CSS (Cascading Style Sheets)

Index :

1.Introduction
2.CSS Syntax
3.CSS Selectors
4.Ways to Insert CSS
5.CSS Comments

6.CSS Colors
7.CSS Background
8.CSS Borders
9.CSS Margins
10.CSS Padding
11.CSS Outline

12.CSS Height and Width
13.CSS Text
14.CSS Font

15.Cursors
16.CSS List
17. display & visibility

18.CSS Overflow
19.CSS Position and Z-index
20.CSS Float and clear

21.CSS Opacity / Transparency
22.CSS Image Sprites
23.CSS Counters

24. 2D & 3D Transforms
25. Transitions
26.CSS Animations

27.CSS Multiple Columns
28.CSS Grid
29.CSS Flex
30.CSS Miscellaneous like css functions,resize..etc

1.Introduction
==========

.CSS stands for Cascading Style Sheets
.CSS is the language we use to style an HTML document.

Example :
The water cascaded over the rocks ..similarly Style Sheets cascaded over the Html elements .

2.CSS Syntax
==========

```
Syntax   :  selector{
                              property:value;
                            }
        ex :     h1      {
                            color:blue;
                           }
```

Here

```
        h1    --->selector [The selector points to the HTML element you want to style.]
        color--->property
        blue ---->value
```

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

3.CSS Selectors
===========

.CSS selectors are used to "find" (or select) the HTML elements that you want to style it.

We can divide CSS selectors into five categories:

1. Simple selectors	(select elements based on name, id, class)
2. Combinator selectors	(select elements based on a specific relationship between them)
3. Attribute selectors	(select elements based on an attribute or attribute value)
4. Pseudo-elements selectors	(select and style a part of an element)
5. Pseudo-class selectors	(select elements based on a certain state)

1. Simple selectors   (select elements based on name, id, class)
----

i)Universal selector (*)
ii)Element Type selector (h1,p,div.....etc)
iii)class selector (.)
iv)Id selector   (#)

2.Combinator Selectors   (Select elements based on a specific relationship b/w them)
----

i)Descendant selector (space)   div   p Selects   ALL   <p> elements inside <div> elements.
ii)Child selector   (>)   div > p Selects   ALL   <p> elements where the PARENT is a <div> element
iii)Adjacent sibling selector   (+)   div + p Selects the FIRST <p> element that are placed IMMEDIATELY after <div> elements
iv)General sibling selector   (~) //~=tilda   p ~ ul Selects   Every<ul> element that are preceded by a <p> element

3.Attribute selector   (Select element based on an attribute or attribute value) [~,|,^,$,*]
----

i)[attribute]

ii)[attribute=value]
iii)[attribute ~=value]   [~=tilda, select elements with an attribute value CONTAINING a SPECIFIC word.]

iv)[attribute |=value]   [|=vertical Line, select elements with the specified attribute STARTING with the specified va
lue. ex :value,value-name,valuename(x)]

v)[attribute^=value] [^=caret , selector is used to select elements whose attribute value BEGINS with a specified val
ue. ex :value,value-name,value_name,valuename]
vi)[attribute$=value] [$=dollar,selector is used to select elements whose attribute value ENDS with a specified value
. ex :value,name-value,name_value,namevalue]
vii)[attribute*=value] [selector is used to select elements whose attribute value CONTAINS a specified value.]

4. Pseudo-elements selectors [::]   (select and style a part of an element)

----

i)::first-letter	p::first-letter	Selects the first LETTER of each <p> element
ii)::first-line	p::first-line	Selects the first LINE of each <p> element
iii)::before	p::before	Insert something BEFORE the content of each <p> element
iv)::after	p::after	Insert something AFTER the content of each <p> element
v)::selection	p::selection	Selects the PORTION of an element that is selected by a user

5. Pseudo-class selectors   [:]   (select elements based on a certain state)

----

i):root

ii):hover
iii):focus

iv):disabled
v):enabled

vi):optional
vii):required

viii):valid
ix):invalid

x):out-of-range
xi):in-range

xii):read-only
xiii):read-write
xiv):checked

xv):link
xvi):visited
xvii):active

xviii):empty
xix) :lang()
xx):not(selector)
xxi):target

xxii):first-child
xxiii):first-of-type

xxiv):last-child
xxv):last-of-type

xxvi):nth-child(n)
xxvii):nth-of-type(n)

xxviii):nth-last-child(n)
xxix):nth-last-of-type(n)

xxx):only-child
xxxi):only-of-type

4.Ways to Insert CSS
===============

There are three ways of inserting a style sheet:

i)External CSS (for all files) :External styles are defined within the <link> element, inside the <he
ad> section of an HTML page: ex <link rel="stylesheet" href="mystyle.css">
ii)Internal CSS (only for one file) :The internal style is defined inside the <style> element, inside the he
ad section.
iii)Inline CSS (only for one HTML ele) :add the style attribute to the relevant element. ex :<h1 style="c
olor:blue;text-align:center;">This is a heading</h1>

5.CSS Comments
============

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


/* This is
a multi-line
comment */

p {
  color: red;
}
```

5.CSS Colors
============

Colors are specified using predefined color names,
or RGB,RGBA,
HEX,
HSL, HSLA values.

i)color names
In CSS, a color can be specified by using a predefined color name:

```
<h1 style="background-color:DodgerBlue;">Hello World</h1>
```

ii)RGB,RGBA
In CSS, a color can be specified as an RGB value, using this formula:

rgb(red, green, blue)

ex :rgb(255, 99, 71)

.Each parameter (red, green, and blue) defines the intensity of the color between 0 and 255.

An RGBA color value is specified with:

rgba(red, green, blue, alpha) [alpha channel - which specifies the opacity (0.0 to 1.0) for a color.]

ex :rgba(255, 99, 71, 0.5)

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

#rrggbb

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

ex :#ff6347

iv)HSL or HSLA
In CSS, a color can be specified using hue, saturation, and lightness (HSL) in the form:

hsl(hue, saturation, lightness)

hsla(hue, saturation, lightness, alpha)

.Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.
.Saturation is a percentage value, 0% means a shade of gray, and 100% is the full color.
.Lightness is also a percentage, 0% is black, 50% is neither light or dark, 100% is white
.The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

ex :hsl(0, 100%, 50%)

ex:hsla(9, 100%, 64%, 0.2)

7.CSS Background:
============

i)background-color : color|transparent|initial|inherit; [defVal:transparent]
(Sets the background color of an element)

ii)background-image : url|none|initial|inherit; [defVal:none]
(Sets the background image for an element)

iii)background-repeat : repeat|repeat-x|repeat-y|no-repeat|space|round|initial|inherit; [defVal:repeat]
(Sets how a background image will be repeated)

iv)background-attachment : scroll|fixed|local|initial|inherit; [defVal:scroll]
(Sets whether a background image is fixed or scrolls with the rest of the page)

v)background-position : value; [defVal :0% 0%]
left top,left center,left bottom,
right top,right center,right bottom
center top,center center,center bottom
(Sets the starting position of a background image)

vi)background-size : auto|length|cover|contain|initial|inherit; [defVal : auto]
(Specifies the size of the background image(s))

vii)background-clip : border-box|padding-box|content-box|initial|inherit; [defVal:border-box]
(Specifies the painting area of the background or trim the background item)

viii)background-origin :background-origin: padding-box|border-box|content-box|initial|inherit; [defVal:padding-bo
x]
(Specifies where the background image(s) is/are positioned)

(short hand property)
ix)background :background-color background-image background-repeat background-attachment ba
ckground-position

|Gradient Backgrounds|
================
.CSS gradients let you display smooth transitions between two or more specified colors.

CSS defines two types of gradients:

.Linear Gradients (goes down/up/left/right/diagonally)
.Radial Gradients (defined by their center)

a)CSS Linear Gradients : Top to Bottom (this is default)

.To create a linear gradient you must define at least two color stops.
Color stops are the colors you want to render smooth transitions among.
You can also set a starting point and a direction (or an angle) along with the gradient effect.

syntax :
background-image: linear-gradient(direction, color-stop1, color-stop2, ...);

```
ex :  background-image: linear-gradient(red, yellow);  //Top to Bottom (this is default)

       background-image:linear-gradient(to right ,red,yellow);//left to right
           //Direction - Diagonal
            background-image:liner-gradient(to bottom right,red,yellow);//top left to bottom right

           background-image:liner-gradient(to bottom left,red,yellow);//top right to bottom left
```

Using Angles

.If you want more control over the direction of the gradient, you can define an angle,
 instead of the predefined directions (to bottom, to top, to right, to left, to bottom right, etc.).
  i)A value of 0deg is equivalent to "to top".
  ii)A value of 90deg is equivalent to "to right".
  iii) A value of 180deg is equivalent to "to bottom".

Syntax
background-image: linear-gradient(angle, color-stop1, color-stop2);

```
  ex :
         background-image: linear-gradient(180deg, red, yellow);   //180deg -- to bottom
```

Using Multiple Color Stops

```
ex :  background-image: linear-gradient(red, yellow, green);
       background-image: linear-gradient(to right, red,orange,yellow,green,blue,indigo,violet); //rainbow
```

Using Transparency

```
  ex :background-image: linear-gradient(to right, rgba(255,0,0,0), rgba(255,0,0,1));
```

Repeating a linear-gradient:

```
ex :  background-image: repeating-linear-gradient(to right,red, yellow 10%, green 20%);
        background-image: repeating-linear-gradient(45deg,red,yellow 7%,green 10%);
        background-image: repeating-linear-gradient(90deg,red,yellow 7%,green 10%);
        background-image: repeating-linear-gradient(190deg,red,yellow 7%,green 10%);
```

b)Radial Gradients
.A radial gradient is defined by its center.
.To create a radial gradient you must also define at least two color stops.
.By default, shape is ellipse, size is farthest-corner, and position is center.

Syntax:
background-image: radial-gradient(shape size at position, start-color, ..., last-color);
.The shape parameter defines the shape. It can take the value circle or ellipse.
 The default value is ellipse.

:shape size at position ---->ellipse|circle

ex :
```
        background-image: radial-gradient(red, yellow, green);
       background-image: radial-gradient(red 5%, yellow 15%, green 60%);
       background-image: radial-gradient(circle, red, yellow, green);
```

Use of Different Size Keywords

The size parameter defines the size of the gradient. It can take four values:

closest-side
farthest-side
closest-corner
farthest-corner

ex :
```
     background-image: radial-gradient(farthest-side at 60% 55%, red, yellow, black);
     background-image: radial-gradient(closest-corner at 60% 55%, red, yellow, black);
     background-image: radial-gradient(farthest-corner at 60% 55%, red, yellow, black);
     background-image: radial-gradient(farthest-corner at 60% 55%, red, yellow, black);
```

Using Repeating a radial-gradient :
----
ex :
```
background-image: repeating-radial-gradient(red, yellow 10%, green 15%);
```

8.CSS Borders
=========
.The CSS border properties allow you to specify the width,style and color of an element's border.

i)border-width :   thin|medium|thick|length;
.The border-width property specifies the width of the four borders.
```
  border-width: 25px 10px 4px 35px; /* 25px top, 10px right, 4px bottom and 35px left */
          or
border-width: 25px;
```

ii)border-style :   dotted |dashed|solid|double|groove|ridge|inset|outset|none|hidden.
The border-style property specifies what kind of border to display.
```
 border-style: dotted dashed solid double;
         or
border-style: solid
      or
border-top-style: dotted;
border-right-style: solid;
border-bottom-style: dotted;
border-left-style: solid;          (similar for border-width & color)
```

iii)border-color
.The border-color property is used to set the color of the four borders.
```
  border-color: red green blue yellow; /* red top, green right, blue bottom and yellow left */
```

or
border-color: red ;

Note : (shorthand property)
border : border-width border-style border-color
ex: border: 5px solid red;
or
ex: border-left/right/top/bottom: 6px solid red;

iv)border-radius :length
.The border-radius property is used to add rounded borders to an element:

ex :border-radius: 5px;
or
border-top-left-radius
border-top-right-radius
border-bottom-right-radius
border-bottom-left-radius

v)border-collapse :separate|collapse|initial|inherit; [defVal :separate] (only for table)
(property sets whether table borders should collapse into a single border or be separated as in standard HTML.)

vi)border-images:
----
i)border-image :border-image-source border-image-slice border-image-width border-image-outset border-image-rep
eat;
.The border-image property allows you to specify an image to be used as the border around an element.

ex : border-image: url(border.png) 30 round;

a)border-image-source :none|image|initial|inherit;
(Specifies the path to the image to be used as a border)
.The border-image-source property specifies the path to the image to be used as a border (instead of the normal bord
er around an element).
Note :
If the value is "none", or if the image cannot be displayed, the border styles will be used.

c)border-image-width :number|%|auto|initial|inherit;
The border-image-width property can take from one to four values (top, right, bottom, and left sides)

d)border-image-outset :length|number|initial|inherit;
.The border-image-outset property specifies the amount by which the border image area extends beyond the border b
ox.
.The border-image-outset property can take from one to four values (top, right, bottom, and left sides).

e)border-image-repeat :stretch|repeat|round|initial|inherit;
(Specifies whether the border image should be repeated, rounded or stretched)

9.CSS Margins

==========

.The CSS margin properties are used to create space around elements, outside of any defined borders.

```
 ex :  margin-top: 100px;
       margin-bottom: 100px;
       margin-right: 150px;
       margin-left: 80px;
```

i)Margin - Shorthand Property [defVal:auto]

```
 -0p     margin :margin-top margin-right margin-bottom margin-left
```

ex : margin: 25px 50px 75px 100px;

10.CSS Padding

==========

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

ex :

```
      padding-top: 50px;
     padding-right: 30px;
     padding-bottom: 50px;
     padding-left: 80px;
```

i)Padding - Shorthand Property

```
  padding :padding-top padding-right padding-bottom padding-left
  padding: 25px 50px 75px 100px;
```

This element has a black border and a green outline with a width of 10px.

11.CSS Outline:

==========

.An outline is a line that is drawn around elements, OUTSIDE the borders, to make the element "stand out".

i)outline-width : thin|medium|thick|length;
(Sets the width of an outline)
ex : outline-width: thin;

ii)outline-style :dotted |dashed|solid|double|groove|ridge|inset|outset|none|hidden.
(Sets the style of an outline)
ex :outline-style: solid;

iii)outline-color
(Sets the color of an outline)
ex : outline-color: red;

A shorthand property

outline :outline-width outline-style outline-color
outline : 2px dashed red;

iv)outline-offset
(Specifies the space between an outline and the edge or border of an element)
ex : outline-offset: 15px;

12.CSS Height and Width

=================

.The height and width properties are used to set the height and width of an element.
.auto - This is default. The browser calculates the height and width
ex :

```
  height: 100px;
  width: 500px;
```

height	Sets the height of an element
max-height	Sets the maximum height of an element
max-width	Sets the maximum width of an element
min-height	Sets the minimum height of an element
min-width	Sets the minimum width of an element
width	Sets the width of an element

13.CSS Text

========

i)color :colornames..
(Sets the color of text)

ii)direction :ltr|rtl|initial|inherit; [defVal :ltr]
(Specifies the text direction/writing direction)

iii)text-align :text-align: left|right|center|justify|initial|inherit; [defVal :left]
(Specifies the horizontal alignment of text)

iv)text-align-last :auto|left|right|center|justify|start|end|initial|inherit; [defVal:auto]
(The text-align-last property specifies how to align the last line of a text.)

v)text-indent :text-indent: length|initial|inherit; [DefVal :opx]
(Specifies the indentation of the first line in a text-block)

vi)letter-spacing : normal|length|initial|inherit; [DefVal:normal]
(Increases or decreases the space between characters in a text)

vii)word-spacing :normal|length|initial|inherit; [DefVal:normal]
(Increases or decreases the space between words in a text)

viii)line-height :normal|number|length|initial|inherit; [DefVal:normal]
(Sets the line height)

ix)text-transform :none|capitalize|uppercase|lowercase|initial|inherit; [DefVal:none]

(Controls the capitalization of text)

x)text-decoration
(Specifies the decoration added to text)

text-decoration-line (required)	: none|underline|overline|line-through|initial|inherit; [defVal:none]
text-decoration-color	:colorNames..
text-decoration-style	: solid|double|dotted|dashed|wavy|initial|inherit; [defVal:solid]

Short hand property

text-decoration :text-decoration-line text-decoration-style text-decoration-color;
ex : text-decoration: overline solid red ;

xi)text-overflow :clip|ellipsis|string|initial|inherit; [defVal:clip]
(Specifies how overflowed content that is not displayed should be signaled to the user)

xii)vertical-align :baseline|length|sub|super|top|text-top|middle|bottom|text-bottom|initial|inherit; [defV al:baseline]
(Sets the vertical alignment of an element)

xiii)white-space :normal|nowrap|pre|pre-line|pre-wrap|initial|inherit; [DefVal:normal]
(The white-space property specifies how white-space inside an element is handled.)

xiv)CSS Text Effects

a)text-overflow
b)word-wrap
c)word-break
d)writing-mode

a)text-overflow :clip|ellipsis|string|initial|inherit; [defVal:clip]
(Specifies how overflowed content that is not displayed should be signaled to the user)

b)word-wrap :normal|break-word|initial|inherit;
.The CSS word-wrap property allows long words to be able to be broken and wrap onto the next line.

c)word-break :normal|break-all|keep-all|break-word|initial|inherit;
The word-break property specifies how words should break when reaching the end of a line.

d)writing-mode : horizontal-tb|vertical-rl|vertical-lr;
Specifies whether lines of text are laid out horizontally or vertically

CSS Shadow Effects
xv)
a)text-shadow :
(Specifies the shadow effect added to text)

syntax :
    text-shadow : h-shadow v-shadow blur-radius color;
ex: text-shadow : 2px 2px 8px #FF0000;

b)box-shadow   :none|unset|...etc
.The CSS box-shadow property applies shadow to elements.
In its simplest use, you only specify the horizontal shadow and the vertical shadow:

syntax :
    box-shadow :offset-x offset-y blur-radius spread-radius color;
ex : box-shadow :2px 2px 2px 1px blue;

14.CSS Font

=========

i)font-style   :normal|italic|oblique|initial|inherit;   [defVal:normal]
(Specifies the font style for text)

ii)font-variant   :normal|small-caps|initial|inherit;   [defVal:normal]
(Specifies whether or not a text should be displayed in a small-caps font)

iii)font-weight   :normal|bold|bolder|lighter|number|initial|inherit;   [defVal:normal]
(Specifies the weight of a font)

iv)font-size   :medium|xx-small|x-small|small|large|x-large|xx-large|smaller|larger|length|initial|inherit; [DefVal :medium]
(Specifies the font size of text)

v)font-family   :family-name|generic-family|initial|inherit;
(Specifies the font family for text)

.Short hand property:

    font: font-style font-variant font-weight font-size/line-height font-family;

ex: font: italic small-caps bold 12px/30px Georgia, serif;

15.Cursors

=======

cursor
.Different types of cursors :
auto ,crosshair,default,e-resize,help,move,n-resize,ne-resize,nw-resize,
pointer,progress,s-resize,se-resize,sw-resize,
text,w-resize,wait....etc

16.CSS List :

=========

i)list-style-type :circle|square|decimal |decimal-leading-zero |lower-alpha |lower-roman |none |upper-alpha |upper-roman|initial ...etc[defVal =disc]
(Specifies the type of list-item marker)

ii)list-style-image   : none|url|initial|inherit   [DefVal:none]
(Specifies an image as the list-item marker)

iii)list-style-position : inside|outside|initial|inherit;   [defVal:outside]
(Specifies the position of the list-item markers (bullet points))

Short Hand property :

list-style: list-style-type list-style-position list-style-image|initial|inherit;
ex: list-style: square inside url("image1.gif");

17. display & visibility :
================
i)visibility : visible|hidden|collapse|initial|inherit;   [DefVal:visible]
(Specifies whether or not an element should be visible)

Note :collapse
Only for table rows (<tr>), row groups (<tbody>), columns (<col>), column groups (<colgroup>).
This value removes a row or column, but it does not affect the table layout.
The space taken up by the row or column will be available for other content.
If collapse is used on other elements, it renders as "hidden

ii)display
.The display property is the most important CSS property for controlling layout.
.Every HTML element has a default display value depending on what type of element it is.
The default display value for most elements is block or inline.

a)Block-level Elements
i)It takes new line
ii)takes up the full width available
iii)it takes height ,width ,margin,padding...etc

ex :<div>,<h1> - <h6>,<p>,<form>,<header>,<footer>,<section>...etc

b)Inline Elements
i)It does not new line
ii)takes up the full width available
iii)it does not takes height ,width ,margin,padding...etc

ex :<span>,<a>,<img>..etc

c)inline-block
i)It does not new line
ii)takes up the full width available
iii)it takes height ,width ,margin,padding...etc

->display possible values are :
block,
inline,
inline-block,

flex,
grid,
inline-flex,
inline-grid,

list-item,

inline-table,
table,(<table>)
table-caption, (<caption>)
table-header-group,(<thead>)
table-row-group, (<tbody>)
table-footer-group, (<tfoot>)
table-column-group,(<colgroup>)
table-row ,   (<tr>)
table-cell   (<td>)
table-column   (<col>)

18.CSS Overflow

===========

overflow   :visible |hidden |scroll|auto [DefVal :visible]
.The overflow property specifies whether to clip the content or to add scrollbars
when the content of an element is too big to fit in the specified area.

Note: The overflow property only works for BLOCK elements with a specified height.

i)overflow-x
(Specifies what to do with the left/right edges of the content if it overflows the element's content area)

ii)overflow-y
(Specifies what to do with the top/bottom edges of the content if it overflows the element's content area)

19.CSS Position and Z-index

===================

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

There are five different position values:

i)static   (Default)
ii)relative
iii)fixed
iv)absolute
v)sticky

.Elements are then positioned using the top, bottom, left, and right properties. However,
 these properties will not work unless the position property is set first.
.top, bottom, left, and right properties sets margin edge for a positioned box

i)position: static;
.Static positioned elements are not affected by the top, bottom, left, and right properties.
.An element with position: static; is not positioned in any special way;
 it is always positioned according to the normal flow of the page:

ii)relative
.An element with position: relative; is positioned relative to its normal position.
.Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position.

iii)fixed
.An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same plac e even if the page is scrolled.
The top, right, bottom, and left properties are used to position the element.

iv)absolute
.An element with position: absolute; is positioned relative to the nearest positioned ancestor .
.However; if an absolute positioned element has no positioned ancestors,
 it uses the document body, and moves along with page scrolling.

Note: A "positioned" element is one whose position is anything except static.

v)sticky
.An element with position: sticky; is positioned based on the user's scroll position.
A sticky element toggles between relative and fixed, depending on the scroll position.

Z-index :
=====
i)z-index: auto|number|initial|inherit;   [DefVal :auto]

.The z-index property specifies the stack order of an element.
.An element with greater stack order is always in front of an element with a lower stack order.

Note: z-index only works on positioned elements (position: absolute, position: relative, position: fixed, or position: s ticky).

ex :
```
position: absolute;
  left: 0px;
  top: 0px;
  z-index: -1;
```

20.CSS Float and clear
===============
i)float   :left|right|none ;   [defVal:none]
.The CSS float property specifies how an element should float.
The float property is used for positioning and formatting content
 e.g. let an image float left to the text in a container

ii)clear   :left|right|both|none;
.The clear property specifies what elements can float beside the cleared element and on which side.
.The most common way to use the clear property is after you have used a float property on an element.
.When clearing floats, you should match the clear to the float:
  If an element is floated to the left, then you should clear to the left.

21.CSS Opacity / Transparency

=====================

.The opacity property specifies the opacity/transparency of an element.
.The opacity property can take a value from 0.0 - 1.0. The lower value, the more transparent:
.Default value is 1.0

ex :
```
  background: rgba(76, 175, 80, 0.3) /* Green background with 30% opacity */
```

22.CSS Image Sprites

===============

.An image sprite is a collection of images put into a single image.
.A web page with many images can take a long time to load and generates multiple server requests.
.Using image sprites will reduce the number of server requests and save bandwidth.

.By using background-position:width height;

ex: background: url('img_navsprites.gif') 0 0;

23.CSS Counters

============

.CSS counters are like "variables". The variable values can be incremented by CSS rules
(which will track how many times they are used).

To work with CSS counters we will use the following properties:

.counter-reset	- Creates or resets a counter
.counter-increment	- Increments a counter value
.content	- Inserts generated content
.counter() or counters() function	- Adds the value of a counter to an element

To use a CSS counter, it must first be created with counter-reset.

ex :
```
body {
  counter-reset: section;
}

h2::before {
  counter-increment: section;
  content: "Section " counter(section) ": ";
}

<body>
```

```
<h1>Using CSS Counters:</h1>
<h2>HTML Tutorial</h2>
<h2>CSS Tutorial</h2>
<h2>JavaScript Tutorial</h2>

</body>
```

output:

```
Using CSS Counters:

Section 1 :HTML Tutorial
Section 2 :CSS Tutorial
Section 3 :JavaScript Tutorial
```

24. 2D & 3D Transforms
===============

.CSS transformations allow you to translate,rotate,scale,and skew elements.
.A transformation is an effect that lets an element change shape,size and position

CSS 2D Transforms Methods :

i) translate()
ii) rotate()

iii) scale(), scaleX(),scaleY()

iv) skew(), skewX(),skewY()

v) matrix()

i)translate :translate(x,y)
The translate() method moves an element from its current position (according to the parameters given for the X-axis and the Y-axis).

```
ex :   transform: translate(50px, 100px);
```

ii)rotate :rotate(deg)
The rotate() method rotates an element clockwise or counter-clockwise according to a given degree.

```
ex :   transform: rotate(20deg);          [negative val :anti or counter-clockwise]
```

iii) scale() ,scaleX(),scaleY()
The scale() method increases or decreases the size of an element (according to the parameters given for the width an d height).
The scaleX() method increases or decreases the width of an element.
The scaleY() method increases or decreases the height of an element.

```
ex :   transform: scale(2, 3);
        transform: scaleX(2);
       transform: scaleY(2);
```

iv) skew(), skewX(),skewY()
The skew() method skews an element along the X and Y-axis by the given angles.
The skewX() method skews an element along the X-axis by the given angle.
The skewY() method skews an element along the Y-axis by the given angle.

```
ex :   transform: skew(20deg, 10deg);
```

        transform: skewX(20deg);
        transform: skewY(20deg);

v) matrix()
The matrix() method combines all the 2D transform methods into one.
syntax :
   matrix(scaleX(),skewY(),skewX(),scaleY(),translateX(),translateY())

  ex: transform: matrix(1, -0.3, 0, 1, 0, 0);

 CSS transform property supports 3D transformation methods:
i)translate3d(x,y,z) ,translateX(x),translateY(y),translateZ(z)
The translate() method moves an element from its current position (according to the parameters given for the X-axis and the Y-axis,Z-axis)
ex :   transform: translate(50px, 100px,40px);

ii)scale3d(x,y,z), scaleX(),scaleY(),scaleZ();
Defines a 3D scale transformation
ex :scale3d(2,3,4)

iii)rotate3d(x,y,z,angle) ,rotateX(), rotateY(), rotateZ()
Defines a 3D rotation
ex :rotate3d(1,3,5,45deg)

iv)perspective(n) :perspective: length|none;
Defines a perspective view for a 3D transformed element.
The perspective property is used to give a 3D-positioned element some perspective.
The perspective property defines how far the object is away from the user.
So, a lower value will result in a more intensive 3D effect than a higher value.
ex :   perspective: 150px;

v)matrix3d
Defines a 3D transformation, using a 4x4 matrix of 16 values
matrix3d(a1, b1, c1, d1, a2, b2, c2, d2, a3, b3, c3, d3, a4, b4, c4, d4)
a1 b1 c1 d1 a2 b2 c2 d2 a3 b3 c3 d3
    Are <number>s describing the linear transformation.
a4 b4 c4 d4
    Are <number>s describing the translation to apply.

25. Transitions:
==========

i)transition-duration   : time|initial|inherit;
Specifies how many seconds or milliseconds a transition effect takes to complete
ex :   transition-duration: 5s;

ii)transition-property   :none|all|property|initial|inherit;
Specifies the name of the CSS property the transition effect is for
ex : transition-property: width;

iii)transition-delay   :time|initial|inherit;
Specifies a delay (in seconds) for the transition effect
ex : transition-delay: 2s;

iv)transition-timing-function
        : linear|ease|ease-in|ease-out|ease-in-out|step-start|step-end|steps(int,start|end)|cubic-bezier(n,n,n,n);
Specifies the speed curve of the transition effect
ex :transition-timing-function: linear;

.Short hand property :
        transition: property duration timing-function delay; //defVal :all 0s ease 0s
   ex : transition: width 2s ease 2s;

26.CSS Animations:

============

@keyframes:
.When you specify CSS styles inside the @keyframes rule, the animation will gradually
  change from the current style to the new style at certain times.
.To get an animation to work, you must bind the animation to an element

```
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}
or
@keyframes example {
  0%   {background-color: red;}
  25%  {background-color: yellow;}
  50%  {background-color: blue;}
  100% {background-color: green;}
}
animation-name: example
```

i)animation-name   :keyframename|none;   [defVal:none]
Specifies the name of the @keyframes animation
ex : animation-name: mymove;

ii)animation-duration   :time|initial|inherit;   [defVal:0]
Specifies how long time an animation should take to complete one cycle
 ex : animation-duration: 3s;

iii)animation-timing-function   [defVal:ease]
:linear|ease|ease-in|ease-out|ease-in-out|step-start|step-end|steps(int,start|end)|cubic-bezier(n,n,n,n);
Specifies the speed curve of the animation
 ex :animation-timing-function :linear

iv)animation-delay   : time|initial|inherit;   [defVal :0]
Specifies a delay for the start of an animation
ex : animation-delay: 2s;

v)animation-iteration-count :number|infinite|initial|inherit;   [defVal:1]
The animation-iteration-count property specifies the number of times an animation should be played.
ex :animation-iteration-count :3;

vi)animation-direction   :normal|reverse|alternate|alternate-reverse|initial|inherit;   [defVal:normal]
The animation-direction property defines whether an animation should be played forwards, backwards or in alternate

cycles.
ex :animation-direction ;reverse;

vii)animation-fill-mode :none|forwards|backwards|both|initial|inherit; [defVal:none]
.The animation-fill-mode property specifies a style for the element when the animation is not playing
(before it starts, after it ends, or both).
.CSS animations do not affect the element before the first keyframe is played or after the last keyframe is played.
The animation-fill-mode property can override this behavior.
ex: animation-fill-mode: forwards;

viii)animation-play-state :paused|running|initial|inherit; [defVal:running]
The animation-play-state property specifies whether the animation is running or paused.
ex :animation-play-state :paused;

Animation short Hand property :
animation: name duration timing-function delay iteration-count direction fill-mode play-state;
ex :animation: example 2s ease 3s 2 reverse forwards running;

27.CSS Multiple Columns
===============
.The CSS multi-column layout allows easy definition of multiple columns of text - just like in newspapers:

i)column-count :number|auto|initial|inherit; [DefVal:auto]
Specifies the number of columns an element should be divided into
ex :column-count :3;

ii) column-gap : length|normal|initial|inherit; [DefVal:normal]
The column-gap property specifies the gap between the columns.
ex ;column-gap :40px;

iii)column-rule-width :medium|thin|thick|length|initial|inherit; [defVal:medium]
Specifies the width of the rule between columns
ex :column-rule-width:medium;

iv)column-rule-style : none|hidden|dotted|dashed|solid|double|groove|ridge|inset|outset|initial|inherit; [defVal:none]

The column-rule-style property specifies the style of the rule between columns.
ex :column-rule-style :dotted;

v)column-rule-color :color|initial|inherit;
The column-rule-color property specifies the color of the rule between columns.
ex :column-rule-color:blue;

vi)column-rule (Short Hand property)
The column-rule property sets the width, style, and color of the rule between columns.

syntax :
column-rule :column-width column-style column-color

v)column-width :auto|length|initial|inherit [defVal:auto]
The column-width property specifies the column width.
.The number of columns will be the minimum number of columns needed to show all the content across the element.

ex :column-width:200px;

vi) columns :auto|column-width column-count|initial|inherit; [defVal:auto auto] (Short Hand property)
The columns property is a shorthand property for:

syntax :
    columns :column-width column-count;
ex : columns:200px 3;

vii)column-span : none|all|initial|inherit; [Defval:none]
The column-span property specifies how many columns an element should span across.
ex :column-span :all;

viii)column-fill :balance|auto|initial|inherit; [defVal:balance]
The column-fill property specifies how to fill columns, balanced or not.
ex :column-fill :auto;

28.CSS Grid
========
The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

.An HTML element becomes a grid container when its display property is set to grid or inline-grid.

```
.grid-container {
  display: grid|inline-grid;
}
```

ex :
```
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto;
  background-color: #2196F3;
  padding: 10px;
}
.grid-item {
  background-color: rgba(255, 255, 255, 0.8);
  border: 1px solid rgba(0, 0, 0, 0.8);
  padding: 20px;
  font-size: 30px;
  text-align: center;
}
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
  <div class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
</div>
```

.All direct children of the grid container(Parent) automatically become grid items.
.The vertical lines of grid items are called columns.
.The horizontal lines of grid items are called rows.
The spaces between each column/row are called gaps.

Parent grid properties :

i)display	:grid|inline-grid;
ii)grid-auto-flow	:row, column, or dense. [defVal:row]
iii)grid-template-rows	:
iv)grid-template-columns	:auto auto auto auto
v) grid-column-gap	:
vi)grid-row-gap	:
vii)grid-gap	:
viii)grid-template-areas	:
ix) justify-content (horizontally)	:start|end|center|space-evenly|space-around|space-between.
x)align-content (vertically)	: start|end|center|space-evenly|space-around|space-between.
xi)grid-auto-rows	
xii)grid-auto-columns	

grid-template-rows :
.The grid-template-columns property defines the number of columns in your grid layout, and it can define the width of each column.
grid-template-columns :auto auto auto auto;

grid-template-rows
.The grid-template-rows property defines the height of each row.
grid-template-rows: 80px 200px;

grid-column-gap
.The grid-column-gap property sets the gap between the columns:
grid-column-gap: 50px;

grid-row-gap
.The grid-row-gap property sets the gap between the rows
grid-column-row: 50px;

grid-gap
.The grid-gap property is a shorthand property for the grid-row-gap and the grid-column-gap properties:
grid-gap :grid-row-gap grid-column-gap ;

justify-content (horizontally):start|end|center|space-evenly|space-around|space-between.
.The justify-content property is used to align the whole grid inside the container.

align-content (vertically) : start|end|center|space-evenly|space-around|space-between.
.The align-content property is used to vertically align the whole grid inside the container.

Child grid properties :
i)grid-row :grid-row-start grid-row-end (ex :2/4)
ii)grid-row-start
iii)grid-row-end
.The grid-row property defines on which row to place an item.
Note: The grid-row property is a shorthand property for the grid-row-start and the grid-row-end properties.

iv)grid-column : grid-column-start grid-column-end (ex :2/4)
v)grid-column-start
vi)grid-column-end
.The grid-column property defines on which column(s) to place an item.
Note: The grid-column property is a shorthand property for the grid-column-start and the grid-column-end propertie
s.

(grid-area short hand property for grid-row-start,grid-column-start,grid-row-end and grid-column-end.)
vii)grid-area :grid-row-start,
 grid-column-start,
 grid-row-end,
 grid-column-end .

Naming Grid Items :

In child item , we have to give like
 grid-area: myArea;

In Parent Container ,we have to give like
 grid-template-areas: 'myArea myArea myArea myArea myArea';
 or
 grid-template-areas: 'myArea myArea myArea myArea .';

.repeat()
grid-template-rows :10px 10px 10px;
 =
 grid-template-rows :repeat(3,10px);
 =
 grid-template-rows :10px repeat(2,10px);

29.CSS Flex :
========
.CSS Flexible Box Module -one dimensionl layout model.
 .flexible and efficient layouts
 .distribute space among items
 .control their alignment.

Before flex box ,there are four layout modes.
 .Block ,for section in a webpage
 .lnline,for text
 .Table,for two-dimensional table data
 .positioned,for explicit position of an element
Note :But these layout modes does not provide flexiblilty

With Flex box :(Why flex box)
.a lot of flexibility
.arrange items
.spacing
.alignment
.order of items
.bootsrap 4 built on top of the flex layout.

Terminology :

```
 <div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
</div>
```

we have two items as
i)Parent or flex-container
ii)Children or flex-items

we have flex-axis,they are
i)Main Axis   (Horizantal)
ii)Cross Axis (Vertical)

```
                  ______________________________  cross start
                 |                                |    |
                 |  item1 item2.....etc           |    |Cross size
                 |________________________|__|___cross end
                 |                                |
                 |---------Main size ------------ -------|
        Main start                          Main end
```

Flex Container properties :
----

i)display	:flex,inline-flex.
ii)flex-direction	:column ,column-reverse,row, row-reverse
iii)flex-wrap	:wrap,nowrap,wrap-reverse.
iv)flex-flow	
v)justify-content	:center,flex-start,flex-end,space-evenly,space-around,space-between.
vi)align-items	:
vii)align-content	

flex-direction   :column ,column-reverse,row, row-reverse   [defVal:row]
.The flex-direction property defines in which direction the container wants to stack the flex items.

flex-wrap   :wrap,nowrap,wrap-reverse.   [defval:nowrap]
.The flex-wrap property specifies whether the flex items should wrap or not.

flex-flow   (Short hand property)
The flex-flow property is a shorthand property for setting both the flex-direction and flex-wrap properties.
flex-flow : flex-direction flex-wrap;
ex : flex-flow: row nowrap;

justify-content   :center,flex-start,flex-end,space-evenly,space-around,space-between (Horizontaly)   [DefVal:flex-start]
The justify-content property is used to align the flex items:
ex : justify-content: center;

align-items   :center,flex-start,flex-end,stretch,baseline   [defVal:stretch]   (Vertically)
The align-items property is used to align the flex items.

ex :align-items :stretch.

align-content : center,flex-start,flex-end,stretch,space-around,space-between [defVal:stretch] (vertically)
The align-content property is used to align the flex lines.

Flex item or children properties:
----
i)order
ii)flex-grow
iii)flex-shrink
iv)flex-basis
v)flex
vi)align-self

order : number [def
Val:0]
The order property specifies the order of the flex items.

flex-grow : number [defVal
:0]
The flex-grow property specifies how much a flex item will grow relative to the rest of the flex items.

flex-shrink : number [defVa
l:1]
The flex-shrink property specifies how much a flex item will shrink relative to the rest of the flex items.

flex-basis : [defVal:auto]

The flex-basis property specifies the initial length of a flex item. (It acts like width,instead of width we can use thse
s)

flex :short hand property
The flex property is a shorthand property for the flex-grow, flex-shrink, and flex-basis properties.
flex :flex-grow flex-shrink flex-basis;
ex :flex :0 1 auto;

align-self :flex-start,flex-end,center; [defVal:auto]
The align-self property specifies the alignment for the selected item inside the flexible container.
The align-self property overrides the default alignment set by the container's align-items property.

30.CSS Miscellaneous like css functions,resize..etc;
================================
i)resize : none|both|horizontal|vertical|initial|inherit; [defVal:none]
The resize property defines if (and how) an element is resizable by the user.

The resize property does not apply to inline elements or to block elements where overflow="visible".
So, make sure that overflow is set to "scroll", "auto", or "hidden".

CSS functions are used as a value for various CSS properties.
i)attr()
The attr() function returns the value of an attribute of the selected elements.
syntax : attr(attribute-name)

ex :
```
a:after {
  content: " (" attr(href) ")";
}
```

ii)calc()
The calc() function performs a calculation to be used as the property value.
 syntax : calc(expression)
 ex :
```
 #div1 {
  position: absolute;
  left: 50px;
  width: calc(100% - 100px);
  border: 1px solid black;
}
```

iii)var()
The var() function is used to insert the value of a CSS variable.
Syntax:var(name, value)

name	Required. The variable name (must start with two dashes)
value	Optional. The fallback value (used if the variable is not found)

ex :
```
:root {
  --main-bg-color: coral;
  --main-txt-color: blue;
  --main-padding: 15px;
}

#div1 {
  background-color: var(--main-bg-color);
  color: var(--main-txt-color);
  padding: var(--main-padding);
}

#div2 {
  background-color: var(--main-bg-color);
  color: var(--main-txt-color);
  padding: var(--main-padding);
}
```

iv)

rgb()	Defines colors using the Red-Green-Blue model (RGB)
rgba()	Defines colors using the Red-Green-Blue-Alpha model (RGBA)
hsl()	Defines colors using the Hue-Saturation-Lightness model (HSL)
hsla()	Defines colors using the Hue-Saturation-Lightness-Alpha model (HSLA)

linear-gradient()	Sets a linear gradient as the background image. Define at least two colors (top to bottom)
radial-gradient()	Sets a radial gradient as the background image. Define at least two colors (center to edges)

repeating-linear-gradient()	Repeats a linear gradient
repeating-radial-gradient()	Repeats a radial gradient