12/08/19

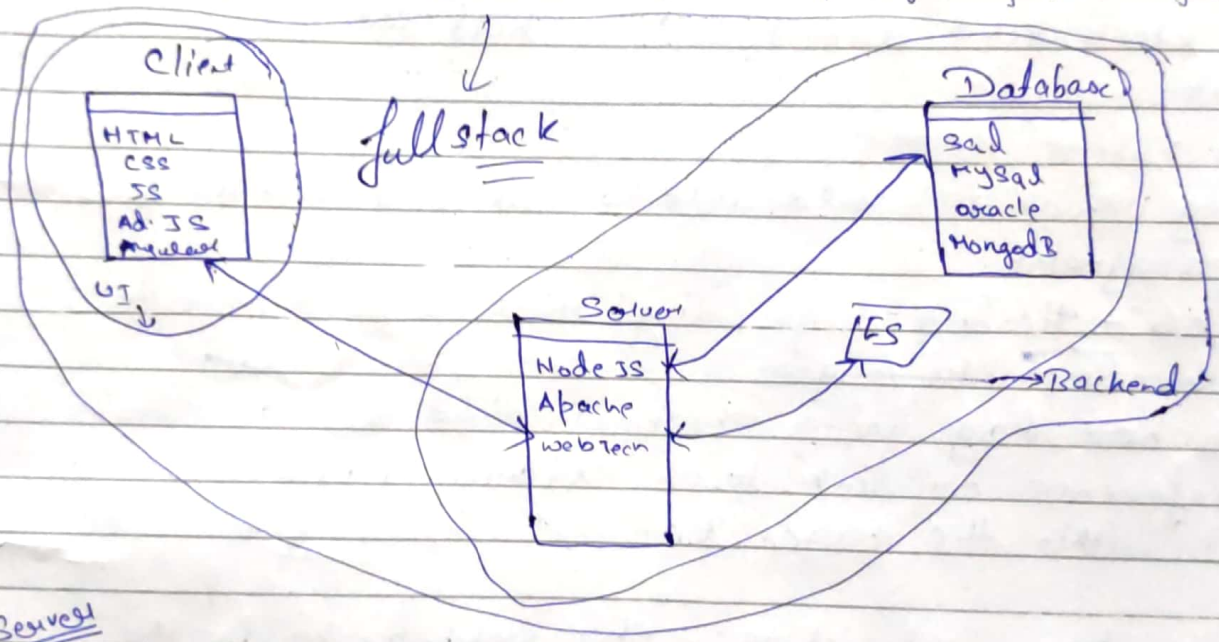
HTML and CSS are static programming
JS is dynamic programming. It is client server.
Jquery is library for JS.
HTML5 and CSS3 are newer version of HTML & CSS.
Bootstrap are used for responsive updates. (extension)

It is UI Technology or Client side

Node JS is server side technology or backend.
Mango DB is database. It is use structure of JSON.

14/08/19

UI is client side technology {(HTML, CSS, JS)
Backend is database & server {(Node JS) (server
Stack developer is Backend & UI {(Sql, MySql, MongodB) (data



Server

We can't store data in server. It only hold the resources. Server is capable of process the resources.

Client side don't store data.

Database is a place where we organize the data.

- Angular JS (1.x) - Javascript (It is client side)
- Angular (2 to 7) - {Typescript (It is a superset of JS). It may use JS or TS. It is created by Microsoft. It is client side}

ECMA is standards for programming language

16/08/19

Framework is also like library. It has some setup & rules. But library don't act like framework.

16/08/19

HTML:- (Hypertext Markup language) :- Every define language using which we could able to add content or resources within the webpage. It comes with set of predefine tags using which we could add the content on the page accordingly.

- html:- which holds the complete webpage content
- head:- holds the page title and external resources (js files, css, metatag etc).
- title:- using which we could able to add title to the webpage
- div:- A block level element, used to hold relevent content into it.
- Span:- Inline element.
- p:- using which we could able to add content in the form of paragraph.
- h1 to h6:- Heading tags using which we could able to add headings to pages.
- a:- Anchor tag using which we could able to create an reference or link to an external page or DOM element within the page. etc.

* Web page structure:- Any webpage holds the following structure:

```
<html>
  <head>
    <title> .... </title>
  </head>
  <body>
    .... actual page content.
  </body>
</html>
```

- (View source-code - ctrl+U) (Indecation is important)
- Extension is important
- Debugger element

19/08/19

HTML & CSS (class-1)

 - Break the line
  - paragraph content

Html Attributes:- Attributes are through which we could able to add extra information to any html tags.
Following are the predefine Html attributes can be added to html elements:-

- id:- using which we could able to add an unique identity to elements.
- name:- can add a name to element, which could be duplicate
- title:- Title text which will be shown while over on element.
- alt:- alternative text
- class:- Using which we could able to add single or multiple class name to a tag.
- style:- Using which we could able to add single or multiple inline css properties. etc.

* CSS (Cascading Style sheets):- The only way through which we could able to add beautification or look & feel changes to a webpage is through CSS. It comes with 'n' number of properties where individual properties are used to change the corresponding look & field.
Following are some of the CSS properties:-

- font-size:- Through which we could able to control the size of font been displayed on page.
- width / height:- Using which we could able to control the dimensions of a block level element.
- display:- Through which we could able to change the default display type of an element.
- opacity:- using which we could able to change the transparency level of an element.
- z-index:- To control the z axis ~~over~~ order for not static elements.

Rendering :- Html content getting from the code

19/08/19

- position :- To change position of element to fall under 2-axis.
  etc.

Note :- style / class, Is the two ways we could able to inject set of css properties to an element, among which class is always recommended.

Syntax :-

Adding css through style tag :-

< tagname style = " cssproname : value; cssproname2 : value; ...
.... content
</ tagname >

- Adding css through class attribute:

< tagname class = " classname1 classname2 ... " >
.... content
</ tagname >

20/08/19

HTML & CSS (class - 2nd)

* Different ways of applying css to HTML elements :-

(i) inline css (ii) css through classes (iii) css using external file.

(i) inline css :- "style" is a predefine attribute been supported in html using which we could able to inject any no. of css properties to a single element.

Syntax :-

< tagname style = " cssproperty name : value; ^ " >
....
</ tagname >

20/08/19

eg:-

```
<b style = "background: green; color: blue; font-size:
        20px; font-family: arial">
    ....
</b>
```

21/08/19 [Class 3rd]

(ii) CSS through class:-

i) The process of defining set of relative css properties as a block been assigned with a user defined name, which can be injected to any html element within the page is called a css class.

ii) Every class being defined should have a custom name assigned to it.

iii) class name should always start with . delimeter operator

iv) "style" is a predefined tag been supported in html ~~using~~ under which we could also define n number of css classes.

v) "class" is a predefined attribute through which we could able to link one or multiple classes for a single html tag or element

Syntax:-

```
. <classname> {
    ....
    ... // set of css properties
}
```

Example:-

```
<style>
    . content {
        font-size: 20px;
        border: 1px solid;
        ....
        ....
    }
</style>
```

21/08/19

Adding an image to page:-

"img" is a predefined tag using which we could able to add an image resource to a page. It takes a mandatory attribute 'src' through which we could able to specify the path of external image which need to be injected to page.

Syntax:-

```
<img src="<relative/absolute path>" />
```

eg:-

```
<img src="c:/data/personal/test.png" />
<img src="http://www.google.com/sample/data
/car.jpg" />
```

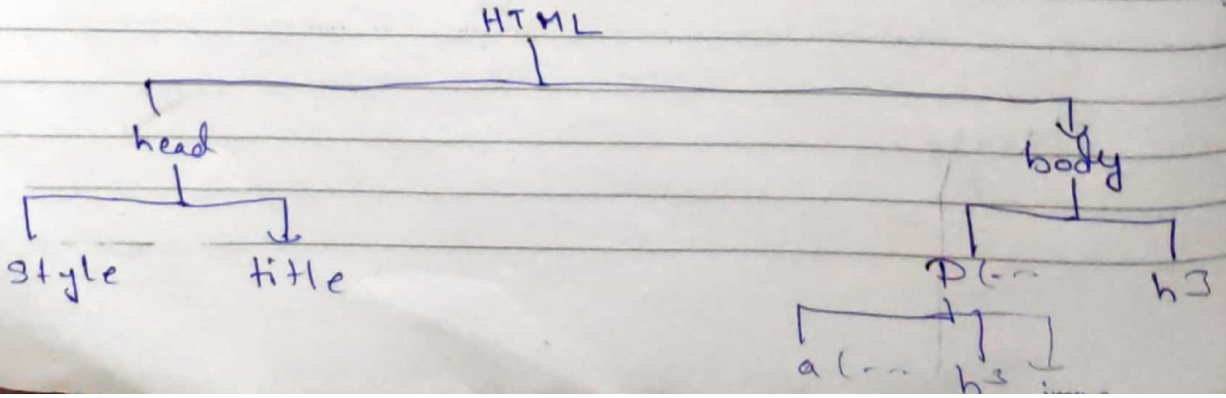
22/08/19 (4th class)

* Different ways of applying css properties to html elements

Following are four different ways we could able to add or inject css properties to html elements:-

(i) Inline css using style (ii) Adding css through classes
(iii) Defining a css class using elements id
(iv) Defining a css class using elements tag name

* HTML DOM (Document object Model) Structure:- The DOM structure of a html page defines the tree structure of the current page in which it holds list of all the parent, child and sibling elements along with their content & attributes it is holding.



(image) - base 64 is resource not path

../ = Take a backspace from the current folder

27/08/19 (5th class)

* ~~Absolute path & Relative path:-~~

* CSS Pseudo classes:- Pseudo classes are special set of classes which doesn't get applied to the DOM elements on load of the page but will be applied based on the current state of the element.

Following are set of predefined pseudo classes ~~being~~ been supported:-

active, focus, hover, first-child, last-child, nth-child(n) etc.

Syntax:-

```
.classname : <pseudo class name> {
    . . . .
    . . . .
}
```

eg:

```
.block : hover {
    . . .
    . . .
}
```

Pseudo Elements:- the special set of css using which we could able to apply the css properties to the html elements content but not to the complete content, we could able to apply the css properties to partial content of element.

:: first-letter, :: first-line, :: after, :: before etc.

Note:- 'content' is the css property, through we could able to ~~content~~ add text content on page. the content been added through 'content' property is not a part of dom structure.

29/08/19 (6th class)

* Inline and Block level Elements :-

All the html elements are been categorized into two types

(i) Inline Elements (ii) Block level Elements :-

*

Block level Elements :- Any DOM element comes under block level category holds the following properties :-

(i) While rendering on the page it automatically comes to a new line
(ii) the element which is following the block level element also automatically comes to new line.
(iii) By default it occupies hundred percentage width of its container.
(iv) Even though it occupies 100% width of its container by default, we can always controls the dimensions of it using css properties width & height.
(v) 'div' is the best example for a block level element.

*** Inline Elements :-** Any DOM element comes under an inline elements category holds the following properties :-

(i) It always tries to render within in the same line.
(ii) the total dimensions of an inline element depends on the content that it is holding.
(iii) We can't control the dimensions of an inline element.
(iv) Even those we add the width & height properties it just skips it.
(v) 'span' is the best example for an inline element.

Assignment - 1

28/02/19

* Different ways of defining css properties to html:-

(i) Creating a css class using id:- In a case an element need to be injected with set of css properties, the properties are oly to this element, it even have an 'id' already been assigned, instead of creating a separate css class we can define a class using the element 'id' itself.

(ii) While defining a css class using elements 'id' it should always starts with '#' operator.

Syntax:-

```
# <element id> {
    . . .
    . . . //set of css properties
}
```

eg:-

```
# sample {
    . . . font-size=20px;
    . . .
}
```

```
<div id="sample"> </div
```

* Creating tagbased css class:- we can even create a css class by using the tagname itself so that no extra attribute 'id' or 'class' need not to be injected to elements.

Syntax:-

```
<element name) {
    . . .
    . . . //set of css properties
}
```

Eg:-

```
div {
    . . font-size=20px;
}
```

```
<div>
</div>
```

29/08/19 (7th class)

* css priority:- If a DOM element been injected with the same css property (different values) using all the four different ways of applying css (inline, id, tagbased & classname). Browser follows the below ~~priority~~ order while applying css to element.

Priority

(i) css been applied through inline always takes high ~~priority~~ in order.
(ii) css been applied through a ~~class~~ id takes second priority.
(iii) css been applied using class it takes third priority in order.
(iv) css been applied using tagname takes the least priority in order.

* debugger tool:- A tool which is by default available in every browser using which we could able to debug local or public webpages within the browser.
Following are the powerful features been supported by any debugger tool:-

(i) Using debugger tool we could able to implement any CURD (create update read delete) operation on any DOM element within the page.
(ii) Using debugger tool we could able to add dynamic changes to the html elements or the corresponding css dynamically while running the page.
(iii) It even provides a feature of debugging javascript code by adding break points, watching variable values, monitoring flow of execution.
(iv) It even provides feature of monitoring any server call or interaction including the total time & size been taken by ~~each~~ individual call.
(v) Using debugger tool we can monitor the performance of a page, add the required changes to increase the performance of it.
(vi) It even helps us to monitor the memory allocation of the current webpage.

29/08/19

Note: Debugger tool can be open for a current webpage by clicking on 'F12' button.

30/08/19

CSS Priority (8th class)

Note:- Any css property which is been assigned using "!important" will always states higher priority in the order irrelevant of the above priority order.

eg:-
```
color: green !important;
font-size: 10px !important;
```

* CSS padding & margin properties:-

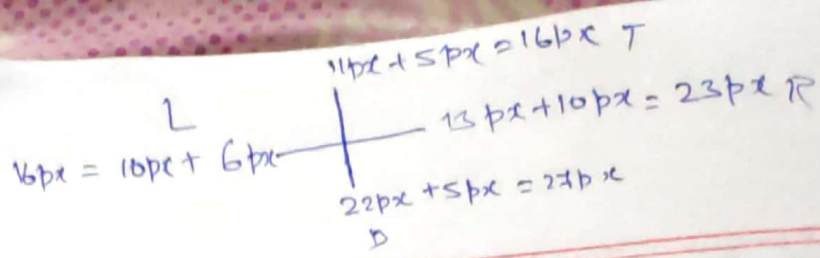
By default any DOM element getting rendered on a page occupies the very next space of the previous element without maintaining any space or gap.

"Padding" and "margin" are the two css properties using which we could able to add space between or within the DOM elements.

* css padding properties:- Using which we could able to add space between the border and content of the container.

Syntax:-

```
padding: 10px; //adds space of 10px between
                border & content of container in all four directions.

padding: 10px 30px; // adds space of 10px on
top & bottom, 30px of space on left & right sides.

padding: 5px 10px 6px 7px;
          top  right bottom left

padding-top: 10px;
padding-bottom: 6px;
padding-right: 10px;
padding-left: 5px;
```



30/08/19

* Css margin property:- Using which we could able to add space or gap ~~around~~ the border of the container. above

Syntax:-

```
margin : 10px; // adds spaces of 10 above border and of container in all four directions

margin : 10px 30px; // adds space of 10px on top & bottom, 30 px of space of left & right sides

margin: 5px 10px 6px 7px;
        top  right bottom left

margin-top : 10px;
margin-bottom: 6px;
margin-left : 10px;
margin-right: 5px;
```

* Css Box Model:- The process of calculating actual & total dimensions been occupied by a container by considering all the below properties is called Box Model.

Border

(i) Space been occupied by container in all directions.
(ii) Actual dimensions been occupied.
(iii) Total margin & padding space been occupied.

05/09/19

(9th class)

* Steps to be followed to install node.js & create a node server using express & express generator modules:-

1) Download and install latest nodejs setup file from https://nodejs.org/en/ (Latest version is 10.16)

2) To check the successful installation of node.js open command prompt & give the command node -v. which throws the current

05/09/19

version of node.js been installed in current machine.

* Following are the step to create a Server using express & express generator node modules:-

1) Install express module by giving the following command within command prompt (npm install express -g) -g global
(node package manager)

2) Install express generator module using the below commands
(npm install express-generator -g)

3) Following is the command through which we create a node server using express module :-

cd..
cd..

Syntax:-

express <server name>

eg:

express webapplication
express testapp
etc.

4) The above command creates a server folder with all the minimum resources needed. ~~Continuation~~
Confirm it by manually going inside the particular drive.

5) Under the command prompt go inside the create a server folder and give the following command to install external dependency node modules:-
(npm install) cd WebApplication

6) Under the folder edit the file with name "app.js", add the following instructions at line 24 to make the server to listen at a particular port number.

```
app.listen(8080, function(){
    console.log("Server is listing at 8080 port");
});
```

05/09/19

(url - uniform resource locator) cmd
then write hostname

7) Under the command prompt go to the server folder give the following command to start the server:-

npm start

To check the successful running of server go to browser and hit the below url which shows the welcome express message. http://localhost:8080

05/09/19 (url - uniform resource locator) cmd then write hostname

localhost:8082/ (system-Name)

7) Under the command prompt go to the server folder give the following command to start the server:-

npm start

To check the successful running of server go to browser and hit the below url which shows the welcome express message. http://localhost:8080

06/09/19 [10th class]

* CSS Float Property:- By default any DOM element gets rendered on the page it tries to occupy the complete left direction by default.

ii) In order to take DOM element to render either to complete left direction or to right direction or to make multiple block level elements to get rendered within the same line we use css float property.

Following are the optional value it takes

float: left / right;

Any DOM element with float property assigned holds the following properties.

(i) DOM elements with float left will render to extreme left, and DOM element with float right will render to extreme right direction.

(ii) Irrelevant of whether block level or inline, multiple continuous DOM elements having changed their default direction using float property, will try to render in the same line.

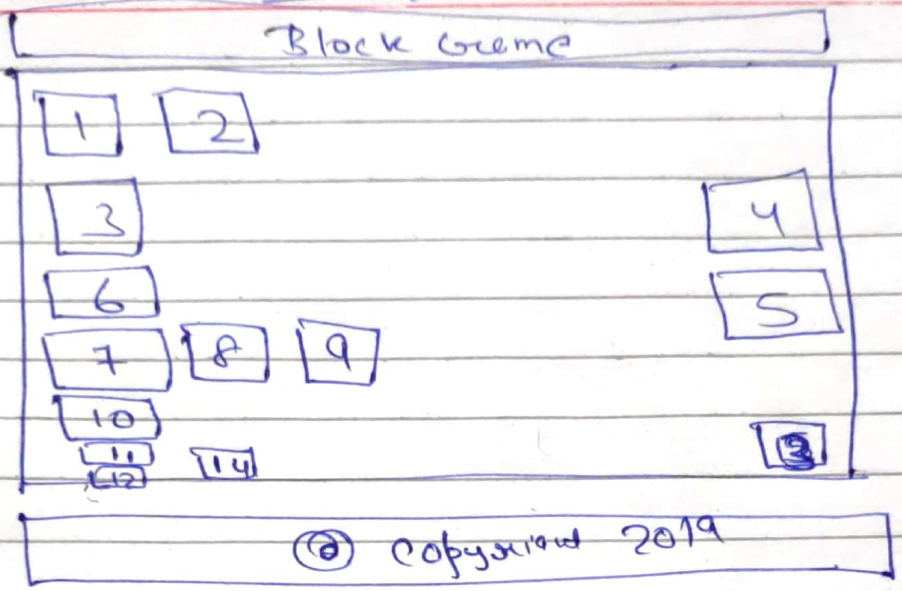
(iii) If a DOM element changes its direction to either left or right, the element which is following it will also try to follow the previous elements direction.

(iv) Css 'clear' is a property through which we can make the DOM element to not to follow the previous element direction, but to follow its own direction.

Syntax:- clear: left / right / both;

06/09/19

Assignment-2



(11th class)

* Adding images as background to containers :-

CSS "background" is the property through which we could able to add an image as background to any container.

Syntax :-

```
background : url("<relative/absolute path of image>");
```

(or)

```
background-image : url("<relative/absolute path of image>");
```

Note : 1. background-repeat (repeat-x, repeat-y, no repeat) is the css property used to control repetiveness of background image.

2. background-position is the css property used to control the position of background image within the container.

→ (10px 20px);

* Css image spriting :- In a case where multiple static images being added in a page, the page has to create 'n' number of calls to the server to load 'n' number of static images.
Every time there is a call to server from client-side, it takes time to get response depends on current application graphic / traffic. The number of interactions to a server increases, decreases the performance of a page.
In order to increase the performance of a page we have to decrease the number of interaction to the server.

07/08/19

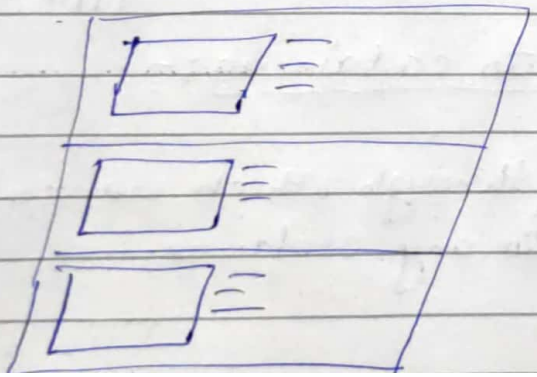
(Use Snipping tool in windows to crop the images)

Image Spriting is one of the concept through which we avoid multiple calls for multiple static images ~~and decrease~~ in step ~~instead~~ we get all the static image in a single call.

The process of combining all the static images to form a single image, showing the required image from multiple images using background-position is called image spriting.

08/08/19

Assignment - 3



07/08/19

(Use Snipping tool in Windows to Crop the images)

Image Spriting is one of the concept through which we avoid multiple calls for multiple static images and decrease in step we get all the static image in a single call.

The process of combining all the static images to form a single image, showing the required image from multiple images using background-position is called image spriting.

Assignment-3

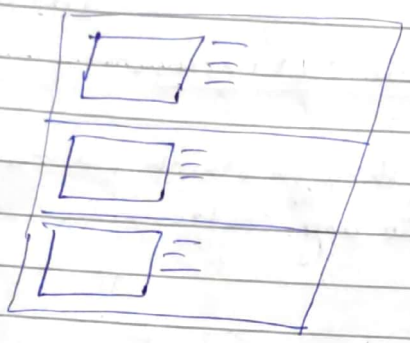


Image only use in png (Key) img extension used it will come out from the background boxes.

09/08/19

* ul tag & ol tag (Unordered list & ordered list):- (12th class)

The two predefined tag used to combine multiple relative items as a block.
'li' is a predefined tag used to add items under ul or ol tag.

Syntax:-

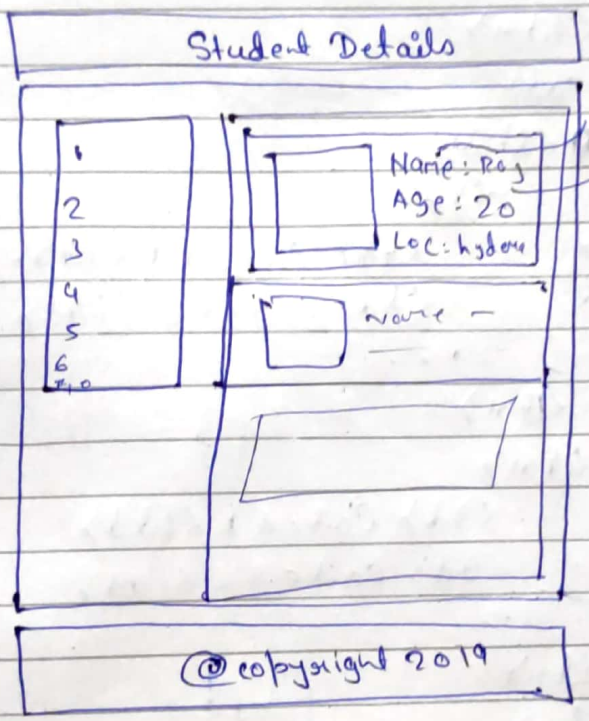
```
<ul>
  <li> ..... </li>
  <li> ..... </li>
</ul>

<ol>
  <li> ..... </li>
  <li> ..... </li>
</ol>
```

09/09/19

Note:- (i) ul & ol tags works almost the same where the only difference is under ol tag all the list items will be by default assigned with a serial numer, Under ul tag all the list items will be assigned with a bullet symbol.

(iii) "list-style" is the css property through which we could control the type of bullet or number being shown for list item

Assignment IV



Different color
Different color

* HTML Tables:- Following are set up of predefined tag been supported in HTML using which we could able to show the content in the form of row-wise & column-wise.

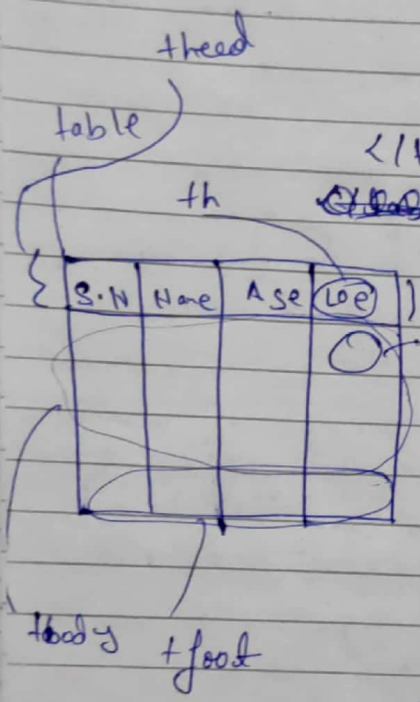
- table - Parent tag used to hold all table content.
- tr - Table row, used to hold an individual row.
- td - Table data, used to hold individual cell data
- th - Table header
- tbody - Used to hold table body
- tfooter - Used to hold table footer
- thead - Used to hold table header

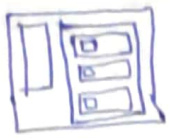
09/09/19

Note:- Adding tbody, tfoot, thead is optional, but it is recommended to add.

Syntax:-

```
<table>                    complete container of table
    <thead>                header
        <tr>
            <th> heading1 </th>
            <th> heading2 </th>
            ...
        </tr>
    </thead>
    <tbody>                body
        <tr>
            <td> Content 1 </td>
            <td> Content 2 </td>
            ...
        </tr>
        <tr>
            <td> Content 1 </td>
            <td> Content 2 </td>
            ...
        </tr>
    </tbody>
    <tfoot>                footer
        <tr>
            <td> ... </td>
            <td> ... </td>
            ...
        </tr>
    </tfoot>
</table>
```





overflow:auto;

09/09/19

Note:- "rowspan" and "colspan" are the two ~~predefined~~ attributes can be added only for 'td' tag using which we could able to merge multiple cells to form a single cell. Both in row-wise & column-wise.

(ii) "cellspacing" and "cellpadding" are the two predefined attributes can be added only for "table" tag using which we could able to gives margin & padding for table.

Note:-

(i) It is recommended to not to use table tags within a page.

(ii) Using the table tag the page/DOM structure spent more time to maintaing/creating the table structure which will decrease the performance of the page.

Assignment V

using table ~~method~~ tags & div tag

S.No	Student Name	Age	Present Location	Permanent Location
1	Raj	20	Hyd.	WB
2	Rohan	21	Mum	SC
3	Rohmi	22	Pune	DNR
4	Neela	23	KOL	HFP

10/09/19 (13th class)

* Creating External CSS File:- It is always recommended to create an external file with all the css classes (file with extension .css).

"link" is a predefined tag through we could include a single css file within a html page.

In a single webpage we can include any number of external css file using multiple link tags.

Syntax:-

<link rel="stylesheet" type="text/css" href="path of external css file">

~~src = stylesheet~~

10/09/19

Assignment VI

table tag

Student Details

to student / sub	SNo	SName	Pic	Location	Marks
	1	RAM		Hyd	

@copy @ Copyright, 2019

* CSS Positions :- Any time if we want to move any DOM element to any required position within the page, can be done using margin & padding properties.

If a DOM element moves to a required position using margin & padding properties, it is actually not moving but the element is increasing its own dimensions.

In order to make/move any DOM element to any required position without increasing dimension we use the following CSS properties :-
(i) top (ii) left (iii) right (iv) bottom

Not all the DOM elements can consider the above css properties but only the DOM elements which are being positioned can consider the above properties.

Css 'position" is a predefined property through which we could able to change the position of any Dom element.

10/09/19

Following are the possible values a position property takes:-

(i) Static (ii) relative (iii) absolute (iv) fixed (v) sticky

* Element with position static:- Every DOM element by default holds the position property with value static.

Any DOM element with position static holds the following properties

(i) Static positioned elements cannot be move from the default position.
(ii) Static positioned elements will not consider top, left, right and bottom properties.
(iii) Even though we assign top, left, right or bottom properties to static elements, it just skips to consider them.

Syntax:-

```
position: static;
```

* Element with position relative:- It holds the following properties:-

(i) It is capable of moving to any required position within the page.
(ii) It is capable of considering properties left, right, top & bottom actual
(iii) ~~By moving~~ When it moves to a new position it never losses the default space been occupied by the element on page load.
(iv) While moving to a new position it always moves relative to its actual default positioned.

Syntax:-

```
position: relative;
```

11/09/19

* Elements with position absolute: Any DOM element with (14th class) the position as absolute holds the below properties:-

4 (Imp topic is position)

11/09/19

In z-axis we can move to

(i) DOM element with position absolute will automatically loose its actual space within default x-y axis.
(ii) It automatically jumps from defaut x-y axis to z-axis.
(iii) While moving to a new position as it doesn't hold its actual position, its moves according to its parents position.
(iv) While depending on the parent position it only depends on the parent who's position is none static.
(v) If the immediate parent position is static, it will traverse to its ancistors until it finds a parent with position none static.
(vi) If none of its parents or ancistors holds the position as static, it depends on the body tag.

Syntax:-

```
position: absolute;
```

* Element with position fixed:- Any DOM element with position fixed is almost like an element with position absolute where the only difference is once the element position is been set it doesn't even move even on scroall.

Syntax:-

```
position: fixed;
```

* Element with position sticky:- Any DOM element with position fixed is almost like a position relative element where the only difference is the sticky element will automatically turns to fixed element when we try to scroall the element out of its viewpoard.
↳ The piece of path what our eye could able to see.

Syntax:-

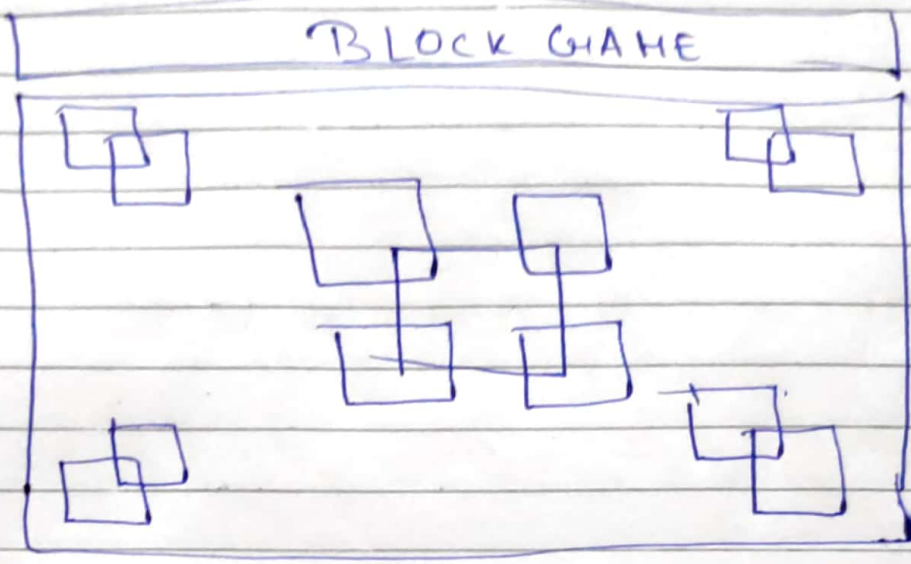
```
position: sticky;
```

11/09/19

(• margin : auto ;)

position : ~~relative~~ absolute
left :
right :

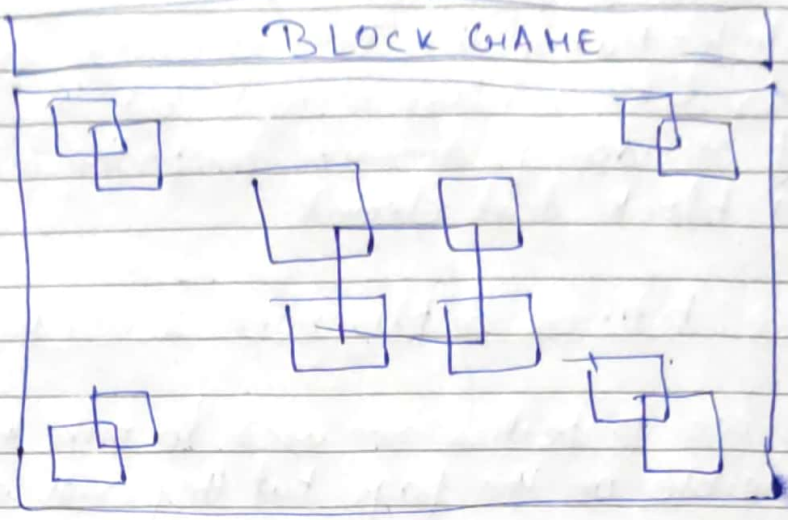
Assignment VI



(• margin: auto;) position: absolute
z-index - if left;
right;

11/09/19

Assignment VII



12/09/19

(15th class)

* CSS z-index property:- A css property through which we could able to control the rendering priority order of a non-static DOM element.

(ii) 'z-index' property takes a numeric value between 1 to any biggest integer.

Syntax:- z-index: 10;

* CSS opacity property:- Through which we could able to control the transperency level of any DOM element.

(ii) It takes the value between 0 to 1.

Syntax:- opacity: 0.6;

* CSS display property:- display is the css property through which we could able to control the default display properties of DOM element while rendering on the page.

(a) Following are the possible values a display property consider:-

(i) none - Makes the dom element to be not visible on page.

(ii) inline - Makes the dom element to get rendered on the page as like a inline element.

(visibity : hidden;)

12/09/19

(iii) block - Makes the DOM element to get rendered on the page as like a block level element.

(iv) inline-block - Makes the DOM element to get rendered on the page as like a inline element, but the element will be capable of considering dimension properties like width & height as like a block level element.

* Difference between display:none & visibility:hidden properties

(i) Both the css properties are used to make the DOM element to be not visible on the page but they exist within the DOM structure.

(ii) The only difference between these two is visibility:hidden makes the DOM element to be not visible on the page but it still occupies its own space on the page, whereas display:none makes the DOM element to be not visible on the page at the same its own
it doesn't even occupy space on the page.

* HTML input elements :- Following are set of predefine Html input elements using which we could able to read data from the user.

(i) We use the corresponding input element based on type and length of data to be read. Following are set of predefined input controllers been supported.

(i) <input type="text"> - Creates a text box container, used to read text type of content.

(ii) <input type="password"> - ~~Creat~~ Used to read sensitive data from user.

(iii) <input type="checkbox"> - Creates a checkbox.

(iv) <input type="radio"> - Creates a radio button.

(v) <select>
<option value="value1"> item1 </option>
<option value="value2"> item2 </option>
...
</select> - Creates a drop down element with multiple values.

12/09/19

(vii) <textarea></textarea> - Used to creates a multiline text box.
(viii) <input type="button" value="value1"> - Creates a button, through which we could able to read the user action.
(viii) <input type="submit" value=Value"> - Creates a button, but can be used only within the ~~form~~ form tag, on click makes the form tag to read all user data and send to specified server.

*Form tag :- A very special tag been supported in html, using which we could able to send user data to server.
It takes two mandatory attributes :-
(i) action
(ii) method

(i) "action" attribute - Using which we could able to specify the path of server, to which data need to be sent.

(ii) "method" attribute - Using which we could able to specify type of communication with servers (secured or non-secured).

• Type of communication while interacting with server :-

(i) While interacting with server, either for sending or receiving data, based on sensitivity of data we need to specify type of communication (secured or non-secured data).

Following are possible ways we could able to specify the type of communication :-
(i) GET communication (ii) POST Communication

(i) GET Communication :- This communication type will be used to send/received Non secured data type. In this type the data will be send to server in the form of url query parameters.

eg :- http://www.sampletest.com/data/user?name=raj&age=20&gender=male

Using label tag

12/09/19 internalization support

2) POST communication:- this communication type is used while sending or receiving sensitive data. In this format data will be added to request header itself, which will not be shown to end user but the server receives it.

13/09/19 (16th class)

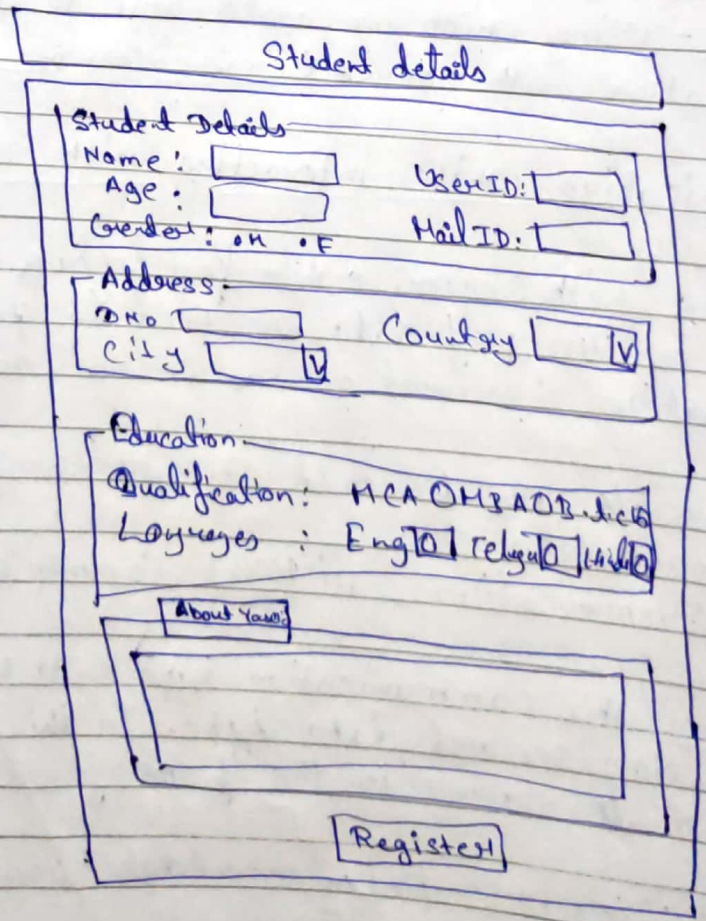
```
<form>
<ul> <li> <label for="uname">Enter Username :</label>
     <input type="text" name="uname">
</li>
```

- url is the one we can identify the server.

16/09/19 limitation (17th class)

- The data validation is not possible in form tag.

(Assignment - VIII)



16/09/19

dynamicness & arithmetic logicness

Javascript (Client Side Scripting Languages)

- Javascript:- A client side scripting language using which we could able to add dynamicness to the web pages & implement arithmetic & logical ~~instructio~~ capabilities to the page.

(ii) Every browser by default comes with individual javascript engines using which javascript instructions gets executed at the client without any dependency of any server.

(iii) It is a super heriac programming language capable of implementing & performing any types of CURD operation on any html element or its corresponding css property.

(iv) It comes with set of predefine objects & methods using which we could able to implement DOM operations within the page.

(v) It provides set of predefine methods & objects using which event listners can be ~~Script handled~~ implemented so that we can handle user action on the pages / within the page.

(vi) It comes with set of predefine objects using which AJAX communication can be established within the page.

* Adding javascript to webpages:-

"script" is a predefine tag using which we can inject inline or external javascript to the webpage.

- In a single webpage we can include any number of script tags.

e.g. Inline javascript:-

```
<script type="text/javascript">
    . . .
    . . .  // javascript code
</script>
```

External javascript:-

```
<script type="text/javascript" path="path of external js file"> </script>
```

```
<script type . . .
    alert("bye all");
</script>
```

16/05/19

* Data types:- In javascript it is mandatory & recommended to define the type of data being stored within a variable.
- java script supports following types of datatypes:-

(i) number (includes both decimals & integers)
(ii) string ⇒ String a = "abc"
(iii) boolean (takes values either True / False)
(iv) Object
(v) Function:
(vi) undefined:
(vii) null

Note:- In javascript its not needed to declare the variable along with its datatype, instead, we just need to specify its a variable using 'var' keyword.
(ii) Javascript holds the dynamicness feature through which, based on the value we assign to a variable, it assigns corresponding datatype to variable.

17/05/19 (18th class)

```
<script>
var StdName, SAge, SGender; var total=0, avg=0;
console.log("student name is" + StdName);
</script>
```

```
<script type="text/javascript">
var stdName, SAge, SGender;
    StdName = "Raj";
    SAge = 10;
    SGender = "Male";
    var m1 = 90;
    var m2 = 80;
    var m3 = 88;
    var m4 = 87;
    var m5 = 40;
    var total=0, avg=0;
```

```
total = M1 + M2 + M3 + M4 + M5;
avg = total/5;

Console.log("Student name is" + stdName);
            ("Student age is" + sAge);
            ("Student Gender is" + sgender);
            ("Student Total Marks is" + total);
            ("Student avg Marks is" + avg);
</script>
```

Q) Write a program to work with employee details like Employee Name, Gender, department & basic salary, calculate total salary of the employee where total salary = basic + Pf + hra and Pf is 14% of basic salary & hra is 25% of basic salary.

```
=> <script type="text/javascript">
     var EmpName, EmpGender, Empdepartment;
     EmpName = "Vishwa";
     EmpGender = 'M';
     Empdepartmant = "CSE";
     var basicSalary, PF, Hra;
     basicSalary = 10000;
     PF = 14% * basicSalary;   14/100 x basicSalary;
     Hra = 25% * basicSalary;  25/100 x basicSalary;
     var total = 0;
     total = basicSalary + PF + Hra;

Console.log("Employee Name is" + EmpName);
Console.log("Employee Gender is" + EmpGender);
Console.log("Employee department is" + Empdepartment);
Console.log("Employee basic salary is" + basicSalary);
Console.log("Employee total salary is" + total);
</script>
```

18/08/19 (13th class)

* "typeof()" Method:- A pre defined method been supported in javascript which takes a variable name as a parameter & the returns type of data it is holding.

Syntax:-

typeof (<variable name>);

e.g:

var a = "ten";
typeof(a) → returns string

* TypeCasting:- The process of converting data type of a variable to another data type is called Typecasting.
Following are the predefined methods been supported in javascript using which we could able to convert data type of one variable into another:-

(i) parseInt() (ii) parsefloat() (iii)* parseBoolean() etc.

* Dynamic TypeCasting:- javascript supports the concept of dynamic typecasting in which it automatically changes or converts datatype of one variable into another based on type of data it is holding.

e.g:-

var a = 90;
typeof(a) - returns number
....
a = "raj"; // javascript converts type of 'a' from number to string.
type of(a) - return string

<script>
// var a = 490;
console.log(type of(a)); // returns number
console.log(type of(alert); // returns function
console.log(type of(abc); // returns undefined
var z;
console.log(type of(z)); // returns undefined
</script>

18/03/19

Typecasting

```
<script>
var a = "30"
console.log(typeof(a));
a = parseInt(a);
console.log(typeof(a));
</script>
```

- Dynamic casting

```
<script>
var a = 50;
console.log(typeof(a));
a = "India";
console.log(typeof(a));
</script>
```

* Variable Hoisting:- In some languages like C or C++ it is mandatory to declare all the variables of the current application only at the starting of main method.
- The above statement is mandatory so that the memory allocation for the variables happens at the starting itself following with execution.
- In javascript developer has a flexibility to declare the variables anywhere within the application, javascript by default picks up all the variable declaration throughout the application, moves to the starting of the application. While moving the variable declarations it only moves the declaration but not the initialized values.
- The process of automatically picking up the variable declaration throughout the application, moving them to the start of the application is called variable hoisting.

e.g:-

```
<script>
var a = 30;
Console.log(a);- returns 30;
Console.log(b);= returns undefined instead of error
var b=90; //Variable declaration of b will be automatically moved to starting point of application
Console.log(b); - returns 90;
</script>
```

- Variable Hoisting

Ex:-

```
<script>
var a=80;
console.log(a);
var b=100;
console.log(b);
console.log("DONE");
</script>
```

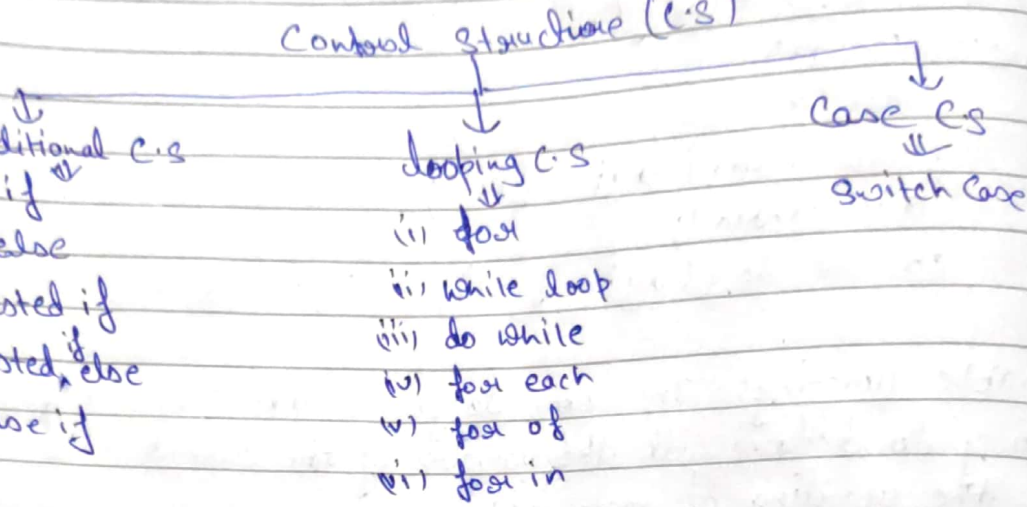
Javascript converts to this ⇒ Internally

```
var a,b;
a=80;
console.log(a);
console.log(b);
b=100;
console.log("DONE");
```

(Variable)

18/08/19

* Control Structures:- Following are different set of type of control structure been supported in javascript:-



• If Conditional Control Structure:-

Syntax:-

```
if (condition) {
...
... // set of instructions only gets executed if the provided condition is satisfied, else Even though the instructions are part of application, the execution of these instructions gets skipped off
}
```

example:-

```
var avg = 2.4;
if (avg > 40) {
....
....
Console.log("done");
}
```

```
    console.log("done");
}
```

19/09/19 (20th class)

- In debugging tool we use ~~console~~ sources it is used for step by step execution of program.
- It is also used for check whether the problem in executed the file.

nested:- using a element in the element.

```
if() {
    console.log(      );
    if( ) {
        console.log(      );
        if() {
            console.log(      );
        }
    } else if( ) {
            console.log(      );
    } else {
            console.log(      );
    }
} else {
        console.log(      );
}
```

Q) Write a program to work with employee details with following conditions to calculate tax value:

Gender = Female

Total Sal > 50000 - Tax % - 10

Total Sal > 30000 - Tax % - 5

Total Sal > 20,000 - Tax % 0

Gender - Male

Total Sal > 50000 > Tax - 15 %

Tal Sal > 30000 → Ta % 10

Total sal 20,000 → Tax % - 5.

⇒

```
if(Gender = Female)
    Total Salary.
var EmpName, empGender, EmpAge, EmpDepartment;
empName = "Vishwa";
empGender = 'M';
empAge = 10;
empDepartment = "CSE";
var basicSalary, pf, Hra;
basic Salary = 20000;
pf   = 14/100 x basic Salary;
Hra  = 25/100 x basicSalary;
```

```
    TotalSt basicSalary + pf + Hra ;
  if (enterGender == Male) {
    if (Total Salary > 50000) {
      var Tax ;
Total = tax + total; Tax = Total Salary * 15/100 ;
      console.log ("Total Salary is" + tax)
    } else if ( Total salary > 20000) {
      var tax ;
      Tax = Total Salary * 10/100 ;
    console.log (" Total Salary is " + tax);
  } else if ( Total Salary > 10000) {
      var tax ;
      Tax = Total Salary * 5/100 ;
    console.log (" Total Salary is " + Tax);
}}

if (enterGender == "Female") {
```

19/09/19

```
if (TotalSalary >= 50000){
    var tax;
      tax = TotalSalary * 10/100;
    console.log("Total Salary is" + tax);
} else if (TotalSalary > 30000){
    var tax;
      tax = TotalSalary * 5/100;
    console.log("Total Salary is" + tax);
} else if (TotalSalary > 10000){
    var tax
      tax = TotalSalary
    console.log("Total Salary" + totalSalary);
}}
```

20/09/19

(double equal to) (Equal value) (Evalue & Etype) (Triple equal to) (21st class)

* Difference between '==' and '===' operators:-

(Assignment operator)

- Both are comparison operators used to compare value of left side variable with right side variable, where the only difference is while comparing values '==' operator only ~~chan~~ checks for equality of values whereas '===' operator checks for value equality along with value type equality

Example:- (i) var a = 10; var b = 30;
a == b; → returns false

(ii) var a = 30; var b = 30;
a == b → returns true

(iii) var a = "10"; var b = 10;
a == b → returns true

(iv) var a = 10; var b = 30;
a === b; → returns false

(v) var a = 30; var b = 30;
a === b; → returns true

(vi) var a = "10"; var b = 10;
a === b; returns false

(Don't use infinite loop for statement)

18/ 2018/19

* Looping Control structure:- Following are set of looping control structure been supported in javascript using which we could able to execute the set of instructions more than once without rewriting the instructions.
(i) for loop (ii) while (iii) do while (iv) for in (v) for each (vi) for of

(i) for loop:-

```
for ( ___ ; ___ ; ___ ) {
      initialization   condition checking   Increment/(++)
                                            Decrement(--)
}
```

```
var counter;
for (counter = 1; counter <= 5; counter += 1) {
                                counter = counter + 1)
                                or
                                counter ++ )
    console.log("Good Morning");
    console.log("Bye");
}
```

first initialization
Second Condition
then Execution

Q) Write a program to display values from 1 to 100.

```
var i;
for (i=1; i<=100; i++); {
   console.log(i);
}
```

or

```
for ( var i=1; i<=100; i++) {
   console.log(i);
}
```

Q) write a program to display values from 99 to 48

```
for ( var i=99; i>=48, i--) {
   console.log(i);
}
```

20/09/19

Q) Write a program to display all the even number blw 1 to 100;

⇒ X
```
for (var i=1; i<=100; i++){
    if (i%2==0){
        console.log("the number is even");
    }
}
```
or
```
for (var i=1; i<=100; i++){
    if (i%2==0){
        console.log("i value is " +i); //odd
    }
}
```

Q) Write a program to find sum of all odd number blw 90 to 35;
```
var s=0;
for (var i=90; i>=35; i--){
    if (i%2!=0){
        s=s+i;
    }
}
console.log("s is " +s);
```

Q) Write a program to find the sum of all even number from 1000 to 99.

⇒
```
var sum=0;
for (var i=1000; i>=99; i--){
    if (i%2==0){
        sum=sum+i;
    }
}
console.log("sum is " +sum);
```

prompt()

20/08/19

2) Write a program to display table of 5 till 20.

```
var num=5;
for(i=1; i<=20; i++){
    var result = n*i;
}
console.log("num is " + num);
```

```
var n=5;
for(i=1; i<=20; i++){
    var result = n*i;
console.log(n + "*" + i + '=' + result);
}
```

3) Write a program to take a number to check whether it is prime number or not.

⇒

```
for(i=1; i<=20; i++){
    if
var a=10;
if(a%1 && a%10){
console.log("it is prime");
}
```

console.log(n + " * " + i +

24/09/19 } (22nd class)

8) Write a program to take a number to check whether it is prime number or not.

⇒ ~~for(i=1; i<=20; i++){~~ (i)
~~if~~
~~var a=10;~~
~~if(a%i or a(i0){~~
~~console.log(it is prime);~~
~~}~~

(ii)

var a=5, c=0;
for(i=1; i<=5; i++);
if(a%i==0)
{ c++;
if(c==2){
Prime number
}else
It is not

<script>
var a = prompt("Enter the number");
a = parseInt(a);
var c=0;
for(i=1; i<= a ; i++){
if(a%i==0)
c++;
}
if(c==2{
console.log("The number is prime");
}else{
console.log("the number is not prime");
}
</script>

④ for(var i=0; i<10; i++){ if(i==5){ continue;
console.log("Hello all"); }
console.log("bye all");
console.log("iteration is "+i);
}

(&& - AND operator
|| - OR operator)

24/09/19

(22nd class)

* 'break' and 'continue' statements:- 'break' is a predefine keyword been supported in javascript using which we could able to break the current iteration of any looping control statetructure irrelevant of whether the condition is still satisfied or not.

'continue' is a predefine keyword been supported in javascript, when the controller reaches to continue statement it automatically jumps back to the incrementation/decrementation step & continue the iteration.

Assignment

① Write the program to list out all the prime numbers blw 99 to 21.

② Write the program to find Out sum of all the prime number blw 1 to 200;

* While loop Control Structure:

Syntax:-

```
while(<condition>){
    ....
    ....
    // set of instructions gets repetively executed, until provided condition gets unsatisfied.
}
```

Ex:-

(i)
```
var i=1;
while(i<=10){
    console.log("Hello all");
    console.log("bye all");
    console.log("iteration no is "+i);
    i++;
}
```

(ii)
```
var n=243;
```

R = n%10 = 243/10 - 3
n = n/10 = 243/10 - 24
R = n/10 → 24/10 → 4
n = n/10 → 24/10 → 2

R = n%10 → 2%10 → 2
n = n/10 → 2/10 → 0

n = 10;
while(n*=2){ do{
}

}while(n*=2);

24/09/19

Ex:- <script>
var n = 243; var sum = 0;
While (n > 0) {
var rem = n % 10;
n = n/10; n = parseInt(n);
sum = sum + rem; // sum += rem;
}
console.log("sum is " + sum);
</script>

Q) Write a program to ~~find~~ read a number from the user & find a lucky number of it.

Q) Write a program to check whether given number is palindrome number or not. Ex: (121 - 121) (253 - 352) / (153 - 251)

Q) Write a program to check whether given number is armstrong or not. Ex: (153 = $1^3 + 5^3 + 3^3$).

Q) Write a program to find reverse of a given number.

Q) Write a program to display all the pallindrome number b/w 1000 to 99.

* do... While Looping Control Structure:-

Syntax:-
do {
...
... // set of instructions gets executed atleast once, irrelevant of whether given condition is satisfied or not.
} while(condition);

* Switch Case Control Structure :- 'switch' is a predefine control structure using which we could able to increase the

24/08/19

performance of a page by avoiding multiple conditions & providing all the possible cases.

Syntax:-

```
switch (variables) {
  case 'value1':
    - - -
    break;
  case 'value2':
    . . . .
    break;
- - -
- - -
default:
. . .
. . . // set of instructions gets executed if none
      of the cases match.
}
```

(23rd class)

26/08/19

* Reverse Number:-

```
<script>
    var rv = 0;
    var n = 572;
    var rem;
while (n > 0) {
        rem = n % 10;
        n = n / 10;
        n = parseInt(n);
        rv = (rv * 10) + rem;
    }
    Console.log("Reverse of a number is " + n);
</script>
```

(n = c);

- Switch Case:-

```
switch (UserSelection) {
    case 'add':
    result = fval + sval;
    console.log("the sum is " + result);
    break;
```

25/09/19

```
case 'mul':
    result = fval * sval;
    console.log("The result is " + result);
    break;
case 'sub':
    result = fval - sval;
    console.log("The result is " + result);
    break;
case 'div':
    result = fval / sval;
    console.log("The result is " + result);
    break;
default:
    console.log("Wrong Selection");
}
```

* Data Structures:-

- Array:- A predefined data structure been supported in javascript using which we could able to hold multiple values under a single variable name.

(i) In any language arrays are defined & identified using a square braces ("[ ]").-

(ii) In javascript arrays are heterogenious datastructure.

(iii) An array in javascript is capable of holding any type of data in it/within it.

(iv) By default all the values inside an array occupies continous memory allocation.

(v) By default every value within an array will be assigned with a default index values which start with 0(zero).

(vi) In order to access or update values inside an array we make use the combination of arrayname & index value.

(vii) Following are the two ways we can create static and dynamic arrays in javascript:-

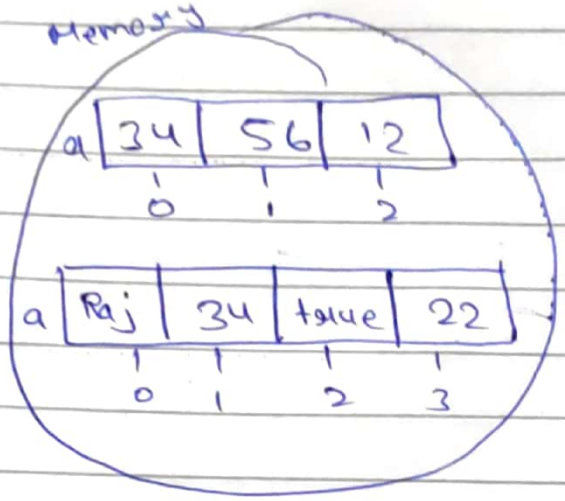
(i) var a = new array(); // dynamic

(ii) var a = []; // static declaration (Recommended)

25/08/19

e.g:-

```
var a = [34, 56, 12]; // Array with declaration
                      and initialized values

var a = ['Raj', 34, true, 22]; // Declaring and
initializing hetrogenious array
```



* Following are predefined methods can applied on Arrays:-

(i) .push() :- Used to insert a single value in an array, from right direction

(ii) .pop() :- Used to delete a single value from array, from left direction

(iii) .shift() :- To delete a single value, from right direction

(iv) .unshift() :- To insert a single value from left direction

(v) .splice(startindex, totalnumberof values to be deleted, optional values to be inserted) :- A single method used to insert or delete values from any position of an array.

(24th class)

26/09/19 Ex:

```
var marks = [99,90,33,66,25,23];
console.log(marks.length);
var total = 0
for(var i=0; i<marks.length; i++){
        total = total + marks[i];
}
    var avg = total + marks.length;
console.log(avg);
console.log(total);
```

26/03/19

Q) Write a program to define an array with static value and find the sum of all values in it.

```
var a = [20,40,30,50,60];
var total = 0;
for (var i=0; i<a.length; i++) {
    total = total + a[i];
}
Console.log("Total Marks is " + total);
```

Q) Write a program to find the sum of all even number within an array.

```
var a = [10, 11, 13, 16, 18]; var sum=0;
for(i=0; i<a.length; i++) {
    if(a[i] % 2 == 0){
        sum = sum+i;
    }
Console.log(sum);
```

Q) Write a program to display the values inside an array to reverse.

```
var a = [245, 346, 545, 231];
for(i = a.length-1; i>=0; i--) {
    console.log(i);
}
```

Q) Write a program to find list of prime number within an array.

```
var a=[10, 11, 12, 13, 15];
```

26/08/19

1) Write a program to find largest & smallest number within an array.

2) Write a program to find second largest number within an array

3) Write a program to sort an array (ascending order).

4) Write a program to display all the palindrome numbers within the array.

5) Write a program to list out all the armstrong numbers within an array

6) Write a program to find lucky number for every value within an array.

→ Smallest Number:-

```
var a = [99,33,66,55,23,44,55];
var sn = a[0];
for(var i=1; i<a.length; i++){
    if(sn > n[i]){
        sn = n[i];
    }
}
console.log("smallest is " + sn);
```

- Demonstrating array predifined;

```
Var a = [45,22,11,52,42];
    console.log(a);
console.log(a.length);
a.push(99);
console.log(a);
a.unshift(101);
console.log(a);
```

26/09/19

```
a.pop();
console.log(a);
a.shift();
console.log(a);
a.splice(2,0,33,44,55,66);
console.log(a);
a.splice(a.length-1,0,11);
console.log(a);
a.splice(0,0,11);
console.log(a);
a.splice(a.length-1,1);
console.log(a);
```

* Javascript Functions:- A group/set of relevant instructions been binded with in '{' and '}', assigned a userdefined name following with keyword 'function' is called a javascript function.

Syntax:-

```
function <userdefined-function-name>(<optional params>){
    ....
    .... // set of instructions
}
```

(i) Every function in javascript will be defined using keyword 'function'.
(ii) function names are all usedefined.
(iii) Once a function been defined can invoked/called for any number of times in an application.
(iv) A function been called within the same definition of function is called, recursive function.
(v) Using functions we could achieve reusability of code, and follow modularity.
(vi) In a single application we can define any number of functions.

26/09/19

(vii) Data been declared in a function can't be accessable outside of function, it becomes private data of current function.
(viii) Data been declared outside of any module or function is called Global or public data.

Following are the different ways we can define functions in javascript :-

(i) function name() {
.....
..... // set of instructions.
}

(ii) var name = function() {
.....
..... // set of instructions
}

28/06/19 (25th class)

Local / Private Data:- Data been declared

- Scope of Variables:- The scope of variables specifies the accessability level of variables. It could be local or global scope
- Local / Private Data:- Data been declared within any module gets local scope, which cann't be aceessable outside of module.
- Golobal / Public Data:- Data been declared outside of module gets public / global scope, which can be accessable anywhere through out the application.
- Following are the different ways, indirectly local variable of a function / module can be accessable outside of it:-
(i) function with parameters (ii) function with returning value
(iii) function with both parameters & returning values.

Q. Write a program to work with Employee details using function.

```
<Script>
    function studentDetais() {
        var, sName, sAge, sDesignation, sGender;
        sName = prompt("Enter Studeat Name");
        sAge = prompt("Enter Age)
        sAge = parseInt(sAge);
```

28/09/19

```
sDesignation = ("Enter Designation");
sGender = ("Enter Gender");
}

function basicSalary(){
    var basicSalary, pf, hra;
basic salary = parseInt(basic Salary)
    basic Salary = prompt("Enter basic Salary");
    pf = 14/100 * basic salary;
    hra = 25/100 * basic Salary;
}

function TotalSalary(){
    var TotalSalary = 0;
console.log("TotalSalary")
    TotalSalary = basicSalary + pf + hra;
if (sGender == "M"){
    if (Total Salary > 5000){
        var tax;
        tax = 15/100 * Total Salary + TotalSalary;
    Console.log("The total Salary is " + TotalSalary);
} else if (Total Salary > 30000){
    var tax;
    tax = 10/100 * Total Salary + Total Salary;
Console.log("The total Salary is " + tax);
} else if (Total Salary > 10000){
    var tax;
    tax = 5/100 * TotalSalary + TotalSalary;
Console.log("The total Salary is " + tax);
}
if (sGender == "F"){
    if (TotalSalary > 50000){
        var tax;
        var tax = 5/100 * TotalSalary + TotalSalary;
    Console.log("The total Salary is " + tax);
} else if (TotalSalary > 30000){
    var tax;
    tax = 10/100 * TotalSalary + Total Salary;
Console.log("The total salary is " + tax);
}
```

2/8/2019

```
else if (totalSalary > 10000) {
    var tax;
    tax = TotalSal
    Console.log("the salary is" + totalSalary);{
    }
}

    studentDetails();
    basicSalary();
    totalSalary();
</script>
```

30/08/19

(26th class)

• Anything b/w these (-,-) is called parameter. It is used for security reason. function readValues() {

```
    var a, b;
var count = 90;
a = prompt("Enter first value");
a = parseInt(a);
b = prompt("Enter second value");
b = parseInt(b);
  addValues(a, b);
}
function addValues(P1, P2){
    var c = 100;
    var result;
    result = P1 + P2;
     a = 900;
Console.log("The sum is " + result);
}
 readValues();
```

* Function with return type / return values:-

```
function readValues(){
    var a, b;
a = prompt("enter first values");
a = parseInt(a);
```

(formal parameter → Arguments - Defining a function
Actual parameter → parameter - calling a function)

30/08/19

```
b = prompt("enter second value");
b = parseInt(b);
var res = addValues(a,b);
console.log("the result is " + res);
}
function addValues(p1, p2){
    var result;
    result = p1 + p2;
    return result;
}
readValues();
```

Actual parameter

formal parameter

goto

② Write a program to work with the student details using function with no global data.

student

```
function readValues(){
    var sname = prompt("enter name");
    var age = prompt("enter age");
        age = parseInt(age);
    var gender = prompt
    var marks = [];
    for(var i=0; i<5; i++){
        marks[i] = prompt("enter subject " +(i+1)+ "
        'marks');
    var total = 0;
    var avg;
}
}
```

② Solve a program as Employee details.

01/10/19

(21th class)

* Date object :- Date is a predefined class in javascript using which we could able to work on current system date or user defined custom date.

Creating Date object:-

```
var date = new Date(); //returns current system date
```

or

```
var date = new Date(<custom date>);
```

Following are the predefined methods can be applied on date object.

- getMonth() - returns value between 0-11.
- getDate() - returns value between 1-31.
- getDay() - returns value between 0-6.
- getMinutes() - , • getFullYear() -
- getHours() -
- getMilliSeconds() -

etc.

* JavaScript String Method:- Following are the predefined methods been supported in javascript which can be applied on Strings.

- length - Is a property to find the number of characters in a string, including spaces.
- subString (startindex, length of cars); ← total number of value you want
- slice (startindex, endindex);
- replace("what", "with"); => (i = ignore case, g = global) replace('1', '**'); replace(/1/g, '**');
- concat (string1, string2, string3, ...);
- charAt(index);
- charCodeAt(index); => ASCII Code (American Standard Code)
- split('seperator') - splits the string with provided seperator and returns an array.

etc.

Q) Write a program to show date format as per the user requirement. Following are format user can offer:

(i) dd/mm/yy (ii) dd/mm/yyyy

=>

```
<Script>
var mydate = new Date();
```

(iii) DayName - MonthName - Fullyear

(iv) dd - FullmonthName - fullyear (v) dayname, monthname Fullyear

(vi) Fullyear - month - dd

2 01/10/19

Q) Write a program to read a string from user & return total no. of vowels been present in it.

Q) Write a program to find out total number of words been present within a string.

Q) Write a program to find out total number of special symbol been present within a string.

Q) Write a program to find total number of characters inside a string. without using predefined length property.

* 'setTimeout' and 'setInterval' Methods:- The two predefined methods been supported in javascript using which we could able to execute the of instructions not just when the controller reaches to it but to exe set of instructions only after the provided delay time.

Syntax:-

```
setTimeout(<callbackfunction>, <delaytime>);
setInterval(<callbackfunction>, <delaytime>);
```

setTimeout & setInterval are almost like the same use to execute pro callback function after an interval of time where the only difference is setTimeout will invoked the callback method only once after the provided delay time whereas setInterval method will keep repeating executing provided callback method for ~~every~~ with a gap provided delay time.

Note:- 'clearInterval()':- It is a predefined method using whi we could stopped a setInterval method execution.

(28th class)

02/10/19

Ex:- 1st method

```
<script
  function showWelcomeMsgs(){
  console.log("welcom to our page");
  console.log("visit again");
}                    contact
function showThankMsgs(){
console.log("thq for visiting");
console.log("come gain");
}
```

(A function without any name is called anonymous function)

Ex:-
```
setTimeout(function(){
  console.log("thank you");
  console.log("visit again");
},5000);
```

02/01/19

```
showWelcomeMsgs();   console.log("about to execute");
setTimeout(showthnxMsgs, 5000);
console.log("Done");
```

(ii) Method Ex:-

```
ShowWelcomeMsgs();
console.log("about to execute");
setTimeout(function(){
  console.log("Thq for visiting our page");
  console.log("Visit Again");
},5000);
console.log("Done");
```

Example for setInterval Method:-

```
var count=0;
var job1 = setInterval(function(){
   count++;
  console.log("JOB1 " + count);
  if(count==5){
     clearInterval(job1);
  }
},3000);
var job2 = setInterval(function(){
    console.log("JOB2");
},3500);
```

*

JSON (JavaScript Object Notation)

- JSON - (i) A data structure been supported in javascript using which you could able to group and store heterogeneous data in it.
- (ii) Under JSON object data will be stored in the form of ~~key~~ key & value pairs.
- (iii) The data type of a key should always be a string, corresponding value could be of any javascript supported data type.
- (iv) In order to access data from the JSON we make use the combination of object name with corresponding key name.

02/01/19

(v) While allocating memory for a JSON object it allocates the memory in the form of an array.

Following are the two ways we can create objects in javascript
(i) var obj = {}; // static (ii) var obj = new object(); // Dynamic (recommended)

Ex:

```
var sDetails = {
    name = "Raj",
    age = 10,
    Gender = "Male",
    location = "Hyderabad",
    marks = [50, 40, 60, 80];
    total = 0,
    avg = 0,
};
for(var i=0; i<sDetails.marks.length; i++){
    sDetails.total = sDetails[i];
}

sDetails.avg = sDetails.marks / sDetails.marks.length;
Console.log("Student details");
console.log("Student name : " + sDetails.name);
Console.log("  age : " + sDetails.age);
Console.log("  gender: " + sDetails.gender);
Console.log("  location: " + sDetails.location);
Console.log("  Total Marks : " + sDetails.total);
Console.log("  avg : " + sDetails.avg);
```

03/10/19 (29th class)

* for in looping control structure :- A looping control structure been supported in javascript using which we could able to iterate over a json object.

(ii) for in looping control structure takes a temporary variable, used to hold the corresponding key while iterating over a json object, the temp variable holds one key at a time, replaces with another key in next iteration.

Syntax :-

```
for (temp in object) {
    .... // set of instruction gets executed until it iterates
}       all key's in json object
```

for (temp in ____){
↑ object name

03/10/19

Ex:- var data = { name: 'Raj', age: 30, location: hyd };
for (var temp in data){
console.log("key is " + temp);
console.log("value is " + data[temp]);
}

Ex: var bookData = { bName: 'Science', bPrice: 350, bpublish: ['Asp', 'web', 'Indian'], bAuthor: 'Rama', Rating: 5.0 };

Contact structure

- Fname :
- lastname :
- Contact Name — official, personal
- Maild Id — personal, office
- Pic :
- dob :
- Gender :
- Address — office — country, state; personal — SNo, location, state, pin
- Organisation
- website
- Social Id — fb, Insta, Linkedin, Twitter
- Relationship

var ContactDetails = {
fname: 'Raj',
lname: 'kumar',
ContactName: { personal: [21456, 22456, 3456]
office: [21564, 34567] },
MaildId: { personal: 'abc@gmail.com',
official: 'abc@yahoo.com' },
pic: 'pic.jpg',
dob: dd/mm/YY,
Gender: 'Male',
Address: { office: { countryName: 'India',
State: 'Telangana',
personal: { SNo - '4', door: 14,
location: 'hyd' State: 'Telangana'
pin: 111006.},
Organisation: 'abc',
website: 'ab.com',
Social Id: { fb: 'abc',
Insta: 'cad',
Twitter: 'Ing',
},
Relationship: 'single'
}

03/10/19

```
console.log(contactDetails.Name);
console.log(contactDetails.Address.office);
}
```

(30th class)

04/10/19

DHTML (Dynamic Hypertext Markup Language)

DHTML :-

* Javascript is a powerful language come with predefine object & method using which we could able to perform any 'CURD' operations on any DOM element or its corresponding 'css' properties dynamically.
→ Using javascript we could able to generate the complete HTML page with dynamic data as and when required.
→ It provides objects & methods using which we can handle events been generated on the page.
→ It provides predefined methods & objects using which we could able to 'Read' or 'Write' from or on the HTML 'input elements.

Note:- 'document' and 'window' are the two predefined objects by default available ~~using~~ which actually holds the complete DOM structure, we can perform any operation on any DOM element using these objects.

* Event Handling :-

- Event - Any action being performed on a page is called an event. Following are types of events might get generated on a page.
→ click, doubleclick, keypress, keyup, mousedown, mouseup, focus, blur, mouseover, mouse out.

- event handling :- The process of performing a job when a particular type of event happen on a page is called event handling.
→ javascript provides two ways of implementing event handling on a page.
(i) Static event handling (ii) Dynamic event handling

- Static event handling :- javascript provides set of predefined event attributes using which we can statically specify to the DOM element itself to invoked what method for what type of event. Following are set of predefine HTML attributes through which we can implement static event handling on HTML element.

(DHTML - HTML + CSS + JS)
↳ 'Group of concept been combined'
'I want to show dynamic data in web page'

outline

e.g:- <div onclick or <event> = "<method need to be invoked>"
...

e.g:-
function displayData() {
...
}

....

- Window and document object:- The two predefined object been supported in javascript using (through which we could perform any type of operations on any HTML element or its corresponding CSS properties. Following are the different ways we could refer a DOM element on page:-

(i) document.getElementById() - Using which we could refer any DOM element on page using its unique id.

(ii) document.getElementByClassName() - Using which we could refer any DOM element using the class name

(iii) document.getElementByTagName() - Using which we could refer any ~~DOM~~ element by its tag name.

(iv) document.querySelector() - A method been supported from HTML 5 using which we can refer any DOM element by id/ classname/ tag name.

- Creating a DOM element through javascript:-

document.createElement("<tagname>") - Method to create any DOM element dynamically using javascript.

Following are the methods can be applied on element reference:-

(i) element.setAttribute("attribute name", 'value') - Using which we could able to set any attribute dynamically to a DOM element.

04/10/19

(ii) element.getAttribute("Attribut Name") - Using which we can get corresponding attribute value of any DOM element.

(iii) element.style :- Using which dynamically any css property of any DOM element can be retrived or updated.

(iv) element.appendChild(<element>) :- Using which we could able to append an element to an existing DOM element on page

Implementing Dynamic event Handling :-

element.addEventListener("event name", <callback Method>) :- Using which we could add a dynamic event handling to existing DOM element.

10/10/19 (31st class)

```
<div class = "container">
  <ul>
    <li>
      Enter first value <input type="text" id="fval">
    </li>
    <li>
      Enter Secand value <input type="text" id="sval">
    </li>
    <li>
      <input type="button" value="ADD" onclick="addvalues()">
    </li>
    <li>
      The Sum is : <span class="block"></span>
    </li>
  </div>
  <script>
    function addvalues() {
      Cowsole.log(document);
    var fvalue = document.querySelector("#fval").value;
      fvalue = parseInt(fvalue);
    var svalue = document.querySelector("#sval").value;
      svalue = parseInt(svalue);
```

10/10/19

'innerHTML / 'innerText

```
var result = fvalue + Svalue;
Console.log(result);
document.getquerySelector('.rblock').'innerHTML= result;
<script> document.querySelector(".rblock").style.colour
= "red";
</script>
```

</script>

12/10/19 [32nd class]

* "event" object :- Every defined object gets generated anytime an event happens on a page, this object holds extra information of the current event being generated.

(ii) 'N' number of event objects gets created when 'n' number of events happens on the page.

(iii) It is not mandatory to catch an event object when it gets generated, we can catch when it is/there is a need.

(iv) Following are the extra information available under event object:-

(i) typeofevent - specifies the type of event got generated.

(ii) target - Reference of the dom element on which event got generated.

(iii) clientX, clientY - Specifies the x & y points at which event got generated.

(iv) ctrlkey, altkey - specifies whether alt or ctrl key been pressed or not.

(v) path - specifies the complete upstream path till html tag.

(vi) timestamp - specifies the time at which event got generated.

(vii) charCode - specifies ASCII value of character been pressed only in case of key press event.

etc.

etc.

(3rd class)

19/10/19

* Creating DOM element using javascript:-

* Adding event listners to DOM element dynamically through javascript:-

- Event Bubbling and Event Capturing:- Whenever a parent element & its child element being added with same type of event handling, the order of events invoking will be decided using event Bubbling or event Capturing.

Callback Method automatically invokes
We create Method manually invokes

15/10/19

- Event Bubbling:- In this type the child event invokes first following with parents.
- Event Capturing:- In this type the parents events gets fired first following with childrens.

Note:- Event Bubbling is the default type whereas to follow the Event Capturing we specify a boolean value which confirms Event Bubbling or Capturing.

Syntax:= element.addEventListener("type of event", <Callback method>, true/false);

//true indicates capturing, false indicates Bubbling.

(35th class)

16/10/19

* Advance Javascript:-

- JSON object with properties & behaviour:-

16/10/19
17/10/19 (35th class)
(36th class)

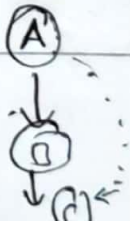
Advance Javascript:-

- JSON object with properties & behaviours:-

- "this" operator:- Every define keyword been supported in javascript which refers to the current object reference.
In order to access the properties or behaviours of an object within the object itself, we can make use of the object name or this operator.

* Inheritance:- The concept of inheriting the data from one object to another is called inheritance,
- In case of inheritance the object from which data is getting inherited is called parent object, ~~the ob~~ or base object, the object which is inheriting the data from parent object is called child object or derived object.
- the main intention of implementing inheritance is avoid code redundancy or duplication, achieve reusability.
- Javascript supports following two types of inheritance:-

(i) Single level inheritance (ii) Multi level inheritance



17/10/19

- Following are the different ways we can implement inheritance in javascript:-

(i) Inheritance between static object.
(ii) Inheritance between a static object and a dynamic object
(iii) Inheritance between dynamic and dynamic object

- Inheritance between static object using "__proto__":-

⇒ Every static object being created within javascript holds the default property "__proto__" using which we could able to extend the capabilities of an object to even point out the properties and behaviour of another existing JSON object.

e.g:-

```
var objA = {
    .....
};
var objB = {
    .....
};

objB.__proto__ = objA; // Now objB gets the capability of pointing its own properties & even to point out / refer object properties & behaviour.
```

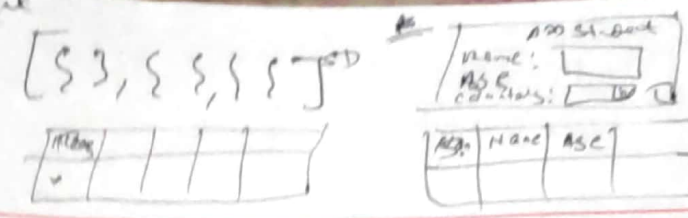
- Inheritance between a static object and a dynamic object:-

⇒ "object.create()" is a predefine method using which we can create a dynamic object which gets accessability of an existing static object.

e.g:-

```
var objA = { // static object
    .....
};

var objB = object.create(objA); // objB is a dynamic object which gets capability of even pointing (referring Static object properties and behaviour of objA.
```

Assignment



18/10/19
19/10/19

(37th class)
(38th class)

* Inheritance between two dynamic object:-

• **Class:-** Classes are templates or blueprint of an object which holds set of properties (values, without any values) & set of behaviours.
Once we create a class, we can create any number of objects or instances for a particular class.
The properties or behaviours within a class cannot be exacted through class name, for sure an object or an instance need to be created for it.
Until ECMAS javascript was not directly supporting javascript classes indirectly we can create a class using javascript function itself.
All the declarations of the class property or behaviours should be done using "this" operator.

Syntax:-

```
function <classname>(<optional args>){
    this.propertyname = 'value';
    this.propertyname2 = 'value2';
    ....
    this.behaviour1 = function(){
    ....
    }
    this.behaviour2 = function(){
    ....
    }
}
```

Creating Instance/object to a class:-

```
var obj = new <classname>(<optionalparamas>);
```

Examples:-

```
function stdDetails(data){
    this.name = data.name;
    this.age = data.age;
    ....
    this.showDetails = function(){
    ....
    }
    this.dosomething = function(){
    ....
}
```

21/10/19 (39th class)

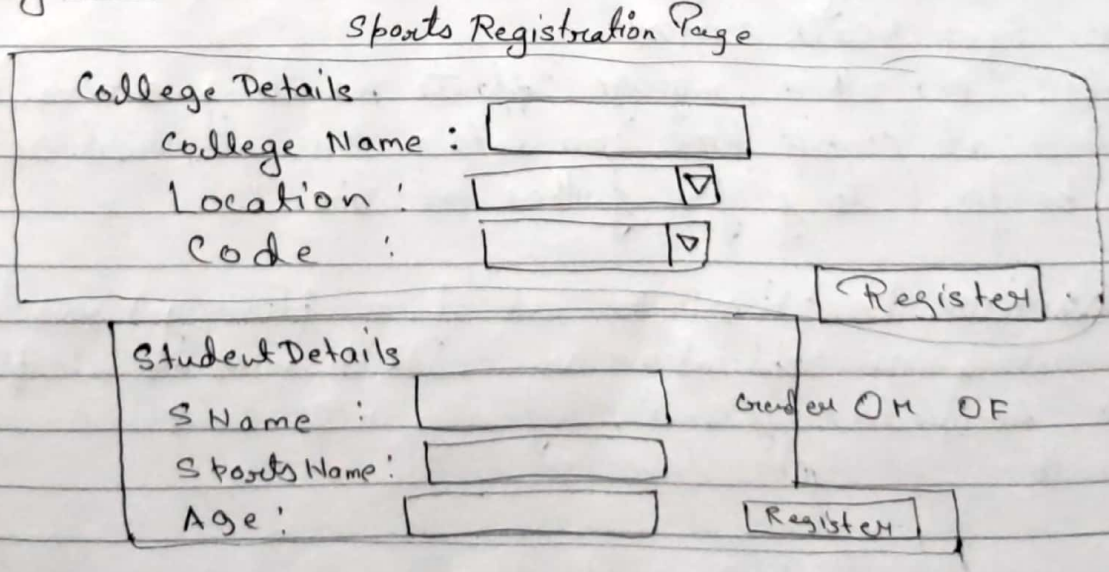
* Inheritance beetween two dynamic object or classes:-

"prototype":- A predefine property by default available under every class in javascript using which we can extend the behaviour of a class to point or refer an existing object properties or behaviour. (indirectly achieving inheritance between classes).

Eg:-
```
function parentclass(){
    this.property1 = value;
    :::
}
function childclass(){
    this.property2 = value;
    :::
}
childclass.prototype = new parentclass(<optional params>);
// Child class object will get capability of pointing its own properties, behaviours along with pointing parentclass properties & behaviour.
```

22/10/19 (40th class)

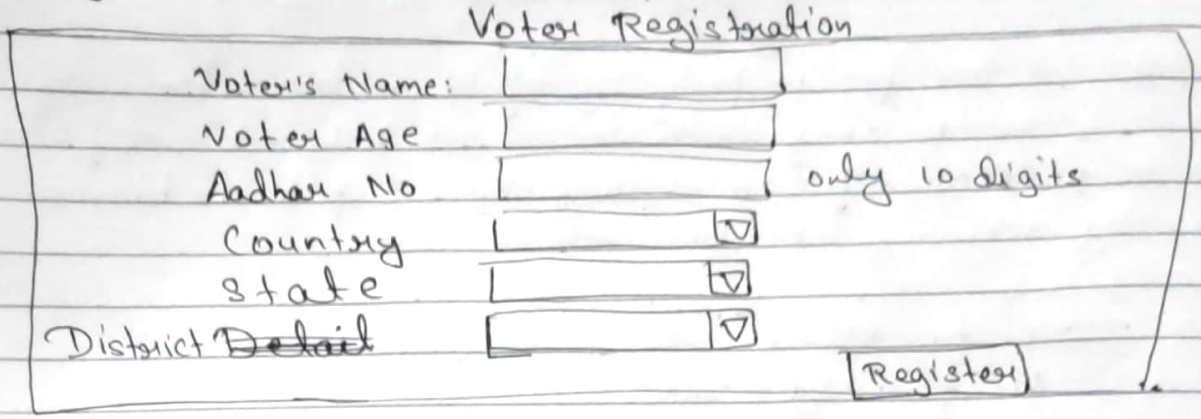
Assignment



22/10/19

Assignment

Voter Registration



* Exception Handling:-

- Exception or Error:- The set of statement or a single line of a statement making the flow of application execution process to stop abortly these called an exception or an error.

Following are the two types of exceptions might get raised in an application :-

(i) Compile time Exception (ii) Runtime Exception
(iii) User define Exception

- Compile time Exception:- The exceptions or errors gets raised at a time of compilation is called compile time errors.

Ex:- Syntaticall errors.

Note:- No other language supports handling errors which gets raised at compile time, mandatory the syntactical error need to be fixed to go for further execution.

- Runtime Exception:- The set of exceptions get raised while executing an application are called runtime exception.

Ex:- Array index out of Bound, Referring an element not existing within the DOM structure.

- Exception Handling:- Javascript supports the concept of handling runtime exceptions so that the application flow of execution still continues in an thow there is an exception been raised.

22/10/19

- Following are the keywords been supported in javascript through which we can handle runtime exceptions
(i) try (ii) Catch (iii) finally (iv) throws

- try block:- The set of instructions in which there is a chance of getting a runtime exception has to be placed within the try block.

- Catch block:- It should be the very immediate & mandatory block need to be placed after the try block.
(i) When an exception gets raised within the try block, the controller automatically jumps to corresponding catch block to handle exception.
(ii) The set of instructions through which we handle the exception has to be placed only within the catch block.
(iv) In javascript we cann't override catch methods.
(v) Exception is a predefine object gets created & will be thrown to catch method. The exception object holds extra information of the current exception been raised.

- finally block:- the set of instructions which need to be for sure executed irrelevant of whether there is an exception been raised within the try block or not ~~is called~~ has to be placed within the finally block.
(ii) Adding finally block to a ~~catch~~ try block is optional.

syntax:-

```
try {
    ..... // set of instructions in which there is a chance
    ..... of getting an exception raised.
} catch(exception) { // Block only gets executed if there
    ..... is an exception been raised in try
    ..... block
    ..... // set of instructions to handle exception
}
finally {
    ..... // set of instructions which gets executed for
    ..... sure irrelevant of whether there is an
    ..... exception been raised or not.
}
```

(It is mandatory to use the 'throw' in try block)

22/10/19
23/10/19

(41th class)

- Handling User defined Exception:- Javascript provides a feature of ~~handle~~ throwing & handling user defined exceptions using "throw" keyword.

Syntax:-

```
try {
    ....
    throw '<msg>'; // From here controller will auto-
    -matically jumps to catch block to handle user defined
    exception.
    ....
} catch(e) {
    .... // set of instruction to handle user defined exception
}
```

* closures:- A feature been supported in javascript using which we could able to add security to the javascript code or data.
i) closure is a self invoked function gets invoked automatically when the controller reaches to it.
ii) All the data been defined within a closure becomes private data to outside closure.

- Steps to be followed to access closure data out of closure:-
(i) Create an assign a variable to closure.
(ii) All the closure data which need to be accessible outside of the closure has to be returned as a JSON object from the closure.
(iii) We can make use of returned object data from the closure, from outside of the closure using closure name.

Syntax:-

```
// closure definition
(function () {
    var a=10; // cannot be accessable outside of
    ...          closure.
    function sample() { // cannot be accessable
        ....
    }
    ....
    var data = { // cannot be
    ....
})();
```

23/10/19

// None of the data been defined in the above closure cannot be accessable outside of closure.

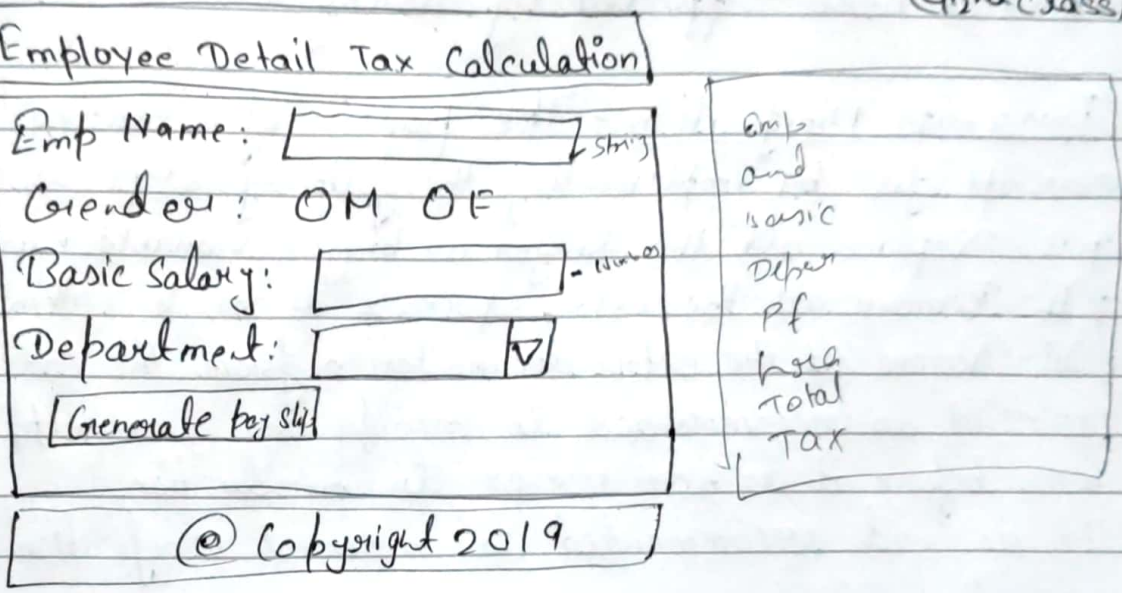
- Example to create a closure whose data can be accessable outside of closure :-

```
var userUtils = (function(){
    var a=10; // cannot be accessable outside of closure
    ...
    function sample(){ // cannot be accessable
        ...
    }
    ...
    var data = { // cannot be
        ...
    }
    return { // all the data under this object can be accessable outside of closure using closure name.
        count: 20,
        displayData: function(){
            ...
        },
        ...
    }
})(<optional params>);
```

24/10/19

Assignment

(42nd class)



25/10/19

JQuery

(43rd class)

* jquery:- A predefine javascript library been developed by Mr. John Resig in the year 2006 and the latest version of it is v3.4.1.

(ii) It comes with set of predefine methods using which any type of DOM operations can be simplified with a lesser code.

(iii) It supports set of predefine methods using which any type of CURD operation can be performed on both DOM elements & corresponding CSS properties.

(iv) Using JQuery as a UI developer we get benefited by not to sped much time on basic DOM operations instead we concentrate on actual business logic of the application.

(v) It comes with set of predefine methods using which event handling on the DOM elements can be more simplified.

(vi) Other than supporting DOM operations methods JQuery also comes with set of predefine methods using which AJAX communication can be simplified.

(vii) It provides set of predefine methods using which DOM traversing can be simplified.

(viii) It provides set of predefine methods using which we could able to animate DOM element within the page.

http://api.jquery.com is the dedicated page to find the documentation of JQuery API.

"$" and "JQuery" are the predefine object or closure names being provided by jquery through which we could able to access methods or properties been supported by JQuery.

* Javascript Minification:- The process of reducing size of a javascript file by implementing the following steps is called minification.

Step I - Replace all the longer or bigger variable names with the small.

Step II - Remove all the extra spaces & linebreak within the code.

Step III - Remove all the extra documentation from the code.

Note:- (i) It is recommended to minify any javascript code being return, before it is been moved to production.

(ii) It is not recommended to implement minification while in development process.

25/10/19

(iii) Minification process helps in reducing the file size which in term help in increasing performance of a page.
(iv) There are the some external tool or websites available which help in minify javascript code at free of cost.

(44th class)

26/10/19

* JQuery Load and Ready Methods:- The two predefined methods supported in jquery, which can be applied on document object, using which set of instructions can be delayed to execute only after the complete DOM structure gets loaded.

'ready' method - Takes a callback function as an input and invokes the function only after the complete DOM structure gets loaded.

```
$(document).ready(function() {
    ....
    // set of instructions which gets executed only after the complete DOM structure gets loaded.
});
```

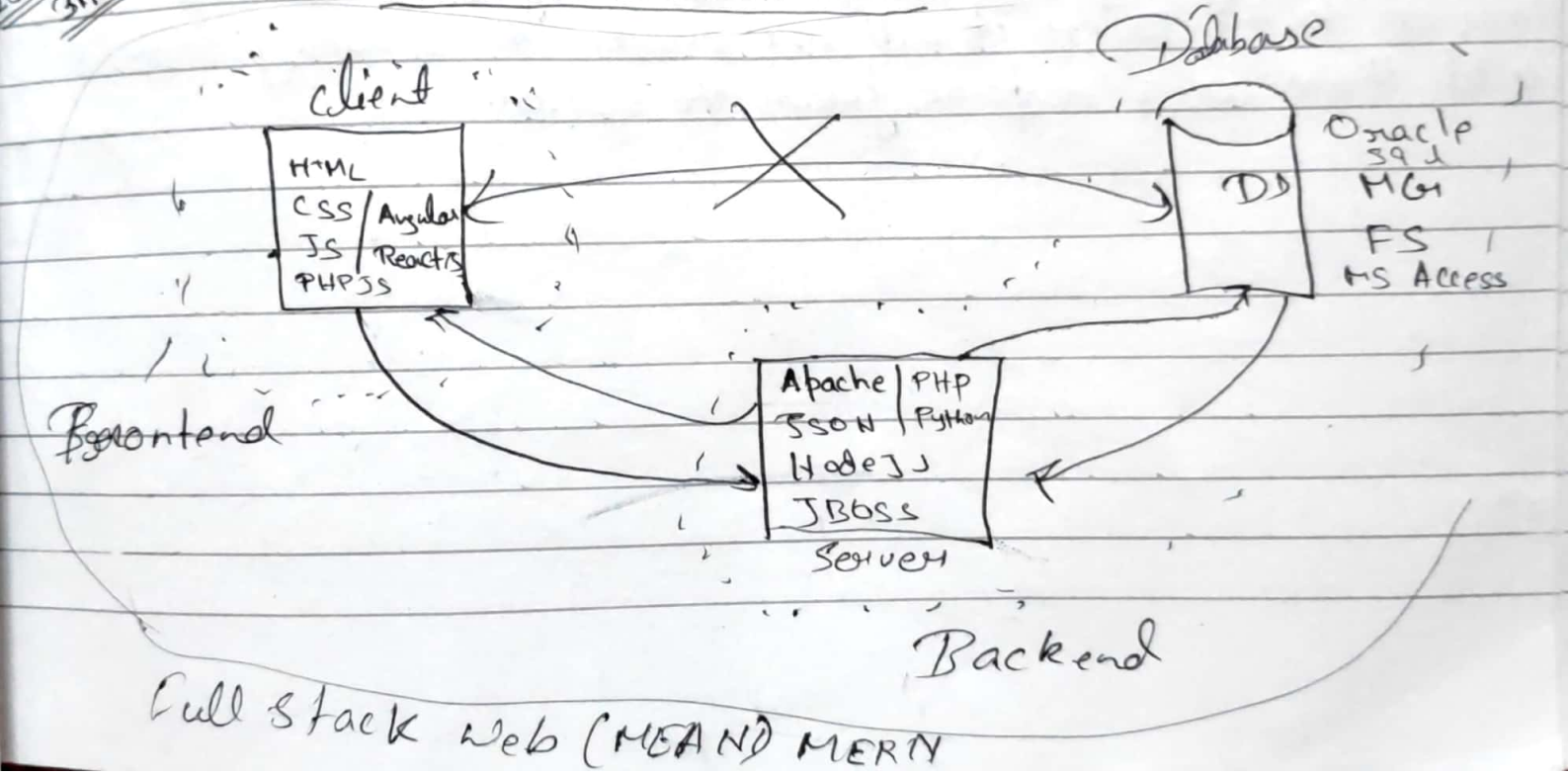
'load' method - It is almost like and ready method, where the only difference is, load method not just waits for the DOM structure to get loaded, but it even waits for complete resources on the page to gets loaded (images with huge size, heavy size js or css external files etc.).

external

31/10/19

(45th class)

AJAX COMMUNICATION



Full stack Web (MEAN) MERN

21/10/19

* Web Service :- Set of independent instructions gets executed at the server side capable of taking input request from the client side, process it, if needed interacts with database and respond back the data to the client is called a webservices.

(i) In a single server it can hold any number of webservices.
(ii) Every ~~indi~~ webservice within a server runs independent to each other.
(iii) Every individual webservice gets indetify with a unique url.
(iv) It is only the webservice through which client can send or receive data from server.

* AJAX (

- The process of creating a request from the client side to a server using a webservices url either to post the data or to GET the data is called AJAX Communication.
- XHL, RSS, String, ISON etc are the popular data formats can be used ~~for~~ while implementing AJAX Communication.

* Synchronous & Asynchronous Communication :- While Communicating with a server through a webservice url, the communication can be either synchronous or Asynchronous.

- In case of asynchronous communication client never waits for a responds from the server after it puts a requests, that call back method will take care when there is a respond from th server.
- In case of synchronous communication after a client puts a request to the server it will not execute the remaining statement until there is a response from the server.

01/11/19 (46th class)

* Creating an AJAX call using jQuery's "$.ajax()" method.

- Other than providing set of predefine methods to interact with the DOM structure, jquery even provides set of predefine methods through which we can implement AJAX communication in a simpler way.
- "$.ajax()" method takes an object as an input with following properties:
- url - url of a webservice
- method - specifies the what type of communication (GET/POST).

- Synchronous Communication for login
- Asynchronous Communication for user profile

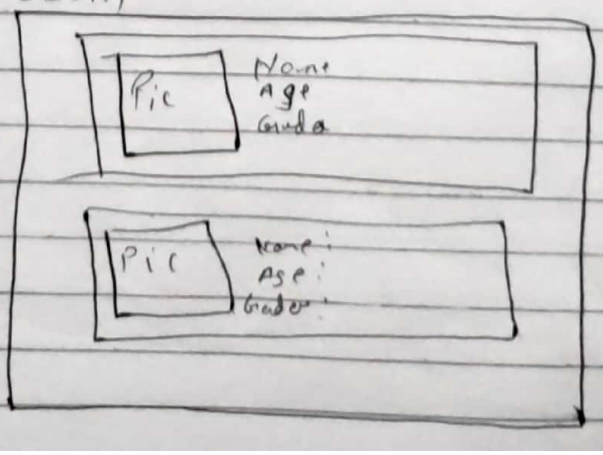
01/11/19 SPA (Single Page Application) Template (only for front)

- data - An optional value with data object, which need to be sent to server.
- dataType - specifies the type of data through which communication happens between client & server. (JSON / XML / string / JSONP etc).
- crossDomain - A boolean value specifies whether the communication is between crossdomain or same domain.
- async - Boolean value specifies whether the communication should be a synchronous & asynchronous communication.
- success - Callback method gets invoked when there is a successful communication to the provided webservice.
- error - Callback method gets invoked when there is error while communicating with server.
etc

Syntax:-

```
$.ajax({
url: 'webservice url';
method: 'GET/POST',
dataType: {... data ..},
success: function(response){
    // response is an object been thrown from server
},
error: function(error){
    // error object holds all the extra information on
    current error while communicating with server.
}
});
```

- Assignment (through JSON)



02/11/19

(47th class)

* Limitations of AJAX :- Due to security the main limitation we have in AJAX is it doesn't allow communication to be happen between crossdomains.

In a case if a sender & receiver with different domains have a handshaking to send or receive data between them following are the two ways we can establish communication between crossdomains :-

(i) JSONP

(iii) CORS setup (Cross origin Resourse Sharing Policy)

* JSONP Datastructure :- It is almost like JSON object structure where the only difference is in JSON it is just an object structure whereas in JSONP object comes as parameter within a user defined function.

Syntax :-

JSON :-

```
{
"name" : "Raj",
"age" : 10
}
```

JSONP :-

```
Customfunction({"name":"Raj", "age":10})
```

(ii) It is the only data format through which we establish communication between ~~dts~~ crossdomains.

Following are the properties need to be changed or added for the AJAX method while creating a communication between crossdomains using JSONP datastructure or data format.

03/11/19 (Sunday 3hours class) (48th class)

CrossDomain: true, dataType: 'JSONP', jsonpCallback: '<callback method name>'

* CORS Setup (Cross Origin Resource Sharing):- In this way we update the configuration of the server so that it can be accesseble for the specified domains.

The steps to be followed to setup CORS within the server changes

03/11/19

according to the type of server we are using.
Following are the instructions need to added under app.js file if the CORS setup is for nodeJS server with Express.

```
app.use(function(req, res, next){
  res.header("Access-Control-Allow-Origin", "*");
  res.header("Access-Control-Allow-Headers", 'Origin, X-Requested-With, Content-Type, Accept");
  next();
});
```

* Creating Multiple AJAX calls and performing a particular job based on the response of multiple Webservices

- jquery promises :- The concept through which we could able to invoked a particular job only after there is a response from multiple webservices.

  Syntax to create a promise in jquery:-

```
var promiseReq = $.ajax({
  url: "service url",
  dataType: '...',
  ......
});
promiseReq.done(function(){
  ... // callback function to handle success response
});
promiseReq.fail(function(){
  ....// callback function to handle error response
});
```

- Handling multiple promises :- "$.when()" is a predefined method takes multiple promise objects as input and invokes the corresponding then method only after there is a response from all the provided promises.

03/11/19

Eg:

```
$.when(promise1, promise2, ....).then(callback1, callback2, ...);
// then method will be only invoked when there is a response from all the provided promises.
```

* Templating:- The process of creating a static html without any data, dynamically taking a copy of template, injecting required data into it, adding the resulting html on to the page is called templating. There are lots of external javascript libraries available helps in implementing templating concept in a simpler way like mustache js, JADE/Pug, handlebars, jquery Load template plugin etc.

Following are the steps need to be followed to implement templating within an application using jquery Load template API.

Step 1:- Download & include jquery Load template API.

Step 2:- From url http://plugins.jquery.com/loadTemplate/

Step 3:- Create a static html code/structure with only static content but not the actual data.

Step 4:- Keep the static html code under a script tag having a unique id.

Step 5:- Under the static html use the following predefined attributes through which we can specify what object data need to be injected to a particular html element.

- data-content:- Adds the object text as element content.
- data-src:- Adds as source attribute to value.
- data-id:-
- data-class:-
- data-name:-

etc.

Step 6:- When the template need to be injected to the page use the following predefined method.

- Syntax:-

```
container.loadTemplate(<reference of template>, <object with data>);
```

03/11/19

the above method takes the template element 'injects the data to ~~it~~ it from the provided object, adds the complete html content on to the specified container

04/11/19
05/11/19

* Jquery Selectors:- Following are the different ways we could refer an existing dom element or page.
- $("#abc") - Returns the reference of an element having id as abc.
- $(".abc") - Returns the reference of an element having class as abc.
- $("div") - Returns the reference of all elements having tag name as div
- $("div#abc") - Returns the reference of a div tag having id as abc.
- $("input[type=text]") - Returns the reference of all input type text elements.
- $("div:even") - Returns the reference of all div elements which are in even position.
- $("td:odd") - Returns the reference of all td elements in odd position.
- $("div:eq(2)") - Returns the reference of all div tags which are in 2nd position. (specifically position)
- $("span:first-child") - Returns reference of all span elements which ever is in first child position.

etc.

* Jquery Traversing Methods:- Following are different ways we could traverse between dom elements
- Traversing Childrens:
  - element.children() - Returns the immediate children of element.
  - element.find("*") - Returns all the childrens and grand children of an element
  - element.find("<pattern>") - Returns all the childrens which are matching the provided pattern.
- Traversing Ancestors:-
  - element.parent() - Returns reference of immediate parent
  - element.parents() - Returns reference of parent & ancestors until html tag.

05/11/19

- element.parentsUntil("<pattern>") - Returns reference of all parents until it finds a parent with provided pattern.

* Traversing Siblings:-
  - element.next() - Returns reference of immediate next siblings
  - element.prev() - Returns reference of immediate previous sibling
  - element.nextAll() - Returns reference of all next siblings
  - element.prevAll() - Returns reference of all previous siblings
  - element.nextUntil("<pattern>") - Returns reference of all next sibling elements until it finds an element with provided pattern.
  - element.prevUntil("<pattern>") - Returns reference of all previous sibling elements until it finds an element with provided pattern.

* Implementing Event handling through jquery methods:-
-> Following are predefined methods through which we can implement event handling on dom elements.

- element.click(<callback method>);
- element.hover(<callback method>);
- element.change(<callback method>);
- element.focus(<callback method>);
- element.mouseover(<callback method>);
- element.keyup(<callback method>);
- element.keydown(<callback method>);
  etc.

* Event handling through on method:-

element.on("<type of event>", <callback method>);

e.g:-

```
element.on("click", function(event){....});
element.on("mouseover", function(event){.....});
etc.    element.on({
            "click", function(){    }
        });     "mouseover", function(){    }
```

05/11/19

* Predefined Methods Supported in jQuery which can be applied on any dom elements.

- element.addClass("<one or more classes>") - Used to add one or more classes to dom elements.
- element.removeClass("<classname>") - Used to remove a specified class from an element.
- element.hasClass("<classname>") - Returns true/false if the class is existing to the specified element.
- element.css("key","value") - To add a css property to elements
- element.css({"key1":"value","key2":"value"}) - To add more than one css property to an element.
- element.hide() - used to hide visibile element.
- element.show() - used to show the hidden element.
- element.contains("hello") - Returns every dom element having text content as hello.
- element.append(<element>) - Appends provided elements as a children
- element.attr("attributename","value") - used to set attribute value to an element.
- element.empty() - Remove all child elements of an element
- element.text() - Used to set or get text content of an element
- element.html() - used to get or set html content of an element
- element.val() - To get or set set value of an input element
- element.remove() - Deletes the element from DOM structure.
- element.prepend(<element>) - Prepends provided element
- element.appendTo(<parent element>) - Appends the current element to a specified parent element.
- element.toggle() - Used to toggle display property (none/block) of DOM element.

of DOM elem...

(51th class)

* Adding Animation to dom element using jquery predefined methods

- element.hide(<delay>, optional callback);
- element.show(<delay>, optional callback);
- element.toggle(<~~display~~delay>, optional callback);

e.g:- $("#abc").hide(2000, function(){...});

Animation using Show/hide Toggle:-

06/11/19

- Animation using slide:-

```
element.slideUp( optional callback, delay);
element.slideDown( delay, optional callback);
element.slideToggle( delay, optional callback);
```

- Animation with Fade effect:-

```
element.fadeIn( delay, callback optional);
element.fadeOut( delay, optional callback);
element.fade toggle( delay, optional callback);
```

- ** Jquery Master Animation method:-

→ Using "animate" method we could provide custom css properties which all need to be loaded with animation effect

Syntax:-

```
element.animate( { - object of css property,
        // set of css property with values
3, <delay value>);
        \ delay time
```

* Difference between JSON & XML:-

XML (extensible markup language):- A predefined markup language used to hold data as like a JSON object where the only difference is incase of JSON data gets stored under a key whereas in XML data is been hold under user defined self descriptive tags.

(5th class)

07/11/19 * Representing data using JSON & XML:-

- JSON:-

```
{
"bookname": "About Yourself",
"author": 'Mr. Raj Kishore',
"publisher": 'ABC tech publisher,
"price": 'Rs. 568',
 id' :  'Book-234-T',
"Barcode": "1534534534:"
```

(Node js is not server (We can create custum server)

Server

07/11/19

- XML Representation:-

```
<? xml version = "1.0" ?>
<BookDetails>
  <book id = "Book-234-T">
    <bookName> Abad Yours df </bookName>
    <author> Mr. Raj kishore </author>
    <publisher> ABC techpublisher </publisher>
    <price> Rs. 568 </price>
    <Barlode> 35345353453 </Barlode>
  </book>
  <book id = "Book-235-T" >
    ..... .....
  </book>
</BookDetails>
```

"NodeJS"

- Nodejs - A popular server was written initially by Mr. Ryan Dal in 2009. It has the high end advantages to create a server which uses javascript as programming language.
  (i) Nodejs is a very first server use javascript as a programming language at server end.
  (ii) It is an open source, cross platform javascript runtime environment developed on "chromes V8 JS Engine" which is directly into the native machine code.
  (iii) It is a light weight framework use to develop server side web application.
  (iv) It uses event driven non blocking input/output Model which makes it the right choice for data intensive real time applications

- Features of NodeJS:- Following are the highend features being supported in nodejs which are making it so popular & powerful:

07/11/19

(i) Open source:- It is an open source platform which means the copyright holder has given various rights of studying and distributing the software to any place for any purpose "commercial or non-comercial".

(ii) Highly Scalable:- As nodejs users Event mechanism, it is highly scalable & helps the server in a non-blocking response

(iii) Simple & Fast:- As nodejs is built on googles chromes V8 javascript engine, its library are highly advanced and hence instructions gets executed in a faster speed.

(iv) Non-Buffering:- The special features of nodejs makes nodejs to not buffer any data.

(v) Single threaded:- As it uses process of event looping, it is able to follow a single threaded module.

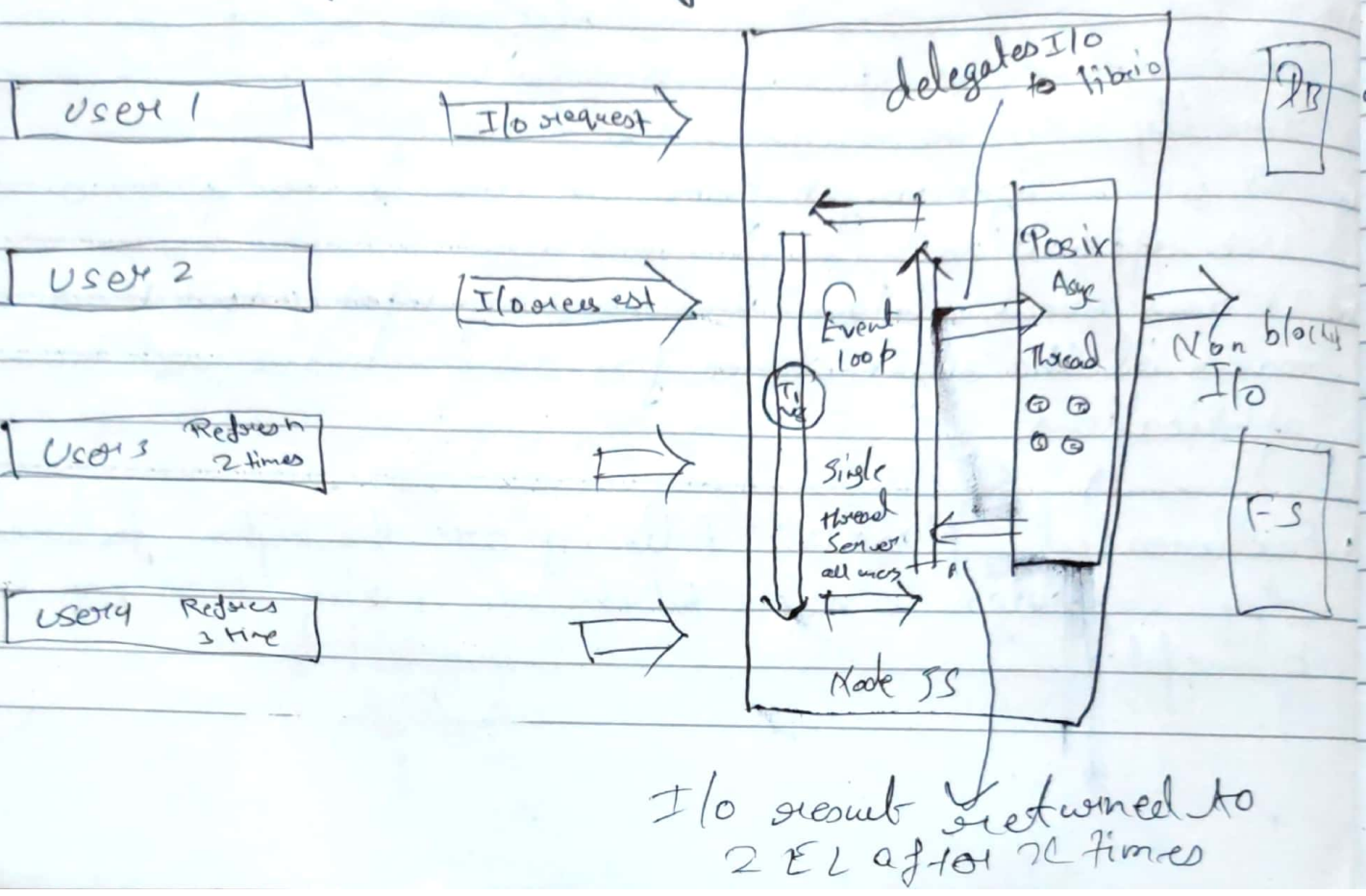
(vi) Cross platform:- Nodejs can be easily creative & deployed into cross platform like Windows, Mac, Linux etc.

* Architecture of NodeJS platform:-

08/11/19

(53rd class)

* Architecture of NodeJS platform:-



nodejs is a platform (npmjs.com → All nodejs modules)

08/11/19

libeio = library of asychoroos I/o operations (msi - microsoft software installer)

Node:- Portable operating System Inteface (POSIX)

* NPM (Node Package Manager):- Following are the main two features of npm. (A place where all code been stored)

(i) It is a online repository for the publishing of open source nodejs projects, reusable node modules.

(iii) It is a command line utility for interacting with npm repository that helps in package installation, version management & dependency management.

Syntax:-

npm install <module name>

Using the above command we could able to install the required module to the local machine. The module will be automatically added to node-modules folder.

* Steps to create a simple nodejs server using predefined node module 'HTTP':-

Step I - Download & install 'HTTP' module eg- npm install http

Step II :- Create a new javascript file which holds the complete server code.

Step III - Under the js file include the 'HTTP' module using require method. e.g:- var http = require('http');

Step IV - Create a node server using "createServer()" method coming from 'HTTP' module.

Syntax:-

```
http.createServer( function(request, response) {
.... // set of instructions to handle request, process
   and respond back with response
});
```

Under the "creatServer()" method add the actual business logic which takes the request process it & responds back.

Step V - "response.end()" is a predefined method through which we can respond back with the data.

Syntax:-

response.end(<data/msg need to send back>);

08/11/19

Step VI - Make the server to listen at a required port number using "server.listen()" method.

Note:- (i) Create server takes a callback method as a parameter, which gets invoked automatically once the server gets created.

(ii) 'Request' & 'Response' the two predefined objects available under the callback method of 'createServer()' method, the two objects holds the extra information of request been generated, response is an object through which we respond back to the user request.

(iii) 'response.writeHead()' is a predefined method through which we respond can specify the status code and the type of content been returned.

(iv) Syntax:-

response.writeHead(200, {"Content-Type": "text/plain"});

(v) * Piece of code snippett to create the simple node server using 'HTTP' module :-

```
var http = require('http');

var server = http.createServer((function(request, response){
    response.writeHead(200, {"Content-Type": "text/plain"});
    response.end("Hello World \n");
}));

server.listen(7000, function(){
    console.log("Server is listing at 7000");});
```

(node <server file>.js) // To start node server

eg:

node myServer.js)

13/11/19 (54th Class)

* Implementing File operations using node predefined FS Module:-

- FS (File Streaming):- It is a predefine module provides set of predfine methods using which we could able to perform any type of operations on files.

Step I:- Download & install FS node module e.g:- npm install fs
Step II:- Include fs module using require method
eg:- var fs = require('fs');
Step III:- Create a ~~write~~ write stream reference to the file which need to be operated using "CreateWriteStream()" method.
Syntax:-
"var writestream = fs.CreateWriteStream("<filename.extension>");"
Step IV:- Using write() method of ~~write~~ write stream reference we could able to specify the content need to be return within the file.
Syntax:-

```
writeStream.write("<user message>");
writeStream.end(); // End stream writing.
```

* Reading File Information using fs module:-

"readFile()" is a predefine method using which we could able to read data either in "binary" or "UTF-8" format
Syntax:-

```
fs.readFile('<file Name>', 'utf8', function(err, data) {
    // 'data' is data been read from file
});
```

* Node Express Module:- It is a web application framework for node js which is specifically design for building single page & multi page web applications.

- Express has become the standard server framework for node js
- Using express we could able to create or develop a node application which can be used to handle multiple request like GET, PUT, POST or DELETE & ~~PUT~~ request.
PUT
e.g:- npm install express

13/11/19

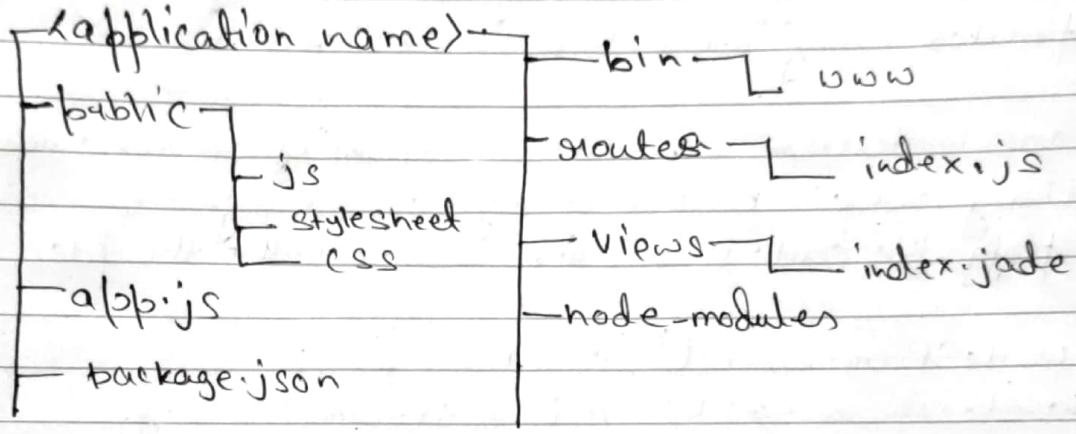
* express-generator module:- ("express-generator") - A predefine node module used to create the complete folder structure for a web server.

Syntax:-

```
npm install express-generator
```

14/11/19 (55th class)

* Following is the folder structure gets created automatically when we try to create a project structure using "express-generator module".



- 'www' file under 'bin' folder:- The first file which gets executed after npm start which 'internally includes app.js file.

→ The 'www' file holds steps to create http server, steps to handle when there is an error, steps to make the server to listen at a particular port number etc.

- public folder:- Under this folder we can add all the static resources which need to be hold by the server, these resources will never get processed at server side, the resources will be sent back as it is to the user when there is a request.
- "express.static()" is a predefined method using which we can make a given folder path/location as static files/resources holder location.

e.g:-

```
app.use(express.static(path.join(__dirname, 'public')));
```

14/11/19

- Views folder :- This folder holds list of template files in the form of .jade or .pug version.

Stateful - SOAP - Service Oriented Architecture protocal
REST - Representation State Transfer protocol
14/11/19 'Restful

- Views folder :- This folder holds list of template files in the form of .jade or .pug version. (response.render() method) (56th class)

15/11/19 * routes folder :- Under which we could able to add the piece of code which handles the request & response from client side. ('REST Services).

- Following are the steps need to follow to create a restful Services in node ~~js~~ express Server :-

Step I :- Create a .js file under routes folder.

Step II :- Add the below code to handle GET or POST type of request from CLIENT.

e.g:-
```
var express = require('express');
var router = express.Router();        // specifies the path
/* Get home page */
router.get('/', function(req, res, next) {
  // code to handle get type of request
});
/* Get home page */                   // what need to be done with this get & then it automatically execute
router.post('/', function(req, res, next) {
  // code to handle post type of request
});
module.exports = router;              // It is predefined object (Export the current into the node environment)
```

Step III :- Under the callback method add the piece of code to handle user request process & respond back.

~~e.g~~. Syntax :-

res.send() :- to send data as response
res.render() :- to send html "template" file as response

Step IV :- Under app.js file create a reference for newly created .js file (webservice).

Syntax :-

```
var reference = require('./routes/<filename>');
```

Step V :- Using "app.use()" method we specify the user defined

15/11/19

custom path through which the source can be accessed at client side:-

Syntax:-

app.use('<custom path>', <reference of router>);
(It is the reference of express)

* Handling ~~request~~ GET & POST request data in node express Application:-

(i) Reading GET data from user request:- In a express based node application 'req.query' holds the data been send by the user in the form of JSON object.

(ii) Reading POST data from user request:- "body-parser" is a predefined node module using which we could able to read the secure post data from user request.

- Following are the steps need to be followed to setup body-parser module to node express platform/environment.

Step I:- Download & install body-parser module
e.g:- npm install body-parser.
Step II:- Under app.js file include body-parser module
Step III:- Add the below code using app.use method through which the application supports JSON encoded & URL enbcoded bodies or data.

Syntax:-

```
app.use(bodyParser.json()); // to support JSON-encoded
                               bodies
app.use(bodyParser.urlencoded({ // to support URL-
    encoded bodies
    extended: true
}));
```

Note:- the extended option allows to choose between parsing url encoded data with queries storing library (when it is false) or qs library (when it is true).

15/11/19

The extended syntax allows for rich objects & arrays to be encoded into the url-encoded format allowing for JSON type of experience.

16/11/19 (57th class)

Step IV:- After adding the above code in every webservice under the current application 'req.body' holds all the post data been send from the server.

- REST:- Stateless , SOAP:- Stateful
  ↳ It is most use protocol nowdays.

* MONGODB (NO-SQL)

- MongodB:- It is an opensource Database Management System (DBMS) which uses a document oriented database model which supports various forms of data.

→ In MongodB, instead of using tables & rows to store data, it uses collections and documents.

→ It is also called as NO-SQL language, as we don't write queries to retrieve data, instead we write simple commands to manipulate data.

- Following are the high end features of mongoDB:-

(i) Each database in mongod contains sets of collections which internn contains documents. Each document can be different with a varying number of fields. The size and content of each document in a collection can be different from each other.

(ii) The document structure is more inline with how developers construct their classes & objects in their respective programming languages.

(iii) The rows or documents in mongodb doesn't need to have a predefined schema, instead the fields can be created on the fly.

(iv) The data model available within MongodB allows you to represent hierarchical relationships, to store arrays, and other more complex structures more easily.

16/11/23

* Key Components of mongodb architecture:-

1) '_id':- This is a required filed in every mongodb document. the _id field represents a unique value in the mongodB document. It is like a primary key. If we create a new document, without _id, Mongo will automatically adds a field _id with a unique random value.

2) Collection:- This is a grouping of mongodb document. A collection is the equivalence of a table which is creates in other RDBMS.

3) Cursor:- This is a pointer to the result set of a query, clients can iterate through a cursor to retrieve results.

4) Database:- This is a container for collections. In a single mongodb server, it can hold any number of databases.

5) Document:- Is a record in a mongodb collection is basically called a document, Document internally consists of field names & values.

6) Field:- A name-value pair in a document. A document can have zero or more number of fields.

* List of command can be run on Mongo shell:-

(i) help:- shows the help options.
(ii) show dbs:- shows the list of current database in mongo server
(iii) use <Databasename>:- Uses the specified database as current.
(iv) show collections:- List out all collections available in current database
(v) db.<collection name>.find():- Returns all documents in specified collection.
(vi) db.<collection name>.find({<pattern>}):- Returns all documents from collection.

REPEL - Read Evaluate Process

16/11/19

- db.createCollection("<collectionname>") - Used to create a single collection

- db.<collection name>.find({gender:"female"}) - Returns all documents from collection having gender as female
(vii) db.<collection name>.insertOne() - Inserts a new document to collection.
(viii) db.collectionname.insertMany(); - To insert many records at a time
(ix) db.collectionname.updateOne(); - To update a single record at a time
(x) db.collectionname.updateMany(); - To update more than one record at a time
(xi) db.collationname.deleteOne(); - To delete single document
(xii) db.collectio name.deleteMany(); - To delete more than one documents
(xiii) db.collectionname.drop(); - Drops the collection from database
(xiv) db.collection name.count(); - Returns total number of documents.
(xv) db.collection name.find.limit(n); - returns only n results.
(xvi) db.collection name.find.sort(<criteria>); - Returns sorted document based on provided criteria.
etc.

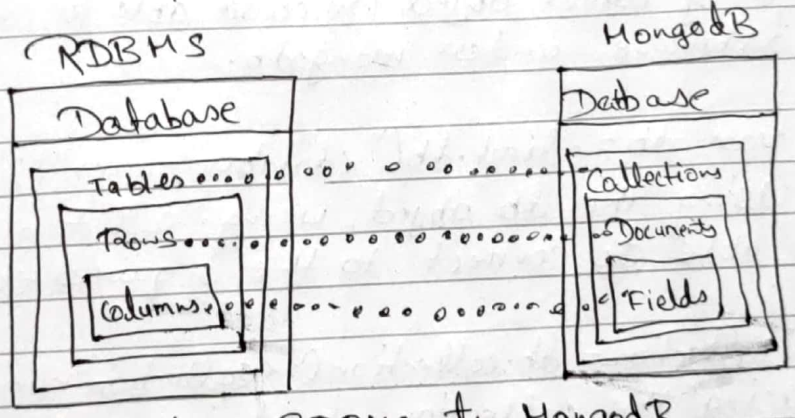


Fig:- Mapping RDBMS to MongodB

- https://www.mongodb.com/ is the dedicated page been maintained by mongodb team from these we can download the mongodb setup, it even provides the documentation of mongodb server.

(58th class)

18/11/19 19/11/19 * Steps need to be followed to establish connection (59th class) to mongo database through nodejs :-

Step 1:- Download & install "mongodb" module e.g npm install mongodb
Step 2:- Include mongodb module within the application.
Syntax:- var mongodb = require("mongodb");

19/11/19

Step 3:- Create a mongo client ~~reference~~ instance through mongodb reference

Syntax:- var mongoClient = mongodb.MongoClient;

Step 4:- Create a url with mongo server name & the port number it is running

Syntax:- var url = "mongodb://localhost:27017";

Step 5:- Connect to the mongo database using "mongoClient.connect()", method, it takes mongo url and a callback method as a parameter.

Syntax:-

```
mongoClient.connect(url, function(err, client){
    // client object is reference of connected mongodb,
    through which we could connect to databases under it.
});
```

Step 6:- Using client object we could able to connect the require database under mongodb.

syntax:-

```
var db = client.db("<database name>");
```

Step 7:- Using the db object, using "collection()" method we could able to connect to the required collection

syntax:-

```
var collection = db.collection("<collection name>");
```

Step 8:- Using the collection object we could run the required commands, handle the response accordingly.

~~syntax~~ eg:-

```
collection.find().toArray(function(err, items){
    // items holds all values been returned
});
```