maxime iste excepturi perferendis quam sit odit modi optio at numquam, nobis dolore magnam blanditiis exercitationem necessitatibus soluta non commodi quasi dolores impedit laudantium quo eaque. Vitae, a.

N</p>

<a href="https://www.google.com/">Click Me</a>

<p id="paratwo">Lorem ipsum dolor sit amet consectetur adipisicing elit. Exercitationem aut ut impedit, dolores vero molestiae eligendi officiis quisquam dignissimos odit mollitia repudiandae, est, commodi voluptatibus. Ex ut commodi eaque illo.

Accusamus doloremque dolorem facilis possimus tempore? Amet sit illum repellendus reprehenderit incidunt facilis delectus, commodi, itaque porro distinctio facere, pariatur aspernatur vero. Quisquam repellendus beatae error! Veniam molestiae doloremque rerum.

Saepe doloremque inventore corrupti harum. Eaque facere rem tempore facilis aut! Inventore et nam non officia sapiente accusamus natus molestias nisi quo quidem, deserunt commodi dicta recusandae? Temporibus, itaque fugit.

Modi corrupti quisquam dolorem repudiandae tempora voluptatum aliquam quia? Repellendus, nemo tempora iste veritatis possimus commodi odio excepturi voluptas minima cum quis quod obcaecati at, illo voluptatum sunt? Nobis, labore!

Id laborum necessitatibus iusto maiores alias impedit non recusandae a mollitia quia eligendi deserunt reiciendis veniam vero ipsum quos, ea error perspiciatis pariatur perferendis consequatur, quaerat quas atque eos? Sapiente.

</p>

</body>

</html>

Style.css

    p{

        text-align:center;

    }

    a{

        text-decoration:none;

    }

    #paraone{

        font-family:Arial;

    }

    #paratwo{

        /\* font-family:'Segoe UI'; \*/

        font-family:'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;'

    }

**Units in CSS**

96px=1 inch(2.54 cm)

font-size:2px;

There are two types of units

Absolute

Relative

The absolute length units are fixed and a length expressed in any of these will appear as exactly that size.

Absolute length units are not recommended for use on screen, because screen sizes vary so much. However, they can be used if the output medium is known, such as for print layout.

|  |  |  |
| --- | --- | --- |
| **Unit** | **Name** | **Explanation** |
| **cm** | Centimeters | It is used to define the measurement in centimeters. |
| **mm** | Millimeters | It is used to define the measurement in millimeters. |
| **in** | Inches | It is used to define the measurement in inches. 1in = 96px = 2.54cm |
| **pt** | Points | It is used to define the measurement in points. 1pt = 1/72 of 1 inch. |
| **pc** | Picas | It is used to define the measurement in picas. 1pc = 12pt so, there 6 picas is equivalent to 1 inch. |
| **px** | Pixels | It is used to define the measurement in pixels. 1px = 1/96th of inch |

We are talking about pixel here let see one example:

**Html Page**

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="style.css" />

    <title>Demo Page</title>

</head>

<body>

    <p id="paraone">Para 1</p>

    <p id="paratwo">Para 2</p>

    <p id="parathree">Para 3</p>

  </body>

  </html>

**style.css**

 #paraone{

    font-size:25px;

 }

 #parathree{

    font-size:10px;

 }

**Text Properties**

**line-height**

It determine the height of the text (amount of space between the line)

line-height:2px

line-height:normal

html file

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="style.css" />

    <title>Demo Page</title>

</head>

<body>

    <p id="paraone">Lorem, ipsum dolor sit amet consectetur adipisicing elit. Atque, nulla labore assumenda nobis officiis fugiat quo facere dolore officia magnam ducimus, mollitia iusto accusantium. Est consequatur culpa provident incidunt explicabo.

    Praesentium corporis aliquid rem molestias quaerat quis facere excepturi? Itaque odio, ratione nisi qui delectus nihil architecto saepe optio eum sunt obcaecati. Laudantium est reprehenderit dolore distinctio provident? Ipsa, laboriosam?</p>

    <p id="paratwo">Para 2</p>

    <p id="parathree">Para 3</p>

  </body>

  </html>

# Style.css

 #paraone{

    font-size:25px;

    /\* line-height: 2px; It overlap all the lines \*/

    /\* line-height:20px; \*/

    /\* line-height:50px; \*/

    /\* line-height:normal ; it is the default property \*/

 }

 #parathree{

    font-size:10px;

 }

# text-transform

# This CSS property allows us to change the case of the text. It is used to control the text capitalization. This CSS property can be used to make the appearance of text in all-lowercase or all-uppercase or can convert the first character of each word to uppercase.

# text-transform: uppercase/lowercase/capitalize/none

**Practice Set 2**

Q1. Create a heading centered on the page with all of its text capitalized by default

Q2. Set the font family of all the content in the document to "Times New Roman"

Q3. Create one div inside another div. Set id & text "outer" for the first one &

"inner" for the second one. Set the outer div text size to 25px & inner div text size

to 10px

**Box Model**

When we inspect(right click browser and select inspec)t each component by hovering each content we visualize each content is a box

**Box Model in Css**

The CSS box model is a container that contains multiple properties including borders, margins, padding, and the content itself. It is used to create the design and layout of web pages. Each content in a web page has height and width and also some time space around the content which is known as padding ,boundary around the content is called border and space between the box is called margin

**Height**

By default,it sets the content area height of the element(by default height is equal to content)

div{

height:50px;

}

**Width**

By default,it sets the content area width of the element

div{

width:50px;

}

Height

Width

Border

Padding

Margin

To check the height and width of the div go to browser and inspect the code select the div it will show div height and width.

**Border**

Use to set an element's border

border-width:2px;

border-style:solid/dotted/dashed

border-color:black;

**Border Shorthand**

For example :

border:2px solid black; (first metioned width,style and color)

**html file**

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="style.css" />

    <title>Demo Page</title>

</head>

<body>

   <h1>Heading</h1>

   <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Soluta, necessitatibus.</p>

   <button>Click Me!</button>

   <div>this is a div.</div>

  </body>

  </html>

**Style.css**

div{

   height:100px;

   background-color:pink;

   width:100px;

   /\* border-width:2px;

   border-style:solid;

   border-color:brown;

   short hand prperty of  \*/

   border:2px solid brown;

   text-align:right;

}

/\* To show height and width of h1 go to browser select inspect

select the h1 element to see height and width of h1 \*/

h1{

   width:200px;

   height:50px;

   background-color:aquamarine;

   border-width:3px;

   border-style:dashed;

   border-color:black;

    text-align:center /\* heading will align according to its width i.e. 200px \*/

}

**Border Radius**

Used to round the corners of an element's outer border

border-radius:10px; (in pixel)

border-radius:50%; (in percentage)

Now create three div

# CSS Padding

**CSS Padding property** is used to define the space between the element content and the element border.

It is different from CSS margin in the way that CSS margin defines the space around elements. CSS padding is affected by the background colors

Top, bottom, left and right padding can be changed independently using separate properties. You can also change all properties at once by using shorthand padding property.

Content will never come in padding area

Whenever we inspect in the browser, blue is the content area

Padding shorthand properties interpret values in the same way:

Clock wise (Top right bottom left)

* One value: All four sides are set uniformly.
* Two values: The first value sets the top and bottom; the second value sets right and left.
* Three values: The first value sets the top; second value sets both the right and left sides; and the third value sets the bottom.
* Four values: Each value sets the top, right, bottom and left, respectively.

For example:

selector {

padding: AllSides;

}

selector {

padding: TopAndBottom RightAndLeft;

}

selector {

padding: Top RightAndLeft Bottom;

}

selector {

padding: Top Right Bottom Left;

}

**Margin**

Margin property is used to define the space around elements. It is completely transparent and doesn't have any background color.

Top, bottom, left and right margin can be changed independently using separate properties. We can also change all properties at once by using shorthand margin property.

Space between two boxes is called margin (area between two boxes border)

In inspect the organe color is considered to be margin area (vary from browser to browser)

## **Margin - Shorthand Property**

To shorten the code, it is possible to specify all the margin properties in one property.

The margin property is a shorthand property for the following individual margin properties:

* margin-top
* margin-right
* margin-bottom
* margin-left

So, here is how it works:

If the margin property has four values:

* **margin: 25px 50px 75px 100px;**
  + top margin is 25px
  + right margin is 50px
  + bottom margin is 75px
  + left margin is 100px

If the margin property has three values:

* **margin: 25px 50px 75px;**
  + top margin is 25px
  + right and left margins are 50px
  + bottom margin is 75px

If the margin property has two values:

* **margin: 25px 50px;**
  + top and bottom margins are 25px
  + right and left margins are 50px

If the margin property has one value:

* **margin: 25px;**
  + all four margins are 25px

**Html File**

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="style.css" />

    <title>Demo Page</title>

</head>

<body>

<!-- To add space between the div we use <br/> tab between the <div></div> -->

<div>div 1</div>

<br/>

<div>div 2</div>

<br/>

<div>div 3</div>

  </body>

  </html>

**Style.css**

div{

   height:100px;

   background-color:pink;

   width:100px;

   border:2px solid brown;

  /\* border-radius:5px;

    border-radius:5px;

   border-radius:15px;

   if we set border-radius:50% and have same height and witdh (square)

   then it will create a circle.

   border-radius:50%;\*/

  /\* padding-left:25px;

    To view the padding in browser select inspect and then select the element

   now the content area decrease blue is the content area

   padding-right:25px;

   padding-bottom:25px;

   padding-top:25px; \*/

   /\* Now apply shorthand property of padding \*/

   /\* padding:25px; \*/

   /\* padding:1px 2px 3px 4px; \*/

   /\* margin-top:50px; \*/

   /\* margin-bottom:50px; \*/

   margin:25px;

   padding:25px;

}

Before any big project developer set universal padding and margin to 0

**Practice Set 3**

Q1. Create a div with height and width of 100px. Set its background color to green

and the border radius to 50%

Q2. Create the following navbar

which have following link:

background color=#0F1111 (black)

foreground color white

amazon.in (25px text) foreground color of amazon.in is #f08804

(all are anchor tags link)

Account

My Cart

Contact Us

gap between each link is 200px

height of the navbar is 60px

after one textbox followed by button (button background color is #f08804 foreground color is white)



**Display Property**

The **Display property** in CSS defines how the components(div, hyperlink, heading, etc) are going to be placed on the web page. As the name suggests, this property is used to define the display of the different parts of a web page.

display:inline/block/inline-block/none

inline - Takes only the space required by the element. (no margin/padding)

block- Takes full space available in width.

inline-block Similar to inline but we can set margin we can set margin & padding.

none :- To remove element from document flow.

For example:

**Html File**

**Style.css**

div{

  background-color: pink;

  display:inline;

  /\* block element become inline \*/

}

button{

    display: block;

    /\* block level element become inline element \*/

}

**display:inline;**

When it comes to margins and padding, browsers treat inline elements differently. You can add space to the left and right on an inline element, but you cannot add height to the top or bottom padding or margin of an inline element.

The display:inline property treats an element as if it were a normal inline element such as <span>. For inline elements, horizontal padding and margins are respected, but the vertical margin and padding is discarded.

When want to element to be treated as inline(i.e. appear in the same line) but apply all the properties (margin,padding,height,width etc) make it inline-block

**Html file**

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" href="style.css">

</head>

<body>

        <h1>Display</h1>

        <div style="background-color: blue;">Box 1</div>

        <div id="box2" style="background-color: green;">Box 2</div>

        <div style="background-color: red;">Box 3</div>

</body>

</html>

**Style.css**

div{

  width:200px;

  height:200px;

  margin: 25px;

padding: 25px;

/\* display:inline; \*/

/\* with this property width,height, and top and bottom margin

will not apply \*/

display:inline-block;

/\* When want to element to be treated as inline(i.e. appear in the same line)

but apply all the properties (margin,padding,height,width etc) make it

inline-block \*/

}

#box2{

    display:none;

}

## **RGBA Colors**

RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.

An RGBA color value is specified with: rgba(red, green, blue, alpha). The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque)

color:rgba(255,0,0,0.5)

color:rgba(255,0,0,1)

With the help of alpha we can change brightness of the color

**Html file**

   <div id="box1">Box 1</div>

        <div id="box2">Box 2</div>

        <div id="box3">Box 3</div>

**Style.css**

div{

  width:200px;

  height:200px;

  margin: 25px;

padding: 25px;

display:inline-block;

}

#box1{

    background-color: rgba(0,0,255,0.25);

}

#box2{

    background-color: rgba(0,0,255,0.50);

}

#box3{

    background-color: rgba(0,0,255,0.75);

}

**Practice Set 4**

Q1: Create a webpage layout with a header,a footer and a content area containing 3 divs. Set the height and width of divs 100px (add the previous navbar in the header) and footer background color should be same as navbar

Content of footer will be like this:

**made with ♥ by Amazon  
© 1996-2023, Amazon.com, Inc. or its affiliates**

and content should be in center

Q2: Add borders to all the divs.

Q3 Add different background to all divs with opacity 0.5 and place all div in a same line

First div contain text “Buy your favourite soap”

Second div contain text “Buy your fav gadgets”

Third div contain text “Rent your fav movies”

Q4: Give content area an appropiate height

## **Relative lengths(Unit)**

Relative units are good to style the responsive site because they scale relative to the window size or the parent. They specify the length, which is relative to another length property.

Depending on the device, if the size of the screen varies too much, then the relative length units are the best because they scale better between the different rendering mediums. We can use the relative units as the default for the responsive units. It helps us to avoid update styles for different screen sizes.

**% :-** It is used to define the measurement as a percentage that is relative to another value. It is often used to define a size as relative to an element's parent object.

For example

width:33%;

margin-left:50%;

For example

<html>

<head>

<link rel="stylesheet" href="style.css">

</head>

<body>

       <div id="box1">

                <div id="box2">

                </div>

       </div>

</body>

</html>

**Style.css**

#box1{

  height:100px;

  width:100px;

  background-color: green;

}

#box2{

  height:50px;

  /\* width:50px; \*/

  width:30%;

  background-color:yellow;

  margin-left:10%

  /\* (10% of total width of the parent( i.e. 100px)

  to check the inspect the box model

  now if we change width of parent box to 200px

  then margin wiil be 20px

  \*/

}

**em**:- It is relative to the font-size of the element. Font size of the parent, in the case of typographical properties like font-size, and font size of the element itself,in the case of other properites like width.

For example:-

Html file

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" href="style.css">

</head>

<body>

       <div id="box1">

        box1

                <div id="box2">

                       box2

                </div>

       </div>

</body>

</html>

Style.css

#box1{

  height:100px;

  width:200px;

  background-color: green;

  font-size:10px;

}

#box2{

  height:50px;

  /\* width:50px; \*/

  /\* width:30%; \*/

  background-color:yellow;

  margin-left:10% ;

  /\* (10% of total width of the parent( i.e. 100px)

  to check the inspect the box model

  now if we change width of parent box to 200px

  then margin wiil be 20px

  \*/

  font-size:2em;

  /\* It means font should be double of font size of parent i.e double of 10 px it gives 20px(font size) \*/

  /\* if we give font-size:.5em this means font will be half of parent \*/

   width:5em;

    /\* here width property treated differently it take size of current font so if we say

    font-size 2em i.e. 20px and   if we give width:5em; then it take 5 time of the current font i.e. 20\*5 =100px

    We conclude in font-size it take the font size of the parent

    and in width it takes its own font size

    \*/

}

**rem :-** It is the font-size of the root element (Root em ) . The root element in this case refers to the html element.

For example

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" href="style.css">

</head>

<body>

       Body Text

       <div id="box1">

        box1

                <div id="box2">

                       box2

                </div>

       </div>

</body>

</html>

Style.css

#box1{

  height:100px;

  width:200px;

  background-color: green;

  font-size:10px;

}

#box2{

  height:50px;

  background-color:yellow;

  margin-left:10% ;

  width:5rem;

  /\* relative to the font size of html i.e. 5 times font size of html which is 16px 5\*16=80  \*/

  /\* inspect the browser and check the body text font it will appear 16px \*/

}

Others

vh stands for viewport height. This unit is based on the height of the viewport. A value of 1vh is equal to 1% of the viewport height. A value of 100vh is equal to 100% of the viewport height.

Html file

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" href="style.css">

</head>

<body>

       <div id="box1">

        box1

                <div id="box2">

                       box2

                </div>

       </div>

</body>

</html>

Style.css

#box1{

  height:100px;

  width:200px;

  background-color: green;

  font-size:10px;

}

#box2{

  height:80vh;

  /\* 1 percent of viewport height (browser) \*/

  /\* now make it 50vh i.e. 50% of viewport height \*/

  /\* now make it 100vh i.e. 100% of viewport height \*/

  background-color:yellow;

  margin-left:10% ;

  width:5rem;

}

vw stands for viewport width. This unit is based on the width of the viewport. A value of 1vw is equal to 1% of the viewport width.

vh:relative to 1% viewport height

vw:relative to 1% viewport width

Sometime we denote viewport to browser

**Position**

The position CSS property sets how an element is positioned in a document

The position property specifies the type of positioning method used for an element (static, relative, absolute, fixed, or sticky).

position: static/relative/absolute/fixed

Four property of position are

1. **Static (Default Position) –normal flow**
2. **Relative**
3. **Fixed**
4. **Absolute**

## **static- default position (The top,right,bottom,left and z-index properties have no effect )**

## **Html file**

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

<div id="parent">

    <div id="one"></div><div id="two"></div><div id="three"></div>

</div>

</body>

</html>

## **position: relative;**

An element with position: relative; is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. **Other content will not be adjusted to fit into any gap left by the element.(no other element occupy the space left by the element, which is positioned relative)**

**relative-element is relative to itself(The top,right,bottom,left, and z-index will work)**

#parent{

    border:1px solid red;

    padding:20px;

    width:50%;

  }

  #one{

    height:100px;

    width:100px;

    background-color: green;

    display:inline-block;

  }

  #two{

    height:100px;

    width:100px;

    background-color: blue;

    display:inline-block;

    position:relative;

    top:50px;

    left:50px;

  }

  #three{

    height:100px;

    width:100px;

    background-color: black;

    display:inline-block;

  }

## **position: absolute;**

An element with position: absolute; is positioned relative to the nearest positioned ancestor (parent is positioned other than static which is default)

However; if an absolute positioned element has no positioned ancestors, it uses the document body (i.e. if its parent is static then it move to its parent (positioned (other then static) otherwise refer to body tag of html), and moves along with page scrolling.

**Note:** Absolute positioned elements are removed from the normal flow( the element will move and other element will occupy its space), and can overlap elements.

Positioned relative to its closest positioned (non-static) ancestor. (removed from the flow)

#parent{

    border:1px solid red;

    padding:20px;

    width:50%;

    margin:100px;

  }

  #one{

    height:100px;

    width:100px;

    background-color: green;

    display:inline-block;

  }

  #two{

    height:100px;

    width:100px;

    background-color: blue;

    display:inline-block;

    position:absolute;

    top:0px;

    left:0px;

   /\* position:relative;

    top:50px;

    left:50px;  \*/

  }

  #three{

    height:100px;

    width:100px;

    background-color: black;

    display:inline-block;

  }

Now make parent div as positional (other than static) div ,make it **relative** (it will not leave its place, i.e. no one will occupy its space )

#parent{

  border:1px solid red;

  padding:20px;

  width:50%;

  margin:100px;

**position: relative;**

}

#one{

  height:100px;

  width:100px;

  background-color: green;

  display:inline-block;

}

#two{

  height:100px;

  width:100px;

  background-color: blue;

  display:inline-block;

  position:absolute;

  top:0px;

  left:0px;

  /\* position:relative;

  top:50px;

  left:50px; \*/

}

#three{

  height:100px;

  width:100px;

  background-color: black;

  display:inline-block;

}

## **position: fixed;**

An element with position: fixed; is positioned relative to the viewport, The **viewport** is the user's visible area of a web page,.which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element. For example sometime navbar is fixed

**It will leave its space and other element occupy the space**

positioned relative to the browser (removed from the flow)

body{

  height:1000px;

}

#parent{

  border:1px solid red;

  padding:20px;

  width:50%;

  margin:100px;

  position: relative;

}

#one{

  height:100px;

  width:100px;

  background-color: green;

  display:inline-block;

}

#two{

  height:100px;

  width:100px;

  background-color: blue;

  display:inline-block;

  position:fixed;

  bottom:10px;

  right:10px;

  /\*top:0px;

  left:0px;

   position:relative;

  top:50px;

  left:50px; \*/

}

#three{

  height:100px;

  width:100px;

  background-color: black;

  display:inline-block;

}

**Sticky** -postioned based on user's scroll position

This CSS property allows the elements to stick when the scroll reaches to a certain point. Depends upon the scroll position, a sticky element toggles in between **fixed** and **relative.** The element will be positioned **relative** until the specified position of offset is met in the viewport. Then, similar to **position: fixed,** the element sticks in one place.

**For example**

**Html page**

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

        <h2>Sticky Element: Scroll Down to See the Effect</h2>

        <p>Scroll down this page to see how  sticky positioning works.</p>

        <div class="sticky">I will stick to the screen when you reach my scroll position</div>

        <p>Some example text..</p>

        <h2>Scroll back up again to "remove" the sticky position.</h2>

**Type lorem\*100 in several p tags**

**Style.css**

.sticky {

  position: sticky;

  top: 25px;

  background-color: yellow;

  padding: 50px;

  font-size: 20px;

}

**Now scroll the text**

The difference between position fixed vs sticky is that fixed always positions an element relative to the viewport, while sticky behaves like a regular element until it reaches the defined offset and then becomes fixed.

# CSS Layers and Z-index

The CSS layers refer to applying the z-index property to elements that overlap with each other. The z-index property is used along with the position property to create an effect of layers. You can specify which element should come on top and which element should come at bottom

# CSS z-index Property

The z-index property specifies the stack order of an element.

An element with greater stack order is always in front of an element with a lower stack order.

**Note:** z-index only works on positioned elements (position: absolute, position: relative, position: fixed)

z-index property does not have maximum and minimum property

## **Stacking context**

A stacking context is a group of elements that have a common parent and move up and down the z axis together.

z-index: auto;

The order is defined by the order in the HTML code:

first in the code = behind

last in the code = in front

Example

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

<div id="parent">

    <div id="one"></div>

    <div id="two"></div>

    <div id="three"></div>

</div>

</body>

</html>

Mystyle.css

body{

  height:1000px;

}

#parent{

  padding:20px;

  width:50%;

  margin:100px;

  position: relative;

}

#one{

  height:100px;

  width:100px;

  background-color: green;

  display:inline-block;

  position: relative;

}

#two{

  height:100px;

  width:100px;

  background-color: blue;

  display:inline-block;

  position: relative;

  top:50px;

  left:-50px;

}

#three{

  height:100px;

  width:100px;

  background-color: black;

  display:inline-block;

  position: relative;

  top:100px;

  left:-100px;

}

**Now apply z-index in each element( and make position property of each box to relative)**

body{

  height:1000px;

}

#parent{

  padding:20px;

  width:50%;

  margin:100px;

  position: relative;

}

#one{

  height:100px;

  width:100px;

  background-color: green;

  display:inline-block;

  position: relative;

  z-index:3;

}

#two{

  height:100px;

  width:100px;

  background-color: blue;

  display:inline-block;

  position: relative;

  top:50px;

  left:-50px;

  z-index:2;

}

#three{

  height:100px;

  width:100px;

  background-color: black;

  display:inline-block;

  position: relative;

  top:100px;

  left:-100px;

  z-index:1;

}

**Background Image**

Used to set an image as background

background-image:url("image.jpeg");

https://unsplash.com/

from this site we get free pics

type animal in search bar and then select free from the dropdown

and open the file and click on download free option

**Background Size**

background-size:cover/contain/auto

**Cover :**It cover all the space and leave no empty space and photo will fit to that area

**Contain:**contain, on the other hand, says to always show the whole image,

even if that leaves a little space to the sides or bottom.(repeat the image)

**auto :**It tells the browser to automatically calculate the size based on the actual size of the image

**For example**

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" href="style.css">

</head>

<body>

       <h1>Position Example</h1>

       <div class="divclass" id="box1">box1</div>

       <div  class="divclass" id="box2">box2</div>

       <div  class="divclass" id="box3">box3</div>

       <div  class="divclass" id="box4">box4</div>

</body>

</html>

**Style.css**

.divclass{

  height:100px;

  width:100px;

  background-color: aqua;

  border:2px solid black;

  display:inline-block;

}

/\* body{

  background-image:url("image/lion.jpg");

  background-size:cover;

} \*/

#box1{

   background-color:red ;

   background-image:url("image/lion.jpg");

  /\* background-size:contain; \*/

  /\* Cover :It cover all the space and leave no empty space and photo will fit to that area \*/

  /\* maximum time we use background-size:cover \*/

  /\* Contain:contain, on the other hand, says to always show the whole image,

  even if that leaves a little space to the sides or bottom.(repeat the image) \*/

  /\* background-repeat: no-repeat; \*/

  background-size:auto;

  /\* auto :It tells the browser to automatically calculate the size based on the actual

  size of the image \*/

}

#box2{

    background-color:green;

 /\* relative- element is relative to itself ( The top,right,bottom,left and z-index will work) \*/

}

#box3{

   background-color:blue;

}

#box4{

  background-color:pink;

 }

**Practice Set 5**

Q. Create the following layout using the given html

Give the div a height,width & some background image.

Use the appropiate position property for the div element to place it at the right end of the page. (The div should not move even on scroll)

<p>lorem \* 5</p>

<div>Love Nature</div>

<p>lorem\*5</p>

# CSS Layout - Overflow

The CSS overflow property controls what happens to content that is too big to fit into an area.

The overflow property specifies whether to clip the content or to add scrollbars when the content of an element is too big to fit in the specified area.

When this happens, it makes the content "overflow" into another area, either horizontally (X-axis) or vertically (Y-axis).

The overflow property has the following values:

* visible - Default. The overflow is not clipped. The content renders outside the element's box
* hidden - The overflow is clipped, and the rest of the content will be invisible
* scroll - The overflow is clipped, and a scrollbar is added to see the rest of the content
* auto - Similar to scroll, but it adds scrollbars only when necessary

Example

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

<div id="overflow">

  <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Laboriosam adipisci ullam repudiandae minus tempore et, dignissimos enim obcaecati possimus quae aperiam? Aut a sequi ipsa, veritatis recusandae in reprehenderit impedit eius rem odit facilis soluta nesciunt nulla provident repudiandae, cumque ullam. Est labore ullam, rem, exercitationem incidunt eaque voluptate dicta itaque ex ipsa vel sapiente facere. Ex voluptate iusto, pariatur quod nihil tempora iure voluptatum nemo commodi nobis qui fugit aut. Architecto ad, expedita dolore iusto illum minima vel nulla et culpa voluptatem doloribus dolores voluptas delectus. Similique animi eligendi vero ullam voluptatem. Sunt vel dolorem incidunt cum fugiat unde culpa architecto vitae, suscipit voluptatem nesciunt laudantium ullam aliquid, impedit distinctio rerum, qui veritatis voluptatum rem voluptate? Veritatis nihil unde totam doloribus ea? Labore ab commodi quasi aliquam at, soluta nostrum veniam est repudiandae accusantium neque praesentium pariatur quis, maiores perspiciatis quia iste sint porro. Beatae voluptates saepe quis architecto delectus dolores iure iste nam excepturi, voluptate ab, ipsum magnam maxime ratione neque error quas deserunt rem, cupiditate ex ad! Inventore, ipsam. Aspernatur unde dolorem consequuntur ipsam! Eveniet recusandae tempora magni cupiditate laudantium nulla qui non, omnis culpa laboriosam earum in fugit odio eius, deserunt, sint ipsa. Repellat, facilis nulla vitae maxime animi, quo voluptatibus, aspernatur saepe recusandae odit enim. Ratione aperiam, animi corrupti at est quo ipsum saepe. Dolores, tempora labore. Aut molestiae harum fuga quis. Laboriosam soluta aliquid nulla? Ex vel itaque exercitationem odit dolorem facere, fuga perspiciatis aspernatur atque corporis ipsa maxime similique? Molestiae accusantium aspernatur facere iure deleniti neque similique quia provident aut reprehenderit perspiciatis deserunt quo possimus nostrum non sint maxime qui alias, quam, tempore consectetur molestias ad? Eligendi saepe nisi quam cumque culpa consequuntur molestias unde. Aliquid fugit quaerat minima provident temporibus? Excepturi explicabo nulla ad ab illum unde provident fugit dignissimos doloribus architecto.

  </p>

</div>

</body>

</html>

Mystyle.css

#overflow{

   background-color: whitesmoke;

   border:1px solid red;

   width:500px;

   height:300px;

   overflow:visible;/\* default property \*/

}

Now property will be hidden

#overflow{

   background-color: whitesmoke;

   border:1px solid red;

   width:500px;

   height:300px;

   overflow:hidden;

}

Now apply scroll bar

#overflow{

   background-color: whitesmoke;

   border:1px solid red;

   width:500px;

   height:300px;

   overflow:scroll;

}

Now make width scroll bar hidden and apply vertical scrollbar

#overflow{

   background-color: whitesmoke;

   border:1px solid red;

   width:500px;

   height:300px;

   overflow-x:hidden;

   overflow-y:scroll;

}

Example

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

<div id="overflow">

  <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Laboriosam adipisci ullam repudiandae minus tempore et, dignissimos enim obcaecati possimus quae aperiam? Aut a sequi ipsa, veritatis recusandae in reprehenderit impedit eius rem odit facilis soluta nesciunt nulla provident repudiandae, cumque ullam. Est labore ullam, rem, exercitationem incidunt eaque voluptate dicta itaque ex ipsa vel sapiente facere. Ex voluptate iusto, pariatur quod nihil tempora iure voluptatum nemo commodi nobis qui fugit aut. Architecto ad, expedita do

  </p>

</div>

</body>

</html>

Mystyle.css

#overflow{

   background-color: whitesmoke;

   border:1px solid red;

   width:500px;

   height:300px;

   overflow-x:hidden;

   overflow-y:scroll;

}

Now the content is less but scrollbar is still there.

Now apply overflow:auto

* auto - Similar to scroll, but it adds scrollbars only when necessary

#overflow{

   background-color: whitesmoke;

   border:1px solid red;

   width:500px;

   height:300px;

  overflow:auto;

}

# CSS Layout - float and clear

The CSS float property specifies how an element should float.

The CSS clear property specifies what elements can float beside the cleared element and on which side.

## **The float Property**

The float property is used for positioning and formatting content e.g. let an image float left to the text in a container.

The float property can have one of the following values:

* left - The element floats to the left of its container
* right - The element floats to the right of its container

In its simplest use, the float property can be used to wrap text around images.

**Example**

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

<p><img src="image/bird.jpg" id="img1" > Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptates est optio culpa necessitatibus eos minus eligendi. Autem itaque dolore eaque libero, dolores iure illo nihil assumenda ratione, ea veritatis? Ad nam, cum, exercitationem sapiente id, itaque sint illo beatae consequatur nisi magnam! Sequi harum totam possimus, ad error corrupti expedita molestiae amet animi, sint commodi in modi velit enim. Similique alias praesentium, voluptate, esse eligendi, expedita temporibus earum quos animi obcaecati voluptatibus cumque libero. Soluta quia ex dolorem! Ad recusandae voluptates, consectetur quaerat doloribus laudantium nam, illum porro deleniti, quasi quae dolorum! Debitis ipsa magnam explicabo eligendi ex harum eum placeat aperiam dolorum nesciunt dolor deserunt doloremque culpa, qui impedit ipsam? Nam sunt non obcaecati commodi officiis accusamus aut aliquam repellat, distinctio odit sed nemo animi! Amet error praesentium maxime excepturi fugiat quia perferendis,<img src="image/img\_forest.jpg" id="img2">placeat sunt voluptatibus repudiandae nisi omnis dicta possimus vero neque atque! Totam et placeat qui quaerat, incidunt quae facere minus ut exercitationem? Hic, corrupti, sequi atque sunt fugiat accusamus sint iure optio quibusdam consequatur aliquid. Expedita consectetur debitis aperiam cumque, maiores ratione nisi modi, quasi a qui vero! Sint hic deleniti dolore excepturi! Maiores deserunt modi deleniti. Minima doloribus repudiandae ducimus fuga error animi ab corporis.</p>

</body>

</html>

**Mystyle.css**

img{

  width:100px;

  height:auto;

}

**Now apply float property to the img1 tag.**

img{

  width:100px;

  height:auto;

}

#img1{

  float:right;

}

**Now apply float property to img2 tag**

img{

  width:100px;

  height:auto;

}

#img1{

  float:right;

}

#img2{

  float:left;

}

**Now apply padding to both the image**

img{

  width:100px;

  height:auto;

  padding:10px;

}

#img1{

  float:right;

}

#img2{

  float:left;

}

**Now making navigational bar**

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

<ul>

    <li>Home</li>

    <li>About Us</li>

    <li>Team</li>

    <li>Contact Us</li>

    <li>Donate</li>

</ul>

</body>

</html>

**Mystyle.css**

ul{

  background-color: green;

  list-style-type: none;

}

li{

  color:white;

  /\* top bottom left right \*/

  padding:5px 10px;

}

**Now apply float:left , when we apply float:left , li come out of the ul tag. To maintain the li we use overflow:auto in ul tag**

ul{

  background-color: green;

  list-style-type: none;

  /\* to fit content into navigation bar\*/

  overflow:auto;

}

li{

  color:white;

  /\* top bottom left right \*/

  padding:10px 15px;

  float:left;

}

**Lets take another example**

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

  <div class="box">

    <div class="item">A</div>

    <div class="item">B</div>

    <div class="item">C</div>

  </div>

</body>

</html>

**Mystyle.css**

.box{

  border:2px solid black;

  overflow:auto;

}

.item

{

  width:100px;

  height:100px;

  margin:2px;

  background-color: aquamarine;

  border:1px solid black;

  float:left;

  /\* Now parent box get collapse as soon as we give float:left  \*/

  /\* to prevent this we give overflow:auto to parent div \*/

}

# CSS clear Property

The clear property controls the flow next to floated elements.

The clear property specifies what should happen with the element that is next to a floating element.

left The element is pushed below left floated elements (remove the element which is floating left)

right The element is pushed below right floated elements (remove the element which is floating right)

both The element is pushed below both left and right floated elements(remove the element from both side left and right)

Example

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

  <div class="container">

    <img src="image/img\_5terre.jpg" id="img2">

    <img src="image/bird.jpg" id="imgb">

    <p>  Lorem ipsum dolor sit amet consectetur adipisicing elit. Maiores quia eos nesciunt perspiciatis, error, quae in repellat nisi ut dicta, id quidem impedit reprehenderit at? Lorem ipsum dolor sit amet consectetur, adipisicing elit. Ea expedita reprehenderit dolorem mollitia. Praesentium error dolores fugit optio est voluptates accusamus tenetur esse cum officiis exercitationem ipsa neque rem, sapiente vero dolore voluptas provident nobis nihil? Blanditiis, cum quasi, ex nostrum animi culpa delectus alias, sapiente odio magni sint voluptatem officiis nihil molestiae odit. Distinctio facere, doloremque expedita hic illo minima, animi numquam quisquam itaque magnam pariatur temporibus facilis placeat delectus dolore praesentium at! Iure quasi, obcaecati esse perspiciatis aut laboriosam quo similique debitis quae quod quam laborum vel hic consectetur. Voluptas earum repellendus fugiat nobis tenetur. Pariatur, maxime. Repellendus!</p>

  </div>

</body>

</html>

Mystyle.css

img{

  width:100px;

  height:auto;

}

#imgb{

  float:right;

  clear:right;

  /\* now there is no element floating to the right of img2 \*/

}

#img2{

  float:right;

}

Clear:right specify element will not be next to float right element instead it will be on next line float:right

Example

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

<div id="content">

    <p>Content</p>

</div>

<div id="sidebar">

    <p>Sidebar</p>

</div>

<div id="footer">

    <p>Footer</p>

</div>

</body>

</html>

**Mystyle.css**

#content{

  width:70%;

  background-color:green;

  height:200px;

  float:left;

}

#sidebar{

  width:30%;

  background-color: indianred;

  height:200px;

  float:left;

}

#footer{

  background-color: yellowgreen;

}

**Now change sidebar height from 200px to 150px**

#content{

  width:70%;

  background-color:green;

  height:200px;

  float:left;

}

#sidebar{

  width:30%;

  background-color: indianred;

  height:150px;

  float:left;

}

#footer{

  background-color: yellowgreen;

}

**This is because the height of the div is small then original div and float left then next div will move below the div.**

**To avoid this we apply clear: both in footer css**

|  |  |
| --- | --- |
| both | The element is pushed below both left and right floated elements |

#content{

  width:70%;

  background-color:green;

  height:200px;

  float:left;

}

#sidebar{

  width:30%;

  background-color: indianred;

  height:150px;

  float:left;

}

#footer{

  background-color: yellowgreen;

**clear:both;**

}

**Lets take another example**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Alignment</title>

    <link rel="stylesheet" href="mystyle.css" />

</head>

<body>

    <div class="container">

        <div id="fruits" class="item">

            <h3>Fruits</h3>

           <p id="fruitspara" class="para">

            Lorem ipsum dolor sit, amet consectetur adipisicing elit. Obcaecati earum quidem maxime, nostrum optio quas autem assumenda nesciunt odio tenetur qui vero nobis reprehenderit quasi error, reiciendis quos ab aliquam. Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptas pariatur accusamus eligendi sunt incidunt mollitia exercitationem aut voluptates sit corporis explicabo, placeat nulla fugit facere, adipisci quae a blanditiis quis ea! Eligendi, debitis repellendus!

           </p>

        </div>

        <div id="computer" class="item">

            <h3>Computer</h3>

           <p id="computerpara" class="para">

            Lorem ipsum dolor sit, amet consectetur adipisicing elit. Obcaecati earum quidem maxime, nostrum optio quas autem assumenda nesciunt odio tenetur qui vero nobis reprehenderit quasi error, reiciendis quos ab aliquam. Lorem ipsum, dolor sit amet consectetur adipisicing elit. Temporibus mollitia aspernatur omnis accusamus at voluptatibus maiores. Rem architecto omnis repudiandae officia eum praesentium esse velit, cum, distinctio sequi consectetur ipsum unde excepturi at fuga! Lorem ipsum dolor sit amet consectetur adipisicing elit. Corporis commodi iste ducimus accusantium voluptate magni ullam totam esse eius, amet a, ipsum ab eaque et aliquam, sunt voluptates assumenda officiis. Architecto voluptates officia rem.

            <!-- now add lorem34 in computer para -->

           </p>

        </div>

        <div id="stationary" class="item">

            <h3>Stationary</h3>

           <p id="stationarypara" class="para">

            Lorem ipsum dolor sit, amet consectetur adipisicing elit. Obcaecati earum quidem maxime, nostrum optio quas autem assumenda nesciunt odio tenetur qui vero nobis reprehenderit quasi error, reiciendis quos ab aliquam.

           </p>

        </div>

    </div>

</body>

</html>

**mystyle.css**

.container{

  width:900px;

  border:3px solid purple;

  margin:33px auto ;

  overflow:auto;

  /\* to keep container in center \*/

}

.item{

  border:3px solid gray;

  margin:12px 3px;

  padding:12px 3px;

  background-color: rgb(228, 207, 207);

}

#fruits{

  float:left;

  width:48%;

}

#computer{

  /\* float:left; \*/

  width:48%;

  /\* now apply float:right:  \*/

  float:right;

}

#stationary{

  /\* float:left; \*/

  /\* clear:left; \*/

  width:100%;

  /\* if we dont apply float:left to the stationary so it get overlapped and that we dont require \*/

  /\* now we want to clear from both side so we say clear both \*/

  clear:both;

}

**Flexbox**

**Html file**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="style.css">

    <title>Document</title>

</head>

<body>

    <h1>Flexbox Example</h1>

    <div id="container">

        <div class="box" id="box1">box1</div>

        <div class="box" id="box2">box2</div>

        <div class="box" id="box3">box3</div>

        <div class="box" id="box4">box4</div>

        <!-- <div id="box5">box5</div> -->

    </div>

</body>

</html>

**style.css**

.box{

  width:200px;

  height:200px;

  border:2px solid brown;

  margin:2px;

  padding:2px;

}

#container{

  background-color:beige ;

  border:5px solid black;

  margin:2px;

  padding:2px;

  /\* height:500px; \*/

  height:1200px;

  display:flex;

  /\* by default flex direction is row

  flex-direction: row;

 display:flex;

   This is the default property of justify-content \*/

  /\* flex-direction: row-reverse; \*/

  /\* flex-direction:column; \*/

  /\* flex-direction: column-reverse; \*/

  /\* flex-wrap:wrap; \*/

  /\* Specifies that the flexible items will wrap if necessary

  (Now when size decrease flex item will not decrease in size if space available ,

  it come to the next line. Otherwise if space not available then flex-item will decrease in size)

  By default property is nowrap;

  \*/

  /\* flex-wrap:wrap-reverse; \*/

     flex-wrap:nowrap;

     /\* justify-content:center; \*/

     /\* flex-items will be placed in center according to main axis in this

     it is row  so we have placed horizontally on the center\*/

  /\* justify-content:flex-start; \*/

     /\* justify-content:start; \*/

  /\* justify-content:flex-end; \*/

  /\* justify-content:end; \*/

  /\* direction does not get reverse \*/

  /\* justify-content:space-between; \*/

  /\* justify-content:space-around; \*/

   /\* justify-content:space-evenly; \*/

   /\*

   To move every item to the center make flex direction:row and justify-content:center \*/

   /\* now lets change direction of flex instead of row it will be column \*/

   /\* flex-direction: column; \*/

   /\* change the height of the container to 1200 and make justify-content: center; \*/

   /\* justify-content: center; \*/

   /\* now make the flex-direction: to row; \*/

   flex-direction:row;

   /\* align-items:center; \*/

   /\* align-items:flex-end; \*/

   /\* align-items: stretch; \*/

   /\* If we want to place vertically center and horizontally center then property will be \*/

   align-items:flex-start;

   justify-content:flex-start;

   /\* flex-wrap:wrap; \*/

   /\* align-content:flex-start; \*/

   /\* align-content:flex-end; \*/

   align-content: center;

   flex-wrap:nowrap;

}

#box1{

  background-color: aqua;

  /\* Now we want that box1 will display last so we give order higher value; \*/

  /\* order:500; \*/

  /\* Now we want that box1 shrink 4 times faster then other box so we give following \*/

  /\* flex-shrink: 4; \*/

  /\* in flex container give flex-wrap:nowrap \*/

  /\* flex-grow:1;

  now change value of flex-grow to 4 \*/

  /\* flex-grow:4; \*/

}

#box2{

  background-color: blue;

  /\* flex-grow:1; \*/

}

#box3{

  background-color: orange;

  /\* flex-grow:1; \*/

}

#box4{

  background-color:brown;

  /\* We want to place box4 at first position so we give order negative initial-value: ; \*/

  /\* order:-1; \*/

  /\* flex-grow:1; \*/

  /\* set width of the flex item to 400 \*/

  flex-basis:400px;

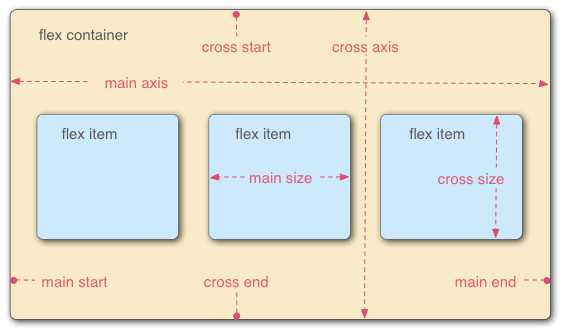
}

**Flexible Box Layout**

It is a one-dimensional layout method for arranging items in rows or columns (It can be either vertical direction or in horizontal direction)

In a container(div,span,navbar,section etc) we put display:flex that container become the flex container. It contains multiple items these items are called flex items. Container contain two direction basically horizontal and vertical

When elements are laid out as flex items, they are laid out along two axes:



* The **main axis** is the axis running in the direction the flex items are laid out in (for example, as a row across the page, or a column down the page.) The start and end of this axis are called the **main start** and **main end**.
* The **cross axis** is the axis running perpendicular to the direction the flex items are laid out in. The start and end of this axis are called the **cross start** and **cross end**.
* The parent element that has display: flex set on is called the **flex container**.

**Flexbox Direction**

It sets how flex items are placed in the flex container, along which axis and direction

flex-direction:row(default)

Now main axis is row and items are arranged from left to right.

flex-direction:row-reverse;

Now main axis is row and items are arranged from right to left

flex-direction:column;

Now main axis is column and items are arranged from top to bottom

flex-direction:column-reverse;

Now main axis is column and items are arranged from bottom to top.

This property is use for flex container not for the individual item.

One property of flex container is that, when we place no of items in the container the items will fit into the container (no matter what is the height width of the items) they will never come out of the flex container.(they adjust their height and width inside the container ) but if content is big or font size is large then item get expand.

**Flex Properties**

**for Flex Container**

flex-wrap:nowrap/wrap/wrap-reverse

flext-wrap:wrap;

Specifies that the flexible items will wrap if necessary (Now when size decrease flex item will not decrease in size(flex-items retains his width) if space available , it come to the next line. Otherwise if space not available then flex-item will decrease in size)

flex-wrap:nowrap;

Specifies that the flexible items will wrap, if necessary, in reverse order(When we decrease the size of the window)

The flex-flow property is a shorthand property for:

* flex-direction
* flex-wrap

flex-flow: row-reverse wrap;

justify-content: alignment along the main axis.

flex-start/flex-end/centre/space-evenly

It will place items according to main axis of the flex-container.

The justify-content property aligns the flexible container's items when the items do not use all available space on the main-axis

flex-start Default value. Items are positioned at the beginning of the container

flex-end: Items are positioned at the end of the container (Direction does not get reverse)

center Items are positioned in the center of the container

space-around Items will have space before, between, and after them(start and end space are half of space between the item)

space-between Items will have space between them(first and last item are placed at the corner(near flex-container) and rest items have equal space )

space-evenly Items will have equal space around them

To move every item to the center make flex direction:row and justify-content:center

**space-around vs space-evenly**

In space-evenly, the empty space in between the flex items is always equal. However, in space-around, only the inner items will have equal spacing in between each other. The first and last item will only be allocated half the spacing.

align-items: alignment along the cross axis

The align-items property specifies the default alignment for items inside a flexbox or grid container.

* In a flexbox container, the flexbox items are aligned on the cross axis, which is vertical by default (opposite of flex-direction)

center Items are positioned at the center of the container( by default vertically)

flex-start Items are positioned at the beginning of the container

flex-end Items are positioned at the end of the container

stretch Items are stretched to fit the container( make sure height of flex-item should not given) by default items are stretched if we don’t give the height.

align-content: This property specifies how flex lines are distributed along the cross axis in a flexbox container. This property is used when flex-wrap: wrap ( so when we try to decrease the size of the screen the flex item wrap (so item go to top and some item go at bottom and keep the gap the between item , to handle gap we use align content property)

flex-start Lines are packed toward the start of the flex container

flex-end Lines are packed toward the end of the flex container

center Lines are packed toward the center of the flex container

# Ordering Flex Items

The **order** property of CSS can be used for ordering flex items. It specifies the order of a flex item with respect to the other flex items. The element has to be a flexible item for the order property to work. The elements are displayed in ascending order of their order values. If two elements have the same order value then they are displayed on the basis of their occurrence in the source code.

Flex items have a default order value of 0, therefore items with an integer value greater than 0 will be displayed after any items that have not been given an explicit order value.

You can also use negative values with order, which can be quite useful. If you want to make one item display first and leave the order of all other items unchanged, you can give that item order of -1. As this is lower than 0 the item will always be displayed first.

**flex-shrink**

It specifies the “flex shrink factor” which determines how much the flex item will shrink relative to the rest of the flex items in the flex container when there isn’t enough space on the row.( It apply on flex-item)

flex-shrink: 2;

When omitted, it is set to 1 and the flex shrink factor is multiplied by the flex basis when distributing negative space.

Flex-shrink:0.5;

Flex-shrink:0; that particular item will not decrease

**flex-grow**

It defines the ability for a flex item to grow if necessary. It accepts a unitless value that serves as a proportion. It dictates what amount of the available space inside the flex container the item should take up.

flex-grow: 2;

For example, if all items have flex-grow set to 1, every child will set to an equal size inside the container. If you were to give one of the children a value of 2, that child would take up twice as much space as the others.

Now set flex-grow of all flex-item to 1 so now all flex-item occupy the whole space of flex-container.

## **The flex-basis Property**

The flex-basis property specifies the initial length of a flex item.

## **The flex Property**

The flex property is a shorthand property for the flex-grow, flex-shrink, and flex-basis properties.

### Example

Make the third flex item not growable (0), not shrinkable (0), and with an initial length of 200 pixels:

<div class="flex-container">  
  <div>1</div>  
  <div>2</div>  
  <div style="flex: 0 0 200px">3</div>  
  <div>4</div>  
</div>

# Differences between flex-basis and width in Flexbox

The **flex-basis** property can be applied only to flex-items and **width** property can be applied to all.

When using flex property, all the three properties of flex-items i.e, **flex-row, flex-shrink**and **flex-basis** can be combined into one declaration, but using width, doing the same thing would require multiple lines of code.

 .flex-container {

            display: flex;

            background: #fff;

            flex-wrap: wrap;

            flex-direction: column;

        }

        .box {

            /\*

      Since flex-direction is set to

      column therefore flex-basis defines

      the height and hence overrides the

      height property. The height of

      flex-item will therefore be 80px.

      The width property has to be separately

      assigned a horizontal width.

      \*/

            height: 100px;

            width: 100px;

            flex-basis: 80px;

        }

When flex-direction is set to

      column therefore flex-basis defines

      the height and hence overrides the

      height property. The height of

      flex-item will therefore be 80px.

      The width property has to be separately

      assigned a horizontal width.

The align-self property specifies the alignment in the block direction for the selected item inside a flexbox (It is basically aliging through cross axis)

center The element is positioned at the center of the container

flex-start The element is positioned at the beginning of the container

flex-end The element is positioned at the end of the container

align self has high priority then flex-container

**Practice Set 6**

Q Create a navbar with 4 options in the form of anchor tags inside list items. Now, use flexbox to place them all space equally in a single line.

Q Use flexbox to center one div inside another div.

Q Which has higher priority -align-items or align-self?

## **CSS Combinators**

A combinator is something that explains the relationship between the selectors.

# Descendant Selector

Descendant Selector is one of the type of Combinators in CSS where the combinators combine 2 selectors in such a way that if an ancestor element matches with the first selector then the elements are matched by the second selector will be selected & these selectors use the descendant Combinator are Descendant Selectors. In simple words, the Descendant Selectors can be any selector having the white-space in between the elements. This selector is used to select all the child elements of the specified tag.

<html>

    <head>

        <title>My Second Page</title>

        <style>

            div p{

                color:blue;

            }

            div p span{

                color:green;

            }

        </style>

    </head>

<body>

    <div>

        <p>Para 1</p>

        <p>Para 2</p>

        <p>Para 3

            <span>Span1</span>

        </p>

        <p>Para 4</p>

    </div>

        <p>Para 5</p>

        <p>Para 6</p>

        <p>Para 7

            <span>span2</span>

        </p>

        <p>Para 8</p>

</body>

</html>

# CSS Box Sizing

The CSS box-sizing property allows us to include the padding and border in an element's total width and height.

Without the CSS box-sizing Property

By default, the width and height of an element is calculated like this:

width + padding + border = actual width of an element

height + padding + border = actual height of an element

This means: When you set the width/height of an element, the element often appears bigger than you have set (because the element's border and padding are added to the element's specified width/height).

With the CSS box-sizing Property

The box-sizing property allows us to include the padding and border in an element's total width and height.

If you set box-sizing: border-box; on an element, padding and border are included in the width and height

For example

<html>

    <head>

        <title>My Second Page</title>

        <style>

            #div1{

               height:150px;

               width:500px;

               background-color: aqua;

            }

            #div2{

                height:150px;

               width:500px;

               background-color: teal;

               padding:20px;

               border:5px solid black;

               box-sizing: border-box;

            }

        </style>

    </head>

<body>

    <div id="div1">

      <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit. Corporis cum eveniet consectetur! Porro officia incidunt quae quod nostrum quos consectetur.</p>

    </div>

    <div id="div2">

        <p>Lorem ipsum dolor, sit amet consectetur adipisicing elit. Odit temporibus amet officiis laborum cum eum? Consequuntur expedita ipsam dolor architecto!</p>

    </div>

</body>

</html>

By default properties of box-sizing:

content-box Default. The width and height properties (and min/max properties) includes only the content. Border and padding are not included

# CSS min-width Property

The min-width property defines the minimum width of an element.

If the content is smaller than the minimum width, the minimum width will be applied.If the content is larger than the minimum width, the min-width property has no effect.

# max-width Property

The max-width property defines the maximum width of an element.

If the content is larger than the maximum width, it will automatically change the height of the element.If the content is smaller than the maximum width, the max-width property has no effect.

**Note:** This prevents the value of the width property from becoming larger than max-width. The value of the max-width property overrides the width property.

For example

<html>

    <head>

        <title>My Second Page</title>

        <style>

            .main{

               min-width:150px;

               max-width:700px;

               background-color: aqua;

               border:1px solid blue;

            }

        </style>

    </head>

<body>

    <div class="main">

       Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptate assumenda repellendus labore ab soluta eveniet sequi doloremque veniam aliquam. Quaerat, tenetur voluptatem repudiandae non itaque voluptates tempore possimus reprehenderit fugit!

    </div>

</body>

</html>

# min-height Property

The min-height property defines the minimum height of an element.If the content of the container grows the height will grow automatically.

# max-height Property

The max-height property defines the maximum height of an element. If the content is larger than the maximum height, it will overflow.

# CSS :hover Selector

Hover is one among the selectors in CSS

Hover is used to select an element when mouse moves over it.

The :hover selector CSS is used to style elements when the mouse hovers over them. It can be used on every element.

<head>

    <style>

    h1:hover {

        color: white;

        background-color: green;

    }

    </style>

</head>

<body>

    <h1 align="center"> hover it</h1>

</body>

</html>

# Create Dropdown Menu using Flexbox

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Flexbox DropDown Menu</title>

    <link rel="stylesheet" href="navbarflex.css">

</head>

<body>

    <nav id="main-nav">

        <ul>

            <li><a href="#">Home</a></li>

            <li><a href="#">Service

                <span class="arrow">&#x25BC</span>

            </a>

                <ul class="submenu">

                    <li><a href="#">Android Dev</a></li>

                    <li><a href="#">IOS Dev</a></li>

                    <li><a href="#">Web Dev

                        <!-- <span class="arrow">&#x25B6</span> -->

                    </a>

                        <!-- <ul class="submenu-2"><li><a href="#">Node Js</a></li>

                        <li><a href="#">Php</a></li>

                        <li><a href="#">Python</a></li></ul> -->

                    </li>

                </ul>

            </li>

            <li><a href="#">Download</a></li>

            <li><a href="#">FAQs

                <span class="arrow">&#x25BC</span>

            </a>

            <ul class="submenu">

                <li><a href="#">Android</a></li>

                <li><a href="#">IOS</a></li>

            </ul>

            </li>

            <li><a href="#">About</a></li>

        </ul>

    </nav>

    <section>

        <h1>Hi! Welcome</h1>

    </section>

</body>

</html>

navbarflex.css

\*{

    margin:0;

    padding:0;

    box-sizing: border-box;

}

#main-nav a{

    color:#f6f7f8;

    background-color: #254441;

    height:64px;

    display:flex;

    font-weight: bold;

    align-items:center;

    justify-content: center;

    text-decoration: none;

}

#main-nav a:hover{

    background-color:#305753;

}

#main-nav ul{

    list-style:none;

    display:flex;

}

#main-nav li{

    width:100%;

    text-align:center;

    position: relative;

}

#main-nav li:hover .submenu > li{

    display:block;

    /\* top:0; \*/

}

/\* .submenu li{

    display:none;

    position: absolute;

    top:0;

} \*/

.submenu{

    display:flex;

    flex-direction: column;

    position: absolute;

    width:100%;

}

.submenu li{

    display:none;

    /\* top:0; \*/

    /\* position: relative; \*/

}

/\* .submenu li:hover .submenu-2 li{

    display:block;

}

.submenu-2 li{

    display:none;

}

.submenu-2{

    display:flex;

    flex-direction:column;

    position:absolute;

    top:0px;

    left:100%;

    width:120px;

} \*/

.arrow{

    font-size:12px;

    color:#eee;

    margin-left:8px;

}

h1{

    font-size:72px;

}

section{

    background-color:#f2f4f3;

    display:flex;

    align-items:center;

    justify-content:center;

    height: 550px;

}

# Backgrounds in CSS

Enter following in html file

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

    <!-- <style>

        p{

            color:blue;

        }

    </style> -->

</head>

<body>

   <div id="bg">

    <p>

Type lorem1000 and then press enter it will create big paragraph and then go to view and then click on word wrap.

**In mystyle.css write following code**

#bg{

  background-color: yellow;

}

body{

  background-color: red;

}

**Now add background image in div**

#bg{

  background-image: url('image/img\_tree.jpg');

}

body{

  background-color: red;

}

**The image is repeating horizontally and vertically**

#bg{

  background-image: url('image/img\_tree.jpg');

  background-repeat:no-repeat; /\* image will not repeat \*/

}

body{

  background-color: red;

}

#bg{

  background-image: url('image/img\_tree.jpg');

  background-repeat:repeat-x; /\* image will repeat in x direction \*/

}

body{

  background-color: red;

}

#bg{

  background-image: url('image/img\_tree.jpg');

  background-repeat:repeat-y; /\* image will repeat in y direction \*/

}

body{

  background-color: red;

}

**How to set background position**

#bg{

  background-image: url('image/img\_tree.jpg');

  background-repeat:no-repeat; /\* image will not repeat \*/

  background-position: right top;

}

body{

  background-color: red;

}

#bg{

  background-image: url('image/img\_tree.jpg');

  background-repeat:no-repeat; /\* image will not repeat \*/

  background-position: right bottom;

}

body{

  background-color: red;

}

#bg{

  background-image: url('image/img\_tree.jpg');

  background-repeat:no-repeat; /\* image will not repeat \*/

  background-position: right center;

}

body{

  background-color: red;

}

#bg{

  background-image: url('image/img\_tree.jpg');

  background-repeat:no-repeat; /\* image will not repeat \*/

  background-position: center center;

}

body{

  background-color: red;

}

#bg{

  background-image: url('image/img\_tree.jpg');

  background-repeat:no-repeat; /\* image will not repeat \*/

  /\* background-position: x axis y axis\*/

  background-position: 50px 100px;

}

body{

  background-color: red;

}

#bg{

  background-image: url('image/img\_tree.jpg');

  background-repeat:no-repeat; /\* image will not repeat \*/

  /\* background-position: x axis in percentage y axis in percentage\*/

  /\* 5% of the width and 10% of the height \*/

  background-position:5% 10%;

}

body{

  background-color: red;

}

**Background size in pixel**

#bg{

  background-image: url('image/img\_tree.jpg');

  background-repeat:no-repeat; /\* image will not repeat \*/

  /\* background-position: x axis in percentage y axis in percentage\*/

  background-position:5% 10%;

  /\* to increase the size of image\*/

  background-size:500px 700px;

}

body{

  background-color: red;

}

#bg{

  background-image: url('image/img\_tree.jpg');

  background-repeat:no-repeat; /\* image will not repeat \*/

  /\* background-position: x axis in percentage y axis in percentage\*/

  background-position:5% 10%;

  /\* to increase the size of image in percentage but proportionally get distorted \*/

  background-size:100% 100%;

}

body{

  background-color: red;

}

# CSS background-attachment Property

The background-attachment property sets whether a background image scrolls with the rest of the page, or is fixed.

#bg{

  background-image: url('image/img\_tree.jpg');

  background-repeat:no-repeat; /\* image will not repeat \*/

  /\* background-position: x axis in percentage y axis in percentage\*/

  background-position:5% 10%;

  /\* to increase the size of image in percentage\*/

  background-size:100% 100%;

  background-attachment: scroll;

height:500px;

}

body{

  background-color: red;

}

#bg{

  background-image: url('image/img\_tree.jpg');

  background-repeat:no-repeat; /\* image will not repeat \*/

  /\* background-position: x axis in percentage y axis in percentage\*/

  background-position:5% 10%;

  /\* to increase the size of image in percentage\*/

  background-size:100% 100%;

  /\* to fixed the background \*/

  background-attachment: fixed;

height:500px;

}

body{

  background-color: red;

}

# CSS background-size Property

The background-size property specifies the size of the background images.

There are four different syntaxes you can use with this property: the keyword syntax ("auto", "cover" and "contain"), the one-value syntax (sets the width of the image (height becomes "auto"), the two-value syntax (first value: width of the image, second value: height)

|  |  |  |
| --- | --- | --- |
| **Value** | **Description** | **Demo** |
| auto | Default value. The background image is displayed in its original size |  |
| cover | Resize the background image to cover the entire container,  even if it has to stretch the image or cut a little bit off one of  the edges | |
| contain | Resize the background image to make sure  the image is fully visible and not get distorted | |

#bg{

  background-image: url('image/img\_tree.jpg');

  background-repeat:no-repeat; /\* image will not repeat \*/

  /\* background-position: x axis in percentage y axis in percentage\*/

  background-position:5% 10%;

  /\* to increase the size of image in percentage\*/

  background-size:cover;

  /\* to fixed the background

  background-attachment: fixed;\*/

}

body{

  background-color: red;

}

#bg{

  background-image: url('image/img\_tree.jpg');

  background-repeat:no-repeat; /\* image will not repeat \*/

  /\* background-position: x axis in percentage y axis in percentage\*/

  background-position:5% 10%;

  /\* to increase the size of image in percentage\*/

  background-size:contain;

  /\* to fixed the background

  background-attachment: fixed;\*/

}

body{

  background-color: red;

}

**Shortcut to the background property**

#bg{

  /\* to increase the size of image in percentage\*/

   background-size:contain;

  /\* to fixed the background

  background-attachment: fixed;\*/

  /\*background: color url background-repeat attachment position;\*/

  background: #ffffff url('image/img\_tree.jpg') no-repeat fixed 5% 10%;

  /\* we can skip any property but order remain the same \*/

}

body{

  background-color: red;

}

# Borders in CSS

Html file

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

    <!-- <style>

        p{

            color:blue;

        }

    </style> -->

</head>

<body>

   <div id="border">

    Lorem ipsum dolor sit amet consectetur, adipisicing elit. Blanditiis optio ducimus, animi possimus veniam ipsa sit dignissimos eius quibusdam iure repellat nemo, neque, illo velit deleniti asperiores recusandae exercitationem totam.

    </div>

</body>

</html>

Mystyle.css

#border{

  border-style:solid;

}

Now run the html code

#border{

  border-style:dotted;

}

#border{

  border-style:double;

}

#border{

  border-style:dashed;

}

#border{

  border-style:groove;

}

(It is just like frame)

#border{

  border-style:outset;

}

#border{

  border-style:inset;

}

#border{

  border-style:ridge;

}

#border{

  /\*border-style:top right bottom left; it goes clock wise \*/

  border-style:solid dashed none dotted;

}

#border{

  /\*border-style:solid; all four side will have solid border \*/

  border-style:solid ;

}

#border{

  /\*border-style:solid dotted; first property top bottom and second property left and right \*/

  border-style:solid dotted ;

}

#border{

  border-style:solid;

  border-width:5px; /\* thickness of border\*/

}

#border{

  border-style:solid;

  /\* border-width:5px 10px define border-width for top and bottom and left and right \*/

  border-width:5px 10px;

}

#border{

  border-style:solid;

  /\* border-width:5px 10px define border-width for top and bottom and left and right \*/

  border-width:5px 10px;

}

**Border Color in Border**

#border{

  border-style:solid;

  /\* border-width:5px 10px 2px 7px ; define width for all sides\*/

  border-width:5px 10px 2px 7px;

  border-color: red;

}

#border{

  border-style:solid;

  /\* border-width:5px 10px 2px 7px ; define width for all sides\*/

  border-width:5px 10px 2px 7px;

  /\* border-color:red yellow; color for top bottom and left right \*/

  border-color: red yellow;

}

#border{

  border-style:solid;

  /\* border-width:5px 10px 2px 7px ; define width for all sides\*/

  border-width:5px 10px 2px 7px;

  /\* border-color:red yellow green blue; color for all sides top right bottom left\*/

  border-color: red yellow green blue;

}

**Shortcut for border**

#border{

  /\*border: border-width border-style border-color; \*/

  border: 5px solid green;

}

#border{

  border-top-style:solid;

  border-top-width :10px;

  border-top-color:red;

}

#border{

 border: 5px solid green;

 border-radius:10px;

}

#border{

 border: 5px solid green;

 border-radius:50%;

}

#border{

 border: 5px solid green;

 /\*border-radius:10px 20px; top bottom left right \*/

 border-radius:10px 20px;

}

#border{

 border: 5px solid green;

 border-top-left-radius:10px;

}

#border{

 border: 5px solid green;

 border-bottom-right-radius:10px;

}

# CSS Height and Width

The CSS height and width properties are used to set the height and width of an element.

Example

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

    <!-- <style>

        p{

            color:blue;

        }

    </style> -->

</head>

<body>

   <div id="div1">

    Lorem ipsum dolor sit amet consectetur, adipisicing elit. Blanditiis optio ducimus, animi possimus veniam ipsa sit dignissimos eius quibusdam iure repellat nemo, neque, illo velit deleniti asperiores recusandae exercitationem totam.

    </div>

</body>

</html>

**mystyle.css**

#div1{

  background-color: blueviolet;

  height:500px;

  width:500px;

}

#div1{

  background-color: blueviolet;

  height:500px;

  width:70%;/\* 70% of the container and container of div is body \*/

}

# max-width

The **max-width** CSS property sets the maximum width of an element. It prevents the used value of the width property from becoming larger than the value specified by max-width.

#div1{

  background-color: blueviolet;

  height:500px;

  width:70%;/\* 70% of the container and container of div is body \*/

  max-width:500px;

 }

#div1{

  background-color: blueviolet;

  height:20%; /\* 20% of the container and container of div is body \*/

  width:70%;/\* 70% of the container and container of div is body \*/

  max-width:500px;

}

body{

    border:1px solid red;

    height:1000px;

}

#div1{

  background-color: blueviolet;

  height:20%; /\* 20% of the container and container of div is body \*/

  width:70%;/\* 70% of the container and container of div is body \*/

  max-width:500px;

}

body{

    border:1px solid red;

    height:1000px;

}

#div1{

  background-color: blueviolet;

  height:50%; /\* 50% of the container and container of div is body \*/

  width:70%;/\* 70% of the container and container of div is body \*/

  max-width:500px;

}

body{

    border:1px solid red;

    height:1000px;

}

#div1{

  background-color: blueviolet;

  height:50%; /\* 20% of the container and container of div is body \*/

  width:70%;/\* 70% of the container and container of div is body \*/

  max-width:500px;

  max-height:300px;

}

body{

    border:1px solid red;

    height:1000px;

}

#div1{

  background-color: blueviolet;

 /\* height:50%;  20% of the container and container of div is body \*/

  width:70%;/\* 70% of the container and container of div is body \*/

  /\* max-width:500px;

  max-height:300px; \*/

  min-height:300px;

}

body{

    border:1px solid red;

    height:1000px;

}

#div1{

  background-color: blueviolet;

 /\* height:50%;  20% of the container and container of div is body \*/

  width:70%;/\* 70% of the container and container of div is body \*/

  /\* max-width:500px;

  max-height:300px; \*/

  min-width:1000px;

  min-height:300px;

}

## **CSS Padding**

Padding is used to create space around an element's content, inside of any defined borders.

The CSS padding properties are used to generate space around an element's content, inside of any defined borders.

With CSS, you have full control over the padding. There are properties for setting the padding for each side of an element (top, right, bottom, and left).

Example

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

    <!-- <style>

        p{

            color:blue;

        }

    </style> -->

</head>

<body>

   <div id="div1">

    <p>

    Lorem ipsum dolor sit amet consectetur, adipisicing elit. Blanditiis optio ducimus, animi possimus veniam ipsa sit dignissimos eius quibusdam iure repellat nemo, neque, illo velit deleniti asperiores recusandae exercitationem totam.

</p>

</div>

</body>

</html>

Mystyle.css

#div1{

    border: 1px solid red;

}

#div1{

    border: 1px solid red;

    padding-top:50px;

}

#div1{

    border: 1px solid red;

    padding-top:50px;

    padding-bottom:20px;

}

#div1{

    border: 1px solid red;

    /\* padding-top:50px;

    padding-bottom:20px;

    padding-left:10%;

    padding-right:20%; \*/

    padding:25px /\* for all four sides \*/

}

#div1{

    border: 1px solid red;

    /\* padding-top:50px;

    padding-bottom:20px;

    padding-left:10%;

    padding-right:20%; \*/

    padding:25px 50px; /\* for top and bottom left and right \*/

}

#div1{

    border: 1px solid red;

    /\* padding-top:50px;

    padding-bottom:20px;

    padding-left:10%;

    padding-right:20%; \*/

    padding:25px 50px 30px 70px; /\* for top right bottom left \*/

}

# CSS Margins

Margins are used to create space around elements, outside of any defined borders.

Example

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

    <!-- <style>

        p{

            color:blue;

        }

    </style> -->

</head>

<body>

   <div class="div1">

    <p>

    Lorem ipsum dolor sit amet consectetur, adipisicing elit. Blanditiis optio ducimus, animi possimus veniam ipsa sit dignissimos eius quibusdam iure repellat nemo, neque, illo velit deleniti asperiores recusandae exercitationem totam.

</p>

</div>

<div class="div1" id="marg">

    <p>

    Lorem ipsum dolor sit amet consectetur, adipisicing elit. Blanditiis optio ducimus, animi possimus veniam ipsa sit dignissimos eius quibusdam iure repellat nemo, neque, illo velit deleniti asperiores recusandae exercitationem totam.

</p>

</div>

<div class="div1">

    <p>

    Lorem ipsum dolor sit amet consectetur, adipisicing elit. Blanditiis optio ducimus, animi possimus veniam ipsa sit dignissimos eius quibusdam iure repellat nemo, neque, illo velit deleniti asperiores recusandae exercitationem totam.

</p>

</div>

</body>

</html>

Mystyle.css

.div1{

    border: 1px solid red;

    /\* padding-top:50px;

    padding-bottom:20px;

    padding-left:10%;

    padding-right:20%; \*/

    padding:25px 50px 30px 70px; /\* for top right bottom left \*/

}

body{

    border:5px dotted black;

}

.div1{

    border: 1px solid red;

    /\* padding-top:50px;

    padding-bottom:20px;

    padding-left:10%;

    padding-right:20%; \*/

    padding:25px 50px 30px 70px; /\* for top right bottom left \*/

}

body{

    border:5px dotted black;

}

#marg{

    margin-top:50px;

}

.div1{

    border: 1px solid red;

    /\* padding-top:50px;

    padding-bottom:20px;

    padding-left:10%;

    padding-right:20%; \*/

    padding:25px 50px 30px 70px; /\* for top right bottom left \*/

}

body{

    border:5px dotted black;

}

#marg{

    margin-top:50px;

    margin-bottom:20px;

    margin-left:60px;

    margin-right:10%;

}

.div1{

    border: 1px solid red;

    /\* padding-top:50px;

    padding-bottom:20px;

    padding-left:10%;

    padding-right:20%; \*/

    padding:25px 50px 30px 70px; /\* for top right bottom left \*/

}

body{

    border:5px dotted black;

}

#marg{

    /\* margin-top:50px;

    margin-bottom:20px;

    margin-left:60px;

    margin-right:10%; \*/

    margin:25px; /\* for each side \*/

}

.div1{

    border: 1px solid red;

    /\* padding-top:50px;

    padding-bottom:20px;

    padding-left:10%;

    padding-right:20%; \*/

    padding:25px 50px 30px 70px; /\* for top right bottom left \*/

}

body{

    border:5px dotted black;

}

#marg{

    /\* margin-top:50px;

    margin-bottom:20px;

    margin-left:60px;

    margin-right:10%; \*/

    margin:25px 15px; /\* for Top bottom and left right \*/

}

.div1{

    border: 1px solid red;

    padding:25px 50px 30px 70px; /\* for top right bottom left \*/

}

body{

    border:5px dotted black;

}

#marg{

    margin:25px 15px 40px 60px; /\* for Top right bottom left \*/

}

In padding we can’t give minus value but in margin we can give negative value.

.div1{

    border: 1px solid red;

    padding:25px 50px 30px 70px; /\* for top right bottom left \*/

}

body{

    border:5px dotted black;

}

#marg{

    margin-top:-30px;

    margin-bottom:20px;

    margin-left:60px;

    margin-right:10%;

}

.div1{

    border: 1px solid red;

    padding:25px 50px 30px 70px; /\* for top right bottom left \*/

}

body{

    border:5px dotted black;

}

#marg{

   width:200px;

   margin:auto /\* to make div in center \*/

}

.div1{

    border: 1px solid red;

    padding:25px 50px 30px 70px; /\* for top right bottom left \*/

}

body{

    border:5px dotted black;

}

#marg{

   width:200px;

   margin:20px auto /\* to make div in center and top and bottom 20 px \*/

}

## **Margin Collapse**

Top and bottom margins of elements are sometimes collapsed into a single margin that is equal to the largest of the two margins.

This does not happen on left and right margins! Only top and bottom margins!

**Example**

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

    <!-- <style>

        p{

            color:blue;

        }

    </style> -->

</head>

<body>

   <div class="div1" id="colsp">

    <p>

    Lorem ipsum dolor sit amet consectetur, adipisicing elit. Blanditiis optio ducimus, animi possimus veniam ipsa sit dignissimos eius quibusdam iure repellat nemo, neque, illo velit deleniti asperiores recusandae exercitationem totam.

</p>

</div>

<div class="div1" id="marg">

    <p>

    Lorem ipsum dolor sit amet consectetur, adipisicing elit. Blanditiis optio ducimus, animi possimus veniam ipsa sit dignissimos eius quibusdam iure repellat nemo, neque, illo velit deleniti asperiores recusandae exercitationem totam.

</p>

</div>

</body>

</html>

**Mystyle.css**

.div1{

    border: 1px solid red;

    padding:25px 50px 30px 70px; /\* for top right bottom left \*/

}

body{

    border:5px dotted black;

}

#marg{

}

#colsp{

}

.div1{

    border: 1px solid red;

    padding:25px 50px 30px 70px; /\* for top right bottom left \*/

}

#marg{

}

#colsp{

margin-bottom:30px;

}

**This is margin collapse**

Top and bottom margins of elements are sometimes collapsed into a single margin that is equal to the largest of the two margins.

This does not happen on left and right margins! Only top and bottom margins!

.div1{

    border: 1px solid red;

    padding:25px 50px 30px 70px; /\* for top right bottom left \*/

}

#marg{

   margin-top:50px;

}

#colsp{

margin-bottom:30px;

}

CSS Text

CSS has a lot of properties for formatting text.

Example

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

   <div id="text">

    <p>

    Lorem ipsum dolor sit amet consectetur, adipisicing elit. Blanditiis optio ducimus, animi possimus veniam ipsa sit dignissimos eius quibusdam iure repellat nemo, neque, illo velit deleniti asperiores recusandae exercitationem totam.

</p>

</div>

</body>

</html>

# CSS text-indent Property

Indent the first line of text with different values:

#text{

    width:500px;

    height:300px;

    border:1px solid black;

    text-indent: 50px;

}

# Letter-spacing

The CSS letter-spacing property **helps developers control the amount of white space between characters**.

#text{

    width:500px;

    height:300px;

    border:1px solid black;

    letter-spacing:10px;

}

# Word-spacing

The word-spacing property increases or decreases the white space between words

#text{

    width:500px;

    height:300px;

    border:1px solid black;

    word-spacing:10px;

}

# CSS white-space Property

The white-space property specifies how white-space inside an element is handled. By default value is normal.

If we want to force the browser to display those line breaks and extra white space characters we can use white-space: pre; It’s called pre because the behaviour is that as if you had wrapped the text in <pre></pre> tags (which by default handle white space and line breaks that way).

Example:

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

   <div id="text">

    <p>

    Lorem ipsum dolor       sit amet consectetur, adipisicing elit. Blanditiis optio ducimus,

    animi possimus veniam ipsa sit dignissimos eius quibusdam iure repellat nemo, neque,

    illo velit deleniti asperiores recusandae exercitationem totam.

</p>

</div>

</body>

</html>

**Mystyle.css**

#text{

    width:500px;

    height:300px;

    border:1px solid black;

    white-space:pre;

}

If we want to prevent the text from wrapping, you can apply white-space: nowrap;

#text{

    width:500px;

    height:300px;

    border:1px solid black;

    white-space:nowrap;

}

# CSS word-wrap Property

Allow long words to be able to break and wrap onto the next line:

Html file

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

   <div id="text">

    <p>

    Lorem ipsum dolor sit amet consectetur adipisicing elit. Maxime quas errorabchhghghhghghghhghhhghghhg deserunt animi. Aliquam voluptatum voluptate exercitationem quis, expedita deleniti atque debitis voluptatem pariatur ex autem. Optio debitis excepturi cum.

</p>

</div>

</body>

</html>

Mystyle.css

#text{

    width:200px;

    height:300px;

    border:1px solid black;

    word-wrap: break-word;

}

#text{

    width:200px;

    height:300px;

    border:1px solid black;

    word-wrap: normal;

}

# CSS text-shadow Property

The text-shadow property adds shadow to text.

Example

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

   <div id="text">

    <p>

    Lorem ipsum dolor sit amet consectetur adipisicing elit. Maxime quas errorabchhghghhghghghhghhhghghhg deserunt animi. Aliquam voluptatum voluptate exercitationem quis, expedita deleniti atque debitis voluptatem pariatur ex autem. Optio debitis excepturi cum.

</p>

</div>

<p id="shadow">lorem ipsum</p>

</body>

</html>

Mystyle.css

#text{

    width:200px;

    height:300px;

    border:1px solid black;

    word-wrap: break-word;

}

#shadow{

    font-size: 70px;

    /\* text-shadow: horizontal distance vertical distance blurr in pixel color of the shadow  \*/

    text-shadow:5px 10px 5px red;

}

# Box Shadow

The CSS box-shadow property is used to apply one or more shadows to an element.In its simplest use, we can only specify a horizontal and a vertical shadow. The default color of the shadow is the current text-color.

Specify a horizontal and a vertical shadow:

div {

box-shadow: 10px 10px;

}

**Specify a color for the shadow:**

div {

box-shadow: 10px 10px lightblue;

}

**Add a Blur Effect to the Shadow**

The blur parameter defines the blur radius. The higher the number, the more blurred the shadow will be.

div {

box-shadow: 10px 10px 5px lightblue;

}

**Set the Spread Radius of the Shadow**

The spread parameter defines the spread radius. A positive value increases the size of the shadow, a negative value decreases the size of the shadow.

Example

div {

box-shadow: 10px 10px 5px 12px lightblue;

}

**Set the inset Parameter**

The inset parameter changes the shadow from an outer shadow (outset) to an inner shadow.

Add the inset parameter:

div {

box-shadow: 10px 10px 5px lightblue inset;

}

**Add Multiple Shadows**

An element can also have multiple shadows:

Example

div {

box-shadow: 5px 5px blue, 10px 10px red, 15px 15px green;

}

Writing-mode CSS

## Featured snippet from the web

The writing-mode CSS property **sets whether lines of text are laid out horizontally or vertically, as well as the direction in which blocks progress**. When set for an entire document

Example

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

   <p id="text">Lorem Ipsum</p>

</body>

</html>

Mystyle.css

#text{

    width:500px;

    border:1px solid black;

/\* writing left to right \*/

    writing-mode: vertical-lr;

}

#text{

    width:500px;

    border:1px solid black;

    /\*writing right to left \*/

    writing-mode: vertical-rl;

}

# CSS Links

In addition, links can be styled differently depending on what **state** they are in.

The four links states are:

* a:link - a normal, unvisited link
* a:visited - a link the user has visited
* a:hover - a link when the user mouses over it
* a:active - a link the moment it is clicked

The active state will be at the last in practice

The link and visted state will always be before hover tag

For example

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

   <!-- Facebook site is already visited -->

   <a href="https://www.facebook12345.com">Facebook</a>

</body>

</html>

Mystyle.css

a{

    text-decoration: none;

}

/\* unvisited link \*/  
a:link {  
  color: red;  
}  
  
/\* visited link \*/  
a:visited {  
  color: green;  
}  
  
/\* mouse over link \*/  
a:hover {  
  color: hotpink;  
}  
  
/\* selected link \*/  
a:active {  
  color: blue;  
}

Another Example

<!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="link.css">

</head>

<body>

    <h2>Link</h2>

    <hr>

    <a href="#">Yahoo!</a><br><br>

    <a href="#">Instagram!</a><br><br>

    <a href="#">Bingo !</a>

</body>

</html>

Link.css

body{

    font-family:tahoma;

    font-size:40px;

    text-align:center;

}

a{

    text-decoration: none;

    background:rgba(255,0,0,.5);

    padding:10px 20px;

}

a:link {

    color:green;

}

a:visited {

    color:red;

}

a:hover{

    color:white;

    background:red;

}

a:active {

    color:magenta;

}

# Pseudo-classes

A Pseudo class in CSS is used to define the special state of an element. It can be combined with a CSS selector to add an effect to existing elements based on their states. For Example, changing the style of an element when the user hovers over it, or when a link is visited. All of these can be done using Pseudo Classes in CSS.

Note that pseudo-class names are not case-sensitive.

Syntax:

selector: pseudo-class{

property: value;

}

**:hover Pseudo-class**: This pseudo-class is used to add a special effect to an element when our mouse pointer is over it. The below example demonstrates that when your mouse enters the box area, its background color changes from yellow to orange. For example

.box:hover {

background-color: orange;

}

**:active Pseudo-class:** This pseudo-class is used to select an element that is activated when the user clicks on it. The following example demonstrates that when you click on the box, its background color changes for a moment.

**:focus Pseudo-class:** This pseudo-class is used to select an element that is currently focused by the user. It works on user input elements used in forms and is triggered as soon as the user clicks on it. In the following example, the background color of the input field which is currently focused changes.

<!DOCTYPE html>

<html>

<head>

<style>

input:focus {

  background-color: yellow;

}

</style>

</head>

<body>

<form action="" method="get">

  First name: <input type="text" name="fname"><br><br>

  Last name: <input type="text" name="lname"><br><br>

  <input type="submit" value="Submit">

</form>

</body>

</html>

**:visited Pseudo-class:** This pseudo-class is used to select the links which have been already visited by the user. In the following example, the color of the link changes once it is visited.

**:not pseudo-class:** This pseudo-class is used to select elements that do not match a specific selector.

<!DOCTYPE html>

<html>

  <head>

    <title>Title of the document</title>

    <style>

      p {

        color: red;

      }

      :not(p) {

        color: #8ebf42;

      }

    </style>

  </head>

  <body>

    <h2>:not() selector example</h2>

    <p>Lorem Ipsum is simply dummy text</p>

    <p>Lorem Ipsum is simply dummy text</p>

    <div>Lorem Ipsum is simply dummy text</div>

    <a href="https://www.google.com" target="\_blank">google.com</a>

  </body>

</html>

# :target pseudo-class: URLs with an # followed by an anchor name link to a certain element within a document. The element being linked to is the target element.

The :target selector can be used to style the current active target element.

<!DOCTYPE html>

<html>

<head>

<style>

:target {

  border: 2px solid #D4D4D4;

  background-color: #e5eecc;

}

</style>

</head>

<body>

<h1>This is a heading</h1>

<p><a href="#news1">Jump to New content 1</a></p>

<p><a href="#news2">Jump to New content 2</a></p>

<p>Click on the links above and the :target selector highlight the current active HTML anchor.</p>

<p id="news1"><b>New content 1...</b></p>

<p id="news2"><b>New content 2...</b></p>

</body>

</html>

# pseudo-class

<!DOCTYPE html>

<html>

<head>

    <style>

        /\* div.container p:first-child{

            color:red;

        }

        div.container p:last-child{

            color:green;

        }

        div.container p:first-of-type{

            color:blue;

        } \*/

        /\* div.container p:last-of-type{

            color:orange;

            font-size:15px;

        } \*/

        /\* div.container p:nth-child(2){

            color:magenta;

            font-size:18px;

        } \*/

        /\* div.container p:nth-child(even){

            color:magenta;

            font-size:18px;

        } \*/

        /\* div.container p:nth-child(odd){

            color:magenta;

            font-size:18px;

        } \*/

        /\* represent every even number \*/

        /\* div.container p:nth-child(2n){

            color:magenta;

            font-size:18px;

        } \*/

        /\* represent every odd number n start from zero \*/

        /\* div.container p:nth-child(2n+1){

            color:magenta;

            font-size:18px;

        } \*/

        /\* div.container p:nth-last-child(2){

            color:magenta;

            font-size:18px;

        } \*/

        /\* div.container :only-of-type{

            color:magenta;

            font-size:18px;

        } \*/

        div.container :only-child{

            color:magenta;

            font-size:18px;

        }

        div.container :empty{

            color:white;

            background:red;

            width:200px;

            height:50px;

        }

        div.box1 :not(p){

            color:red

        }

    </style>

</head>

<body>

<div class="container">

    <div class="box1">

        <h1>Heading 1</h1>

        <p>Para 1</p>

        <p>Para 2</p>

        <p>Para 3</p>

        <p>Para 4</p>

        <p></p>

    </div>

    <p>Container Para 1</p>

    <p>Container Para 2</p>

    <div class="box2">

        <p>Box2 para 1</p>

        <p>Box2 para 2</p>

        <p>Box2 para 3</p>

    </div>

    <div class="box3">

        <p>Box3 para 1</p>

    </div>

</div>

</body>

</html>

# :first-child Selector (pseudo-class)

Select and style every <p> element that is the first child of its parent:

p:first-child {

background-color: yellow;

}

The :first-child selector is used to select the specified selector, only if it is the first child of its parent.

# :last-child Selector (pseudo-class)

Specify a background color for the <p> element that is the last child of its parent:

p:last-child {

background: #ff0000;

}

The :last-child selector matches every element that is the last child of its parent.

# :first-of-type Selector(pseudo-class)

Specify a background color for the first <p> element of its parent:

p:first-of-type {

background: red;

}

The :first-of-type selector matches every element that is the first child, of a particular type, of its parent.

# :last-of-type Selector(pseudo-class)

# Specify a background color for the last <p> element of its parent:

# p:last-of-type {

# background: #ff0000;

# }

The :last-of-type selector matches every element that is the last child, of a particular type, of its parent.

# :nth-child() Selector (pseudo-class)

The :nth-child(n) selector matches every element that is the nth child of its parent.

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# /\* Selects the second element of div siblings \*/

# div:nth-child(2) {

# background: red;

# }

# /\* Selects the second li element in a list \*/

# li:nth-child(2) {

# background: lightgreen;

# }

# /\* Selects every third element among any group of siblings \*/

# :nth-child(3) {

# background: yellow;

# }

# </style>

# </head>

# <body>

# <div>

# <p>This is div 1.</p>

# </div>

# <div>

# <p>This is div 2.</p>

# </div>

# <div>

# <p>This is div 3.</p>

# </div>

# <ul>

# <li>First list item</li>

# <li>Second list item</li>

# <li>Third list item</li>

# <li>Fourth list item</li>

# <li>Fifth list item</li>

# </ul>

# </body>

# </html>

# :nth-last-child() Selector (pseudo-class)

# Specify a background color for every <p> element that is the second child of its parent, counting from the last child:

# p:nth-last-child(2) {

# background: red;

# }

The :nth-last-child(n) selector matches every element that is the nth child, of its parent, counting from the last child.

# :only-of-type Selector (pseudo-class)

Specify a background color for every <p> element that is the only child of its type, of its parent:

p:only-of-type {

background: #ff0000;

}

The :only-of-type selector matches every element that is the only child of its type, of its parent.

# :only-child Selector (pseudo-class)

Specify a background color for every <p> element that is the only child of its parent:

p:only-child {

background: #ff0000;

}

The :only-child selector matches every element that is the only child of its parent.

# :empty Selector (pseudo-class)

Specify a background color for empty <p> elements:

p:empty {

background: #ff0000;

}

# :not Selector (pseudo-class)

Set a background color for all elements that are not a <p> element:

:not(p) {

background: #ff0000;

}

The :not(selector) selector matches every element that is NOT the specified element/selector.

It search inside container div search for first p child of the parent (neither container parent have first child is p neither box1 contain first child p tag . The box2 parent contain first child p tag so its color will changed to red)

Example of :only-of-type

<!DOCTYPE html>

<html>

<head>

<style>

p:only-of-type {

background: red;

}

</style>

</head>

<body>

<div><p>This is a paragraph.</p></div>

<div><p>This is a paragraph.</p><p>This is a paragraph.</p></div>

</body>

</html>

# CSS Fonts

**Example**

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

   <h1>Heading</h1>

   <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Aliquid, in facilis esse cum tempore tempora doloremque, doloribus eaque odio nihil magnam eius nesciunt? Tempora ipsam soluta at molestiae? Tempore, officia.</p>

</body>

</html>

Mystyle.css

h1{

}

p{

    font-size:30px;

}

h1{

    font-weight:200px;

}

p{

    font-size:30px;

    font-weight: bold;

}

h1{

    font-weight:200px;

}

p{

    font-size:30px;

    font-weight: regular;

}

h1{

    font-weight:200;

}

p{

    font-size:30px;

    font-weight: regular;

}

h1{

    font-weight:200px;

}

p{

    font-size:30px;

    font-weight: regular;

    font-style:italic

}

# CSS font-variant Property

Set a paragraph to a small-caps font.

In a small-caps font, all lowercase letters are converted to uppercase letters. However, the converted uppercase letters appears in a smaller font size than the original uppercase letters in the text.

The font-variant property specifies whether or not a text should be displayed in a small-caps font.

p{

    font-variant: small-caps;

}

## **The CSS font-family Property**

In CSS, we use the font-family property to specify the font of a text.

**Note**: If the font name is more than one word, it must be in quotation marks, like: "Times New Roman".

**Tip:** The font-family property should hold several font names as a "fallback" system, to ensure maximum compatibility between browsers/operating systems. Start with the font you want, and end with a generic family (to let the browser pick a similar font in the generic family, if no other fonts are available). The font names should be separated with comma.

**<!DOCTYPE html>**

**<html>**

**<head>**

**<style>**

**.p1 {**

**font-family: "Times New Roman", Times, serif;**

**}**

**.p2 {**

**font-family: Arial, Helvetica, sans-serif;**

**}**

**.p3 {**

**font-family: "Lucida Console", "Courier New", monospace;**

**}**

**</style>**

**</head>**

**<body>**

**<h1>CSS font-family</h1>**

**<p class="p1">This is a paragraph, shown in the Times New Roman font.</p>**

**<p class="p2">This is a paragraph, shown in the Arial font.</p>**

**<p class="p3">This is a paragraph, shown in the Lucida Console font.</p>**

**</body>**

**</html>**

# CSS Web Safe Fonts

## **What are Web Safe Fonts?**

Web safe fonts are fonts that are universally installed across all browsers and devices.

## **Fallback Fonts**

However, there are no 100% completely web safe fonts. There is always a chance that a font is not found or is not installed properly.

Therefore, it is very important to always use fallback fonts.

This means that you should add a list of similar "backup fonts" in the font-family property. If the first font does not work, the browser will try the next one, and the next one, and so on. Always end the list with a generic font family name.

<!DOCTYPE html>

<html>

<head>

<style>

p {

font-family: Tahoma, Verdana, sans-serif;

}

</style>

</head>

<body>

<h1>CSS Fallback Fonts</h1>

<p>This is a paragraph.</p>

<p>This is another paragraph.</p>

</body>

</html>

### Example

Here, there are three font types: Tahoma, Verdana, and sans-serif. The second and third fonts are backups, in case the first one is not found.

## **Best Web Safe Fonts for HTML and CSS**

The following list are the best web safe fonts for HTML and CSS:

* Arial (sans-serif)
* Verdana (sans-serif)
* Tahoma (sans-serif)
* Trebuchet MS (sans-serif)
* Times New Roman (serif)
* Georgia (serif)
* Garamond (serif)
* Courier New (monospace)
* Brush Script MT (cursive)

# Web Fonts

Web fonts allow Web designers to use fonts that are not installed on the user's computer.

Google Fonts launched in 2010, quickly becoming the Internet’s largest, free, open-source selection of fonts. All Google Fonts are free for commercial and personal use. The Google Fonts website makes it easy for anyone to quickly select and utilize different fonts for their own design needs.

How to Use Google Fonts — Step by step

Go to the Google Fonts website(https://fonts.google.com/)

# CSS Cursor

The cursor property specifies the mouse cursor to be displayed when pointing over an element.

Example

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

   <div>

      <p>This is text</p>

   </div>

</body>

</html>

Mystyle.css

div{

    border:1px solid red;

    cursor:auto;/\* default \*/

}

div{

    border:1px solid red;

    cursor:all-scroll;

}

div{

    border:1px solid red;

    cursor:cell;

}

div{

    border:1px solid red;

    cursor:col-resize;

}

div{

    border:1px solid red;

    cursor:crosshair;

}

div{

    border:1px solid red;

    cursor:e-resize;

}

div{

    border:1px solid red;

    cursor:grab;

}

div{

    border:1px solid red;

    cursor:help;

}

div{

    border:1px solid red;

    cursor:not-allowed;

}

div{

    border:1px solid red;

    cursor:pointer;

}

div{

    border:1px solid red;

    cursor:progress;

}

div{

    border:1px solid red;

    cursor:zoom-in;

}

div{

  border:1px solid red;

  height:200px;

  width:400px;

  cursor:url('image/android.png'),pointer;

}

**Cursor Program**

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

 <div id="main-div">

  <div id="heading">

  <h1>The Cursors of CSS</h1>

</div>

  <div class="cursors">

      <div class="auto">auto</div>

      <div class="default">default</div>

      <!-- <div class="none">none</div> -->

      <!-- <div class="context-menu">context-menu</div> -->

      <div class="help">help</div>

      <div class="pointer">pointer</div>

      <div class="progress">progress</div>

      <div class="wait">wait</div>

      <div class="cell">cell</div>

      <div class="crosshair">crosshair</div>

      <div class="text">text</div>

      <div class="vertical-text">vertical-text</div>

      <div class="alias">alias</div>

      <div class="copy">copy</div>

      <div class="move">move</div>

      <div class="no-drop">no-drop</div>

      <div class="not-allowed">not-allowed</div>

      <div class="all-scroll">all-scroll</div>

      <div class="col-resize">col-resize</div>

      <div class="row-resize">row-resize</div>

      <div class="n-resize">n-resize</div>

      <div class="s-resize">s-resize</div>

      <div class="e-resize">e-resize</div>

      <div class="w-resize">w-resize</div>

      <div class="ns-resize">ns-resize</div>

      <div class="ew-resize">ew-resize</div>

      <div class="ne-resize">ne-resize</div>

      <div class="nw-resize">nw-resize</div>

      <div class="se-resize">se-resize</div>

      <div class="sw-resize">sw-resize</div>

      <div class="nesw-resize">nesw-resize</div>

      <div class="nwse-resize">nwse-resize</div>

  </div>

</div>

</body>

</html>

**mystyle.css**

.auto            { cursor: auto; }

.default         { cursor: default; }

.none            { cursor: none; }

.context-menu    { cursor: context-menu; }

.help            { cursor: help; }

.pointer         { cursor: pointer; }

.progress        { cursor: progress; }

.wait            { cursor: wait; }

.cell            { cursor: cell; }

.crosshair       { cursor: crosshair; }

.text            { cursor: text; }

.vertical-text   { cursor: vertical-text; }

.alias           { cursor: alias; }

.copy            { cursor: copy; }

.move            { cursor: move; }

.no-drop         { cursor: no-drop; }

.not-allowed     { cursor: not-allowed; }

.all-scroll      { cursor: all-scroll; }

.col-resize      { cursor: col-resize; }

.row-resize      { cursor: row-resize; }

.n-resize        { cursor: n-resize; }

.e-resize        { cursor: e-resize; }

.s-resize        { cursor: s-resize; }

.w-resize        { cursor: w-resize; }

.ns-resize       { cursor: ns-resize; }

.ew-resize       { cursor: ew-resize; }

.ne-resize       { cursor: ne-resize; }

.nw-resize       { cursor: nw-resize; }

.se-resize       { cursor: se-resize; }

.sw-resize       { cursor: sw-resize; }

.nesw-resize     { cursor: nesw-resize; }

.nwse-resize     { cursor: nwse-resize; }

body {

  text-align: center;

font-family: "Segoe UI", Roboto, Helvetica, Arial, sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol";

display:flex;

justify-content:center;

}

#heading{

  display:flex;

  justify-content:center;

  width:100%;

  height:70px;

  /\* background:yellow; \*/

}

.cursors {

  display: flex;

  flex-wrap: wrap;

  width:100%;

  justify-content:space-around;

}

#main-div{

  display:flex;

  width:60%;

  flex-direction:column;

  justify-content:space-between;

  /\* background:aqua; \*/

  height:570px;

}

.cursors div {

  padding: 10px 2px;

  width:190px;

  justify-content:space-around;

  border: 1px solid teal;

  border-radius: 5px;

  margin: 0 5px 5px 0;

}

.cursors div:hover {

  background: #eee;

}

# CSS The !important Rule

## **What is !important?**

The !important rule in CSS is used to add more importance to a property/value than normal.

In fact, if you use the !important rule, it will override ALL previous styling rules for that specific property on that element!

**An element having all three selector element,class and id and all the selector point to color , then priority order will be**

* **Id**
* **Class**
* **Element**

**Example**

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

   <div>

      <p>This is text</p>

   </div>

</body>

</html>

**Mystyle.css**

div{

    border:1px solid red;

}

**Now take two div in the mystyle.css file**

div{

    border:1px solid red;

}

div{

    border:1px solid green;

}

**Now border color will be green due to order**

**Now html file add inline style sheet**

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

   <div style="border: 1px solid yellow;">

      <p>This is text</p>

   </div>

</body>

</html>

**Now the border will become yellow due to inline style sheet**

**Now apply importance on first div**

div{

    border:1px solid red !important;

}

div{

    border:1px solid green;

}

**Now it will override all the order and rules and make the border red**

div{

    border:1px solid green;

    width:500px;

    background-color:yellowgreen;

    margin:100px;

    padding:20px;

    /\* box-shadow: x axis y-asxis blur spread-radius color\*/

    box-shadow: 10px 10px 15px 3px gray;

}

# CSS Opacity / Transparency

The opacity property specifies the opacity/transparency of an element.

## **Transparent Image**

The opacity property can take a value from 0.0 - 1.0. The lower the value, the more transparent:

For example

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

   <div id="color-box">

   </div>

   <img src="image/bird.jpg" alt="" />

</body>

</html>

Mystyle.css

#color-box

{

    background-color: green;

    height:200px;

    width:500px;

}

**To make box fully transparent write opacity:0**

#color-box

{

    background-color: green;

    height:200px;

    width:500px;

    margin-bottom: 20px;

    opacity:0;

}

**Make box half transparent**

#color-box

{

    background-color: green;

    height:200px;

    width:500px;

    margin-bottom: 20px;

    opacity:0.5;

}

#color-box

{

    background-color: green;

    height:200px;

    width:500px;

    margin-bottom: 20px;

    opacity:0.5;

}

img{

    opacity:0.3;

}

**To make image on hover**

#color-box

{

    background-color: green;

    height:200px;

    width:500px;

    margin-bottom: 20px;

    opacity:0.5;

}

img:hover{

    opacity:0.7;

}

**Now take another example**

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

   <div id="color-box">

   </div>

   <div id="red-box"></div>

</body>

</html>

**Mystyle.css**

#color-box

{

    background-color: green;

    height:200px;

    width:500px;

    margin-bottom: 20px;

    opacity:0.5;

}

#red-box{

    height:300px;

    width:200px;

    background-color:red;

    margin-top:-100px;

}

# CSS filter Property

The filter property defines visual effects (like blur and saturation) to an element (often <img>)

Example

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

  <img src="image/bird.jpg" alt=""/>

</body>

</html>

**Mystyle.css**

img{

    filter:blur(10px);/\* radius in pixels \*/

}

**Brightness**

img{

    filter:brightness(50%); /\*Brightness in percentage \*/

}

**Contrastness**

img{

    filter:contrast(70%);/\*in percentage \*/

}

**Drop-Shadow**

img{

    /\* drop-shadow(horizontal in pixel vertical in pixel blur shadow color) \*/

    filter:drop-shadow(8px 8px 5px red);

}

**Grayscale**

img{

   filter:grayscale(80%);

}

|  |  |
| --- | --- |
| invert(*%*) | Inverts the samples in the image.  0% (0) is default and represents the original image. 100% will make the image completely inverted.  **Note:** Negative values are not allowed. |

img{

   filter:invert(80%);

}

**Opacity**

img{

   filter:opacity(30%);

}

**Applying more then one filter at a time**

img{

   filter:brightness(30%),contrast(80%);

}

# CSS Gradient

CSS gradients let you display smooth transitions between two or more specified colors.

CSS defines two types of gradients:

* **Linear Gradients (goes down/up/left/right/diagonally)**
* **Radial Gradients (defined by their center)**

## **CSS Linear Gradients**

To create a linear gradient you must define at least two color stops. Color stops are the colors you want to render smooth transitions among. You can also set a starting point and a direction (or an angle) along with the gradient effect.

### Syntax

background-image: linear-gradient(direction, color-stop1, color-stop2, ...);

Gradients belong to the image data type, they can only be used where images can be used. For this reason, linear-gradient property won't work on background-color property and other properties that use the color data type.

Example

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

 <div id="div1">

 </div>

</body>

</html>

Mystyle.css

#div1{

   height:500px;

   border:1px solid gray;

   background-image: linear-gradient(red,yellow);

}

Default direction from top to bottom to change from bottom to top write following

#div1{

   height:500px;

   border:1px solid gray;

   background-image: linear-gradient(to top,red,yellow);

}

Now change direction left to right

#div1{

   height:500px;

   border:1px solid gray;

   background-image: linear-gradient(to right ,red,yellow);

}

Now change direction right to left

#div1{

   height:500px;

   border:1px solid gray;

   background-image: linear-gradient(to left ,red,yellow);

}

Now change direction from one corner (top left) to another corner (bottom right)

#div1{

   height:500px;

   border:1px solid gray;

   background-image: linear-gradient(to bottom right ,red,yellow);

}

Now change to bottom left

#div1{

   height:500px;

   border:1px solid gray;

   background-image: linear-gradient(to bottom left ,red,yellow);

}

Now change to top left,top right

Now change direction to degree

#div1{

   height:500px;

   border:1px solid gray;

   background-image: linear-gradient(45deg ,red,yellow);

}

#div1{

   height:500px;

   border:1px solid gray;

   background-image: linear-gradient(120deg ,red,yellow);

}

#div1{

   height:500px;

   border:1px solid gray;

   background-image: linear-gradient(150deg ,red,yellow);

}

# we can add multiple colors to linear gradient

#div1{

  height:500px;

  border:1px solid gray;

  background-image: linear-gradient(to right,red,orange,yellow,green,blue,indigo,violet);

}

Now change to -45 degree

#div1{

  height:500px;

  border:1px solid gray;

  background-image: linear-gradient(-45deg,red,orange,yellow,green,blue,indigo,violet);

}

We can give percentage to the color

#div1{

  height:500px;

  border:1px solid gray;

  background-image: linear-gradient(to right,red,yellow,green 50%);

}

#div1{

  height:500px;

  border:1px solid gray;

  background-image: linear-gradient(to right,red,yellow 30%,green 50%);

}

We can give transparency to the color

#div1{

  height:500px;

  border:1px solid gray;

  background-image: linear-gradient(to right,rgba(255,0,0,1),rgba(255,0,0,0));

}

# Repeating Linear Gradient

#div1{

  height:500px;

  border:1px solid gray;

  background-image: repeating-linear-gradient(to right,red,yellow 10%,green 20%);

}

For see the effects of repeating linear gradient we have to give color in percentage We can change the angle also

#div1{

  height:500px;

  border:1px solid gray;

  background-image: repeating-linear-gradient(45deg,red,yellow 10%,green 20%);

}

# CSS Radial Gradients

A radial gradient is defined by its center.

To create a radial gradient you must also define at least two color stops.

### Syntax

background-image: radial-gradient(shape size at position, start-color, ..., last-color);

#div1{

   height:500px;

   border:1px solid gray;

   background-image: radial-gradient(red,yellow,green);

}

We can give color in percentage

#div1{

   height:500px;

   border:1px solid gray;

   background-image: radial-gradient(red 30%,yellow 40%,green 50%);

}

By default shape of the radial is eclipse we can make it circle

#div1{

   height:500px;

   border:1px solid gray;

   background-image: radial-gradient(circle,red 30%,yellow 40%,green 50%);

}

We can also define position of the circle (top,bottom,left,right)

#div1{

   height:500px;

   border:1px solid gray;

   background-image: radial-gradient(circle at top,red,yellow,green);

}

#div1{

   height:500px;

   border:1px solid gray;

   background-image: radial-gradient(circle at bottom,red,yellow,green);

}

Color stops are the colors you want to render smooth transitions among. This value consists of a color value, followed by an optional stop position

#div1{

height:500px;

width:700px;

border:1px solid gray;

background-image: radial-gradient(circle,red 0%,blue 100%);

}

Here red is at centre position (0%) and blue is at border (100%)

#div1{

height:500px;

width:700px;

border:1px solid gray;

background-image: radial-gradient(circle,red 0%,yellow 30%,green 60%);

}

#div1{

   height:500px;

   border:1px solid gray;

   background-image: radial-gradient(circle at center center,red 30%,yellow 40%,green 50%);

}

If we use same color side by side, we can create color strip of solid colors

#div1{

height:500px;

width:700px;

border:1px solid gray;

background:radial-gradient(circle at center center,red 0%, red 25%,green 25%,green 50%,blue 50%,blue 100%)

}

#div1{

   height:500px;

   border:1px solid gray;

   background-image: radial-gradient(circle at top left,red 30%,yellow 40%,green 50%);

}

#div1{

  height:500px;

  border:1px solid gray;

  background-image: radial-gradient(circle at center left,red 30%,yellow 40%,green 50%);

}

We can give position through percentage

#div1{

  height:500px;

  border:1px solid gray;

  background-image: radial-gradient(circle at 0%,red 30%,yellow 40%,green 50%);

}

#div1{

  height:500px;

  border:1px solid gray;

  background-image: radial-gradient(circle at 0%,red 30%,yellow 40%,green 50%);

}

#div1{

  height:500px;

  border:1px solid gray;

  background-image: radial-gradient(circle at 75%,red 30%,yellow 40%,green 50%);

}

#div1{

  height:500px;

  border:1px solid gray;

  background-image: radial-gradient(circle at 100%,red 30%,yellow 40%,green 50%);

}

We can change position of circle in x and y direction for instance:

#div1{

  height:500px;

  border:1px solid gray;

  /\* circle at x axis,y axis \*/

  background-image: radial-gradient(circle at 25% 25%,red 10%,yellow 40%,green 50%);

}

#div1{

  height:500px;

  border:1px solid gray;

  /\* circle at x axis,y axis \*/

  background-image: radial-gradient(circle at 25% 60%,red 10%,yellow 40%,green 50%);

}

#div1{

  height:500px;

  border:1px solid gray;

  /\* circle at x axis,y axis \*/

  background-image: radial-gradient(circle at 90% 90%,red 10%,yellow 40%,green 50%);

}

We can do same for ellipse

For instance

#div1{

  height:500px;

  border:1px solid gray;

  /\* circle at x axis,y axis \*/

  background-image: radial-gradient(ellipse at 90% 90%,red ,yellow ,green );

}

We can also give transparent through transparent also

#div1{

  height:500px;

  border:1px solid gray;

  /\* circle at x axis,y axis \*/

  background-image: radial-gradient(ellipse at top,red ,transparent );

}

We can apply two radial gradient

#div1{

  height:500px;

  border:1px solid gray;

  background-image: radial-gradient(ellipse at top,red,transparent),radial-gradient(ellipse at bottom,yellow,transparent);

}

#div1 {

  height: 500px;

  border: 1px solid gray;

  /\* circle at x axis,y axis \*/

  background-image: radial-gradient(ellipse at top, red, transparent), radial-gradient(ellipse at bottom, yellow, transparent),

  radial-gradient(ellipse at left, green, transparent), radial-gradient(ellipse at right, blue, transparent);

}

The repeating-radial-gradient() function is used to repeat radial gradients.

#div1 {

  height: 500px;

  border: 1px solid gray;

  /\* circle at x axis,y axis \*/

  background-image: repeating-radial-gradient(red,yellow 10%,green 15%);

}

# CSS resize Property

The resize property defines if (and how) an element is resizable by the user.

**Note:** The resize property does not apply to inline elements or to block elements where overflow="visible". So, make sure that overflow is set to "scroll", "auto", or "hidden".

Example

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

<div id="overflow">

  <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Perspiciatis amet aperiam aliquid maiores. Omnis, sapiente. Ipsa, similique. Voluptatem eveniet modi odit voluptates cum perferendis quibusdam, nam natus, repudiandae numquam reprehenderit velit recusandae, ut impedit nemo. Ullam quasi hic earum reprehenderit, esse voluptatem eveniet necessitatibus. Eligendi autem impedit quasi quisquam aspernatur amet exercitationem perferendis corrupti, molestiae earum, tempore ducimus sunt itaque doloremque dicta accusamus. Sequi, unde eveniet ipsa eaque beatae quia doloremque asperiores impedit voluptas expedita quasi, maxime odit sit perspiciatis esse dignissimos, labore est corporis magni explicabo nihil. Excepturi hic consequuntur provident dolorem molestiae aspernatur mollitia necessitatibus dolor quia expedita nobis ab rerum, omnis quod, maiores nihil architecto nemo earum! Ab, velit? Mollitia aliquam vero rem placeat. Eum corrupti harum consequuntur voluptas iusto facere voluptatibus assumenda deleniti quos, error incidunt neque at fuga dolorum fugit nemo animi rem ab vitae necessitatibus expedita nihil dolore modi. Saepe atque ipsam, libero voluptatem est iusto ullam suscipit debitis eos, iste, recusandae vitae autem dignissimos minus eaque veritatis fugiat beatae. Maiores asperiores quaerat, assumenda perferendis veniam tenetur laboriosam harum consequuntur hic quas minus quia accusantium illo cumque expedita quod quos aspernatur repellat itaque! Libero adipisci veniam illo beatae fugiat. Ex ea nemo pariatur ullam consequuntur quam laboriosam fuga impedit! Aliquam, aut? Nostrum iure rem dignissimos ad est error voluptatem veniam praesentium, at pariatur dolorum delectus porro explicabo quia laboriosam laborum asperiores illo accusamus incidunt eius numquam aperiam nulla. Labore vel inventore nostrum eligendi ad qui, minus laborum eaque quod nulla totam rerum fugit deleniti mollitia illo, perferendis, vero quaerat provident aspernatur obcaecati! Ducimus commodi, iusto deserunt, at rerum aut accusantium ipsum, aliquid eos molestiae unde saepe consequatur placeat nobis quod non cumque labore doloremque. Mollitia omnis porro temporibus corporis quisquam quibusdam! Sit, vel, vitae adipisci repellat quaerat ipsum id obcaecati est sunt earum quidem!

  </p>

</div>

 <textarea name="" id="" cols="30" rows="10"></textarea>

</body>

</html>

Mystyle.css

#overflow{

   background-color: whitesmoke;

   border:1px solid red;

   width:500px;

   height:300px;

  overflow:auto;

}

By default we can resize the textarea

Prevent to resize the textarea

#overflow{

   background-color: whitesmoke;

   border:1px solid red;

   width:500px;

   height:300px;

  overflow:auto;

}

textarea{

  resize:none;

}

Make resize the div in both direction:

#overflow{

   background-color: whitesmoke;

   border:1px solid red;

   width:500px;

   height:300px;

  overflow:auto;

  resize:both;

}

textarea{

  resize:none;

}

**horizontal** The user can resize the width of the element

#overflow{

   background-color: whitesmoke;

   border:1px solid red;

   width:500px;

   height:300px;

  overflow:auto;

  resize:horizontal;

}

textarea{

  resize:none;

}

**vertical** The user can resize the height of the element

#overflow{

   background-color: whitesmoke;

   border:1px solid red;

   width:500px;

   height:300px;

  overflow:auto;

  resize:vertical;

}

textarea{

  resize:none;

}

# CSS Lists

Example

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

  <h2>Unordered List</h2>

  <ul>

    <li>Item 1</li>

    <li>Item 2</li>

    <li>Item 3</li>

    <li>Item 4</li>

    <li>Item 5</li>

  </ul>

  <hr/>

  <h2>Ordered List</h2>

  <ol>

    <li>Item 1</li>

    <li>Item 2</li>

    <li>Item 3</li>

    <li>Item 4</li>

    <li>Item 5</li>

  </o>

</body>

</html>

Mystyle.css

ul{

  list-style-type:square; /\* value can be none,disc,square and circle \*/

}

ul{

  list-style-type:square; /\* value can be none,disc,square and circle \*/

}

ol{

  list-style-type:upper-alpha;

}

ul{

  list-style-type:square; /\* value can be none,disc,square and circle \*/

}

ol{

  list-style-type:decimal-leading-zero;/\* value can be lower-alpha,lower-roman,upper-roman,decimal,decimal-leading-zero\*/

}

Now apply list-style-image

ul{

  list-style-image: url('sqpurple.jpg'); /\* value can be none,disc (default value) ,square and circle \*/

}

ol{

  list-style-type:decimal-leading-zero;/\* value can be lower-alpha,lower-roman,upper-roman,lower-roman,decimal (default value) ,decimal-leading-zero\*/

}

The list-style-position property specifies the position of the list-item markers (bullet points).

"list-style-position: outside;" (default property) means that the bullet points will be outside the list item. The start of each line of a list item will be aligned vertically.

"list-style-position: inside;" means that the bullet points will be inside the list item. As it is part of the list item, it will be part of the text and push the text at the start:

For image we can search <https://www.iconfinder.com/>

In the site search for arrow icon then check free icon

Select the icon

Select 16px size

And then click download in png button

ul{

  list-style-image: url('sqpurple.jpg'); /\* value can be none,disc,square and circle \*/

  list-style-position: inside;

}

Now in place of item1 type lorem25 in html (ol)

ol{

  list-style-type:decimal-leading-zero;/\* value can be lower-alpha,lower-roman,upper-roman,decimal,decimal-leading-zero\*/

  list-style-position:outside;

}

li{

  border:1px solid black;

}

Now apply shorthand property

ol{

  /\* list-style-type:decimal-leading-zero; value can be lower-alpha,lower-roman,upper-roman,decimal,decimal-leading-zero

  list-style-position:inside; \*/

  /\* now we can place shorthand property \*/

  list-style: decimal-leading-zero inside;

}

ul{

  /\* list-style-image: url('sqpurple.jpg'); value can be none,disc,square and circle \*/

  /\* list-style-image: url('arrow.png');

  list-style-position: inside; \*/

  /\* now we can place shorthand property \*/

  list-style: url('arrow.png') inside;

}

**Now change image name and add square in list-style**

ul{

  /\* list-style-image: url('sqpurple.jpg'); value can be none,disc,square and circle \*/

  /\* list-style-image: url('arrow.png');

  list-style-position: inside; \*/

  /\* now we can place shorthand property \*/

  list-style: square url('arrw.png') inside;

  /\* if image not found it apply the square \*/

}

# CSS Table Style

**Tr will not take the border**

**Example**

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

  <table>

    <caption>Table Css</caption>

    <tr>

      <th>Heading 1</th>

      <th>Heading 2</th>

      <th>Heading 3</th>

    </tr>

    <tr>

      <td>Content 1</td>

      <td>Content 2</td>

      <td>Content 3</td>

    </tr>

    <tr>

      <td>Content 4</td>

      <td>Content 5</td>

      <td>Content 6</td>

    </tr>

  </table>

</body>

</html>

**Mystyle.css**

table{

  border:1px solid black;

}

caption{

  border:1px solid black;

}

tr{

  border:1px solid black;

}

th{

  border:1px solid black;

}

td{

  border:1px solid black;

}

By default border are separated

# CSS border-collapse Property

The border-collapse property sets whether table borders should collapse into a single border or be separated as in standard HTML. (by default  border-collapse: separate )

table{

  border-collapse: collapse;

}

caption{

  border:1px solid black;

}

tr{

  border:1px solid black;

}

th{

  border:1px solid black;

}

td{

  border:1px solid black;

}

tr:hover{

  background-color:gray ;

}

**Applying height and padding to th and td**

table{

  border:1px solid black;

  border-collapse: collapse;

}

caption{

  border:1px solid black;

}

tr{

  border:1px solid black;

}

th{

  border:1px solid black;

  height:30px;

  padding:5px;

}

td{

  border:1px solid black;

  height:25px;

  padding:5px;

}

tr:nth-child(even){

  background:green;

  color:white;

}

tr:hover {

  background-color:#BCD2E5;

}

**Applying text-align to th and td**

table{

  border:1px solid black;

  border-collapse: collapse;

}

caption{

  border:1px solid black;

}

tr{

  border:1px solid black;

}

th{

  border:1px solid black;

  height:30px;

  padding:15px;

  text-align: right;

}

td{

  border:1px solid black;

  height:25px;

  padding:5px;

  text-align:center;

}

**We can apply vertical-align to th and td**

table{

  border:1px solid black;

  border-collapse: collapse;

}

caption{

  border:1px solid black;

}

tr{

  border:1px solid black;

}

th{

  border:1px solid black;

  height:30px;

  padding:15px;

  text-align: right;

  vertical-align: top;

}

td{

  border:1px solid black;

  height:25px;

  padding:5px;

  text-align:center;

  vertical-align:bottom;/\* value will be top,bottom,middle \*/

}

**We can’t apply margin to table instead of that we apply border-spacing**

# CSS border-spacing Property

The border-spacing property sets the distance between the borders of adjacent cells.

**Note:** This property works only when border-collapse is separate.

If one value is specified, it defines both the horizontal and vertical spacing between cells

If two values are specified, the first sets the horizontal spacing and the second sets the vertical spacing

table{

  border:1px solid black;

  border-collapse: separate;

  border-spacing: 15px;

}

caption{

  border:1px solid black;

}

tr{

  border:1px solid black;

}

th{

  border:1px solid black;

  height:30px;

  padding:15px;

  text-align: right;

  vertical-align: top;

}

td{

  border:1px solid black;

  height:25px;

  padding:5px;

  text-align:center;

  vertical-align:bottom;/\* value will be top,bottom,middle \*/

}

table{

  border:1px solid black;

  border-collapse: separate;

  border-spacing: 15px 25px;/\*top bottom and left right\*/

}

caption{

  border:1px solid black;

}

tr{

  border:1px solid black;

}

th{

  border:1px solid black;

  height:30px;

  padding:15px;

  text-align: right;

  vertical-align: top;

}

td{

  border:1px solid black;

  height:25px;

  padding:5px;

  text-align:center;

  vertical-align:bottom;/\* value will be top,bottom,middle \*/

}

# CSS caption-side Property

The caption-side property specifies the placement of a table caption.

**top** Puts the caption above the table. This is default

**bottom**  Puts the caption below the table

table{

  border:1px solid black;

  border-collapse: separate;

  border-spacing: 15px 25px;/\*top bottom and left right\*/

}

caption{

  border:1px solid black;

  caption-side:bottom;

}

tr{

  border:1px solid black;

}

th{

  border:1px solid black;

  height:30px;

  padding:15px;

  text-align: right;

  vertical-align: top;

}

td{

  border:1px solid black;

  height:25px;

  padding:5px;

  text-align:center;

  vertical-align:bottom;/\* value will be top,bottom,middle \*/

}

**If last table data has no content**

# CSS empty-cells Property

The empty-cells property sets whether or not to display borders on empty cells in a table.

**Note:** This property has no effect if border-collapse is "collapse".

Example

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

  <table>

    <caption>Table Css</caption>

    <tr>

      <th>Heading 1</th>

      <th>Heading 2</th>

      <th>Heading 3</th>

    </tr>

    <tr>

      <td>Content 1</td>

      <td>Content 2</td>

      <td>Content 3</td>

    </tr>

    <tr>

      <td>Content 4</td>

      <td>Content 5</td>

      <td> </td>

    </tr>

  </table>

</body>

</html>

Mystyle.css

table{

  border:1px solid black;

  border-collapse: separate;

  border-spacing: 15px 25px;/\*top bottom and left right\*/

  empty-cells: hide;

}

caption{

  border:1px solid black;

  caption-side:bottom;

}

tr{

  border:1px solid black;

}

th{

  border:1px solid black;

  height:30px;

  padding:15px;

  text-align: right;

  vertical-align: top;

}

td{

  border:1px solid black;

  height:25px;

  padding:5px;

  text-align:center;

  vertical-align:bottom;/\* value will be top,bottom,middle \*/

}

# CSS Grid Layout Module

CSS Grid is the new Layout Model in CSS

From Oct 2017 all major browser started supporting CSS Grid

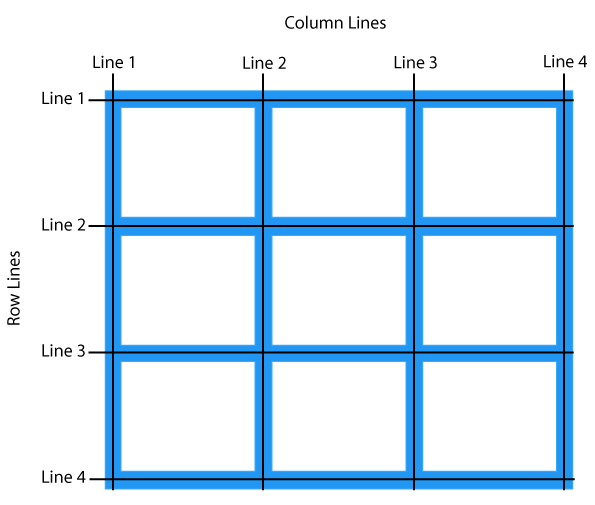
First we design with the help of table then by div, after div come flex and now come CSS Grid.

The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

A grid-based layout system, with rows and columns

A layout just like Ms. Excel

Offers an easy and convenient layout



Grid container is the entire grid container (3 rows and 3 columns in the diagram )

Grid lines are the blue lines

Grid cell is one small square box

Grid track is space between 2 adjacent lines The area between two adjacent grid lines. The gap between two grid row lines is a row track, and the gap between two grid column lines is a column track. The size of a grid track determines the width and height of our grid items.

Grid items are the children(direct) of the grid container

## **Grid Elements**

A grid layout consists of a parent element, with one or more child elements.

# grid-template-rows Property

The grid-template-rows property specifies the number (and the heights) of the rows in a grid layout.The values are a space-separated list, where each value specifies the height of the respective row.

# grid-template-columns Property

The grid-template-columns property specifies the number (and the widths) of columns in a grid layout.The values are a space separated list, where each value specifies the size of the respective column.

**What’s a fraction (1FR)?**

A fraction or 1FR is one part of the whole. One 1FR fraction is 100% of the available space. Two 1FR fractions are 50% each of the available space. So, in that case, 1FR is 1/2 of the available space. If there are 250 1FR fractions? Then each fraction (1FR) is 1/250 or 0.4%.

In fractions: 1FR = 1/total number of FRs

In percentages: 1FR = 100/total number of FRs

**Repeat()**

Repeat() is a notation that we can use with the grid-template-columns and grid-template-rows properties to make your rules more concise and easier to understand when creating a large amount of columns or rows.

For example, let’s say we have this definition for a grid container:

.container {

display: grid;

grid-template-columns: 1fr 2fr 1fr 2fr 1fr 2fr;

}

We can use the repeat() notation like this instead to simplify:

.container {

display: grid;

grid-gap: 10px 15px;

grid-template-columns: repeat(3, 1fr 2fr);

}

The first value passed to repeat() is the number of repetitions and the second value is the grid tracks to repeat

# grid-column-start

The grid-column-start property defines on which column-line the item will start.

# grid-column-end

The grid-column-end property defines how many columns an item will span, or on which column-line the item will end

# grid-row-start

The grid-row-start property defines on which row-line the item will start

# grid-row-end

The grid-row-end property defines how many rows an item will span, or on which row-line the item will end

# grid-column-gap( column-gap)

The grid-column-gap property defines the size of the gap between the columns in a grid layout, property was renamed to column-gap in CSS3.

# grid-row-gap (row-gap)

The grid-row-gap property defines the size of the gap between the rows in a grid layout.This property was renamed to row-gap in CSS3.

# grid-gap

The grid-gap property defines the size of the gap between the rows and columns in a grid layout, and is a shorthand property for the following properties:

row-gap

column-gap

This property was renamed to gap in CSS3.

.grid-container {

grid-gap: 50px;

}

Set the gap between rows to 20px, and the columns to 50px:

.grid-container {

grid-gap: 20px 50px;

}

**Grid Columns**

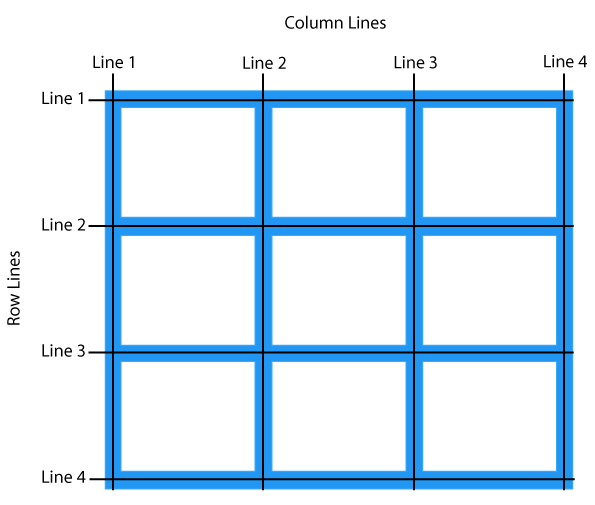
The vertical lines of grid items are called columns

The horizontal lines of grid items are called rows.

The space between each column/row are called gaps

The line between columns are called column lines

The line between rows are called row lines



**Difference between inline-grid and grid**

The difference between the values inline-grid and grid is that the inline-grid will make the element inline while grid will make it a block-level element.

Inline-grid will depent on content size,while grid will take full width container.

Example

<!DOCTYPE html>

<html>

<head>

    <title>Document</title>

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

</head>

<body>

<div class="grid-container">

    <div class="grid-item" id="grid-1">1</div>

    <div class="grid-item" id="grid-2">2</div>

    <div class="grid-item" id="grid-3">3</div>

    <div class="grid-item" id="grid-4">4</div>

    <div class="grid-item" id="grid-5">5</div>

    <div class="grid-item" id="grid-6">6</div>

    <div class="grid-item" id="grid-7">7</div>

    <div class="grid-item" id="grid-8">8</div>

    <div class="grid-item" id="grid-9">9</div>

</div>

</body>

</html>

Mystyle.css

.grid-container{

  background-color: indianred;

  padding:2rem;

  display: grid;

}

 .grid-item{

    background-color: wheat;

    border:1px solid black;

    font-size:3rem;

    display:flex;

    justify-content: center;

    align-items:center;

  }

# CSS grid-template-columns Property

The grid-template-columns property specifies the number (and the widths) of columns in a grid layout.

The values are a space separated list, where each value specifies the size of the respective column.

.grid-container{

  background-color: indianred;

  padding:2rem;

  display: grid;

}

 .grid-item{

    background-color: wheat;

    border:1px solid black;

    font-size:3rem;

    display:flex;

    justify-content: center;

    align-items:center;

  }

It automatically divide the div into column

|  |  |
| --- | --- |
| auto | The size of the columns is determined by the size of the container and  on the size of the content of the items in the column,it try to occupy whole space |

.grid-container{

    background-color: indianred;

    padding:2rem;

    display: grid;

    grid-template-columns:200px 400px auto;

  }

  .grid-item{

    background-color: wheat;

    border:1px solid black;

    font-size:3rem;

    display:flex;

    justify-content: center;

    align-items:center;

  }

If you want all column to equal in size , apply auto to each column

. grid-container{

    background-color: indianred;

    padding:2rem;

    display: grid;

    grid-template-columns:auto auto auto;

  }

  .grid-item{

    background-color: wheat;

    border:1px solid black;

    font-size:3rem;

    display:flex;

    justify-content: center;

    align-items:center;

  }

We can give column value in percentage

.grid-container{

    background-color: indianred;

    padding:2rem;

    display: grid;

    grid-template-columns:auto 50% auto;

  }

  .grid-item{

    background-color: wheat;

    border:1px solid black;

    font-size:3rem;

    display:flex;

    justify-content: center;

    align-items:center;

  }

We can give as many column value as we can , we give another column of 50px.

grid-container{

    background-color: indianred;

    padding:2rem;

    display: grid;

    grid-template-columns:auto 50% auto 50px;

  }

  .grid-item{

    background-color: wheat;

    border:1px solid black;

    font-size:3rem;

    display:flex;

    justify-content: center;

    align-items:center;

  }

# CSS grid-template-rows Property

The grid-template-rows property specifies the number (and the heights) of the rows in a grid layout.

The values are a space-separated list, where each value specifies the height of the respective row.

|  |  |
| --- | --- |
| auto | The size of the rows is determined by the size of the container,  and on the size of the content of the items in the row |

.grid-container{

    background-color: indianred;

    padding:2rem;

    display: grid;

    grid-template-columns:auto 50% auto;

    grid-template-rows:100px 200px auto;

  }

  .grid-item{

    background-color: wheat;

    border:1px solid black;

    font-size:3rem;

    display:flex;

    justify-content: center;

    align-items:center;

  }

# CSS grid-column-gap Property

The  column-gap property defines the size of the gap between the columns in a grid layout.

.grid-container{

    background-color: indianred;

    padding:2rem;

    display: grid;

    grid-template-columns:auto 50% auto;

    grid-template-rows:100px 200px auto;

    column-gap:2rem;

  }

  .grid-item{

    background-color: wheat;

    border:1px solid black;

    font-size:3rem;

    display:flex;

    justify-content: center;

    align-items:center;

  }

# CSS  row-gap Property

The  row-gap property defines the size of the gap between the rows in a grid layout.

.grid-container{

    background-color: indianred;

    padding:2rem;

    display: grid;

    grid-template-columns:auto 50% auto;

    grid-template-rows:100px 200px auto;

    column-gap:2rem;

    row-gap:1rem;

  }

  .grid-item{

    background-color: wheat;

    border:1px solid black;

    font-size:3rem;

    display:flex;

    justify-content: center;

    align-items:center;

  }

Now make column-gap 1rem and row-gap 1rem and grid-template-column: 200px 300px 200px

.grid-container{

    background-color: indianred;

    padding:2rem;

    display: grid;

    grid-template-columns:200px 300px 200px;

    grid-template-rows:100px 200px auto;

    column-gap:1rem;

    row-gap:1rem;

  }

  .grid-item{

    background-color: wheat;

    border:1px solid black;

    font-size:3rem;

    display:flex;

    justify-content: center;

    align-items:center;

  }

# CSS justify-content Property

The justify-content property aligns the flexible container's items when the items do not use all available space on the main-axis (horizontally).

|  |  |
| --- | --- |
| start | Default value. Items are positioned at the beginning of the container |
| end | Items are positioned at the end of the container |

.grid-container{

    background-color: indianred;

    padding:2rem;

    display: grid;

    grid-template-columns:200px 300px 200px;

    grid-template-rows:100px 200px auto;

    column-gap:1rem;

    row-gap:1rem;

    justify-content: center;

  }

  .grid-item{

    background-color: wheat;

    border:1px solid black;

    font-size:3rem;

    display:flex;

    justify-content: center;

    align-items:center;

  }

|  |  |
| --- | --- |
| center | Items are positioned in the center of the container |

|  |  |
| --- | --- |
| space-between | Items will have space between them |

|  |  |
| --- | --- |
| space-around | Items will have space before, between, and after them |

|  |  |
| --- | --- |
| space-evenly | Items will have equal space around them |

Now give height to .grid-container as height:600px and remove grid-template-rows property

.grid-container{

    background-color: indianred;

    padding:2rem;

    display: grid;

    grid-template-columns:200px 300px 200px;

    grid-template-rows:100px 200px auto;

    column-gap:1rem;

    row-gap:1rem;

    justify-content: space-evenly;

    height:600px;

  }

  .grid-item{

    background-color: wheat;

    border:1px solid black;

    font-size:3rem;

    display:flex;

    justify-content: center;

    align-items:center;

  }

# CSS align-content Property

The align-content property can also be used on a grid container to align grid items in the block direction.

|  |  |
| --- | --- |
| start | Lines are packed toward the start of the grid |

And also decrease the size of container to 400px

.grid-container{

    background-color: indianred;

    padding:2rem;

    display: grid;

    grid-template-columns:200px 300px 200px;

    /\* grid-template-rows:100px 200px auto; \*/

    column-gap:1rem;

    row-gap:1rem;

    justify-content: space-evenly;

    align-content:start;

    height:400px;

  }

  .grid-item{

    background-color: wheat;

    border:1px solid black;

    font-size:3rem;

    display:flex;

    justify-content: center;

    align-items:center;

  }

|  |  |
| --- | --- |
| end | Lines are packed toward the end of the grid |

|  |  |
| --- | --- |
| center | Lines are packed toward the center of the grid |

|  |  |
| --- | --- |
| space-between | Lines are evenly distributed in the grid |

|  |  |
| --- | --- |
| space-around | Lines are evenly distributed in the grid,  with half-size spaces on either end |

|  |  |
| --- | --- |
| space-evenly | Lines are evenly distributed in the grid,  with equal space around them |

Now we are going to merge column1 to column3

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-1{

**column-start: 1;**

**column-end: 3;**

}

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-1{

**grid-column-start: 1;**

**grid-column-end: 3;**

}

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-1{

 /\*  grid-column-start: 1;

  grid-column-end: 3; \*/

  /\*Short cut to this is \*/

**grid-column:1/3;**

}

We can also write grid-column:1/span 2(i.e. how many column to merge)

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-1{

 /\*  grid-column-start: 1;

  grid-column-end: 3; \*/

  /\*Short cut to this is \*/

**grid-column:1/span 2;**

}

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-1{

**grid-row-start: 1;**

**grid-row-end: 4;**

}

Shortcut to grid row

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-1{

  /\* grid-row-start: 1;

  grid-row-end: 4;  \*/

**grid-row:1/4;**

}

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-1{

  /\* grid-row-start: 1;

  grid-row-end: 4;  \*/

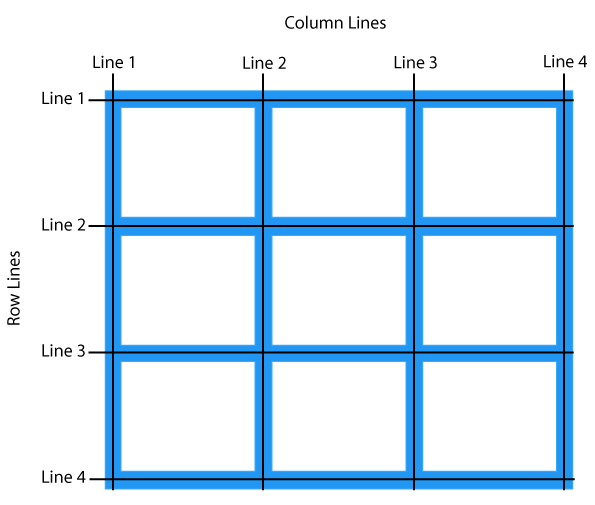
  grid-row:1/span 3;

}

# CSS grid-area Property

The grid-area property specifies a grid item's size and location in a grid layout, and is a shorthand property for the following properties:

* grid-row-start
* grid-column-start
* grid-row-end
* grid-column-end



<!DOCTYPE html>

<html>

<head>

    <title>Css Positioning Using Grid Area</title>

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

</head>

<body>

<div class="grid-container">

    <div class="grid-item" id="grid-1">first</div>

    <div class="grid-item" id="grid-2">second</div>

    <div class="grid-item" id="grid-3">third</div>

    <div class="grid-item" id="grid-4">fourth</div>

    <div class="grid-item" id="grid-5">fifth</div>

    <div class="grid-item" id="grid-6">Sixth</div>

    <!-- <div class="grid-item" id="grid-7">7</div>

    <div class="grid-item" id="grid-8">8</div>

    <div class="grid-item" id="grid-9">9</div> -->

</div>

</body>

</html>

**Style.css**

\*{

    margin:0;

    padding:0;

    box-sizing: border-box;

    font-family:Arial, Helvetica, sans-serif;

}

body{

    height:100%;

    width:100%;

}

.grid-container{

    background-color: #eee;

    padding:2rem;

    display:grid;

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

    grid-template-rows:repeat(2,150px);

  column-gap:1rem;

  row-gap:1rem;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly;

  }

  .grid-item{

    background-color: wheat;

    border:1px solid black;

    font-size:3rem;

    display:flex;

    justify-content: center;

    align-items:center;

    color:#fff;

    font-weight:bold;

  }

  #grid-1{

    background:orangered;

    /\* go to second 2 row \*/

    /\* grid-row-start:2 ;

    grid-row-end:3; \*/

    /\* go to second 2 row and column 3 \*/

    /\* grid-column-start:3 ;

    grid-column-end:4;  \*/

      /\* first row line /second column line/end row line/end column line \*/

    grid-area:2/3/3/4;

  }

  #grid-2{

    background:yellowgreen;

  }

  #grid-3{

    background:slateblue;

  }

  #grid-4{

    background:hotpink;

  }

  #grid-5{

    background:royalblue;

  }

  #grid-6{

    background:goldenrod;

    /\* go to first row \*/

    /\* grid-row-start:1 ;

    grid-row-end:2; \*/

    /\* go to first row second column \*/

    /\* grid-column-start:2 ;

    grid-column-end:3;  \*/

  }

**Another Example**

<!DOCTYPE html>

<html>

<head>

    <title>Document</title>

    <link rel="stylesheet" type="text/css" href="gridexample2.css"/>

</head>

<body>

<div class="grid-container">

    <div class="grid-item" id="grid-1">1</div>

    <div class="grid-item" id="grid-2">2</div>

    <div class="grid-item" id="grid-3">3</div>

    <div class="grid-item" id="grid-4">4</div>

    <div class="grid-item" id="grid-5">5</div>

    <div class="grid-item" id="grid-6">6</div>

    <div class="grid-item" id="grid-7">7</div>

    <div class="grid-item" id="grid-8">8</div>

    <div class="grid-item" id="grid-9">9</div>

</div>

</body>

</html>

**Style.css**

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-5{

  /\* first row line /second column line/end row line/end column line \*/

**grid-area: 1/2/3/4;**

}

Or use span to merge row and column

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-5{

  /\* first row line /second column line/end row line/end column line \*/

**grid-area: 1/2/span 2/span 3;**/\* span tell how many row or col to be merge \*/

}

# CSS grid-area Property

Now create new html file

<!DOCTYPE html>

<html>

<head>

    <title>Document</title>

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

</head>

<body>

<div class="grid-container">

    <div class="grid-item" id="grid-1">1</div>

    <div class="grid-item" id="grid-2">2</div>

    <div class="grid-item" id="grid-3">3</div>

    <div class="grid-item" id="grid-4">4</div>

    <div class="grid-item" id="grid-5">5</div>

    <!-- <div class="grid-item" id="grid-6">6</div>

    <div class="grid-item" id="grid-7">7</div>

    <div class="grid-item" id="grid-8">8</div>

    <div class="grid-item" id="grid-9">9</div> -->

</div>

</body>

</html>

The grid-area property can also be used to assign a name to a grid item. Named grid items can then be referenced to by the grid-template-areas property of the grid container.

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-1{

  grid-area: 'header';

}

#grid-2{

  grid-area: 'menu';

}

#grid-3{

  grid-area: 'main';

}

#grid-4{

  grid-area: 'right';

}

#grid-5{

  grid-area: 'footer';

}

Now comment following properties in grid-container

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  /\* grid-template-columns:200px 300px 200px;

   grid-template-rows:100px 200px auto;

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly; \*/

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-1{

  grid-area: header;

}

#grid-2{

  grid-area: menu;

}

#grid-3{

  grid-area: main;

}

#grid-4{

  grid-area: right;

}

#grid-5{

  grid-area: footer;

}

# CSS grid-template-areas Property

The grid-template-areas property specifies areas within the grid layout.

You can name grid items by using the grid-area property, and then reference to the name in the grid-template-areas property.

Each area is defined by apostrophes. Use a period sign to refer to a grid item with no name.

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  /\* grid-template-columns:200px 300px 200px;

   grid-template-rows:100px 200px auto;

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly; \*/

  grid-template-areas:

  'header header header header header header'

  'menu main main main right right'

  'menu footer footer footer footer footer'

  ;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-1{

  grid-area: header;

}

#grid-2{

  grid-area: menu;

}

#grid-3{

  grid-area: main;

}

#grid-4{

  grid-area: right;

}

#grid-5{

  grid-area: footer;

}

Now give grid-gap:10px in grid container

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  /\* grid-template-columns:200px 300px 200px;

   grid-template-rows:100px 200px auto;

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly; \*/

  grid-template-areas:

  'header header header header header header'

  'menu main main main right right'

  'menu footer footer footer footer footer'

  ;

  grid-gap: 10px;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-1{

  grid-area: header;

}

#grid-2{

  grid-area: menu;

}

#grid-3{

  grid-area: main;

}

#grid-4{

  grid-area: right;

}

#grid-5{

  grid-area: footer;

}

**Let take an example**

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Basics</title>

  <link rel="stylesheet" href="basic.css">

</head>

<body>

  <div class="grid-container">

    <div>1</div>

    <div>2</div>

    <div>3</div>

    <div>4</div>

    <div>5</div>

    <div>6</div>

  </div>

</body>

</html>

**Basic.css**

\*{

    margin:0px;

    padding:0px;

    box-sizing: border-box;

}

body {

  display:flex;

  justify-content:center ;

}

.grid-container {

  background: #eee;

  width: 1100px;

  display: grid;

  /\* grid-template-columns: 300px 300px 300px; \*/

  grid-template-columns: repeat(6, 1fr);

  padding:2rem;

}

.grid-container > div {

  background: #ccc;

  border: 1px solid #777;

  display:flex;

  justify-content: center;

  align-items:center;

  font-weight:bold;

  font-size:2rem;

}

/\* it merge first two column \*/

 .grid-container > div:nth-child(1) {

  grid-column: 1 / span 2;

 }

/\* It don't take first agrument in the property so it automatically start with next available space span 3 column \*/

/\* for first div item we can also write grid-column:span 2; \*/

.grid-container > div:nth-child(2) {

  grid-column: span 3;

}

/\* This property has no effect \*/

.grid-container > div:nth-child(3) {

    grid-column: span 1;

  }

 /\* This property change the position of the column from 1 to 2  \*/

 .grid-container > div:nth-child(4) {

  grid-column: 2 / 6;

}

/\* Now it start from 6 grid lines and space is not availabe in second row so it moves to the third row \*/

.grid-container > div:nth-child(5) {

  grid-column: span 3;

}

.grid-container > div:nth-child(6) {

  grid-column: span 3;

}

/\* Now give grid-template-rows: 100px 200px 300px; in container  \*/

/\* Now give gap:1rem or grid-gap:1rem \*/

/\* Now give property justify-items:stretch; (it is the default property of justify-items)

now give justify-items:end; now change to justify-items:start justify-items:center now change it default property

justify-items:stretch;

\*/

/\* Now give property align-items:stretch; (it is the default property of justify-items)

now give align-items:end; now change to align-items:start align-items:center now change it default property

align-items:stretch;

\*/

/\* Now apply align-self:end to div 4 \*/

/\* Now apply justify-self: end;

align-self: start; to div 6  \*/

# justify-items Property

justify-items aligns grid items along the row (inline) axis. Specifically, this property allows you to set alignment for items inside a grid container (not the grid itself) in a specific position (e.g. start, center and end) or with a behavior (e.g. stretch).

When justify-items is used, it also sets the default justify-self value for all grid items, though this can be overridden at the child level by using the justify-self property on the child itself.

# align-items Property

Aligns grid items along the block (column) axis (as opposed to justify-items which aligns along the inline (row) axis). This value applies to all grid items inside the container. This property allows you to set alignment for items inside a grid container (not the grid itself) in a specific position (e.g. start, center and end) or with a behavior (e.g. stretch).

When align-items is used, it also sets the default align-self value for all grid items, though this can be overridden at the child level by using the align-self property on the child itself.

# align-self Property

Aligns a grid item inside a cell along the block (column) axis (as opposed to justify-self which aligns along the inline (row) axis). This value applies to the content inside a single grid item.

# justify-self Property

Aligns a grid item inside a cell along the inline (row) axis (as opposed to align-self which aligns along the block (column) axis). This value applies to a grid item inside a single cell.

# place-self Property

place-self sets both the align-self and justify-self properties in a single declaration.

<align-self> / <justify-self> – The first value sets align-self, the second value justify-self. If the second value is omitted, the first value is assigned to both properties.

# CSS initial

The initial keyword is used to set a CSS property to its default value.

The initial keyword can be used for any CSS property, and on any HTML element.

# CSS inherit

The inherit keyword specifies that a property should inherit its value from its parent element.

The inherit keyword can be used for any CSS property, and on any HTML element.

Example

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

<div id="green">

  Green Text<br/>

  <a href="" id="blue">Link 1</a><br/>

  <a href="" id="initial">Link 2</a><br/>

  <a href="" id="inherit">Link 3</a><br/>

</div>

</body>

</html>

Mystyle.css

#green{

color:green;

}

#blue{

  color:red;

}

#initial{

  color:initial;

}

#inherit{

  color:inherit;

}

# CSS The object-fit Property

The CSS object-fit property is used to specify how an <img> or <video> should be resized to fit its container.

We see that the image is being squished to fit the container of 200x300 pixels (its original aspect ratio is destroyed).

Here is where the object-fit property comes in. The object-fit property can take one of the following values:

* contain - The image keeps its aspect ratio, but is resized to fit within the given dimension

Example

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

<img src="image/paris.jpg" alt=""/>

</body>

</html>

Mystyle.css

img{

  border:1px solid red;

  width:500px;

  height:400px;

  object-fit:contain;

}

fill - This is default. The image is resized to fill the given dimension. If necessary, the image will be stretched or squished to fit

img{

  border:1px solid red;

  width:500px;

  height:400px;

  object-fit:fill;

}

cover - The image keeps its aspect ratio and fills the given dimension. The image will be clipped (some portion will be cut) to fit

img{

  border:1px solid red;

  width:500px;

  height:300px;

  object-fit:cover;

}

# CSS object-position Property

The object-position property is used together with object-fit to specify how an <img> or <video> should be positioned with x/y coordinates inside its "own content box".

img{

  border:1px solid red;

  width:500px;

  height:300px;

  object-fit:cover;

  object-position: top center;

}

img{

  border:1px solid red;

  width:500px;

  height:300px;

  object-fit:contain;

  object-position: top left;

}

# CSS Transforms

CSS transforms allow you to move, rotate, scale, and skew elements. We cannot transform inline element to transform element should be either block or inline-block.

**Now to make anchor tag behave as button we have to do following task:**

**Example**

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

<div id="wrapper">

    <a href="">Click Here</a>

</div>

</body>

</html>

**Mystyle.css**

#wrapper{

  padding:20px;

  margin:20px;

}

a{

  text-decoration: none;

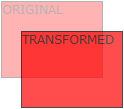
  background-color: lightcoral;

  padding:10px 20px;

  color:white;

}

## **The translate() Method**



The translate() method moves an element from its current position (according to the parameters given for the X-axis and the Y-axis). (right and down) value can be minus also

#wrapper{

  padding:20px;

  margin:20px;

}

a{

  text-decoration: none;

  background-color: lightcoral;

  padding:10px 20px;

  color:white;

  transform:translate(50px,50px);

}

It will not transform the element as anchor tag is inline element so first we have to make it inline-block

Now it will translate the anchor tag as it now become inline-block

#wrapper{

  padding:20px;

  margin:20px;

}

a{

  text-decoration: none;

  background-color: lightcoral;

  padding:10px 20px;

  color:white;

  transform:translate(50px,50px);

**display:inline-block**;

}

Now we transform the element through hover

#wrapper{

  padding:20px;

  margin:20px;

}

a{

  text-decoration: none;

  background-color: lightcoral;

  padding:10px 20px;

  color:white;

}

**a:hover{**

**display:inline-block;**

**transform:translate(50px,50px);**

**}**

## **The rotate() Method**



The rotate() method rotates an element clockwise or counter-clockwise according to a given degree.

The following example rotates the <div> element clockwise with 20 degrees:

### Example

#wrapper{

  padding:20px;

  margin:20px;

}

a{

  text-decoration: none;

  background-color: lightcoral;

  padding:10px 20px;

  color:white;

}

a:hover{

  display:inline-block;

  transform:rotate(20deg)

}

Using negative values will rotate the element counter-clockwise.

**Now make the angle -20deg**

#wrapper{

  padding:20px;

  margin:20px;

}

a{

  text-decoration: none;

  background-color: lightcoral;

  padding:10px 20px;

  color:white;

}

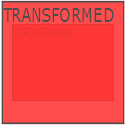
a:hover{

  display:inline-block;

**transform:rotate(-20deg)**

}

## **The scale() Method**



The scale() method increases or decreases the size of an element (according to the parameters given for the width and height).

The following example increases the <div> element to be two times of its original width, and three times of its original height:

### Example

div {  
  transform: scale(2, 3);  
}

#wrapper{

  padding:20px;

  margin:20px;

}

a{

  text-decoration: none;

  background-color: lightcoral;

  padding:10px 20px;

  color:white;

}

a:hover{

  display:inline-block;

**transform:scale(2,3);**

}

We can also decrease width and height of the element

#wrapper{

  padding:20px;

  margin:20px;

}

a{

  text-decoration: none;

  background-color: lightcoral;

  padding:10px 20px;

  color:white;

}

a:hover{

  display:inline-block;

**transform:scale(0.8,0.7);**

}

## **The scaleX() Method**

The scaleX() method increases or decreases the width of an element.

The following example increases the <div> element to be two times of its original width:

### Example

div {  
  transform: scaleX(2);  
}

#wrapper{

  padding:20px;

  margin:20px;

}

a{

  text-decoration: none;

  background-color: lightcoral;

  padding:10px 20px;

  color:white;

}

a:hover{

  display:inline-block;

**transform:scalex(2);**

}

## **The scaleY() Method**

The scaleY() method increases or decreases the height of an element.

The following example increases the <div> element to be three times of its original height:

### Example

div {  
  transform: scaleY(3);  
}

#wrapper{

  padding:20px;

  margin:20px;

}

a{

  text-decoration: none;

  background-color: lightcoral;

  padding:10px 20px;

  color:white;

}

a:hover{

  display:inline-block;

  transform:scaley(2);

}

## **The skew () Method**

The skew () method skews(tirsha) an element along the X-axis by the given angle.and y axis

The following example skews the <div> element 20 degrees along the X-axis and 10 degrees along y axis:

### Example

div {  
  transform: skew(20deg,10deg);  
}

#wrapper{

  padding:20px;

  margin:20px;

}

a{

  text-decoration: none;

  background-color: lightcoral;

  padding:10px 20px;

  color:white;

  display:inline-block;

  transform:skew(20deg,10deg);

}

/\* a:hover{

  display:inline-block;

  transform:scaley(2);

} \*/

## **The skewX() Method**

The skewX() method skews an element along the X-axis by the given angle.

The following example skews the <div> element 20 degrees along the X-axis:

### Example

div {  
  transform: skewX(20deg);  
}

## **The skewY() Method**

The skewY() method skews an element along the Y-axis by the given angle.

The following example skews the <div> element 20 degrees along the Y-axis:

### Example

div {  
  transform: skewY(20deg);  
}

# CSS transform-origin Property

The transform-origin property allows you to change the position of transformed elements.

2D transformations can change the x- and y-axis of an element. 3D transformations can also change the z-axis of an element.

## **CSS Syntax**

transform-origin: *x-axis y-axis z-axis*|initial|inherit;

## **Property Values**

|  |  |
| --- | --- |
| **Property Value** | **Description** |
| *x-axis* | Defines where the view is placed at the x-axis. Possible values:   * left * center * right * *length* * *%* |
| *y-axis* | Defines where the view is placed at the y-axis. Possible values:   * top * center * bottom * *length* * *%* |
|  |  |

We basically use % (percentage)

We rotate the element by 20 degree not from centre but 0 x-axis and 100 y-axis.

So the transform-origin:0% 100%

#wrapper{

  padding:20px;

  margin:20px;

}

a{

  text-decoration: none;

  background-color: lightcoral;

  padding:10px 20px;

  color:white;

  display:inline-block;

  transform:rotate(20deg);

  transform-origin: 0% 100%;

}

/\* a:hover{

  display:inline-block;

  transform:scaley(2);

} \*/

# CSS Transitions

CSS transitions allows you to change property values smoothly, over a given duration.

Example

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

<div id="div1">

</div>

</body>

</html>

Mystyle.css

#div1{

  height:100px;

  width:100px;

  background-color: red;

  margin:100px;

}

On hover we want to change width of the div

We have to apply two properties for smooth transition

The transition-property property specifies the name of the CSS property the transition effect is for (the transition effect will start when the specified CSS property changes).

The transition-duration property specifies how many seconds (s) or milliseconds (ms) a transition effect takes to complete.

#div1{

  height:100px;

  width:100px;

  background-color: red;

  margin:100px;

**transition-property:width;**

**transition-duration:1s;**

}

#div1:hover{

  width:200px;

}

Now in transition effect we want background-color.

#div1{

  height:100px;

  width:100px;

  background-color: red;

  margin:100px;

**transition-property:background-color;**

**transition-duration:1s;**

}

#div1:hover{

  background-color: green;

}

We can give multiple transition property

#div1{

  height:100px;

  width:100px;

  background-color: red;

  margin:100px;

**transition-property:width,background-color;**

**transition-duration:1s;**

}

#div1:hover{

  background-color: green;

  width:200px;

}

|  |  |
| --- | --- |
| transition-property:all | Default value. All properties will get a transition effect |

#div1{

  height:100px;

  width:100px;

  background-color: red;

  margin:100px;

**transition-property:all;**

  transition-duration:1s;

}

#div1:hover{

  background-color: green;

  width:200px;

}

We can give transition-duration in milliseconds for example

#div1{

  height:100px;

  width:100px;

  background-color: red;

  margin:100px;

  transition-property:width;

**transition-duration:50ms;**

}

#div1:hover{

  background-color: green;

  width:200px;

}

We can increase the duration in milli second

#div1{

  height:100px;

  width:100px;

  background-color: red;

  margin:100px;

  transition-property:width;

**transition-duration:500ms;**

}

#div1:hover{

  background-color: green;

  width:200px;

}

# CSS transition-delay Property

The transition-delay property specifies when the transition effect will start.

The transition-delay value is defined in seconds (s) or milliseconds (ms)

#div1{

  height:100px;

  width:100px;

  background-color: red;

  margin:100px;

  transition-property:all;

  transition-duration:500ms;

**transition-delay:500ms**

}

#div1:hover{

   width:200px;

}

# Print Specific Style in CSS

The @media rule is used in media queries to apply different styles for different media types/devices. (We basically used to print the data)

@media screen is for screen

@media print is for print

Example

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

<div id="div1">

    <img src="image/bird.jpg" alt="">

    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Nostrum cupiditate veniam autem commodi inventore iure accusantium, consequatur dolor iusto alias molestiae, ipsum maiores odit tempora aperiam earum harum, laudantium quo beatae amet aliquam. Numquam consequatur enim eaque! Fugit consectetur eaque odio perspiciatis, eum excepturi, eius tempore voluptate reiciendis magnam rerum quidem in provident optio aliquid exercitationem quaerat labore ullam quia omnis cum! Sit voluptas obcaecati aut iure voluptate voluptates reprehenderit reiciendis odio, deserunt dolor officiis accusamus eius officia necessitatibus assumenda possimus mollitia cupiditate ad ex fugit sequi nobis. Quaerat laboriosam itaque, tenetur debitis praesentium aperiam nemo laborum. Ratione ipsa ad illo doloremque dolor libero fugiat accusantium dolorum iste odit. Dolorum cum aut laboriosam omnis officiis quaerat magnam vero soluta dolorem nisi unde harum, quisquam illum fuga, illo ducimus sapiente ut! Autem corporis aspernatur, consectetur facere quos odit expedita optio nobis ipsa eaque, quasi deleniti error quidem veritatis blanditiis iste sapiente distinctio fugiat quod nemo velit doloribus. Amet placeat est officiis, mollitia libero eveniet tempora, voluptate quas explicabo temporibus maxime soluta. Nesciunt eum quaerat voluptatibus similique, adipisci cupiditate autem officiis minima expedita eos veritatis voluptate iste ab mollitia aperiam ducimus earum nobis delectus dignissimos laudantium. Debitis rerum fuga illo minus inventore iure nihil quas quidem vel doloremque distinctio doloribus nulla quis, delectus quae ipsa alias, praesentium quibusdam? Iusto, quis mollitia, sunt nobis nulla magni architecto laborum ducimus iure, veritatis repellendus suscipit. Dicta sit cum numquam sapiente quod libero tempora dolore nihil, eius quo. Recusandae ipsam maxime molestias. Deleniti, amet rem sunt tempora voluptatibus aspernatur, repellendus ad consectetur exercitationem natus illum veniam quisquam? Quas libero omnis, sunt illo porro reiciendis corrupti molestias quasi modi, iusto itaque! Odit perspiciatis iste excepturi quisquam laborum in minus quae libero ipsa voluptatibus nulla, rem deleniti fugiat! Ad porro odit sed magni maiores. Doloremque, eligendi totam debitis hic obcaecati consectetur ipsum pariatur amet, illum officia quam labore omnis in veritatis sit eaque itaque exercitationem accusantium quo qui est maiores, cum animi officiis. Ullam quas, alias, id necessitatibus qui, inventore minus consequatur sequi ducimus eveniet dignissimos officiis sunt? Nihil ex eaque voluptatum, sint quo maiores ea commodi asperiores placeat magni eveniet explicabo. Iure magni quibusdam nisi, nam eos, perferendis error sed ab culpa sunt laborum! Iste enim nobis laboriosam omnis obcaecati voluptatibus sunt quis quaerat provident iure tempora veniam unde amet, blanditiis nesciunt perspiciatis, accusamus totam, sed aut nulla ullam. Pariatur modi aspernatur aliquid eaque tenetur repellat deserunt qui animi, quia minima. Blanditiis corporis nam velit harum natus soluta aliquam, deleniti voluptatum magni ea odit omnis beatae, quod labore iste! Cumque sunt suscipit beatae eveniet dignissimos facilis velit sit earum cupiditate ullam, tempore numquam repellat eius magnam quibusdam mollitia cum voluptatem, obcaecati at tempora. Eligendi blanditiis exercitationem dolore quibusdam obcaecati, libero repellat distinctio vel tempore possimus non accusantium praesentium voluptates aliquid corporis optio pariatur! Repudiandae impedit sunt facere, odio cum rem commodi veritatis hic aliquam deleniti, nobis quam iste laudantium accusamus? Et quidem aliquid, laborum alias voluptatibus numquam eius, iusto sequi error deserunt qui animi sint quam maxime est soluta non adipisci atque sit! Rem molestias dolorum quibusdam in voluptatibus laboriosam odit dicta quaerat voluptas veniam ea animi amet beatae aperiam sequi voluptates minima, itaque deserunt mollitia earum culpa labore dolore! Veniam harum nobis aut officia expedita, fuga asperiores perspiciatis eius architecto a error? Inventore ducimus, non unde reprehenderit temporibus quibusdam mollitia omnis exercitationem pariatur, aliquid aut maiores suscipit ex itaque accusamus vero excepturi dolor molestiae quis eum. Totam eligendi ipsam amet modi consequuntur alias necessitatibus quisquam consequatur accusamus vel iure sapiente dolores perferendis iusto non inventore asperiores aperiam animi, consectetur voluptates fugiat laborum commodi molestias unde. Vero possimus odio unde tenetur ipsum eveniet doloribus exercitationem mollitia reprehenderit iure qui quod voluptate, totam excepturi modi soluta deserunt. Fugit, a. At incidunt quia, repudiandae iste voluptatem quo harum cumque excepturi dolorum veritatis soluta aspernatur vel, repellendus dolor expedita. Necessitatibus sapiente officiis debitis cupiditate quo obcaecati odio minima illo ducimus voluptatum quis reiciendis quia vel quisquam tempore exercitationem, dicta nihil! Molestias, commodi, nemo porro aliquam repellat perferendis totam aperiam expedita officia culpa, rerum necessitatibus! Amet, laborum veniam porro quo velit nisi provident, fugit, consequatur temporibus rerum soluta quis iusto maxime? A laudantium quam hic magnam ullam illum labore numquam facilis cupiditate maxime, velit consectetur eos quo perspiciatis ea? Sequi exercitationem laboriosam impedit odio animi natus corporis, enim non iusto accusamus inventore. Maiores itaque quis enim, impedit odio obcaecati repellat culpa voluptate sapiente. Eaque et consequuntur quaerat itaque culpa vel quisquam quasi odit doloremque nemo. Voluptate dignissimos aspernatur nam vitae, dolor dicta ipsum itaque reprehenderit, dolorem illum nesciunt enim nihil! Optio vel placeat soluta sed explicabo quae, ab aliquid et excepturi dolore voluptatibus non animi! Fugiat laudantium veritatis nulla aliquam illum eum necessitatibus, reprehenderit cupiditate deleniti non delectus voluptatibus perferendis magni ab amet ipsa? Suscipit, blanditiis! Minima, possimus quaerat similique odit nulla dolorem officiis id amet. Nulla in animi ipsam! Illo repudiandae id ipsam non sequi, ad iure alias doloremque quasi animi et sed aperiam, ea tempore blanditiis inventore quia repellat nobis cum aliquid! Deserunt totam eaque necessitatibus beatae dolor dolore nihil, laudantium odit quasi perspiciatis numquam accusamus facilis tempore laborum corrupti nam similique itaque ullam quaerat tempora vitae. Quibusdam suscipit dicta dignissimos pariatur assumenda soluta enim repellat veniam laudantium adipisci, autem a. Nam a necessitatibus corporis tempora nesciunt beatae, dolor placeat porro laboriosam id! Quis soluta dignissimos a at adipisci eaque minus veritatis ipsam eius minima voluptatibus similique quos porro nam exercitationem esse suscipit molestias reprehenderit, provident voluptatem? Deleniti odio voluptates corrupti sed voluptatum laudantium in nesciunt quo atque ullam assumenda magni est, tempora enim sapiente modi! Id error excepturi ipsum magnam quibusdam doloribus quod dolores impedit voluptatibus quae quas minima architecto laboriosam dolore nulla nisi ex, harum et iusto cupiditate eius debitis vero. Maxime nobis, voluptas commodi eum itaque minima nulla ipsam quasi assumenda praesentium est earum accusantium nemo accusamus unde repellendus soluta voluptatibus incidunt dolor a tenetur? Quaerat corporis accusantium ea voluptatibus eius rerum a sequi, doloribus recusandae accusamus. Accusantium dolorem dolore vero, quibusdam excepturi tempore iste! Quae.

    </p>

</div>

</body>

</html>

**Mystyle.css**

@media screen{

  p{

    font-size:20px;

  }

}

@media print{

  p{

    font-size:30px;

  }

  img{

    display:none;

  }

  div{

    width:50%;

  }

}

# CSS Multiple Columns

The CSS multi-column layout allows easy definition of multiple columns of text - just like in newspapers:

The column-count property specifies the number of columns an element should be divided into.

Example

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

<div>

    Lorem ipsum dolor sit amet consectetur adipisicing elit. Iure repudiandae, quisquam dolorem quis ipsam dolor eaque quasi! Accusamus quas blanditiis nobis fuga recusandae officia iusto facilis, officiis laborum provident. Molestiae voluptate id velit dolores a eaque, fuga odio tenetur dolore ipsam in repellendus, dolorem, rerum reiciendis fugiat quo? Libero voluptatem labore maxime officiis perspiciatis, laborum sint dicta harum esse deleniti praesentium sunt recusandae ipsam, reiciendis et impedit? Quaerat ipsum, non amet harum quo nam. Assumenda aspernatur animi, aliquid quidem a voluptate in eligendi eius ipsam quis iure dolore atque, quasi debitis fugit facilis magnam totam numquam blanditiis earum, eaque dicta. Sit, temporibus vero iste eveniet possimus et. Rerum odio nemo ea, laboriosam odit magnam accusamus numquam pariatur quos eius aliquid laborum, quis dolores! Vero consequuntur, ipsam necessitatibus consequatur suscipit, quae blanditiis obcaecati quam illum repudiandae expedita. Maiores ipsam deleniti aspernatur, esse blanditiis autem iusto perferendis voluptates. Ullam reprehenderit aliquid voluptatum, magnam iste sint. Magnam, omnis? Quia omnis accusamus velit error ratione impedit nisi voluptas aut aperiam iusto voluptatem, itaque laudantium officia, unde fugit delectus rem quas deserunt minus maxime, dignissimos molestias? Porro expedita autem dolor dolorum similique natus commodi, explicabo, illo facilis voluptas aperiam et, cum quasi itaque id aliquam?

</div>

</body>

</html>

Mystyle.css

div{

  column-count: 3;

}

# CSS column-gap Property

The column-gap property specifies the gap between the columns in grid

div{

  column-count: 3;

  column-gap:50px;

}

# CSS column-rule-style Property

The column-rule-style property specifies the style of the rule between columns.

div{

  column-count: 3;

  column-gap:50px;

  column-rule-style:solid;

}

# CSS column-rule-width Property

The column-rule-width property specifies the width of the rule between columns.

div{

  column-count: 3;

  column-gap:50px;

  column-rule-style:solid;

  column-rule-width:1px;

}

# CSS column-rule-color Property

The column-rule-color property specifies the color of the rule between columns.

Short Cut for column-rule

div{

  column-count: 3;

  column-gap:50px;

  /\* column-rule-style:solid;

  column-rule-width:1px;

  column-rule-color:red; \*/

  column-rule:1px solid red;

}

Now give heading in the div

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Document</title>

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

 </head>

<body>

<div>

    <h1>Lorem ipsum dolor sit, amet consectetur adipisicing elit. Quisquam officia fuga voluptatum quam quidem aliquam nulla dolores dolorum, praesentium id.</h1>

    Lorem ipsum dolor sit amet consectetur adipisicing elit. Iure repudiandae, quisquam dolorem quis ipsam dolor eaque quasi! Accusamus quas blanditiis nobis fuga recusandae officia iusto facilis, officiis laborum provident. Molestiae voluptate id velit dolores a eaque, fuga odio tenetur dolore ipsam in repellendus, dolorem, rerum reiciendis fugiat quo? Libero voluptatem labore maxime officiis perspiciatis, laborum sint dicta harum esse deleniti praesentium sunt recusandae ipsam, reiciendis et impedit? Quaerat ipsum, non amet harum quo nam. Assumenda aspernatur animi, aliquid quidem a voluptate in eligendi eius ipsam quis iure dolore atque, quasi debitis fugit facilis magnam totam numquam blanditiis earum, eaque dicta. Sit, temporibus vero iste eveniet possimus et. Rerum odio nemo ea, laboriosam odit magnam accusamus numquam pariatur quos eius aliquid laborum, quis dolores! Vero consequuntur, ipsam necessitatibus consequatur suscipit, quae blanditiis obcaecati quam illum repudiandae expedita. Maiores ipsam deleniti aspernatur, esse blanditiis autem iusto perferendis voluptates. Ullam reprehenderit aliquid voluptatum, magnam iste sint. Magnam, omnis? Quia omnis accusamus velit error ratione impedit nisi voluptas aut aperiam iusto voluptatem, itaque laudantium officia, unde fugit delectus rem quas deserunt minus maxime, dignissimos molestias? Porro expedita autem dolor dolorum similique natus commodi, explicabo, illo facilis voluptas aperiam et, cum quasi itaque id aliquam?

</div>

</body>

</html>

# CSS column-span Property

The column-span property specifies how many columns an element should span across

Mystyle.css

div{

  column-count: 3;

  column-gap:50px;

  /\* column-rule-style:solid;

  column-rule-width:1px;

  column-rule-color:red; \*/

  column-rule:1px solid red;

}

h1{

  column-span: all;

}

# CSS Grid Layout Module

CSS Grid is the new Layout Model in CSS

From Oct 2017 all major browser started supporting CSS Grid

First we design with the help of table then by div, after div come flex and now come CSS Grid.

The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

## **Grid Elements**

A grid layout consists of a parent element, with one or more child elements.

# Difference between CSS Grid and CSS Flexbox

CSS Grid Layout, is a two-dimensional grid-based layout system with rows and columns, making it easier to design web pages without having to use floats and positioning. Like tables, grid layout allow us to align elements into columns and rows.

To get started you have to define a container element as a grid with **display: grid,** set the column and row sizes with grid-template-columns and grid-template-rows, and then place its child elements into the grid with grid-column and grid-row.

The CSS Flexbox offers a one-dimensional layout. It is helpful in allocating and aligning the space among items in a container (made of grids). It works with all kinds of display devices and screen sizes.

To get started you have to define a container element as a grid with **display: flex;**

**Uniqueness In Grid And Flexbox:**

**One Vs Two Dimension:**

* Grid is made for two-dimensional layout while Flexbox is for one. This means Flexbox can work on either row or columns at a time, but Grids can work on both.
* Flexbox, gives you more flexibility while working on either element (row or column). HTML markup and CSS will be easy to manage in this type of scenario.
* GRID gives you more flexibility to move around the blocks irrespective of your HTML markup.

**Content-First vs Layout-First:**

* The Flexbox layout is best suited to application components and small-scale layouts, while the Grid layout is designed for larger-scale layouts that are not linear in design.

Both Flexbox and css grid has its own importance and should be used for designing a web page

Grid words for 2 dimensional aligment

Flexbox is good space distrubtion

**Grid Columns**

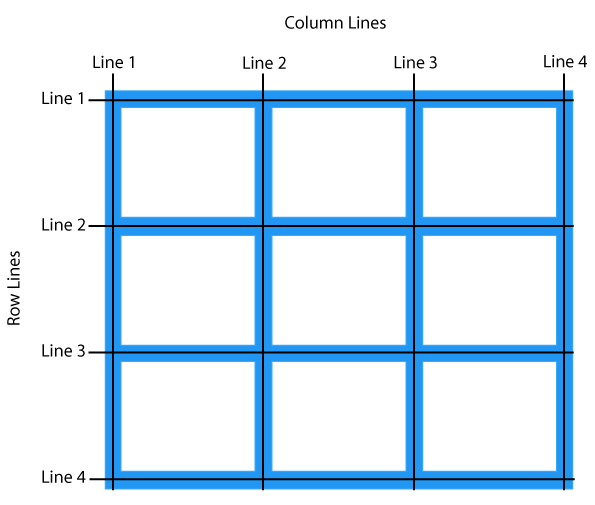
The vertical lines of grid items are called columns

The horizontal lines of grid items are called rows.

The space between each column/row are called gaps

The line between columns are called column lines

The line between rows are called row lines



**Difference between inline-grid and grid**

The difference between the values inline-grid and grid is that the inline-grid will make the element inline while grid will make it a block-level element.

Inline-grid will depent on content size,while grid will take full width container.

Example

<!DOCTYPE html>

<html>

<head>

    <title>Document</title>

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

</head>

<body>

<div class="grid-container">

    <div class="grid-item" id="grid-1">1</div>

    <div class="grid-item" id="grid-2">2</div>

    <div class="grid-item" id="grid-3">3</div>

    <div class="grid-item" id="grid-4">4</div>

    <div class="grid-item" id="grid-5">5</div>

    <div class="grid-item" id="grid-6">6</div>

    <div class="grid-item" id="grid-7">7</div>

    <div class="grid-item" id="grid-8">8</div>

    <div class="grid-item" id="grid-9">9</div>

</div>

</body>

</html>

Mystyle.css

.grid-container{

  background-color: indianred;

  padding:20px;

  display:inline-grid;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

# CSS grid-template-columns Property

The grid-template-columns property specifies the number (and the widths) of columns in a grid layout.

The values are a space separated list, where each value specifies the size of the respective column.

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 400px;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

It automatically divide the div into column

|  |  |
| --- | --- |
| auto | The size of the columns is determined by the size of the container and  on the size of the content of the items in the column,it try to occupy whole space |

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 400px auto;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

If you want all column to equal in size , apply auto to each column

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:auto auto auto;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

We can give column value in percentage

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:**auto 50% auto;**

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

We can give as many column value as we can , we give another column of 50px.

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:auto 50% auto 50px;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

# CSS grid-template-rows Property

The grid-template-rows property specifies the number (and the heights) of the rows in a grid layout.

The values are a space-separated list, where each value specifies the height of the respective row.

|  |  |
| --- | --- |
| auto | The size of the rows is determined by the size of the container,  and on the size of the content of the items in the row |

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:auto 50% auto;

  grid-template-rows:100px 200px auto;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

# CSS grid-column-gap Property

The grid-column-gap property defines the size of the gap between the columns in a grid layout.

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:auto 50% auto;

  grid-template-rows:100px 200px auto;

  grid-column-gap:30px;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

# CSS grid-row-gap Property

The grid-row-gap property defines the size of the gap between the rows in a grid layout.

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:auto 50% auto;

  grid-template-rows:100px 200px auto;

  grid-column-gap:30px;

  grid-row-gap:10px;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

Now make column-gap 10px and row-gap 10px and grid-template-column: 200px 300px 200px

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  grid-template-rows:100px 200px auto;

  grid-column-gap:10px;

  grid-row-gap:10px;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

# CSS justify-content Property

The justify-content property aligns the flexible container's items when the items do not use all available space on the main-axis (horizontally).

|  |  |
| --- | --- |
| start | Default value. Items are positioned at the beginning of the container |
| end | Items are positioned at the end of the container |

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  grid-template-rows:100px 200px auto;

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: start;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  grid-template-rows:100px 200px auto;

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: end;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

|  |  |
| --- | --- |
| center | Items are positioned in the center of the container |

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  grid-template-rows:100px 200px auto;

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: center;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

|  |  |
| --- | --- |
| space-between | Items will have space between them |

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  grid-template-rows:100px 200px auto;

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-between;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

|  |  |
| --- | --- |
| space-around | Items will have space before, between, and after them |

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  grid-template-rows:100px 200px auto;

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-around;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

|  |  |
| --- | --- |
| space-evenly | Items will have equal space around them |

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  grid-template-rows:100px 200px auto;

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

Now give height to .grid-container as height:600px and remove grid-template-rows property

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:600px;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

# CSS align-content Property

The align-content property can also be used on a grid container to align grid items in the block direction.

|  |  |
| --- | --- |
| start | Lines are packed toward the start of the grid |

And also decrease the size of container to 400px

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

**align-content:start;**

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

|  |  |
| --- | --- |
| end | Lines are packed toward the end of the grid |

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

**align-content:end;**

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

|  |  |
| --- | --- |
| center | Lines are packed toward the center of the grid |

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:center;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

|  |  |
| --- | --- |
| space-between | Lines are evenly distributed in the grid |

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

**align-content:space-between;**

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

|  |  |
| --- | --- |
| space-around | Lines are evenly distributed in the grid,  with half-size spaces on either end |

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

**align-content:space-around;**

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

|  |  |
| --- | --- |
| space-evenly | Lines are evenly distributed in the grid,  with equal space around them |

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

**align-content:space-evenly;**

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

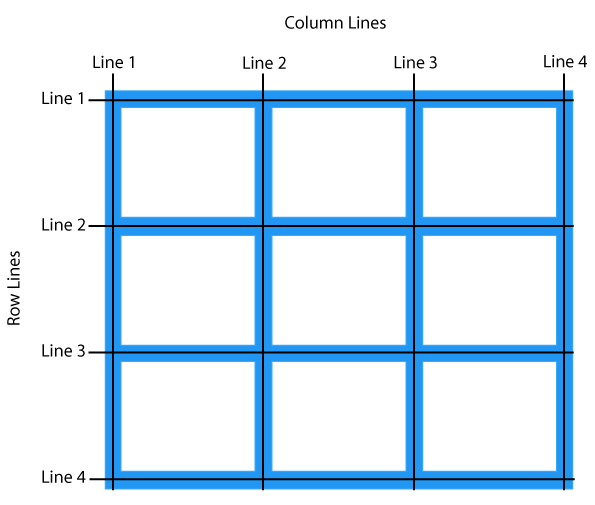
  text-align: center;

}

## **Grid Lines**

The lines between columns are called column lines.

The lines between rows are called row lines.



Now we are going to merge column1 to column3

# CSS grid-column-start Property

The grid-column-start property defines on which column-line the item will start.

# CSS grid-column-end Property

The grid-column-end property defines how many columns an item will span, or on which column-line the item will end( how many column will merge)

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-1{

**grid-column-start: 1;**

**grid-column-end: 3;**

}

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-1{

**grid-column-start: 1;**

**grid-column-end: 3;**

}

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-1{

 /\*  grid-column-start: 1;

  grid-column-end: 3; \*/

  /\*Short cut to this is \*/

**grid-column:1/3;**

}

We can also write grid-column:1/span 2(i.e. how many column to merge)

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-1{

 /\*  grid-column-start: 1;

  grid-column-end: 3; \*/

  /\*Short cut to this is \*/

**grid-column:1/span 2;**

}

# CSS grid-row-start Property

The grid-row-start property defines on which row-line the item will start.

# CSS grid-row-end Property

The grid-row-end property defines how many rows an item will span, or on which row-line the item will end (how many rows to be merge)

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-1{

**grid-row-start: 1;**

**grid-row-end: 4;**

}

Shortcut to grid row

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-1{

  /\* grid-row-start: 1;

  grid-row-end: 4;  \*/

**grid-row:1/4;**

}

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-1{

  /\* grid-row-start: 1;

  grid-row-end: 4;  \*/

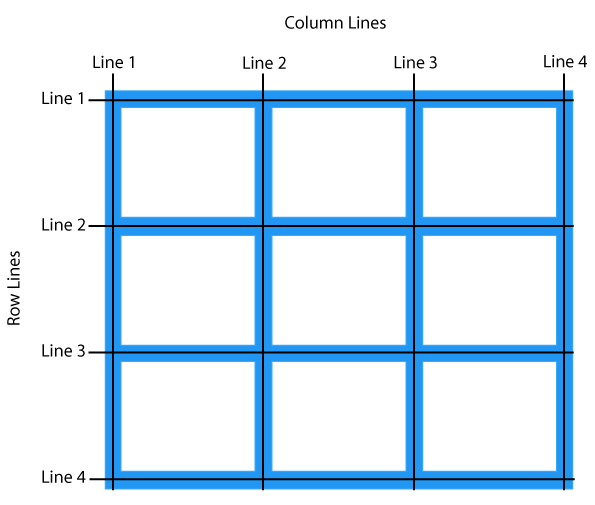
  grid-row:1/span 3;

}

# CSS grid-area Property

The grid-area property specifies a grid item's size and location in a grid layout, and is a shorthand property for the following properties:

* grid-row-start
* grid-column-start
* grid-row-end
* grid-column-end



.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-5{

  /\* first row line /second column line/end row line/end column line \*/

**grid-area: 1/2/3/4;**

}

Or use span to merge row and column

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-5{

  /\* first row line /second column line/end row line/end column line \*/

**grid-area: 1/2/span 2/span 3;**/\* span tell how many row or col to be merge \*/

}

# CSS grid-area Property

Now create new html file

<!DOCTYPE html>

<html>

<head>

    <title>Document</title>

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

</head>

<body>

<div class="grid-container">

    <div class="grid-item" id="grid-1">1</div>

    <div class="grid-item" id="grid-2">2</div>

    <div class="grid-item" id="grid-3">3</div>

    <div class="grid-item" id="grid-4">4</div>

    <div class="grid-item" id="grid-5">5</div>

    <!-- <div class="grid-item" id="grid-6">6</div>

    <div class="grid-item" id="grid-7">7</div>

    <div class="grid-item" id="grid-8">8</div>

    <div class="grid-item" id="grid-9">9</div> -->

</div>

</body>

</html>

The grid-area property can also be used to assign a name to a grid item. Named grid items can then be referenced to by the grid-template-areas property of the grid container.

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  grid-template-columns:200px 300px 200px;

  /\* grid-template-rows:100px 200px auto; \*/

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-1{

  grid-area: 'header';

}

#grid-2{

  grid-area: 'menu';

}

#grid-3{

  grid-area: 'main';

}

#grid-4{

  grid-area: 'right';

}

#grid-5{

  grid-area: 'footer';

}

Now comment following properties in grid-container

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  /\* grid-template-columns:200px 300px 200px;

   grid-template-rows:100px 200px auto;

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly; \*/

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-1{

  grid-area: header;

}

#grid-2{

  grid-area: menu;

}

#grid-3{

  grid-area: main;

}

#grid-4{

  grid-area: right;

}

#grid-5{

  grid-area: footer;

}

# CSS grid-template-areas Property

The grid-template-areas property specifies areas within the grid layout.

You can name grid items by using the grid-area property, and then reference to the name in the grid-template-areas property.

Each area is defined by apostrophes. Use a period sign to refer to a grid item with no name.

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  /\* grid-template-columns:200px 300px 200px;

   grid-template-rows:100px 200px auto;

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly; \*/

  grid-template-areas:

  'header header header header header header'

  'menu main main main right right'

  'menu footer footer footer footer footer'

  ;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-1{

  grid-area: header;

}

#grid-2{

  grid-area: menu;

}

#grid-3{

  grid-area: main;

}

#grid-4{

  grid-area: right;

}

#grid-5{

  grid-area: footer;

}

Now give grid-gap:10px in grid container

.grid-container{

  background-color: indianred;

  padding:20px;

  display:grid;

  /\* grid-template-columns:200px 300px 200px;

   grid-template-rows:100px 200px auto;

  grid-column-gap:10px;

  grid-row-gap:10px;

  justify-content: space-evenly;

  height:400px;

  align-content:space-evenly; \*/

  grid-template-areas:

  'header header header header header header'

  'menu main main main right right'

  'menu footer footer footer footer footer'

  ;

  grid-gap: 10px;

}

.grid-item{

  background-color: wheat;

  border:1px solid black;

  padding:20px;

  font-size:40px;

  text-align: center;

}

#grid-1{

  grid-area: header;

}

#grid-2{

  grid-area: menu;

}

#grid-3{

  grid-area: main;

}

#grid-4{

  grid-area: right;

}

#grid-5{

  grid-area: footer;

}

**To prepare a chessboard**

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <meta http-equiv="X-UA-Compatible" content="ie=edge" />

    <title>HTML Tutorial Step 1</title>

    <link rel="stylesheet" href="styles.css" />

  </head>

  <body>

    <div board" class="board">

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

      <div class="piece-square dark-square"></div>

      <div class="piece-square light-square"></div>

    </div>

  </body>

</html>

**Styles.css**

.board {

  height: 400px;

  width: 400px;

  display: block;

  border: 15px solid black;

}

.piece-square {

  width: 50px;

  height: 50px;

  box-sizing: border-box;

  position: relative;

  display: flex;

  justify-content: center;

  align-items: center;

  float: left;

}

.light-square {

  background: blue;

}

.dark-square {

  background: red;

}