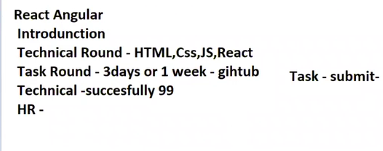
**Java script**

****

**Javascript:**

**-**Scriptting Programming language

**Why ?**

**-**to build dynamic website.

-validation

-dynamic operations on html elements**.**

**Based up on the user interactive the contect will change---dynamic.**

**(**if u enter the ligin credentials the content will c vary with others like facebook)

**Web architecture**

1. **Clint application-🡪front end,HTML,CSS,JAVASCRIPT,BOOTRAP,REACT JS,ANGULAR**
2. **Server side application-🡪backend**
3. **Data base**

database SIDE APPLICATIONS

SERVER SIDE APPLICATIONS

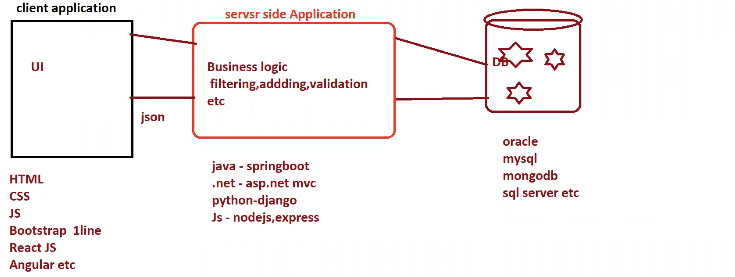
CLINT SIDE APPLICATIONS

**How the login page will work /open to next?**

**1)O**nce we setup the login into the page then send a request to the server side(backend).

2) here it will do some validations nd send to the database

3) in the data base every thing is with credentials it will send to server from server to clint side in the form of “**json” format.**

****

**What kind of application you can build js?**

**1)**web applica tion

2) standalone application--🡪run and installed in our system.-🡪electron js

3) mobile application-🡪reactive native is used for mobile application.

4) iot

5) ML

**Software requires**

1)nodejs

2) visual studio code

**WHO will run the javascript code?**

**-java**script program is a collection of statements/intructions or programs

-javascript engine: who execute the java script code.(internally crome,firezilla,)

**Node js has internally have the javascript engine.**

**MDN IS THE FAMOUS WEBSITE FOR JAVASCRIPT.**

**Variable**

**-**container for storing the data to the element:

**Rule:**

Start with \_ under score or alphabit.

**Syntax:**

**Var x ; --🡪 declaration**

**X = 55; -🡪initialization**

**Datatype**  information(var name) =”sagar”

**DATA TYPES**

1)NUMBER: 12,100,10.11(number = int,float)

2)STRING:”ababd”

3)BOOLEAN:true,false;

4) DATE:05-06-2021

5)CHAR: single letter;----‘a’

**Chapter:1**

**FUNCTIONS:** is a set of statements or instructions which are used to do particular task

**Define function:**

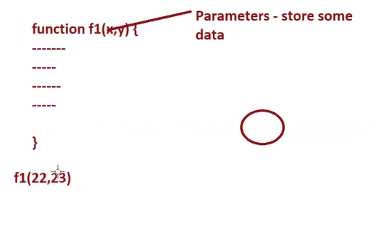
**Function my firstfunction(){**

**-------------statements or instructions**

**}**

**Flow: how to run the code.**

**Write the code inside the fn and open the folder go the open terminal and type “node space file name (be careful”**

****

**Parameters :** while define the function we pass the value/define the variable is called as “parameters”.

**Arguments: while** calling the function we pass the value to the variable is called “arguments”

**Ex:** function fnSum(x,y){

var num1=x;

var num2=y

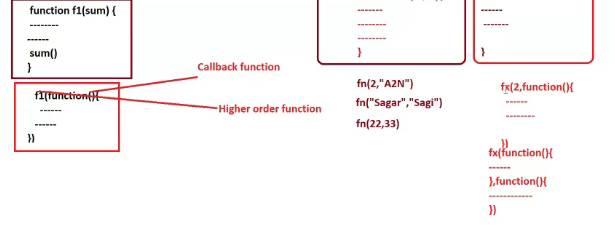
var result=num1+num2

console.log(result)

}

fnSum(20,30)

fnSum(2,3)

**callback function: **

**higher order fn:** fn having at least one argument function is called “hof”.

**callback function**:means higher order fn having the fn called “callback fn”./function which we pass as “parameter to other function is called as”**callback function**”

function fn(f1,f2){

    console.log("this is the execution part")

     f1()

    f2()

    }

   // fn(function(){

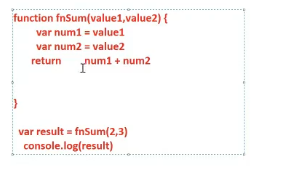
     //   console.log("calling the f1 function")

    //},function(){

    //    console.log("calling the f2 function")

    //})

**Return:**

****

**or**

function fn(a1,a2){

num1=a1

num2=a2

//return  num1+num2   or

result = num1+num2

return result

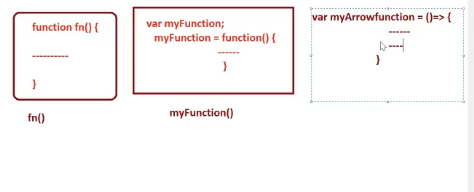
}

//result=fn(3,8) or

fn(3,8)

console.log(result)

**types of functions**

****

**1**.normal

2.var with function

3.arrow function

**\* Difference b/t normal fn and arrow fn?**

**Normal fn:** var myvarfn=function(x,y){

    num1=x

    num2=y

    return num1+num2

}

**Arrow fn:**

**Type 1:**

myvarfn=(num1)=> num1+30;

    console.log(myvarfn(20))

**when the parameter is only one “ here small bra is optional”**

**Type 2:**

myvarfn1=(num1,num2)=>{

num=num1

num0=num2

return num+num0

    }

    console.log(myvarfn1(20,12))

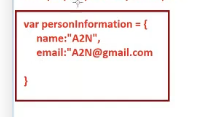
**OBJECT**

**-is a** data structure which is used to store the information.

-is a collection of property and each one having the key value.

**Data structure:**

**Structure /format the data**

****

**How the data goes to backend(server side)?**

Data/value which r there in the html page/text boxes we can access nd can be arranged by creating the **object** nd then send it to server side by requesting.

**Most imp:How to access ,delete,insert,update the object data.?**

function fnobject(){

    var myobject={

        Name:"madhu",

        email:"mangalamadhu@gmail.com",

        phone:9182674284,

        city:"chennai"

}

//console.log(myobject)

//access the object elements

console.log(myobject.Name)

//insert the object elements

myobject.city="madurai"

console.log(myobject)

//update the object data.

myobject.city="dhone"

console.log(myobject)

//delete the object data

delete myobject.city

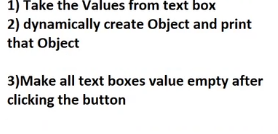
console.log(myobject)

}

fnobject()

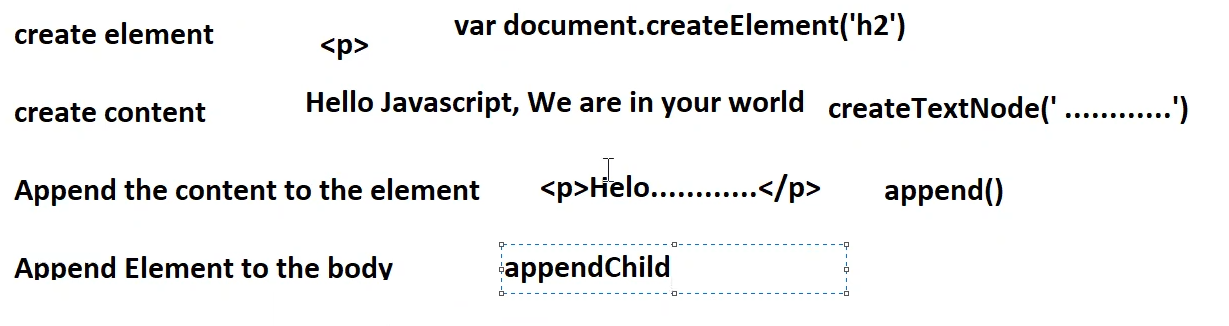
practice: if u want a code/html with meta just(“doc+enter”)

1. **How to create a dynamic form nd stored in object nd print it.**

****

1. <html lang="en">
2. <head>
3. <meta charset="UTF-8">
4. <meta http-equiv="X-UA-Compatible" content="IE=edge">
5. <meta name="viewport" content="width=device-width, initial-scale=1.0">
6. <title>Document</title>
7. <script>
8. function printform() {
9. var kanakam = document.getElementById('madhu')
10. var inputpass=document.getElementById('passwordinput')
11. var h1reference = document.getElementById('refform')
12. var amma= kanakam.value
13. var inputpasstext=inputpass.value
14. h1reference.innerText="form submitted"
15. var information={
16. Name:amma,
17. password:inputpasstext
18. }
19. console.log(information)
20. }
21. </script>
22. </head>
23. <body>
24. <div class="formality">
25. <div class="formss">
26. <h1 id="refform">login form</h1>
27. <div>
28. <label type="text">userName:</label>
29. <input class="border" type="text" placeholder="username" id="madhu"/>
30. </div><br>
31. <div>
32. <label type="text">password:</label>
33. <input  class="border"type="password" placeholder="password" id="passwordinput"/>
34. </div><br>
35. <div>
36. <input class="resert" type="button" value="login" onclick="printform()"/>
37. </div>
39. </div>
40. </div>
41. </body>
42. </html>

H**ow to create a dynamic web element?**

****

<script>

   function dynfunction() {

    var a2ref= document.createElement('p')

    var a2text= document.createTextNode('we came from tamilnadu')

    a2ref.append(a2text)

    document.getElementById('refere').appendChild(a2ref)

 }

</script>

</head>

<body>

    <div style="text-align: center;margin-top: 20px"id="refere">

        <h1> dynamic element form</h1>

        <button onclick="dynfunction()">createElement</button>

</div>

</body>

**OPERATORS**

****

**Why loop means?**

**“** to control the flow of program”.

**How to get access the attribute.**

**We** 3 pre -defined functions

**1)getattribute: it will** take the key of the attribute.

function showpass() {

var a2password= document.getElementById('password')

var killer= a2password.getAttribute('type')

//alert('working good')

console.log(killer)

**2) set attribute:** it will tahe the two fields form the attributes

**a) key b) value.**

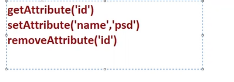
function showpass() {

var a2password= document.getElementById('password')

 a2password.setAttribute('placeholder','enter text')

console.log(a2password.getAttribute('placeholder'))

3) **removeAttribute(‘id)** : it will remove the element of that id.



How to **conver the text into password/viceversa**

var a2password= document.getElementById('password')

 if(a2password.getAttribute('type')=='password') {

     console.log(a2password.setAttribute('type','text'))

 }

 else

 console.log(a2password.setAttribute('type','password'))

}

**How to conver the text into password/viceversa along with show password and hide password.**

var a2password= document.getElementById('password')

var a12password= document.getElementById('passwordbox')

 if(a2password.getAttribute('type')=='password') {

     a2password.setAttribute('type','text'),

     a12password.value="hide password"

 }

 else

 a2password.setAttribute('type','password'),

 a12password.value="show password"

for loop functionality is much faster than while loop.

**CHAPTER:2 ARRY**

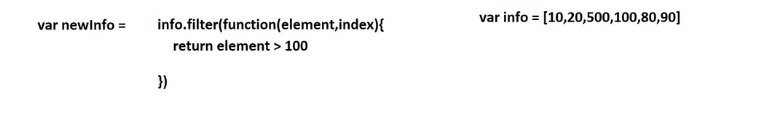
**ARRY**  IS A DS used to store the data

Var info = [“sagar”,”Madhu”,”true”]-------------🡪 each name as element in js

* Value of index start from the ‘0’.
* function fnarry() {
* var info=["indhu","bala","divya","madhu"]
* console.log(info[0])
* }
* fnarry()

**They are 23 + pre-definied functions in arrys**

1. push () : add the new value to the arry, at the last position.
2. Pop(): delete the last element from the arry.
3. Unshift(): add the element at the “**first”**
4. **Shift():** remove the element from the first.
5. Filter():used to filter an arry.



info.push("kaiyel")

//pop

info.pop("bala")

//unshift

info.unshift("anusha")

//shift

info.shift()

 console.log(info)

**filter:**

function filterarr() {

var info=[ 20,24,49,59,44,50,53,89]

     var infoarry= info.filter(function(e,i) {

return e > 20

    })

    return infoarry

}

fnarry()

console.log(filterarr())

**storages;**

function fnstore() {

         sessionStorage.setItem('name','madhu')

         sessionStorage.setItem('city','banga')

        }

        function fnget() {

        var name= sessionStorage.getItem('name')

        var city= sessionStorage.getItem('city')

        console.log(name)

        console.log(city)

        }

        function fnupdate() {

            sessionStorage.setItem('name','karim')

             sessionStorage.setItem('city','madurai')

             console.log(name)

             console.log(city)

        }

</script>

</head>

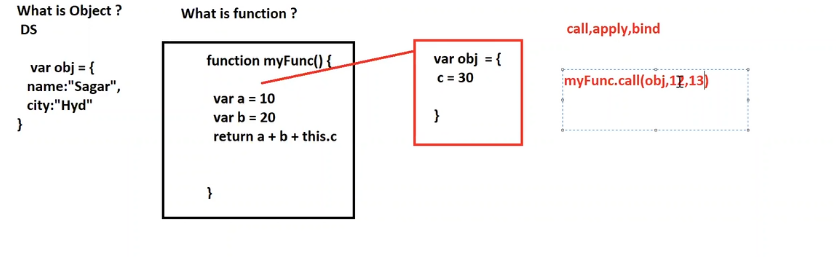
<body>

    <input type="button"  onclick="fnstore()" value="store data"/>

    <input type="button"  onclick="fnget()" value="get data"/>

    <input type="button" onclick="fnupdate()" value="update data"/>

**How to use the external objects.**



If we want to take the object from the external to inside the fn we should use **“call,apply,bind”.**

**-be concentrate at the arguments**

**1)call: it** will call the external obj and having” multiple arguments”

 var obj = {

     c:100

     }

function fncall(x,y) {

    num1= x

num2 = y

return num1+num2+this.c

}

console.log(fncall.call(obj,20,30))

2) **apply:** make the external obj to inside the fuction, but the it will take two arguments

1) obj 2) arry in arry we can pass number of variables.

var obj = {

    c:100

    }

function fncall(x,y) {

   num1= x

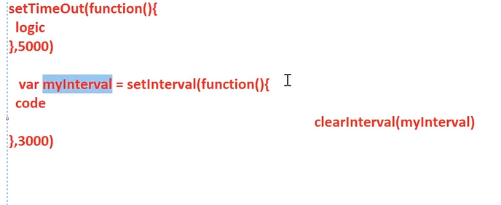
num2 = y

return num1+num2+this.c

}

console.log(fncall.apply(obj,[20,30]))

1. **Bind:**  it will return to the new function.
2. var obj = {
3. c:100
4. }
5. function fncall(x,y) {
6. num1= x
7. num2 = y
8. return num1+num2+this.c
9. }
10. var newfun= fncall.bind(obj,20,30)
11. console.log(newfun())
12. // another way we should use the bind function.
13. console.log(fncall.bind(obj,20,30)())
14. **setTimeout**: it is mainly used to call the function at a time./after what time we need to call in the milli seconds.(1000)--🡪1seconds=1000milli seconds.
15. function settimeout() {
16. console.log("before")
17. setTimeout(function(){
18. console.log("anusha is a motion girl")
19. },4000)
20. console.log("after")
21. }
22. settimeout()
23. **setInterval:** it is maily used to call the function after every 5 or respective time.



function setinterval() {

    //console.log("before")

      setInterval(function(){

        console.log("anusha is a motion girl")

    },2000)

}

    setinterval()

function setinterval() {

    //console.log("before")

var div;

    div=  setInterval(function(){

     console.log("anusha is a motion girl")

    },2000)

    setTimeout(function () {

        clearInterval(div)

    },10000)

}

setinterval()

**Hoisiting-----🡪vimp**

**“ Assigning the** memory to the variable before initialization / or execution of actual function.

****

**If the** value is defined it will allowcate some memory for that like

A =10

If u forget to declare any variable but u want to print the output of “a” means it will allowcate some memory for that too like a**=” undefined”.**

= if u defined the variable before and after the console. but not initialized --🡪undefinied.

= …--🡪 **is not defined.**

a;

console.log(a)

chapter:3 **clouser:**

**“function inside the parent function”**

**why?(see the file as clouser r)**

1. giving security that variable
2. Could not provide any access to use varible to any outside function.
3. function fb() {
4. var a=0;
5. return function() {
6. a=a+1;
7. console.log(`click me ${a} times`)
8. }
9. }
10. function fd() {
11. a = 6
12. }
14. var innerfn= fb()

**var let and const**

|  |  |
| --- | --- |
| **Var** comes under global scope(from where u can access) | **Let and** const comes under block scope |
| **Change** value of var and let is possible | **C**annot be change. |
| **Both var and let You** can declare first then u can initialize. | **Cannot po**ssible in const |
|  |  |
|  |  |

**Promise:**

Promise is a object. Which contains success or failure information. 

In order to handle that type of promise information we need one predefinied object

Called “then”

Syntax: then(function(){

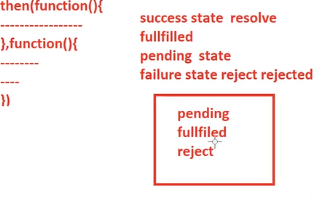
Contains the +ve information.

},function(){

Contains the -ve information.

})

3 states in promise object.



function fnpromise(num) {

return new Promise(function(resolve,reject) {

if (num%2==0){

    resolve("even number")

}else{

    reject("odd number")

}

})

}

fnpromise(20).then(function(message){

    console.log(message)

},function(message){

    console.log(message)

})

Class:

-user defined data type

-collection os some functions and properties.

**Async await** :

Fn with prefex as “**async “ is** called as **async**  function.

And called main fn with “**await”**

function fnpromise(num) {

    return new Promise(function(resolve,reject) {

    if (num%2==0){

        resolve("even number")

    }else{

        reject("odd number")

    }

    })

    }

    async function fn(){

       console.log(await fnpromise(28))

}

    fn()

**Events:**

**-**onclick

-onchange

-onmouseover

-keydown

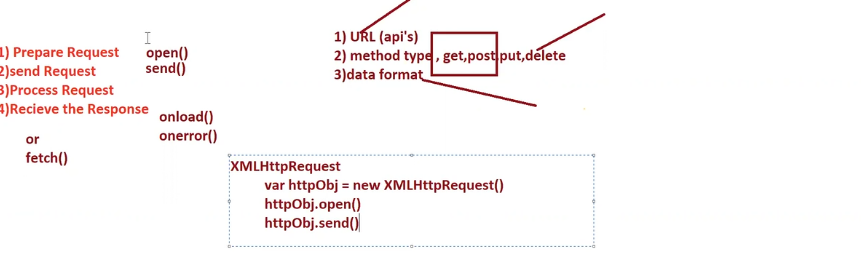
-keyup

-keypress…

**Ajaxs:**

**-In** order to send the request to the server we need Ajax.

-**A**syncronous **J**avascript **A**nd **X**ml.



When ever we prepare the request from the clint to server side

We want 4 steps:

1. prepare request: we want prepare this request we need to use the”open()”
2. send request: for “send()”
3. process request
4. receive the response: onload():on success meassage.

onerror()

all these functions are kept in the “**one class**”.

XMLHttpRequest--🡪 class name.

Constructor:”**new XMLHttpRequest()**

**Example:**

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <script src="ajax.js">

</script>

</head>

<body>

    <button onclick="fnsend()">send button</button>

</body>

</html>

**External js:**

 function fnsend() {

//alert("working")

var http=new XMLHttpRequest()

http.open('get',"https://jsonplaceholder.typicode.com/users")

http.send()

//http.onerror=function ()

http.onload = function() {

var data =http.responseText

console.log(data)

}

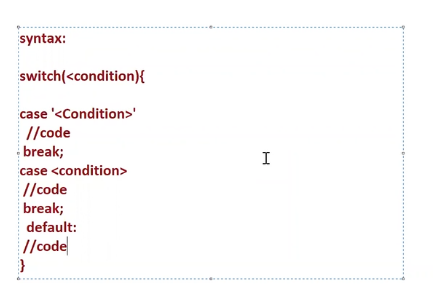
http.onerror=function () {

alert("error message while fetching")

}

 }

**SWITCH:**

****

<script src="switch.js">

 </script>

</head>

<body>

    <h3 id="h3">hi loose how r you</h3>

    <button onclick="fngoodm(event)"  id="GM"> good morning</button>

    <button  onclick="fngoodm(event)"  id="GA">good afternoon</button>

    <button  onclick="fngoodm(event)"  id="GE">good evening</button>

</body>

</html>

**External:**

function fngoodm(e) {

var h3ref = document.getElementById("h3")

switch(e.target.id)

{

case 'GM':

    h3ref.innerText="hi loose how r you goodmorning"

    h3ref.style.color="silver"

    break

    case 'GA':

    h3ref.innerText="hi loose how r you goodafternoon"

    h3ref.style.color="red"

    break

    case 'GE':

    h3ref.innerText="hi loose how r you goodevening"

    h3ref.style.color="blue"

    break

}

}