FULLSTACK CSS

INDEX: -

Introduction CSS3: -

it was first introduced in 1994. The CSS1 was developed by World Wide Web Consortium (W3C) in 1996. W3C is standards organization for the World Wide Web. CSS2 specification was finalized in 1997. W2C start work on CSS3 but they put CSS3 module in draft because of inconsistency in Internet Explorer and start working on CSS2.1, but after 10 years other browsers start implementing CSS3 modules. CSS3 was released in 2005. It stands for Cascading Style Sheet.

Creating Cascading Styles: -

Inline CSS: - it declares with style attribute.
Ex: - <div style=" color:red; "> … … … <div>

Internal CSS: - it declares with style tag in head tag.
Ex: -
```
<head>
        <style>
                #d1{
                … … …
                }
        </style>
</head>
<div id='d1'>… … …</div>
```

External File CSS: - create another file to write CSS code with .css extension. Link that file in html file for apply CSS style from that file.
Ex: -
```
<head>
        <link href="1style.css" rel="stylesheet">
</head>
```

CSS Comments: - it is used for write information about code, it is multi-line comment.
/* … … …*/

Ex: -
```
<style>
/* … this is CSS comment … */
</style>
```

CSS Properties: - its properties define as follow

```
Selector{
        Property1: values;
        Preperty2: values;
        …
}
```
Ex: -

```
#div1{
	background-color: antiquewhite; //background color of element
	width: 700px; //width of element
	height: 300px; //height of element
	border: 1px solid black; //border of element
}
```

Basic Css properties: -

Properties: -

border: 1px solid black;
color: white;
background-color: red;
height: 150px;
width: 300px;
text-align: center;
font-size: 40px;

Types of CSS Selectors: -

DIV and SPAN: -

DIV: - it is container tag that can style by CSS properties. It takes all space that after it. It takes all allocated space by CSS style on webpage.
Ex: - <div>… … …</div>

SPAN: - it is container tag that can style by CSS properties. It does not take all space that after it. It takes only space that contains other html elements or texts, on webpage.
Ex: - <span>… … …</ span >

Id selector: - id=”name”; in CSS = #name (access with # hashtag)
Ex: -
```
#box{
… … …
}
<div id=”box”>… … …</div>
```

Element/Tag selector: - div, body, etc. in CSS = div, body
Ex: -
```
div{
… … …
}
<div>… … …</div>
```

Class selector: - class=”class1 class2 class3” in CSS = .class1 .class2 .class3 (access with .dot)
Ex: -

```
.box{
… … …
}
<div class=”box”>… … …</div>
```

Ex: - target particular element with same class.

```
div.box{
… … …
}
<div class=”box”>… … …</div>
<span class=”box”>… … …</span>
```

Universal selector: - in CSS *{….} (access with *)

Ex: -

```
*{
… … …
}
```

Group selector: - in CSS h1, h2, p {….}

Ex: -

```
h1, h2, p {
… … …
}

<h1>… … …</h1>
<h2>… … …</h2>
<p >… … …</p>
```

Selector precedence: - in CSS is precedence set for, which selector style will apply to element.

1) **Inline CSS: -** it has higher precedence in CSS. **Ex: -** style="color: red;"
2) **Id selectors: -** it has lower precedence than inline CSS. **Ex: -** #d1{color:green;}
3) **Class selectors: -** it has lower precedence than id selectors. **Ex: -** .div1{color:yellow;}
4) **Element selectors: -** it has lower precedence than class selectors. **Ex: -** div{color:yellow;}

CSS Float: - it set hoe element should float, it is used for positioning and formatting content.
float: value; //left,right,none;
Ex: - float: left;
Left: - float element to left side of parent element.
Right: - float element to right side of parent element.
None: - remove float value it is apply.

If ‘element without float property’ come after ‘element with float property’, it will get overlapped by element that have float property.

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

clear: value; //none, both, left, right
Ex: - clear: both;

- **None: -** The element is not pushed below left or right floated elements. This is default
- **Left: -** The element is pushed below left floated elements
- **Right: -** The element is pushed below right floated elements
- **Both: -** The element is pushed below both left and right floated elements

<br clear="all"> also can clear floating value using br element attributes.

CSS Height/Width: -

Height: - it set height of element.
height: 100px;
height: 30%; // it can also set in percentage, it take percent from parent element height.
height: 100vh; vh= viewport height // it is height of browser web screen.

Width: - it set width of element.
width: 100px;
width: 30%; // it can also set in percentage, it take percent from parent element width.
width: 100vw; vw= viewport height // it is width of browser web screen.

Measurement can also define in CSS as follow: -
cm centimeters **ex: -** height: 2cm;
mm millimeters **ex: -** height: 2mm;
in inches **ex: -** width: 2in;   // 1in =2.54cm
pt points **ex: -** width: 2pt; // 1pt = 1/72 of 1in
px pixels **ex: -** width: 2px; // 1px = 1/96 of 1in
pc picas **ex: -** width: 2pc;   // 1pc =12 pt

CSS Borders: -

border: size style color;

border-style: value; dashed, dotted, double, groove, hidden, inset, outset, none, ridge.
Border-style: top/bottom-style left/right-style;
border-style: top–style right-left-style bottom-style;
border-style: top right bottom left;

border-width: distance; thin, thick, medium
border-width: top/bottom-width left/right-width;
border-width: top–width right-left-width bottom-width;
border-width: top right bottom left;

border-color: color_name;
border-color: top/bottom-color left/right-color;
border-color: top-color right-left-color bottom-color;
border-color: top right bottom left;

border-left: 10px solid green;
border-top: 5px solid blue;
border-bottom: 2px solid black;
border-right: 20px solid green;

change particular: -
border-top-width: 5px;
border-top-style: double;
border-top-color: green;
same will work for right bottom and left side.

Curve-border: -
border-radius: distance; px, %
border-top-left-radius: 20px;
border-top-right-radius: 5px;
border-bottom-left-radius: 40px;
border-bottom-right-radius: 10px;

border-radius: - top-left/bottom-right top-right/bottom-left;
Ex: - border-radius: 10px 50px;
border-radius: - top-left top-right/bottom-left bottom-right;
Ex: - border-radius: 10px 50px 30px;
border-radius: - top-left top-right bottom-right bottom-left;
Ex: - border-radius: 10px 50px 30px 100px;

CSS Margin AND Padding: -

Margin: - it separate element from other element by applied distance.
margin: distance; (from all side)
margin-left: distance;
margin-right: distance;
margin-top: distance;
margin-bottom: distance;
margin: top/bottom   left/right;
margin: top right bottom left;

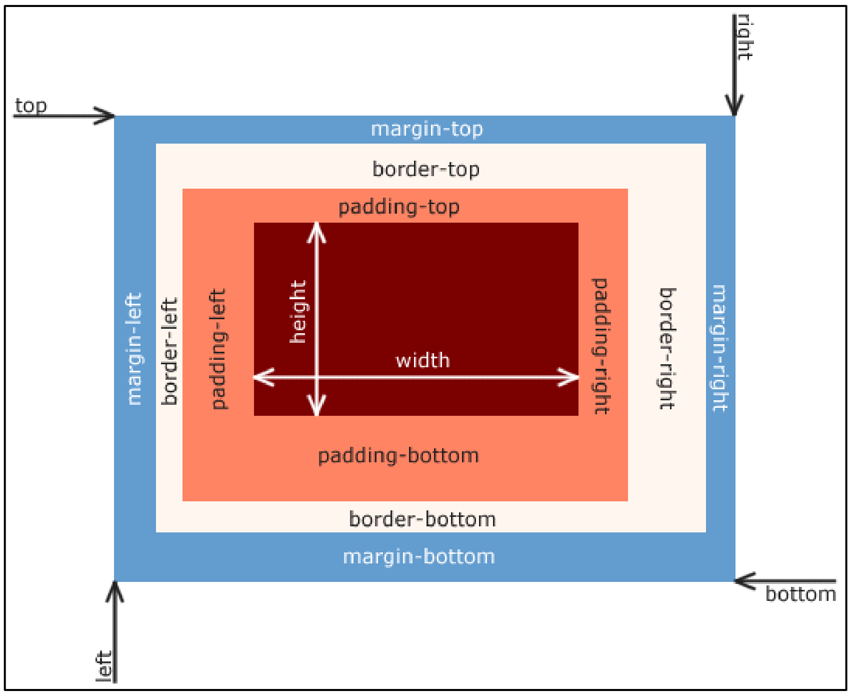
Ex: - show element at center of page. It may not work if element does not have width define.
margin: 30 auto;

Padding: - it separate inner content from element border.
padding: distance;
padding-left: distance;
padding-right: distance;
padding-top: distance;
padding-bottom: distance;

padding: top/bottom   left/right;
padding: top right bottom left;

CSS Box Model: -

In CSS, term “box model” is used for design and layout. The CSS box model is essentially a box that wraps around every HTML element. It consists of: content, padding, borders and margins. The image below illustrates the box model:



- **Content -** The content of the box, where text and media appear, it is center of box model and visible part of box model.
- **Padding -** It is blank space around the content. The padding is transparent
- **Border -** A border that goes around the padding and content
- **Margin -** An area outside the border and inside browser is margin. The margin is transparent

CSS Text/Fonts: -

Text: -

Color: - change text color
Ex: - color: aqua;

Letter spacing: - space between letters
Ex: - letter-spacing: 6px;

Word spacing: - space between words
Ex: - word-spacing: 10px;

Text indent: - leave space at starting of paragraph

Ex: - text-indent: 20px;

Text align: - align text in container
Ex: - text-align: center; //values center, right, left, justify, justify-all

Text transform: - transform text in different cases.
Ex: - text-transform:uppercase; // uppercase, lowercase, capitalize

Text decoration: - decorate text
<u>text-decoration-line</u>: overline; // also can add multiple lines
Ex: - text-decoration-line: underline overline; //none, underline, overline, line-through

<u>text-decoration-color</u>: aquamarine;
Ex: - text-decoration-color: aquamarine; // red, green,blue

text-decoration-style: dashed;
Ex: - text-decoration-style: dotted; // solid, dashed, dotted, double, wavy

text-decoration-thickness: 5px; //this property may mot render in some browsers
ex: - text-decoration-thickness: 5px; // 1em, 2rem, 1vh, 1vw

Font: -

Font family: - it set face of font.
Ex: - font-family: sans-serif; // serif, sans-serif, cursive, fantasy, monospace, arial, Brush Script MT
font-family: arial, Times; // also can set multiple font faces here if first assign face did not show, it called fallback font. Use **web safe font** for always display text

Web safe font: - there is possibility for some font did not render (appear) on all web browser in that case use web safe fonts for display text.

- 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)

Font size: - it set size of font
Ex: - font-size: 15px;
Font size values: -
xx-small, x-small, small, smaller, medium, large, larger, x-large, xx-large

Font size unit: -
%: - it takes percentage base of parent element size.

Ex: - font-size: 20%;
Em: - it takes font size of relative element as 1em.
Ex: - font-size: 2em;
font-size: 0.3em;
Rem: - it takes font size responsive to root element (html element).
Ex: - font-size: 1rem;
font-size: 1.5rem;

Font style: - it set styling of font.
Ex: - font-style: italic; // normal, oblique
Oblique: - similar to italic it used for some fonts that do not supports italic style.

Font weight: - it set weight (broadness) of font
Ex: - font-weight: bold; // bold, bolder, lighter, normal, 100 – 900 can use in font.

Font variant: - it set variant of font.
Ex: - font-variant: small-caps; // all-small-caps, small-caps, normal, unicase
Small-caps: -it set all lowercase letters in uppercase and these letters appear smaller size compare to regular uppercase letters.
Unicase: - it set uppercase letters font smaller as lowercase letters.

Line height: - it set space between two lines.
Ex: - line-height: 10px; //also can set as 1.2em, 1%, etc

Font shorthand: - font: style variant weight size/line-height family;
Ex: - font: italic normal lighter 20px/15px monospace;

External font (Google font): - set Google fonts in webpage.
- Go to web browser and search Google fonts.
- Click on link of Google font website.
- There will open multiple fonts with filter options for choose fonts.
- Click on font to choose, also can preview custom text in those fonts.
- Click on "get font" for add font in "view selected families".
- Click on "view selected families" to see selected fonts.
- There will show all selected font, click on "get embed code" copy link from "web" section.
- In "link" option, copy link and add it in "head" tag of html page.
- Also in "import" option, copy code of import and paste in html's head tag or in CSS file or style tag and remove style tags if those paste with import code.
- Copy code of those fonts class code and add them in to CSS file or style tag.
- Then apply that class to html element with class attribute.

CSS Images: -

Background image: - set image to background
Background color: - it set back ground color of element
background-color: color_name;

color_name: - set color with color name **ex: -** background-color: red;
rgb value: - set red, green and blue color value in range 0-255
Ex: - background-color: rgb(220,45,67);
rgba value: - set rgb value with alpha in range 0-1 **Ex: -** background-color: rgba(0,255,0,0.4);
hex code: -set hex code with '#' as three or six digit in range #000-#fff or #000000-#ffffff
ex: - background-color: #3f3; background-color: #3ffff3;

Color wheel: -

- search color wheel on Google and click to color.adobe.com website
- it will open page with color wheel, set and select color
- copy Hexcode of that color from pallet and paste it in color properties
- also in extract theme upload image from device and select color from that image

Opacity: - this property specifies the opacity/transparency of an element.
Ex: - ocpacity: 0.5; //value from 0 to 1.

Background image: - set url of image for background
background-image: url(image_name.extension);
Ex: - background-image: url("C:/Users/IANT/Downloads/redland.jpg");
background-image: url(res/earthimg.jpg);

Background size: - set size of background
background-size: value; cover, contain, width, width height
Ex: - background-size: contain; background-size: 100%; background-size: 300px 260px;

Background repeat: - set repeating of background
background-repeat: value; repeat, no-repeat, space, round, repeat-y, repeat-x.
Ex: - background-repeat: repeat;

Background position: - set position of background
background-position: value; //top, bottom, left, right, center, left top, left bottom, right top, right bottom, x-axis y-axis
Ex: - background-position: top left; background-position: 10px 100px;

Background origin: - it set origin position of background
background-origin: value; //border-box, padding-box, content-box
Ex: - background-origin: content-box;

Background clip: - it set painting area of background
background-clip: value; //content-box, padding-box, border-box, text(webkit)
Ex: - background-clip: content-box; -webkit-background-clip: text;

Multiple background images: - specify different images and properties with comma separation.
background-image: url(res/btrfly.webp), url(res/background.jpg);
background-size: 50px, 100% 100%;
background-repeat: repeat-y, no-repeat;
background-position: top right, top left;

Object: -

Object-fit: - this property is used to specify how an <img> or <video> should be resized to fit its container.
object-fit: fill; contain, cover, fill

Ex: -
```
img{
   width: 300px;
   height: 260px;
   object-fit: contain;
}
<div>
   <img src="res/background.jpg">
</div>
```

Object-position: - this property is used to specify how an <img> or <video> should be positioned within its container.
object-position: top; top, bottom, right, left, x-axis y-axis

Ex1: -
```
img{
   width: 300px;
   height: 260px;
   object-fit: contain;
   object-position: top;
}
<div>
   <img src="res/earthimg.jpg">
</div>
```

Ex2: -
```
img{
   width: 300px;
   height: 260px;
   object-fit: cover;
   object-position: left;
}
<div>
   <img src="res/earthimg.jpg">
</div>
```

Aspect ratio: - The aspect ratio of an element describes the proportional relationship between its width and its height.
Ex: - aspect-ratio: 3 / 2;
Aspect ratio: 4/3, 3/2, 1/1, 8/5

Example: -

```
.con{
   width: 100px;
   border: 1px solid black;
   aspect-ratio:3/4;
}

<div class="con">
   <img src="res/background.jpg" height="100%" width="100%">
</div>
```

Google icon: - set icons from Google

- search Google icons and click on link of font.google.com/icons
- select icon and it will details of that icon
- in web section copy code of font link and paste it in head tag
- then copy and paste CSS code in style tag
- then copy inserting icon code and paste it in html

Ex: - fill icon when hover on it

```
<!DOCTYPE html>
<html>
   <head>
      <title> google icon </title>
      <link rel="stylesheet"
href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD
@20..48,100..700,0..1,-50..200" />
      <style>
         .material-symbols-outlined {
            font-variation-settings:
            'FILL' 0,
            'wght' 400,
            'GRAD' 0,
            'opsz' 24
         }
         .material-symbols-outlined:hover{
            font-variation-settings:
            'FILL' 1;
            color: forestgreen;
            font-size: 1.55em;
         }
      </style>
   </head>
   <body>
      <span class="material-symbols-outlined"> home </span>
      <span class="material-symbols-outlined"> menu_book </span>
      <span class="material-symbols-outlined"> call </span>
```

```
    <span class="material-symbols-outlined"> settings </span>
  </body>
</html>
```

Border-image: -

border-image-source: url("res/fr1.jpg"); // take source image

border-image-slice: 64; //it divide the image into regions. Also use %
border-image-slice: top/bottom left/right;
border-image-slice: top left/right bottom;
border-image-slice: top right bottom left;
border-image-slice: fill 64; //use fill keyword for image background.
Slice image from corner of frame to where it design end

border-image-repeat: stretch; round, repeat, space //it is used for repeat border part.

border-image-outset: 2px; // set out distance from its border.

border-image-width: 20px; //set width of border image

Short hand: -
border-image: source slice/ width/ outset repeat;
Ex: - border-image: url(res/earthimg.jpg) 100/ 10px / 5px repeat;

CSS Combinators: - it combines other selectors in a way that gives them a useful relationship to each other and the location of content in the document.

Descendant selector (space): - The descendant selector matches all elements that are descendants of a specified element.

Selector1 selector2{ styling}
Ex: - div h1{   background-color: yellow; }

Child selector (>): - The child selector selects all elements that are the children of a specified element.

Selector1>selector2{ styling}
Ex: - div>h1{   background-color: red; }

Adjacent sibling selector (+): - The adjacent sibling selector is used to select an element that is directly after another specific element. "Adjacent" means "immediately following".

Selector1+selector2{ styling}
Ex: - div+div{   background-color: aqua; }

General sibling selector (~): - The general sibling selector selects all elements that are next siblings of a specified element.

Selector1~selector2{ styling}
Ex: - div~span{ background-color: green; }

CSS Pseudo-class: - it is a keyword added to a selector that specifies a special state of the selected element(s). A pseudo-class consists of a colon (:) followed by the pseudo-class name. A functional pseudo-class also contains a pair of parentheses to define the arguments

First-child: - this selector matches a specified element, if that is the first child of parent element.
element: first-child{ styling}
Ex: -
```
p:first-child{
   background-color: red;
}
```

Last-child: - this selector matches every element, if that is the last child of its parent.
element: last-child{ styling}
Ex: -
```
p:last-child{
   background-color: royalblue;
}
```

Only-child: - this selector matches every element, if that is the only child of its parent.
element: only-child{ styling}
Ex: -
```
p:only-child{
   background-color:brown;
}
```

Nth-child: - this selector matches every element, if that is the nth child of its parent.
element: nth-child(n){ styling} //n = number, keyword, formula
Ex: -
```
p:nth-child(2){
   background-color:forestgreen;
}
```

Nth-child values: -
(n): - 'n' is counter that start with 0, apply to every child.
Ex: - p:nth-child(n){ styling}

Even: - it is for child with even number
Ex: - p:nth-child(even){ styling}

Odd: - it is for child with odd number

Ex: - p:nth-child(odd){ styling}

(an): - ‘a’ is value to multiply counter ‘n’.
Ex: - p:nth-child(3n){ styling}

(an+b): - ‘b’ is offset value add to change effect with ‘a’ and counter ‘n’, also can subtract value
Ex: - p:nth-child(3n+1){ styling}
Ex: - p: nth-child(3n-1){ styling}

Nth-last-child: - this selector matches every element that is the nth child, regardless of type, of its parent, counting from the last child.
element:nth-last-child(n){ styling} //n = number, keyword, formula
Ex: -

```
p:nth-last-child(2){
   background-color:forestgreen;
}
```

First-of-type: - this selector matches every element that is the first child, of a particular type, of its parent.
Ex: - p:first-of-type{ styling}

Last-of-type: - this selector matches every element that is the last child, of a particular type, of its parent.
Ex: - p:last-of-type{ styling}

Nth-of-type: - this selector matches every element that is the nth child, of the same type (tag name), of its parent.
Ex: - p:nth-of-type(n){ styling} //n = number, keyword, formula

Nth-last-of-type: - this selector matches every element that is the nth child, of a particular type, of its parent, counting from the last child.
Ex: - p:nth-last-of-type(n){ styling} //n = number, keyword, formula

Only-of-type: - this selector matches every element that is the only child of its type, of its parent.
Ex: - h1:only-of-type{ styling}

Empty: - this selector matches every element that has no children (including text nodes).
Ex: - div:empty{ styling}

Not: - this selector matches every element that is NOT the specified element/selector.
Ex: - div :not(p){ styling}

Has: - this selector matches every element that has the specified element/selector
Ex: - div:has(h1){ styling}

CSS Links: - hyperlink can be style with CSS style properties, Also can use pseudo classes for style multiple states of hyperlink.

Link: - it is used for style normal and unvisited link.
Ex: -
```
a:link{
    color: green;
    text-decoration: none;
}
```

Visited: - it is used for style visited link
Ex: -
```
a:visited{
  color: red;
}
```

Hover: - it is used for style when cursor is on link.
Ex: -
```
a:hover{
  color: blue;
  text-decoration: underline;
}
```

Active: - it is used for style when user clicks on link.
Ex: -
```
a:active{
  color: yellow;
  text-decoration: underline;
}
```

When setting the style for several link states, there are some order rules:

- a:hover MUST come after a:link and a:visited
- a:active MUST come after a:hover

Active and hover pseudo classes can also use with other elements.

CSS Pseudo-element: - it is a keyword added to a selector that lets you style a specific part of the selected element(s). Double colons (::) are used for pseudo-elements. This distinguishes pseudo-elements from pseudo-classes that use single colon (:) in their notation.

First-letter: - this pseudo-element is used to add a special style to the first letter of a text.
Element::first-letter{ styling};
Ex: -
```
p::first-letter{
  background-color: red;
  color: white;
  text-transform: capitalize;
```

```
   font-size: 50px;
}
```

First-line: - this pseudo-element is used to add a special style to the first line of a text.
Element:: first-line{ styling}
Ex: -

```
p::first-line{
  background-color: yellow;
   color: red;
}
```

The following properties apply to the first-letter and first-line pseudo classes:

- font properties
- color properties
- background properties
- margin properties
- padding properties
- border properties
- text-decoration
- vertical-align (only if "float" is "none" for first-letter)
- text-transform
- line-height
- float
- clear

Selection: - this pseudo-element matches the portion of an element that is selected by a user. Background-color, color are properties can use with selection.
Element::selection{ styling}
Ex: -

```
p::selection{
   background-color: red;
   color: white;
}
```

Marker: - this pseudo-element selects the markers of list items.
li::marker{ styling}
Ex: -

```
li::marker{
   color: red;
   font-size: 20px;
   font-weight:bold;
}
```

Before: - this pseudo-element can be used to insert some content before the content of an element.
Element::before{ styling}

Ex: -

```
p::before{
   content:">";
   color: white;
   background-color: red;
}
```

Content: - enter string value to display before or after pseudo element

After: - this pseudo-element can be used to insert some content after the content of an element.
Element::after{ styling}
Ex: -

```
p::after{
   content: " >";
   color: white;
   background-color: red;
}
```

Content: - this property replaces content with a generated value. It can be used to define what is rendered inside an element or pseudo-element.

String value: - content: " > ";
None value: - content: none;
Image url: - content: url("res/bullets.jpg") / "alt"; //alt:- alternate text
Quote values: - content: open-qoute, close-qoute

CSS Tables: -

Border: - it display border for table.
Only table border: - border for only table.
Ex: -

```
table{
   border: 1px solid black;
}
```

All table borders: - border for table with td and th
Ex: -

```
table, th, td{
   border: 1px solid black;
}
```

Border-collapse: - it separate or collapse border of table. //values: collapse; separate
Ex: -

```
table{
   border-collapse: collapse;
}
```

Border-spacing: - it set space between borders.
Border-spacing: all-sides;
Border-spacing: left-right top-bottom; // Border-spacing: 5px 10px;
Ex: -
```
table{
   border-spacing: 10px;
}
```

Cell styling: - set vertical align, padding etc. of cells.

Vertical-align: - it set vertical align of content in table.
vertical-align: middle; //value: top, middle, bottom
Ex: -
```
td{
   vertical-align: bottom;
}
```

Cell-padding: - get padding to table cell.
padding: value;
Ex: -
```
th, td{
   border: 1px solid black;
   padding: 20px;
}
```

Caption side: - it set position of caption in table.
caption-side: value; //bottom, top
Ex: -
```
caption{
   caption-side: bottom;
}
```

Empty cells: - it display or hide empty cell
empty-cells: show; //hide
Ex: -
```
table{
   empty-cells: hide;
}
```

Table styling: - style table using CSS styling properties.

Table layout: - it set layout of table
table-layout: value; // auto, fixed – take only defined size.
Ex: -
```
table{
   table-layout:fixed;
}
```

Hoverable table: -
Ex: -
```
tr:hover{
   background-color: gray;
   color: white;
}
```

Clickable table: -
Ex: -
```
td:active{
   background-color: red;
   color: white;
}
```

Striped tables: -
Ex: -
```
tr:nth-child(even){
   background-color: gray;
   color: white;
}
```

CSS Lists: - there are multiple properties for style lists in css.

List style type: - it set type of bullet for display in list.
Ex: - list-style-type: disk;

Unordered list style: - disk, circle, square, none, disclosure-open, disclosure-closed,
Custom style: - list-style-type: "^ "; //"$ ", "@ ", "+ ", etc

Ordered list style: - decimal, decimal-leading-zero, none, upper-alpha, lower-alpha, upper-roman, lower-roman, arabic-indic, Bengali, lower-greek, armenian, bengali, cambodian, devanagari, ethiopic-numeric, georgian, gujarati, gurmukhi, kannada, lower-armenian, tamil, telugu, Malayalam, urdu

List style position: - it set position of marker relative to list items.
Ex: - list-style-position: outside; // inside, outside

List style image: - it set image for bulleting, (16*16)px is idle size or as font size.
Ex: - list-style-image: url(res/bullets.jpg);

List style shorthand: -
list-style: type position image url;
Ex: - list-style: circle inside url(res/bullets.jpg);

CSS Counters: - there are two functions counter and counters for increase counters

Counter: - it is dependent with other counter property
Counter-reset: - set counter reset for restart counter, define this in parent of element that will get counter
counter-reset: counter-name start-after-value;

Counter-increment: - increment counter value of given counter, define this in element that will get counter
counter-increment: counter-name increment-value;

counter(): - it set display counter values in content on pseudo element, also can apply list-style
counter(counter-name, list-style);

Ex: -
```
div{
   counter-reset: c1;
   height: 200px;
   width: 200px;
}
h2::before{
   counter-increment: c1;
   content: "CH." counter(c1, upper-roman) " ";
}
<div>
   <h2>good1</h2>
   <h2>good2</h2>
   <h2>good3</h2>
   <h2>good4</h2>
   <h2>good5</h2>
</div>
```

Sub counter: - it set counter
Ex: -
```
div{
   counter-reset: c1;
   height: 200px;
   width: 200px;
}
h2::before{
   counter-increment: c1;
   content: "CH." counter(c1) " ";
}
h2{
   counter-reset: c2;
}
h3::before{
   counter-increment: c2;
   content: "CH." counter(c1) ":" counter(c2,lower-alpha) " ";
}

<div>
    <h2>good1</h2>
    <h2>good2
        <h3>in1</h3>
        <h3>in2</h3>
        <h3>in3</h3>
     </h2>
     <h2>good3</h2>
     <h2>good4
        <h3>in1</h3>
        <h3>in2</h3>
        <h3>in3</h3>
    </h2>
    <h2>good5</h2>
</div>
```

Counters(): - it also need counter-reset and increment property it take name, sign between counter and sub-counter and list style of counter.
counters(counter_name, ".(sign)",list-style);
Ex: -

```
<style>
.counts{
    counter-reset: item;
    margin-left: 2em;
}
li{
    counter-increment: item;
}
li::marker{
    content: "items " counters(item, ".", lower-alpha) ": ";
}
</style>
<body>
     <ol class="counts">
         <li>good</li>
         <li>day
            <ol class="counts">
                <li>good</li>
                <li>god</li>
                <li>gd</li>
                <li>d</li>
            </ol>
         </li>
         <li>to</li>
         <li>all</li>
```

```
        <li>here</li>
    </ol>
</body>
```

CSS Shadows: - it is for apply shadow on elements or text.

Box shadow: -
box-shadow: vertical-shadow horizontal-shadow;
Ex: -
box-shadow: 5px -5px;

Color shadow: -
box-shadow: vertical-shadow horizontal-shadow color;
Ex: -
box-shadow: 5px 5px green;

Blur effect: -
box-shadow: vertical-shadow horizontal-shadow blur color;
Ex: -
box-shadow: 5px 1px 4px red;

Spread shadow: -
box-shadow: vertical-shadow horizontal-shadow blur spread-radius color;
Ex: -
box-shadow: 5px 1px 5px 10px red;

Transparent shadow: -
box-shadow: 5px 5px 5px rgba(0,0,0,0.3);

Inset shadow: - set shadow in element.
box-shadow: 5px 5px 5px red inset;

Multiple shadows: -
Box-shadow: shadow1 specification, shadow2 specification;
Ex: -
box-shadow: 5px 1px 5px red, -5px -1px 5px green;

Text shadow: -
text-shadow: horizontal-shadow vertical-shadow color;
Ex: -
text-shadow: 5px 5px red;

Blur shadow: -
text-shadow: vertical-shadow horizontal-shadow blur color;
Ex: -
text-shadow: 5px 5px 3px limegreen;

Multiple shadows: -
Text-shadow: shadow1 specification, shadow2 specification;
Ex: -
text-shadow: -5px 5px red,5px -5px green;
text-shadow: 1px 1px 2px black, 0 0 25px blue, 0 0 5px darkblue;

Border around text: -
text-shadow: -1px 0 black, 0 1px black, 1px 0 black, 0 -1px black;

CSS Gradients: - it is transition between two or more specific colors in background of element.

Linear gradient: - it display transition in straight line.
background-image: linear-gradient(red,yellow);

Flow(direction & diagonal) of transition: -
background-image: linear-gradient(to top,red, yellow); //bottom to top

to bottom : - from top to bottom flow
to top : - from bottom to top flow
to right : - from left to right flow
to left : - from right to left flow
to top right : - from bottom left to top right
to top left : - from bottom right to top left
to bottom right : - from top left to bottom right
to bottom left : - from top right to bottom left

In degree(angle): - it can also apply in minus angle.
background-image: linear-gradient(45deg,red, green);
background-image: linear-gradient(-45deg,red, green);

Multi colors: -
background-image: linear-gradient(to right, red, white, green, pink, yellow, maroon);

Customizing gradients: - add color stop points in gradients to display color in frequent length. It can define by percentage or length to defined direction
background-image: linear-gradient(red 0%, orange 20%, yellow 50%, green 75%, blue 100%);
background-image: linear-gradient(red 10px, green 50px, blue 70px);
background-image: linear-gradient(to right, red 10px, green 50px, blue 70px);

Also can add color start and stop value: -
background-image: linear-gradient(orange 0px 100px,white 100px 200px,green 200px 300px);

Repeating linear gradient: -
background-image: repeating-linear-gradient(red 10%,yellow 20%,green 30%);
background-image: repeating-linear-gradient(red 0% 10%,yellow 11% 20%,green 21% 30%);
background-image: repeating-linear-gradient(red 10px,yellow 20px,green 30px);

background-image: repeating-linear-gradient(to right,red 10px,yellow 20px,green 30px);

Colors: - it can define in multiple types.

Names: - it can set name of color.
Ex: - background-color: antiquewhite;

Rgb and rgba: - it can set in rgb with number from 0 to 255 range and alpha with 0.00 to 1 value.
Ex: -
background-color: rgb(123,50,65);
background-color: rgba(255, 0, 71, 0.2);

Hex code: - it can define with 3 or 6 digit hex code from 0 to f range with #, as #rrggbb or #rgb
Ex: -

```
background-color: #3f5;       //3 digit hex code
background-color: #fff257;    //6 digit hex code
```

transparent color: -
background-color: transparent;
background-color: rgba(0,0,0,0);

Transparent gradient: - background-image: linear-gradient(to right, red 20%, transparent);

Radial gradient: -
background-image: radial-gradient(red, yellow, green);

Gradient with spaced color: -
background-image: radial-gradient(red 30%, yellow 40%, green 50%);

Shape of radial gradient: -
Ellipse: - background-image: radial-gradient(ellipse,red 10%, yellow 40%, green 80%); //default
Circle: - background-image: radial-gradient(circle,red 10%, yellow 40%, green 80%);

Position of shape: -
background-image: radial-gradient(circle at top,red 30%, yellow 40%, green 50%);
circle at top
circle at bottom
circle at right
circle at left
circle at top right
circle at top left
circle at bottom right
circle at bottom left

Also can apply with ellipse shape
Ex: - ellipse at bottom right etc.

Position in distance: - it can give in % and px.
background-image: radial-gradient(circle at 10%, red 20%, yellow 30%, green 40%);//circle at X-axis
background-image: radial-gradient(circle at 10% 20%, red 20%, yellow 30%, green 40%);//circle at X-axis Y-axis
Also can apply with pixels as: -**circle at 10px 50px**

Size key words: - Determines the size of the gradient's ending shape.
Closest-side at: - it set size of gradient from location to closest side of gradient.
background-image: radial-gradient(closest-side at 50% 50%, red,yellow,green);

Farthest-side at: - it set size of gradient from location to farthest-side of gradient.
background-image: radial-gradient(farthest-side at 90% 90%, red,yellow,green);

Farthest-corner at: - it set size of gradient from location to farthest-corner of gradient.
background-image: radial-gradient(farthest-corner at 90% 50%, red,yellow,green);

Repeating radial gradient: -
background-image: repeating-radial-gradient(red,yellow 10%,green 20%);
background-image: repeating-radial-gradient(circle,red,yellow 10%,green 20%);
background-image: repeating-radial-gradient(circle at 20% 20%,red,yellow 5%,green 10%);
background-image: repeating-radial-gradient(closest-side at 80% 50%,red 0px 10px,transparent 10px 20px,green 20px 30px);

Multiple gradients: -
background-image: radial-gradient(circle at left, red 20% 30%, yellow 30% 40%, transparent 40%), linear-gradient(to right, blue 0% 50%,green 50% 70%,red 70%);

background-image: radial-gradient(circle at left, red 0% 20%, yellow 20% 25%, transparent 40%), radial-gradient(circle at right, blue 0% 20%,green 20% 25%,transparent 40%);

Conical Gradient: - it is a gradient with color transitions rotated around a center point.
background-image: conic-gradient(red,green,blue);

Gradient starts from specific angle: -
background-image: conic-gradient(from 45deg,red,green,blue);

Continue gradient: -
background-image: conic-gradient(red,green,yellow,blue,red);

Gradient with colors and degree: -
background-image: conic-gradient(red 110deg,green 200deg,yellow);

Gradient center position: -
background-image: conic-gradient(at 30% 50%,red 110deg,green 200deg,yellow,red);

Gradient with color start and stop: -

background-image: conic-gradient(red 0deg 90deg, yellow 90deg 180deg, green 180deg 270deg, blue 270deg);

background-image: conic-gradient(from -45deg,red 0deg 90deg, yellow 90deg 180deg, green 180deg 270deg, blue 270deg);

Repeating conic gradient: -
background-image: repeating-conic-gradient(red 5deg,yellow 10deg,red 15deg);
background-image: repeating-conic-gradient(red 10deg 20deg,yellow 20deg 30deg);
transparent: -
background-image: repeating-conic-gradient(rgba(0,0,0,0) 10deg 20deg,rgba(0,0,0,0.5) 20deg 30deg);
background-color: red;

conic gradient with background size: -
background-image: repeating-conic-gradient(black 0deg 90deg, yellow 90deg 180deg);
background-size: 50px 50px;

background-image: repeating-conic-gradient(red 5deg,yellow 10deg,red 15deg);
background-size: 60px 60px;

background-image: repeating-conic-gradient(red 5deg,yellow 10deg);
background-size: 60px 60px;

CSS 2D Transforms: - this property allow to move, rotate, scale and skew elements.

Transition: - it allows changing property values smoothly, over a given duration. The transition effect will start when the specified CSS property changes value.
Ex: -
```
.box{
   width: 100px;
   height: 90px;
   background-color: aqua;
   transition: width 2s;
}
.box:hover{
   width: 200px;
}
```

Multiple values: -
```
.box{
   transition: width 2s, height 2000ms; //time can set in seconds and miliseconds
}
```

Transition delay: - this property specifies when the transition effect will start. Its value is defined in seconds (s) or milliseconds (ms).

Transition duration: - This property specifies how many seconds (s) or milliseconds (ms) a transition effect takes to complete.
Transition property: - 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).
Tip: A transition effect could typically occur when a user hover over an element.
Note: Always specify the transition-duration property, otherwise the duration is 0, and the transition will have no effect.
Transition timing function: - this property specifies the speed curve of the transition effect.
The transition-timing-function property can have the following values:

- Ease: - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
- Linear: - specifies a transition effect with the same speed from start to end
- ease-in: - specifies a transition effect with a slow start
- ease-out: - specifies a transition effect with a slow end
- ease-in-out: - specifies a transition effect with a slow start and end
- step-end: - specifies a transition effect with duration end
- step-start: - specifies a transition effect with duration start
- cubic-bezier: - specifies a transform effect with different time at x1 and x2 points from 0 to 1 as slow to fast, y1, y2 value apply for progression and bounce effect if applicable, its value can more than 1.
  Ex: - cubic-bezier(x1, y1, x2, y2)
  cubic-bezier(0.1, 0.8, 0.1, 2);
- steps: - specifies a transiction effect with provided steps
  Ex: - steps(5);
  Jumpterm: - it is optional value for specifies the point at which change of value occurs.
  - jump-start: - first jump happens when the transition begins
  - jump-end: - the last jump happens when the transition ends
  - jump-both: - Includes pauses at both the 0% and 100% marks, effectively adding a step during the transition time

  Ex: - steps(10,jump-start)

Example: -

```
.box{
   width: 100px;
   height: 90px;
   background-color: aqua;
   transition-delay: 100ms;
   transition-duration: 2s;
   transition-property: width, height;
   transition-timing-function: linear;
}
.box:hover{
   width: 200px;
   height: 120px;
}
```

Translate: - it move element on x and y axis.

translateX: - move element on x-axis
transform: translateX(distance);
Ex: - transform: translateX(20px);

translateY: - move element on y-axis
transform: translateY(distance);
Ex: - transform: translateY(20px);

translate(x, y): - move element on both x and y axis
transform: translate(x-axis, y-axis)
Ex: - transform: translate(10px,10px);

Rotate: - it rotate element in given degree as clockwise or anti-clockwise.
transform: rotate(angle); // rotate element clock wise
Ex: - transform: rotate(45deg); -45deg, 0.2turn

Scale: - it increase or decrease size of element in given scale
scaleX: - scale on x-axis of element
transform: scaleX(scale);
Ex: - transform: scaleX(0.5);

scaleY: - scale on y-axis of element
transform: scaleY(scale);
Ex: - transform: scaleY(0.5);

scale(x, y): - scale on both axis of element
transform: scale(x, y);
Ex: - transform: scale(1.2,0.9);

Skew: - it skew element on given axis.
skewX: - skew element on x-axis.
transform: skewX(angle);
Ex: - transform: skewX(85deg);

skewY: - skew element on y-axis.
transform: skewY(angle);
Ex: - transform: skewY(85deg);

skew(x, y): - skew element on both axis.
transform: skew(x, y);
Ex: - transform: skew(34deg,40deg);

Multiple transform: - also can set multiple values
Ex: - transform: rotate(30deg) skew(30deg,1deg) scale(0.87);

Matrix: - it combines scale and skew translate method.

transform: matrix(scaleX(), skewX(), skewY(), scaleY(), translateX(), traslateY());
Ex: - transform: matrix(1, 2, 1, 1, 80, 80); //its values effect different than scale, skew and translate method.

Transform origin: - it set origin position of element transform.
transform-origin: position;
Ex: - transform-origin: left; // left, right, top, bottom, center
Ex: - transform-origin: left top; // left top, right top, left bottom, right bottom
Ex: - transform-origin: 30px 40px; // 20% 30%

CSS 3D Transform: -

Rotate: - rotate element on x, y and z axis.
rotateX: - rotate element on X-axis.
Ex: - transform: rotateX(120deg);

rotateY: - rotate element on Y-axis.
Ex: - transform: rotateY(120deg);

rotateZ: - rotate element on Z-axis.
Ex: - transform: rotateZ(120deg);

rotate3d: - it rotates an element around a fixed axis in 3D space.
transform: rotate3d(x, y, z, angle); //x, y and z coordinates of vector denoting axis of rotation. Angle represent degree of rotation.
Ex: - transform: rotate3d(1,2,0,20deg); //it take negative values also

Perspective: - it set distance between z=0 plane and the user for give a 3D-positioned element some perspective. It set in parent element of child element that set transform.
perspective: length;
Ex: - perspective: 1000px; // none, 23rem, 5.2cm
Ex: -
```
.parent{
    Perspective: 1000px;
}
.child{
    transform: rotateX(50deg);
}
```

Translate: - it move element in given axis.
translateZ: - it move element on z axis.
Ex: - transform: translateZ(30px);
Ex: - transform: rotateY(60deg) translateZ(300px);

translate3d: - it move element in x, y and z axis.
Ex: - transform: translate3d(30px,40px,10px);
Ex: - transform: rotateY(60deg) translate3d(10px, 10px, -100px);

Scale: - it increase or decrease size of element.
scaleZ: - it scale of Z axis.
Ex: - transform: scaleZ(3);
Ex: - transform: scaleZ(3) rotateY(60deg);

scale3d: - it scale element in x, y and z axis.
Ex: - transform: scale3d(1,1,3);
Ex: - transform:scale3d(1,1,3) rotateY(60deg);

Transform style: - it set property for children of element are positioned in the 3D space or flat plane.
transform-style: - preserve-3d; // transform in 3d space.
transform-style: - flat; // transform in flat plane.

Ex: -
```
.parent{
     transform-style: - preserve-3d;
}
.child{
    transform: rotateY(-60deg);
}
```

Perspective origin: - it set the position at which viewer is looking. This property set to parent element of child element that will get transform in 3d.

perspective-orgin: x-position y-position;
Ex: - perspective-orgin: 120% 300%;
Ex: - perspective-orgin: center; //top right, top left, bottom right, bottom left, left, right, top, bottom

CSS Display: - it specifies the display behavior of an element.

display: value; // block, inline, none, initial

Block: - it takes all allocated size to element, and these elements begin in new line.
Inline: - it takes only space that it need for content inside it, and these elements begin anywhere in line.
None: - it hides element and remove it from webpage.
Initial: - it set elements in its initial behavior.

Visibility: - it is property for set visibility of elements.
visibility: value; // hidden, collapse, visible

Hidden: - it hides elements, but reserve space of hidden elements on web page

Collapse: - it hides elements, but does not reserve it's space, work with table, tbody, thead and tfoot elements only, with other elements it behave like 'hidden' value.
Visible: - it makes elements visible, if it is hidden or collapse.

CSS Inline-block: - it is value of display property for show block elements in line.

Inline-block: - it takes all allocated size to element but begin anywhere in line.
Ex: - display: inline-block;

CSS Position: - it set position of elements on webpage

Position: - value; // static, absolute, fixed, relative, static, sticky
Ex: -
position: absolute;

static: - it is default position of every html elements, it set element according to normal flow of page.
relative: - it is relative position to its normal position.
fixed: - it is relative position to viewport.
absolute: - it is relative to nearest positioned ancestor, if it has not positioned ancestor it will relative to body element.
sticky: - it is toggle between relative and fixed, depend on scroll position, it stick to parent element on viewport.

Set position: - it set position of element from sides of parent element or view port when, element in not static position.
Left: - left: length;
Ex: - left: 200px;
Right: - right: length;
Ex: - right: 200px;
Top: - top: length;
Ex: - top: 200px;
Bottom: - bottom: length;
Ex: - bottom: 200px;

Z-index: - it set the stack order of an element. When elements are positioned, they can overlap other elements. This property work when position is not static.
z-index: value; // -1,1,2,-10
Ex: - z-index: -2;

CSS Navigation Bar: - it is made with ul and li tag and with webpage link and style with css properties. It is for surfing through other web-page.

Example: -
<!DOCTYPE html>

```
<html>
   <head>
      <meta http-equiv="refresh" content="2">
      <title> nav bar </title>
      <style>
         *{
            margin: 0;
            padding: 0;
         }
         div{
            padding-left: 100px;
            background-color: azure;
            float: left;
            width: 100%;
            position: sticky;
            top: 0px;
         }
         ul{
            list-style: none;
            float: left;
         }
         li{
            background-color:#058015;
            float: left;
            width: 100px;
            height: 50px;
            line-height: 50px;
            text-align: center;
            margin-right: 5px;
            font-size: 25px;
            border-radius: 8px;
         }
         a{
            text-decoration: none;
            color: white;
            text-align: center;
            height: 100%;
            width: 100%;
         }
         li:hover{
            background-color:forestgreen;
         }
         a:hover{
            float: left;
            background-color: azure;
            color: #058015;
         }
         a:active{
            color: royalblue;
         }
      </style>
   </head>
   <body>
      <div>
         <nav>
            <ul>
               <li><a href="#">home</a></li>
               <li><a href="#">courses</a></li>
               <li><a href="#">gallary</a></li>
               <li><a href="#">contact</a></li>
               <li><a href="#">about</a></li>
            </ul>
         </nav>
      </div>
   </body>
</html>
```

CSS Dropdowns: - it is concept when hover on element it will show more data about it.

Image dropdown: -

```
<html>
   <head>
      <title> Dropdowns </title>
      <style>
         *{
            margin: 0;
            padding: 0;
         }
         .img{
            background-image: url(res/earthimg.jpg);
            background-size: 100%;
            width: 100px;
            height: 56px;
         }
         img{
            width: 100%;
            height: 100%;
         }
         .dropdown{
            width: 400px;
            height: 224px;
            display: none;
         }
         .img,.dropdown{
                margin-left: 10px;
            }
            figcaption{
                font-size: 30px;
                text-align: center;
                padding: 10px;
            }
            .dropdown, figcaption{
                box-shadow: 5px 5px 5px 0px rgba(0,0,0,0.2);
            }
            .img:hover+.dropdown{
                display: block;
            }
        </style>
    </head>
    <body>
        <div class="img"></div>
            <figure class="dropdown">
                <img src="res/earthimg.jpg" height="100%" width="100%">
                <figcaption>Earth Image</figcaption>
            </figure>
    </body>
</html>
```

Menu dropdown: -

```
<!DOCTYPE html>
<html>
    <head>
        <title> Dropdown </title>
        <style>
            *{
                margin: 0;
                padding: 0;
            }
            .head{
                background-color: aliceblue;
                height: 70px;
            }
            ul{
                list-style: none;
            }
            li{
                background-color: black;
                float: left;
                width: 120px;
                height: 70px;
                margin-left: 10px;
                text-align: center;
                line-height: 70px;
                font-size: 32px;
                font-weight: bold;
                font-family: sans-serif;
                position: relative;
            }
            a{
                text-decoration: none;
                color: ghostwhite;
                width: 100%;
                height: 100%;
            }
            .dropdown{
                position: absolute;
                top:70px;
                right:0px;
                display: none;
            }
            a:hover{
                float: left;
                background-color: wheat;
                color: white;
            }
            li:hover>ul{
                display: block;
            }
        </style>
    </head>
    <body>
        <div class="head">
            <nav>
                <ul>
                    <li><a href="#">Home</a></li>
                    <li><a href="#">course</a>
                        <ul class="dropdown">
                            <li><a href="#">C</a></li>
                            <li><a href="#">C++</a></li>
                            <li><a href="#">C#</a></li>
                            <li><a href="#">Python</a></li>
                        </ul>
                    </li>
                    <li><a href="#">about</a></li>
                    <li><a href="#">contact</a></li>
                </ul>
            </nav>
        </div>
```

```
   </body>
</html>
```

Multilevel dropdown: -

```
<!DOCTYPE html>
<html>
   <head>
      <title> Dropdown </title>
      <style>
         *{
            margin: 0;
            padding: 0;
         }
         .head{
            background-color: aliceblue;
            height: 50px;
         }
         ul{
            list-style: none;
         }
         li{
            background-color: black;
            float: left;
            width: 120px;
            height: 50px;
            text-align: center;
            line-height: 50px;
            font-size: 22px;
            font-weight: bold;
            font-family: sans-serif;
            position: relative;
         }
         a{
            text-decoration: none;
            color: ghostwhite;
            width: 100%;
            height: 100%;
         }
         .dropdown{
            position: absolute;
            top:50px;
            right:0px;
            display: none;
         }
         .dropdown2{
            position: absolute;
            top:0px;
            left:120px;
            display: none;
        }
        .dropdown3{
            position: absolute;
            top:0px;
            left:120px;
            display: none;
        }
        a:hover{
            float: left;
            background-color: wheat;
            color: black;
        }
        li:hover{
            outline: 1px solid white;
            outline-offset: -1px;
        }
        li:hover>.show{
            display: block;
        }
    </style>
</head>
<body>

    <div class="head">
        <nav>
            <ul>
                <li><a href="#">Home</a></li>
                <li class="hover"><a href="#">course</a>
                    <ul class="dropdown show">
                        <li><a href="#">C</a></li>
                        <li><a href="#">C++</a></li>
                        <li><a href="#">C#</a></li>
                        <li><a href="#">Python</a></li>
                        <li class="hover"><a href="#">web dev</a>
                            <ul class="dropdown2 show">
                                <li><a href="#">HTML</a></li>
                                <li><a href="#">CSS</a></li>
                                <li><a href="#">JavaScript</a>
                                    <ul class="dropdown3 show">
                                        <li><a href="#">Vanila</a></li>
                                        <li><a href="#">Angular</a></li>
                                        <li><a href="#">React</a></li>
                                        <li><a href="#">Node</a></li>
                                    </ul>
                                </li>
                </ul>
              </li>
            </ul>
          </li>
          <li><a href="#">about</a></li>
          <li><a href="#">contact</a></li>
        </ul>
      </nav>
    </div>
  </body>
</html>
```

CSS Max-width: - it use for set maximum size to elements.

Width: - set maximum and minimum width of elements
Min width: - set minimum width of elements.
Ex: - min-width: 300px;

Max width: - set maximum width of elements.
Ex: - max-width: 300px;

Height: - set maximum and minimum height of elements
Min height: - set minimum height of elements.
Ex: - min- height: 300px;

Max height: - set maximum height of elements.
Ex: - max- height: 300px;

Overflow: - it is property to handle data when it overflows from elements.
Overflow: - set overflow property on both axis.
overflow: value; // auto, visible, hidden, scroll
Ex: -
Auto: - it set scroll bars if required
Visible: - it shows overflow data
Hidden: - it hides overflow data
Scroll: - it set scroll bar to elements

Overflow-x: - it set overflow on x-axis.
Ex: - overflow-x: hidden; // visible, auto, hidden, scroll

Overflow-x: - it set overflow on y-axis.
Ex: - overflow-y: scroll; // visible, auto, hidden, scroll

Background attachment: - it specifies whether the background image should scroll or be fixed local property is get used with overflow
background-attachment: value; //fixed, scroll, local

Ex: - background-attachment: fixed;
Ex: - // local is used for fixed background in overflow

```
div{
   width: 200px;
   height: 200px;
   border: 1px solid black;
   background-image: url(img1.jpg);
   overflow-y:scroll;
   background-attachment: scroll;
}
```

Html: -

```
<div>good … … … bye</div>
```

Text-overflow: - The text-overflow property specifies how overflowed content that is not displayed should be signaled to the user.
properties are required for text-overflow:
white-space: nowrap;
overflow: hidden;

text-overflow: value; clip ellipsis(…)
Ex: -

```
p{
   width: 200px;
   height: 20px;
   overflow: hidden;
   white-space: nowrap;
   text-overflow: ellipsis;
}

<p title="https://www.domainname123456.com//goodnight">
   https://www.domainname123456.com//goodnight
</p>
```

Title: - this attribute specifies extra information about an element.

Viewport max and min: -

vmax: - it is relative to 1% of viewport larger dimension as 1vmax.
Ex: -
max-width: 90vmax;
max-height: 90vmax;

vmin: - it is relative to 1% of viewport smaller dimension as 1vmin.
Ex: -
min-width: 50vmin;
min-height: 50vmin;

CSS Outline: -

it is part of border element. It does not effect on element size.

Outline style: - set style of outline

Outline style: - it set style of outline.
outline-style: value;// dashed, dotted, double, groove, inset, outset, ridge.

Ex: -
```
div{
  outline-style: solid;
}
```

Outline width: - set width of outline
outline-width: size;// thin, thick, medium, px, %,etc

Ex: -
```
div{
  outline-width: 5px;
}
```

Outline color: - set color of outline
outline-color: color_name; //rgba, hexcode

Ex: -
```
div{
  outline-color: red;
}
```

Outline shorthand: - set outline properties in one line
outline: width style color;

Ex: -
```
div{
  outline: 5px solid yellow;
}
```

CSS Align: -

Align-content: - it aligns content in element vertically.
align-content: value; //center, end, start

Ex: - align-content: center;

It is align content property with block element it may supports on some browser

Center align element: -
```
width: 50%;
padding: 10px;
border: 1px solid gold;
margin: auto;
```

Centers align text: -
```
text-align: center;
```

Vertically align text: -

Using line-height: -

```
width: 300px;
height: 200px;
border: 1px solid gold;
line-height: 200px;
```

Multiple lines vertical align: -

```
.con{
   width: 300px;
   height: 200px;
   border: 1px solid gold;
   line-height: 200px;
}
.con span{
   line-height: 1.5;
   display: inline-block;
   vertical-align: middle;
}
```

Note: - it may not applicable in new browsers, instead use align-content: center;

Center using transform and position: -

```
.con{
   width: 300px;
   height: 200px;
   border: 1px solid gold;
   position: relative;
}
.con span{
   position: absolute;
   top: 50%;
   left: 50%;
   transform: translate(-50%,-50%);
   display: inline-block;
   background-color: red;
}
```

Images align: - it can align images at vertical position with text

Ex: -

```
div{
   margin: 50px;
   height: 50px;
   border: 1px solid black;
   line-height: 30px;
}
```

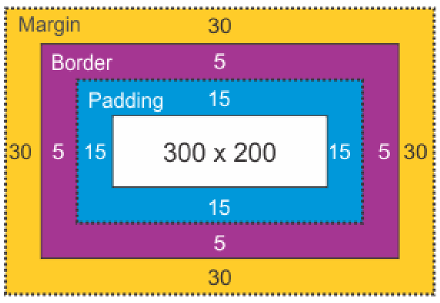
```
img{
   width: 20px;
   vertical-align: middle;
}
<div>    Earth <img src="res/earthimg.jpg">Image </div>
```

Vertical align values: - baseline, bottom, middle, sub, super, text-bottom, text-top, top

CSS Box Sizing: -

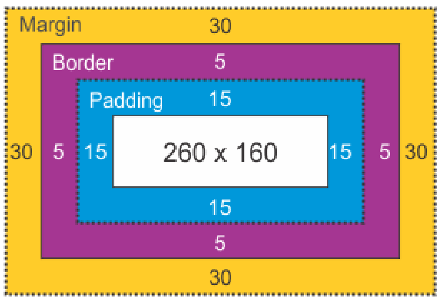
this property allows us to include the padding and border in an element's total width and height.

Box Model is content-box



```
div{
      width: 300px;
      height: 200px;
      padding: 15px;
      border: 5px solid grey;
      margin: 30px;
      -moz-box-sizing: content-box;
      -webkit-box-sizing: content-box;
      box-sizing: content-box;
    }
```

Box Model is border-box



```
div{
      width: 300px;
      height: 200px;
      padding: 15px;
      border: 5px solid grey;
      margin: 30px;
      -moz-box-sizing: border-box;
      -webkit-box-sizing: border-box;
      box-sizing: border-box;
    }
```

Ex: - box-sizing: content-box; // border-box

Border-box: - this value includes padding and border size in height and width.

Example: -

```
div{
   width: 200px;
   height: 200px;
   padding: 20px;
   border: 2px solid black;
}
.box1{
   box-sizing: content-box;
}
.box2{
   box-sizing: border-box;
}

<div class="box1">Hello world</div>
<div class="box2">Hello world</div>
```

CSS Forms: -

it can style form and input elements

Style particular input: - select particular using attribute tag
Ex: - select text field only with value in square bracket

```
input[type="text"]{
   background-color: red;
}

<form>
   <input type="text">
</form>
```

Form pseudo classes: -

Focus: - it apply when input in focus
Ex: -

```
input:focus{
   background-color: aliceblue;
}
<input type="text">
```

Checked: - it apply when checkbox or radio button selected
Ex: -

```
Input[type="radio"]:checked{
   box-shadow: 0px 0px 0px 2px red;
}
<input type="radio">
```

Enabled: - it apply style to enabled input
Ex: -

```
input:enabled{
   background-color: yellow;
}
```

Disabled: - it apply style to disabled input
Ex: -

```
input:disabled{
   background-color: green;
}
<input type="text" disabled>
```

Required: - it apply style to required input
Ex: -

```
input:required{
   background-color: aqua;
}
<input type="text" required>
```

Optional: - it apply style to optional input
Ex: -

```
input:optional{
   background-color: purple;
}
```

In range: - it apply style to input when value is in range
Ex: -

```
input:in-range{
   background-color: greenyellow;
}
<input type="number" min="1" max="100">
```

Out of range: - it apply style to input when value is in out of range
Ex: -

```
input:out-of-range{
   background-color: red;
}
<input type="number" min="1" max="100">
```

Read write: - it apply style to read-write input

```
input:read-write{
   background-color: springgreen;
}
```

Read only: - it apply style to read only input
Ex: -

```
input:read-only{
   background-color: gray;
}
<input type="text" readonly>
```

Valid: - it apply style to valid input
Ex: -

```
input:valid{
   background-color: green;
}
<input type="email">
```

Invalid: - it apply style to invalid input
Ex: -

```
input:invalid{
   background-color: red;
}
<input type="email">
```

Default: - it apply style to default input work with checkbox and radio
Ex: -

```
input:default{
   box-shadow: 0px 0px 0px 2px red;
}
<input type="checkbox" checked>
<input type="radio" checked>
```

Accent color: - this property set color of input. Work with checkbox, radio, range input.

Ex: -

```
input[type="checkbox"] {
   accent-color: green;
}
```

Appearance: - this property is used to display UI elements with platform-specific styling, based on the operating system's theme, especially for checkbox, range and radio button

appearance: none; button, checkbox, radio, range

Ex: - -webkit-appearance: none; -moz-appearance: none;

Custom checkbox: -

```
input[type="checkbox"]{
   -webkit-appearance: none;
   width: 1em;
   height: 1em;
   background-color: red;
   border: 1px solid blue;
   border-radius: 3px;
}
input[type="checkbox"]:checked{
   background-color: green;
}

<form>
   <input type="checkbox">
</form>
```

Custom range: -

```
input[type="range"]{
   -webkit-appearance: none;
   width: 100px;
   height: 1em;
   background-color: red;
   border: 1px solid blue;
   border-radius: 3px;
   scroll-behavior: smooth;
}
input[type="range"]:hover{
   background-color: greenyellow;
}
input[type="range"]:active{
  background-color: green;
}

<input type="range">
```

CSS Tooltips: - it is often used to specify extra information about something when the user moves the mouse pointer over an element

```
<!DOCTYPE html>
<html>
  <head>
    <title> tooltips </title>
    <style>
      .center{
        position: absolute;
        top: 50%;
        left: 50%;
        transform: translate(-50%,-50%);
      }
      .contain{
        width: 400px;
        height: 400px;
        background-color: aqua;
        border: 1px solid black;
      }
      .design{
        position: absolute;
        z-index: 2;
        background-color: black;
        padding: 5px;
        border-radius: 5px;
        color: white;
        display: none;
      }
      .arrow::after{
        content: "";
        position: absolute;
        border-width: 7px;
        border-style: solid;
      }
      .arrow-pos-bottom::after{
        top: 99%;
        left: 50%;
        border-color: black transparent transparent transparent;
      }
```

```
        .arrow-pos-top::after{
          bottom: 96.4%;
          left: 50%;
          border-color: transparent transparent black transparent;
        }
        .arrow-pos-left::after{
          top: 25%;
          left: 99%;
          border-color: transparent transparent transparent black;
        }
        .arrow-pos-right::after{
          top: 25%;
          right: 99%;
          border-color: transparent black transparent transparent;
        }
        .tooltip1{
          top:-40px;
          left: 30px;
        }
        .tooltip2{
          top: 70px;
          left: 100px;
          width: 60%;
        }
        input:hover+.design{
          display: block;
        }
        span{
          text-align: center;
          width: 100%;
          float: left;
          padding-top: 50px;
          font-size: 30px;
        }
      </style>
    </head>
    <body>
      <div class="contain center">
        <span>LOG IN</span>
        <form class="center">
          <input type="text">
          <div class="design tooltip1 arrow arrow-pos-bottom">
            Enter UserID
          </div><br><br>
          <input type="password">
          <div class="design tooltip2 arrow arrow-pos-top">
            Enter Password
            </div><br><br>
            <input type="submit">
          </form>
        </div>
    </body>
</html>
```

ANIMATING WITH CSS: - Create animations in CSS with animation’s properties.

Animation: - it lets an element gradually change from one style to another.
Animation name: - set name of animation that will apply to keyframes name.
Ex: - animation-name: anim;

Animation duration: - set time for play animation. It can set in seconds and milliseconds
Ex: - animation-duration: 5s; //1000ms;

Animation delay: - set time for animation delay time.
Ex: - animation-delay: 1s;

Animation fill mode: - this property specifies a style for the target element when the animation is not playing

- **none -** Default value. Animation will not apply any styles to the element before or after it is executing
- **forwards -** The element will retain the style values that is set by the last keyframe
- **backwards -** The element will get the style values that is set by the first keyframe
- **both -** The animation will follow the rules for both forwards and backwards, extending the animation properties in both directions

Ex: - animation-fill-mod: forwards; //none, forwards, backwards, both

Animation iteration count: - it set how many time will animation play
Ex: - animation-iteration-count: 2; //count, infinite

Key frames: - it set animation movement of element when animation plays.
@keyframes name_1{ … … }

From-To: - it set element movement from start to end in animation.
Ex: -

```
@keyframes anim1{
  from{… … …}
  to{… … …}
}
```

Building blocks of CSS Animation: -

Block of animation: - it set element movement in percentages of given time duration animation.

```
@keyframes name1{
        0%{…}
        50%{…}
        100%{…}
}
```
Ex: -
```
@keyframes anim1{
   0%{
      left: 0%;
   }
   50%{
      left 50%;
   }
   100%{
      left: 90%;
   }
}
```

Animation timing function: - this property specifies the speed curve of the animation.

- **ease -** Specifies an animation with a slow start, then fast, then end slowly (this is default)
- **linear -** Specifies an animation with the same speed from start to end
- **ease-in -** Specifies an animation with a slow start
- **ease-out -** Specifies an animation with a slow end
- **ease-in-out -** Specifies an animation with a slow start and end
- **cubic-bezier(x1,y1,x2,y2) –** specifies a transform effect with different time at x1 and x2 points from 0 to 1 as slow to fast, y1, y2 value apply for progression and bounce effect if applicable, its value can more than 1.
- **steps -** specifies an animation with provided steps

Ex: - animation-timing-function: linear;
Ex: - animation-timing-function: cubic-bezier(1,.02,.89,1.5);
Ex: - animation- timing-function: steps(5);

Cubic Bezier Website: - visits this site for create cubic Bezier function values.
Website: - https://cubic-bezier.com

Animation-direction: - this property specifies whether an animation should be played forwards, backwards or in alternate cycles.

- **normal -** The animation is played as normal (forwards). This is default
- **reverse -** The animation is played in reverse direction (backwards)
- **alternate -** The animation is played forwards first, then backwards
- **alternate-reverse -** The animation is played backwards first, then forwards

Ex: - animation-direction: reverse; //normal, reverse, alternate, alternate-reverse

Animation play state: - this property specifies animation is running or paused.
Ex: - animation-play-state: running; //paused

Animation shorthand: -
animation: name duration timing-function delay iteration-count direction;
Ex: - animation: anim1 5s ease-out 1s infinite alternate;

CSS Buttons: - customize styling for buttons

Button style: -

Example: -

```
.btn{
   background-color: red;
   width: 100px;
   padding: 10px;
   border: 5px solid maroon;
   font-size: 20px;
   font-weight: bolder;
   color: white;
   border-radius: 10px;
   position: relative;
   transition-duration: 0.4s;
   transition-property: all;
}
.btn:hover{
   border-color: darkgreen;
   background-image: linear-gradient(to right,greenyellow,darkgreen);
   color: white;
   font-weight: bolder;
   box-shadow: 2px 2px 4px rgba(0,0,0,0.7), -2px -2px 4px rgba(0,0,0,0.7);
   text-shadow: 1px 1px 1px black;
}

<form>
   <input type="submit" class="btn">
</form>
```

Link button: -

Example: -

```
.link{
   background-color: coral;
   text-decoration: none;
   color: black;
   display: inline-block;
   width: 75px;
   border: 2px solid green;
   font-size: 20px;
   text-align: center;
   padding: 10px;
}
.link>span{
   position: relative;
   transition: all 0.5s;
}
.link>span::after{
   content:" \00bb";
   position: absolute;
   opacity: 0;
}
.link:hover>span{
   padding-right: 10px;
}
.link:hover>span::after{
   right: -5px;
   opacity: 1;
}

<a href="#" class="link"><span>good</span></a>
```

Toggle button: -

Example: -

```
#ct[type="checkbox"]{
   display: none;
}
.toggle{
   border: 1px solid gold;
   display: inline-block;
   position: relative;
   width: 40px;
   height: 10px;
   background-color: white;
   border-radius: 10px;
   transition: 0.5s;
}
.toggle:after{
   content: "";
   background-color: yellow;
   width: 8px;
   height: 8px;
   position: absolute;
   top: 1px;
   left: 1px;
   border-radius: 8px;
   transition: 0.3s;
}
```

```
#ct:checked+.toggle{
   background-color: black;
   border: 1px solid silver;
}
#ct:checked+.toggle:after{
   background-color: gray;
   left: 31px;
}
.toggle:active::after{
   width: 9px;
}

<form>
   <input type="checkbox" id="ct">
   <label for="ct" class="toggle"></label>
</form>
```

CSS Pagination: - create paging index style in CSS.

Example: -

```
.pages{
   text-align: center;
   width: 100%;
   height: 100px;
   display: inline-block;
}
.pages a{
   border: 2px solid gainsboro;
   color: black;
   display: inline-block;
   padding: 8px 16px;
   text-decoration: none;
   margin: 0px 5px;
   border-radius: 5px;
   transition: background-color 400ms;
}
a:hover{
   background-color: lavender;
}
a.active{
   background-color: greenyellow;
   color: white;
   border-radius: 5px;
}

<div class="pages">
   <a href="#">&laquo;</a>
```

```
    <a href="#">1</a>
    <a href="#">2</a>
    <a href="#" class="active">3</a>
    <a href="#">4</a>
    <a href="#">5</a>
    <a href="#">6</a>
    <a href="#">&raquo;</a>
</div>
```

Bread Crumb: -

```
.breadcrumb{
    list-style: none;
}
.breadcrumb>li{
    display: inline;
}
.breadcrumb>li+li:before{
    content: "/\00a0";
    padding: 10px;
    color: black;
}
.breadcrumb>li>a{
    text-decoration: none;
    color: green;
}

<ul class="breadcrumb">
    <li><a href="#">Home</a></li>
    <li><a href="#">About</a></li>
    <li><a href="#">Contact</a></li>
    <li><a href="#">Help</a></li>
    <li>Some</li>
</ul>
```

CSS Multiple Columns: - this layout allows easy definition of multiple columns of text, like in newspapers

Column setup: -

Column count: - this property specifies the number of columns an element should be divided into.
Ex: - column-count: 4; //count

Column gap: - this property specifies the gap between the columns.
Ex: - column-gap: 30px; //2vw, 5%

Column design: -

Column rule style: - this property specifies the style of the rule between columns
Ex: - column-rule-style: dashed; //solid, dashed, dotted,groove

Column rule width: - this property specifies the width of the rule between columns
Ex: - column-rule-width: thick; // thin, medium, 10px

Column rule color: - this property specifies the color of the rule between columns
Ex: - column-rule-color: aqua; //rgb(0,0,0), #dddddd

Column rule: - this property is a shorthand property for setting all the column-rule properties.
column-rule: width style color;
Ex: - column-rule: 5px solid red;

Column extra: -

Column span: - this property specifies how many columns an element should span across. Used for heading in column
Ex: - column-span: all; //none
Ex: -

```
<style>
    div{column-count: 4;}
    h1{column-span: all;}
</style>
<div>
    <h1>heading</h1>
    Text ….
</div>
```

Column width: - this property specifies a suggested, optimal width for the columns.
Ex: - column-width: 200px;

Column fill: - this property specifies how to fill columns, balanced or not.
Ex: - column-fill: auto; // balance
Balance: - (Default value) Fills each column with about the same amount of content, but will not allow the columns to be taller than the height
Auto: - Fills each column until it reaches the height, and does this until it runs out of content

CSS User Interface: - there is two properties used for user interface.

Resize property: - set give access to use resizable input

resize: value; //both, none, horizontal, vertical

Ex: -

```
textarea{
   resize: none;
}
<textarea rows="5" cols="30"></textarea>
```

Ex: - resize form

```
form{
     max-width: 300px;
     max-height: 250px;
     min-width: 100px;
     min-height: 100px;
     width: 100px;
     height: 100px;
     overflow:hidden;
     resize: both;
     background-color: aqua;
}
<form>
    <input type="checkbox">
    <input type="text"><br><br>
    <textarea rows="5" cols="30"></textarea>
</form>
```

Outline offset: - it set distance between border of element and outline.

outline-offset: size; //px,%, etc also can apply negative value for show outline inside border.

Ex: -

```
div{
   outline-offset: 10px;
}
```

Resize and outline offset property apply on div: -

Ex: -

```
div{
    border: 1px solid black;
    width: 200px;
    height: 200px;
    overflow: hidden;
    resize:both;
    outline: 2px solid red;
    outline-offset: 5px;
}

<div>… … … </div>
```

Cursor: - this property specifies the mouse cursor to be displayed when pointing over an element.

cursor: value; **ex: -** cursor: wait;

Alias	all-scroll	auto	cell	col-resize	context-menu
Copy	crosshair	default	e-resize	grab	grabbing
Help	Move	n-resize	ne-resize	nesw-resize	ns-resize
nw-resize	nwse-resize	no-drop	none	not-allowed	pointer

progress	row-resize	s-resize	se-resize	sw-resize	text
w-resize	Wait	zoom-in	zoom-out		

Custom cursor image: - use url to image and auto in cursor property. Use suitable image only.
Ex: - cursor: url('../3css/res/bullets.jpg'), auto;

CSS Filters: -

Filter: - this property applies graphical effects to an element.
filter: values;

Blur: - Applies a blur effect to the image. A larger value will create more blur.
Ex: - filter: blur(5px);

None: - Default value. Specifies no effects
Ex: - filter: none;

Brightness: - Adjusts the brightness of the image. 0%=black, 100%=original
Ex: - filter: brightness(50%);

Contrast: - Adjusts the contrast of the image. 0%=black, 100%=original
Ex: - filter: contrast(150%);

Drop shadow: - Applies a drop shadow effect to the image.
filter: drop-showdow(x-axis y-axis blur color) blur and color is optional
Ex: - filter: drop-shadow(5px 5px 10px red);

Grayscale: - Converts the image to grayscale. 0%=original, 100%= complete grayscale
Ex: - filter: grayscale(100%);

Hue rotate: - Applies a hue rotation on the image. Take value in deg. 0deg to 360deg 0deg=original
Ex: - filter: hue-rotate(100deg);

Invert: - Inverts the samples in the image. 0%=original 100%= complete invert
Ex: - filter: invert(100%);

Opacity: - Sets the transparency level for the image. 0%=complete transparent 100%=original
Ex: - filter: opacity(50%);

Saturate: - Saturates the image. 0%= un-saturate 100%=original over 100%=super saturate
Ex: - filter: saturate(500%);

Sepia: - Converts the image to sepia. 0%=original 100%= complete sepia
Ex: - filter: sepia(100%);

Multiple filters: -
filter: hue-rotate(270deg) contrast(180%) invert(100%);

Backdrop filter: - The backdrop-filter property is used to apply a graphical effect to the area behind an element.
Tip: To see the effect, the element or its background must be at least partially transparent.

Backdrop-filter: values; // blur(2px), none, brightness(50%), contrast(150%), grayscale(100%), hue-rotate(100deg), invert(100%), saturate(500%), sepia(100%);

Multiple backdrop filters: -
backdrop-filter: invert(70%) hue-rotate(270deg);

Ex: -

```
.backdrop{
   border: 1px solid black;
   width: 500px;
   height: 300px;
   background-image: url(res/earthimg.jpg);
   background-size: 100% 100%;
   position: relative;
}
.inner{
   border: 1px solid black;
   position: absolute;
   left: 50%;
   top: 50%;
   transform: translate(-50%,-50%);
   width: 200px;
   height: 100px;
   backdrop-filter: invert(70%) hue-rotate(270deg);
}

<div class="backdrop">
   <div class="inner"></div>
</div>
```

CSS Responsive: -
Responsive web design makes web page look good on all devices. This uses only HTML and CSS.

Media queries: - The @media rule, introduced in CSS2, made it possible to define different style rules for different media types.
Media queries in CSS3 extended the CSS2 media type idea: Instead of looking for a type of device, they look at the capability of the device.

Media type: -
All: - Used for all media type devices
Print: - Used for print preview mode
Screen: - Used for computer screens, tablets, smart-phones etc.

Common media features: -
Orientation: - Orientation of the viewport. Landscape or portrait
Max-height: - Maximum height of the viewport
Min-height: - Minimum height of the viewport
Height: - Height of the viewport (including scrollbar)
Max-width: - Maximum width of the viewport
Min-width: - Minimum width of the viewport
Width: - Width of the viewport (including scrollbar)

Logical keywords: -
Not: - This keyword inverts the meaning of an entire media query.
Only: - This keyword prevents older browsers that do not support media queries from applying the specified styles. **It has no effect on modern browsers.**
And: - This keyword combines a media type and one or more media features.

Media query syntax: -
```
@media not|only mediatype and (media feature) and (media feature) {
  CSS-Code;
}
```

Example: -
Apply CSS effect on screen until given max width: -
```
        @media only screen and (max-width: 1290px){
          div{
             background-color: red;
             width: 1000px;
          }
        }
        @media only screen and (max-width: 700px){
          div{
             background-color: green;
             width: 500px;
          }
        }
        @media only screen and (aspect-ratio: 4/3){
          div{
             background-color: green;
             width: 500px;
          }
        }
```
Aspect ratio: 4/3, 3/2, 1/1, 8/5, etc.
Set media query for large values initially

Apply CSS effect when printing: -
```
        @media only print{
          div{
            color: red;
        }
    }
```
Display effect in print preview

Apply CSS effect when orientation is portrait: -
```
    @media (orientation: portrait)
    {
        div{
            background-color: pink;
        }
    }
```
Orientation: - landscape and portrait

Apply different style sheets to different media: -
Media: - this attribute is used for specify the media in link tag.
Ex: - <link rel="stylesheet" media="media specification" href="CSS style sheet">

Example1: -
```
<link rel="stylesheet" media="print" href="stylesheet1.css">
```
CSS style sheet: - (stylesheet1.css)
```
div{
   background-color: aqua;
}
```

Example2: -
```
<link rel="stylesheet" media="screen and (max-width: 1290px)" href="stylesheet2.css">
```
CSS style sheet: - (stylesheet2.css)
```
div{
   background-color: red;
}
```

Example: -
```
<link rel="stylesheet" media="print" href="print.css">
<link rel="stylesheet" media="screen" href="screen.css">
<link rel="stylesheet" media="screen and (min-width: 480px)" href="example1.css">
<link rel="stylesheet" media="screen and (min-width: 701px) and (max-width: 900px)"
href="example2.css">
```

Flexbox: - The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning.
Tip: - To start using the Flexbox model, you need to first define a flex container.
display: value; // flex, inline-flex
Ex: - display: flex; //define container as flex with block characteristics
```
.flexcontainer{
   display: flex;
}
```

Ex: - display: inline-flex; //define container as flex with inline characteristics

```
.flexcontainer{
   height: 50px;
   display: inline-flex;
}
```

Flex direction: - this property defines in which direction the container wants to stack the flex items.
flex-direction: value; // column, row, column-reverse, row- reverse
Ex: - flex-direction: column;

```
.flexcontainer{
   display: flex;
   flex-direction: column;
}
```

Flex wrap: - this property specifies whether the flex items should wrap or not.
flex-wrap: value; // wrap, nowrap, wrap-reverse
Ex: - flex-wrap: wrap;

```
.flexcontainer{
   display: flex;
   flex-wrap: wrap;
}
```

Flex flow: - this property is a shorthand property for setting both the flex-direction and flex-wrap properties.
flex flow: direction wrap;
Ex: - flex-flow: column wrap;

```
.flexcontainer{
   display: flex;
   flex-flow: column wrap;
}
```

Justify content: - property is used to align the flex items horizontally
justify-content: value; // center, flex-start, flex-end, space-around, space-between, space-evenly
Ex: - justify-content: center;

```
.flexcontainer{
   display: flex;
   justify-content: center;
}
```

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

Align items: - this property is used to align the flex items vertically
align-items: value; // center, flex-start, flex-end, stretch, baseline
Ex: - align-items: baseline;

```
.flexcontainer{
   display: flex;
   align-items:baseline;
   flex-wrap: wrap;
}
.o1{
   font-size: 20px;
}
```

- **stretch: -** Items are stretched to fit the container
- **center: -** Items are positioned at the center of the container
- **flex-start: -** Items are positioned at the beginning of the container
- **flex-end: -** Items are positioned at the end of the container
- **baseline: -** Items are positioned at the baseline of the container

Align content: - this property is used to align the flex lines. Set flex-wrap: wrap;
align-content: value; // space-between, space-around, stretch, center, flex-start, flex-end, space-evenly
Ex: - align-content: space-evenly;

```
.flexcontainer{
   display: flex;
   flex-wrap: wrap;
   align-content:space-evenly;
}
```

- **Stretch: -** Default value. Lines stretch to take up the remaining space
- **center: -** Lines are packed toward the center of the flex container
- **flex-start: -** Lines are packed toward the start of the flex container
- **flex-end: -** Lines are packed toward the end of the flex container
- **space-between: -** Lines are evenly distributed in the flex container
- **space-around: -** Lines are evenly distributed in the flex container, with half-size spaces on either end
- **space-evenly: -** Lines are evenly distributed in the flex container, with equal space around them

Gap: - this property specifies gap between flex boxes.
Column-gap: - it specifies column gap between flex boxes
Ex: - column-gap: size; px, %

```
.flexcontainer{
   display: flex;
   flex-wrap: wrap;
   align-content: flex-start;
   column-gap: 30px;
}
```

Row-gap: - it specifies row gap between flex boxes
Ex: - row-gap: size; px, %

```
.flexcontainer{
   display: flex;
   flex-wrap: wrap;
   align-content: flex-start;
   row-gap: 30px;
}
```

Gap: - it is shorthand property for specify row and column gap between flex boxes
gap: row/column-gap;
gap: row-gap column-gap;
Ex: -

```
gap: 30px;
gap: 10px 5px;
```

Perfect center: -

```
.flexcontainer{
   display: flex;
   align-items:center;
   justify-content: center;
}
```

Flex items: - there are properties that apply to child elements of flex container.

Order: - this property specifies the order of the flex items. The order value must be a number, default value is 0.
Ex: - order: 1;

```
.flexcontainer{ display: flex;}
#ob1{ order: 3; }
#ob2{ order: 1; }
#ob3{ order: 4; }
#ob4{ order: 2; }
```

Flex grow: - this property specifies how much a flex item will grow relative to the rest of the flex items. The value must be a number, default value is 0.
Ex: - flex-grow: 2;

```
.flexcontainer{ display: flex;}
#ob1{ flex-grow: 1; }
#ob2{ flex-grow: 1; }
#ob3{ flex-grow: 2; }
#ob4{ flex-grow: 4; }
```

Flex shrink: - this property specifies how much a flex item will shrink relative to the rest of the flex items. The value must be a number, default value is 1.
Ex: - flex-shrink: 0;

.flexcontainer{ display: flex;}
#ob2{ flex-shrink: 0; }

Flex basis: - this property specifies the initial length of a flex item.
Ex: - flex-basis: 300px;
.flexcontainer{ display: flex;}
#ob2{ flex-basis: 300px; }

Flex: - this property is a shorthand property for the flex-grow, flex-shrink, and flex-basis properties.
flex: grow shrink basis;
Ex: - flex: 0 1 200px;
.flexcontainer{ display: flex;}
#ob2{ flex: 0 1 200px; }
Ex: - flex apply in count
#ob2{ flex: 2; }
Ex: - flex apply in percentage
#ob2{ flex: 45%; }

Align self: - this property specifies the alignment for the selected item inside the flexible container. It overrides align-items property.
align-self: value; // center, flex-start, flex-end, baseline, stretch
Ex: - align-self: center;
.flexcontainer{ display: flex;}
#ob2{ align-self: center; }

Responsive menu: -
```
<!DOCTYPE html>
<html>
  <head>
    <title> Responsive Menu </title>
    <style>
      *{
        margin: 0;
        padding: 0;
      }
      ul{
        list-style: none;
        width:100%;
        background-color: black;
        display: flex;
      }
      li{
        min-width: 100px;
        height: 50px;
        border: 1px solid blue;
        background-color: royalblue;
```

```
            text-align: center;
            line-height: 50px;
            flex-basis: 250px;
         }
         a{
            text-decoration: none;
            font-size: 30px;
            color: white;
         }
         @media only screen and (max-width:1010px)
         {
            li{
               border-color: black;
            }
         }
         @media only screen and (max-width:500px)
         {
            ul{
               flex-direction: column;
               width: 100%;
            }
            li{
               width: 100%;
               height: 50px;
               flex-basis: 50px;
               border-color: black;
            }
         }
      </style>
   </head>
   <body>

      <ul>
         <li><a href="#">Home</a></li>
         <li><a href="#">About</a></li>
         <li><a href="#">Contact</a></li>
         <li><a href="#">Help</a></li>
      </ul>

   </body>
</html>
```

Responsive layout: -

```
<!DOCTYPE html>
<html>
   <head>
      <title> Web layout </title>
```

```
<style>
  *{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
  }
  .font{
    text-align: center;
    font-size: 30px;
  }
  body{
    display: flex;
    flex-direction: column;
  }
  footer,header{
    width: 100%;
    height: 200px;
    border: 1px solid black;
    background-color:yellow;
  }
  section.main{
    display: flex;
  }
  section.content{
    width: 100%;
    height: 300px;
    border: 1px solid black;
    background-color: lawngreen;
  }
  aside{
    width: 100%;
    height: 300px;
    border: 1px solid black;
    background-color: antiquewhite;
    flex-basis: 350px;
  }
  footer{
    background-color: darkgreen;
  }
  @media only screen and (max-width:600px) and (orientation: portrait)
  {
    section.main{
      flex-direction: column;
    }

    aside{
      flex-basis: 300px;
```

```
            }
         }
      </style>
   </head>
   <body>
      <header class="font">
         Header
      </header>
      <section class="main font">
         <section class="content">
            Main
         </section>
         <aside class="font">
         Side
         </aside>
      </section>
      <footer class="font">
         Footer
      </footer>
   </body>
</html>
```

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.
Display: - An HTML element becomes a grid container when its display property is set to grid or inline-grid.
display: value; // grid, inline-grid
Ex: -

```
.gridContainer{
   display:grid;
}
```

Grid template columns: - this property defines the number of columns in your grid layout, and it can define the width of each column.
grid-template-columns: col1size col2size … colNsize;
Ex: - grid-template-columns: auto auto auto;

```
.gridContainer{
   display:grid;
   grid-template-columns: auto auto auto;
}
```

Grid template rows: - this property defines the height of each row.
grid-template-rows: auto auto auto;
Ex: - grid-template-rows: auto auto auto;

```
.gridContainer{
   display:grid;
   grid-template-columns: auto auto auto;
   grid-template-rows: auto auto auto;
}
```
Tip: - if there is more items in than define columns that remaining items will get automatically added to new row.

Justify content: - this property is used to align the whole grid inside the container.
Note: - The grid's total width has to be less than the container's width for the justify-content property to have any effect.
justify-content: values; // space-evenly, space-around, space-between, center, start, end
Ex: - justify-content: center;
```
.gridContainer{
   display:grid;
   grid-template-columns: auto auto auto;
   grid-template-rows: auto auto auto;
   justify-content: center;
}
```

Align content: - this property is used to vertically align the whole grid inside the container.
Note: - The grid's total height has to be less than the container's height for the align-content property to have any effect.
align-content: values; // center, space-evenly, space-around, space-between, start, end
Ex: - align-content: center;
```
.gridContainer{
   display:grid;
   grid-template-columns: auto auto auto;
   grid-template-rows: auto auto auto;
   align-content: center;
}
```

Justify items: - this property set default alignment of all items in grid horizontally.
justify-items: values; center, start, end, stretch, baseline
Ex: - justify-items: center;
```
.gridContainer{
   display:grid;
   grid-template-columns: auto auto auto;
   grid-template-rows: auto auto auto;
   justify-items: center;
}
```

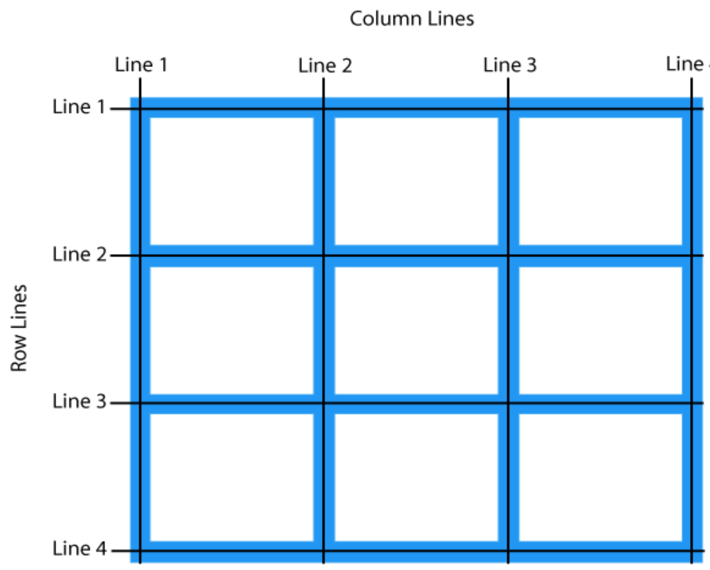
Align items: - this property set default alignment of all items in grid vertically.
justify-items: values; center, start, end, stretch, baseline
Ex: - align-items: center;
```
.gridContainer{
   display:grid;
   grid-template-columns: auto auto auto;
   grid-template-rows: auto auto auto;
   align-items: center;
```

}

Gap: - it define gap between grid items.
Grid column gap: - Specifies the size of the gap between colums
grid-column-gap: 30px;
Grid row gap: - Specifies the size of the gap between rows
grid-row-gap: 30px;
Grid gap: - A shorthand property for the grid-row-gap and grid-column-gap properties
grid-gap: 30px 10px;

Grid items: - there are properties that apply to child elements of grid container.

Grid lines: -
The lines between columns are called column lines.
The lines between rows are called row lines.



Grid column and row: - By default, a container has one grid item for each column, in each row; style the grid items for span multiple columns or rows.

Grid column start: - Specifies where to start the grid item in column
Ex: - grid-column-start: 1;
Grid column end: - Specifies where to end the grid item in column
Ex: - grid-column-end: 3;
Grid column: - it is short hand property of grid-column-start and grid-column-end
grid-column: start;
Ex: - grid-column: 3;
grid-column: start/end;
Ex: - grid-column: 1/3;
grid-column: start/span count;
Ex: - grid-column: 1/span 3;

Grid row start: - Specifies where to start the grid item in row
Ex: - grid-row-start: 1;
Grid row end: - Specifies where to end the grid item in row
Ex: - grid-row-end: 3;
Grid row: - it is short hand property of grid-row-start and grid-row-end
grid-row: start;
Ex: - grid-row: 3;
grid-row: start/end;
Ex: - grid-row: 1/3;
grid-row: start/span row;
Ex: - grid-row: 1/span 3;

Grid area: - it is short hand property of grid-row-start, grid-column-start, grid-row-end and the grid-column-end properties.
grid-area: row-line-start/column-line-start/row-line-end/ column-line-end
Ex: -
grid-area: 1/1/3/2;
grid-area: 1/1/span 3/span 2;

Justify self: - it set alignment to particular grid item horizontally.
justify-self: value; //baseline, center, start, end, stretch
Ex: -
justify-self: end;

Align self: - it set alignment to particular grid item vertically.
justify-self: value; //baseline, center, start, end, stretch
Ex: -
justify-self: end;

Grid auto column and row: -
Grid auto flow: - this property controls how auto-placed items get inserted in the grid.
Grid-auto-flow: value; row, column, dense, row dense, column dense
Row: - Default value. Places items by filling each row
column: - Places items by filling each column
dense: - Place items to fill any holes in the grid
row dense: - Places items by filling each row, and fill any holes in the grid
column dense: - Places items by filling each column, and fill any holes in the grid
Ex: -

```
.gridcontainer{
   grid-template-columns: auto auto auto;
   grid-template-areas: auto auto;
   grid-auto-flow: dense;
}
.item3{
   grid-column: auto /span 2;
}
```

Grid auto column: - set default size for all columns, this property is get override by grid-template property.
Note: - Set grid-area in particular item for create new row
grid-auto-columns: length; // 25%, 20px
Ex: -
```
.gridcontainer{
   grid-auto-columns: 90px;
}
.item3{
   grid-area: 1/3/1/4;
}
```

Grid auto row: - this property sets a size for the rows in a grid container.
This property affects only rows with the size not set.
Ex: -
```
.gridcontainer{
   grid-auto-rows: 30px;
   grid-auto-flow: column;
}
.item2{
   grid-area: 2/1/2/2;
}
```

Naming grid: - The grid-area property can also be used to assign names to grid items.
Grid template area: - Named grid items can be referred to by the grid-template-areas property of the grid container.
Each row is defined by apostrophes (' ').
The columns in each row are defined inside the apostrophes, separated by a space.

Ex: -
```
.gridContainer{
   grid-template-areas: 'myarea myarea myarea myarea';
}
.item1{
   grid-area: myarea;
}
```

Note: - A period sign (.) represents a grid item with no name.
Ex: - span named grid-item with multiple column
```
.gridContainer{
   grid-template-areas: 'myarea myarea . .';
}
.item1{
   grid-area: myarea;
}
```

Ex: - define multiple rows with another set of apostrophes

```
.gridContainer{
   grid-template-areas: 'myarea myarea . .' . . . .';
}
.item1{
   grid-area: myarea;
}
```

Ex: - span multiple row and column

```
.gridContainer{
   grid-template-areas: 'myarea myarea . .' 'myarea myarea . .';
}
.item1{
   grid-area: myarea;
}
```

Ex: - create ready to use web-template

```
<!DOCTYPE html>
<html>
   <head>
      <title> Grid </title>
      <style>
         *{
            margin: 0;
            padding: 0;
         }
         body{
            height: 150vh;
            display: grid;
            grid-template-areas: 'header header header header'
                                 'nav main main main'
                                 'nav main main main'
                                 'nav main main main'
                                 'nav footer footer footer';
         }
         .border{   border: 1px solid black;}
         header{   grid-area: header; }
         div.nav{   grid-area: nav; }
         section.main{   grid-area: main; }
         footer{   grid-area: footer; }
         @media only screen and (max-width:600px) and (orientation:portrait)
         {
            body{
               height: 150vh;
               grid-template-areas: 'header header header'
                                    'nav main main'
                                    'nav main main'
                                    'nav main main'
```

```
                                'footer footer footer';
            }
          }
        </style>
    </head>
    <body>
        <header class="border"> Header </header>
        <div class="nav border"> Navbar </div>
        <section class="main border"> Content </section>
        <footer class="border"> Footer <footer>
    </body>
</html>
```

CSS Extra: -

Root pseudo class: - root pseudo class represents html tag to apply style to html element.
Class: - :root{ ... }
Ex: -

```
<!DOCTYPE html>
<html>
    <head>
        <title> root </title>
        <style>
          :root{
            background-color: red;
            color: white;
          }
        </style>
    </head>
    <body>
      <h1>Hello world</h1>
    </body>
</html>
```

Variables: - declare variable in CSS with '--'before variables name and assign value to that variable. These variables can be declared inside selector's body or root pseudo class for global declaration. Local variables override global variables value. Use var function to assign variables values to properties.
Variables: - --border: 1px solid black, --size: 400px
Function: - var(--border), var(--size)

Ex: -

```
<!DOCTYPE html>
<html>
    <head>
        <title> variables </title>
        <style>
```

```
      :root{
        --size: 400px;
        --border: 1px solid black;
      }
      .d1{
        --size: 300px;
        border: var(--border);
        width: var(--size);
        height: var(--size);
      }
    </style>
  </head>
  <body>
    <h1>Hello world</h1>
    <div class="d1"> </div>
  </body>
</html>
```

Scroll behavior: - set smooth scrolling for links clicking focus targeted id location in same page.
scroll-behavior:smooth; //auto

Ex: -

```
*{
  margin: 0px;
  padding: 0px;
  scroll-behavior:smooth;
}
```