Web Technology

31/07/19

MEAN → NODE JS.
- MongoDB
- Angular JS
- Express JS

MERN → React JS

MEVN → Vue JS.

M → It can be used instead of SQL

E → It can be used instead of J2EE (which provides connection b/w front end, back end & business logics)

N → used instead of programming languages.

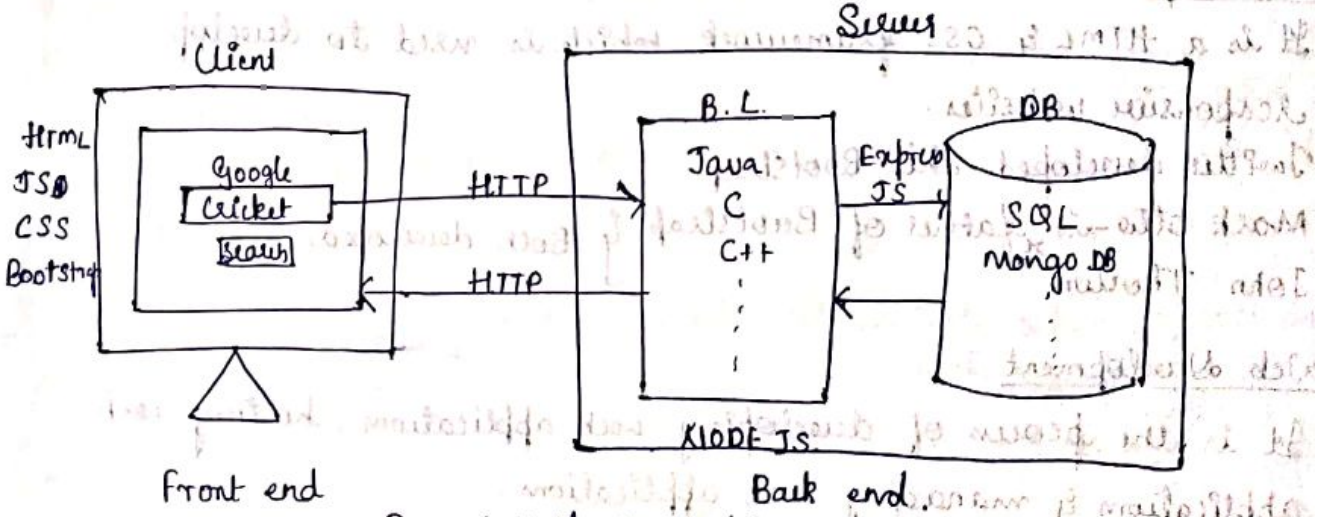
A → To get the data in the single page without loading

HTML → Structure to webpage

CSS → design

JS → actions performed by webpage.

Control + u → To get the code.



Request response model

HTML

* HTML stands for hypertext Markup language which is used to create layout or structure of the web pages.
* HTML is developed by ~~Dinu~~ Tim Berners Lee
* By using HTML we can create only static web pages.
* HTML is case Insensitive programming language
* HTML are .html or .htm are the only format supported by web browser
* Current version is 5

CSS (3)

* CSS stands for cascading Style Sheet which is used to design the web pages
* It is a case Insensitive
* It is developed by Hakon wring lie.

JS (JavaScript) but using 6

* Java Script is a client side scripting language which is used to perform actions on the web page & to make the web pages dynamic.
* It is a loosely typed and dynamic typed scripting language
* It is a case sensitive programming language
* Brendan Eich father of JS
* ECMA Script (European Computer Manufacturers Association) or ES6 is the other name of JS.

Bootstrap

It is a HTML & CSS framework which is used to develop responsive websites.
Twitter developed this Bootstrap
Mark otto is father of Bootstrap & Both developed
John Thorton

Web Development

It is the process of developing web applications, hosting web applications & managing web applications.

Network

Connection b/w two or more computers is known as network
Internet is a media which is used to access the web pages.
Internet is a collection of networks

Web

Web is an Information space which contains web related resources such as text, Images, audio, video, documents & etc

web page

A single HTML document is known as web page.

website

Collection of web pages is known as website

HTTP

It stands for hypertext transfer protocol which is responsible to carry request from client to the server as well as respons from server to the client.

Browser

It is a client side software which is used to access the web pages

Ex - Mozilla, Safari, Opera, Crome.

Search engine

It is a software which searches the data based on keywords.

Ex- google.

Server

It is a computer where all the websites are hosted.

1/8/2019

Structure of HTML

<!DOCTYPE HTML> → Version information.

<HTML>

<HEAD>
} Head Section
</HEAD>

<BODY>
} Body section. (visible) contents that appears on webpage
</BODY>

</HTML>

DOCTYPE HTML Indicates the current version of HTML

Head section contains invisible part of the web page

Body section contains visible part of the webpage

Tags

1. Paired tags | Double tags
2. unpaired tags

<tagname> Content </tagname> → for paired tags.
 Dinga

<tagname/> → for unpaired tags.
<img/>

<head>
① <link/>
② <meta/>
③ <title> </title>
④ <style> </style>
⑤ <script> </script>.
</head>

only 5 tags can be used under head section. The tags in head section can't be inserted in the body section.

Text editors.

① MS word
② Word pad
③ Note pad
④ Note pad++
⑤ Edit+
⑥ Visual code
⑦ Sublime text
⑧ Code blocks.
⑨ Brackets
⑩ Eclipse

Attributes

These are the properties which provides an extra effect to the html tags

<Tagname Attribute = "value"> Dinga </tagname>
 Dinga

<tagname Attribute = "value"/>.

Body tag attributes

Attributes	Parameters
Bgcolor	Colour Name, Color Code
Background	Imagename, Image path, Image address
Text	Color name, color code

alt + tab → to change windows

control + tab → to change the tabs.

Paragraph attributes

Attributes	Parameters
Align	Left (Default)
	Right
	Center
	Justify

 Sahara
 Sahara

Attributes of font tag

Attributes	Parameters.
Color	Colourname, Code
Size	Px ; % .
Face	Font - Family name

<body>

Jspiders
Jspiders.

2/08/19

Image tag

Attributes	Parameters :
Src	Image name, Image path, Image address.
title	Any name
Height	Px , % .
Width	—" —
Alt	Any name

<head>
<title> Background Image </title>
</head>
<body>
<img src = "alia.jpg" height = "200px" width = "200px">
<img src = "dhoni.jpg" height = "200px" width = "200px">
</body>

Marquee tag → used to apply movement to the html tags.

Attributes	Parameters.
Behavior	Scroll (default), Slide, Alternate
Direction	Left (default), Right, up, down
bgcolor	Color name, code
Height	Px, %
width	— " —
Scroll Amount	Any number.
Loop	Any number.
Scroll Delay	Time in ms.

<body>
<marquee> jspiders </marquee>
<marquee behaviour = "Slide" > Qspiders </marquee>
<marquee direction = "right" bgcolor = "purple" > Jspiders </marquee>

3/08/19

Table

In HTML table is a collection of cells.
By default table has no border

[If we pass border by default it will be 1px]

table tag Attributes	Parameters
Border	Px, %
Bgcolor	Color name, Code
Background	Image name, Image path, Image address
Height	Px, %
Width	Px, %
Cellspacing - Space b/w the cells	Px, %
Cellpadding - Space b/w the content & the border of cell	Px, %

Align	left(default), right, center
Bordercolor	Color name, code
Rule	Rows, cols, All

<Tr> attributes [table row]

Attributes	Parameters
Align	left(default), right, center
Valign	middle(default), top, bottom
Bgcolor	colorname, color code

<Td> / <Th> attributes

Attributes	Parameters
Align	Left(default), right, center
Valign	middle(default), top, bottom
Bgcolor	Colorname, code
Rowspan	Any number
Colspan	Any number

Rowspan → If we want to merge two or more rows

Colspan → to merge two or more columns.

Rules → It is used to represent the table in the form of rows & columns

Note - valign works with rowspan
Align works with colspan.

```
<table>
  <tr>
    <th>Name</th>
    <th>USN</th>
  </tr>
  <tr>
    <td>Dinga</td>
    <td>123</td>
  </tr>
</table>
```

```
<body>
<table border cellspacing="3px" cellpadding="3px" height="200px"
width="500px" rules="all">
  <tr>
    <th>Name</th>
    <th>USN</th>
    <th>marks</th>
    <th>Dept</th>
    <th>gender</th>
  </tr>
  <tr>
    <td>Sahana</td>
    <td>123</td>
    <td>95</td>
    <td>CS</td>
    <td>female</td>
  </tr>
</table>
```

<!-- -- -- -->

for commenting

Student

Student	name	usn	dept
	Dinga	123	CSE
	Dingi	456	ECE
	Sheila	789	ISE
faculty	name	Sub	Dept
	Leela	Co	CSE
	Laila	ACA	CSE
	Raaj	CN	CSE

title

	1	2	3	4
1	A		B	
2	C			D
3	F	E		

alphabets

	1	2	3	4	5	6
1	A	B		C	D	E
2	F					G
3	H	I	J	K		
4	L		N	O		Q
5		M			P	R

b ktm

Dinga					1
D	S	Lalal			2
I	H	J	P	Hello	3
N	E	S	A	qli	4
G	E	F	J	Bye Bye	5
I	L A				

1 2 3 4 5 6

5/8/19

[Audio & Video tag] Multimedia tags

<Audio>
<Video>

Attributes — Parameter

controls (mandatory)

height — Px. / %

width — Px. / %.

<iframe>
is used to Insert multiple web pages or multiple web sites at a time on a single web page

Ex- <iframe src="http://www.naukri.com"></iframe>
<iframe src="http://www.hirist.com"></iframe>

<pre> tag

It is used to preserve the white spaces

<body>
<pre>
Jspiders
Jspiders
Jspiders
</pre>
</body>

output

Jspiders
Jspiders
Jspiders.

will get the same output as written in code

Heddings - 6 types of hedding

h_1 is the bigger one & h_6 is the smaller one.

tag for underline = <u>

tags for bold = <b>,

tags for Italics = <i>,

tags for ~~Strike~~ Strik = <s>, <strike>, <del>.

<sup> tag => Superscript tag.

x²

=> x^2.

x^{jspiders}

=> $x^{jspiders}$.

<sub> tag => Subscript tag

log_x

=> $\log_x$

<q> </q> for double quotes

<q> Sahara </q>

=> "Sahara"

<mark> </mark> for highlighting purpose

 Js <mark> pid </mark> ers

=> Jspiders

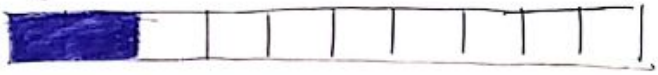
pid will display with yellow background & black text.

<details> </details>, <summary> </summary>

To get the details & summary

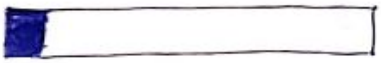
<progress> </progress>

<progress max = "200" value = "20"> </progress>.

=> 

<meter> </meter> => To create pie chart.

<meter max = "100" value = "10"> </meter>



Special characters

These are case sensitive & it always starts with amphasent (&) and ends with semicolon.

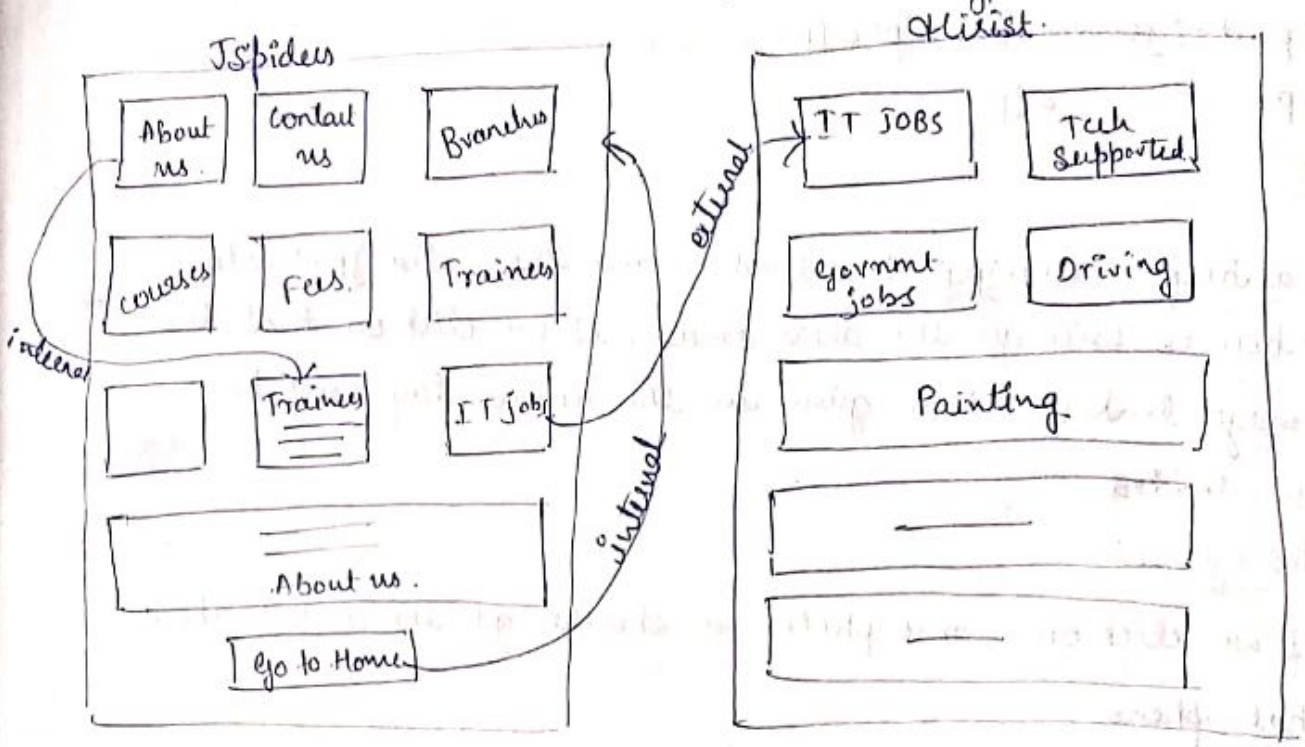
← ←
→ →
↑ ↑
↓ ↓
‘ Jspiders ’ 'Jspiders'
“ Jspiders ” "Jspiders"
© ©
™ ™
® ®
° °
♥ ♥
♦ ♢
♠ ♤
♣ ♧

nbsp; → non breaking space
It is used to provide multiple spaces

6/8/19

Hyperlinks

It can be created by using Anchor tag indicated by <a>
There are two types of hyperlinks. 1. Internal hyperlink
2. External hyperlink



Attributes	Parameters
href (mandatory)	HTML file, URL
title	Any name
target	Parent (default), blank
Download	

```
<body>
<a href="main.html" title="Dinga" target="blank">JSpiders</a>
<a href="http://www.hirist.com">It-jobs</a>
```

 Java
 Javascript
 HTML
 CSS
 Bootstrap

 Java .
 — ———
 Javascript
 — —

 <img src = "alia.jpg" > .
=> here we will get the alia's image, if we click on that the image link which is given in the anchor tag will be downloaded

Assignment

* if we click on some photos we should get the content about that photo.

2.

Name	Category	Image	Information
tiger	wild animal		
dog	Domestic animal		

7/8/19

<form>
① <input>
② <select> </select>
③ <optgroup> </optgroup>
④ <option> </option>
⑤ <textarea> </textarea>
⑥ <button> </button>
⑦ <fieldset> </fiddset> <datalist>
⑧ <legend> </legend>
</form>

<input/> Attributes | Parameters

type | text, password, checkbox, radio, file, date, color, number, email, url, reset, submit

Maxlength | ~~Rest~~ any number

size | ~~any number~~ Px, %.

value | any ~~number~~ name

Placeholder | any ~~number~~ name

Name | any name.

Required

checked

disabled

<textarea>

Attributes	Parameters
Rows (height)	Px, %
Cols (width)	Px, %

<Select>

Attributes	Parameters
Multiple	

<form>

Attributes	Parameters
Action	HTML file, URL
target	parent (default), Blank
Method	* display on url. get (default), post
Name	Any name

</form>

username: <input type="text" name="un" placeholder="Enter valid username" maxlength="7">

password: <input type="password" name="pw" placeholder="Enter valid password" size="30px" required>

upload documents: <input type="file">

<input type="number">
. [we can restrict by using max & min attributes]

email: <input type="email">

URL: <input type="url">

Select your subjects:

<input type="checkbox" name="c1">Java.

<input type="checkbox" name="c1">Javascript

<input type="checkbox" name="c1">HTML

<input type="checkbox" name="c1">CSS.

Select your gender:

<input type="radio" name="r1">male

<input type="radio" name="r2">female

<input type="radio" name="r1">others
.

<input type="reset" value="clear">.

<input type="submit" value="login">.

</form>

<form>

username: <input type="text" name="un" placeholder="Enter valid username" maxlength="7">

password: <input type="password" name="pw" placeholder="Enter valid password" size="30px" required></br>

<select>

<option>Select your state</option>

<option

Select any item:

<select multiple>

<optgroup label="Nonveg">

```
<option>chiken</option>
<option>mutton</option>
<option>chicken 65</option>
<option>kabab</option>
</optgroup>
<optgroup label="veg">
<option>curd rice</option>
<option>lemon rice</option>
<option>Pulav</option>
<option>gobi</option>
</optgroup>
<textarea rows="10px" cols="50px" placeholder="Enter your feedback">
</textarea><br>
Select your state: <input type="text" list="s"><br>
<datalist id="s">
<option>karnataka</option>
<option>goa</option>
<option>jammu Kashmir</option>
<option>kerala</option>
<option>Delhi</option>
</datalist>
<button>click</button><br>
<fieldset>
<legend align="center">click any button</legend>
<input type="reset" value="clear">
<input type="submit" value="login">
</form>
```

https://bit.ly/2X2c2NO

<form action="a.htm" method="post">
username: <input type="text" name="un">
password: <input type="password" name="pwd">
<input type="submit" value="login">
</form>

8/9/18

CSS – Cascading Style Sheets

It is used to design the web pages & to make the web pages interactive.

It is case insensitive.

Different types of Style Sheets

① Inline style sheets.
② Document level stylesheets / Internal
③ External Style Sheets

① Inline Style Sheets

Syntax:

<tagname style="cssproperty1: value1; cssproperty2: value2;">
content</tagname>

Ex - Dinga .

② Document level style sheets

Syntax

Tagname { cssproperty: value; - - - - }

Ex
```
<head>
  <style>
    Tagname{ cssproperty: value; - - - - }
      p, a { color: red }
  </style>
</head>
<body>
  <p> Dinga </p>
  <p> Dingi </p>
```

```
<P> JSP </P>.
<a href = " " > QSP </a>.
</body>
```

③ External style sheets

```
<head>.
<link href = "style.css" rel = "stylesheet" />
</head>
<body>.
<P> Dinga </P>.
<P> Sheela </P>.
<P> Leela </P>.
<a href = " " > JSP </a>.
</Body>.
```

P1.html.

```
<head>
<link ---- />
</head>
<body>
<P> Nick </P>.
<P> Mony </P>
<a href = " " > PSP </P>
<body>
```

P3.html

```
<head>.
<link ------ />.
</head>
<Body>.
<P> JSpiders </P>.
<P> QSpiders </P>.
<P> Pyspiders </P>.
<a href = " " > QSP </a>.
</Body>.
```

P2.html.

```
Tagname.
{
 cssproperty1: values;
  :
 csspropertyn; valuen;
}
p
{
color: green;
}
```

Style.css

Inline Style Sheets

Whenever we are passing CSS properties using style attributes it become inline style sheet. It always overwrite the same properties of document level style sheets as well as external style sheets. If the properties are different then it inherits the properties from the external style sheets as well as from internal style sheets.

Example

```
<!doctype html>
<html>
<head>
<title>ES1</title>
<link href="alia.jpg" rel="icon">
<link href="style.css" rel="stylesheet">
</head>
<body>
<p>Jspiders</p>
<p>Qspiders</p>
<a href=" ">Pyspiders</a>
</body>
</html>
```

Same as ES1

ES1.htm

```
p
{
color: red;
}
```

Style.css

ES2.htm

9/8/2019

Background properties-

① Background-color : color name, color code
② Background-Image : URL(Image Name)
③ Background-Repeat : Repeat-x, Repeat-y, No-Repeat.
④ Background-Attachment : scroll(default), fixed
⑤ Background-size : width, height, cover
⑥ Background-Position : left, right, center

Document level

Whenever we are writing CSS properties by using style tag then the style sheet becomes document level style sheets.
Whenever we are passing CSS properties by creating an seperate file with .CSS extension then it becomes external style sheets.

External style sheet can be imported by using <link> tag.
We can call (import) a n number of style sheets in a single document

```
<style>
body{
background-color: pink;
background-image: url(alia.jpg);
background-repeat: no-repeat;
background-attachment: fixed;
color: white;
background-size: cover;
}

p
{
background-color: rgba(250, 61, 113, 0.2);
height: 500px;
width: 500px;
}
```

Font Properties

① Font-style: Italic, Normal(default)

② Font-weight: Bolder, lighter(dufault).

③ Font-family: Any Font-family Name

④ Font-size: Px

```
<body>
<p style="font-size: Italic">jspiders</p>
<p style="font-weight: bolder">jspiders</p>
<p style="font-family: algerian">jspiders</p>
<p style="font-size: 100px">jspiders</p>
</body>
```

Text properties

① Color : color name, color code

② Text-indent : Px

③ Text-align : left (default), Right, Center

④ Text-transform : uppercase, lowercase, capitalize

⑤ letter-spacing : Px

⑥ word-spacing : Px

⑦ line-height : Px

⑧ Text-shadow : x axis Y axis Blur Ratio Color.

```
<body>
<p style="color: rgba(255, 87, 99, 1)">JSpiders </p>
<p style = "text-indent : 8px"> JSpiders blr ~ </p>.
<p style = "text-align : center"> JSpiders </p>
<p style = "text-transform : uppercase"> JSpiders </p>
```

⑨ Text-decoration : underline, overline, line-through, none.

```
<body>
<p style = "text-decoration : overline"> Dinga </p>
<a href = " " style = "text-decoration : none"> Dinga </p>
```

to remove the default underline provided by anchor tag.

10/8/19

HTML Lists

List is used to arrange the html elements in proper order

There are 3 types of list

1. Ordered list (OL)
2. Unordered list (UL)
3. Definition list (DL)

<OL>

Attributes	Parameters
Type	1 (default), a, A, i, I.
Start	Any Number

.

Attributes

Type

Parameters:

Disc (default), Square, circle

Assignment

1. Programming Languages.
   - Java
   - JS
   - HTML
   - CSS
   - SQL
2. Java Frameworks.
   - React JS
   - Express JS
   - Angulas
4. Vue JS
   - Lime JS

dt → definition Term

dd → definition data

Definition List

```
<dl>
    <dt> Javascript </dt>
    <dd> Javascript is client side scripting language </dd>
</dl>
```

To insert image

```
<ul>
<li> Java </li>
<li style = "list-style-image: url(o.png)"> Javascript </li>
<li style = "color: red"> HTML </li>
<li> CSS </li>
<li> SQL </li>
</ul>
```

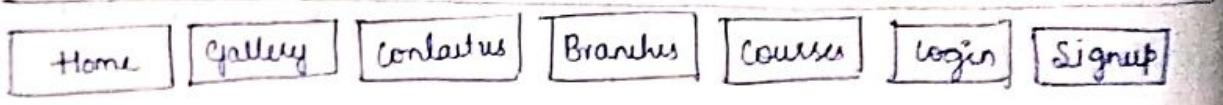
horizontal tag <hr/>

hr tag is used to create horizontal lines on web pages.

Attributes	Parameters
Size	Px, %
Color	Colornam, code.
width	Px, %. (100%) Default.
Noshade	
Align	Center (Default), left, right.

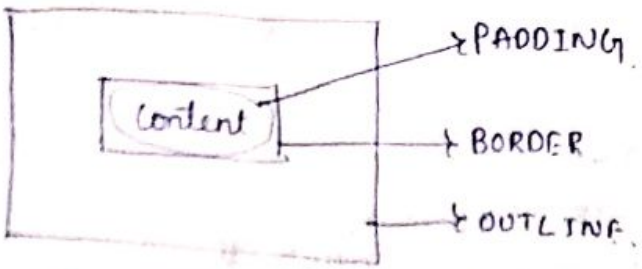
```
<body>
<hr size="100px" width="50%" align="left" noshade/>
<hr color="red" size="100px" width="50%"/>
<hr color="purple" size="100px" width="50%" align="right"/>
</body>
```

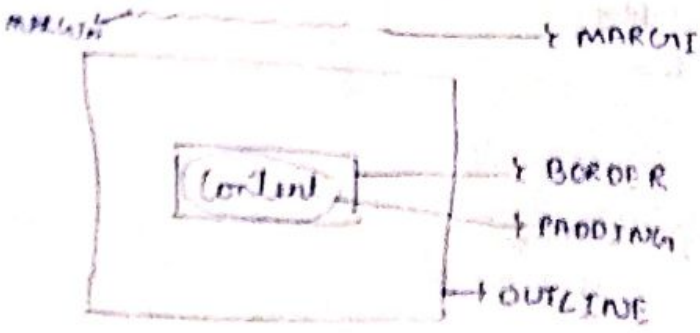
Assignment



13/8/19

Box Model





Border properties

① Border-style : solid, groove, dotted, dashed, double, none ...

② Border-width : Px, %

③ Border-color : color name, color code

④ Border-Radius: TLC TRC BRC BLC

Shortform

Border: width Style color

① Border-top-style: solid, groove ...

② Border-bottom-style: " "

③ Border-left-style: " "

④ Border-right-style: " "

```
<body>
<p style="border-style: solid; border-color: red; border-width: 4px">
Jspiders</p>
<p style="border-style: Dotted; border-bottom-color: lime; border-top
Style: double"> Qspiders</p>
<p style="border-style: solid; border-radius: 30px 0px 30px 0px">
Pyspiders</p>
<p style="border: 2px dashed green"> Dinga</p>
</body>
```

outline

```
<body>
<p style="border: 2px Solid green"> jspiders</p>
<p style="outline: 3px solid red"> qspiders</p>
<p style="border: 2px solid lime; outline: 2px solid purple"> pyspiders
</body>
```

Outline-offset: 3px

Outline Properties

① Outline-style : Solid, groove, dotted, dashed, double, none

② Outline-width : Px, %.

③ Outline-color : Color name, color code.

④ Outline-offset : Px, %. [to provide space b/w outline & border]

Padding Properties

① Padding-left : Px, %.

② Padding-top : Px, %.

③ Padding-right : Px, %.

④ Padding-bottom : Px, %.

Shorthand Padding : Px, Pt

```
<body>
<p style="border: 2px solid pink; background-color: lime;
padding-left: 10px; padding-top: 20px">Jspiders</p>
<p style="border: 2px solid pink; background-color: lime;
padding: 20px">Jspiders</p>
</body>
```

Margin Properties

① Margin-left : Px, %.

② Margin-top : Px, %.

③ Margin-right : Px, %.

④ Margin-bottom : Px, %.

Shorthand

Margin : Px, P%

```
<style>
img{border-radius: 100%; height: 200px; width: 200px;
    box-shadow: 4px 4px 3px red;
button{background-color: lime;
       border: 2px solid red;
       outline: none;
```

```
border-radius : 100%;
padding : 10px }
</style>
</head>
<body>
<img src="1.jpg"><br/>
<button> click button </button>
</body>
</html>
```

To pass shadow to the boxes

box-shadow : X axis Y axis Blur ratio color

14/8/19

Selectors

① Element selector / Simple selector.
② Group selector
③ class selector.
④ Generic selector
⑤ ID selector.
⑥ Universal selector.

Selectors are used to select the content we want to style.

① Element selector

Syntax

Tagname { css properties : values; ---- }

Ex

```
<head>
  <style>
    P { Color : Red }
  </style>
</head>
<body>
  <P> Dinga </P>
  <P> Dingi </P>
  <a href=" "> JSP </a>
</body>
```

Element selector selects the content based on Tag name.

② Group Selector

① Syntax

② Tagname1, Tagname2 ... { cssproperties : values; ..... }

③ Ex

div, p, a { color : red }

④ group selector selects multiple tags at a time.

③ Class Selector

Syntax

① tagname . classname { cssproperties : values; ..... }

② Ex

③ <head>
<style>

④ P. Dinga { Color : red }

</style>
</head>
<body>
<P class = "Dinga"> Dinga </P>
<P> Dingi </P>
 JSP
 QSP
</body>

! Class selector selects the content based on tag name as well as class name.

② Class selector can be called only inside particular tags.

③ ④ Generic Selector

Syntax

④ .classname { cssproperties : values; ..... }

< Ex

<head>
<style>
.Dinga { Color : red }
</style>
</head>
<body>

```
<p class="Dinga">Dinga</p>
<p>Dingi</p>
<a href="">JSP</a>
<div class="Dinga">QSP</div>
</body>
```

Generic selector selects the content only based on classname.
It can be called inside any tag.

⑤ ID Selector

Syntax

```
#idname{ cspeoperties : values; ---- }
```

Ex

```
<head>
<style>
  #Dinga{color:Red}
</style>
</head>
<body>
<p id="Dinga">Dinga</p>
<p>Dingi</p>
<a href="">JSP</a>
<div Id="Dinga">QSP</div>
```

id selector selects the content based on id name
whenever we are calling id selector and generic selector in a same Tag then the prefference will be given to id selector.
id selector can be called inside any Tag.

⑥ Universal selector

Syntax

```
* { cspeoperties : values; ---- }
```

Ex

```
<head>
<Style>
  * { color: red }
</style>
</head>
<body>
```

 Dinga
 Dingi
 QSP
 HTML .
</body>

The CSS property is applicable to all the tags present in body.

Example

<!doctype html>.
<html>
<head>
<title> FS2 </title>.
<style>
.a { color: red }.
.b { ~~color: green~~ border: solid 2px line }
.c { background-color: pink }
</style>.
</head>
<body>
 Dinga
 Dingi
 JSpiders
 QSpiders
 JSp
 QSp .
 PSp
 HTML

```
<style>
#a{color: red}
.b{background-color: pink}
#c{color: pink}
.d{color: green}
</style>
</head>
<body>
<p id="b">Dingo</p>
<p id="a" class="b">Dingi</p>
<p>Jspiders</p>
<p>Qspiders</p>
<div id="a">Jsp</div>
<div id="c" class="d">Qsp</div>
<div>Psp</div>
<a href="" id="a">Sheela</a>
<a href="">Leela</a>
<a href="">Laila</a>
</body>
```

we can't use two or more id name in a same tag.

<!-- preference is given to id selector -->

Example

```
<style>
.a{border: 2px solid red; background-color: pink}
.b{box-shadow: 5px 5px 3px lime; border-radius: 100%}
</style>
</head>
<body>
<img src="1.jpg" class="b a"><br>
<p class="a">Jspiders</p>
</body>
</html>
```

15/8/19

Combinators

① Descendent selector (Space)

② child Selector (>)

③ Adjacent sibling selector (+)

④ General sibling selector (~)

Combinators explains the relationship b/w the other selectors.

① Descendent selectors - It selects all the direct & indirect childs and it is indicated by space

② Child selector - It selects only direct childs and it is indicated by > symbol

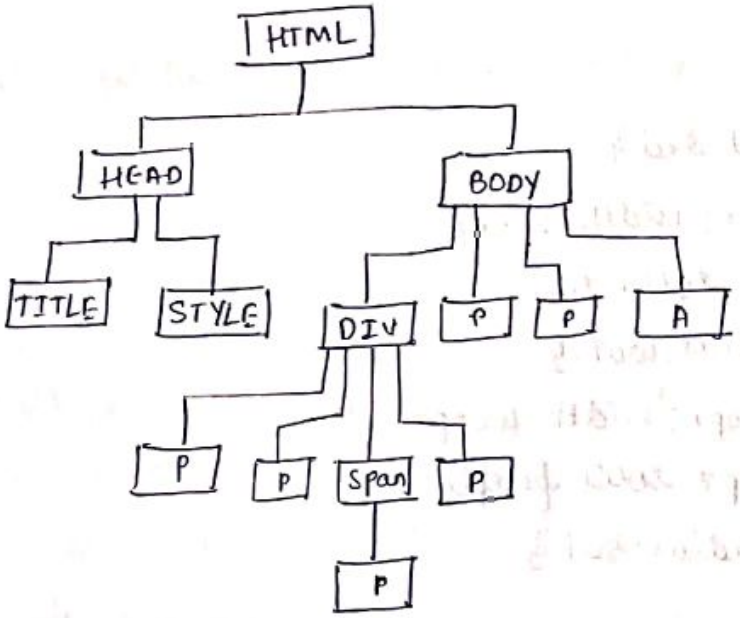
③ Adjacent sibling selectors - It selects only direct (immediate) siblings. and it is indicated by + symbol.

④ General sibling selectors - It selects all the direct & indirect sibling indicated by ~ symbol

Structure of HTML document

```
<HTML>
  <head>
    <title> Hello </title>
    <style> </style>
  </head>
  <body>
  <div>
     <p> Sheela </p>
     <p> Leela </p>
     <span> <p> Laila </p> </span>
      <p> Kamala </p>
  </div>
     <p> Dinga </p>
     <p> Dingi </p>
     <a href=" "> JSP </a>
  </body>
</HTML>
```

Tree structure of HTML document



Example

```
<style>
div p { color: red }
</style>
</head>
<body>
 <div>
    <p> Sheela</p>
    <p> leela </p>
    <span> <p> laila</p> </span>
    <p> Kamala</p>
 </div>
<a href=" " > QSP </a>
<p>dinga</p >
<p> dingi</p>.
<a href=" " > Tspiders</a>
</body>
</html>.
```

Example

```
<html>
<head>
<style>
div { boedes: 12px solid red }
div img { height: 200px; width: 200px;
          boder : 20px solid lime;
          border-radius: 100%. }
div:nth-child img { height : 400px; width: 400px;
          boeder : 20px solid purple;
          border-radius : 50%. }
</style>
</head>
<body>
<div>
<img src="1.htm "/>
<img src = "2.htm"/>
<img src = "3.htm"/>
</div>
<img src = "4.htm"/>
<img src = "5.htm"/>
</body>
</html>
```

Position property

① Position : Static (default), Relative, Sticky

② Top : Px, %.

③ Left : Px, %

④ Right : Px, %.

⑤ bottom : Px, %.

```
<head>
<style>
 p { position: relative; left: 79px; top: 72px }
</style>
</head>
<body>
<p> Jspiders </p>
</body>
</html>

<style>
 input { position: sticky;
         left: 110px;
         bottom: 0px;
         border-radius: 50px;
         outline: none;
         }
</style> <body>
<input> Jspiders
<input type = "text" placeholder = "Enter search element">
<pre>
</pre>
</body>
</html>
 <style>
 .a { background-color: orange; height: 100px }
 .b { background-color: white; height: 100px }
 .c { background-color: green; height: 100px }
 img { position: relative; top: -200px; left = 250px; height: 100px;
       width: 100px }
 </style>
 <head>
```

```
<body>
<div class="a"></div>
<div class="b"></div>
<div class="c"></div>
<img src="  ">
</body>
```

16/8/19

Pseudo Classes

Pseudo classes are used to specify the current state of html elem

syntax:

selector: pseudo.classname{ css properties }

① selector : link { css properties } } only for <a> tag
② selector : visited { css properties } }
③ selector : hover { css properties }
④ selector : Active { css properties }
⑤ selector : Focus { css properties } } only for <input/> tag.
⑥ selector : valid { css properties }
⑦ selector : Invalid { css properties } }
⑧ selector : first-child { css properties }
⑨ selector : Last-child { css properties }
⑩ selector : n^{th}-child (Any Nos) { css properties }
⑪ selector : only-child { css properties }
⑫ selector : n^{th} last child (Any no's) { css properties }
⑬ : not (selector) { css properties }

```
<style>
a:link { text-decoration: none }
a:hover { text-decoration: underline }
a:visited { color: orange }
a:active { color: lime }
</style>
</head>
<body>
<a href="www.html">Jspiders</a>
```

```
<a href="qqq.html"> QSpiders </a>
</body>
</html>

<Style>.
input:focus { background-color: lime ; color: red }
input:hover { background-color: pink ; }
</Style>.
</head>
<body>
username: <input type="text" placeholder="Enter valid username">
</body>
</html>

<Style>.
input: valid { color: green }
input : invalid { color: red }
</Style>.
<body>
username: <input type="Email" placeholder="Enter valid Username">
</body>
```

17/8/19

Pseudo Elements

It is used to change the current state of the content and it is indicated by ::

Syntax

selector :: pseudo-element-name { css property }

① selector :: first-line { css properties }

② selector :: first-letter { css properties }

③ selector :: selection { css properties }

④ selector :: after { css properties }

⑤ selector :: before { css properties }

```
<html>
<head>
<style>
p.b::first-line{color:red}
p.a::first-letter{color:lime}
</style>
</head>
<body>
<p class="b">welcome to jspiders <br/>
welcome to jspiders <br/>
welcome to jspiders <br/>
welcom to jspiders <br/><p>
<p class="a">welcome to jspiders <br/>
welcome to jspiders <br/>
welcome to jspiders <br/>
welcome to jspiders <br/></p>
</body>
</html>
<head>
<style>
::selection{background-color:lime; color:red}
p::selection{background-color:purple; color:pink}
</style>
</head>
<body>
<p>welcome to jspiders <br/>
welcome to JS <br/>
welcome to JS<br/>
welcome to JS <br/></p>
<span>welcome to JS <br/>
welcome to JS <br/>
welcome to JS <br/>
welcome to JS </span>
</body>
</html>
```

```
<html>
<head>
<style>
p.a::before { content:"Jspiders"; color:lime }
p.b::after { content:"Java training institute"; color:red }
span::before { content:url(1.jpg) }
</style>
</head>
<body>
<p class="a">Java training institute</p>
<p class="b">Jspiders
<span>Alia Bhat</span></p>
</body>
</html>
```

Overflow property → add scroll bar to overflow content

① overflow: visible(default), scroll, hidden, auto

auto → if the content is more by default it will add scroll b

```
<style>
p { background-color:lime; width:100px; height:100px; overflow:auto }
</style>
<body>
<p>welcome to jspiders
_____
_____ _
_____ ____
____ _ _
_____ _
_ _ _</p>
</body>
</html>
```

Visibility property → is used to hide the content

① visibility: visible(Default), hidden.

```
<body>
<p>Welcome to jspiders
Welcome to jspiders
</p>
```

```
<p style="visibility: hidden"> welcom to Jspiders
welcom to Jspiders</p>
</body>
</html>
```

Opacity properties

① Opacity : 0.1 – 1

```
<style>
img { opacity: 0.5 }
img : hover { opacity: 1 }
</style>
</head>
<body>
  <img src="1.jpg">
</body>
</html>
```

Display properties

① Display : Inline (Default), Block (default)

```
<style>
img { height: 200px;
      width: 200px;
      border 4px solid red;
      border-radius: 100%;
      box shadow: 4px 4px 3px lime;
      visibility: hidden }
span : hover + img { visibility: visible }
</style>
</head>
<body>
<span> Kajal agarwal </span>
<br>
<img src="1.jpg">
</body>
```

```
<body>
<p>JS<span style="color:red">p</span>iders</p>
</body>
```

Span is a dummy tag which has no effect.

display properties

```
<html>
<head>
<style>
p{background-color:red; display:inline}
span{background-color:pink; display:block}
</style>
</head>
<body>
<p>Jspiders</p>
<p>jspiders</p>
<p>jspiders</p>
<span>jspiders</span>
<span>jspiders</span>
<span>jspiders</span>
</body>
</html>
```

```
<style>
img{height:300px; width:300px; float:right}
</style>
<body>
<img src="4.jpg">
<p>Kajal Agarwal tollywood actress</p>
</body>
</html>
```

```
<style>          linear-gradient( - - ---- - - )}
p{background-image: radial-gradient(red, green, blue, cyan)}
</style>
<body><p>Jspiders</p>
</body>
```

26/08/19

Transform Properties.

① Transform : Rotate(180 deg)
Scale(width, height)
Translate(x-axis, y-axis)

Transform property is used to rotate, translate, scale the html elements.

Scale - to increase height & width

translate - to move the html elements.

```
<style>
div { height : 100px;
      width : 100px; transform: translate (10px, 50px);
      background-color : purple; transform : scale (1, 2);
      transform : rotate(30deg) }
</style>
<body>
<div>
</div>.

<style>
div { height :100px;
      width : 100px ;
      background-color : purple; }

div : hover { background-color : yellow; transform : rotate(30deg) }.
</style>
</head>
<div body>
<div>
</div>
</body>.
</html>
```

Transition.

Transition allows us to change the css properties smoothly over a given duration.

① Transition : Csspropertyname timeinsecond(s)

② Transition-delay: Time in su(s)

③ Transition-duration: Time in su(s)

```
<style>
div {height: 100px;
     width: 100px;
     background-color: purple;
     transition: background-color 2s, height 4s, width 4s, transform 2s;
    transition-duration: 20s;
    transition-delay: 5s;
}
div:hover {background-color: red; height: 300px; width: 400px;
           transform: rotate(180deg)}
</style>
</head>
<body>
<div>
</div>
</body>
</html>
```

Animation Property

① Animation-Name : Any name

② Animation-duration : Time in su(s)

③ Animation-delay : Time in su(s)

④ Animation-Iteration-count : Anynumber, Infinite

⑤ Animation-Direction : Forward(default), Reverse

```
<style>
div { height: 100px;
      width: 100px;
      background-color: purple;
      animation-name: dinga;
      animation-duration: 20s;
    }
@keyframes dinga {
  0% { background-color: pink }
  10% { - - - - - - - - }
  .
  .
  .
  100% { background-color: yellow }
}
</style>
</head>
<body>
<div>
</div>
</body>
```

@keyframes dinga {
from { cssproperties : value }
to { cssproperties : value }
}

27/08/19 Sign up & login page

28/08/19

Java Script

It is a client side scripting language which is used to create dynamic web pages.

Java Script is also used to perform actions on the web page

- It is developed by Netscape
- The another name of JS is ECMA script (ES)
- The current version of JS is ES10 but we are using ES6 because of its features
- Brendan Eich is the father of Java Script.

following are the frameworks of Java Script.

1) Angular JS
2) React JS
3) Node JS
4) Angular
5) Express JS
6) Vue JS

Node JS is a server side programming language & is used to write business logics.

Express JS is also a server side language & is used for connection purpose

Angular JS, React JS, Angular, Vue JS are used to create single page applications.

Java	Javascript
1) Java is a server side programing language	It is both server & client side programming language.
2) Java is a compiled and intrepreted programming language	Java script is only intrepreted programming language
3) Before execution of JAVA program compilation is mandatory	Compilation is not mandatory
4) Javac is a compiler & JVM is the execution unit.	Browser is the execution unit.
5) Whole block of code will be executed at a time.	Line by line execution happens.
6) Java is statically typed programming language.	JS is a dynamically typed programming language
7) Java is not loosely typed programming language.	JS is a loosely typed programing language.

Types of Javascript

There are two types of Java Script

1) Internal JS
2) External JS.

1. Internal JS - Internal JS can be inserted in two ways by using script tag.

The two ways are -

1. Inside head tag
2. Inside body tag

Script tag should contain only JS code, it should not contain any HTML tag.

Note - HTML tag should not be direct child to script tag.

Whenever we are inserting script tag inside the body it should be the last tag of the body.

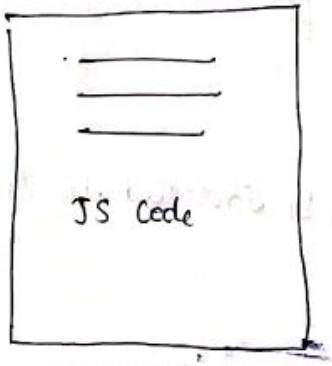
```
<HTML>
  <head>
    <script>
    // JS code
    </script>
  </head>
  <body>
    <script>
    // JS code
    </script>
  </body>
</html>
```

index1.html

```
<HTML>
 <head>
  <script src="app.JS">
  </script>
 </head>
 <body>
  <script src="app.JS">
  </script>
 </body>
</html>
```

index2.html

```
<HTML>
 <head>
 </head>
 <Body>
 <script src="app.JS">
 </script>
 </body>
</html>
```



app.JS

Java Script display statements / Printing statements

1] document.write()
2] document.writeln()
3] console.log()
4] inner HTML => [DOM element]

```
<!doctype html>
<html>
<head>
<script>
document.write("Hello");
document.write("Hi");
document.writeln("Hello");
document.write('Hi');
document.write("Hello" + "<br>");
document.write("Hi");

</script>
</head>
<body>
</body>
</html>
```

2] External

app.js

```
document.write("Hello" + "<br>");
document.write("Hi");
console.log("Bye Bye");
```

Javascript.html

```
<html>
  <head>
  </head>
<body>
<noscript> This browser can't support java script.
</noscript>
<script src="app.js">
</script>
</body>
</html>
```

Note - No script tag works only when the browser is not supporting Java Script.

app.js

```
document.write("Hello" + "<br/>");
console.log("Bye Bye");
document.write("Hi");
```

o/p

Browser

Hello

Console

Uncaught TypeError
console.log is not a function
at app.js:3

29/08/19

Keywords to delare the variables in Javascript

① var

② let

③ const

Datatypes in Javascript.

① Primitive

Number

String

Boolean

undefined

null

② Non-Primitive

object()

array()

Date()

:

① Number

Syntax

① var Variable_Name;

② var Variable_Name = value;

③ var Variable_Name1 = Variable_Name2 = value;

Example

```
var a = 20;
var a = 20.44f;   } Number
var a = -80;
```

② String

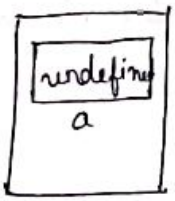
```
var a = "Divya";
var a = 'Jeee';
var a = "20";
var a = 'AB';
```

③ Boolean

```
var a = true;
var b = false;
```

④ Undefined

```
var a;
d.w(a) => undefined
```



⑤ Null

```
var a = null;
```

Example

```
var a = 20;
var b = a;
a = 80;
document.write(a)
document.write(b)
```

output
80
20

Dynamically Typed

```
var a = 20;
console.log(a);
a = "Dinga";
console.log(a);
a = true;
console.log(a);
```

output
20.
Dinga
true

```
var a = 20;
console.log(a);
var b;
console.log(b);
console.log(c);
```

output
20.
undefined.
variable c is not defined

var	let	const
var a; declaration ✓	let ~~var~~ a; ✓	const ~~var~~ a=50 ✓
a = 20; initialization ✓	a = 20; ✓	a = 20; X
a = 30; reinitialization ✓	a = 30; ✓	a = 30; X
var a = "Dinga"; redeclaration ✓	~~var~~ a = "Dinga"; X (let)	~~var~~ a = "Dinga"; X (const)

Line by line execution takes place

```
var a;
console.log(a);
a=20;
console.log(a);
a=30;
console.log(a);
var a="Dinga";
console.log(a);
```

output
undefined
20
30
Dinga

```
let a;
console.log(a);
a=20;
console.log(a);
a="Sheila";
console.log(a);
let a="Dinga";
console.log(a);
```

output:
undefined
20
Sheila
error: variable a is redefined.

```
const a=20;
console.log(a);
a="Sheila";
Console.log(a);
const a="Dinga";
console.log(a);
```

output
20.
error: const variable a can't be reinitialised

```
var a=20;
if(true)
{
  var a="Dinga";
  console.log(a);
}
console.log(a);
```

output
Dinga
Dinga

Scope of var is not limited

```
let a = 20;
if (true)
{
  let a = "Dinga";
  console.log(a);
}
console.log(a);
```

Output
Dinga
20

```
const a = 20;
if (true)
{
  const a = "Dinga";
  console.log(a);
}
console.log(a);
```

Output
Dinga
20.

whenever we are declaring variable using var keyword then declaration, initialisation, reinitialisation and redeclaration is possible.

whenever we are declaring variable using var keyword then scope is not limited to the blocks except function So that we call var is a function scope ~~variable~~ keyword.

whenever we are declaring variable using let keyword then declaration, initialisation and reinitialisation is possible. Re-declaration is not possible. Here the scope of the variable is limited.

whenever we are declaring variable using const keyword then declaration with initialisation is the only possible thing. Here the scope is limited to the blocks

① =

```
var a = 20;
```
↓ assignment

② == → value

```
var a = 20;
var b = 20;
if (a == b);
{ c.l ("True");
}else
{ c.l ("False");
}
```
output: True

③ === → values & datatypes

```
var a = 20;
var b = "20";
if (a == b);
{ c.l ("True");
}else
{ c.l ("False");
}
```
Output: false

= operator is an assignment operator which is used to assign the values.

== operator is a relational operator which checks the condition based on values.

=== operator is a relational operator in JS which is used to check the condition based on the values as well as based on the datatypes.

30/08/19

```
var i = "Dinga";
for(var i=1; i<=10; i++)
{
  console.log(i);
}
console.log(i);
```

var output	let output	const output
1	1	error
:	:	
10	10	
~~Dinga~~	Dinga	
11		

```
document.write("<span style='color:red'>"+"Hello"+"</span>"+"<br>");
document.write("Hi");
```

To avoid the ambiguity in JS we are using single & double quot

Interpolation

Interpolation is the process of evaluating template literals which contains one or more place holders.

Placeholder excepts variable names object properties, functions arrays, etc.

```
var day = "Sunday";
var boy = "Dinga";
var girl = "Dingi";
var place = "Ooty";
document.write(boy + " and " + girl + " going to marry on " + day + " and they are going to " + place + "<br>");
```

```
var a = "Dinga" // string literals
var b = 'Dingi'
```

var c = `Dinga` => Template literals

Syntax

document.write(`${variablename}.....${variablename}`);

Ex - document.write(`${a} and ${b}`);

Anything that is enclosed inside single & double quotes are called String literals.

Anything that is enclosed inside backticks are called template literals

Example

document.write(`${boy} and ${girl} going to marry on ${day} and they are going to ${place} <br/>`)

Note Interpolation is used to avoid multiple concatination of string literals

```
var a = 20;
var b = 40;
var c = a+b;
document.write("The sum of "+a" and "+b" is " +c + "<br/>" );
document.write("<span style='color: red'>" + "Hello" + "</span>"+"<br/>
document.write(`<span style="color: green"> Hello </span> <br/>`);
```

Type Of operator

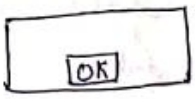
It is used to check the type of data or variable.

```
var a = 20;
var 1 = 20.22;
var b = "Dinga";
var c = 'Dingi';
var d = true;
var e = false;
var f;
var g = null;
document.write(typeof a+"<br/>")
    "       "       "   1+   "
    "       "       "   g+   "   )
```

Output

number
number
string
string
boolean
boolean
undefined
object.

Pop up boxes



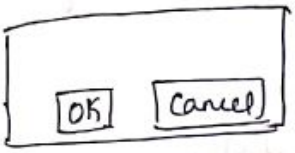
① alert()

It is used to alert the user which provides warning messages. until & unless we press OK browser will not allow us to access the web page

② confirm()

is used to get confirmation of user.

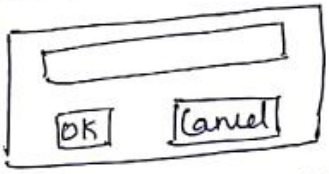
when we press OK true value will be assigned. when we press cancel false value will be assigned



③ prompt()

It is used to take the user inputs

when we press OK entered data will be stored and when we press cancel null value will be stored.



prompt box always accepts all the values as strings

```
alert('Danger');
confirm('Are you sure want to download');
prompt("Enter any value");

var a = confirm("Press ok or cancel");
alert(a)
var a = prompt("Enter value of A")
var b = prompt("Enter value of B")
alert(a+b)
```

3/9/19

```
var a = Number(prompt("Enter value of A"));
var b = Number(prompt("Enter value of B"));
alert(`The sum of ${a} and ${b} is ${a+b}`)
```

```
console.log(isNaN(34));          false
console.log(isNaN("34"));        false
console.log(isNaN(true));        false
console.log(isNaN(undefined));   true
console.log(isNaN("Dinga"));     true.
```

NaN stands for Not a Number.

isNaN() is a function which is used to check whether the given data is a number or not.

It always returns boolean value.

If the given data is a number then it returns false

If the given data is not a number then it returns true.

Number() is used to convert the string value to a number.

It converts the string to a number only when the string contains a number. else it returns NAN.

```
console.log(Number("34"));          34
console.log(Number("Dingas"));      NaN
console.log(Number(true));          1
console.log(Number("false"));       NaN.
console.log(Number("a"));           NaN.
```

Functions in Java Script

Function is a block of code which is used to perform a particular task. By using functions we can achieve code reusability and code modularity. function is a keyword which is used to declare functions in JS.

There are 4 types of functions

1) function with argument & with return type

Syntax

```
function functionname(Parameters)
{
    // block of code.
    return "Dinga";
}
```

Ex. function valid(a,b)
{
console.log("Hello");
return a+b;
}
valid("Hello", 3)

2: Function with argument and noreturn type.

Syntax
function functionname(Parameters)
{
// block of code
}

Ex. function valid(a,b)
{
console.log("Hello");
}
valid(4,5)

3: Function with no arguments with returntype

Syntax
function functionname( )
{
// block of code
return "Hello";
}

Ex
function valid( )
{
console.log("Hello");
return 20+30;
}

4: Function with no arguments and no return types.

Syntax
function functionname( )
{
// block of code
}

```
function valid()
{
console.log("Hello");
}
```

In JS we can assign the functions to the variables, we can return the function
we can pass function as a argument.

```
function valid(a, b=2)
{
console.log("Hello");
console.log(a+b);
}
valid(2);
```

Output

Hello
4.

Anonymous function

A function which is declared without a name is known as anonymous functions.

```
var c = function()
{
return "Hello"
}
console.log(c);
```

Output

```
f()
{
return "Hello"
}
```

```
var c = function()
{
return "Hello"
}
console.log(c());
```

Output

Hello

Arrow function (or) Fat Arrow (or) Lambda function

Syntax

```
var variablename = () => Statements;
```

Arrow function.

or

```
var variablename = () => { Set of Statements };
```

```
var c = function(a)
{
return(a);
}
console.log(c(45));
var d = (e) => {e};
console.log(d(7));
var d = (e,f) => {console.log(e+f); console.log(e+f);}
d(7,8);
```

output
45
7

output
15
15

Arrow function is used to make our code more simple (concise) & it is declared by using anonymous function.

If a function is returning only one statement then no need to use return keyword.

If there is no parameter parenthesis is mandatory.

If there is only one parameter parenthesis is not mandatory.

If there are more than one parameter then the parenthesis is manda

If the function contains only one statement then the flower brack are not mandatory.

If the function contains more than one statements then flower brackets is mandatory.

4/9/19

```
var a = 20;
function valid(){
 var a = "dinga";
 console.log(a);
}
valid();
console.log(a);
```

output
Dinga
20

IIFE: Immediately Invoked Function Expression.

Syntax

```
(function(a)
{
 // Block of code
})(a));
```

Ex

```
(function()
 {
  console.log("Hello Hi Bye Bye");
 })());

(function(a,b)
 {
  console.log(a+b);
 })(9,9));
```

The self invoking anonymous function is known as immediat invoked function expression.

Passing function as a parameter

```
function valid(a,b)
{
  console.log(a+b())
}
valid(5, function(){
          return 10;
        })
```

output

15

Function as a return statement

```
function valid()
{
 return function a()
  {
   console.log("Hello");
  }
}
var b = valid();
b();
```

output

Hello.

Event Handlers

① Button Events

1. onclick()
2. ondblclick()

② Input Events

1. onblur()
2. onselect()
3. onfocus()
4. onsubmit() → inside form tag

③ Mouse Events

1. onmouseover()
2. onmouseout()

```
function valid(a)
{
 a.style.backgroundColor = "red"
// alert("Hello");
}
```

```
<html>
<head>
</head>
<body>
<button onclick = "valid()"> click </button>
<button ondblclick = "valid()"> clear </button>
<span onmouseover = "valid()"> Dinga </span>
<span onmouseout = "valid()"> dinga </span>
<input type = "text" onblur = "valid()" placeholder = "Onblur">
<input type = "text" onselect = "valid()" placeholder = "onselect">
<input type = "text" onfocus = "valid(this)" placeholder = "onfocus">
</body>
</html>
```

Date methods

```
var date = new Date();
document.write('${date}<br/>');

var year = date.getFullYear();
document.write('${year}<br/>');

var month = date.getMonth();
document.write('${month}<br/>');

var day = date.getDay();
document.write('${day}<br/>');

var hours = date.getHours();
document.write('${hours}<br/>');

var min = date.getMinutes();
document.write('${min}<br/>');

var sec = date.getSeconds();
document.write('${sec}<br/>');

var ms = date.getMilliseconds();
document.write('${ms}<br/>');

var dt = date.getDate();
document.write('${dt}<br/>');
```

Event is an ocurance of any happening by the user on the web page.

Event handler - It is a program which is used to handle the events by using functions.

```
function watch()
{
  var date = new Date();
  var hours = date.getHours();
  var min = date.getMinutes();
  var sec = date.getSeconds();
  var time = '${hours}:${min}:${sec}';
  document.getElementById("i").innerHTML=time;
}
SetInterval(watch, 1000)
```

5/9/19.

```
<html>
<head>
</head>
<body>
<form action = "dinga.html" onsubmit = "return valid()">
username : <input type = "text" placeholder = "Enter valid username"><br>
</form>.
<script src = "app.js"></script>.
</body>
</html>.
```

index.html

```
<html>
<head>
</head>.
<body>
Welcome to Jspiders
</body>
</html>.
```

dinga.html

```
function valid()
{
var res = confirm("press OK or cancel");
  if(res == true)
  {
   return true;
  }
  else
  {return false;
  }
}
```

app.js

Arrays

Syntax var arrayname = [value1, value2 ---- valuen]

Ex var a = ['Dinga', 22, 48.99, true, null, undefined, {a:25}]

An array is a collection of hetirogenious elements. that means we can store any type of data in an array.

In JS array there is no fixed size.

The main advantage of JS array is we can completely remove the element from array. as well as we can add new elements to array where ever we want.

```
var a = ["dinga", 34, 567.999, null, undefined, true];
document.write('${a} <br>');
document.write('${a.length} <br>');
document.write('${a[0]} <br>');
document.write('${a[7]} <br>');
document.write('${a[0][2]} <br>');
document.write('${a[1][1]} <br>');
a[2] = 'Sheela';
document.write('${a} <br>');
```

output

dinga, 34, 567.999, , true
6
dinga
undefined
n
undefined.
dinga, 34, Sheela, , true

Array Methods

1. Push (arg1, ----)
2. unshift (arg1, ---)
3. pop ()
4. shift ()
5. join ("arg1")
6. concat (arg1, ----)
7. reverse ()
8. indexof (arg1, arg2)
9. lastIndexof (arg1)
10. Slice (arg1, arg2)
11. Splice (arg1, arg2, (arg3))

```
var a=["dinga", "Leila", "Laila", "dinga", "dingi"];
document.write('${a} <br>');
document.write('${a.length} <br>');
var b = a.push ("Jsp")
document.write('${a} <br>');
document.write('${b} <br>');
var c = a.unshift("Qsp")
document.write('${a} <br>');
document.write('${c} <br>');
var d = a.pop();
document.write('${a} <br>');
document.write('${d} <br>');
var e = a.shift();
document.write('${a} <br>');
document.write('${e} <br>');
var z = a.join("111");
document.write('${z} <br>');
var x = a.concat(["monu", 23])
document.write('${x} <br>');
document.write('${a.reverse()} <br>');
var m = a.indexof('dinga')
document.write('${m} <br>');
var n = a.indexof('dinga', 2)
document.write('${n} <br>');
var y = a.indexof('majnu')
document.write('${y} <br>');
var q = a.lastIndexof('dinga')
document.write('${q} <br>');
```

```
var q = a.lastIndexof("Dinga")
document.write('${q} <br/>');
var z = a.slice(2,5)
document.write('${z} <br/>');
var y = a.slice(1)
document.write('${y} <br/>');
```

Splice (arg1, arg2, (arg3, ...))

- arg1 → index value
- arg2 → numbers of elements to be removed
- arg3 → new elements to be added

```
var z = a.Splice(2,0,"Majnu");
document.write('${a} <br/>');
document.write('The removed element: ${z} <br/>');
var x = a.Splice(3,2,"Jsp");
document.write('${a} <br/>');
document.write('The removed element: ${x} <br/>');
```

Output:

dinga, leela, laila, dinga, Dingi, majnu.
6
dinga, Leela, majnu, laila, dinga, Dingi, majnu
7
TRE:
dinga, Leela, Majnu, jsp, Dingi, Majnu
TRE: Laila, dinga

6/09/19

push() - push method always inserts the element at the end. It always returns the updated array length.

unshift() - It always inserts the element at the starting of an array. It always returns the updated array length.

pop() - It removes an element from end of an array and it returns removed element from the array.
shift() - It is used to remove the elements from the starting of an array and it returns removed elements from array.
slice() - Accepts two arguments where as first argument is starting index value and second argument is last index value. It is used to cut the array based on index values but the last index will be vomitted. If we aren't passing second argument then it cuts the array till the end from the starting index.
join() - It accepts one string argument & it is used to divide the array based on the argument passed
concat() - It is used to join two arrays and accepts 'n' number of arguments.
indexof() - It accepts two arguments where as first argument is array element. Second argument is index value.
It is used to search the index value of the particular element.
It always searches the first occurences index value if we aren't passing second argument
If the element is present it returns the index value of the particular element else it returns -1.
lastIndexof() - It accepts one argument and it always searches the last occurance index value. If the element is present it returns the index value of the particular element. else it returns -1
reverse() - It is used to reverse the array elements.
splice() - It accepts 'n' number of arguments where as first argument is index value and second argument is number of elements to be removed and from the third argument the new elements to be added to an existing array.

String
1. touppercase()
2. tolowercase()
3. slice(args, arg2)
4. substring(args, arg2)
5. substr(args, arg2)
6. replace(args, arg2)

⑦ includes(arg1)
⑧ endswith(arg1)
⑨ startswith(arg1)
⑩ indexof(arg1, arg2)
⑪ lastIndexof(arg1)
⑫ concat(arg1, ...)
⑬ charAt(arg1)
⑭ charcodeAt(arg1)
⑮ string.fromcharcode(arg1)
⑯ repeat(arg1)
⑰ split(arg1)

String is a collection of characters, we can represent strings in a single quotes, double quotes and backticks.
Strings which is represented by using single & double quotes is known as string literals.
String which is represented by using backticks is known as template literals

Ex

```
var a = "Ispiders Qspiders";
var b = "Sheela";
document.write('${a.length} <br/>');                    17
document.write('${a.toupperCase()} <br/>');             ISPIDERS QSPIDERS
document.write('${a.toLowerCase()} <br/>');             ispiders qspiders
document.write('${a.slice(2,7)} <br/>');                pider
document.write('${a.slice(-2,7)} <br/>');
document.write('${a.slice(-12,7)} <br/>');              er
document.write('${a.substring(2,7)} <br/>');            pider
document.write('${a.substr(-2,8)} <br/>');              piders Q
document.write('${a.replace("S","JSP")} <br/>');        JJSPpiders Qspiders
document.write('${a.includes('S')} <br/>');             true
document.write('${a.includes("Z")} <br/>');             false
document.write('${a.includes("S")} <br/>');             false
document.write('${a.endswith("S")} <br/>');             true
document.write('${a.endswith("J")} <br/>');             false
```

document.write('${a.startswith("$")}
'); false
document.write('${a.startswith("J")}
'); true
document.write('${a.indexof("e",8)}
'); 19
document.write('${a.indexof("e")}
'); 5
document.write('${a.indexof("z",8)}
'); -1
document.write('${a.lastIndexof("e")}
'); 14
document.write('${a.lastIndexof("z")}
'); -1
document.write('${a.concat("Dinga")}
'); Jspiders Qspiders Dinga
document.write('${a.concat(b)}
'); Jspiders Qspiders Sheela
document.write('${a.repeat(3)}
'); Jspiders Qspiders Jspiders Qspiders Jspiders Qspiders.
d.w('${a.split("s")}
'); J,pider,Q,pider
d.w('${a.split("i")}
'); Jsp,ders,Qsp.
d.w('${a.charAt(9)}
'); Q
d.w('${a.charCodeAt(9)}
'); 81
d.w('${string.fromcharcode(81)}
'); Q
d.w('${string.fromcharcode(65)}
'); A.

Object in Java Script

Object is the collection of variables and functions that are represented as key-value pairs.

In JS object can be created in three ways

1) Using newOperator
2) Using literals.
3) Using constructor function.

1. Using new Operator

Syntax `var variablename = new Object();`

Eg
```
var student = new Object();
d.w(Student);
Student.name = "Dinga";  (or) student["name"] = "Dinga";
student.age = 20;  (or) student['age"] = 20;
Student.marks = 75;  (or) student["marks"] = 75;
```

to delete attribute

delete student.name;

app.js

```
var student = new Object;
d.w('${student}<br/>');
d.w('${student.name}<br/>');
student.name = "Dinga";
student.age = 26;
student.marks = 75;
student["gender"] = "Male";
d.w('${student.name}<br/>');
d.w('${student["age"]}<br/>');
d.w('${student["marks"]}<br/>');
d.w('${student.gender}<br/>');
delete student.age;
d.w('${student.age}<br/>');
```

Output

```
Object Object
undefined
Dinga
26
75
Male
undefined
```

2. Using literals

Syntax `var objectname = { key1: value1, key2: value2, --- }`

```
var student = {name: "Dinga", age: 34, marks: 35, gender: "Male"}
d.w('${student}<br/>');
d.w('${student.name}<br/>');
d.w('${student.dept}<br/>');
student.dept = "CSE";
d.w('${student.dept}<br/>');
d.w('${student.age}<br/>');
delete student.age;
d.w('${student.age}<br/>');
```

Output

```
[object Object]
Dinga
undefined
CSE
34
undefined.
```

For-in loop

Syntax
```
for (temp variablename in objectname)
{
  ===
}
```

```
var student = { Name:"Dinga", age: 34, marks: 35, gender: "Male"};
  for( var a in student)
    {
      d.w ('${a}: ${student[a]} <br/>');
    }
```

output

name: Dinga
age: 30
marks: 35
gender: Male.

3. Using Constructor function

.js

```
function student (Name, age, Marks)
 {
   this.Name = Name;
   this.age = age;
   this.marks = Marks;
 }
var S1 = new student("Dinga", 21, 75);
var S2 = new student ("Dingi", 21, 85);
 for( var a in S1)
   {
    d.w('${a}: ${S1[a]}');
   }
```

output

name: Dinga
age: 21
Marks: 75

```
var a = ["Dinga", "Sheela", "Leela", "Laila", "Dingi"]
if (a.indexof("Dingi") == -1)
  {
    a.push("Dingi");
  }
else
{
d.w('Element already present');
}
d.w('${a} <br/>');
```

output - Dinga, Sheela, Leela, Laila, Dingi.

DOM -

* DOM stands for Document Object Model
* The html element document will be converted as a tree structure at the time of execution is known as DOM
* DOM is used to apply the html elements and css properties dynamically
It is also used to acces html document dynamically by using Javascript code

① document.querySelector("Tagname, Classname, Idname");
② document.getElementsByTagName("TagName");
③ document.getElementsByClassName("ClassName");
④ document.getElementById("IdName");
⑤ document.querySelectorAll("Tagname, classname, Idname");
① document.querySelector("Tagname, classname, Idname)

app.js

```
var res = document.querySelector("p");
res.style.border = "3px solid red";
var res1 = document.querySelector(".a");
res1.style.border = "3px groove lime";
var res2 = document.querySelector("#c");
res.style.cssText = "font-family: algerian; // to apply multiple
font-size: 30px; border: 5px dashed orange" properties at a time
```

② document.getElementsByTagName("Tag Name")

```
var res = document.getElementsByTagName("p");
for(var i=0; i<res; i++)
 {
  res[i].style.fontsize = "50px";
 }
```

③ document.getElementsByClassName("Class Name")

```
document.getElementsByClassName("a");
for (var i=0; i<=res.length; i++)
 {
  res[i].style.fontsize = "50px"
 }
```

④ document.getElementById("Id Name")

```
document.getElementById("c");
res.style.border = "5px solid red"
```

⑤ document.querySelectorAll("TagName, ClassName, IdName");

```
var res = document.querySelectorAll("#c");
for(var i=0; i<= res.length; i++)
 {
  res[i].style.cssTxt = "font-size: 50px; border: 2px dotted tomato"
 }
```

① → It selects the html element based on classname, Idname, Tagname

→ In document.querySelector we have to indicate classname by using . operator and id by #.

→ If the element is present it returns the first occurence of the element.

→ If the element is not present, it returns null.

② document.getElementByTagName.

→ It selects the element based on TagNames.

→ It returns all the elements in the form of HTML collection.

→ If the element isn't present, it returns empty html collection.

③ document.getElementsByClassName

→ It selects the elements by className
→ If the class is present it returns all the classes along with elements in the form of HTML collection.
→ If the class is not present it returns empty HTML collection

④ document.getElementsById

→ It selects the elements based on IdName
→ It always selects the first occurance Id along with HTML elements
→ If the Id is not present it returns null

⑤ document.querySelector

→ It selects the elements based on Tag Name, Class Name, Id Name
→ In this we have to indicate the class Name by dot (.) Operator and Id Name by (#)
→ If the element is present it returns all the html element in the form of Nodelist.
→ If the element is not present it returns empty nodelist

Adding Multiple Classes

```
res = document.querySelector("p")
<p class = "b">JSP</p>
res.className = "a"
res.className += "b"
```

Adding Classes [Classlist = Add, Remove, Toggle]

→ res = document.querySelectorAll("p")
res[2].classlist.add("d") → add the class
res = document.querySelector("p")
res.classlist.remove("a") → remove the class

→ res = document.querySelector("p")
JSP
res.classlist.toggle("a")
false
res = document.querySelector("span")

 JSpiders
res.classlist.toggle("d")
true.

+ Class Name is used to add the new classes to the HTML elements by using DOM.
+ Class list contains three methods add(), remove(), toggle()
+ Add() is used to add the classes.
+ Remove() - is used to remove the classes.
+ Toggle() - If the class is already present then toggle() removes the class & returns false. If the class is not present then toggle() add the class and returns true.

(*) res = document.querySelector("p")
 JSP .
res.setAttribute("align", "center") + attribute will be added

Note - Scope of the attribute is limited i.e we should pass specific attributes within specific tags.

(**) res = document.querySelectorAll("p")
NodeList(3) [p, a,
res[0].setAttribute("class", "b")
res[1].setAttribute("align", "center")
res[2].getAttribute("align");
null
res[1].getAttribute("align");
"center"

setAttribute() is used to add new attributes to the html tags
It accepts two parameters where as first parameter attribute name, & second parameter is attribute value
If the attribute is not present, set attribute method adds the attribute to the tag.
If the attribute is present setAttribute() reinitialize the value of existing attributes.

getAttribute()

It accepts one parameter

If the attribute is present it returns the attribute value else it returns null

The parameter is attribute name

Collections in JavaScript

① Map() } types of collection.
② Set()

Map() & Set() are the part of javascript collection

It is also known as datastructure of javascript

Map()

Syntax

var variablename = new Map([["key", "value"],["key", "value"]])

Ex

var student = new([["Name", "Dinga"], ["age", "26"]])

① size
② set(arg1, arg2)
③ get(arg1)
④ delete(arg1)
⑤ has(arg1)
⑥ values()
⑦ key()
⑧ entries()
⑨ clear()

var student = new Map([["name", "Dinga"],["age", 45],["marks", 78],
["gender", "Male"]])

d.w('${student}
');

d.w('${student.size}
'); → 4

d.w('${student.get("marks")}
'); → 78.

d.w('${student.get("dept")}
'); → undefined

student.set("dept", "CSE")

```
d.w('${student.size}<br>');  → 5
d.w('${student.get("dept")}<br>');  → CSE.

var c = student.delete("marks");
d.w('${student.get("marks")}<br>');  → undefined.
d.w('${c}<br>');  → true

var d = student.delete('college");
d.w('${d}<br>');  → false
d.w('${student.has("marks")}<br>');  → false
d.w('${student.has("name")}<br>');  → true.

var p = student.values();
d.w('${p.next().value}<br>');  → Dinga.
d.w('${p.next().value}<br>');  → 45
d.w('${p.next().value}<br>');  → male

var q = student.keys();
d.w('${p.next().value}<br>');  → Name
d.w('${p.next().value}<br>');  → Age
d.w('${p.next().value}<br>');  → Gender

var r = student.entries();
d.w('${p.next().value}<br>');  → name, Dinga
d.w('${p.next().value}<br>');  → age, 45
d.w('${p.next().value}<br>');  → gender, male
d.w('${student.size}<br>');  → 4.
student.clear();
d.w('${student.size}<br>');  → 0
```

for of loop

Syntax

```
for(var Temp-variable of Iterable.object)
{
  //code
}
```

⊛

```
var student = new Map();
d.w('${student.size} <br>');
Student.set("name", "Dinga");
student.set("age", 34);
Student.set("marks", 90);
student.set("dept", "CSE");
student.set("gender", "Male");
d.w('${student.size} <br>');
d.w('--- Printing only keys --- <br>');
for(var temp of Student.keys())
{
d.w('${temp} <br>')
}
d.w('Printing only values ---- <br>');
for(var temp of student.values())
{
d.w('${temp} <br>');
}
d.w('-- Printing only entites ---<br>');
for(var temp of Student.entries())
{
d.w('${temp} <br>')
}
d.w('--- for in loop started ... <br>')
for(var temp in Student.values())
{
  d.w('${temp} <br>')
}
```

Map is a building object in JavaScript

Map is used to store the data in the form of key & value pairs.

size - It is used to findout the current size of the map

set() - set attribute accepts two parameters where as first parameter is key and second parameter is a value.
It is used to add new elements to the existing map.

get() - It accepts one argument and that argument is key. It is used to get the values based on keys.

delete() - It accepts one argument and the argument is key. It is used to delete the elements from the map based on keys.
If the key is present it delets the value of that particular key & returns true.
If the key is not present it returns false

has() - It accepts one argument & the argument is key. It is used to check whether the keys are present or not in the map.
If the keys are present it returns true else it returns false

values() - It is used to fetch the values from the map.

keys() - It is used to fetch the keys from the map.

entities() - It is used to fetch both key & value from the map.

clear() - It is used to clear all the data from the map.

* For of Loop - It is used to traverse the iterable objects

Set() -

Syntax

```
var variableName = new Set([value1, ....])
```

Ex

```
var student = new Set(["Dinga", 45, 78.99, "CSE"]);
```

Methods of Set:

1. add(arg1)
2. delete(arg1)

③ clear()
④ size
⑤ has(arg1)
⑥ values()
⑦ entries()

```
var student = new set(["Dingi", "Female", "ECE", 34, 99.99]);
d.w('${student}<br> ---- <br>');
d.w('${student.size}<br> ---- <br>');
d.w('${student.delete("PESIT")}<br> ---- <br>');
student.add("PESIT");
d.w('${student.size}<br> ---- <br>');
d.w('${student.delete("PESIT")}<br> ---- <br>');
d.w('${student.size}<br> ---- <br>');
student.add("ECE");
d.w('${student.size}<br> ---- <br>");
d.w('${student.has("Dingi")}<br> ---- <br>');
d.w('${student.has("Dinga")}<br> ---- <br>');
var q = student.values();
d.w('${q.next().value}<br> ---- <br>');
d.w('${q.next().value}<br> ---- <br>');
var p = student.entries();
d.w('${q.next().value}<br> ---- <br>');
d.w('${q.next().value}<br> ---- <br>');
d.w('${student.size}<br> ---- <br>');
student.clear();
d.w('${student.size}<br> ---- <br>');
```

Output

[Object Set]

5

---

false

6
true
5
5
true
false
Singi
Female
Singi , Singi
Female, Female
5
0

```
d.w('.... only values by using values() ---- <br>');
for(var temp of student.values())
{
 d.w('${temp} <br>');
}
d.w('... only by using entries() .. <br>');
for(var temp of student. entries())
{
d.w('${temp} <br>');
}
```

Set is inbuilt object in javascript which is same as array & it won't allows duplicates

① add(arg1) - It accepts one argument and the argument is the value. It is used to add the elements

② delete(arg1) - It accepts one argument and the argument is the value. It is used to delete the elements from the map based on value.

③ clear() - It is used to clear all the data or elements from the map.

④ size() - It is used to find the current size of the map.

⑤ has(arg1) - It accepts one argument and the argument is the value.
It is used to check whether the value is present or not in the map

If the value is present it returns true else it returns false

⑥ values() - It is used to fetch the values from the map.

⑦ entries() - It is used to fetch both key and values from the map.

Eventbubbling & Event capturing

These are two ways of event propagation in HTML DOM API. When an event occurs in an element inside another element and both the element have registered a handle for that event. The event propagation mode determines in which order the elements receives the event.

→ with bubbling, the event is first captured and handled by the innermost element and then propagated to outer element.

→ with capturing the event is first captured by the outermost element & propagated to the inner elements.

Capturing is also called "tricking" which helps to remember the propagation order

We can use addEventListener(type, list

to register event handler, for in either bubbling (default) or capturing mode. To use the capturing mode pass third argument as 'true'.

Example

```
<div>
    <ul>
        <li> </li>
    </ul>
</div>
```

hasAttribute()

```
ss.hasAttribute("align")
```

true.

It checks whether the code has that particular attribute or not. If it has that attribute it will return true or else it will returns false

className - is only to add new classes to the elements.

11/09/19

Class in JS

Syntax - class classname
{
code to be executed
}

```
class Demo
{
constructor(a)
{
 d.w(a)
}
}
var obj=new Demo(2);
```

ex
```
class Demo
{
 constructor()
  {
   document.write("Hello from Demo<br>");
  }
}
class Sample
{
 constructor(a)
  {
   d.w('${a} <br>');
  }
}
var d1 = new Demo();
var s1 = new Sample();
```

Output:
Hello from Demo
undefined.

```
class Demo
{
 var c = "Dinga";
 constructor(a)
 {
  d.w('${a} <br>');
 }
}
var d1 = new Demo(20);
```

output
unexpected identifier.

constructor()
{
}

```
class Demo
{
constructor(a)
{
 d.w('${a} <br>');
}
constructor()
{
d.w('Hello <br>');
}
}
var d1 = new Demo(20);
```

Output

A class can not have more than one constructor.

```
class Demo
{
 constructor(a)
  {
   document.write('${a} <br>');
  }
  test()
  {
   document.write('Hello Hi Bye Bye <br>');
  }
  static m1()
  {
   d.w('Hello Hi Bye Bye from m1() <br>');
  }
}
var d1 = new Demo(20);
d.w('${d1} <br>');
d.w('${typeof(Demo)} <br>');
d1.test();
Demo.m1();
```

Output

20.
[object object]
function.
Hello Hi Bye Bye
Hello Hi Bye Bye from m1()

```
class Demo
{
 constructor(a)
  {
   d.w('${a} <br>');
  }
  m1(b)
  {
   d.w('${b} <br>');
  }
```

```
m1()
{
d.w("Hello Hi Eg Eg from m1()<br>");
}
}
var d1 = new Demo(20);
d1.m1(3);
```

Output
20
Hello Hi Eg Eg from.

Class is a blue print of objects or collection of objects.
In JS class can be created by using class keyword. In JS constructor can be created by using constructor keyword. Constructor is used to initialize the variables. In JS class we should not create more than one constructor.
In JS class we should not declare the variables. In JS class methods can be created only by name. & there is no method overloading

Inheritance

```
class Demo
{
 constructor()
 {
  document.write("constructor from demo<br>");
 }
}
class Sample extends Demo{
 constructor()
 {
 document.write('constructor from Sample<br>');
 }
}
// var d1 = new Demo();
var s1 = new Sample();
```

Output
Constructor from sample

```
class Demo
{
 constructor()
  {
   d.w('constructor from demo <br>');
  }
 m1()
 {
  d.w('Hello Hi Bye Bye from demo <br>');
 }
}
class Sample extends Demo
 {
  constructor()
   {
    super();
    d.w('constructor from sample <br>');
   }
   m1()
   {
    d.w('Hello Hi Bye Bye from sample <br>');
   }
}.
var s1 = new Sample;
 s1.m1();
```

Output

Constructor from demo.
Constructor from sample
Hello Hi Bye Bye from sample

Inheritance is a process of acquiring properties from super class to sub class or parent to child class.
Inheritance can be achieved by using extends keyword
Whenever we are extending super class it is mandatory to call super ~~call~~ class constructor inside subclass constructor by using super() method.

```
function outer()
{
 console.log("outer event");
}
function inner()
{
console.log('inner event');
}
var res = document.getElementById("d");
var res1 = document.getElementById("b");
res.addEventListener("click", outer, true);
res1.addEventListener("click", inner, true);
```

app.js

```
<html>
<head>
</head>
<body>
 <div onclick="outer()" id="d">
    <button onclick="inner()" id="b"> click</button>
</div>
<script src="app.js"></script>
</body>
</html>
```

Output

```
Outer event
inner event
inner event
outer event
```

The event which is occuring from inner element to outer element is known as event bubbling.

The event which is occuring from outer element to inner element is known as event capturing

Font-awesome Snippet

Animate.css snippest.

Material icon theme

Bootstrap4 snippets.

HTML5 snippets.

CSS3 snippets

cmd

code.

control / → for commenting

./ → to open files in folder, like to add the path

every web page contains 12 columns.

① Container