Syllabus

1. Basic - variable, loops, array,string,
2. DOM manipulation and event handlings
3. Validation using string and using regular expression
4. BOM(Browser Object Model)
5. Arrow function , rest operator, spread operator, template string
6. Promises, async await, event loop ,
7. fetch() function and fetch REST API
8. JSON + Web services =(SOP services, REST)
9. Ajax
10. OOP
11. Object literals
12. Modules
13. TypeScript

Etc

Q. What is JavaScript?

JavaScript is a client side as well as server side scripting language

Q. What is client side scripting language?

Client side scripting language means those languages required browser for execute its application called as client side scripting language

Q. What is server side scripting language?

Those languages required a server environment to execute its application and if we think of javascript nowadays node is an application which provides a server environment to execute JavaScript applications.

Q. What are the benefits of JavaScript/Why Learn JavaScript?

1. It is a dynamic type language
2. JavaScript can help us execute code on HTML events
3. JavaScript can apply runtime CSS on HTML element
4. JavaScript can create a new html element at run time or remove element at run time or replace element at run time by using DOM manipulations.
5. JavaScript provide facility to work with REST API using promises and fetch() function
6. JavaScript provide JSON format to us which is language independent format and can work as REST API data transfency medium
7. JavaScript provide AJAX feature to us which help us work with partial page updation
8. JavaScript is backbone of all JavaScript libraries and framework like as React, Angular etc
   Means if we want to learn and understand any library of javascript like as react or express we must be know about the JavaScript

   How to work with JavaScript

   If we want to use JavaScript we have two environment
   1. Browser environment /client side environment
   2. Node environment /server environment

How to use JavaScript in browser environment

There are two ways to use JavaScript in browser environment

1. **Using Internal JavaScript :** if we think about Internal JavaScript we use JavaScript code within page or within same web page
2. **Using External JavaScript :** External JavaScript means we create separate JavaScript file and include that file in HTML page using <script src /> tag

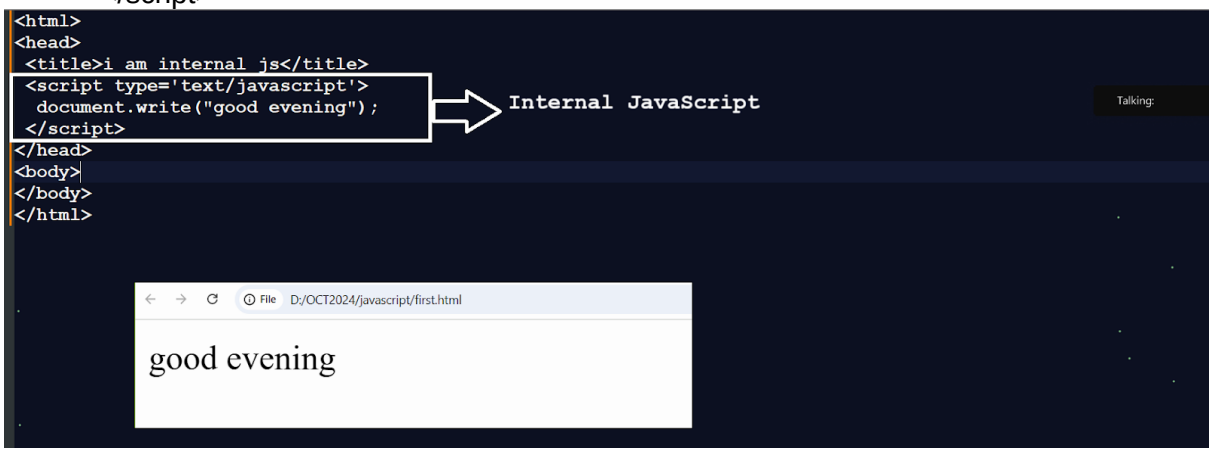
How to use JavaScript by Using Internal JS Technique

1. **Create web page using HTML**
   Demo.html

```
<html>
<head>
 <title>i am internal js</title>
</head>
<body>

</body>
</html>
```

2. **Add the following tag in HTML page**
   If we want to use the internal JavaScript we have to add <script> tag in HTML page

Syntax:<script type='text/javascript'>
Write here your logics
</script>



How to use JavaScript using External JS Technique

If we want to use JavaScript using External JS we have to create separate .js file and write JavaScript logics in it and add that .js file in HTML page using <script > tag

Syntax:<script type='text/javascript' src='filepath.js'></script>

Steps to work with External JS

---

1. **Create HTML page**

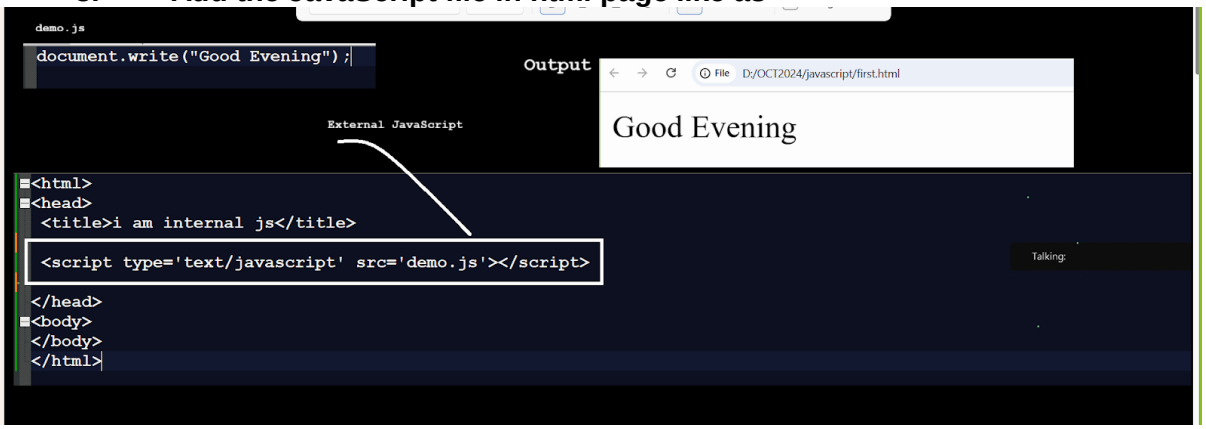
```
<html>
<head>
 <title>i am internal js</title>

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

2. **Create Separate JavaScript file with .js extension**

demo.js

```
document.write("Good Evening");
```

3. **Add the JavaScript file in html page like as**



How to use JavaScript code in Node Environment

Steps.

1. **Install Node js Application in your Machine**

Note: Node js is application which provide environment to us execute JavaScript code without browser as well as help us to provide server environment to us for execute JavaScript code and provide Sub application like as NPM which help us to download the required JavaScript libraries

Note: if we want to download the node we can visit the following URL

https://nodejs.org/en/download

Note: Download and install node like as regular software

After Installation of node if we want to cross verify node installed properly or not we have to open the command prompt and type the following command

```
C:\Users\Admin>node -v
v22.14.0
```

2. **Create JavaScript code and save it using .js extension**

Note: when we want to JavaScript code using node environment we have create separate file with extension of .js

test.js

```
console.log("Hey I am JavaScript");
```

3. Run Javascript file using following command

```
console.log("Hey I am JavaScript");
```

```
D:\OCT2024\javascript>node test.js
Hey I am JavaScript

D:\OCT2024\javascript>
```

How to declare variable using JavaScript

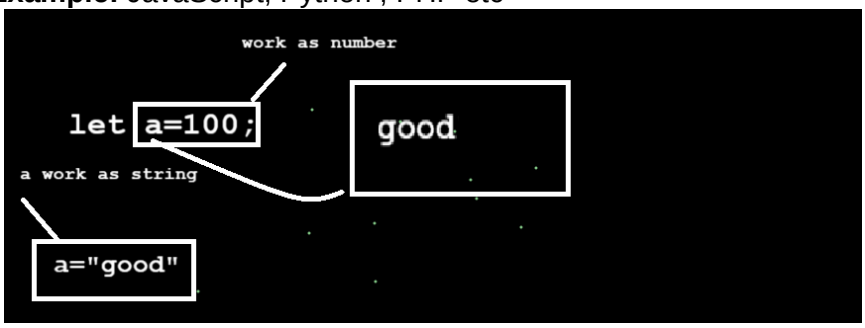
JavaScript is dynamic type language

Q. What is a dynamic type language?

Dynamic type language means a language not need to specify data type at the time of variable declaration

Variable can change its data type according values assign in it by developer called as dynamic type language

Example: JavaScript, Python , PHP etc



If we want to check type of the variable in JavaScript we have typeof() function

```
let a=100;
console.log("Now a is type  of  "+typeof(a));
a="good";
console.log("Now a is type of "+typeof(a));
a=new Date();
console.log("Now a is type of "+typeof(a));
```

Output

```
D:\OCT2024\javascript>node test.js
Now a is type  of  number
Now a is type of string
Now a is type of object
```

Q. What is static type language?

When we declare variable then must be specify data type of the variable called as static type of language

Example: C,C++,JAVA,C# etc
int a=100;

If we want to declare variable using JavaScript we have three approaches

---

1. **Using var keyword**

```
var a=100;
var b=200;
var c=a+b;
console.log("Addition is "+c);
```

```
D:\OCT2024\javascript>node test.js
Addition is 300

D:\OCT2024\javascript>
```

2. **Using let keyword**

```
let  a=100;
let b=200;
let c=a+b;
console.log("Addition is "+c);
```

```
D:\OCT2024\javascript>node test.js
Addition is 300

D:\OCT2024\javascript>
```

3. **Using const keyword**

```
const  a=100;
const b=200;
const c=a+b;
console.log("Addition is "+c);
```

```
D:\OCT2024\javascript>node test.js
Addition is 300
```

Q. What is the difference between let var and constant keyword?

---

1. Var keyword allow redeclaration variable within same block means override new declare variable space on previous memory and let and const not allow redeclaration of variable within a same block

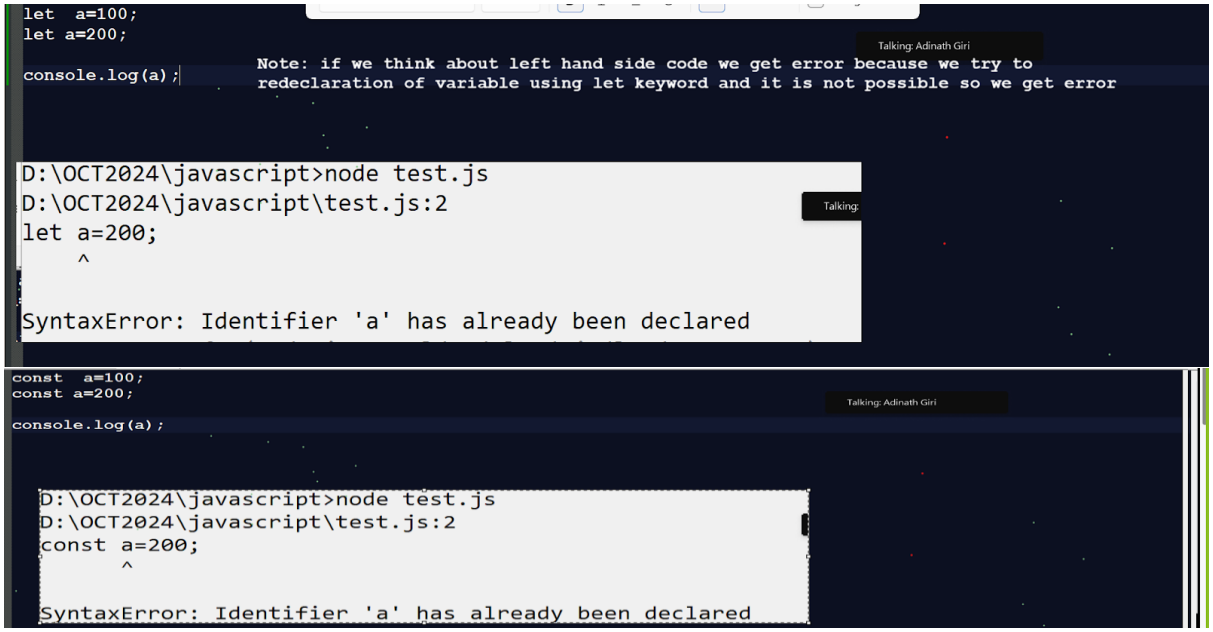
Example using var keyword

```
var a=100;
var a=200;

console.log(a);
```

```
a
200
```

```
D:\OCT2024\javascript>node test.js
200

D:\OCT2024\javascript>
```

Example using let and const keyword



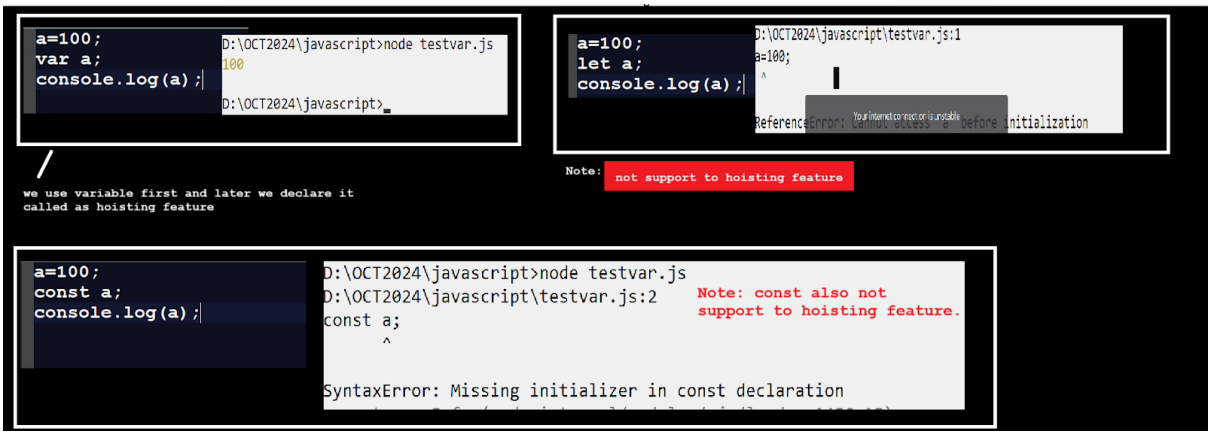
2. Var and let keyword allow reinitialization of variable and const keyword not allow reinitialization of variable.

3. Var keyword has function level scope and let and const has block level scope

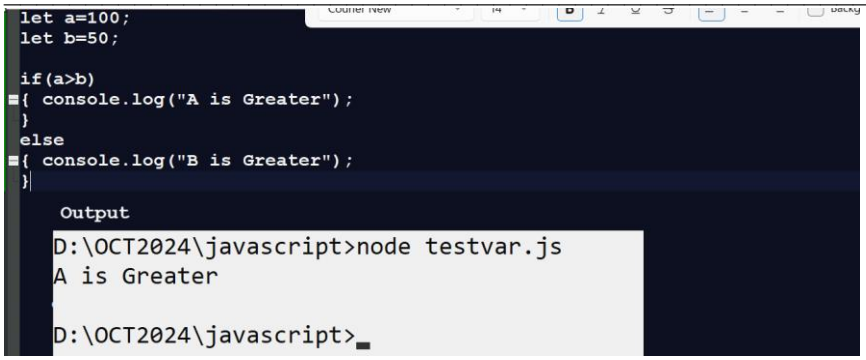
4. var keyword support hoisting feature of JavaScript but let and const not

Q. What is a hoisting feature?

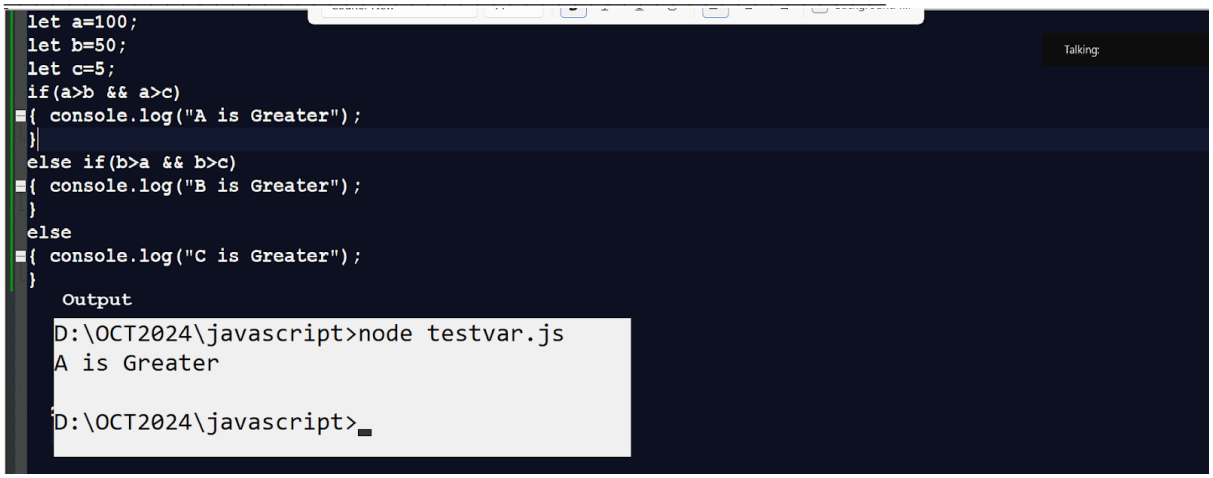
Hoisting is a feature in JavaScript where we can use a variable with value before declaration and after that we can declare it using let or var keyword but it is not possible using const keyword.



How to use If else using JavaScript



How to use else if ladder



How to use switch statement using JavaScript

```
let a=100;
let b=50;
let choice=1;

switch(choice)
{
     case 1:
     console.log("Addition is  "+(a+b));
     break;
     case 2:
     console.log("Multiplication is "+(a*b));
     break;
     case 3:
     console.log("Division is "+(a/b));
     break;
     default:
      console.log("Wrong choice");
}
```

```
D:\OCT2024\javascript>node testvar.js
Addition is  150

D:\OCT2024\javascript>
```

How to use for loop and while loop using JavaScript

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

```
D:\OCT2024\javascript>node testvar.js
Addition is  150

D:\OCT2024\javascript>node testvar.js
Good Morning
Good Morning
Good Morning
Good Morning
Good Morning

D:\OCT2024\javascript>
```

```
let i=1;
while(i<=5)
{
     console.log("Good Morning");
     i++;
}
```

```
D:\OCT2024\javascript>node testvar.js
Good Morning
Good Morning
Good Morning
Good Morning
Good Morning

D:\OCT2024\javascript>
```

```
let i=1;
do{
    console.log("good morning");
    i++;
}while(i<0);
```

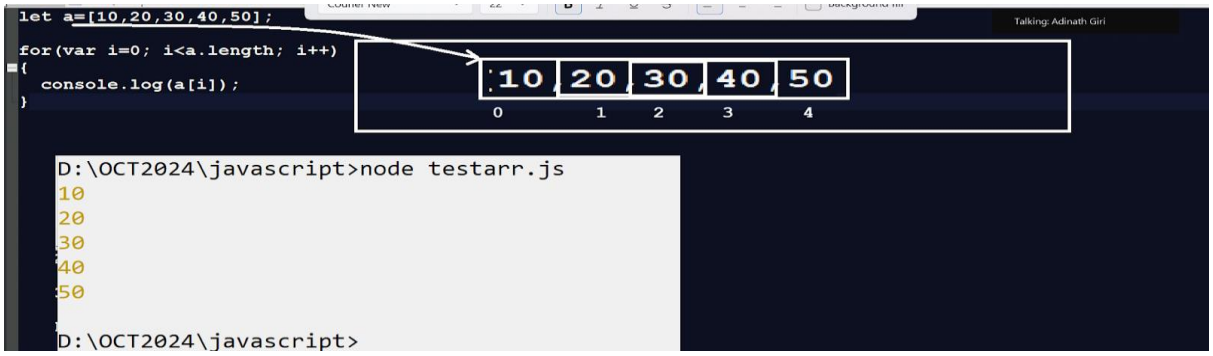
```
D:\OCT2024\javascript>node testvar.js
good morning

D:\OCT2024\javascript>
```

Array Using JavaScript

How to create an array in JavaScript?

If we want to create array in JavaScript we have two ways

1.using [ ] (subscript)

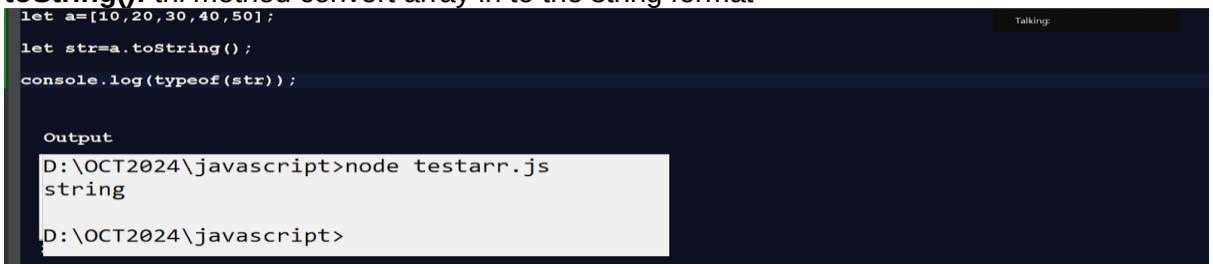


2. Using Array class

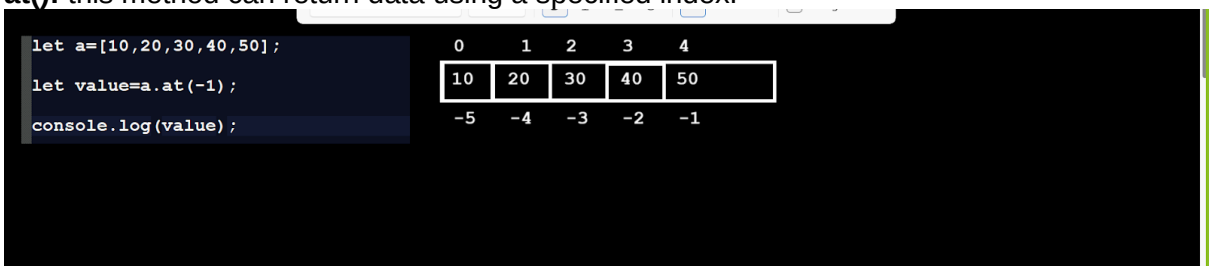
If we want to work with Array in JavaScript we have some inbuilt method provided by JavaScript to us

length: this is the constant which return size of array

toString(): thi method convert array in to the string format



at(): this method can return data using a specified index.



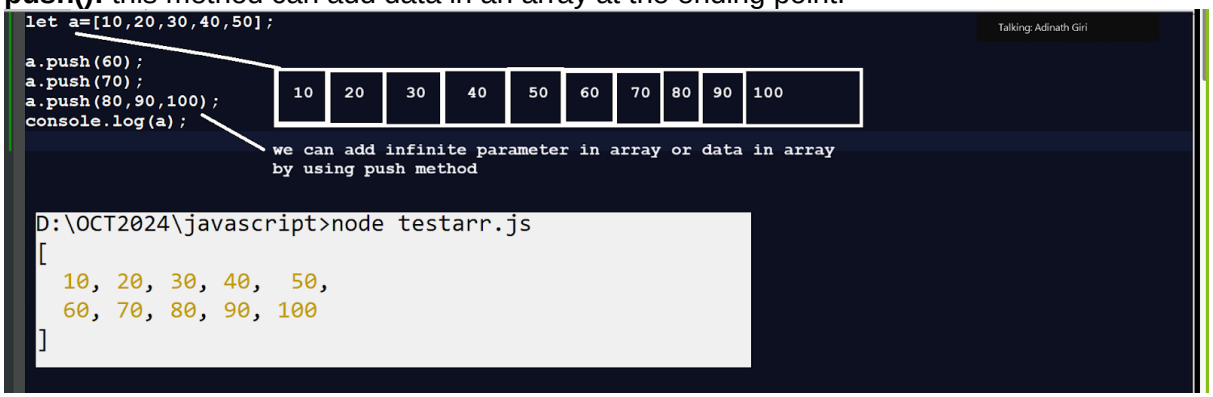
Note: at() method can accept data or fetch data from array using indexing means we can fetch data using negative indexing also.

Means we pass negative index in at() method then we get backward traveling

join(): this method join() all array elements into string. This method behaves like toString() but in addition you can specify the separator.

```
let a=[10,20,30,40,50];
let s=a.join("*");

console.log(s);


 Output
D:\OCT2024\javascript>node testarr.js
10*20*30*40*50

D:\OCT2024\javascript>
```

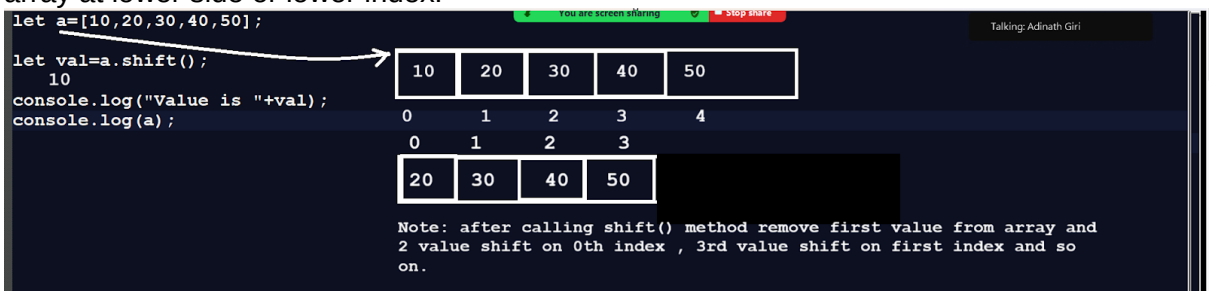
push(): this method can add data in an array at the ending point.



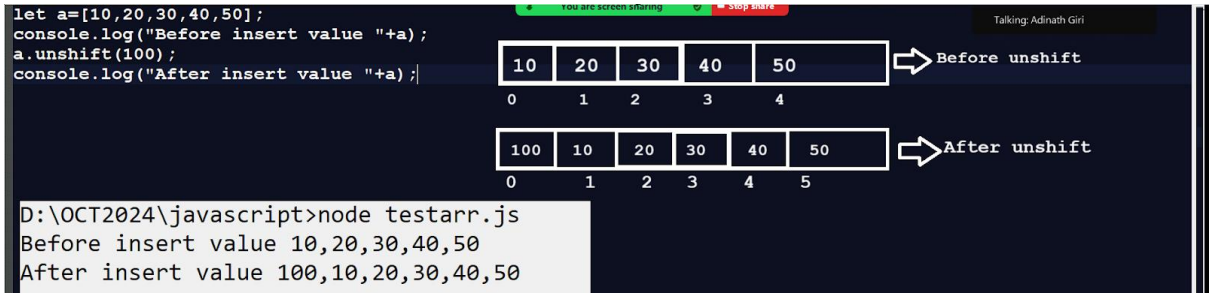
pop(): this method is used to remove the last data from the array.

```
let a=[10,20,30,40,50];

let val = a.pop();

console.log("Removed data is "+val);

console.log("After remove array is "+a);

  Output
D:\OCT2024\javascript>node testarr.js
Removed data is 50
After remove array is 10,20,30,40

D:\OCT2024\javascript>
```

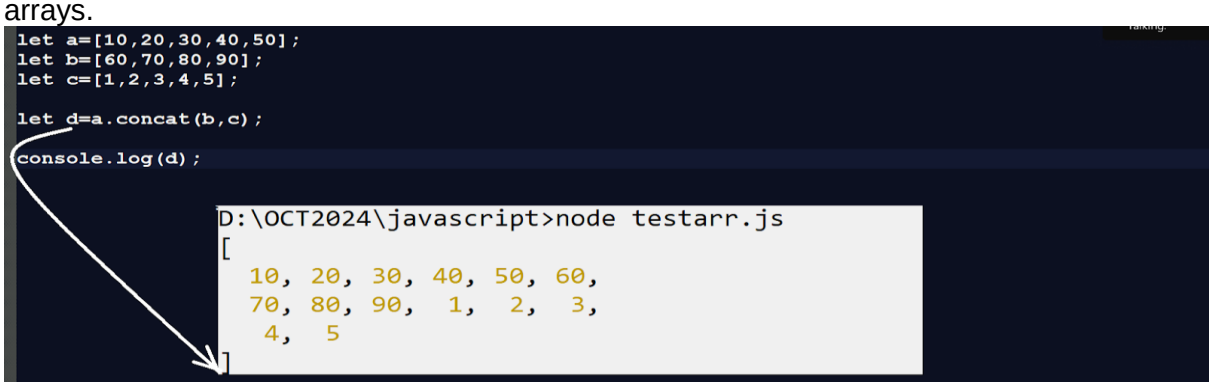
shift(): this method can remove the first element from the array and shift all indexes of the array at lower side or lower index.



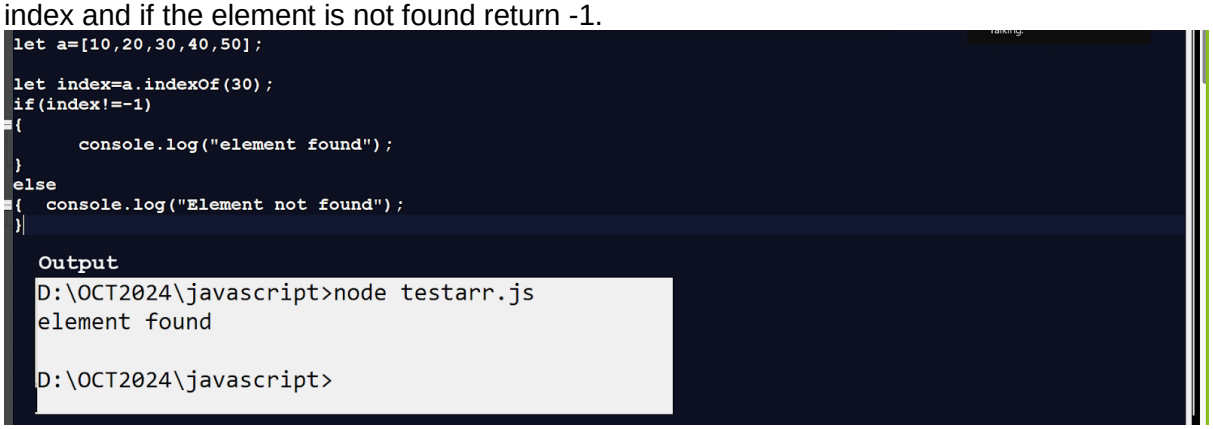
unshift(): unshift() method is used for insert data on 0th index and move index by 1 at upper side.



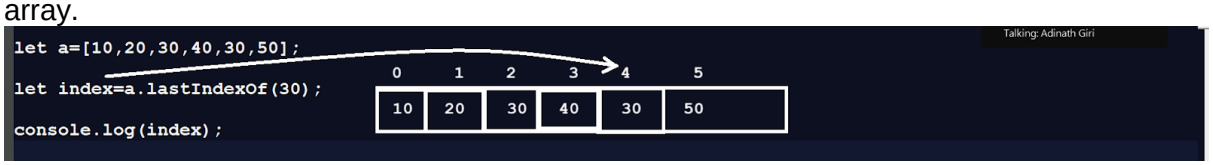
concat(): this method can concat two or more than two arrays at time and generate new arrays.



indexOf(): this method searches an element in an array and returns its first occurrence index and if the element is not found return -1.



lastIndexOf(): this method can search data from array and return last occurred element index means if we have duplicated search value in array then we get last occurred index of array.



Sorting related methods of array

---

sort(): this method can sort the array.

```
let arr=["good","abc","bad"];

let result = arr.sort();

console.log(result);
```

Output

```
D:\OCT2024\javascript>node testchar.js
[ 'abc', 'bad', 'good' ]

D:\OCT2024\javascript>
```

sort () method for number sorting purpose

```
let a=[10,5,6,8,2];

console.log(a);
let result =a.sort(function(a,b){
    return a-b;
});
console.log(result);
```

Note: if we want to sort any numeric value we required to use following syntax of sort method

```
D:\OCT2024\javascript>node testarr.js
[ 10, 5, 6, 8, 2 ]
[ 2, 5, 6, 8, 10 ]

D:\OCT2024\javascript>
```

reverse() method of array

```
let a=[10,5,6,8,2];

console.log(a);
let result =a.reverse();
console.log(result);
```

```
D:\OCT2024\javascript>node testarr.js
[ 10, 5, 6, 8, 2 ]
[ 2, 8, 6, 5, 10 ]

D:\OCT2024\javascript>
```

Iteration methods of Array

---

forEach(): this method helps us to fetch data from an array.

```
let a=[10,5,6,8,2];

a.forEach(show);

function show(value,index,a){
    console.log(value);
}
```

```
D:\OCT2024\javascript>node testarr.js
10
5
6
8
2

D:\OCT2024\javascript>
```

Or

```
let a=[10,5,6,8,2];

a.forEach(function (value,index,a){
    console.log(value);
});
```

```
D:\OCT2024\javascript>node testarr.js
10
5
6
8
2

D:\OCT2024\javascript>
```

Or

```
let a=[10,5,6,8,2];
a.forEach((value,index,a)=>console.log(value));
```

```
D:\OCT2024\javascript>node testarr.js
10
5
6
8
2
```

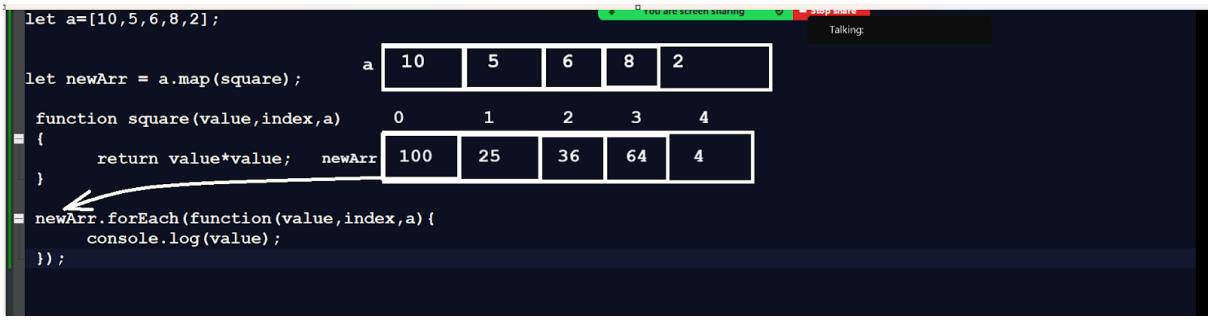
map() function

---

map() function is used to travel the array and perform operation on every element in the array and return data in the new array.

Syntax: map(function(value,index,array){
  Write here your logics
});

Example: we want to travel the array and calculate square every element in the array and store data in the new array.



Or

```
let a=[10,5,6,8,2];

let newArr = a.map(function  (value,index,a){
            return value*value;
 });

 newArr.forEach(function(value,index,a){
            console.log(value);
 });
```

Or

```
let a=[10,5,6,8,2];
let newArr = a.map((value,index,a)=>value*value);

 newArr.forEach(function(value,index,a){
            console.log(value);
 });
```

Or

Or

```
let a=[10,5,6,8,2];
a.map((value,index,a)=>value*value).forEach((value,index,a)=>console.log(value));
```

filter() method

The filter() method create a new array with array elements using a specified condition
Or that pass the test

.

```
Syntax:

let newarray= filter(function(value,index,array){
    write here your logics
});
```

Example: we want to filter only odd elements from the array.

```
let a=[10,5,6,8,2,9,11];

let newArray=a.filter(function(value,index,a){
    return value%2==1;
});

newArray.forEach(function(value,index,a){
    console.log(value);
});
```

```
D:\OCT2024\javascript>node testarr.js
5
9
11
```

reduce() method

This method is used for perform operation on array element and generate single value result like as sum of all elements from array etc

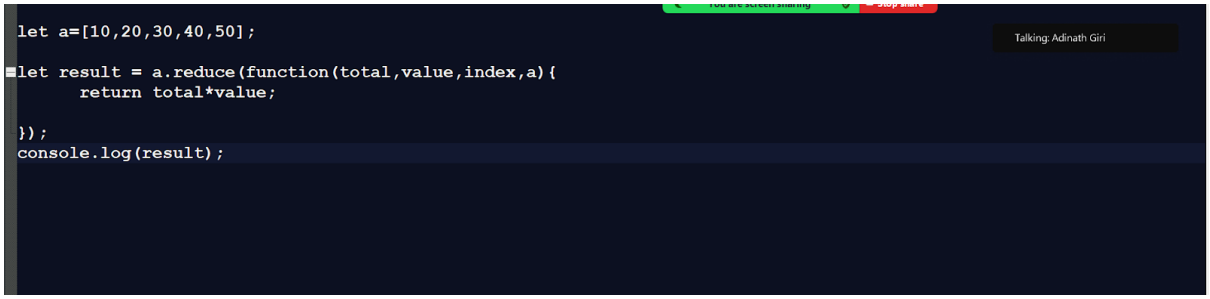
Example: Suppose we have array with five values and we want to calculate the sum of all values and return it

Syntax: let variable =reduce(function(total,value,index,array){
});

```
let a=[10,20,30,40,50];

let result = a.reduce(function(total,value,index,a){
    return total+value;

});
console.log(result);
```

```
D:\OCT2024\javascript>node testarr.js
150

D:\OCT2024\javascript>
```

Example: we want to calculate multiplication of all elements and return its result as single value like as 10*20*30*40*50



Event Handling using JavaScript

We can call JavaScript functions or logics on specified events on an HTML element
So we have number of event we can use with html elements

1. **onclick :** onclick this event fire or execute when we click on button
2. **Onmouseover :** this event fire or execute when we enter cursor on any html element
3. **Onmouseleave:** this event fire or execute when we release the mouse from any html element
4. **Onchange** : this event fire when we change state of html element like as select checkbox or drop down element etc
5. **Onload :** normally we use this event with the body of the web page and this event executes when we load the web page.
6. **Onkeyup :** this event normally we use with textbox or text control when the user presses a button and releases it immediately called onkeyup.
7. **Onkeypress:** this event normally we use with textbox or text control when user press button for long time and after that release then this event fire
8. **Onfocus:** when the cursor enters in any html element or textbox then the onfocus event gets executed.
9. **Onblur:** onblur event means when the cursor releases the control then this event is known as onblur event.

Etc

How to work with event practically In JavaScript

Steps to work with event practically in JavaScript

---

a. Define function in Java and write logic in function

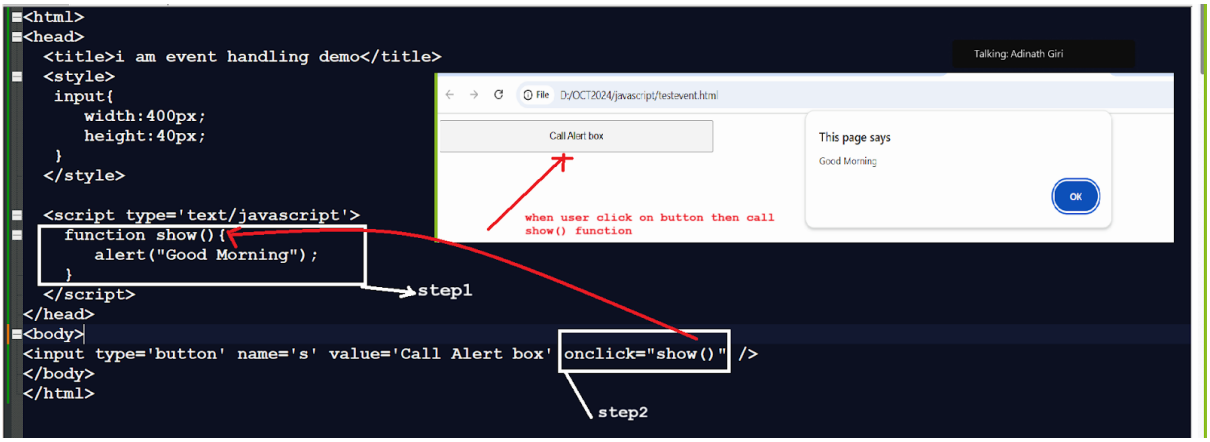
Syntax:

```
function functionname(){
  Write here your logics
}
```

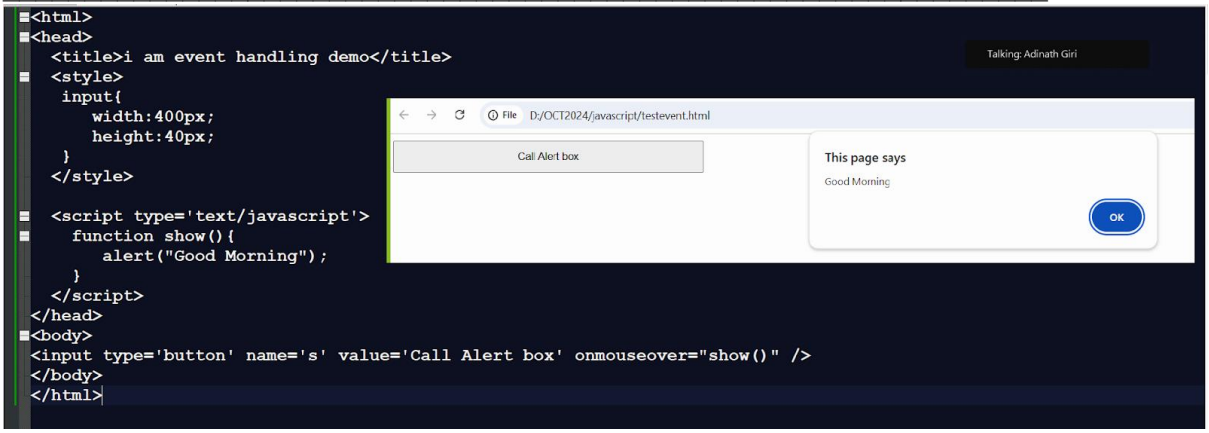
b. Use the event with html control and call JavaScript function on that event

Syntax: <html element eventtype=”functionname()” />

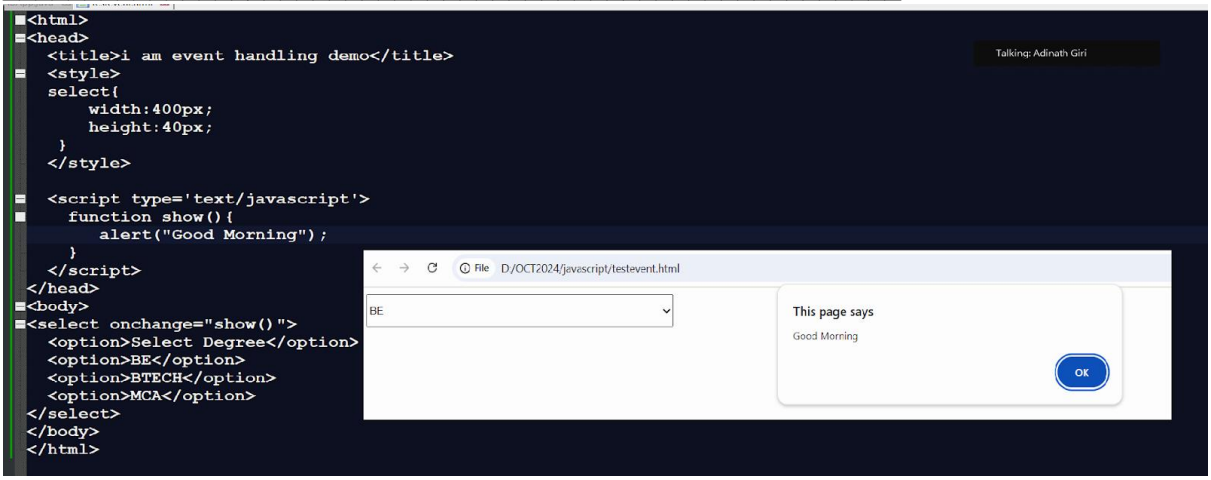
Example: we want to design a web page with one button and when the user clicks on the button then we want to call the alert box good morning on button click.



Example of onmouseover event

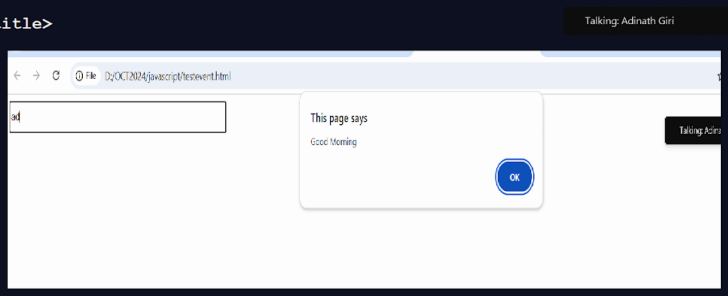


Example of onchange event



Onkeyup

```html
<html>
<head>
  <title>i am event handling demo</title>
  <style>
 input{
      width:400px;
      height:40px;
   }
  </style>

  <script type='text/javascript'>
    function show(){
       alert("Good Morning");
    }
  </script>
</head>
<body>
 <input type='text' name='name' value='' onkeyup='show()'/>
</body>
</html>
```



DOM

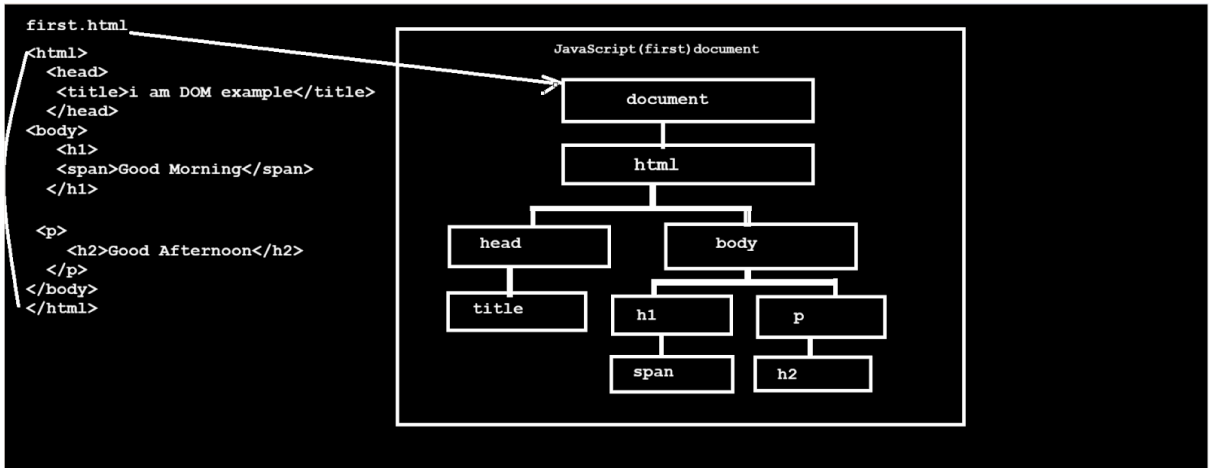
DOM stands for document object model which is used for manipulating HTML elements or creating html elements or removing html elements using JavaScript.

Q. What are the benefits of DOM?

---

1. Access HTML element in JavaScript code
2. Create New HTML element using JavaScript and add on web page
3. Remove HTML element from web page
4. Update HTML element using JavaScript
5. Apply Runtime CSS on HTML element using JavaScript
6. Apply runtime events on HTML element using JavaScript

Note: in html we say web page and in javascript web page consider as document and html elements or tags consider as object in document so this is major JavaScript provide one default parent object to all html element known as document object.
When we load any web page and if we use DOM using JavaScript internally JavaScript engine create DOM tree of web page in memory according tags align on web page



If we want to work with DOM in JavaScript we have some inbuilt methods and property provided by JavaScript to us

Methods of DOM

---

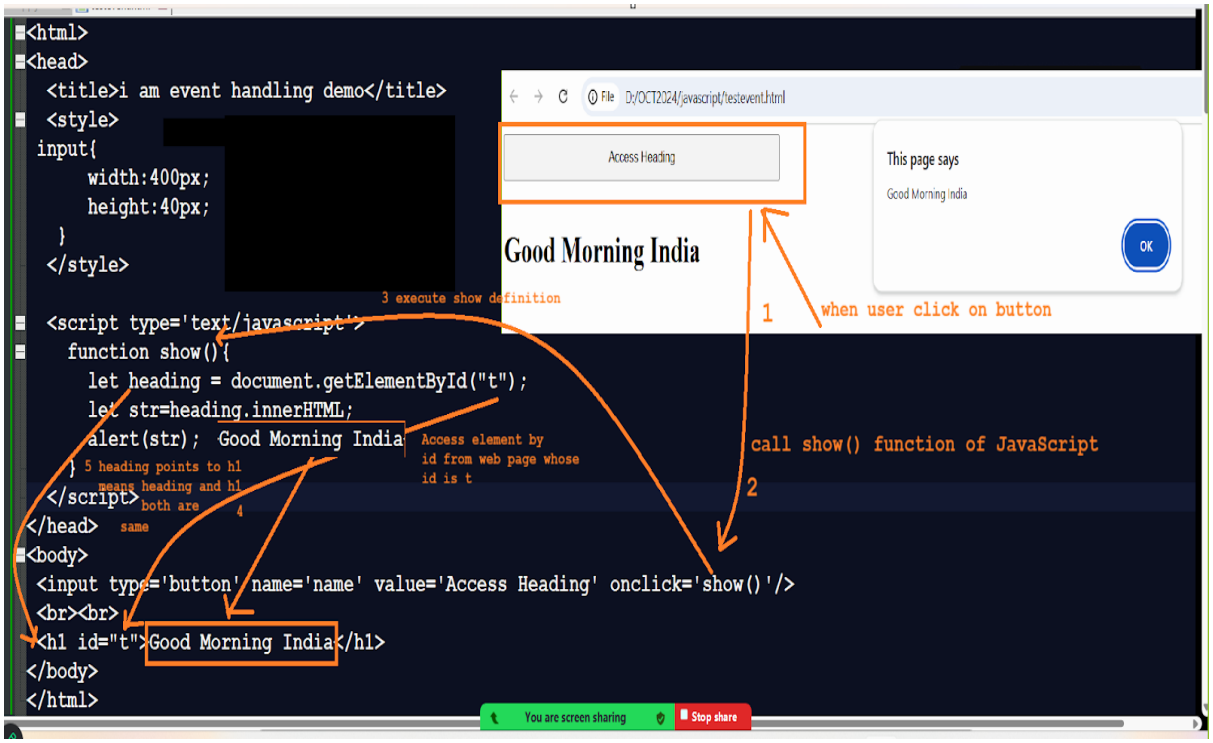
document.getElementById("idname"): this method can access any HTML element in JavaScript by using its id.
document.createElement("element"): this method can create any HTML element or tag by using JavaScript
document.appendChild("element"): this method can add any HTML element as child of another html element by using JavaScript.
element.setAttribute("name/key","value"): this method can add new attribute on with html element
document.getElementByTagNames("tagname"): this method can return array of tags from web page
document.removeElement("element"): this method can remove html element by using JavaScript

Properties of DOM

---

element.innerHTML : this property can access text or data between opening and closing of tag from a web page in JavaScript code as well as add or replace textual data between opening and closing of tag

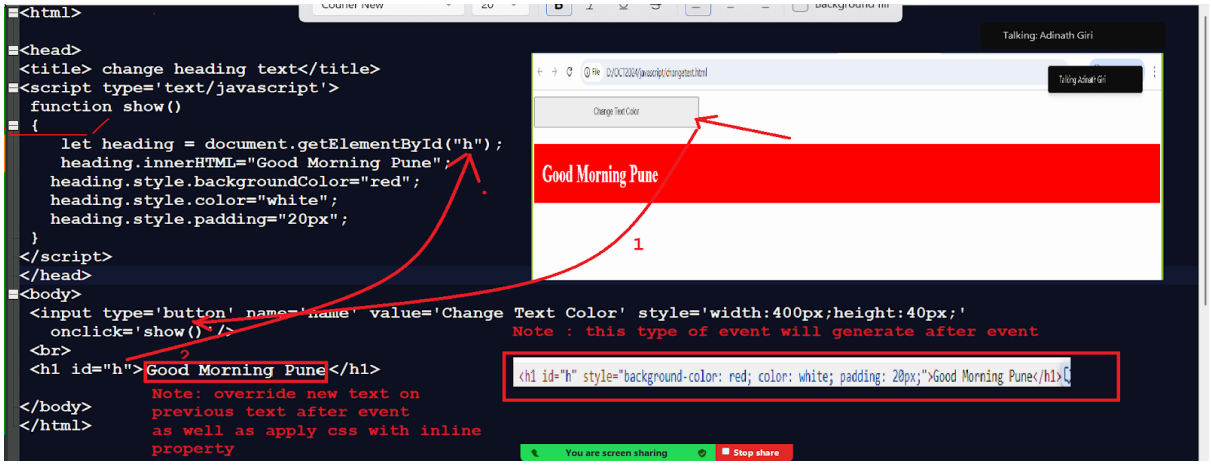
Element.attribute: this property help us to access the attribute of html element
Example: we want to design a web page with one button and heading tag and when the user clicks on the button then access text of heading javascript and display it in the alert box.



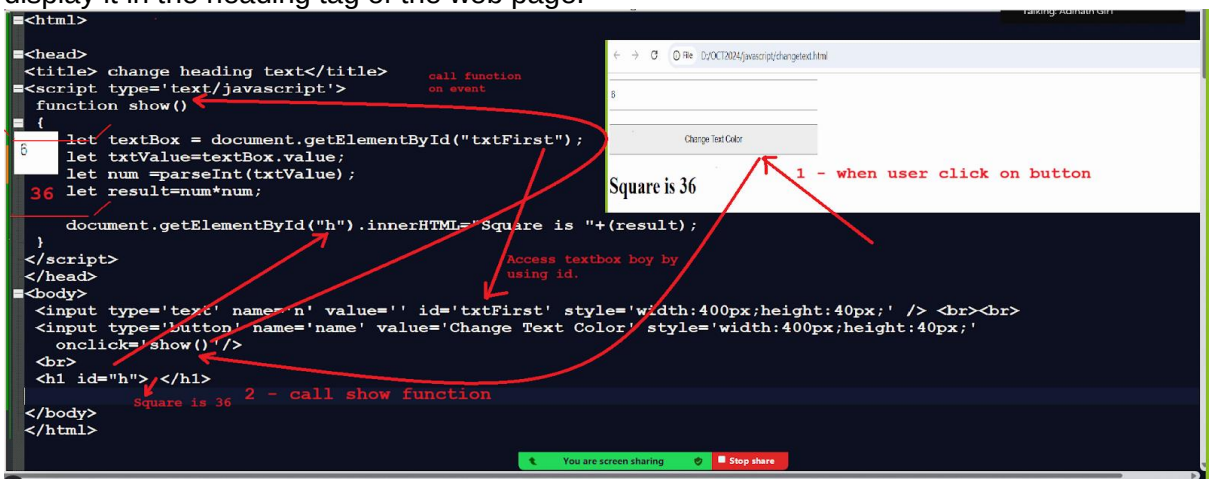
Example: WAP to create web page with one button and heading tag

---

When the user clicks on the button, change the text heading at run time.



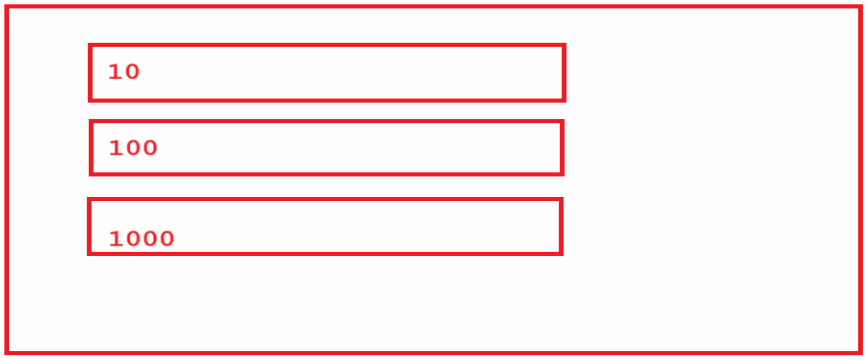
Example: we want to design a web page with one button and text box and when the user clicks on the button then we want to accept the textbox value and calculate its square and display it in the heading tag of the web page.



Example: we want to design billing application using JavaScript

---

We want to design a web page with three textboxes, one is type quantity and second is rate and third text can show the result when we input the second value.



Example with source code

<html>

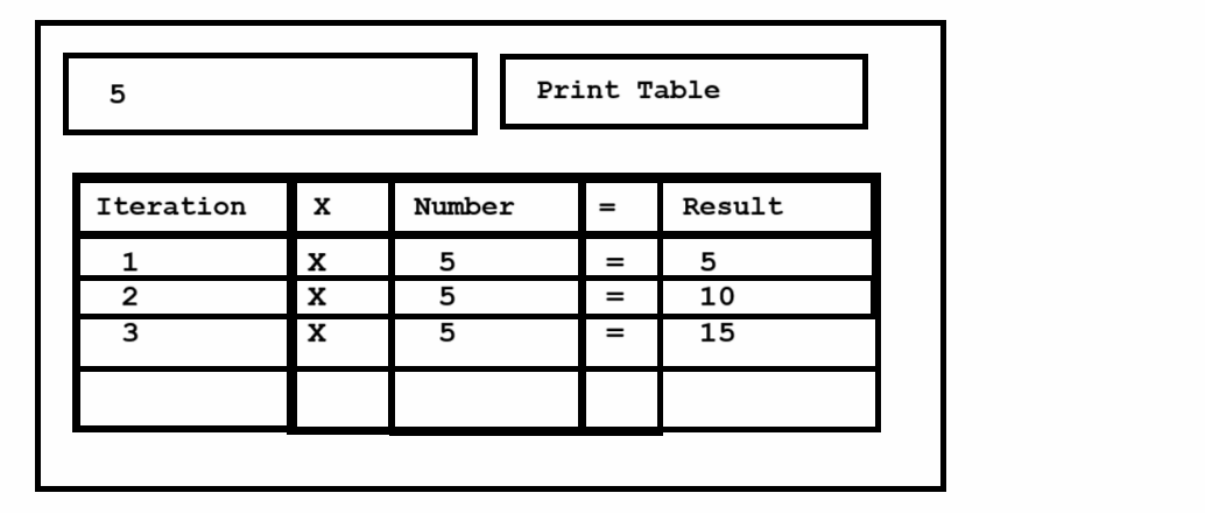
<head>
<title>Billing App</title>
<style>

```
 input{
   width:400px;
   height:40px;
 }
</style>
<script type='text/javascript'>
  let qty,rate,total;
  function acceptQty(){
    qty=parseInt(document.getElementById("qty").value);
  }
  function acceptRate(){
    rate=parseInt(document.getElementById("rate").value);
          total=qty*rate;
          document.getElementById("total").value=""+total;
  }

</script>
</head>
<body>
  <input type='text' name='f' id='qty' value='' placeholder='Enter quantity'
onkeyup='acceptQty()'/> <br><br>
  <input type='text' name='s' id='rate' value='' placeholder='Enter Rate'
onkeyup='acceptRate()'/><br><br>
  <input type='text' name='r' id='total' value='' placeholder='Show Total'/>

</body>
</html>
```

Output

Example: Emi calculator using JavaScript

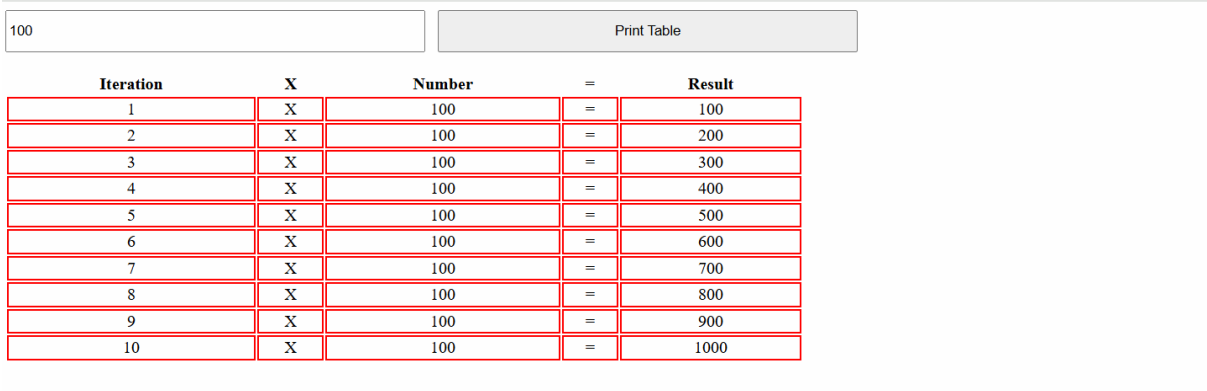
Example: we want to design a web page with one textbox and button and when the user clicks on the button then accepts the number from the textbox and prints its table.



Example with source code

```
<html>
 <head>
   <title>table example</title>
   <style>
     input{
            width:400px;
            height:40px;
          }
   </style>
   <script type='text/javascript'>
      function genTable(){
            document.getElementById("gridContainer").innerHTML="";
              let no =parseInt(document.getElementById("f").value);
              let table=document.createElement("table");
                    table.style.width="50%";

                    let tr=document.createElement("tr");
                    let th=document.createElement("th");
                    th.innerHTML="Iteration";
                    tr.appendChild(th);
                    th=document.createElement("th");
                    th.innerHTML="X";
                    tr.appendChild(th);

                    th=document.createElement("th");
                    th.innerHTML="Number";
                    tr.appendChild(th);

                    th=document.createElement("th");
                    th.innerHTML="=";
                    tr.appendChild(th);
```

```
                th=document.createElement("th");
                th.innerHTML="Result";
                tr.appendChild(th);

                table.appendChild(tr);

                for(var i=1;i<=10; i++)
                {
                   let row=document.createElement("tr");
                    row.style.textAlign="center";
                        let col=document.createElement("td");
                        col.innerHTML=""+i;
                        row.appendChild(col);
                        col.style.border="2px solid red";
                        col=document.createElement("td");
                        col.innerHTML="X";
                        col.style.border="2px solid red";

                        row.appendChild(col);

                        col=document.createElement("td");
                        col.innerHTML=""+no;
                        col.style.border="2px solid red";
                        row.appendChild(col);

                        col=document.createElement("td");
                        col.innerHTML="=";;
                        col.style.border="2px solid red";
                        row.appendChild(col);

                        col=document.createElement("td");
                        col.innerHTML=""+(i*no);
                        col.style.border="2px solid red";
                        row.appendChild(col);
                        table.appendChild(row);

                }


                let container=document.getElementById("gridContainer");
                container.appendChild(table);

            }
    </script>
 </head>
<body>
 <input type='text' name='f' id='f' value="/>&amp;nbsp;&amp;nbsp;
 <input type='button' name='b' value='Print Table' onclick='genTable()' />
  <br><br>
<div id='gridContainer'>
</div>
</body>
</html>
```



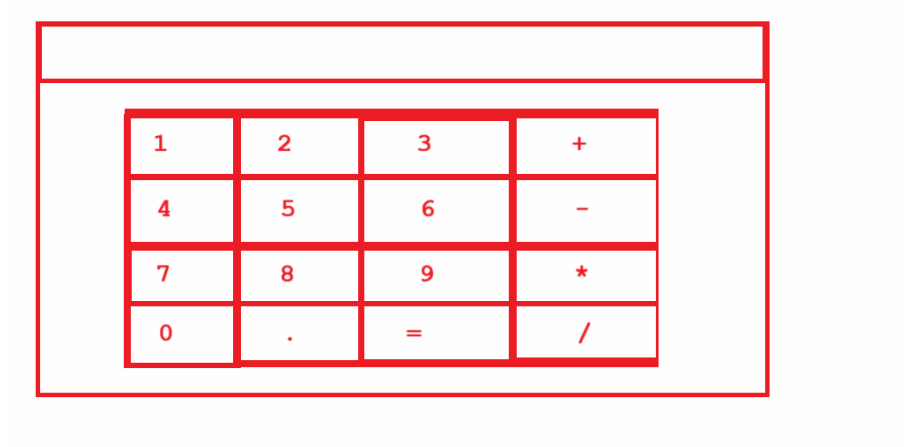
Example: we want to create web page with button and when user click on button we want to create HTML form and display on web page

```
<html>
 <head>
  <title> i am form control</title>
  <style>
    #btn{
            width:400px;
                height:40px;
          }
  </style>

  <script type='text/javascript'>
     function createForm(){
              let myForm=document.createElement("form");
                  myForm.setAttribute("name","frm");
                  myForm.setAttribute("method","POST");
                  myForm.setAttribute("action","");

                    let nameText=document.createElement("input");
                        nameText.setAttribute("type","text");
                        nameText.setAttribute("name","firstName");
                        nameText.setAttribute("id","txtName");
                        nameText.setAttribute("placeHolder","Enter name");
                        nameText.setAttribute("value","");
                        nameText.style.width="400px";
                        nameText.style.padding="20px";
                        nameText.style.marginTop="20px";

                        myForm.appendChild(nameText);


                        let emailText=document.createElement("input");
                        emailText.setAttribute("type","email");
                        emailText.setAttribute("name","emailText");
                        emailText.setAttribute("id","emailTxt");
                        emailText.setAttribute("placeHolder","Enter Email");
                        emailText.setAttribute("value","");
                        emailText.style.width="400px";
                        emailText.style.padding="20px";
                        emailText.style.marginTop="20px";
```

```
                myForm.appendChild(emailText);

                    let button=document.createElement("input");
                button.setAttribute("type","button");
                button.setAttribute("name","btn");
                button.setAttribute("id","btn");
                button.setAttribute("value","Register");
                button.style.width="400px";
                button.style.padding="20px";
                button.style.marginTop="20px";

                button.addEventListener("mouseover",function(){
                    this.style.color="white";
                        this.style.backgroundColor="teal";
                });

                 button.addEventListener("mouseleave",function(){
                    this.style.color="black";
                        this.style.backgroundColor="white";
                });
                  myForm.appendChild(button);

            let pageBody=document.getElementById("mainContainer");
            pageBody.appendChild(myForm);
        }
  </script>
 </head>
 <body id="mainContainer">
  <input type="button" name='b' id="btn" value='Create Form Using JavaScript'
onclick="createForm()" />
 </body>
</html>
```

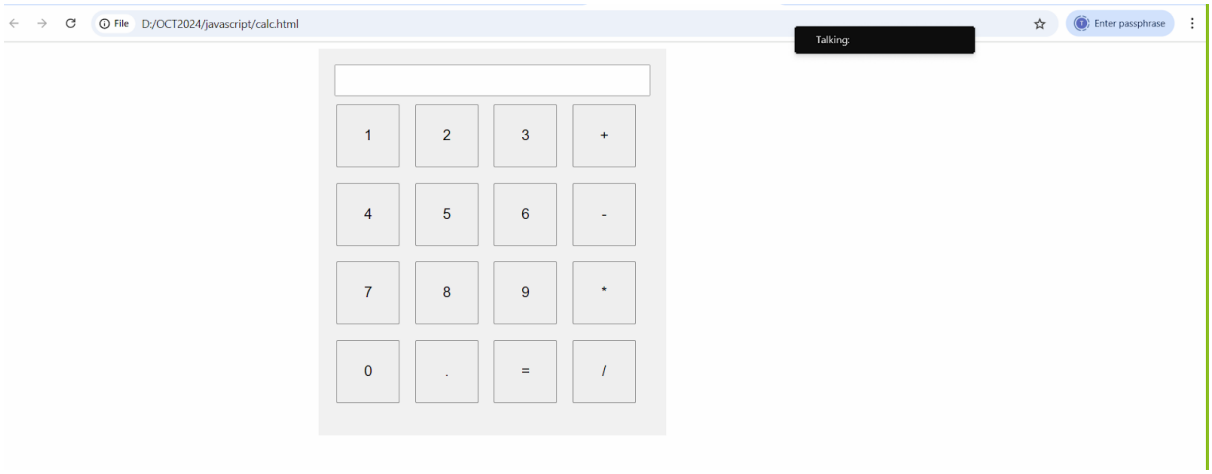
Output

Example: Design web with calculator

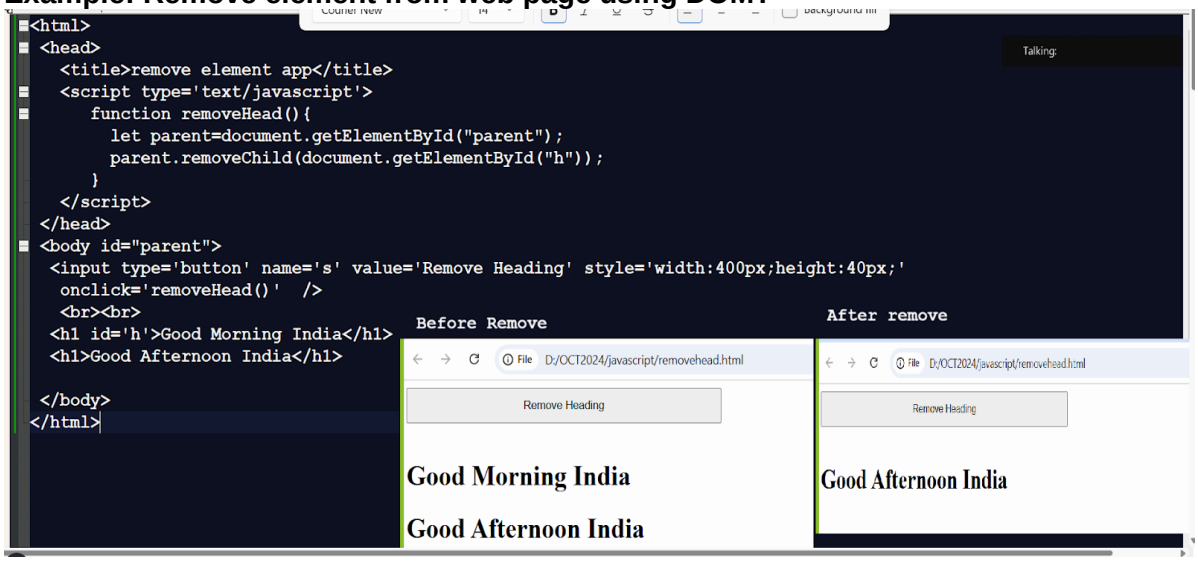


```
<html>
 <head>
   <title>calculator application</title>
<script type='text/javascript'>

        function designCalc(){
            let calcButtons=[
                                        [1,2,3,'+'],
                                         [4,5,6,'-'],
                                         [7,8,9,'*'],
                                         [0,'.','=','/']
                                    ];
                let table=document.createElement("table");
                        table.style.width="400px";
                        table.style.height="400px";
                        let prevVal="";
                        let first,second,result,choice;
                        for(var i=0; i<calcButtons.length; i++)
                        {   let row=document.createElement("tr");
                           for(var j=0; j<calcButtons.length;j++)
                           {     let txtCtrl=document.getElementById("txt");
                                    let firstCol=document.createElement("td");
                                    let firstBtn=document.createElement("input");
                                        firstBtn.setAttribute("type","button");
                                        firstBtn.setAttribute("value",""+calcButtons[i][j]);
                                        firstBtn.style.width="80px";
                                        firstBtn.style.height="80px";
                                        firstBtn.style.fontSize="18px";
                                        firstCol.appendChild(firstBtn);
                                        row.appendChild(firstCol);

                                        firstBtn.addEventListener('click',function(){
                                            let currVal=this.value;
                                            prevVal=txtCtrl.value+currVal;
                                            txtCtrl.value=prevVal;

                                                if(this.value=='+')
                                                { first=parseInt(prevVal.trim());
                                                  txtCtrl.value="";
                                                        choice=1;
                                                }
                                                if(this.value=='*')
                                                { first=parseInt(prevVal.trim());
                                                  txtCtrl.value="";
                                                        choice=2;
                                                }
                                                if(this.value=='-')
                                                { first=parseInt(prevVal.trim());
                                                  txtCtrl.value="";
                                                        choice=3;
                                                }
                                                if(this.value=='/')
                                                { first=parseInt(prevVal.trim());
                                                  txtCtrl.value="";
```

```
                                        choice=4;
                                    }
                                    if(this.value=='=')
                                    {
                                      second=parseInt(prevVal.trim());
                                        switch(choice)
                                        {
                                          case 1:
                                          result=first+second;
                                                txtCtrl.value=""+result;
                                                break;
                                                case 2:
                                          result=first*second;
                                                txtCtrl.value=""+result;
                                                break;
                                                case 3:
                                          result=first-second;
                                                txtCtrl.value=""+result;
                                                break;
                                                case 4:
                                          result=first/second;
                                                txtCtrl.value=""+result;
                                                break;
                                                default:
                                                alert("Wrong operation");
                                        }
                                    }
                                });

                            }
                            table.appendChild(row);
                        }
                let c=document.getElementById("calcContainer");
                c.appendChild(table);
        }
</script>
 </head>
<body onload="designCalc()">
 <div id='calcContainer'  style="width:400px;height:450px;background-color:#f2f2f2;
 margin-left:400px;padding:20px;">
   <input type='text' name='txt' id='txt' value='' style='width:400px;height:40px;'/>
 </div>
</body>
</html>
```



Example: Remove element from web page using DOM?



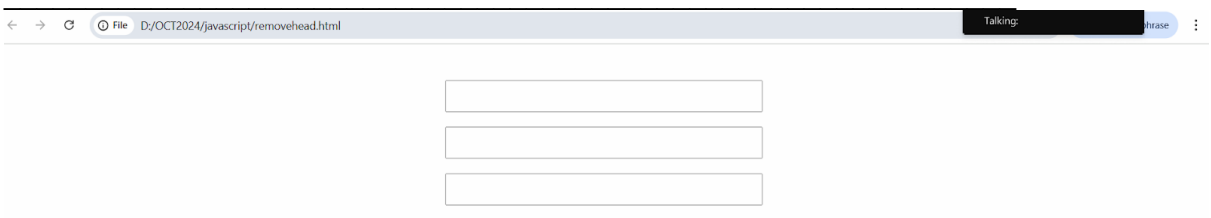
Example: we want to design form zoom html control when we perform mouseover event

---

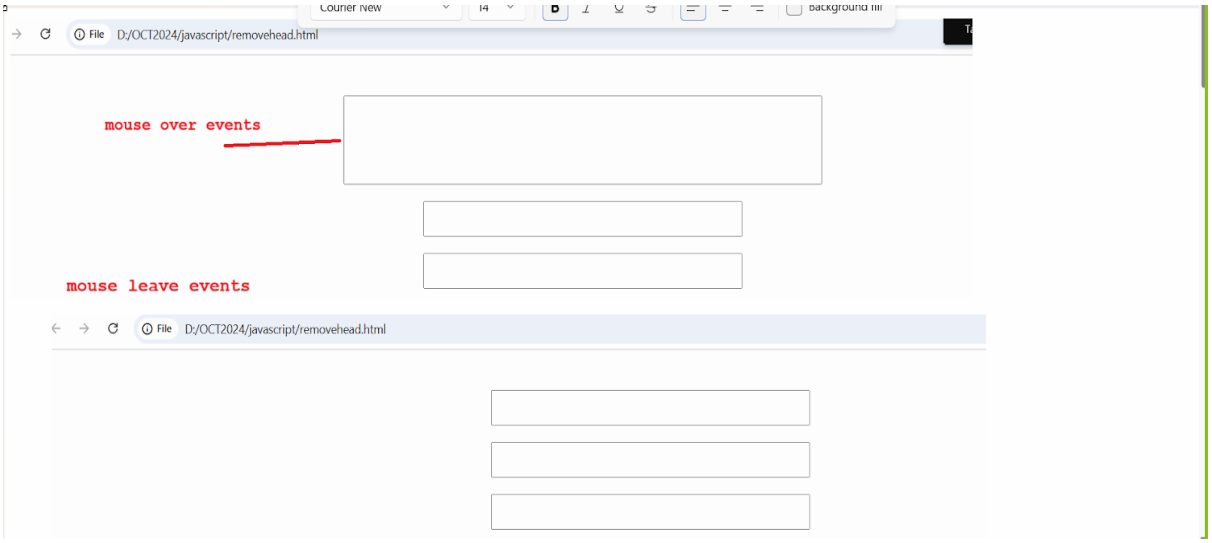
```
<html>
 <head>
  <title>remove element app</title>
  <script type='text/javascript'>
    function zoomCtrl(currEle)
          {
             if(currEle.id=='f')
                  {  currEle.style.width="600px";
                     currEle.style.height="100px";

                  }
                  else if(currEle.id=='s')
                  {   currEle.style.width="600px";
                     currEle.style.height="100px";
                  }
```

```
                else if(currEle.id=='r')
                {   currEle.style.width="600px";
                   currEle.style.height="100px";

                }

            }
             function originalCtrl(currEle)
            {
                if(currEle.id=='f')
                {  currEle.style.width="400px";
                   currEle.style.height="40px";
                }
                else if(currEle.id=='s')
                {   currEle.style.width="400px";
                   currEle.style.height="40px";
                }
                else if(currEle.id=='r')
                {  currEle.style.width="400px";
                   currEle.style.height="40px";
                }

            }
  </script>
  <style>
    input{
            width:400px;
            height:40px;
          }
  </style>
 </head>
 <body id="parent">
  <center>
  <br><br>
  <input type='text' name='txtCtrl' id='f' value='' onmouseover='zoomCtrl(this)'
   onmouseleave='originalCtrl(this)'/> <br><br>
  <input type='text' name='txtCtrl' id='s' value='' onmouseover='zoomCtrl(this)'
    onmouseleave='originalCtrl(this)'/> <br><br>
  <input type='text' name='txtCtrl' id='r' value='' onmouseover='zoomCtrl(this)'
   onmouseleave='originalCtrl(this)'/> <br><br>
  </center>

 </body>
</html>
```

Original form



After event form



Example: WAP to change image when we click on button using DOM

```
<html>
 <head>
   <title>remove element app</title>
   <script>
    function changeImage(){
              let ictrl=document.getElementById("j");
              ictrl.src='second.png';
           }
   </script>
   <style>
     input{
              width:400px;
              height:40px;
              margin-left:100px;
            }
            img{
             margin-left:100px;
             margin-top:50px;
            }

   </style>
 </head>
 <body id="parent">
   <img src='first.png'  id="j" alt="image not found" /> <br><br>
   <input type='submit' name='s' value='Change Image' onclick='changeImage()'/>
 </body>
</html>
```

String Handling in JavaScript

String in javascript is an immutable object which means once we initialize a value we cannot change later called immutable.
Syntax: let variable=”value”;
Methods & property of String in JavaScript

length: length is a method of string which is used to return the size of string.

```
let str="good";

let len=str.length;

console.log("Length of string is "+len);
```

```
D:\OCT2024>node str.js
Length of string is 4
```

Talking:

charAt(index): this method character from a string using its specified index.

```
let str="good";
let ch=str.charAt(1);
console.log(ch);
```

```
D:\OCT2024>node str.js
o
```

Talking: Adinath Giri

0 1 2 3

g o o d

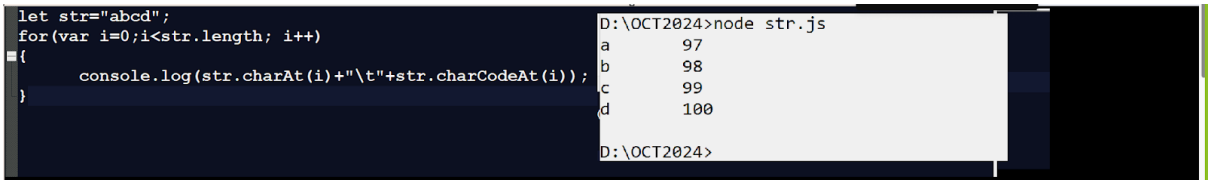
Or

```
let str="good";
for(var i=0;i<str.length; i++)
{
      console.log(str.charAt(i));
}
```

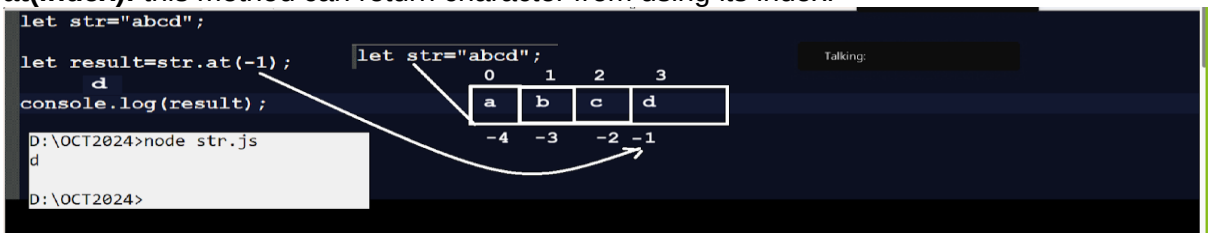
```
D:\OCT2024>node str.js
g
o
o
d
```

Talking: Adinath Giri

charCodeAt(position): this method accepts index of character and returns its ascii code.



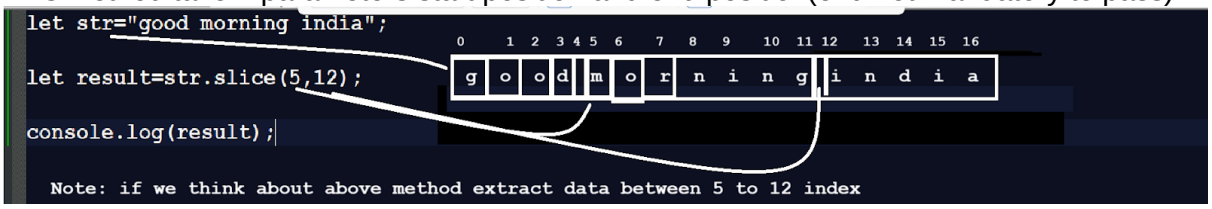
at(index): this method can return character from using its index.



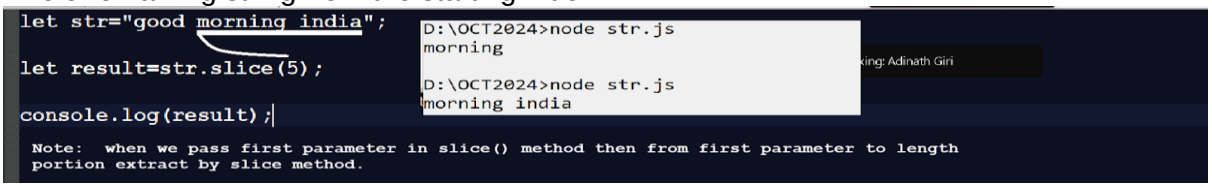
Note: charAt() method not supporting negative index just only works with positive index but if we think about at() method then at() method can work with positive as well as negative indexing also.

slice() method of String

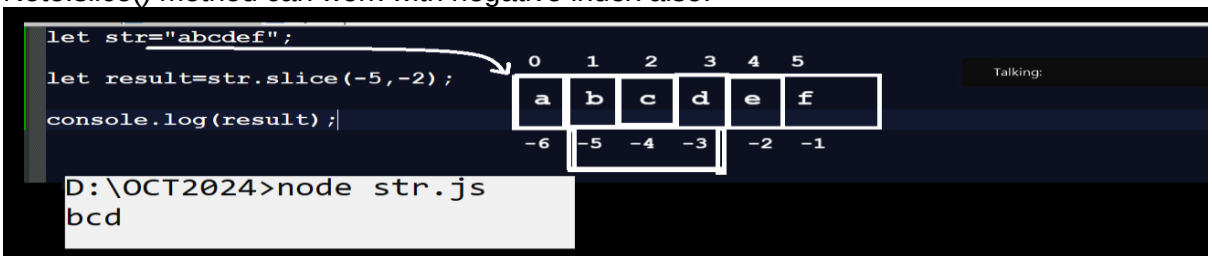
slice() method can extract part of a string and returns the extracted part in a new string. This method take 2 parameters start position and end position(end not mandatory to pass)



Note: when we do not provide a second parameter in the slice method it will extract the whole remaining string from the starting index.



Note:slice() method can work with negative index also.

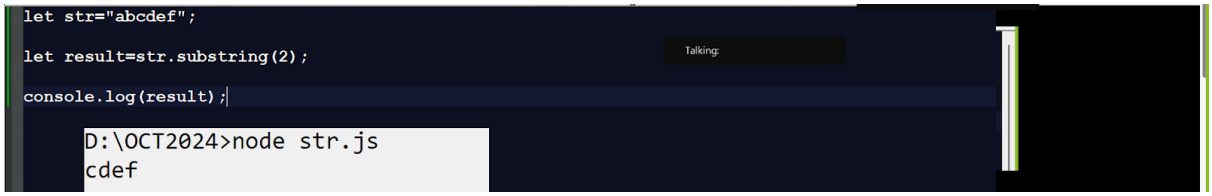


substring():this method can extract some portion of string between two specified index This method accept two parameters and substring not support to negative indexing.

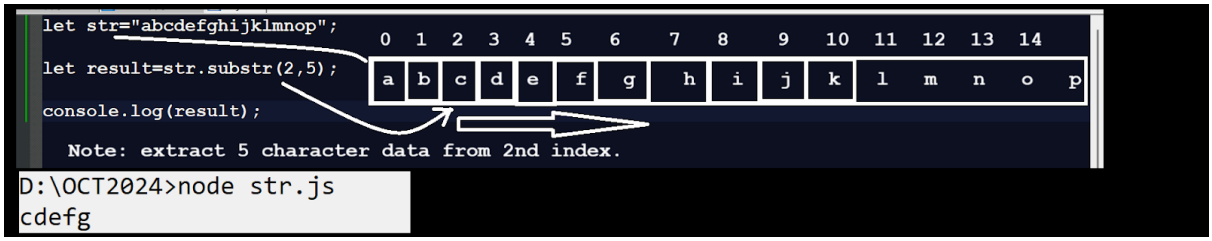
Syntax: substring(startIndex,endIndex):

Note: second parameter is not mandatory to provide and if not provide second parameter then substring extract from index to length

Or



substr(startindex,length): substr accepts two parameters: first parameter as index and second parameter length of character which we want to extract.



Note: when we do not provide a second parameter then extract the complete string from the index.

toUpperCase(): this method can convert lower case string to upper case and store new string

toLowerCase(): this method convert upper case string to lower case string

concat(): this method can combine or merge more than one string and generate new string objects.

```
let s="abc ";
let s1="Mno ";
let s2=" PQR";

let s3=s.concat(s1,s2," xyz", " pqr");

console.log(s3);
```

```
D:\OCT2024\javascript>node concattest.js
abc Mno  PQR xyz pqr

D:\OCT2024\javascript>
```

trim(): remove the beginning and ending white spaces from string.

```
let s ="     abcd     ";

let s1 = s.trim();

console.log(s1);
```

```
D:\OCT2024\javascript>node concattest.js
abcd

D:\OCT2024\javascript>
```

trimStart(): this method can remove white spaces at beginning of string
trimEnd(): this method can remove white spaces at the end of a string.

padStart(): add some data with string at beginning

```
let s="555";

let s1=s.padStart(4,'$');

console.log(s1);
```

total count - length of string
remaining symbol add at beginning

padEnd(): add some data at the end of the string.

```
let s="555";

let s1=s.padEnd(4,'$');

console.log(s1);
```

```
D:\OCT2024\javascript>node concattest.js
555$

D:\OCT2024\javascript>
```

repeat(count): repeat string with number of times according to value pass in it.

```
let a="abc";
let str=a.repeat(4);
console.log(str);
```

```
D:\OCT2024\javascript>node concattest.js
abcabcabcabc

D:\OCT2024\javascript>
```

replace(olderstring,newstring): replace existing string data with new string value.

```
let a="Good Morning India";
let str=a.replace("Morning","Afternoon");
console.log(str);
```

this string value replace with morning or old string

```
D:\OCT2024\javascript>node concattest.js
Good Afternoon India

D:\OCT2024\javascript>
```

indexOf(string): this method can accept string as parameter and return its index value and data not found return -1. This method return first occurrence of element

```
let a="Good Morning India";
let index=a.indexOf("Morning");
if(index!=-1)
{ console.log("Value found ");
}
else
{  console.log("Value not found");
}
```

Output

```
D:\OCT2024\javascript>node concattest.js
Value found

D:\OCT2024\javascript>
```

lastIndexOf(): this method can return the last occurrence index of an element.

```
let a="Good Morning India good Morning Pune good Morning Warje";
let index=a.lastIndexOf("Morning");
if(index!=-1)
{ console.log("Value found "+index);
}
else
{  console.log("Value not found");
}
```

```
D:\OCT2024\javascript>node concattest.js
Value found 42
```

search(): this method can search data from string and return its index. The major difference between search and indexOf() is the search method can work with regular expressions as well as string but indexOf() only works with string and if data not found return -1

```
let a="Good Morning India good Morning Pune good Morning Warje";
let index=a.search("Morning");
if(index!=-1)
{ console.log("Value found "+index);
}
else
{  console.log("Value not found");
}
```

Client Side Validation using string handling

What is validation or client side validation?

When we perform validation at client or user side for verify or provide appropriate data from client page to server page as request

What is the importance of client side validation on a website?

---

a. Provide proper and clean data as request to server

b. Avoid some attacks on web page

Examples of attach

SQL Injection : attacker injects malicious SQL code to manipulate website database.

Cross Site Scripting Attack : normally attacker can inject some malicious code or script into websites to redirect user on malicious website

Cross Site Requesting forgery(CSRF):

There are two types of validation

a. Client side validation (normally we can perform at client like as using javascript)

b. Server side validation (validation perform using backend language)

Now we want to discuss about client side validation using JavaScript

---

1. Name validation
2. Phone number validation
3. Email validation
4. Date of birth validation
5. Aadhar card validation
6. PAN CARD Validation
7. Voter ID validation
8. Light bill validation

   Etc

Name Validation

---

1. **Special symbol not allow**
2. **Digit not allowed**

Example with source code

```
<html>
<head>
  <title>name validation</title>
  <script type='text/javascript'>
          function validateName(str)
          {  str=str.toLowerCase();
             let flag=true;
             let span=document.getElementById("s");
             for(var i=0;i<str.length; i++)
             {
                if(!(str.charCodeAt(i)>=97 && str.charCodeAt(i)<=122))
                      {  flag=false; //when we found any char other than alphabet
                      }
             }
             if(flag)
             { span.innerHTML="";
                      span.style.color="white";
                      span.style.padding="20px";
             }
             else{
               span.innerHTML="Invalid name";
                      span.style.color="red";
                      span.style.padding="20px";
             }
          }
  </script>
</head>
<body>
 <input type='text' name='name' value='' style='width:400px;height:40px;'
  onkeyup='validateName(this.value)' />
   <span id="s"></span>
</body>
</html>
```

Email validation

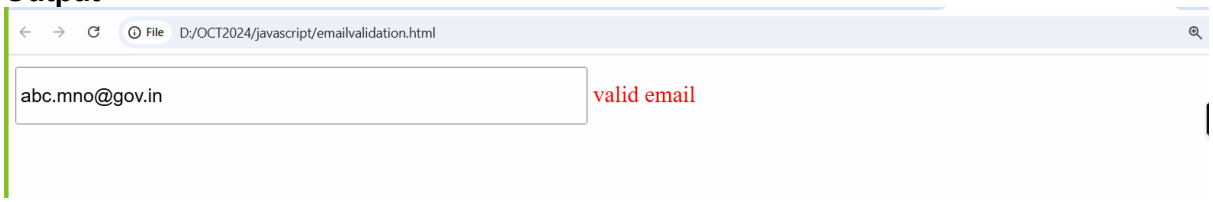
1. @ symbol must be present in email
2. @ symbol should not be first letter
3. @ symbol should not be occur more than once in email
4. dot (.) can occur more than one time in email but minimum one dot should be present after @ symbol
5. Capital letter not allowed
6. Special symbol not allowed in email except dot,underscore and @ symbol

Source code invalidation

```
<html>
 <head>
```

```
  <title>email validation</title>
  <script type='text/javascript'>
    function validateEmail(str){
            let index=str.indexOf("@");
                  let index1=str.lastIndexOf("@");
                  let span=document.getElementById("s");
                   if(index<=0 || index!=index1)
                   {   span.innerHTML="Invalid email";
                             span.style.color="red";
                   }
                   else
                   {   let as=str.substring((index+1));
          index=as.indexOf(".");
          index1=as.lastIndexOf(".");
           if((index==-1 || index!=index1) || (index!=(as.length-4) && index!=(as.length-3) ))
                              { span.innerHTML="Invalid email";
                                span.style.color="red";
                              }
                              else{
                               span.innerHTML="valid email ";
                                span.style.color="color";
                             }
                   }

           }
  </script>
 </head>
 <body>
<input type='text' name='email' id='e' value='' placeholder='Enter email'
 style='width:400px;height:40px;' onkeyup="validateEmail(this.value)"/>
 <span id='s'></span><br><br>
 </body>
</html>
```

Output

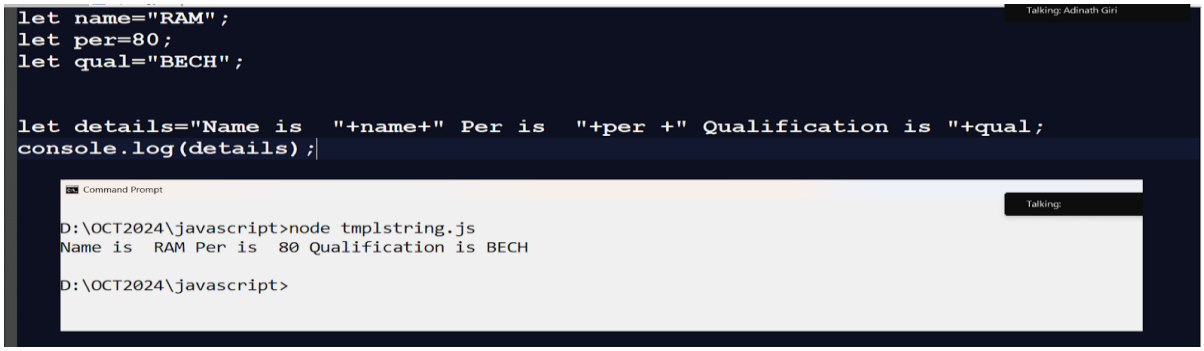


Advanced JavaScript

Topic List

Arrow function , rest operator , template string , spread operator, promises, async await , Fetch api,web services, rest api , json , ajax, event loop , memory structure JS ,BOM, Module export and import ,OOP, object literals , call back function

Template string : template string is used for avoid string concation for adding run time data in string means suppose we have some fix string value and we have some variables which we want to add in string so we required concat string so when we have n number of cocatination so string handling is very difficult so avoid string concat operation we have one option name as template string

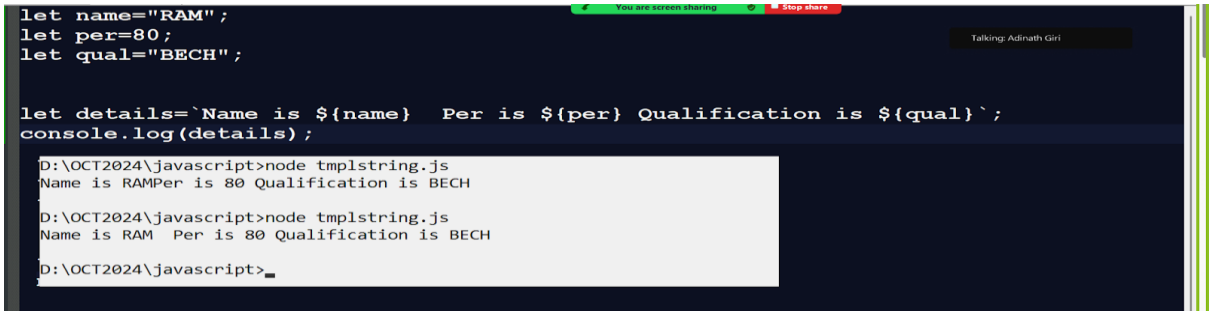


If we think about the above string we have a string with variable and for that we have to use “”+ for concat in string or combine in string.
So if we want to add run time data in string without “”+ so we can template string concept

How to declare template string with string or use template string with string

---

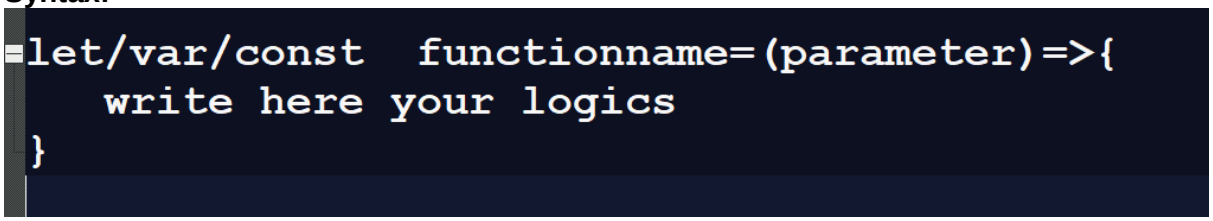
If we want to declare template string we have to use `` backtick operator



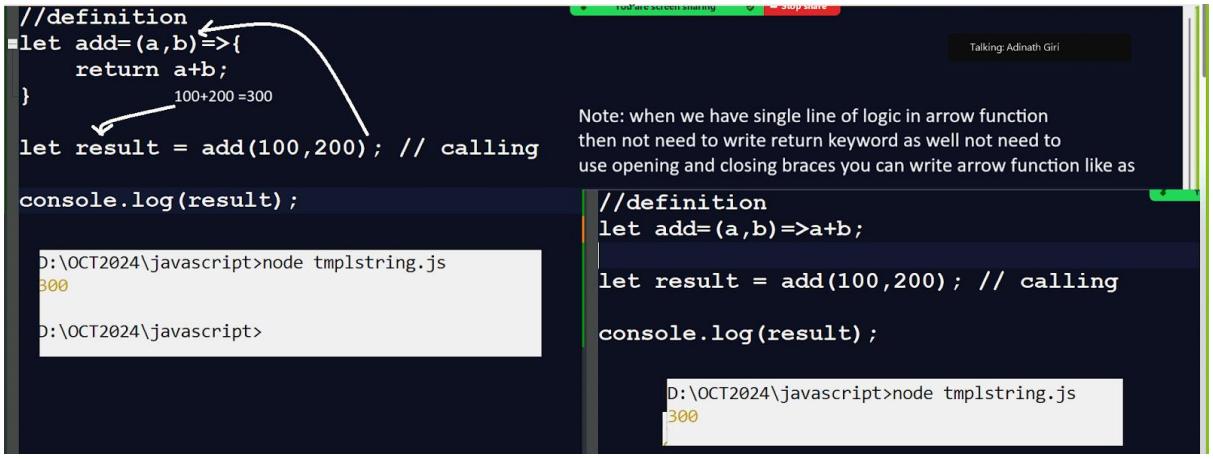
Arrow function : Arrow function is technique to write function in JavaScript basically it is used for writing short logics in function and it is very easy to write anonymous function in JavaScript

How to write arrow function in JavaScript

Syntax:



Example: we want to create function name as add which accept two values and calculate its addition

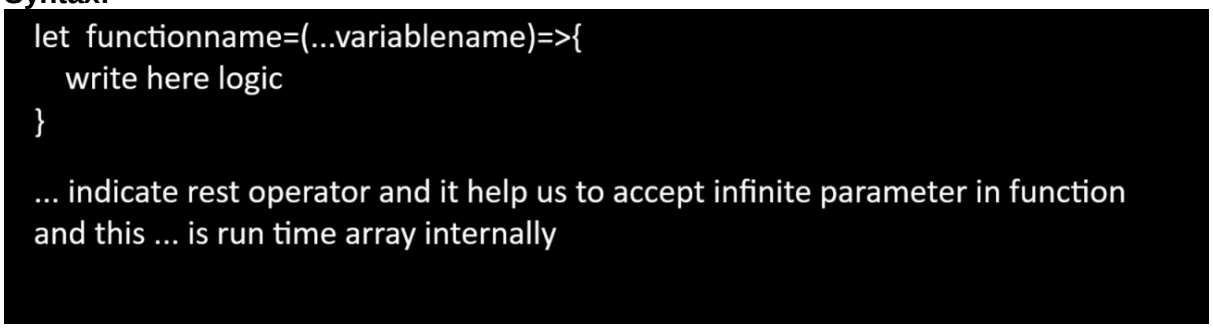


Example: WAP to pass 10 to function and display its table.

```
let table=(no)=>{
        for(var i=1; i<=10; i++)
        { console.log(i*no);
        }
}
table(10);
```

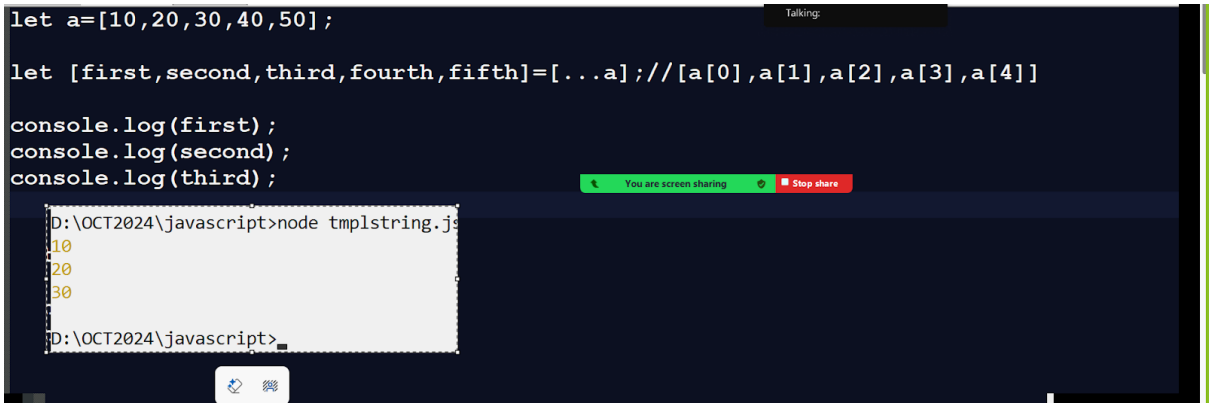
Rest operator : rest operator is used for pass infinite parameter in function from function calling point

Syntax:



Example: WAP to create function name as sum() and pass infinite parameter in function and calculate sum.

Spread operator: spread operator is used for destructing the array.
Array destructing means we initialize array values in variables by using index calls as array destructors but spread operators provide short cut ways to initialize array values in normal variables.



JSON

Note: before understanding we need know the concept

a. Web services
b. API

Q.What are web services?

Web services is a computer system or architecture where we can share the data between two applications or systems in network those are build using a same tech stack or may be different tech stack as well as it help us to develop the client and server base application using same technology or may be different technology etc
If we think about web service your client application may be a website, or mobile or electronic device.

How web services share data between two applications

Webservices can share data between two application or more than application by using API

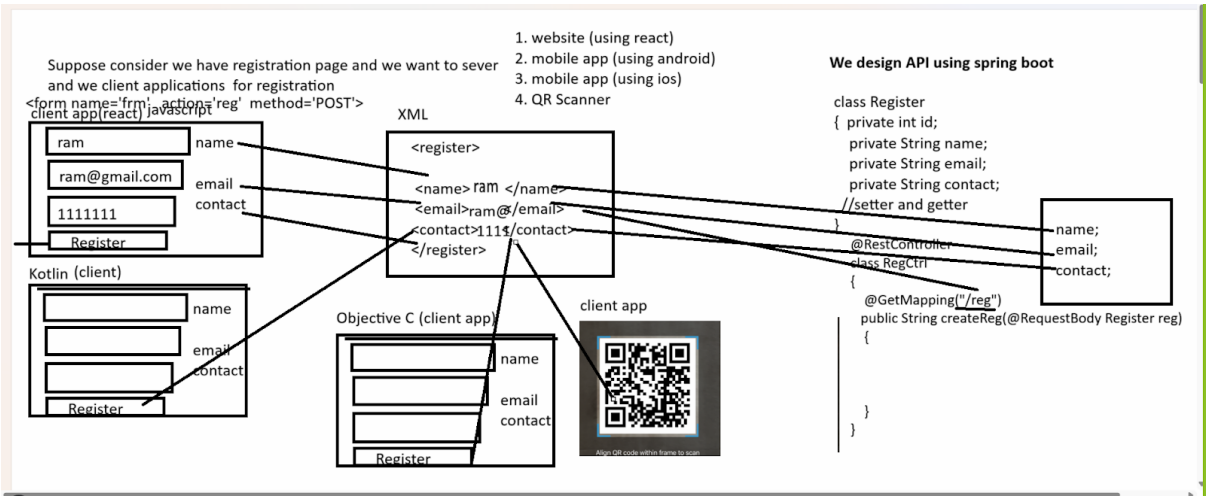
Q. What is an API?

API stands for Application programming interface means we can say API are the end points or URL which is used for accept the request from some other software or application as well as can accept request from same software also and provide response according to request send by application

Example of API or End POINT : or URL AT SERVER SIDE http://localhost:3000/register

How we can share data between API

If we want to share between API we have two ways
a. Using XML format : XML is a custom development language and it is platform independent language which is used for sharing data between two different technologies and it helps us to map data between two different technologies.



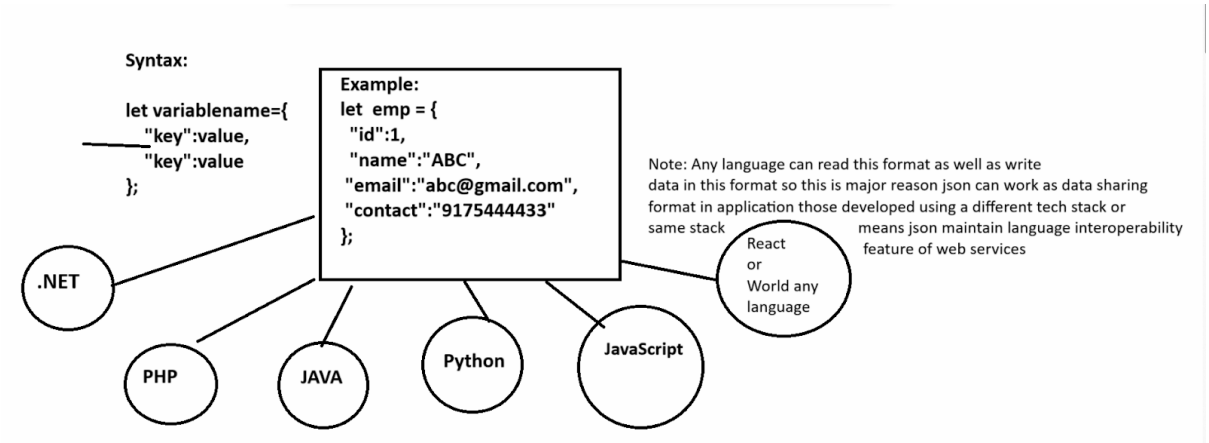
Here we think we have one server based application which is able communicate with different client application those are develop under the different technical stack and we use common data sharing format in all application i.e XML using XML format we can share data between different technical stack
Means we can XML is way to share data between two application or software developed using same tech stack or different tech stack via API i.e url so it is part of web services

Note: XML required heavy band width for message sharing as well as it require more configuration so we have one more format for data sharing known as JSON

b. Using JSON format

Q. What is JSON?

JSON stands for JavaScript object notation and it is lightweight format which is used for share data between two different application and json format is platform independent as well as language independent means json format can use by using any programming language
So json is used for sharing data between two applications developed by using the same technical stack or different technical stack. This is the major reason JSON is used by web services and the JSON data sharing format known as REST API or RESTFUL Web services .
JSON is basically an object which can store data in the form of key and value pairs. Json key cannot duplicate and represent in string format and value may be any type And may be duplicated



Example: we want to create an employee object using json format, store data in it and display its data.

Note: we can store array as value for single key in json

```
let stud={
        "id":1,
        "name":"ram",
        "email":"ram@gmail.com",
        "contact":"1234567891",
        "marks":[60,70,60,70,60,60]
};
console.log("Id is  "+stud.id);
console.log("Name is  "+stud.name);
console.log("Email is  "+stud.email);
console.log("Contact is  "+stud.contact);

let m=stud.marks;
let s=0;
m.forEach((val)=>{
        s=s+val;
});
console.log("Percentage is  "+(s/m.length));
```

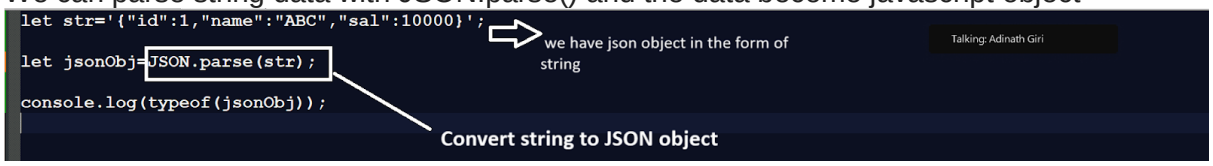
```
D:\OCT2024\javascript>node testjson.js
Id is  1
Name is  ram
Email is  ram@gmail.com
Contact is  1234567891
Percentage is  63.333333333333336
```

Q. What are the benefits of JSON or why use JSON

1. Lightweight format means json store data in textual format so it is very light weight
2. Language independent and platform independent format
3. Store data in the form of key and value pair so data fetching or retrieval is very easy using any technology
4. Json support to different types of data or array as value or nested object also

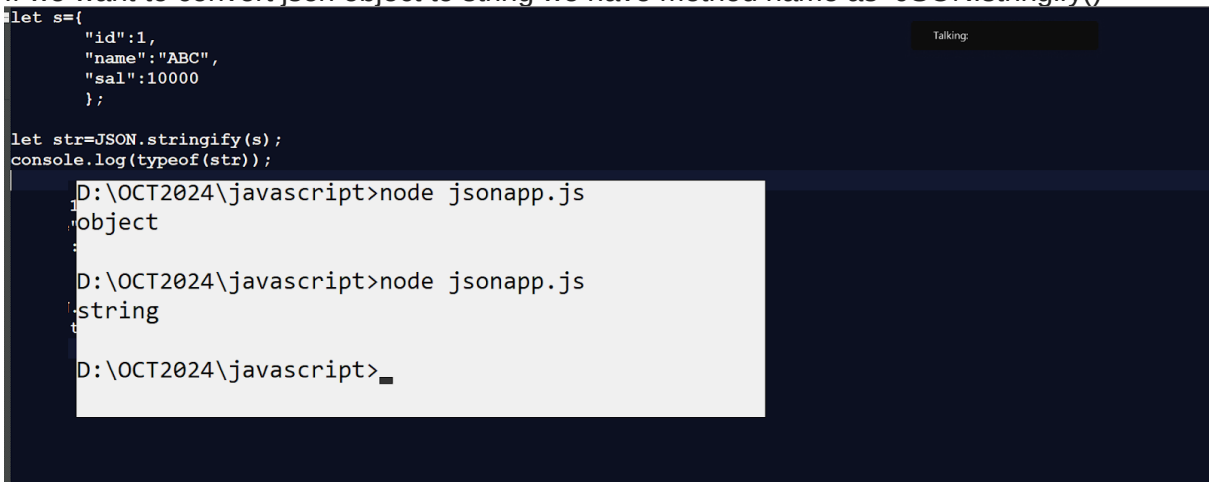
JSON server or exchanging data between JSON server & client as request and response

Common use of JSON is to exchange data to/from a web server .
When server send data to client or when receive data from a server the default is always in the form of string
We can parse string data with JSON.parse() and the data become javascript object



How to convert json object to string

If we want to convert json object to string we have method name as JSON.stringify()



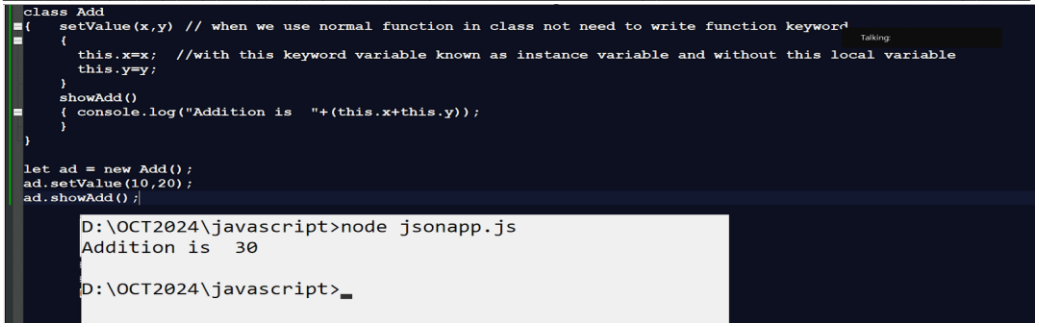
Types of Web services?

SOAP : SOAP stands for Simple object access protocol called SOAP. These protocols are based on XML which is a data exchange language and these protocols are language independent protocols and we can run on any platform.

SOAP support to stateful and stateless operations. Stateful means server track the information of client page or user
Stateless means server not store or not track information about client app or user page'
Many companies like as IBM, microsoft are producing implementation of web services
REST : REST stands for representational state transfer. Basically it is not protocol architecture style and this architectural style uses the http protocol for sending request and getting response and REST API can share data as request using any format and getting response from a server sign any format.
If we think about REST API we can send request using JSON format or getting response using JSON format
As well as we can send request using XML or response in XML or we can send request using string format
Etc if we think about REST
Note: standard way is JSON used in REST API for sending request and getting response

OOP concept using JavaScript

How to class and object in JavaScript

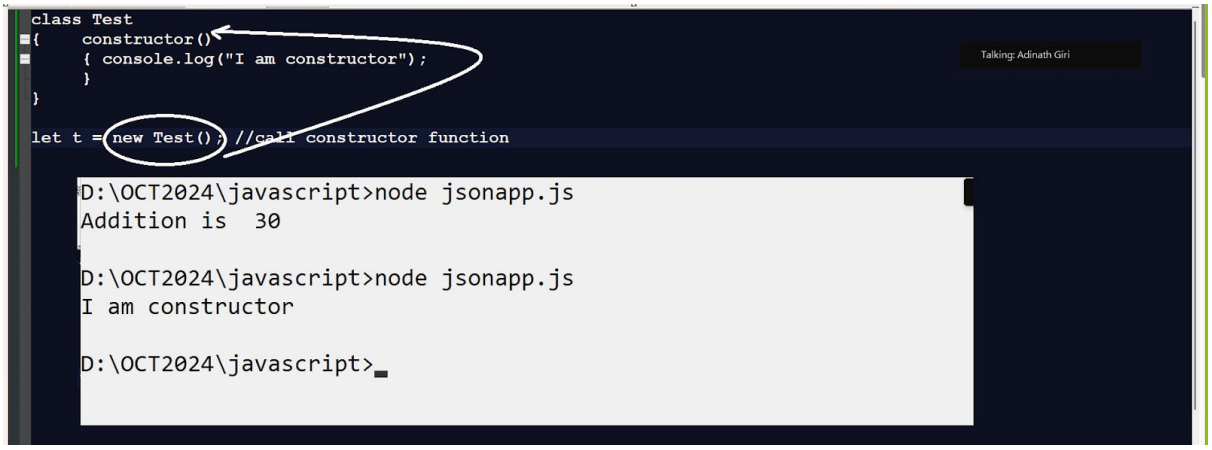


How to use arrow in function in class

If we want to define arrow function in class then we not required to write let or var or const keyword with function definition

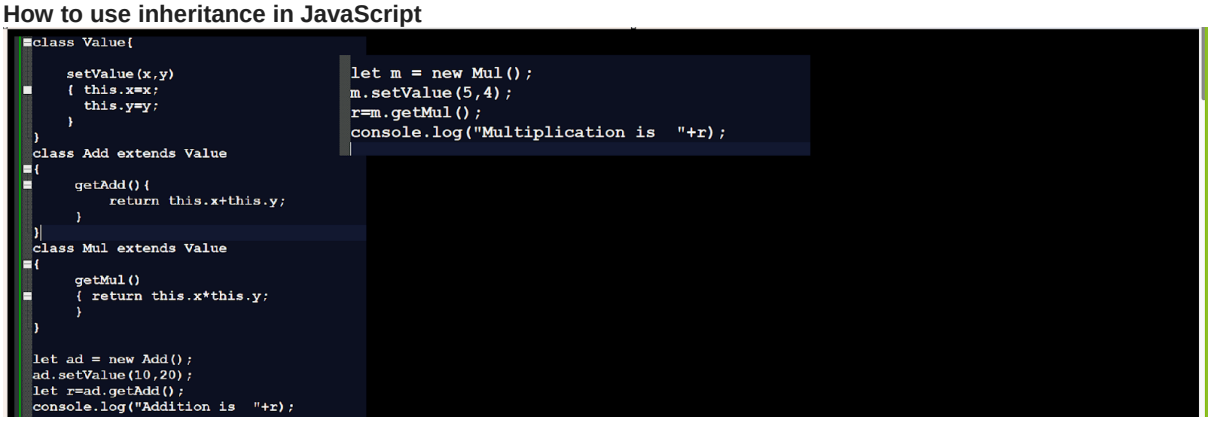
How to use constructor in JavaScript

If we want to use a constructor in JavaScript we have a constructor function in JavaScript which executes when we create an object of class.



How to pass parameter to constructor

How to use inheritance in JavaScript



How to use super constructor

---

super() constructor help us call parent class constructor

```
class Value{
    constructor(){
        console.log("i Value parent constructor");
    }
}
class Add extends Value
{
    constructor()
    { super(); //call parent constructor
    console.log("i am Add child constructor");
    }
}
let ad = new Add();
```

```
D:\OCT2024\javascript>node jsonapp.js
i Value parent constructor
i am Add child constructor

D:\OCT2024\javascript>
```

Note: if parent contain parameter so we can pass parameter from child class constructor using super()

```
class Value{
    constructor(x){
        console.log("i Value parent constructor "+(x*x));
    }
}
class Add extends Value
{
    constructor()
    { super(10); //call parent constructor
    console.log("i am Add child constructor");
    }
}
let ad = new Add();
```

```
D:\OCT2024\javascript>node jsonapp.js
i Value parent constructor 100
i am Add child constructor

D:\OCT2024\javascript>
```

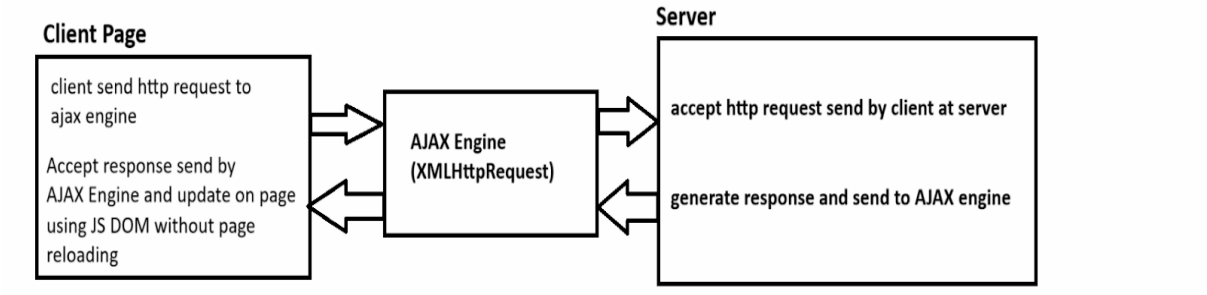
How to perform overriding

```
class Value{
    show(){
      console.log("I am parent show");
    }
}
class Add extends Value
{
     show(){
      console.log("I am child show");
    }
}
let ad = new Add();
ad.show(); //call overriden method
```

```
D:\OCT2024\javascript>node jsonapp.js
I am child show

D:\OCT2024\javascript>
```

AJAX

AJAX (Asynchronous JavaScript and XML) it is not a language just it is technology develop by google In 2005.
The Major goal of AJAX is to perform partial web page updation when user send request or get response means ajax engine send request to server in background asynchronously and get response from a server in background and update on web page.

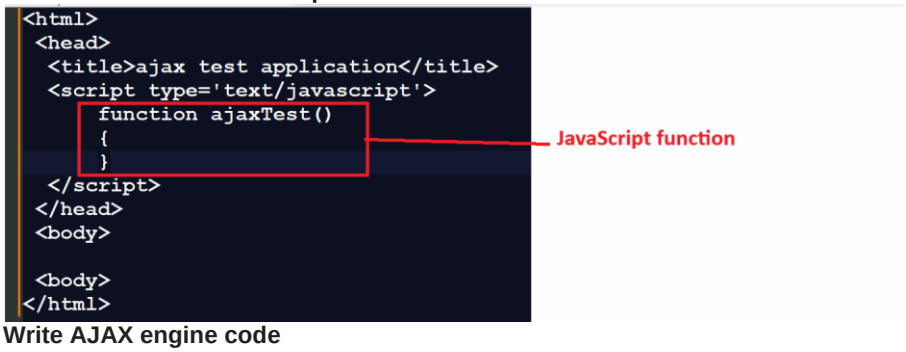


Note: if we want to execute code of AJAX need server environment

How to use AJAX Practically

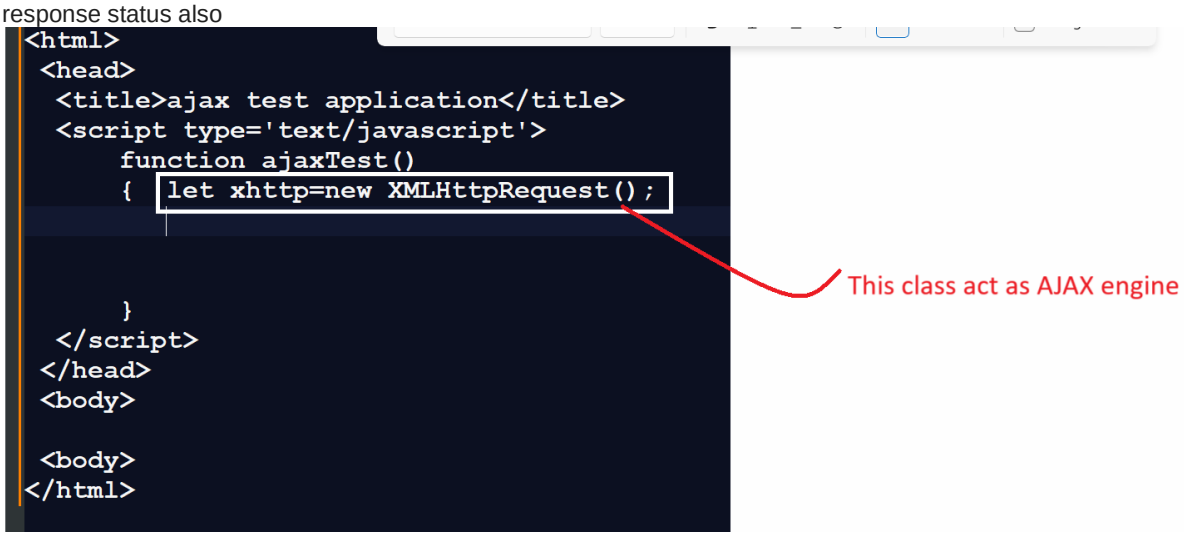
If we want to use AJAX practically we have some important steps.

1. **Define function in JavaScript**



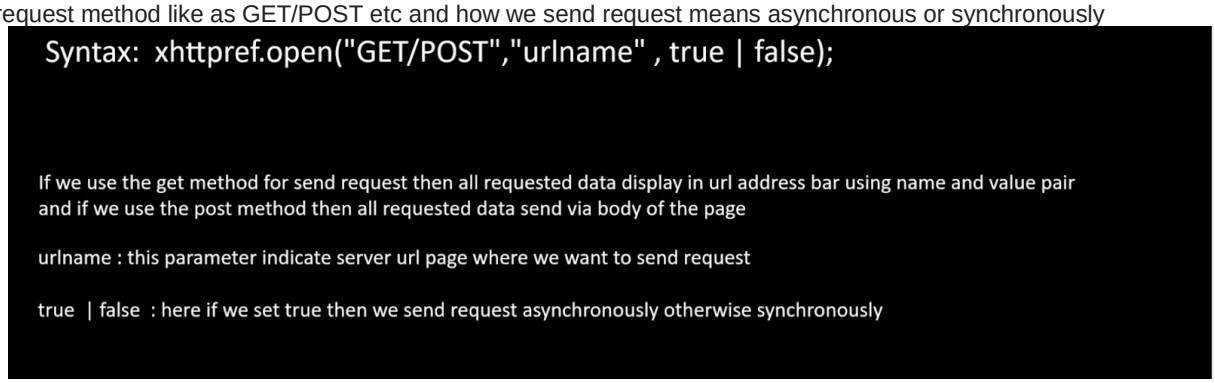
2. **Write AJAX engine code**

If we want to create ajax Engine practically we have one inbuilt class name as XMLHttpRequest
In JavaScript this class act as AJAX engine means in this class contain some methods which is responsible send request to server url or page and get response from a server and check request and response status also



3. **Send request to server using AJAX : if we want to send request to server using a ajax we have two methods**

a. **Open :** this method is used for open request means here we need to specify the requested server url as well as request method like as GET/POST etc and how we send request means asynchronous or synchronously




```
<html>
 <head>
  <title>ajax test application</title>
  <script type='text/javascript'>
      function ajaxTest()
      {  let xhttp=new XMLHttpRequest();


         xhttp.open("GET","emp.csv",true);


      }
  </script>
 </head>
 <body>

 <body>
</html>
```

b. Send : if we want to send request to server we have send() method of XMLHttpRequest xhttpref.send();

```
function ajaxTest()
{  let xhttp=new XMLHttpRequest();


   xhttp.open("GET","emp.csv",true);
    xhttp.send();

}
```

4. GET Response from a server using AJAX engine

If we want to get response from a server using ajax engine we have to check request status as well as response status
Here response status means we have some standard code which denoted or indicate what kind of response send by server to client and every response status code has some standard meaning provided by W3C

Example of response code

Informational responses

`100 Continue`

This interim response indicates that the client should continue the request or ignore the response if the request is already finished.

`101 Switching Protocols`

This code is sent in response to an `Upgrade` request header from the client and indicates the protocol the server is switching to.

`102 Processing`

This code was used in WebDAV contexts to indicate that a request has been received by the server, but no status was available at the time of the response.

`103 Early Hints`

This status code is primarily intended to be used with the `Link` header, letting the user agent start preloading resources while the server prepares a response or preconnect to an origin from which the page will need resources.

Successful responses

`200 OK`

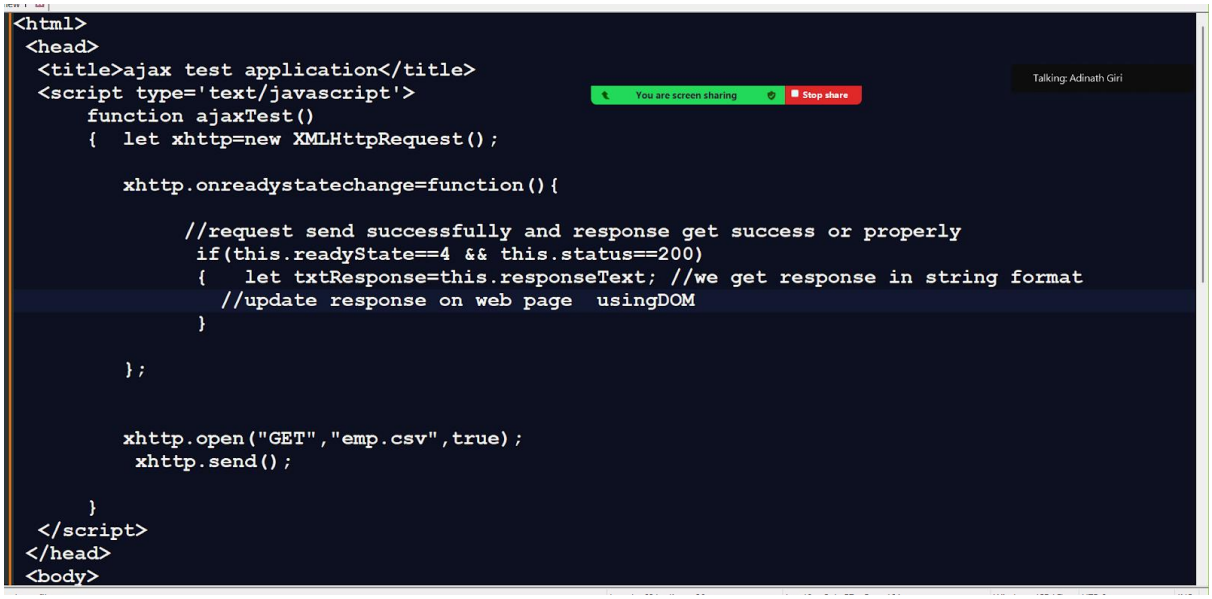
The request succeeded. The result and meaning of "success" depends on the HTTP method:

- `GET`: The resource has been fetched and transmitted in the message body.
- `HEAD`: Representation headers are included in the response without any message body.
- `PUT` or `POST`: The resource describing the result of the action is transmitted in the message body.
- `TRACE`: The message body contains the request as received by the server.

`201 Created`

The request succeeded, and a new resource was created as a result. This is typically the response sent after `POST` requests, or some `PUT` requests.

Etc
If we want to accept the send by server in AJAX we have to define one method name as onreadystate change




```
<html>
 <head>
  <title>ajax test application</title>
  <script type='text/javascript'>
      function ajaxTest()
      {  let xhttp=new XMLHttpRequest();

         xhttp.onreadystatechange=function(){

              //request send successfully and response get success or properly
               if(this.readyState==4 && this.status==200)
               {   let txtResponse=this.responseText; //we get response in string format
                 //update response on web page  usingDOM
               }

         };


         xhttp.open("GET","emp.csv",true);
          xhttp.send();

      }
  </script>
 </head>
 <body>
```

5. Update response send by server on web page using DOM

Example with source code
Test.html

```
<html>
 <head>
  <title>ajax test application</title>
  <script type='text/javascript'>
     function ajaxTest()
            {  let xhttp=new XMLHttpRequest();
                     xhttp.onreadystatechange=function(){
                        if(this.readyState==4 && this.status==200)
                              {   let txtResponse=this.responseText;
                                 console.log(typeof(txtResponse));
                                       let jsonObj=JSON.parse(txtResponse);
                                       let div=document.getElementById("container");
                                       let table=document.createElement("table");
                                       let tr=document.createElement("tr");
                                       let th=document.createElement("th");
                                       th.innerHTML="Employee ID";
                                       tr.appendChild(th);
                                       th=document.createElement("th");
                                       th.innerHTML="Employee Name";
                                       tr.appendChild(th);
                                       th=document.createElement("th");
                                       th.innerHTML="Employee Salary";
                                       tr.appendChild(th);
                                       table.appendChild(tr);

                                       jsonObj.forEach((item)=>{
                                         tr=document.createElement("tr");
                                                td=document.createElement("td");
                                                td.innerHTML=""+item.id;
                                                tr.appendChild(td);
                                                 td=document.createElement("td");
                                                td.innerHTML=""+item.name;
                                                tr.appendChild(td);
                                                 td=document.createElement("td");
                                                td.innerHTML=""+item.sal;
                                                tr.appendChild(td);
                                                table.appendChild(tr);
                                       })
                                       div.appendChild(table);
```

```
                        }
                    };
                    xhttp.open("GET","/viewemp",true);
                xhttp.send();

            }
  </script>
 </head>
 <body>
  <input type="button"
  name="s" value="Update Page"
  style='width:400px;height:40px;padding:20px; margin-left:100px'
  onclick="ajaxTest()" />
  <div id="container">
  </div>
 <body>
</html>
```

index.js

```
//show the employee details
app.get('/viewemp',(req,res)=>{
    let emp=[{
            "id":1,
            "name":"ABC",
            "sal":10000
                },
                {
             "id":2,
            "name":"MNO",
            "sal":20000

        },
        {
                        "id":3,
            "name":"PQR",
            "sal":30000
        }
         ,
        {  "id":4,
            "name":"XYZ",
            "sal":50000

        }]
                            res.json(emp);

});

app.listen(3000,()=>{
  console.log("Server started on 3000 port");
});
```

Promises

Promises are objects that represent the eventual completion or failure of an asynchronous operation and resulting value.

If we want to work with promises we should have to two things

1. **Synchronous operation :** synchronous operation means execute the code line by line means first operation successfully completed after that start second operation execute called as synchronous operation. Means java script is by default synchronous language or single threaded language.

```
let a=100;
let b=200;
let c=a+b;
console.log("Addition is  "+c);
```

```
Command Prompt

D:\OCT2024\javascript>node testsync.js
Addition is  300

D:\OCT2024\javascript>
```

2. Asynchronous operation : Asynchronous operation means if the first operation requires some time for completion then within that time period the program executes some other logic or code called asynchronous.

```
let a=100;
let b=200;
let c=a+b;
setTimeout(function(){
    console.log("Hello i am set time out");
},5000);
console.log("Addition is  "+c);
```

Note: if we think about left hand side code we have asynchronous operation means first execute a=100 , after that execute b=200 and we have setTimeout() which required 5 sec of time for execution within that time period javascript engine execute the statement Addition is 300 and after that execute Hello I am set time out function
means here we have waiting period for 5 second in that period javascript engine execute the another code called as asynchronous operation

```
D:\OCT2024\javascript>node testsync.js
Addition is  300
Hello i am set time out

D:\OCT2024\javascript>
```

Q. If JavaScript is by default a synchronous language then how does JavaScript manage the asynchronous operation?

So JavaScript use the event loop concept for manage the asynchronous operation

Q. What is an event loop?

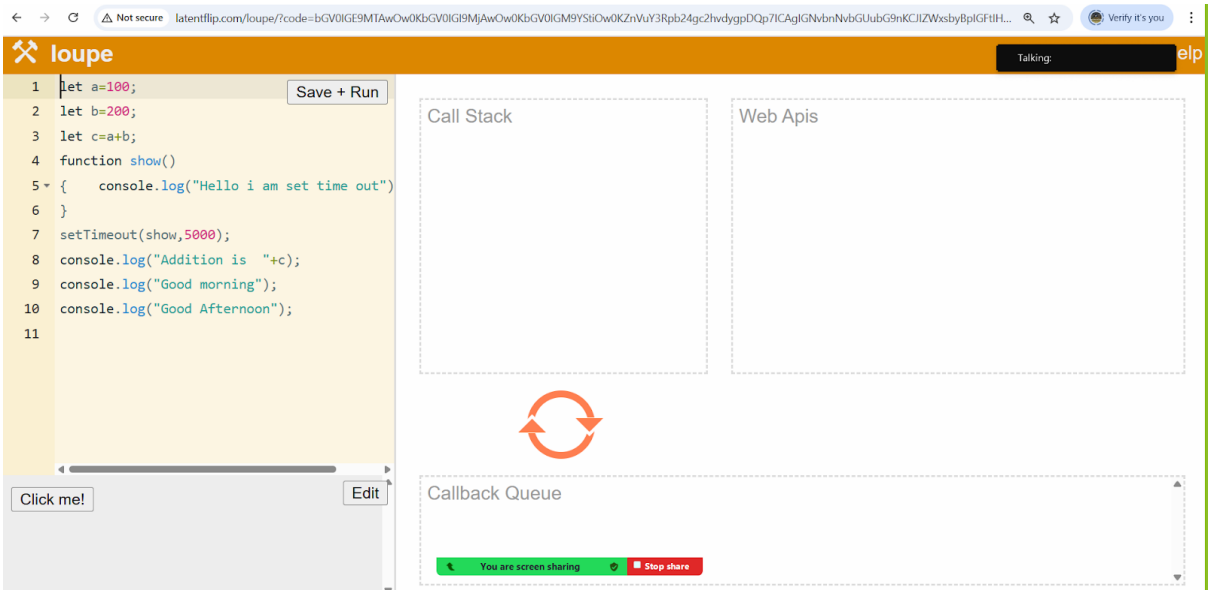
Event loop is a concept in JavaScript that enable asynchronous feature of programming by handling multiple task simultaneously in single threaded environment

How event loop works

If we want to work with event loop we required to know the following things

1. **Call stack :** Call stack where function executes is managed in last in first out order
2. **Web API'S or background task :** these includes setTimeout or setInterval () ,fetch as well as DOM events executed by browser in background

3. **Callback Queue:** when asynchronous operation is completed its callback is pushed into the queue task
4. **Microtask queue:** Promises and other micro task go into the microtask queue which is processed before the queue
5. **Event loop:** continuously check the stack if empty move task from queue to the stack for execution



Note : test your event loop execution using above website

Now we want to discuss about promises

Q. What are promises in JavaScript?

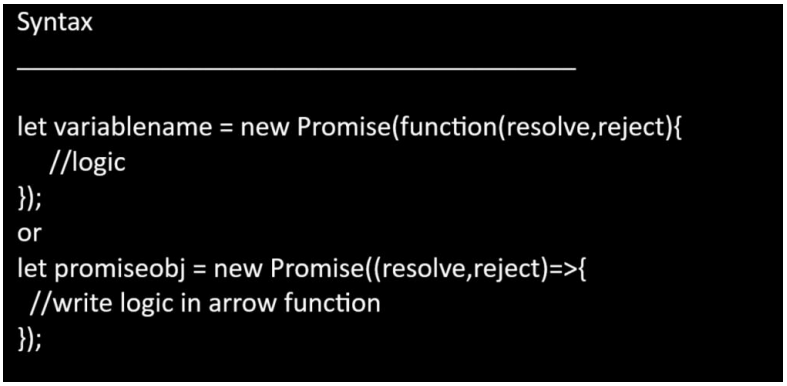
Promises are the object in JavaScript which is used for execute the some logics and the result of that logic may be success, fail or pending

Promise object has three state

1. **Resolve :** Resolve means our logic executed successfully and we going send success result from promise object or we returning success result from promise object
2. **Reject :** reject means our logic not executed successfully and we going send promise fail or failure logics called as reject
3. **Pending :** pending means we going to finalize the promise state means we not giving resolve or reject result called as pending

How to use promises practically in JavaScript

If we want to use promises practically in JavaScript we have Promise class and we required to create object and Promise class has one constructor which contain one callback function and call back function contain two parameter function one is resolve and reject when our promise logic is success we call resolve function and when our promise logic failed we call reject function



Promise object has two function which help us catch result generated by promise object
Promise objects either generate success i.e resolve result or failure i.e reject result.

then((result)=>{
}) : this function call when promise object executed success logic or call resolve function from promise constructor

catch((result)=>{
}); : this function call when a promise object is executed , rejects logic or error logic.

Example: we want to write JavaScript program compare two string with each other if strings are equal then execute resolve function from promise and if string are not execute then execute reject function from promise object
And generate result success when strings are equal and fail when strings are not equal
Example with source

```
let promise= new Promise((resolve,reject)=>{
    let str1="good";
    let str2="bad";
    if(str1==str2)
          {  resolve("Success");
          }
    else{
                  reject("Fail");
          }
});

 promise.then((result)=>{
            console.log(result);
 });
 promise.catch((result)=>{
          console.log(result);
 });
```

Example of promises

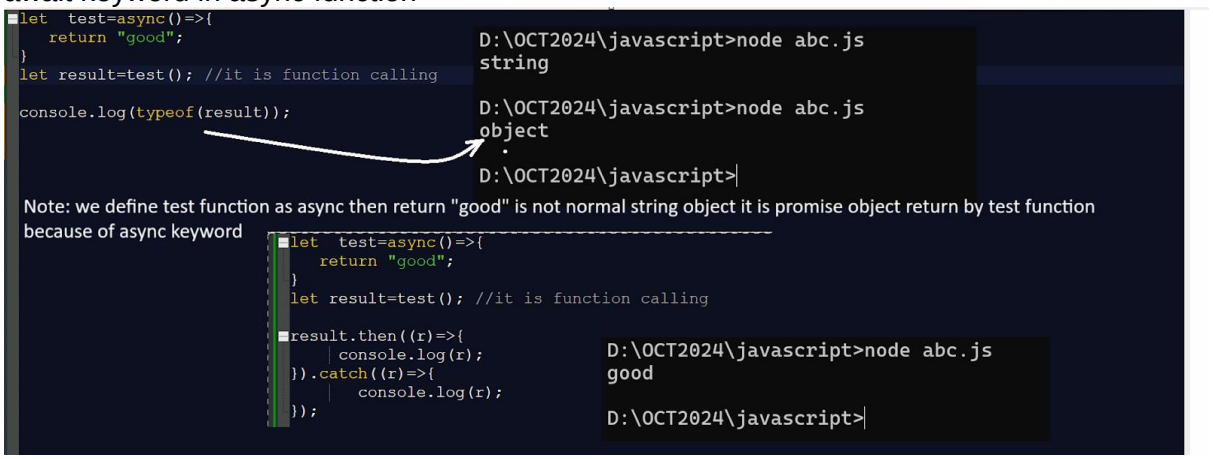
We want to create promise object which is used for check number is even or odd if number is even we want to promise success message and if number is odd then promise fail result

```
let promise=new Promise((resolve,reject)=>{
   let no =11;
         if(no%2==0)
```

```
            { resolve("Success");
            }
            else
            { reject("Failed");
            }
});
promise.then((result)=>{
   console.log("EVEN "+result);
}).catch((result)=>{
    console.log("ODD  "+result);
});
```

Async and Await

Async and await is a feature of JavaScript. You can put your code on hold first, we can declare async to function and holding many pieces of code in it we can wait for return value Means we can use async keyword with function and function always return promise object and using promise object can get result from async function execution means you can use await keyword in async function



What is await keyword

Await is used to wait for the promise. It could be used within an async block only and the goal of await keyword is make code wait until the promise returns the result.

Syntax: let variable = await promise;

Example: we want to create one function which return temperature of multiple cities

```
let tempCalc=async()=>{
   let pune=new Promise((resolve,reject)=>{
         setTimeout(()=>{
                        resolve("44");
                    },1000);

   });
  let smn=new Promise((resolve,reject)=>{
          setTimeout(()=>{
                         resolve("46");
                     },10000);
      });
```

```
    let p=await pune;
        let s=await smn;

        return [p,s];
}
let result= tempCalc();
result.then((r)=>{
        console.log(r);
});
```

Fetch function in JavaScript /Fetch API

Fetch API or function is modern interface in JavaScript which is responsible for send http request to server and get resource or data from a server
Fetch function replaces the older XMLHttpRequest method and provides a flexible way to fetch resources or data from the server means we can say fetch function is flexible for work with asynchronous data

Syntax:

```
fetch(url,options).
      then(response=>response.json())
      .then(data=>{
          data
   }).catch(err=>{

   });
```

URL : the API endpoint from which data is fetched
Options (optional) : specify method like as get,post etc as headers , body etc
response.json(): parse the response as JSON
catch(error): handle any errors that occur during the request

How fetch API works

1. A request is sent to the specified URL
2. The server processes the request and send a response
3. The response is converted to JSON(or another format)
4. Error handling using catch or try catch block.

Common HTTP Request methods in Fetch API

1. **GET :** get method is used for send request to server and retrieve data from a server **Example:** suppose we want to send the id of an employee to a server and fetch employee data.
2. **POST : this request is used to add some data or create some data on server** **Example:** suppose consider we want to submit form to server and save record in database at server side then we can use post
3. **PUT :** this method is used for updating data or resources at server side.
4. **DELETE :** this method is used for deleting resources at server side.

Example: we want to fetch dummy data from a following url

```
<html>
 <head>
   <title>fetching api</title>
   <script type='text/javascript'>

     function loadData(){

              let promise=fetch('https://jsonplaceholder.typicode.com/todos/');
      promise.
             then(response => response.json()).
                 then(json =>{

                              let tableBody=document.getElementById("tbd");
                               json.forEach((item)=>{
                                  let row=document.createElement("tr");

                                        let col=document.createElement("td");
                                        col.innerHTML=""+item.userId;
                                        row.appendChild(col);
                                        col=document.createElement("td");
                                        col.innerHTML=""+item.id;
                                        row.appendChild(col);
                                        col=document.createElement("td");
                                        col.innerHTML=""+item.title;
                                        row.appendChild(col);
                                        col=document.createElement("td");
                                        col.innerHTML=""+item.completed;
                                        row.appendChild(col);
                                        tableBody.appendChild(row);
                               });
                              });

             promise.catch((err)=>{
                console.log(err);
             });
            }

             let search=(str)=>{
               if(str.length>0)
                     {
                let id=parseInt(str);
                fetch(`https://jsonplaceholder.typicode.com/todos/${id}`)
           .then(response => response.json())
           .then(
                       json =>{
                              document.getElementById("tbd").innerHTML="";
                              let tableBody=document.getElementById("tbd");

                                 let row=document.createElement("tr");

                                        let col=document.createElement("td");
                                        col.innerHTML=""+json.userId;
                                        row.appendChild(col);
                                        col=document.createElement("td");
```

```
                                col.innerHTML=""+json.id;
                                row.appendChild(col);
                                col=document.createElement("td");
                                col.innerHTML=""+json.title;
                                row.appendChild(col);
                                col=document.createElement("td");
                                col.innerHTML=""+json.completed;
                                row.appendChild(col);
                                tableBody.appendChild(row);

                        })
                    }
                    else{
                      loadData();
                    }
            }
  </script>
 </head>
 <body onload="loadData()">
 <input type='text' name='name' value=''  style='width:80%;height:40px;margin-left:140px;'
 onkeyup='search(this.value)'/>
 <br><br>
   <table id="t" border="5" align="center"  width="80%">
     <tr>
             <th>USERID</th>
             <th>Id</th>
             <th>TITLE</th>
             <th>COMPLETED</th>
           </tr>
          <tbody id="tbd">

          </tbody>
   </table>
 </body>
</html>
```

JavaScript Module concept

Sunday,monday,tuesday,wed - study
Monday -

Core java + SQL +HTML + CSS +JS

Programming