CSS

It stands for cascading style sheet
used to style HTML element
Invented by Hakon Wium Lie on October
10, 1994.

CSS syntax

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

Selectors

Selectors are used to find (or select) the HTML element you want to style.

CSS selectors can be devided into five categories.
1. Simple Selectors (based on name, id, class)
2. Combinator Selectors
3. Pseudo class selectors (based on pseudo class)
4. Pseudo elements selectors (based on pseudo element)
5. Attribute selectors (based on attribute)

Simple Selectors

1 Tag Selectors

```
Tag {                    h1 {
    property value;          text-align: center;
    property : value;        font-size : 12px;
}                        }
```

2 class selectors

```
. class name {           . black {
    property: value;         float : left;
}                        }
```

3 Id / # Selectors

```
# h1 {
    text-align: left;
}
```

4 Universal Selector - select all HTML elements

```
* {
    font-size : 15px;
}
```

3. Grouping Selectors

```
h1, h2, h3, p {
    letter-spacing : 4 em;
    font-weight : normal;
}
```

Combinators Selectors

A combinator is something that explains the relationship between the selectors There are 4 different combinators in CSS -

1. descendant Selector (space)

```
div p {
    background-color: red;
}
```

2. Child Selector (>)

```
div > p {
    background-color : yellow;
}
```

3. Adjacent Sibling Selector (+)
It is used to select an element that is directly after another specific element.

```
div + p {
    background-color : yellow;
}
```

4. General Sibling Selector (~)
It selects all elements that are next siblings of a specified elements

```
div ~ p {
    background-color : red;
}
```

Pseudo class selector
A pseudo class is used to define a special state of an element.
The synax of its selector is-

```
selector : pseudo-class {
    property : value;
}
```

Note- pseudo-class can be hover, visited, link(unvisited), ~~first-letter~~, active, first-child, focus

Pseudo-Elements selector

A CSS pseudo-element is used to style specified part of an element.

* It is used to style the first letter or line of an element
* insert content before or after, the content of an element.

Syntax-

```
selector:: pseudo-element{
    Property: value;
}
```

Note- Pseudo-element may be first-letter, before, after, first-line, markers(select the markers of list), selection (matches the portion of an element that is selected by a user)

Attribute Selector

This selector is used to select element with a specified attribute.

[attribute] Selector
[attribute = "value"]
[attribute |= "value"] - started with value.
[attribute ~= "value"] - select an element having a specified word.
[attribute ^= "value"] - begins with a specified value
[attribute $= "value"] - ends with value.
[attribute *= "value"] - contains specified character.

Three Ways to Insert CSS in HTML

1- External CSS - by using link tag in HTML

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

2- Inline CSS - using style attribute in element

```
<p style="color: red; background-color: blue;
   hi, I am raf
</p>
```

3- Internal CSS - by using style tag under <head> tag with use of selector

```
<style>
  #hi {
      background-color: blue;
       color: red;
      }
</style>
```

CSS Comments

It is placed inside the style tag

```
/* Comment here */
```

CSS Colors

Colors can be defined by their name, RGB, HEX value. Colors are used to define the property -

```
background-color : red ;
color : #ff0000;
border-color : rgb(255,0,0);
```

CSS Backgrounds

To add background effects for elements -
The various background properties are -

```
background-size : cover;
background-color :
background-image : url("image.jpg");
background-repeat : repeat-x;
background-position : right;
background-attachment : fixed or scroll;
```

background (short hand property)

- repeat-x ;
- repeat-y ;
- no-repeat ;

Is used to specify the position of background image

Specify whether the background image should be scroll or be fixed

```
background : #ffffff url("image.jpg") no-repeat
             fixed right;
```

It should be in order.

CSS Borders

It allow you to specify the style, width and color of an element's border.
Various border's properties are -

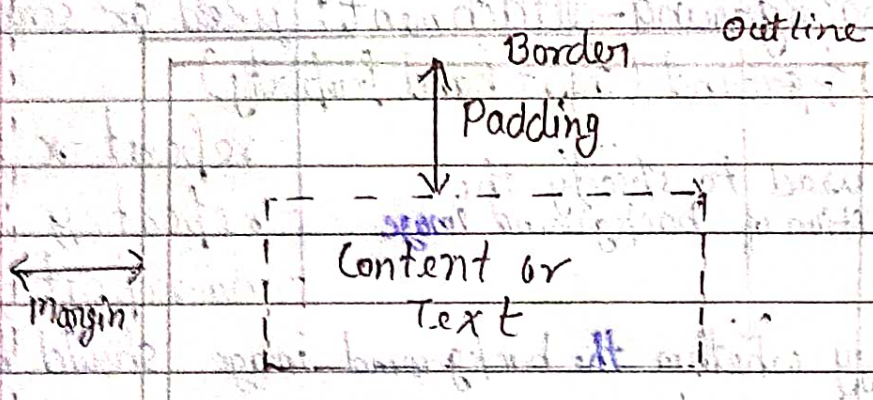
```
border-style: solid;
border-width: 5px;
border-color: red;
border: 5px solid black;
```

shorthand property (It should be in order width, style & color)

```
border-radius: 5px;
```

used to add rounded border

CSS Box Model



CSS Text

CSS has lot of properties for formatting text.

- color (to add color in text)
- text-align (used to set horizontal alignment of text)

vertical-align

direction : rtl;

text-decoration : none; (underline, overline, line-through) (used to remove underline from links)

- text-transform : uppercase; (lowercase, capitalize)

capitalize the first letter of each word.

- text-indent : 50px; (used to specify the indentation of the first line of a text)
- letter-spacing : 3px;
- word-spacing : 5px;

line-height : 2px; (used to specify the space bet^n lines)

white-space : nowrap;

text-shadow : 2px 2px red;

CSS Opacity (अपारदर्शिता, अस्पष्टता)

Opacity is the condition of lacking transparency. Its value is between 0 and 1

opacity : 0.15;

CSS Fonts

Font means the particular shape and style of a set of letters that are used in printing, on a computer screen, etc. Various font-properties are -

* font-family (used to change the face of a font)
  font-family: Serif; (Arial, Verdana)
* font-style: italic; (normal, oblique)
* font-weight: bold; (normal, 40px, lighter)
* font-variant: small-caps; (normal)
* font (shorthand property)

CSS Links

We add colors, padding, background-color, text align etc by using pseudo class. as -

- a: link
- a: visited
- a: hover
- a: active

CSS List

* list-style-type
  list-style image
  marker-offset (specifies the distance between marker & text)
  list-style : square inside url("image.gif");

* ul li : : markers {
    background-color : red;
    color : black;
    list-style : circle inside url('smiley.gif');
  }

CSS tables

To specify the table borders in CSS, we use border property.

```
table, th, td {
    width border : 1px solid black;
    border-collapse : collapse;
    text-align : center;
    padding : 16px;
    color : white
}
th { background-color : green }
```

CSS Responsive Table

A responsive table will display a horizontal scroll bar if the screen is too small to display the full content.
Add a container element ( ) with overflow-x: auto;

CSS Display

- none - to hide the element
- block
- inline

block value is used to list the items inline. (in a line)

CSS Position

static - element will move with flow of page
fixed - element will not move with " "
sticky -
relative -
absolute - positioned relative to nearest positioned ancestors

```
div {
  position: fixed;
  left: 0px; right: 5px; top: 10px; bottom: 0p
  border: 3px solid green;
}
```

Overlapping elements

When elements are positioned, they can overlap with within elements.
The z-index property specifies the stack order of an element

```
z-index : -1 ;   (it will place behind the text)
z-index : 1 ;    (  "  "  "  front of - "- )
```

Overflow property

When the content is too large, then it is overflowing from the border. so, we use overflow property to controls the content.

```
overflow : visible;
         : hidden;  (It makes the content clipped with the box)
         : scroll;
         : auto;  Similar to scroll, but it adds scrollbars only when necessary

overflow-x : auto;
overflow-y : auto;
```

Float property

- left
- right
- none
- inherit

clear property

When we use float property and we want the next element below (not right or left), we have to use the clear property

clear : ;
- left
- right
- both
- inherit
- none

The clearfix Hack

When the floating element is larger than the containing element, it will overflow outside of its container. We can then add a clearfix hack to solve this problem

```
div.hi {
    overflow: auto;
}
```

or

```
div.hi::after {
    content: " ";
    clear: both;
    display: block;
}
```

Horizontally Alignment (Alignment to Centre)

* To align the border in centre we use margin: auto;
* To align the text, we use text-align: center;
* One method is also used for alignment by position: absolute; left: 4px; right: 0px; top: 40px; bottom: 40px;
* One method is float to align the element in left or right.

Vertical Alignment

* Using Padding, we can align the element in centre.
  padding: centre;
* Using line-height: height;
* Using position property with top and bottom value

Navigation bars (list of links)

* It is created as <ul> and <li> with <a>.
* Now, removing the markers and margins and padding from the list.

```
li a {
    display: block;
    list-style-type: none;
    margin: 0;
    padding: 0;
}
```

It will create vertical navigation bars.

* To obtain horizontal navigation menu, we use float: left; or by display: inline;

Responsive Top navigation

By using

```
@ media screen and (max-width: 600px)
{
  ul.topnav li {
                float: none;
               }
}
```

Responsive Side Navigation

```
@ media screen and (max-width: 900px)
{
  ul.sidenav { width: 100%;
               height: auto;
               position: relative;
             }
  ul.sidenav li a {
                   float: left;
                   padding: 15px;
                  }
  div.content { margin-left: 0; }
}
```

CSS Counters

These are variables maintained by CSS.

Counter-reset

counter-increment

content

counter( ) or counters( )

```
body {
    counter-reset: section;
}
h2::before {
    counter-increment: section;
    content: "section" counter(section) ": ";
}
```

(counter → function)

Nesting Counters-

```
body {
    counter-reset: section;
}
h1 {
    counter-reset: subsection;
}
h1::before {
    counter-increment: section;
    content: "Section " counter(section) ": ";
}
h2::before {
    counter-increment: subsection;
    content: counter(section) "." counter(subsection) " ";
}
```

Specificity

If there are 2 or more CSS rules that match same element, a CSS rule will apply on the element whose specificity is high.
It is like a rank that determine which style declarations are ultimately applied to an element.

* selector has low specificity
- specificity = 1000 for style attribute
- " = 100 for each id
- " = 10 for each attribute, class or pseudoclass
- " = 1 for element name or pseudo-element

!important

The !important rule in CSS is used to add more importance to a property value than normal
Ex background-color: grey !important;

CSS Gradients

It let you display smooth transition between two or more specified colors
* Linear gradients
* Radial gradients

Linear gradients

background-image : linear-gradient (direction, color 1, color 2, ...);

- to right
- to right bottom
- using angles as
  - 180 deg
  - 90 deg
- to left.

Radial Gradients

background-image : radial-gradient (shape size at position, color 1, color 2, ...);

- circle
- ellipse
- closest-side at 60% 55%
- farthest-side at 60% 55%

Shadow

text-shadow

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

(5px → blur)

Transform

It allow you to move, rotate, scale and skew elements

* 2D transform
* 3D transform

2D transform

```
transform:
    translate (50px, 100px);
    rotate (20 deg);
    scale (2, 3)
    scaleX (2);
    scaleY (3);
    skew (20deg, 10deg);
```

3D transform

```
transform:
    rotateX (150 deg);
    rotateY (150 deg);
    rotateZ (90 deg);
    rotate3d (x, y, z, angle);
```

Date
Page

generally used to change width height when an event occur.

Transition

It allow you to change property values smoothly, over a given duration.

Use of CSS transition

To create it, you must specify two things -

* the CSS property you want to add an effect to
* the duration of the effect

Ex-
```
div { width : 100px; height : 150px;
    transition : width 2s , height 2s ;
    }
div : hover {
        width : 300px; height : 200px;
    }
```

* transition-delay : ;
* transition-duration : ;
* transition-property : ;
* transition-timing-function : ;

used for making speed of transition

- ease
- linear
- ease-in
- ease-out
- ease-in-out
- ease-out

Animations (change in style)

It allows animation of HTML element without using js and flash. It is a method in which figure are manipulated to appear as moving images.

Ex- #div {
animation-name : example;
animation-duration : 5s;
}

animation-name

@keyframes example {
from { background-color: red; }
to { background-color: black; }
}

(2) #div {
animation-name : Raj;
animation-duration: 4s;
}

@keyframes Raj {
0% { background-color: red;
color : black;
}
25% { background-color: blue;
color: white;
}
75% { background-color: yellow;
left: 25px; bottom: 25px;
}
100% { background: grey; }
}

@keyframes
animation-name;
animation-delay;
animation-duration; ; it can also be in minute.
animation-iteration-count;
animation-timing-function; speed
animation-direction;
animation-fill-mode;

- none
- forwards
- backwards

- normal (as forward)
- reverse
- alternate
- alternate-reverse

* animation (shorthand property)
animation: example 4s ease 2s 3 reverse;
duration, ease = speed, 2s = delay, 3 = count, reverse = dir.

Some property used in CSS

* border-radius — also used in image to make it circular, elliptical etc.
* box-reflect — used to creat image reflection. Its value can be left, right, below, above.
* box-sizing: border-box;

Flexbox

It is made for one-dimensional layouts and the grid is made for two-dimensional layouts. It means flex box can work on either rows or columns at a time, but grids can work on both. Flex box takes a basis in the content while grid takes a basis in layout.

Html –

flex container – `<div class = "flex-container">`

flex items:

```
<div> 1 </div>
<div> 2 </div>
<div> 3 </div>
```

`</div>`

css –

```
.flex-container {
        display: flex;
    background-color: dogdeblue;
        }
.flex-container > div {
        background-color: grey;
        margin : 10px 5px;
        padding : 20px;
        font size : 30px;
        }
```

Flex-container property-

flex-direction : column; (row or row-reverse)
flex-wrap : wrap or nowrap;
flex-flow : column wrap;
perfect centering [justify content : center or flex-start or flex-end;
align-items
align-content

Flex-item property -

order
flex-grow
flex-shrink
flex:basis
align-self

Flex box Responsiveness

```
.flex-container {
        display: flex;
        flex-direction: row;
        }
@ media screen and (max-width: 800px)
  {     .flex container {
                flex-direction: column;
                }
}
```

Grid

```
<div class="grid-container">
    <div> 1 </div>
    <div> 2 </div>
    <div> 3 </div>
    <div> 4 </div>
    <div> 5 </div>
    <div> 6 </div>
</div>
```

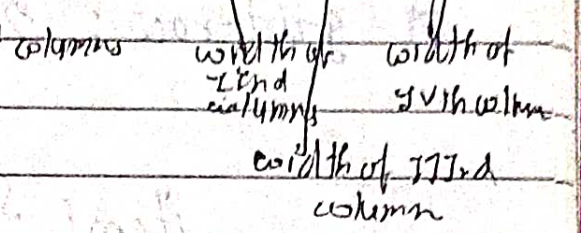
```
.grid-container{
        display: grid;
        background-color: blue;
grid-template-columns: auto auto auto;
}
.grid-container > p {
        background-color: yellow;
        border : 1px solid black;
        padding : 20px;
        font size : 30px;
        text-align : center;
    }
```

Grid-container:

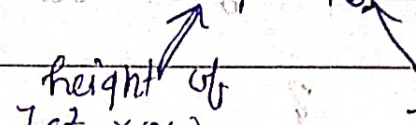
- display : grid;
- grid-template-columns: auto auto ;
  " no. of auto used is 2. Thus, columns in your grid container will be 2.

grid-template-columns: auto auto auto auto;
It shows you used 4 columns.

grid-template-columns: 80px 200px auto 90px;
It shows width of columns respectively.



- grid-template-rows: 80px 190px;



- grid-gap: 10px;
- justify-content: ← (horizontally align)
  - center
  - start
  - end
  - space-between
  - space-around
  - space-evenly
- align-content: ← (vertically align)
- height

Grid - Item (child of grid-container)

* grid-column: 1 / 5;
  start on column 1 and end before column 5
* grid-column: 1 / span 2;
  start from column 1 and span 2 columns
* grid-row: 1 / 4;
* grid-row: 2 / span 2;

Ex.

```
.item1 {
    grid-row: 2 / span 3;
}
```

* grid-area: 1 / 2 / 5 / 6;
  1 - row started
  2 - column started
  5 - row end before
  6 - column ended before
* grid-area: 3 / 5 / span 2 / span 3;