Web Development

→HTML is used to create Layout of the webpage. By using HTML we can't decide what content we need to display HTML Layout.

→Css is used to apply styles for HTML Layout's.

→Bootstrap is Library of HTML and Css; Bootstrap is used to create fast responsive websites.

→JavaScript is client side programming language which is used to create dynamic functionality on (HTML) webpages.

Tag

The text which surrounded between angular brackets is known as tag.

<text> -- opening tag
</text> -- closing tag

Element

The combination of content, and opening tag and closing tag is called element.

(or)

The content which is enclosed with in the opening and closing tag is known as element.

<text>
Content
</text>

HTML

→ HTML stands for Hyper Text Markup language.

Hyper Text

Hyper text is also known as Link (or) Hyper link.

* Hyper text is a text which contains references of new web page.

Markup language

The language/(lang) which the programs are written by using tags is known as Markup language.

Ex:- XML (Extensable Markup language)

demo.html

```
<html>
    <head>
        <title> New Tab </title>
    </head>
    <body>
        <h1> happy html☺ </h1>
    </body>
</html>
```

HTML attribute

→ By using Html attribute we can't provided to addition functionality HTML element.

Syntax

```
<tag Name attName = "value">
    ==
    ==
</tag Name>
```

Align

Align is an attribute

Align has 3 values:

left, centre, right

The default is left

Ex:-

```
<h1 align = "right"> happy html1 ☺ </h1>
```

Note:-

* we can use multiple attributes for HTML elements.

* In HTML we cannot user defined tags.

HTML DocType declaration

* The Doctype declaration is used to represent the version of HTML.
* The latest version of HTML is HTML 5.0
* In HTML 5.0 version, the DocType declaration is simple.

```
<!DOCTYPE html>
```

```
<!DOCTYPE html>
<html>
    <head>
        <title> New Tab </title>
    </head>
    <body>
        <h1 align = "Centre"> Happy html ☺ </h1>
    </body>
</html>
```

```
<! DOC TYPE html>
<html lang="en">
<head>
  <title> Document </title>
</head>
<body>
  <marque scroll="80px" scroll delay="
   direction="right" behaviour="alternate"
   LOOP="4">
  <h1> Happy HTML ☺ </h1>
  </marque>
</body>
</html>
```

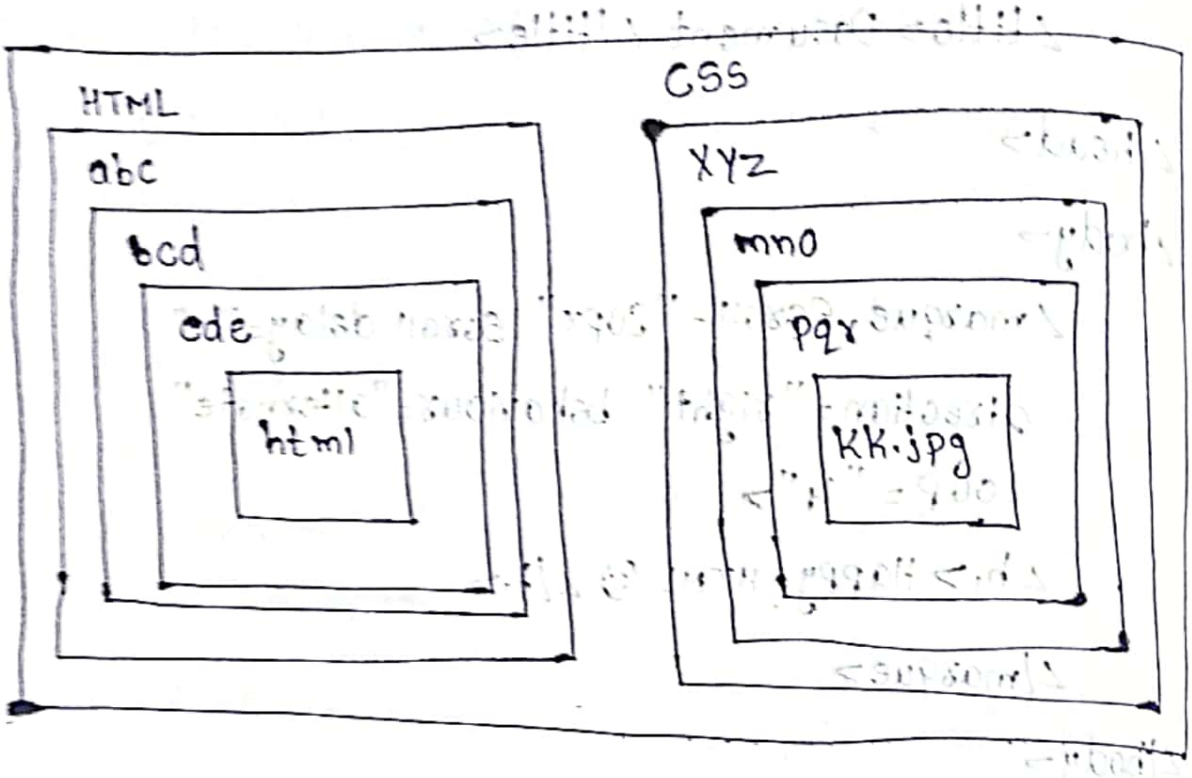
Ex2:-

```
<! DOC TYPE html>
<html lang="en">
<head>
<title> Document </title>
</head>
<body>
    <h1> Image Handling:- </h1>
    <Img src=" Link in go " width="300px"
    height="250px" alt="Image Not found"/>
</body>
</html>
```

Image path

<img src = "../../../../css/xyz/mno/pqr/KK.jpg">

WBD_A7



Hypertext:-

* Hypertext is a text which contain the reference of new webpage.
* To create hypertext we need to use "anker" tag.

 Shop IN Amazon

 Shop IN' Amazon

* CSS is used to apply styles for Html layout

* There are 3 ways to apply styles for any html element.

1) Inline Style Sheet
2) Internal Style Sheet
3) External Style Sheet.

Inline Style Sheet

In this style sheet we can write css styles within tag itself.

To write css properties within the tag, we need to use style attribute.

Syntax

```
<tagName style="key:value;">
    content
</tagName>
```

Ex:

```
<!DocType html>
<html lang="en">
<head>
    <title>Document</title>
</head>
<body style="background-color:yellow;">
    <h1 style="color:red;">Keerthi Is Pretty Girl</h1>
</body>
</html>
```

2) Internal style sheet

* In this style sheet we need to write Css styles within the html document.
* To write styles within the document we have to use style tag.
* The recommended place is to write style tag is within head tag.

Syntax

```
<style>
    select the tag
    {
        key:value;
    }
</style>
```

Ex:-

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title> Document </title>
    <style>
        body
        {
            background-color: cyan;
        }
        h1
        {
            color: blue;
        }
    </style>
</head>
```

```
</body>
    <h1>Keerthi Is pretty Girl</h1>
</body>
</html>
```

External Style Sheet

* In this Style-Sheet we need to write css styles outside of html document.
* The external Style sheet must be saved with '.css' extension.
* one external style sheet can be connected for any number of html files.
* To connect css file to html file we have to use link tag.

Syntax:-

```
<link rel="stylesheet" href="css file address"/>
```

filename.css

```
select the tag
{
  key: value;
}
```

Ex:-

```
<link rel="stylesheet" href="../HTML/one.css"/>
<link rel="stylesheet" href="two.css"/>
```

```
body
{
  background-color: □color
}
```

```
h1
{
    color: blue;
}
```

Css Selectors

By using Css selectors we can apply the styles for group of elements.

TagName Selector

Applying the styles with the help of tagname we can called it as Tagname selector.

Syntax

```
TagName
{
    property: value;
}
```

EX:-

```
<title>css</title>
<style>
    h1
    {
        background-color: aqua;
    }
    h2
    {
        background-color: tomato;
    }
</style>
```

Class selector

* By using class selector we can apply css styles for group of elements.
* Group selector is represented by using . operator.

Syntax:-

```
.className
 {
   property: value;
 }
```

Ex:-

```
<title>class selector</title>
<style>
   .abc
    {
       background-color: black;
        color : white;
    }
   .mno
    {
       background-color : orange;
        color : green;
    }
</style>
</head>
<body>
   <h1 class="xyz abc">class selector</h1>
   <h1 class="xyz">class selector</h1>
   <h1 class="mno">class selector</h1>
   <h1 class="mno">class selector</h1>
   <h1 class selector</h1>
   <h1 class selector</h1>
```

</body>
</html>

Id selector

* By using Id selector we can apply unique styles for the html elements.
* Id selector is represented by using # operator.

Syntax

```
#idname
 {
   property: value;
 }
```

Ex:-

```
<style>
  #one
  {
    background-color: □
    Color: □ white;
  }
  #two
  {
    background-color: □ brown;
    color: □ white;
  }
</style>
</head>
<body>
  <h1 id="one">Id selector</h1>
  <h1 id="two">Id selector</h1>
  <h1>Id selector</h1>
</body>
```

Attribute selector

* Applying the styles with the help of attributes, we can called it as Attribute selector.

*Attribute selector is represented by using '[ ]'.

Ex:-

```
<title>Attribute Selector</title>
<style>
    h2 [align="left"]
    {
      color: red;
    }
    [align="center"]
    {
      color: blue;
    }
    [align="right"]
    {
      color: forestgreen;
    }
</Style>
</head>
<body>
  <h1 align="left"> Heading one</h1>
  <h1 align="center"> Heading two</h1>
  <h1 align="right"> Heading Three</h1>
  <h2 align="left"> Heading one</h2>
  <h2 align="center"> Heading two</h2>
  <h2 align="right"> Heading three</h2>
</body>
</html>
```

Universal Selector

* By using universal selector we can apply common styles for all the html elements.
* universal selector is represented by using *

Ex:-

```
*
{
    margin:0px;
}
```

Group selector

*The combination of more than one selector is called Group selector.
* Group selector is represented by using comma operator.

Syntax

```
selector1, selector2, selector3.... selectorn
{
    property:value;
}
```

Pseudo class Selector

* Pseudo class selector is used to apply the styles entire html element.
* Pseudo class selector is represented by using :
*we have predefined values like:

1) hover
2) link
3) active
4) visited

Ex:-

```
<style>
    h1:hover
    {
        background-color: blue;
        color: white;
    }
</style>
</head>
<body>
    <h1> Jai Balayya</h1>
</body>
</html>
```

Pseudo element

* By using pseudo element selector we can apply the styles for specific part of the html element.
* Pseudo element selector is represented by using ::

Ex:-

```
<style>
    #one
    {
        color: orange;
    }
    h2::first-letter
    {
```

```
        color: orange;
    }
    p::selection{
        background-color: green;
        color: white;
    }
    p::before
    {
        content: "This is uday";
    }
    p::after
    {
        content: "This is uday";
    }
    p::firstline
    {
        background-color: orange;
    }
</style>
</head>
<body>
    <h1><span id="one">J</span>spiderss</h1>
    <h2>Jspiderss</h2>
    <p>Lorem</p>
</body>
</html>
```

* Lorem is used to dummy paragraph.

Block level Elements

The elements which occupies 100% width of the webpage those element will be called it as block level elements.

Ex:-

All heading tags, nav tag, header tag, dev tag, Paragraph tag

Inline elements

The elements which occupies content level space those elements we can called it as Inline elements.

Ex:-

span tag, anker tag, image tag, input tag, label tag, Button tag.

Combinators

Combinators are also a type of selectors.

whenever there is a parent and child relationship then we should go for combinators.

There are 4 types of combinators.

1) child selector
2) descendent selector
3) adjacent selector
4) General sibbling selector

Child selector

* By using child selector... we can apply the styles for directly childs but not for indirect childs
* child selector is represented by using > operator

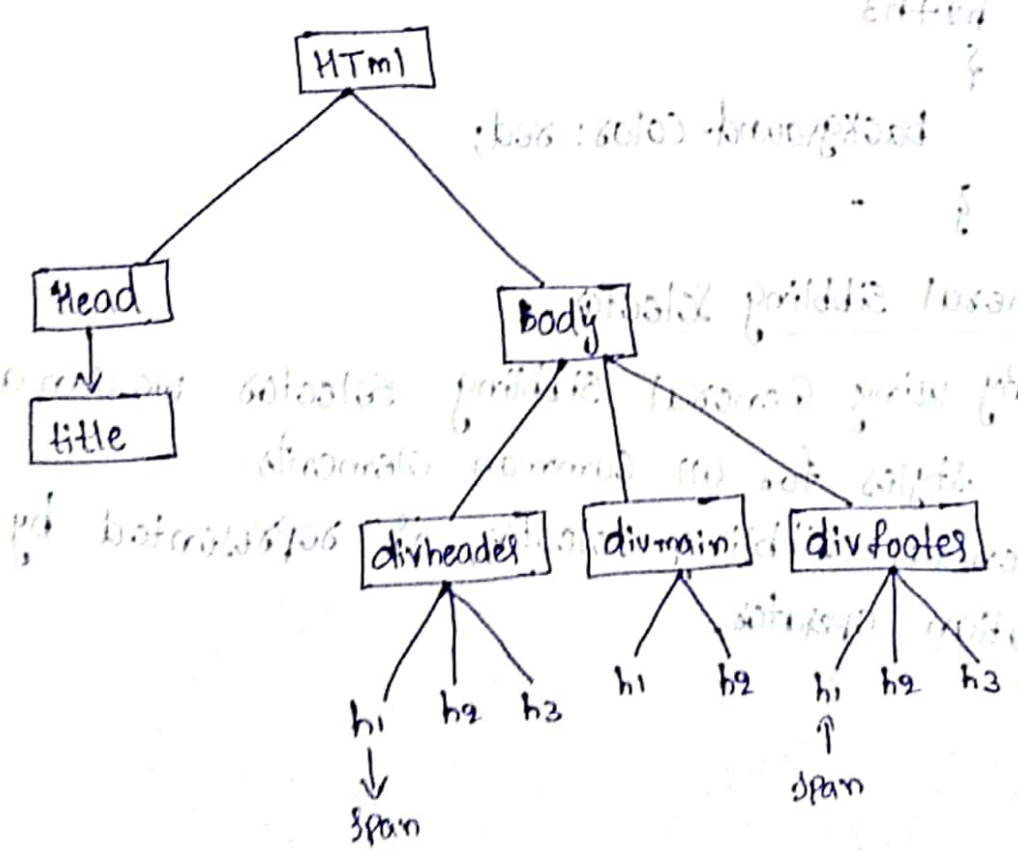
Syntax

```
Parent selector > child selector
{
    property:value;
}
```

Ex:

```
<style>
    .header>h1
    {
        background-color: red;
    }
    .header>h1>span
    {
        background-color: blue;
    }
</style>
</head>
<body>
    <div class="header">
        <h1> Heading one</span>spanTag</span></h1>
        <h2> Heading Two </h2>
```

```
    <div><h1> Jspiders</h1></div>
    <h3>Heading Three</h3>
</div>
<hr>
<div class="main">
    <h1>Heading one</h1>
    <h2>Heading two</h2>
</div>
<hr>
<div class="footer">
    <h1>Heading one <span>spanTag</span></h1>
    <h2>Heading two</h2>
    <h3>Heading Three</h3>
</div>
```



Descendent selector

* By using descendent selector we can apply the styles for direct elements indirect to the parent element.
* Descendent selector is represented by using space.

Ex:-

```
.header h1
{
  background-color: red;
}
```

Adjacent selector

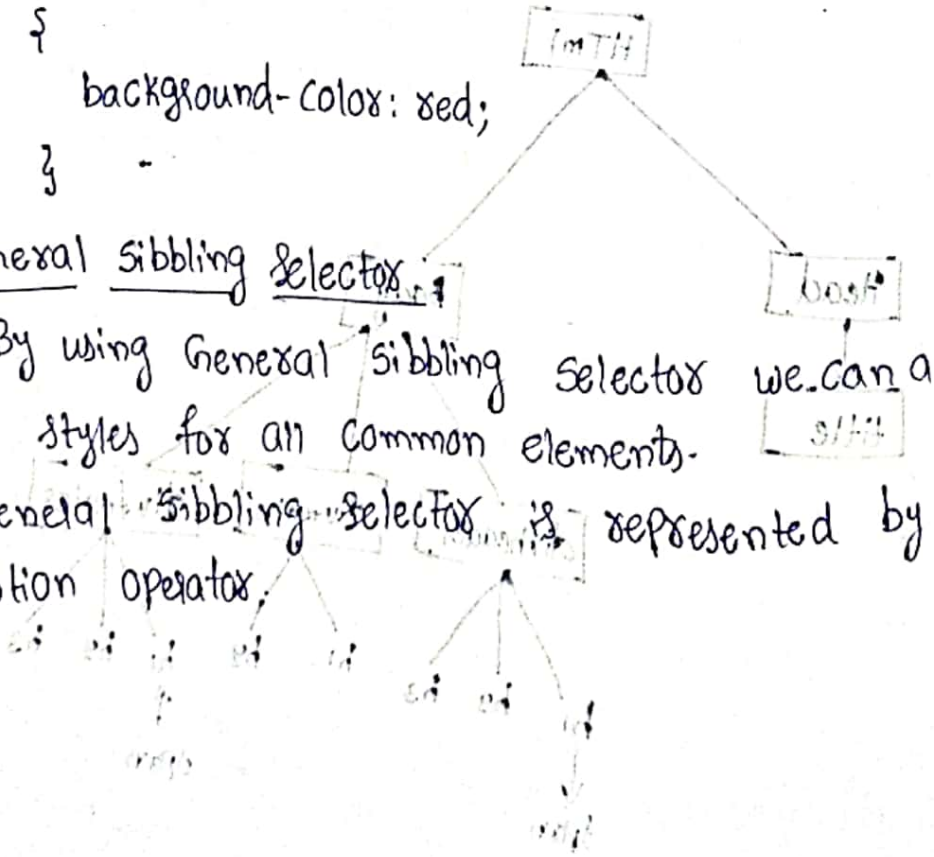
* By using adjacent selector we can apply the styles for adjacent elements.
* Adjacent selector is represented by using + operator.

Ex:-

```
h2+h3
{
  background-color: red;
}
```

General sibbling selector

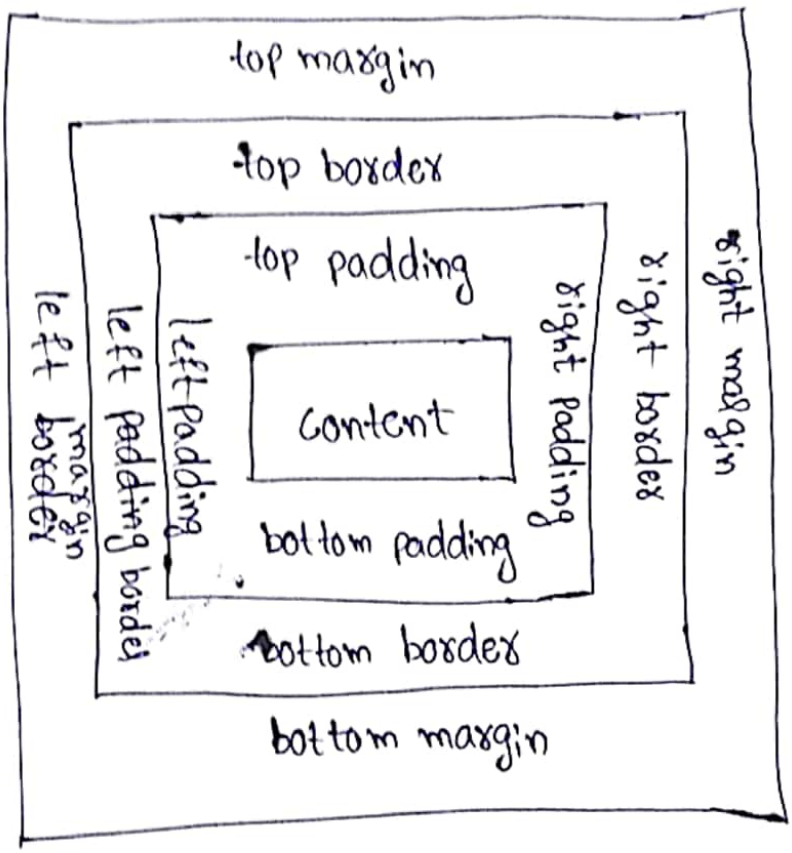
* By using General sibbling selector we can apply the styles for all common elements.
* General sibbling selector is represented by using negation operator.



Ex:-

```
h2~h3
{
    background-color: red;
}
```

CSS BOX model



To create a table we need to use

- <table>
- <thead> table heading
- <tbody> table body
- <tr> table row
- <td> table data
- <th> table heading data

Ex:-

```
<style>
h1
{
  background-color: black;
  color: yellow;
  text-align: center;
  width: 200px;
  /* Padding-top: 50px;
     Padding-right: 50px;
     Padding-bottom: 50px;
     Padding-left: 50px; */
     Padding: 50px 100px 20px 200px;
}
</style>
```

Ex:-

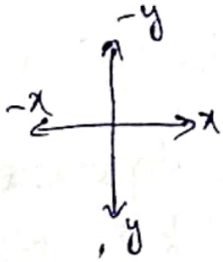
```
<style>
h1
{
    Padding: 50px;
    /* border-top-width: 10px;
    border-top-style: solid;
    border-top-color: yellow; */
    /* border-top: 5px dotted yellow;
    border-right: 5px dotted yellow;
    border-bottom: 5px dotted yellow;
    border-left: 5px dotted yellow; */
    border: 5px dotted aqua;
```

```
/* border-right-width: 10px;
border-right-style: solid;
border-right-color: aqua; */
margin: 200px auto;
}
</style>
```

box-shadow

Syntax

```
box-shadow: x-axis y-axis ratio color;
```



Ex:-

```
<style>
h1
{
/* border-top-left-radius: 30px;
border-top-right-radius: 30px;
border-bottom-left-radius: 30px;
border-bottom-right-radius: 30px; */
border-bottom-left-radius: 30px;
border-radius: 0px 40px;
box-shadow: 0px 0px 90px red;
}
h1:hover
{
background-color: orange;
color: brown;
}
```

table
thead
tbody
tr
td
th

Background properties

Linear gradient is used to apply multiple color.

background-image: linear-gradient(45deg, black, orange, black);

background-repeat: repeat-x | repeat-y | no-repeat;

background-image: url();

background-position: top left, top right, center left, center right, center center, bottom left, bottom right.

background-attachment: scroll/fixed;

background-size: cover;

Display-inline

If we make display-inline the elements will arrange of the element will get side by side but height and width properties will not work.

display:inline;

Display-inlineblock

If we make display: inline-block the elements will arrange side by side with include height and width properties.

display: inline-block;

Display-block

If we want to make the elements line by line then we can make it as display block.

when if you make display block the width it occupies 100% width of the webpage.

Ex:-

```
<style>
    a
    {
        display: block;
        background-color: orange;
    }
</style>
</head>
<body>
```

```
<div class="container">
    <a href="" >Home</a>
    <a href="" >placements</a>
    <a href="" >Courses</a>
    <a href="" >contacts</a>
    <a href="" >Enquiry</a>
```

display: none

If I use ~~none~~ make display none property we won't display any on the webpage.

Css positions

By using css position we can change position of an html element.

There are 5 types of css positions

1) Static position
2) fixed position
3) relative position
4) sticky position
5) absolute position

Static position

* If we make the position as static we can't change the position of an html element but we can scroll.

Note

The default position of every element is static.

Ex:-

```
<style>
  h1
  {
    left:200px;
    top:100px;
    position:static;
  }
  p
  {
    line-height:30px;
  }
</style>
```

Fixed position

If we make the position as fixed we can change the position of an html ~~text~~ element but we can't scroll fixed values.

Ex:-

```
<style>
  h1
  {
    top:100px;
    left:200px;
  }
```

Relative position

* If we make the position as relative we can change the position of an html element as well as scroll relative values.

* Whenever if we use relative position the values it takes from original position but not from top of the webpage.

Ex:-

```
<style>
  .one
  {
    top:100px;
    left:100px;
    position:relative;
  }
```

Sticky Position

If we use the position has sticky it sticks to the given values.

Ex:-

```
<style>
  p
  {
    line-height:50px;
  }
  h1
  {
    background-color:red;
    color:white;
    text-align:center;
    padding:20px;
    left:0px;
    top:0px;
    position:sticky;
  }
```

Absolute position

* Absolute position always its depends on parent relative position.
* If parent position is not relative then it's checks for grand parent position.

Even if grandparent position is not relative then it takes values from top of the webpage.

* If the parent position is relative then it takes values from parent.

Ex:-

```
<style>
  .child
  {
    width: 100px;
    height: 100px;
    background-color: blue;
    top: 200px;
    position: absolute;
  }
  .parent
  {
    width: 100px;
    height: 100px;
    background-color: yellow;
    padding: 40px;
    border: 2px solid black;
  }
  .gp
  {
    width: 200px;
```

```
height:200px;
background-color: black;
padding: 40px 50px 40px 70px;
}
</style>
```

Text-Properties

1) text-align: start color end justify;
2) letter-spacing: 10px;
3) word-spacing: 10 px;
4) line-height: 50 px;
5) text-shadow : X-axis y-axis ratio color;
6) text-decoration-width :-
   color :-
   style :- solid dotted doubled double
7) text-indent : 50px;

flex

```
display : flex;
flex-direction: row column row-reverse column-reverse;
column-gap: 50 px;
row-gap: 20px;
flex-wrap : nowrap wrap
flex-flow: direction wrap;
justify-content: start center end space-around
                 space-evenly space-between
align-content:
```

gap: row column;

gap: 50px;

Flex

flex is used to create one-dimensional layout. which means we can arrange the elements in row-wise (or) column-wise.

Item properties

order property

* By using order property we can change the order of items.
* The default value of every order property is zero.

flex-grow property

If you want to occupy the space which is present in container then we should go for flex-grow property.

The default value of flex-grow property is zero.

align-self: start end center end

flex-shrink

By using flex-shrink property we can reduce the width of the items.

* The default value of flex-shrink in case 1.

Grid

Grid is nothing but it is used to create two dimensional layout which means we can allange the element in row-wise (or) column wise.

```
display: Grid;
grid-column-gap: 100px;
grid-row-gap: 10px;
gap: 20px;
grid-template-columns: repeat(3, auto);
grid-template-rows: auto;
grid-column-start: ;
grid-column-end: ;
grid-row-start: ;
grid-row-end: ;
```

Transform

By using Transforms we can change html element of the state of an html element

```
transition-property:
transition-duration:
transition-timing translate:
transform: translate X:
transform: translate Y
transform: rotate
```

transform: rotate x

transform: rotate y

transform : scale*(1.5)

Javascript

* Javascript was developed by "Brendon Eich" in the year of 1994 at Netscape Navigor Company.
* Initially Javascript was developed for creating dynamic web pages and more interact to user.
* By using Javascript we can perform client side validations.
* By using Javascript we can manipulate html elements.
* Initially Javascript is client-side programming language but modern Javascript can be used for client-side programming and server-side programming.
* The modern Javascript is known as ECMA script-ES6
* ECMA stands for European computer manufacturing Association.
* Javascript is high level programming language (which means user understandable language).

ex:-

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Document</title>
    <script>
        console.log("Hello world!!");
    </script>
</head>
<body>
    <h1>Js Basics</h1>
</body>
</html>
```

Ex:-

```
<script src="external.js"></script>
</head>
<body>
    <h1>js external file</h1>
</body>
</html>
```

folder create

external.js

```
Console.log("This is external file");
```

*To execute Javascript code inside browser, Javascript file embed with html page.

*we can attach Javascript code to html page in two ways

1) Internal Js
2) External Js

Ex:-

```
<script>
    Console.log("Hi");
    Console.log("Hello")
    Console.log("Bye");
</script>
```

*Javascript is implicit semicolonad programming language.

*Javascript is multiparadigm(approach) programming language. (we can write procedural oriented programming language and object oriented programming language.

local static non-static variable

Ex:-

```
<script>
    function main()
    {
        console.log("main starts");
        m1();
        console.log("main ends");
    }
    main();
    function m1()
    {
        console.log("m1 starts");
        console.log("m1 ends");
    }
</script>
```

Functions

* Function is a set of statements which is used to perform specific task.
* To execute function statements function call is compulsory.

Syntax:-

```
function fun_name()
{
    Statements
}
fun_name();
```

Note:-

In Javascript a function can be called before its declaration.

Javascript is case-sensitive programming language.

Identifier

Identifiers are the names which are provided by user (programmer).

Ex:- function names, variable names, class names etc....

Rules

* Identifier should not starts with digits.
* Identifier should not contains any special characters except $ and _
* Keywords (Reserved words) cannot be used for identifier.

Examples

valid

demo demo1 num_1 $num_1

Invalid

1demo @num1 ~~$num~~ num 1 else if ~~for~~ while

```
function main()
{
 console.log("This is log msg");
 console.log("This is warn msg")
 Console.log("This is error msg")
}
```

Variables

Variable is a container which is used to store a single value.

Syntax :-

```
Keyword varname; //Declaration
varname = value; //Initialization
Keyword varname = value;
```

* To create a variable we can use three keywords

1) var
2) let
3) const

Ex:-

```
<script>
    main();
    function main()
    {
        let a; //Declaration
        a=10; //Initialization
        console.log(a);
        //let a;
        a=100; //Reinitialization
        console.log(a);
    }
</script>
```

*Variable redeclaration is not possible but re-initialization is possible.

*Variables are temporary memory.

*In Javascript variables can be created without using any keyword (bad practice)

Ex:-

```
main();
function main()
{
    a=10;
    console.log(a);
}
```

Ex:-

```
<script>
main();
function main()
{
    let a; //undefined
    console.log(a);
    a=10;
    console.log(a);
}
</script>
```

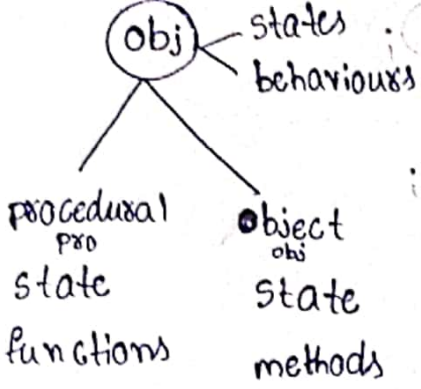
```
Const a;
a=10; //Syntax error
console.log(a);
```

Const val must be initialized at declaration only.

* In Js if we create a variable without initialization then the default value is given by Js engine.
* In Js the default value is undefined.

Ex:-

```
StudentDetails();
function StudentDetails()
{
    let sname, sage, semail, sno;
    sname="Kumar";
    sage=24;
    semail="Kumar@gmail.com";
    sno=9030925131;
    console.log(sname, sage, semail, sno);
    console.log(sname);
    console.log(typeof sname);
    console.log(typeof sage);
}
```



let a;
Console.log(a);

2 types

local variable

local variables are used for creating inside the function space.

Global variable

Global variables are used for creating outside the function and inside the function.

* Javascript is Dynamic typed programming language.

what is Dynamic type Programming language?

* In Dynamic type programming languages datatypes are not used at the time of variable declaration.

* In Dynamic type programming languages data types are assigned to a variable at runtime by Javascript Engine.

Ex:-

```
main();
function main()
{
    let a;
    console.log(a);
    a=10;
    Console.log(a, typeof a);
    a="mahesh";
```

```
Console.log(a, typeof a);
a = true;
Console.log(a, typeof a);
```

Datatypes

Datatypes are used to represent the type of value which we assigned to a variable.

There are two types of datatypes

1) Primitive
2) Non-primitive

There are 7 primitive types

1) Number
2) String
3) boolean
4) undefined
5) Null
6) BigInt (ES11)
7) Symbol (ES6)

Non primitive has derived types

1) Object
2) Arrays
3) Class
4) Reg Exp

Number

Number type is used to store any numeric value (which can be decimal or non-decimal).

Ex:-

```
<script>
    let a=10;
    console.log(a,typeof a);
    a= 10.07;
    console.log(a,typeof a);
</script>
```

string

string type is used to store a single letter or group of letters.

In Javascript string can be enclosed within double quotes or single quotes ` `

Ex:-

```
<script>
    let a="mahesh";
    console.log(a,typeof a);
    a='M';
    Console.log(a,typeof a);
    a=`mahi`;
    Console.log(a,typeof a);
</script>
```

Boolean

Boolean type is used to store True or false values

Ex:-

```
<script>
    let ename="mahesh";
    let isMarried=true;
    console.log(isMarried, typeof isMarried);
    let hasKids=false;
    console.log(hasKids, typeof hasKids);
</script>
```

undefined

undefined is a type of variable which is not yet initialized.

Ex:-

```
<script>
    let gender;
    console.log(gender, typeof gender);
</script>
```

Null

Null is a empty value are intensionally absence of object value.

```
let s1=new student();
    s1=null;
    console.log(s1);
    console.log(typeof s1); object
```

BigInt

* BigInt datatype is introduced in ES11 version.
* BigInt type is used to store very large number which cannot fit in number type.
* To represent BigInteger we have to use 'n'

ex:-

```
<script>
    let maxvalue = Number.MAX_SAFE_INTEGER;
    console.log(maxvalue, typeof maxvalue);

    let n1 = 900719925474099134875688n;
    console.log(n1, typeof n1);
</script>
```

Symbol

Symbol datatype is used to create unique identifiers for an object.

Operators

Arithmetic operators

+ (add/conc)
- sub
* mul
/ quotient
% Remainder

Ex:-

```
</script>
main();
function main()
{
    let a=5;
    let b=9;
    console.log (Math.trunc(a,b));
    Console.log (a%.b);
}
</script>
```

unary operator

```
let a=1;
a+b=++a;
console.log (a,b);
```

```
let a=1
let b=a++;
Console.log(a,b);
```

```
let a=1;
a = a++;
console.log(a);
```

```
let a=1;
a=++a;
Console.log(a);
```

Ex:-

```
let a=1;
let b=0;
b=a++ + ++a + a++;
console.log (a,b);
```

```
let a=1;
console.log (a++);
console.log(a++);
```

Ex:-

```
let a=1;
a=a++ + a;
console.log(a);
a=a++ + a++;
console.log (a);
a=++a + a++;
Console.log(a);
a=a++ + a;
corsole.log(a);
```

Concatenation

* Concatenation means joining or combining two or more operands.
* we have to use + operator for concatenation.
* for concatenation atleast one operand should be string type.

num + num → Add(num)

num + str → conc(str)

str + num

str + str

1 + 1 2

1 + "Js" "1Js"

"Js" + 1 "Js1"

"Java" + "script" = "Javascript"

1 + 2 + "Js" + 3 + 4 + "Java"

"3Js34Java"

'1' + 'a' '1a'

ex:-

```
let a=11;
let b=2;
let res=a+b;
Console.log("a value is "+a);
Console.log("b value is "+b);
Console.log("Addition of Two numbers Is "+res);
Console.log(a+" + "+b+" = " + " +res); 11+2=3
Console.log("---- Template string ----- ");
Console.log(`a value is ${a}`);
Console.log(`${a} + ${b} = ${res}`);
```

Template literal OR Template string (Es6)

`string ${Exp}`

Type Coercion

Type Coercion are type of one type to another type.

Ex:-

```
Console.log(2+"3");
Console.log(2+"3");
Console.log(2+"JS");
let a=NaN;
Console.log(a, typeof a);
```

Converting one type to another type automatically or implicity is known as type coercion.

Type coercion is done by Js engine.

To take user input in Javascript we have to use prompt function.

Ex:-

```
let n=prompt("Enter n value");
Console.log("n value is ${n}");
```

Type Conversion

* Converting one type to another type manually or explicitly is known as type conversion.
* Type conversion is done by Developer.

Ex:-

```
let n = Number(prompt("enter n value"));
Console.log(`n value is ${n}`);
console.log(typeof n);
```

Falsy values

* The values which returns false when we are converting into boolean type is known as falsy values.
* There are 6 falsy values

1) "" (empty string)
2) 0 (zero)
3) NaN
4) undefined
5) false
6) null

```
Boolean("");
Boolean(0)
Boolean(null)
Boolean(NaN)
Boolean(undefined)
Boolean(false)
```

Truthy values

* The values which returns True when we are converting into boolean type is known as Truthy values.
apart from falsy values remaining all the Truthy values.

```
Boolean(123)
Boolean([1,2,3])
Boolean("Js")
```

Number(), String(), Boolean() these are constructors used to type coercion

Relational operator

>, <, >=, <=, ==, !=, ===, !==

Ex:-

```
main();
function main()
{
    let a=5;
    let b="5";
    if (a==b)  //true
      console.log("Happy morning");
    else
      console.log("Happy Birthday");
    if (a==b)  //false
      console.log("Bad morning");
    else
      console.log("sad Birthday");
}
```

* == is a normal comparison operator.
  == it will check only the value but not datatype.
  incase == type coercion happens.

* === is a strict comparison operator.
  === It will check value as well as datatype
  Incase of === type coercion won't happens

Logical operator

&& (and)	|| (or)	! Not
t t → t	t t → t	! t → f
t f → f	t f → t	! f → t
f t → f	f t → t	
f f → f	f f → f	

Ex:-

```
let a=5;
let b=6;
console.log(a>b); //false
console.log(a>5 && b>5); //false
console.log(a>b || b>5); //true
console.log(!a>5); //false
console.log(a++>5 && b++>5); //false
console.log(a,b); // 6, 6
```

* In and operator if the first condition is false, it won't check other constraints.

```
let a=5;
let b=6;
console.log(a++>=5 || b++>=5); //true
console.log(a,b); //6, 6
```

* In or operator if first condition is true, it won't check other (2nd) condition.

Bitwise operator

& logical and
| logical or
^ logical xor
~ logical not
>> right shift
<< left shift
>>> unsigned rightshift

Ex:-

```
let a=5;
let b=7;
Console.log(5&7); //5
Console.log(a|b); //7
Console.log(a^b); //2
```

XOR

1 1 → 0
1 0 → 1
0 1 → 1
0 0 → 0

5 → 1 0 1
7 → 1 1 1
0 1 0 ⇒ 2

Ex:-

```
let a=5;
let b=6;
Console.log(~a); //-6
```

+ve sign ← 0 . . . . . 0 0 0 1 0 1 → 5

-ve sign ← 1 . . . . . 1 1 1 0 1 0 → !5
1 . . . . . 0 0 0 1 0 1 → 1st complement
add 1
-ve sign ← 1 . . . . . 0 0 0 1 1 0 ⇒ -6

0 && "Hi" 0

"Hi" && 0 0

"Hi" && "Hello" 0

"JSp" && "Hi" && null && NaN && "str"

"Hi" && null
T F

null && NaN
F

null

EX:-

☑ html

```
let checked = false;
              True
if (checked)
{
   <div>
     =
   </div>
}
else
{
   display:None;
}
```

EX:-

```
main();
function main()
{
  let ename = "mahesh";
  let hasMoney = 0;
  if (hasmoney)
    console.log("Hi frnds lets party");
  else
    console.log ("Hi frnds lets party");
}
```

strict mode

strict mode is used to maintain secured Javascript code (which means we can avoid the small mistakes which is commonly doing by developers)

Ex:-

```
<script>
 "use strict";
  main();
  function main()
  {
    let ename = "mahesh";
    let passTest = false;
    let hasDrivingLic = false;
    passTest = true;
    if(passTest)
    {
      hasDrivingLi = true;
    }
    console.log(hasDrivingLic);
  }
```

1) write a program to check the given number is even or odd.

a)
```
<script>
main();
function main()
{
  let n = Number(prompt("enter n value"));
  if(n%2==0)
    console.log(`${n} is even`);
  else
    console.log(`${n} is odd`);
}
```

ex:-

```
let n=4;
if (n%.2==0)
n++;
++n;
n++;
console.log('n value is ${n}');
```

```
let n=3;
if(n%.2==0)
{
n++;
++n;
}
n++;
console.log('n value is
${n}');
```

9) write a program for currency calculator.

A)

```
<script>
main();
function main()
{
  let amt=1500;
  if(amt>=2000)
  {
    let count2000=math.trunc(amt/2000);
    console.log('2000* ${count2000}=${2000*count2000}');
    amt=amt%.2000;
  }
  if(amt>=500)
  {
    let count500=Math.trunc(amt/500);
    console.log('500* ${count500}=${500*count500}');
    amt=amt%.500;
  }
  if(amt>=200)
  {
    let count200=Math.trunc(amt/200);
    console.log('200* ${count200}=${200*count200}');
    amt=amt%.200;
  }
```

```
if (amt>=100)
{
    let Count100=Math.trunc(amt/100);
    console.log(`100*${Count100}=${100*Count100}`);
    amt=amt%100;
}
}
```

1) leap year

2) electricity bill

units = 356;

0-200 --> free(200)

201-250 --> 3rs

251-300 --> 5rs

>=301 --> 7rs

0+50*3+50*5+56*7

3) Student final exams (5)

S1, S2, S3, S4, S5 --> marks

if any subject scored < 35 → fail

if all subject scored >=35 → pass --> find the Aggregate

--> based on agg percentage need to calc grades

>=90 A+ grade --- >=80 && <90 A grade ---- >=70 && B grade.