JAVASCRIPT

Q. What is JavaScript?

JavaScript is a versatile, high level programming language and it is used for developing interactive and dynamic web pages in combination with HTML and CSS.
Now a days Javascript work as client side programming language as well as server side programming language using a node environment

Q. Why Use JavaScript or What are the benefits of JavaScript?

---

1. **Client side execution:** JavaScript runs directly in the user browser reducing server load and providing immediate feedback without reloading the page.
2. **Asynchronous operation:** If we use JavaScript we can update the partial web page or some portion of web page without loading complete page by using AJAX(Asynchronous javascript and XML)
3. **Versatility:** JavaScript is used for both client side and server side development(node.js) enabling full stack development with single language
4. **Rich Ecosystem :** JavaScript provide huge number of libraries to us and framework also like as React, vue.js,angular etc

5. **Cross Browser compatibility :** Modern Browser supports JavaScript ensuring that applications work consistently across different platforms and devices.
6. **Mobile Development :** JavaScript frameworks like React Native allow developers to build mobile for iOS and android reusing much of the code from web applications.

7. **Extensive Community support:** A large active developers community means you can find plenty of resources ,tutorials and support when needed
8. **Event Handling: JavaScript provides facility to apply events on HTML elements and execute some logic at run time according to the requirement of users.**

9. **Interactive web page development :** if we use JavaScript we can develop interactive web page using following feature
    a. Can apply runtime CSS on HTML element using JavaScript
    b. Can Create HTML element at run time and add on web page using DOM
    c. Can remove HTML element at run time from web page using DOM Feature
    d. JavaScript provide libraries to us for animation purpose

Etc

How to Work with JavaScript Practically

If we want to work with JavaScript we have two ways

1. **Using Client environment :** client side environment we can use JavaScript browser or embed JavaScript code in html file
2. **Using Server Environment:** if we want to JavaScript using server environment means we have to use JS in node environment

How to use JavaScript in client side environment

If we want to use JavaScript in browser or client side we have two ways

a. **Internal JavaScript :** Internal JavaScript means we embed the JavaScript code with html page or within same web page called as internal JavaScript

   How to use internal JavaScript

   If we want to use internal JavaScript we have the following steps.

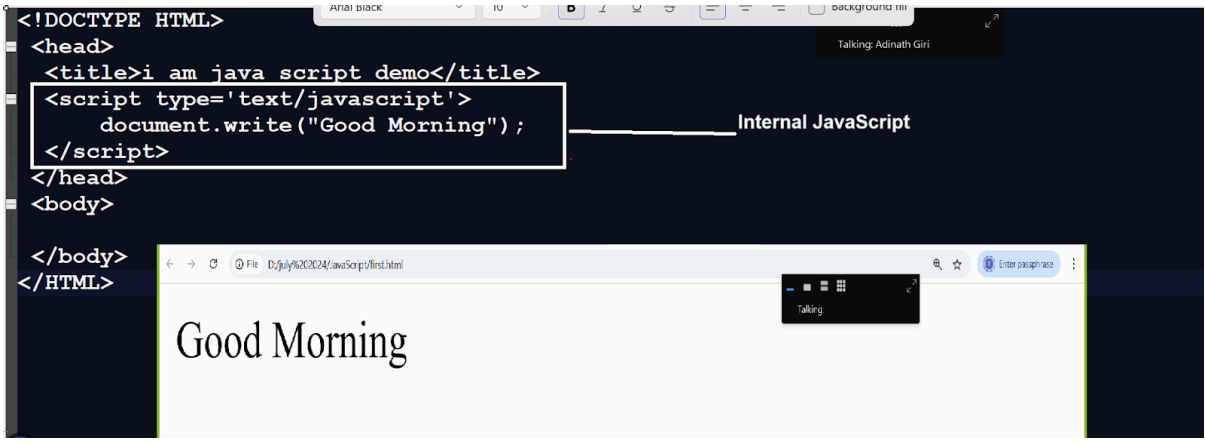
   a. **Create html page**
      Demo.html

      ```
      <!DOCTYPE HTML>
       <head>
        <title>i am java script demo</title>
       </head>
       <body>
      ```

</body>
</HTML>

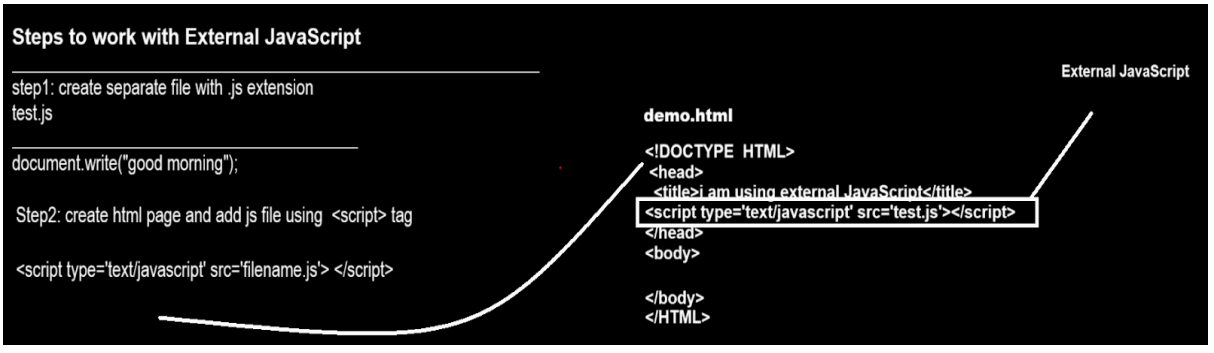
3. **Use Script tag in head section of webpage or body**

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



b. External JavaScript : External JavaScript means we can create a separate javascript file and write logic in that file and embed the file in an HTML page called external JavaScript.

Steps



How to use JavaScript in server environment

If we want to create server environment for JavaScript we required to installed node

Steps to work with javascript in server environment

Step1: install node software

If we want to download and install node we have to use following URL
https://nodejs.org/en/download

Step2: Download the node

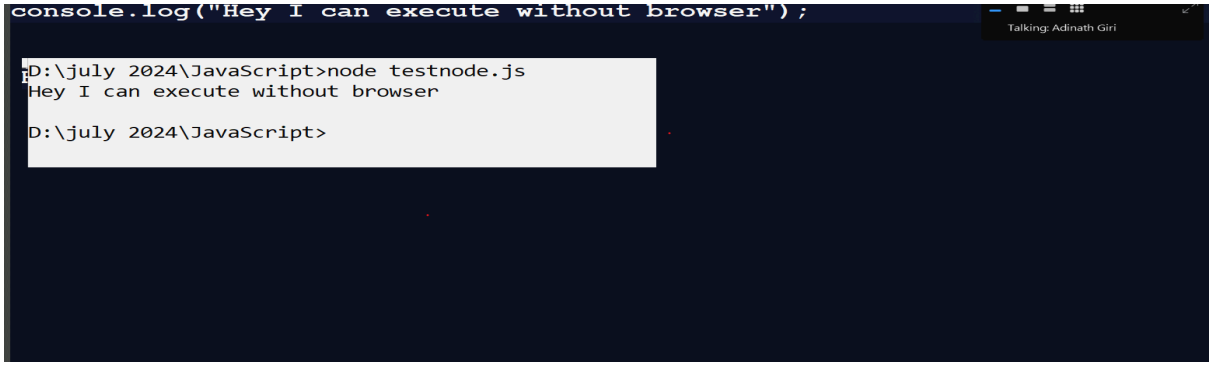
Note: when we install the node environment we can execute our JavaScript code without browser

Steps to run javascript code in node environment

---

1. **Create java script file**
   testnode.js
   document.write("Hey I can execute without browser");
2. **Open command and go where java script file save**

3. Type the command node filename.js



How to declare the variable in JavaScript

Note: JavaScript is dynamic type language

There are two types of languages we have for variable declaration

---

1. **Static type language:** static type language means language need to specify data type for variable declaration purpose called as static type language.
   Example of static type of language: C,JAVA,C++ ,C# etc
   int a;

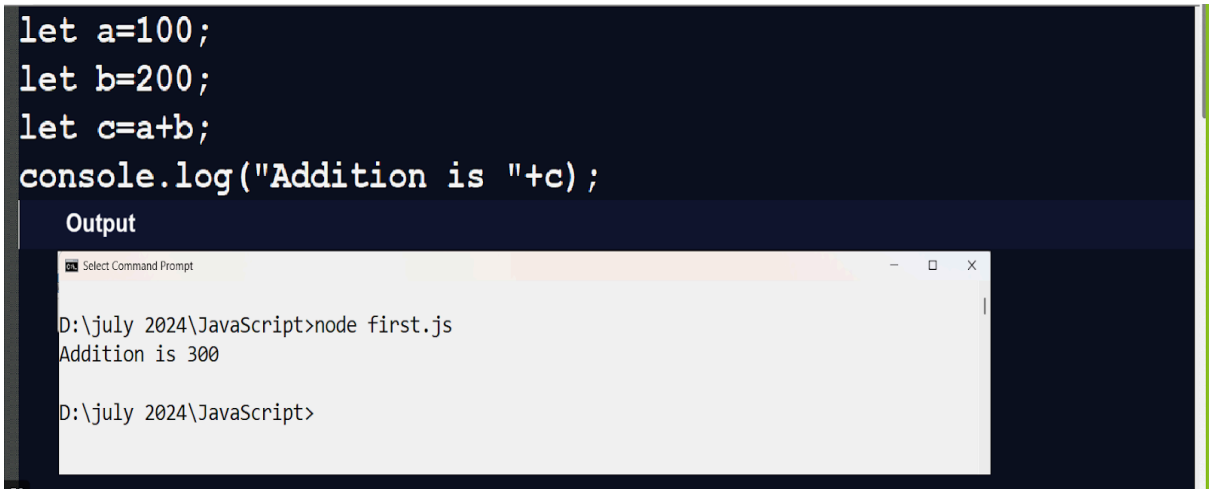
2. **Dynamic type language:** dynamic type language means language does not need specify data type at the time of variable declaration called as dynamic type language.

   Language: python , php, JavaScript etc

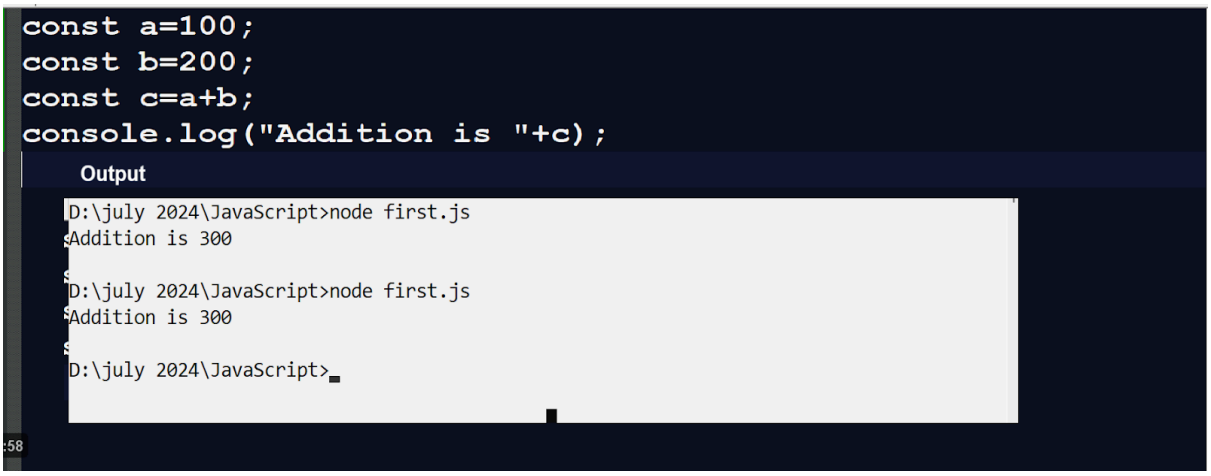
Note: if we think about dynamic type language, a variable can change its data type according to the value assigned in it.

If we want to declare the variable using JavaScript we have three ways

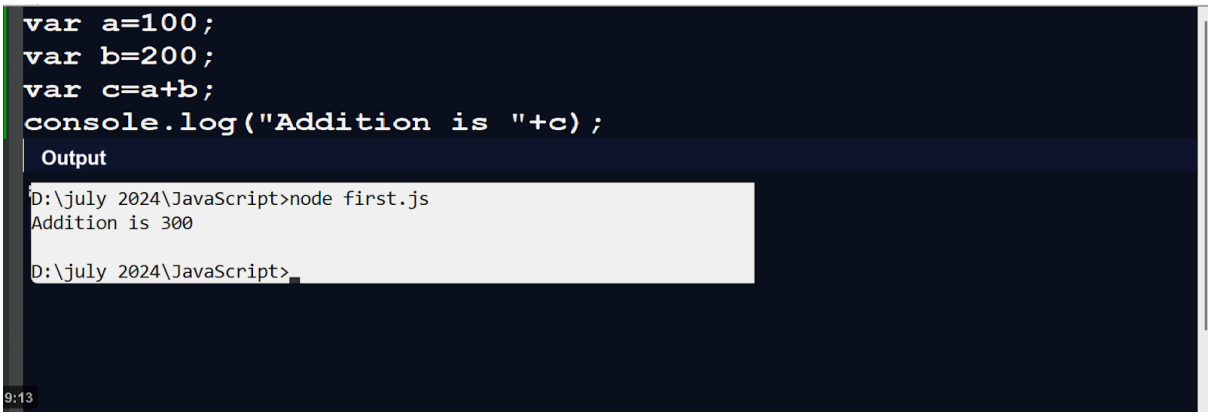
1. **Using let keyword**



2. **Using const keyword**

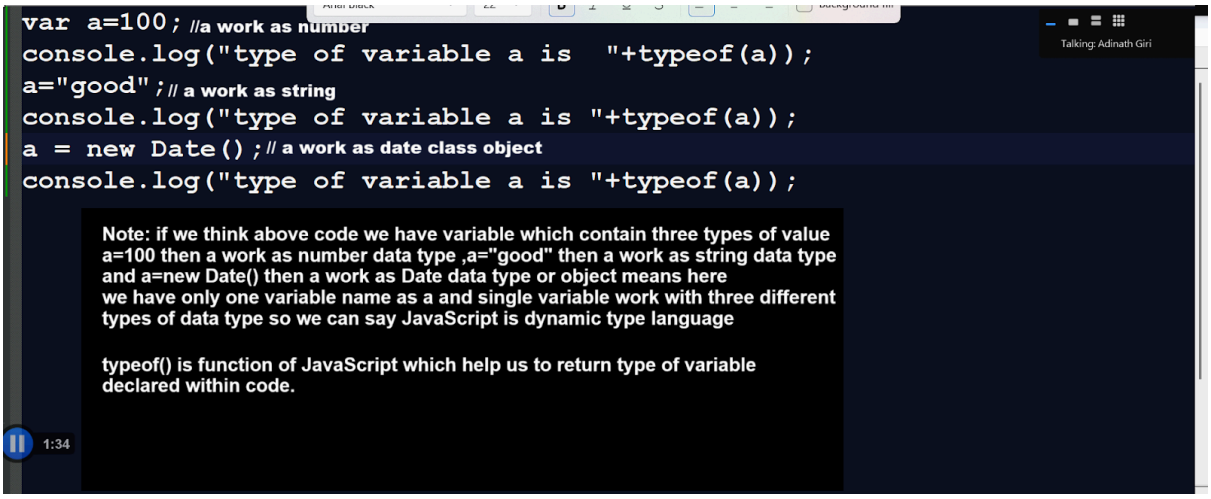


3. **Using var keyword**



Q1. How can we prove JavaScript is a dynamic type language?

A JavaScript variable can modify its data type according to the value assigned to it.

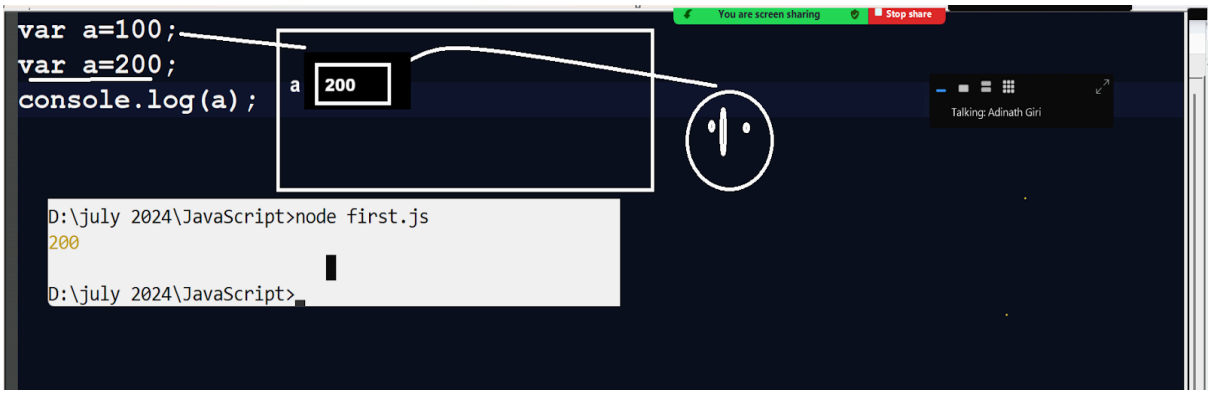


Q2. What is the difference between var , let and const keyword?

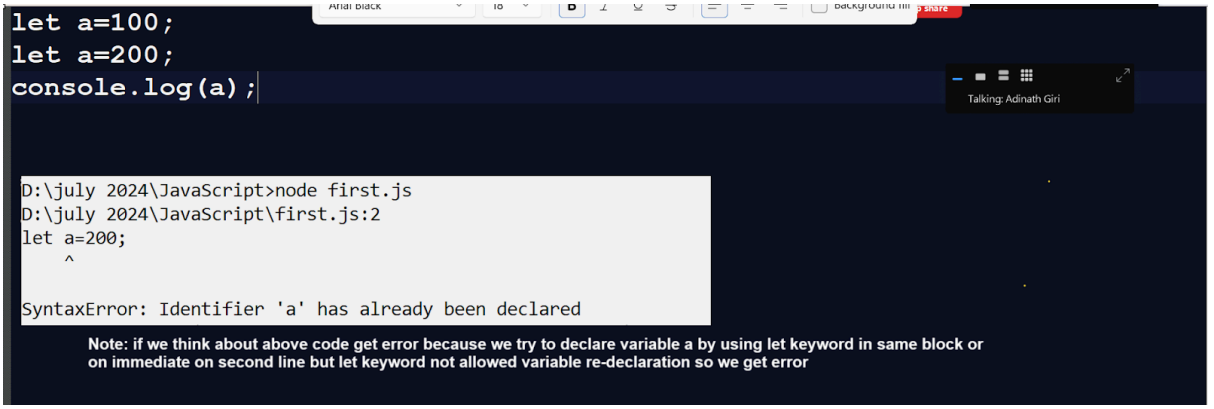
---

1. Var keyword allows redeclaration of variable and let and const not allowed re-declaration of variable.

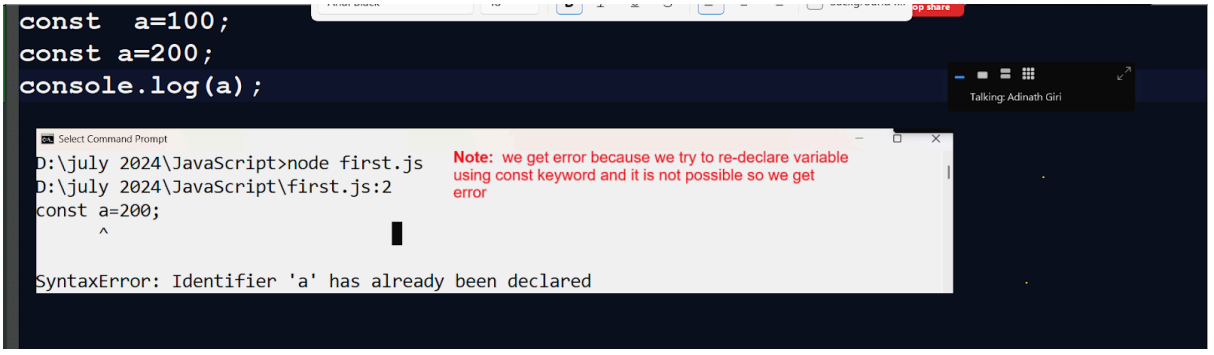
   Example using var keyword



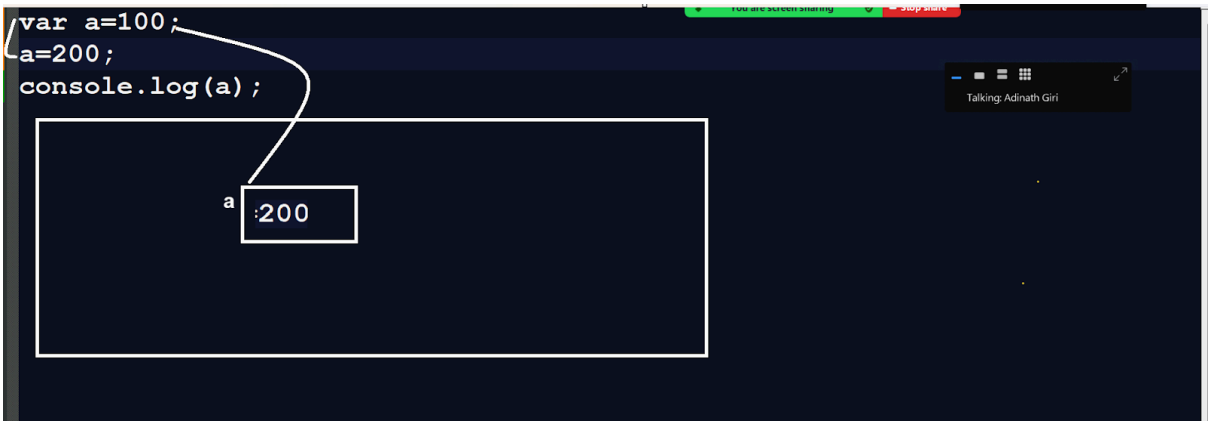
   Example using let keyword



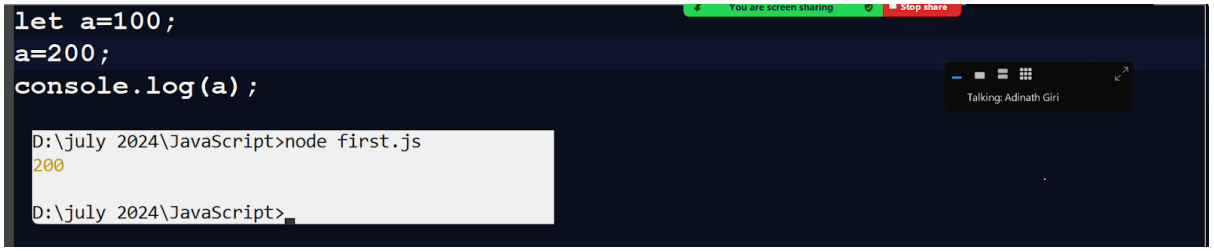
   Example using const keyword



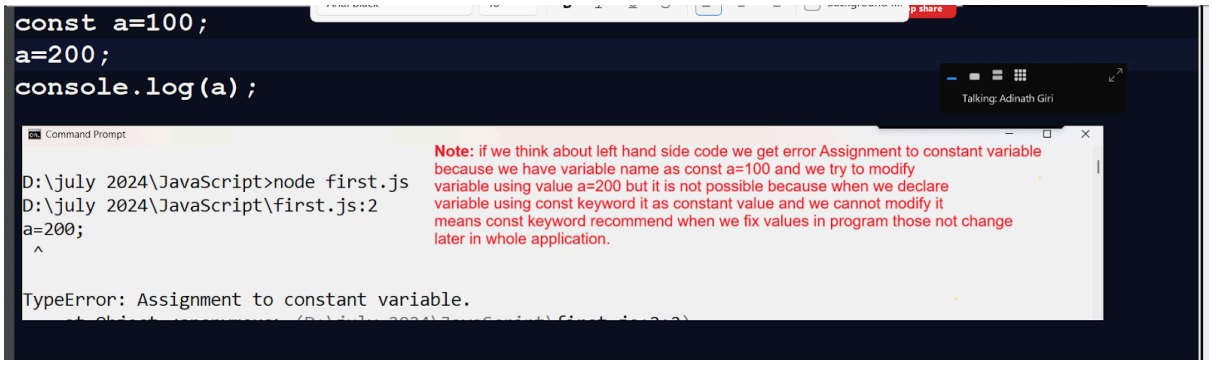
2. **Var and let keyword allow re-initialization of variable but const not allow re-initialization**



Example using let keyword

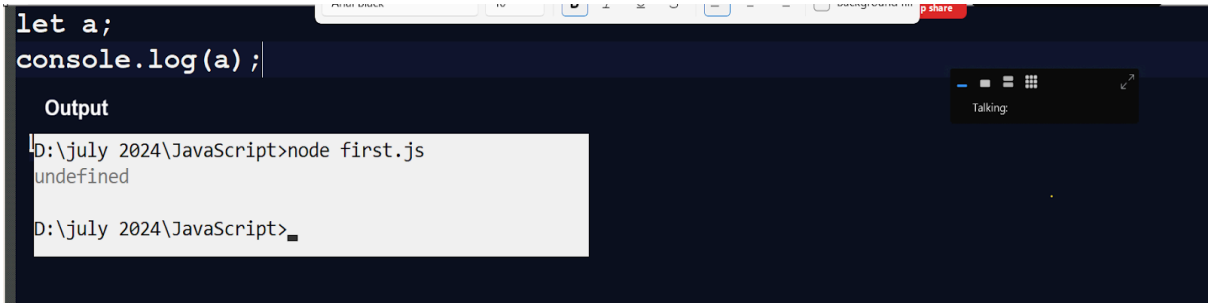


Example using const keyword

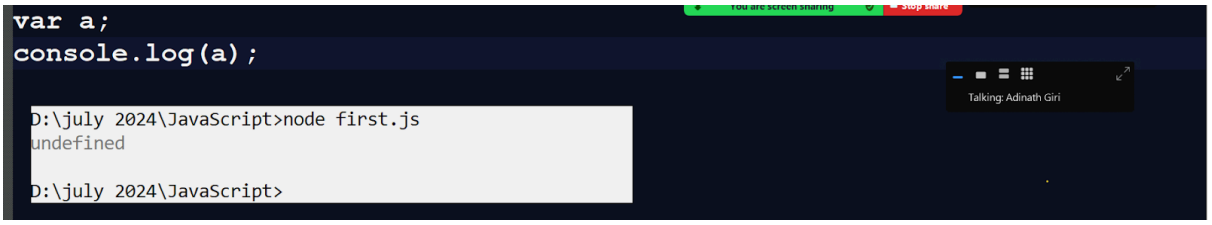


3. We can declare variable without initialization using let and var keyword but const cannot declare without initialization

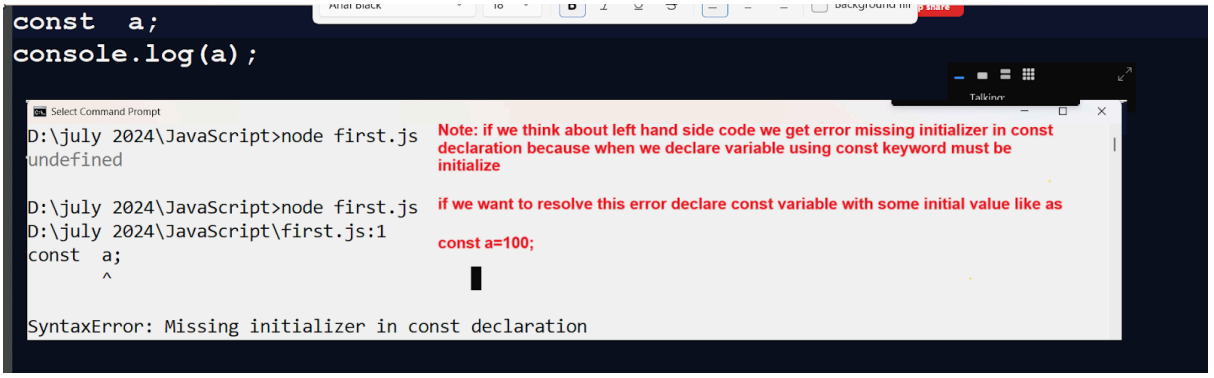
   Example using let keyword



Example using var keyword

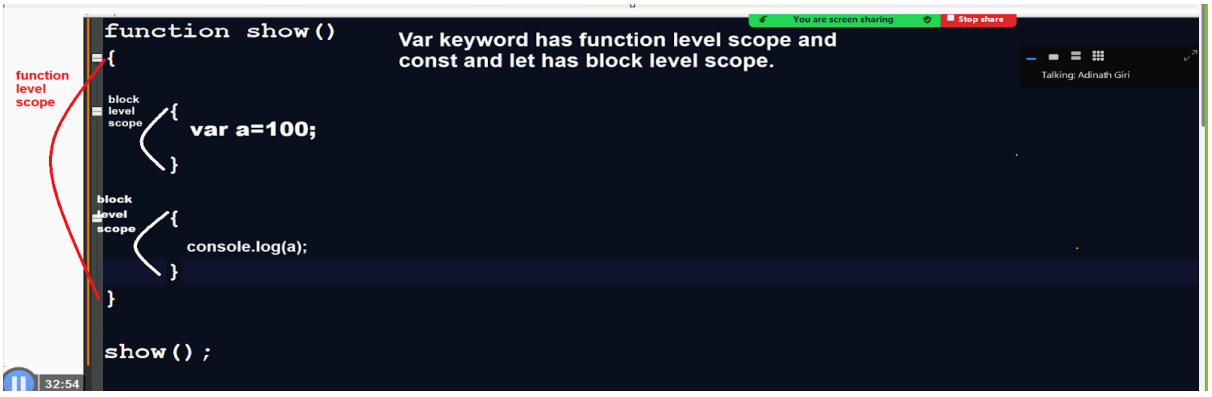


Example using const keyword

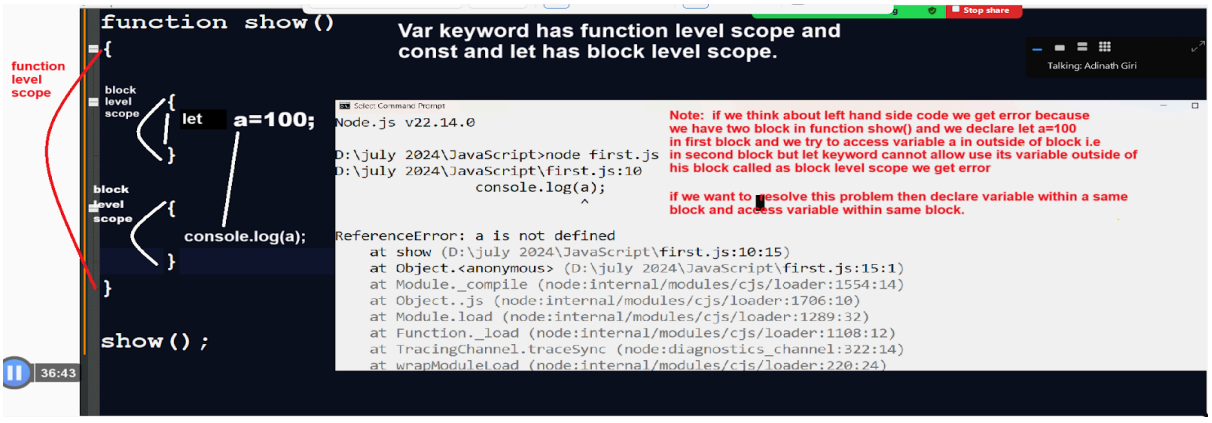


4. Var keyword has function level scope and const and let has block level scope.

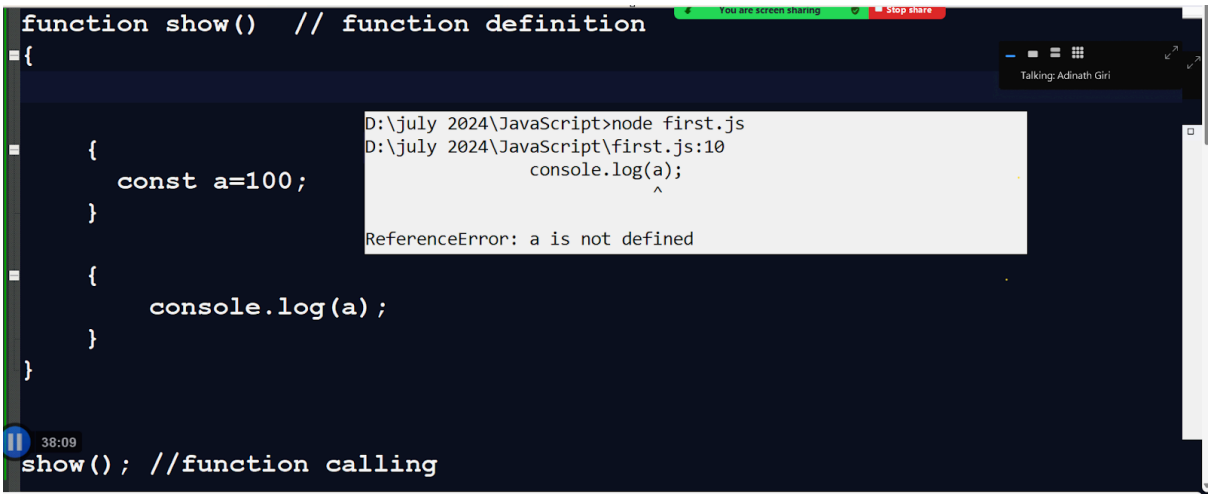
Example using var keyword



Example using let keyword



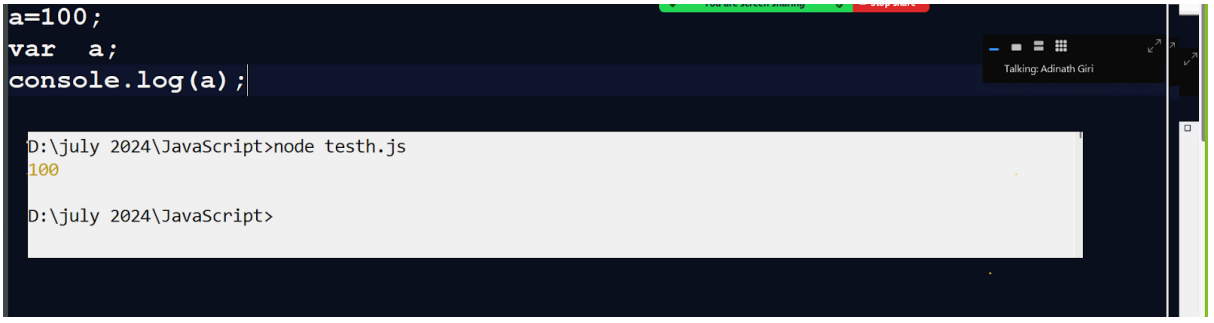
Example using const keyword



5. If we declare variable using var then it is hoisted if we declare variable using let and const not hoisted

Q. What is the hoisting variable?

Hoisting is a facility in JavaScript where we can initialize a variable first and after that we can declare it means we can use a variable before declaration called as a hosting feature in JavaScript.



How to declare input using javascript

If we want to accept input using JavaScript we have to use prompt() dialog

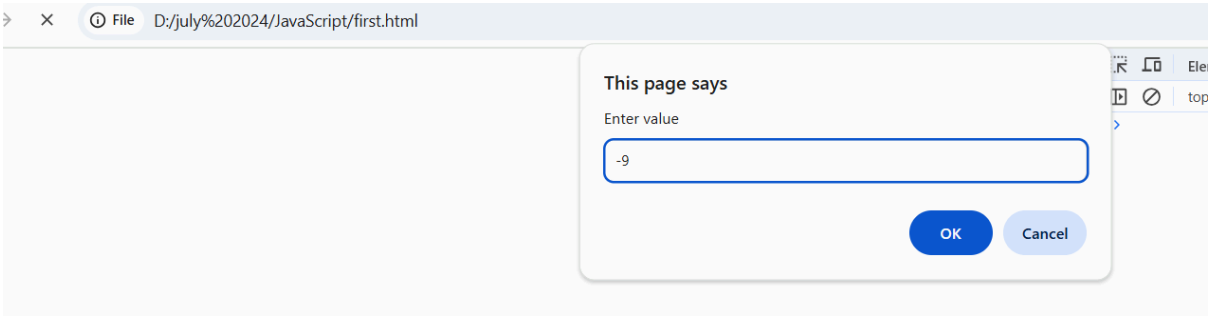
Example:

<html>

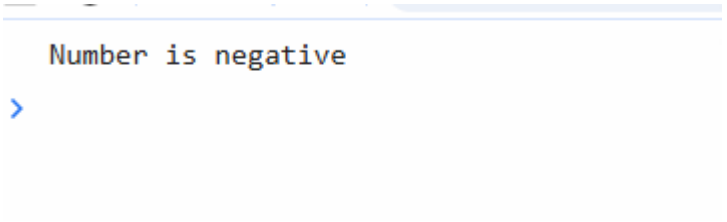
<head>

```
 <title>test input dialog</title>
  <script type='text/javascript'>
   let a=prompt("Enter value");
        let num=parseInt(a);
         if(num>0)
         { console.log("Number is positive");
         }
         else
         {  console.log("Number is negative");
         }
   </script>
 </head>

 <body>

 </body>
</html>
```

Output



Output

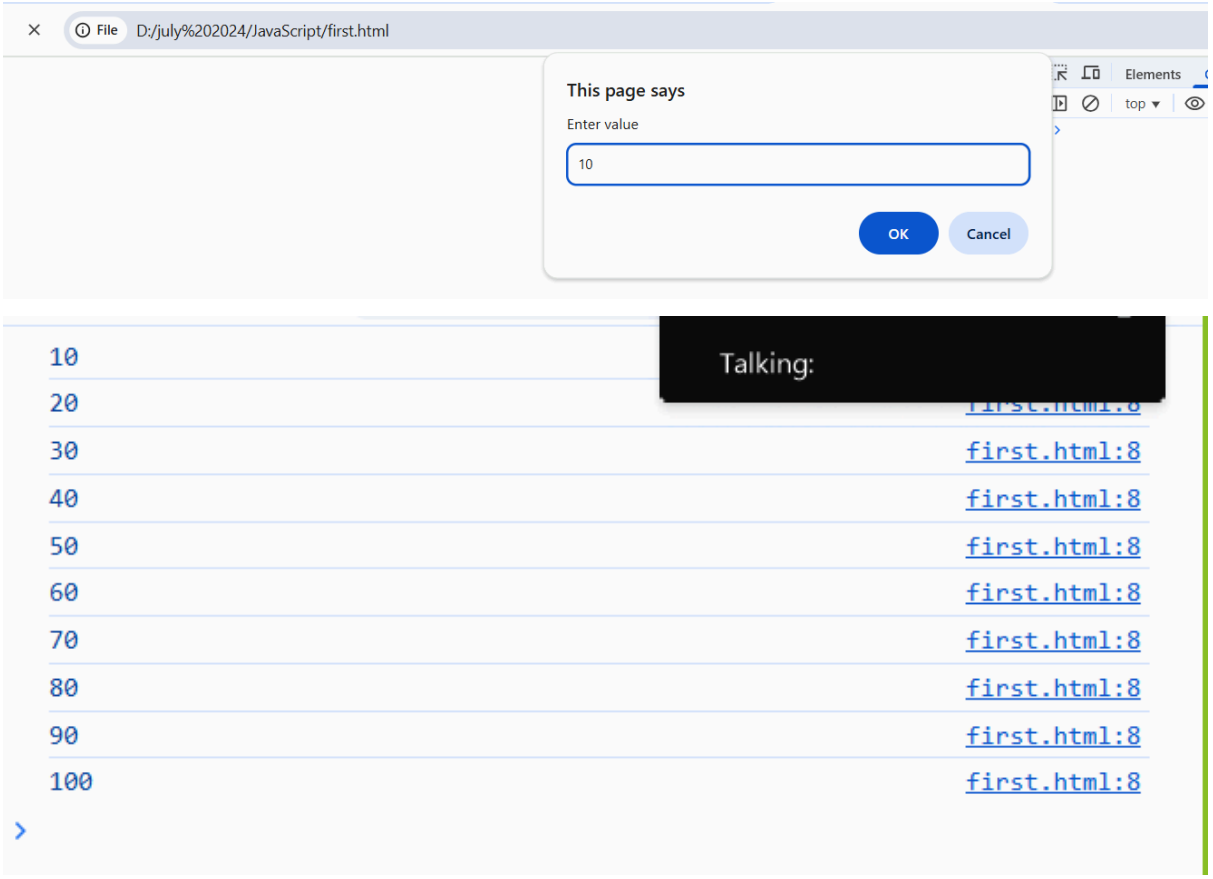


Example: WAP to input number and print its table

```
<html>
 <head>
  <title>test input dialog</title>
   <script type='text/javascript'>
    let a=prompt("Enter value");
        let num=parseInt(a);
```

```
        for(var i=1; i<=10; i++)
        { console.log(i*num);
        }
  </script>
 </head>

 <body>

 </body>
</html>
```

Output



Event Handling in JavaScript

Event means action perform by user on html element and execute some JavaScript logic on event or action by calling JavaScript function

Example: button click , key press, key up, key down , mouse over on html element etc

How to perform event by using JavaScript

Steps to perform event by using JavaScript

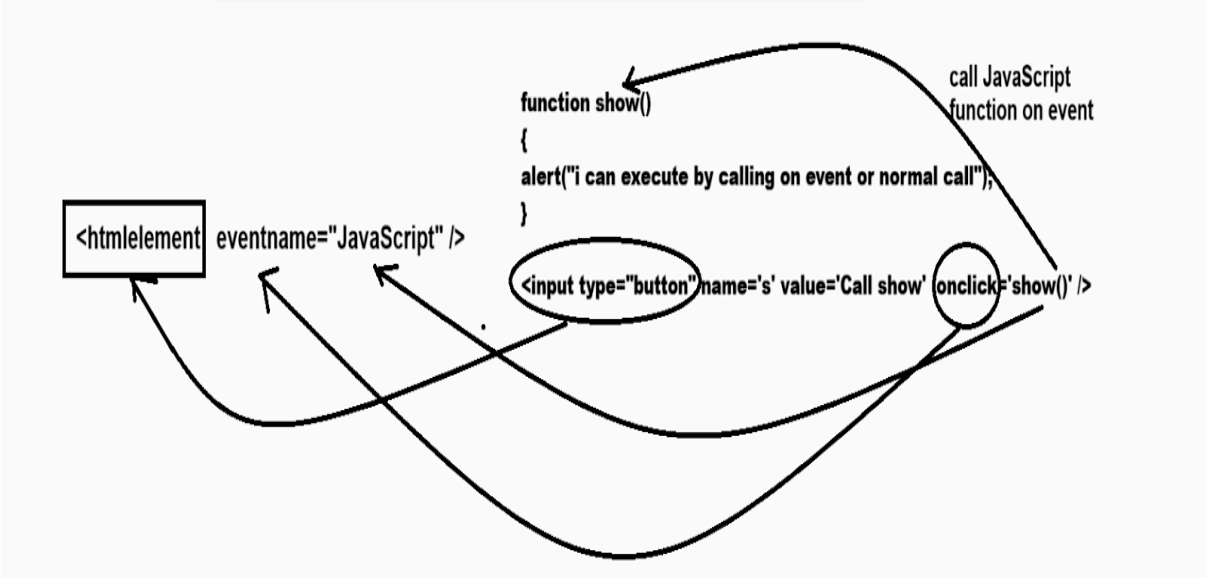
---

1. **Define function for writing event logics**
   Syntax for defining function in JavaScript

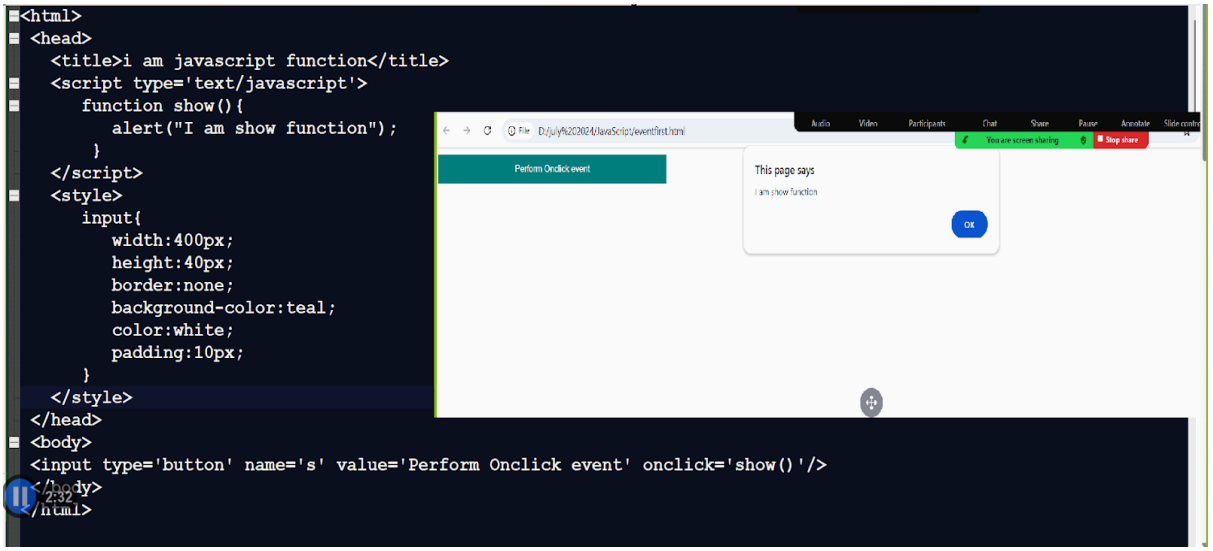
```
function functioname(variables)
{
    write here your logics
}
```

   Example
   function show()
   { alert(“i can execute by calling on event or normal call”);
   }

2. **Create HTML elements and apply event on it and call function on event**



Example: we want to design a web page with one button and when a user clicks on a button we want to call a JavaScript function and display some message using an alert box.



Types of events in JavaScript

Onclick : this event can execute when we click on html element like as click on button etc

Onmouseover : this event execute when enter mouse on html element

```
<html>
 <head>
  <title>i am javascript function</title>
  <script type='text/javascript'>
    function show(){
          alert("I am show function");
        }
  </script>
  <style>
    input{
          width:400px;
              height:40px;
              border:none;
              background-color:teal;
              color:white;
              padding:10px;
        }
  </style>
 </head>
 <body>
```

```
 <input type='button' name='s' value='Perform Onclick event'
onmouseover='show()'/>
 </body>
</html>
```

Onmouseleave : this event execute when your mouse leave from html element

Example with source code

```
<html>
 <head>
   <title>i am javascript function</title>
   <script type='text/javascript'>
      function show(){
             alert("I am show function");
           }
   </script>
   <style>
      input{
             width:400px;
                 height:40px;
                 border:none;
                 background-color:teal;
                 color:white;
                 padding:10px;
           }
   </style>
 </head>
 <body>
 <input type='button' name='s' value='Perform Onclick event'
onmouseleave='show()'/>
 </body>
</html>
```

Onkeyup :
Onkeydown

Onkeypress

Example with source code

```
<html>
 <head>
  <title>i am javascript function</title>
  <script type='text/javascript'>
    function show(){
          alert("I am show function");
        }
  </script>
  <style>
    input{
          width:400px;
              height:40px;
              border:none;
              background-color:teal;
              color:white;
              padding:10px;
        }
  </style>
 </head>
 <body>
 <input type='text' name='s' value='' onkeypress='show()'/>
 </body>
</html>
```

Onchange : onchange event execute when html element change its state normally we use this event on drop down list, checkbox ,radio etc

Example with source code

```
<html>
 <head>
  <title>i am javascript function</title>
  <script type='text/javascript'>
    function show(){
```

```
            alert("I am show function");
          }
  </script>
  <style>
    select{
          width:400px;
              height:40px;
              border:none;
              background-color:teal;
              color:white;
              padding:10px;
         }
  </style>
 </head>
 <body>
  <select onchange="show()">
    <option>JAVA</option>
        <option>PHP</option>
        <option>DOTNET</option>
  </select>
 </body>
</html>
```

onblur: when cursor left from html control called as onblur events

```
<html>
 <head>
  <title>i am javascript function</title>
  <script type='text/javascript'>
    function show(){
          alert("I am show function");
         }
  </script>
  <style>
    input{
          width:400px;
              height:40px;
```

```
                padding:10px;
            }
    </style>
 </head>
 <body>
    <input type='text' name='s' value='' onblur='show()' /><br><br>
        <input type='text' name='s1' value=''  /><br><br>

 </body>
</html>
```

Onfocus: when cursor enters in html control called as onfocus event.

```
<html>
 <head>
   <title>i am javascript function</title>
   <script type='text/javascript'>
     function show(){
            alert("I am show function");
          }
   </script>
   <style>
     input{
            width:400px;
                height:40px;

                padding:10px;
          }
   </style>
 </head>
 <body>
    <input type='text' name='s' value='' /><br><br>
        <input type='text' name='s1' value=''   onfocus='show()'/><br><br>

 </body>
</html>
```

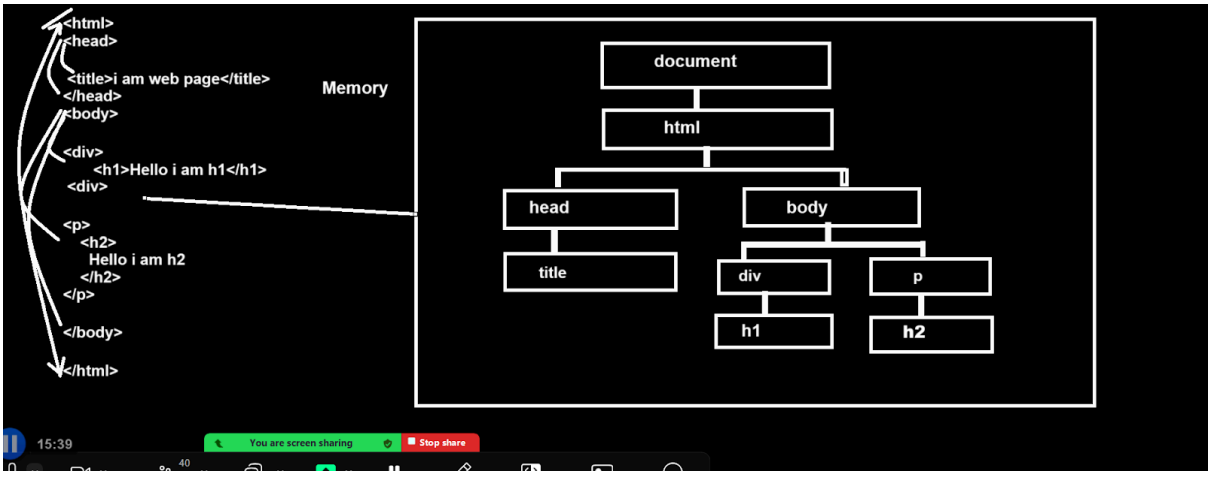
DOM Manipulation using JavaScript

DOM stands for document object model

Q. What is a document?

Document is object in JavaScript and which is parent of every object of JavaScript
Means if we think about JavaScript web page consider as object and every html consider as child of document
DOM indicates how a web page is represented in memory by a javascript engine called DOM.

How web page represent in memory by DOM Tree



Q. Why use DOM in JavaScript or What are the benefits of DOM in JavaScript?

---

1. DOM help us to change the HTML element at run time
   Means using DOM we can modify the text of html tag or text from input control
2. DOM help us to apply runtime CSS on html page or html element
3. DOM can create HTML element at run time
4. DOM can remove HTML element at run time
5. DOM can apply run time event on html element by using listernes Etc

If we want to work with DOM practically JavaScript provide one inbuilt object to us name as document and document object provide n number of methods to us which help us to work with DOM or can modify or create or remove etc operation on html element at run time by using JavaScript

Methods of document object

_____

document.createElement(): this method help us to create new HTML element by using JavaScript

Example: document.createElement("h1");

document.getElementById("id"): this method helps us to fetch html elements in javascript using id.

document.getElementsByClassName("classname): this method fetches or access HTML elements by using its class name.

document.appendChild(element): this method can append html element in outer element or mark newly created html element as child of another element

document.removeElement(element): this method can remove element from DOM

document.write(string): this method can write text on a web page using a document object.

document.replaceChild(new,old): this method can replace new element on old HTML element

Properties of DOM

_____

element.innerHTML=html element content : this property help us to add new data in opening and closing brace of html element

Element.attribute =new value : this method access the html element attribute in javascript and provide value to them

element.style.property : this property help us apply runtime css on HTML element using JavaScript

element.setAttribute(attribute,value): change attribute value of an HTML element

Example: we want to design a web page with one button and heading and when the user clicks on the button then access the text of heading in Javascript and display using the alert box.

1. **Design web page using html with one button and heading**

```
<html>
 <head>
  <title>first example of DOM</title>
  <style>
   input{
        width:400px;
            height:40px;
      }
  </style>
 </head>
 <body>
  <input type='button' name='s' value='Access Heading Data'/>
  <br><br>
  <h1>good morning india</h1>
 </body>
</html>
```

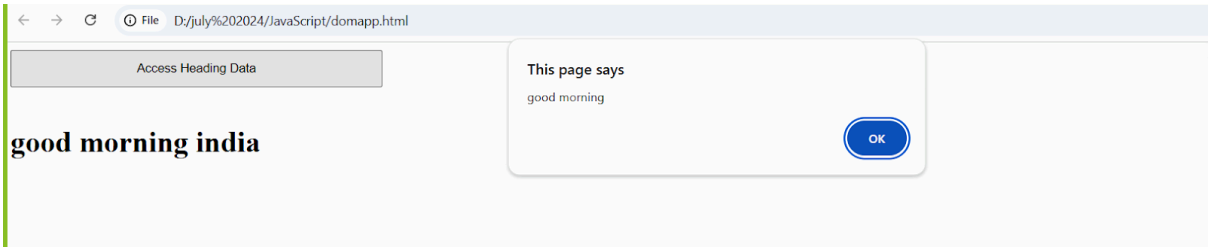
2. **Assign id to html element i.e h1 to in our example and define function using java script and call it on button click**

   Example with source code

   _____
   _

```
<html>
 <head>
  <title>first example of DOM</title>
  <script type='text/javascript'>
   function accessHeading(){
        alert("good morning");
```

```
        }
  </script>
  <style>
   input{
          width:400px;
              height:40px;
        }
  </style>
 </head>
 <body>
  <input type='button' name='s' value='Access Heading Data'
onClick="accessHeading()"/>
  <br><br>
  <h1 id="h">good morning india</h1>
 </body>
</html>
```



3. Access heading by using getElementById() method in JavaScript and using its innerHTML property for accessing data.

Complete source code

```
<html>
 <head>
   <title>first example of DOM</title>
   <script type='text/javascript'>
     function accessHeading(){
              let heading = document.getElementById("h");
                  let text=heading.innerHTML;
                  alert(text);
          }
   </script>
   <style>
     input{
            width:400px;
                height:40px;
         }
   </style>
 </head>
 <body>
   <input type='button' name='s' value='Access Heading Data'
onClick="accessHeading()"/>
   <br><br>
   <h1 id="h">good morning india</h1>
 </body>
</html>
```

Example: Design web page with one button and heading tag and heading tag contain initial text good morning and we want to change text good afternoon in heading tag after button click.

Example with source code

```
<html>
 <head>
   <title>first example of DOM</title>
   <script type='text/javascript'>
     function accessHeading(){
            let heading = document.getElementById("h");
```

```
                heading.innerHTML="good afternoon";

        }
  </script>
  <style>
   input{
          width:400px;
              height:40px;
        }
  </style>
 </head>
 <body>
  <input type='button' name='s' value='Access Heading Data'
onClick="accessHeading()"/>
  <br><br>
  <h1 id="h">good morning india</h1>
 </body>
</html>
```

Output (before button click)

Output: After button click

How to access the input control values in Javascript

If we want to access input control values in JavaScript we have to access the value attribute of input control

Example: we want to design a web page with two textboxes and one button and we want to input in the first textbox and when the user click on the button then we want to access the value from the first textbox and calculate its square and show it in the second textbox.

Example with source code

```
<html>
 <head>
  <title>first example of DOM</title>
  <script type='text/javascript'>
   function accessHeading(){
            /*
                let firstTextBox=document.getElementById("first");
                let val=firstTextBox.value;
                let a=parseInt(val);
                let result=a*a;

                let secondTextBox=document.getElementById("second");
                secondTextBox.value=""+result;
                */

               //short cut logics

document.getElementById("second").value=""+parseInt(document.getElementById("first").value)*
                parseInt(document.getElementById("first").value);
        }
  </script>
  <style>
   input{
          width:400px;
              height:40px;
       }
  </style>
 </head>
 <body>
```

```
   <input type='text' name='f' id='first' value='' /><br><br>
   <input type='text' name='s' id='second' value='' /><br><br>
   <input type='button' name='s' value='Calculate Square'
onClick="accessHeading()"/>
   <br><br>
 </body>
</html>
```

How to create elements at run time by using DOM?

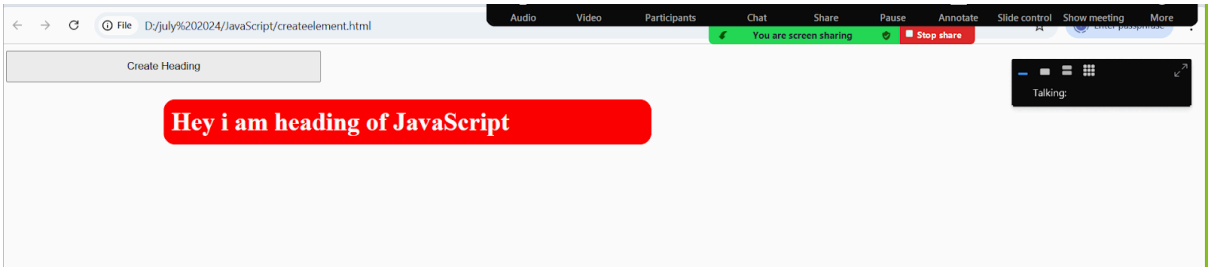
If we want create html element by using DOM we have method name as createElement() and this method help us to create element usingDOM

Steps to create HTML element using DOM

---

1. **Create element using document.createElement() method**
2. **Add element as child of another element**

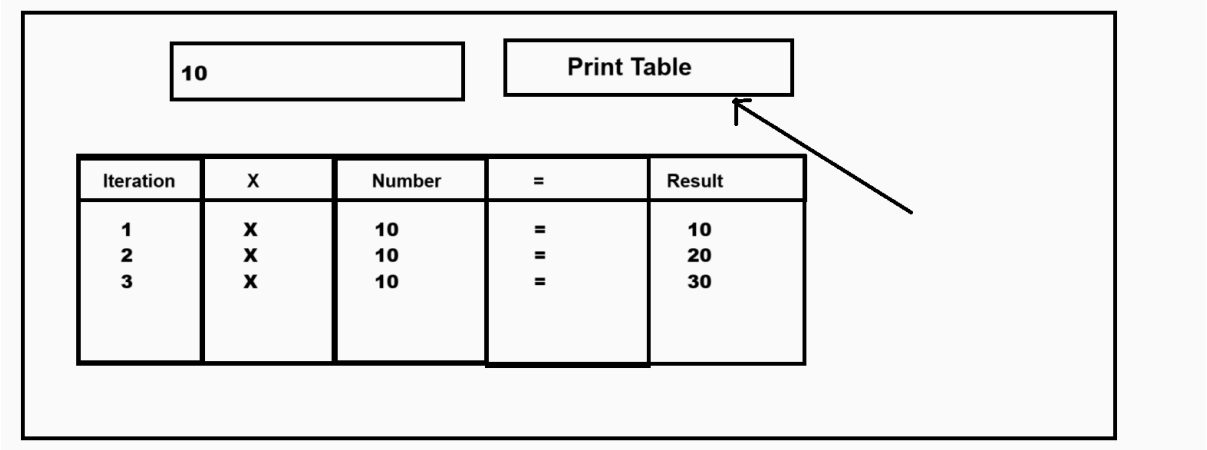
Example:



Example:

```
<html>
 <head>
  <title>creating element</title>
  <script type='text/javascript'>
     function createTag()
         {
              let myheading=document.createElement("h1");
                  myheading.innerHTML="Hey i am heading of JavaScript";
                       myheading.style.backgroundColor="red";
                       myheading.style.color="white";
                       myheading.style.padding="10px";
                       myheading.style.borderRadius="15px";
                       myheading.style.width="600px";
                       myheading.style.marginLeft="200px";

                  let bodyTag=document.getElementById("b");
                  bodyTag.appendChild(myheading);
         }
  </script>
   <style>
          input{
                width:400px;
                height:40px;
               }
       </style>
 </head>
 <body id="b">
  <input type='button' name='s' value='Create Heading'
onclick='createTag()' />

 </body>
</html>
```

Example: we want to design web page like as



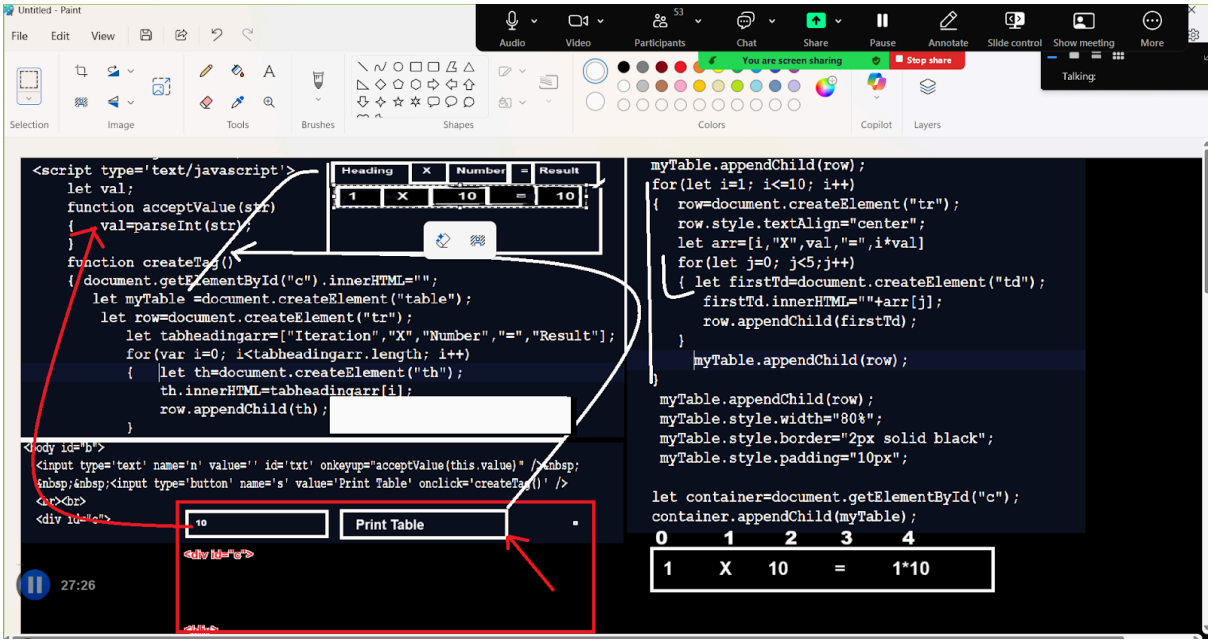
Example with source code

```
<html>
 <head>
  <title>creating element</title>
  <script type='text/javascript'>
     let val;
         function acceptValue(str)
         {   val=parseInt(str);
         }
     function createTag()
         { document.getElementById("c").innerHTML="";
           let myTable =document.createElement("table");
                let row=document.createElement("tr");
                  let tabheadingarr=["Iteration","X","Number","=","Result"];
                      for(var i=0; i<tabheadingarr.length; i++)
                      {   let th=document.createElement("th");
                            th.innerHTML=tabheadingarr[i];
                            row.appendChild(th);
                      }
                      myTable.appendChild(row);
                      for(let i=1; i<=10; i++)
                      {  row=document.createElement("tr");
                        row.style.textAlign="center";
                            let firstTd=document.createElement("td");
                            firstTd.innerHTML=""+i;
                            row.appendChild(firstTd);
```

```
                                firstTd=document.createElement("td");
                                firstTd.innerHTML="X";
                                row.appendChild(firstTd);
                                firstTd=document.createElement("td");
                                firstTd.innerHTML=""+val;
                                row.appendChild(firstTd);
                               firstTd=document.createElement("td");
                               firstTd.innerHTML="=";
                               row.appendChild(firstTd);
                                firstTd=document.createElement("td");
                                firstTd.innerHTML=""+(i*val);
                                row.appendChild(firstTd);

                                myTable.appendChild(row);
                            }
                        myTable.appendChild(row);
                            myTable.style.width="80%";
                            myTable.style.border="2px solid black";
                            myTable.style.padding="10px";

                           let container=document.getElementById("c");
                           container.appendChild(myTable);

            }
    </script>
     <style>
             input{
                    width:400px;
                    height:40px;
                  }
          </style>
 </head>
 <body id="b">
   <input type='text' name='n' value='' id='txt'
onkeyup="acceptValue(this.value)" /> 
     <input type='button' name='s' value='Print Table'
onclick='createTag()' />
```

```
  <br><br>
  <div id="c">

  </div>
 </body>
</html>
```

Or



Example with source code

```
<html>
 <head>
  <title>creating element</title>
  <script type='text/javascript'>
     let val;
          function acceptValue(str)
          {   val=parseInt(str);
          }
     function createTag()
          { document.getElementById("c").innerHTML="";
            let myTable =document.createElement("table");
                 let row=document.createElement("tr");
                   let tabheadingarr=["Iteration","X","Number","=","Result"];
                        for(var i=0; i<tabheadingarr.length; i++)
                        {   let th=document.createElement("th");
                              th.innerHTML=tabheadingarr[i];
```

```
                    row.appendChild(th);
                }
                myTable.appendChild(row);
                for(let i=1; i<=10; i++)
                {  row=document.createElement("tr");
                   row.style.textAlign="center";
                     let arr=[i,"X",val,"=",i*val]
                     for(let j=0; j<5;j++)
                     { let firstTd=document.createElement("td");
                          firstTd.innerHTML=""+arr[j];
                          row.appendChild(firstTd);
                     }
                  myTable.appendChild(row);
                  }
               myTable.appendChild(row);
                 myTable.style.width="80%";
                 myTable.style.border="2px solid black";
                 myTable.style.padding="10px";

                let container=document.getElementById("c");
                container.appendChild(myTable);

        }
  </script>
   <style>
          input{
               width:400px;
               height:40px;
             }
       </style>
 </head>
 <body id="b">
  <input type='text' name='n' value='' id='txt'
onkeyup="acceptValue(this.value)" /> 
    <input type='button' name='s' value='Print Table'
onclick='createTag()' />
  <br><br>
```

```
  <div id="c">

  </div>
 </body>
</html>
```

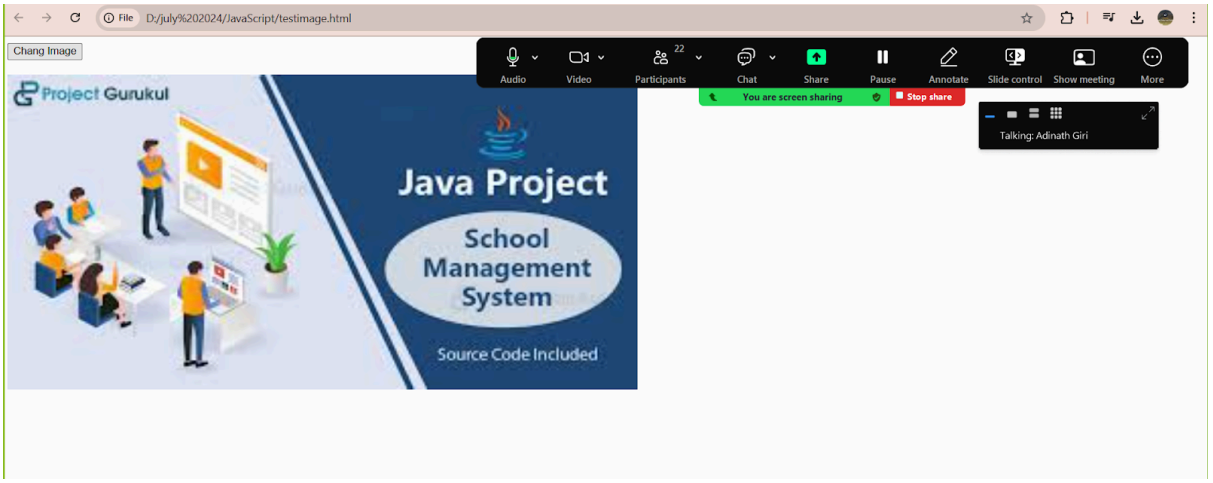
How to create input control using DOM

Example: we want to design web page with one button and when user click on button then we want to textbox using DOM and add in web page

```
<html>
 <head>
  <title>creating element</title>
  <script type='text/javascript'>
       function createTag()
         {   let myTextBox=document.createElement("input");
                  myTextBox.setAttribute("type","text");
                      myTextBox.setAttribute("name","name");
                      myTextBox.setAttribute("value","100");
                     let container=document.getElementById("c");
                     container.appendChild(myTextBox);
         }
  </script>
   <style>
          input{
                width:400px;
                height:40px;
               }
       </style>
 </head>
 <body id="b">
   &amp;nbsp;&amp;nbsp;<input type='button' name='s' value='Print Table'
onclick='createTag()' />
  <br><br>
  <div id="c">
  </div>
 </body>
</html>
```

We want to access HTML element attribute or tag using DOM

```
<html>
 <head>
  <title>image gallay application </title>
  <script type='text/javascript'>
    let count=-1;
        let imgArr=['a.jpeg','b.jpeg','c.jpeg','d.jpeg']
    function changeImage()
        { let myImage=document.getElementById('img');
            ++count;
                if(imgArr.length!=count)
                  myImage.src=imgArr[count];
                 else{
                      count=0;
                       myImage.src=imgArr[count];
                 }
        }
  </script>
 </head>
 <body>
  <input type='button' name='s' value='Chang Image'
onClick='changeImage()'/><br/><br/>
  <img src='a.jpeg' width='800px' height='400px' id='img'/>
 </body>
</html>
```



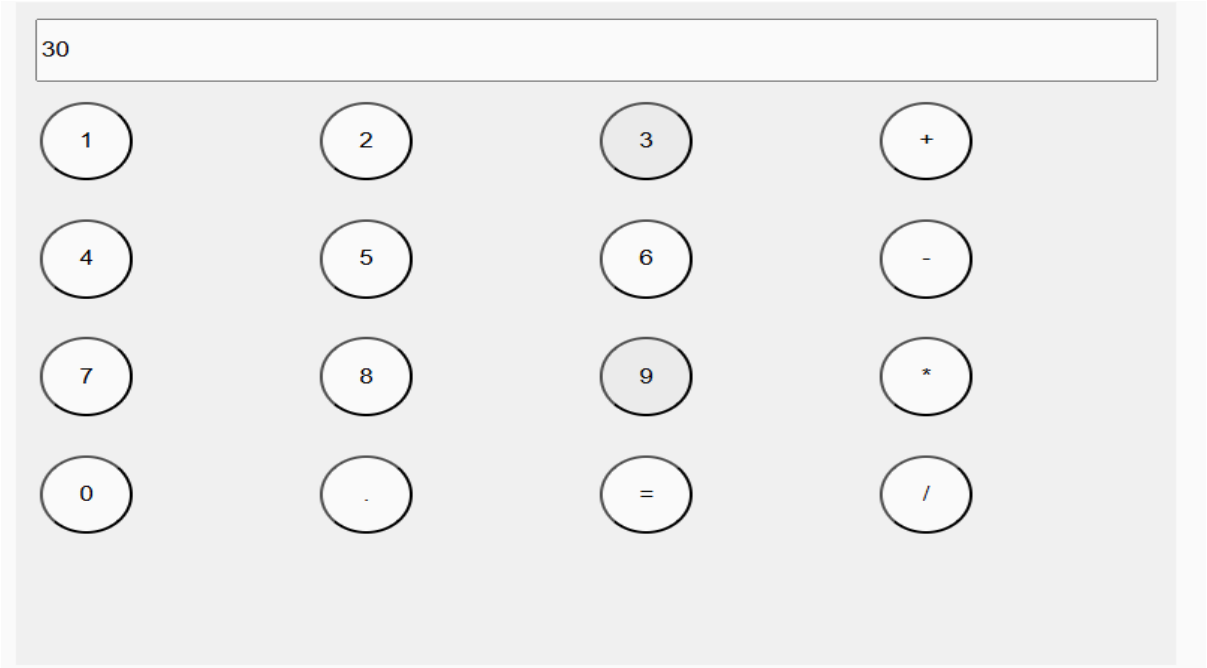
Example with source code

```
let labels=[
            ['1','2','3','+'],
                        ['4','5','6','-'],
                        ['7','8','9','*'],
                        ['0','.','=','/']
                      ];
let btnArr=[];
function createPanel()
{    let str="";
   let first,second,result;
        let grid=document.createElement("table");
        grid.style.width='600px';
        grid.style.height='300px';
        let count=0;
        let choice=0;
        for(var i=0; i<4;i++)
        {   var row=document.createElement("tr");
              for(var j=0; j<4; j++)
              {
                    var col=document.createElement("td");
                        var btn=document.createElement("input");
                        btn.setAttribute("type","button");
                        btn.setAttribute("value",labels[i][j]);
                        btn.setAttribute("name","btn");
                        btn.setAttribute("id",labels[i][j]);
                        btn.style.width="50px";
                        btn.style.height="50px";
                        btn.style.borderRadius="50%";

                        btn.addEventListener('mouseover',function(){

                                this.style.backgroundColor="teal"
                                this.style.color="white";
                        });
```

```
                btn.addEventListener('mouseleave',function(){

                    this.style.backgroundColor="white"
                    this.style.color="black";
                });
                btn.addEventListener('click',function(){
                    let
textInput=document.getElementById("txt");
                    str=textInput.value;
                    if(this.value=='+')
                    {
first=document.getElementById("txt").value;
                    document.getElementById("txt").value="";
                        choice=1;
                    }
                    else  if(this.value=='-')
                    {
first=document.getElementById("txt").value;
                    document.getElementById("txt").value="";
                        choice=2
                    }
                    else if(this.value=="=")
                    {

second=document.getElementById("txt").value;
                            switch(choice)
                            {
                                case 1:

result=parseInt(first)+parseInt(second);

document.getElementById("txt").value=""+result;
                                                break;
                                                case 2:

result=parseInt(first)-parseInt(second);
```

```
document.getElementById("txt").value=""+result;
                                                    break;
                                                    default:
                                                    alert("wrong choice");
                                                }
                                            }
                                            else
                                            {
                                              str=str+this.value;
                                              textInput.value=str;
                                            }
                                        });
                                        ++count;
                                        col.appendChild(btn);
                                        row.appendChild(col);
                                }
                                grid.appendChild(row);
                        }
                        let panel=document.getElementById("panel");
                        panel.appendChild(grid);
}
```

Output



Array in JavaScript

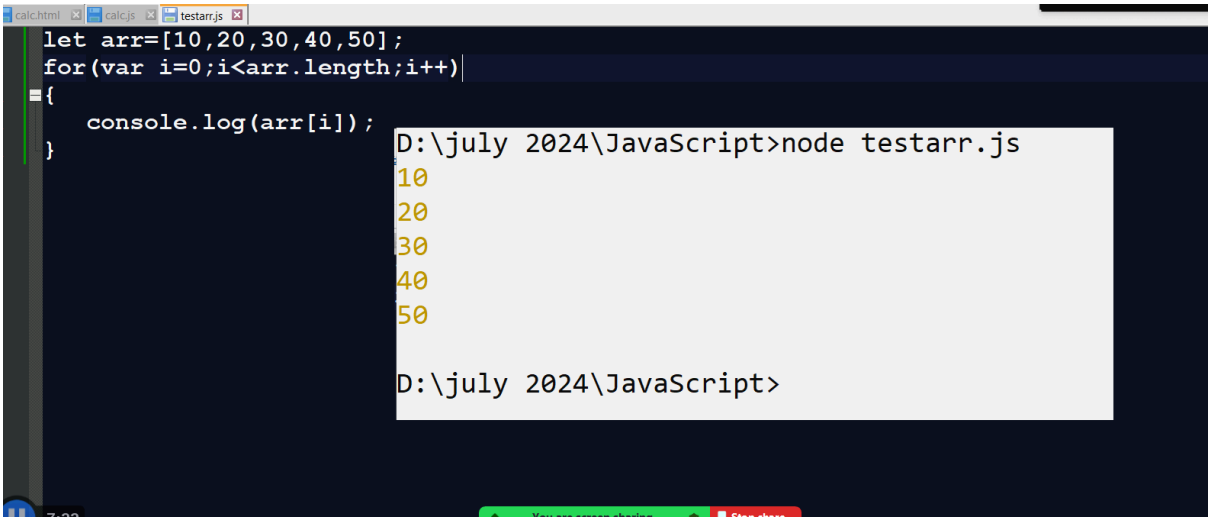
Array is a collection of similar type of data
Note: if we think about JavaScript we can store different types of data because JavaScriptis dynamic type language.

How to work with array in JavaScript or how to create array in JavaScript

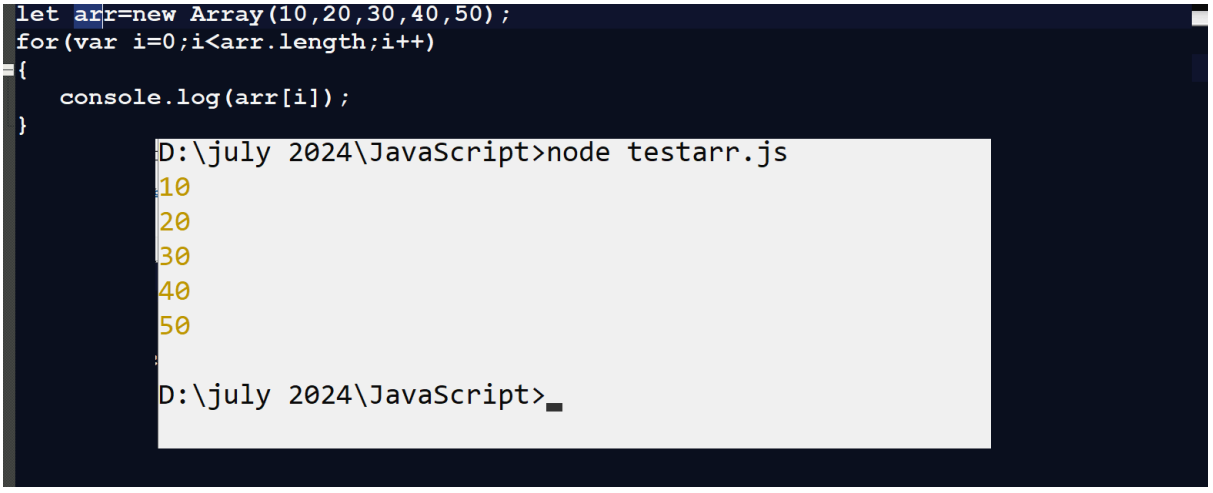
---

There are two ways to use array in javascript

1. By using []



2. By creating object of Array class



Methods of Array in JavaScript

---

toString(): this is used for converting your array in the string format.

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

let str=a.toString();

console.log("Data of array using  string  "+str);
console.log("type of array after converting  "+typeof(str));
```

Output

```
D:\july 2024\JavaScript>node testarr.js
Data of array using  string  10,20,30,40,50
type of array after converting  string

D:\july 2024\JavaScript>
```

at(index): this method returns data from array using specified index and if index not found return undefined.

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

let val= a.at(1);
    20
console.log(val);
```

0	1	2	3	4
10	20	30	40	50

```
D:\july 2024\JavaScript>node testarr.js
20

D:\july 2024\JavaScript>
```

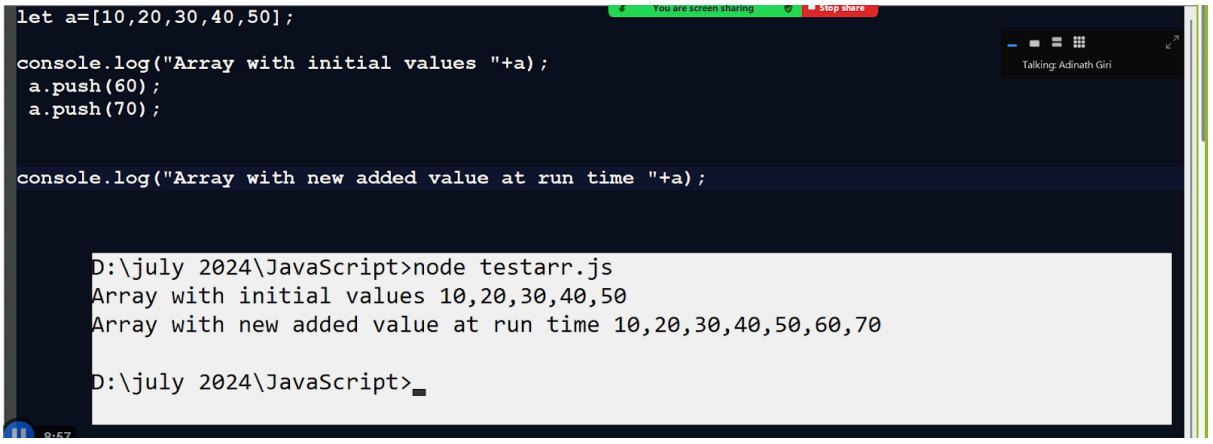
join(): this method is used for joining all array elements into string. It behave like as toString() method we can specify separator if we use join() method

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

let val= a.join("*");

console.log(val);
```

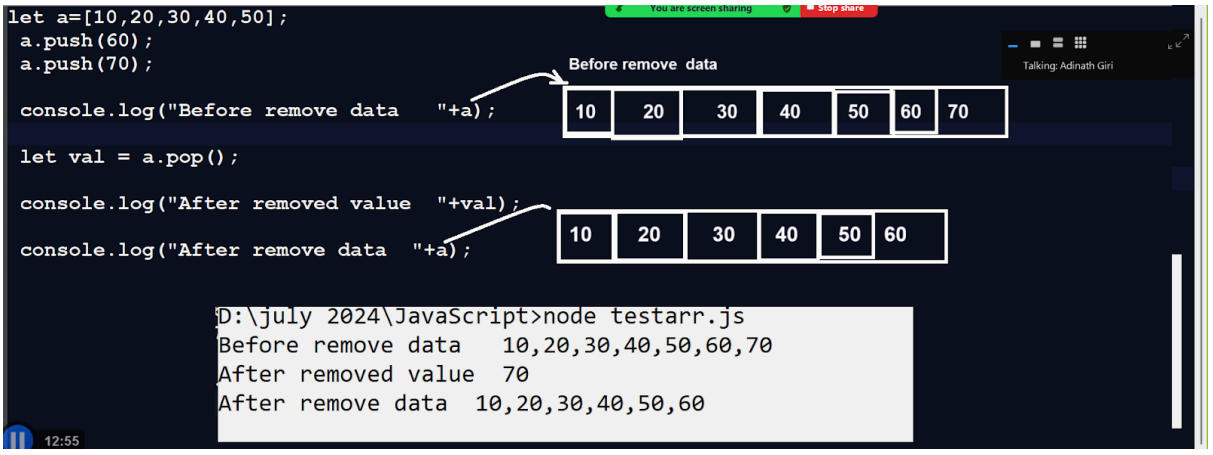
Note: we use * in every element as separate using join method

```
D:\july 2024\JavaScript>node testarr.js
10*20*30*40*50

D:\july 2024\JavaScript>
```

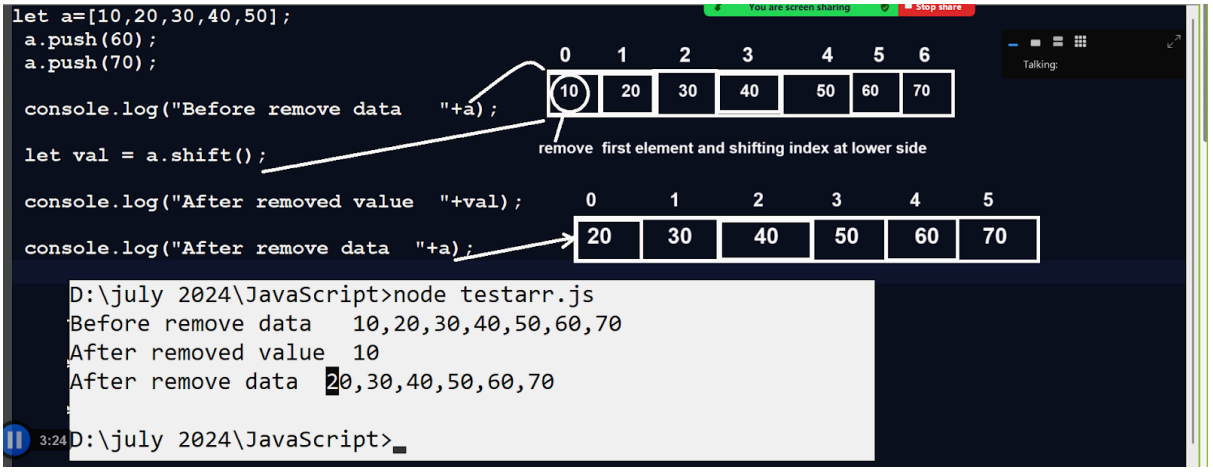
push(): this method helps us add new data in an array at program run time.



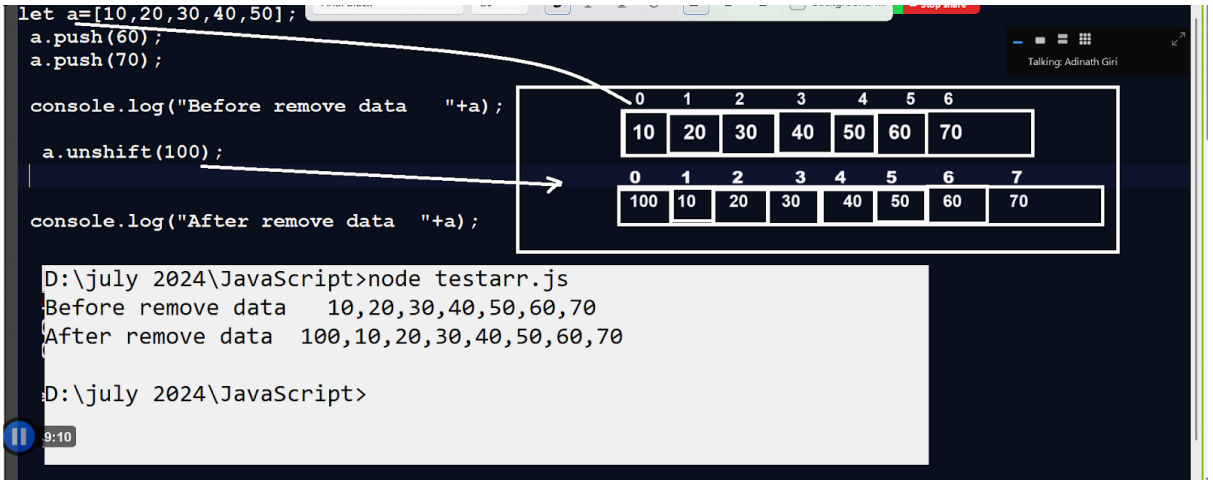
pop(): this method can remove the last element from array



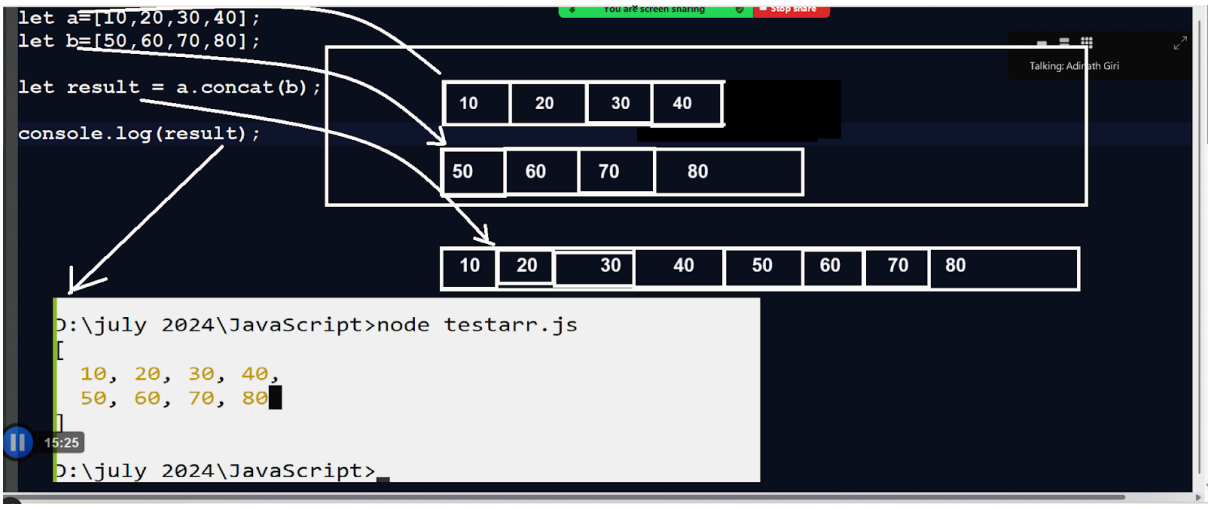
shift(): this method help us to remove the first array element and shift all other elements to lower index



unshift(): the unshift method adds a new element to an array (at the beginning) and unshifts older elements.



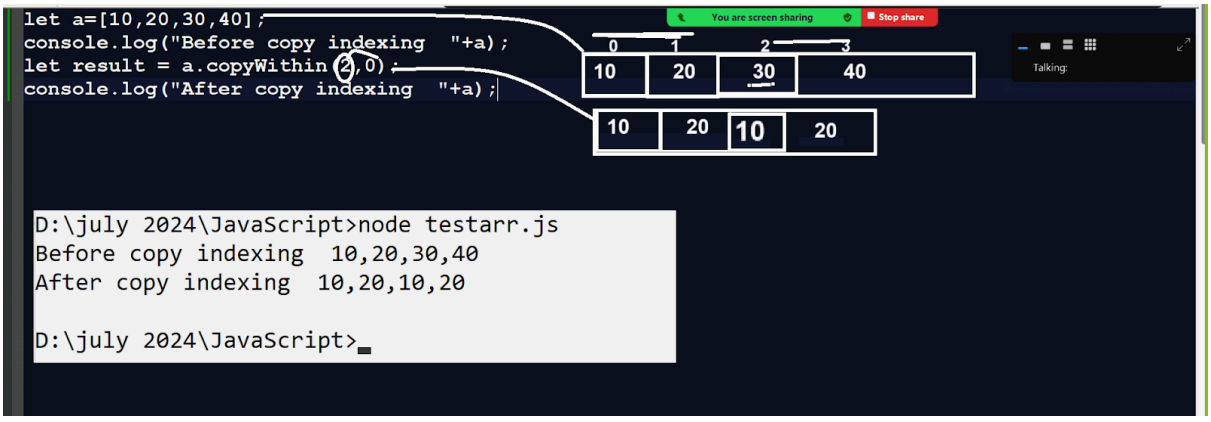
concat(): this method help us to concat the two array with each other and generate new third array



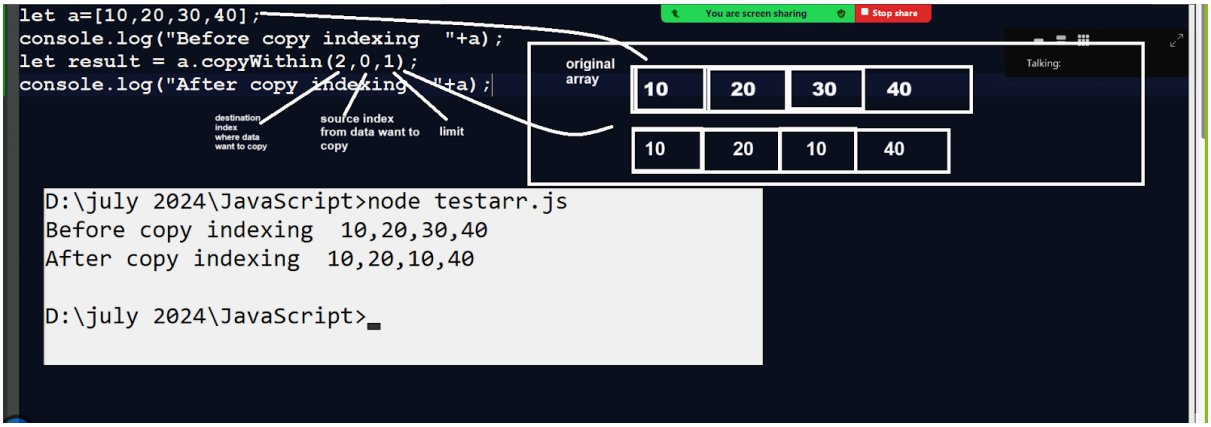
Note: you can pass an infinite array as a parameter in the concat method.

Note: you can also merge single value in array using concat() method

copyWithin(): this method can copy array elements to another position in an array.

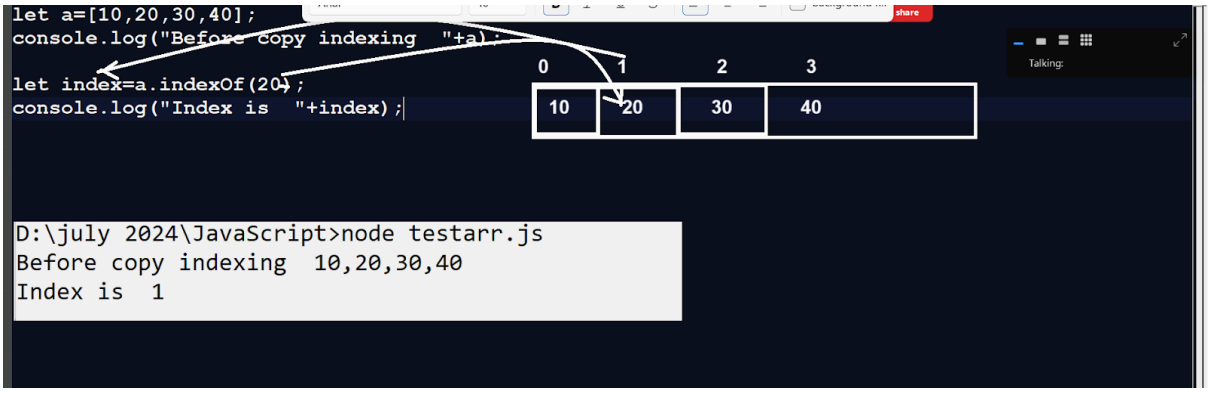


Or

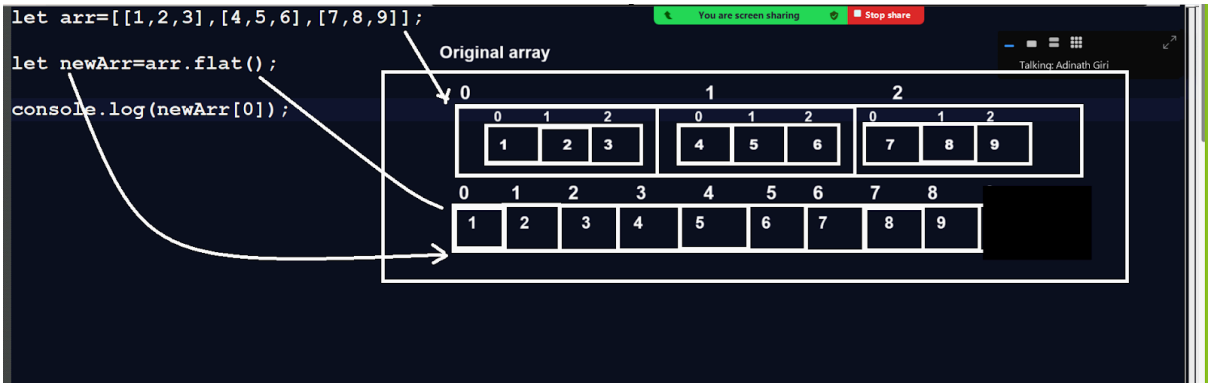


i**ndexOf():** this method helps us to search element from array and if element found return its index otherwise return -1.

Example:



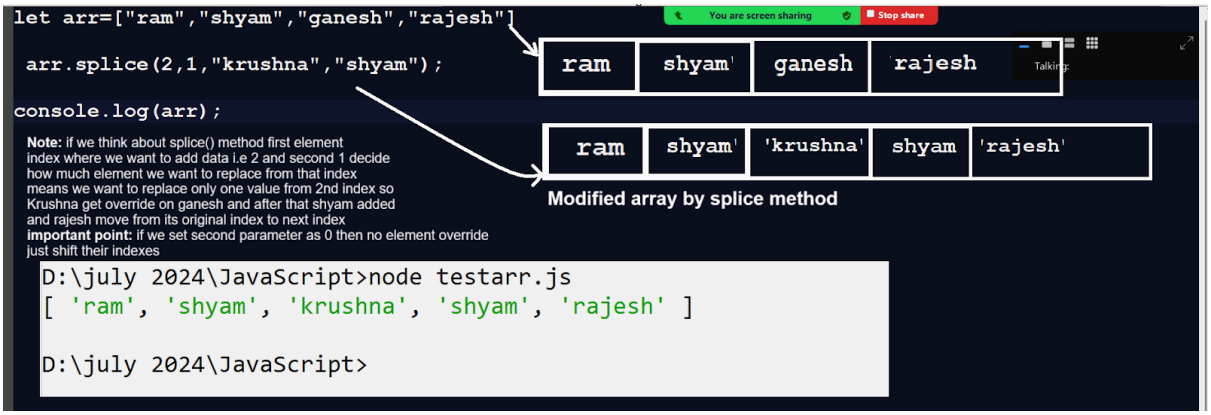
flat(): create new array with sub array elements concatenated a specified depth



flatMap(): this method can convert sub array into single dimension array and perform operation on every element means operation with every element in array and return new array as result

splice(): this method can add new elements in array or add new items in array

Syntax: splice(index, count,...values);



Array Iteration function

---

1. forEach()
2. map()
3. filter()
4. reduce()
5. every()
6. some()
7. keys()
8. entries()
9. spread()
10. with()

forEach() method

The forEach() function method calls a function (a callback function) once for each and every array elements means forEach function help us to fetch data from array

Syntax: arrayvariable.forEach(functionname);
function functionname(value,index,array){
    //write here logis
}

```
const nums=[10,20,30,40,50];
nums.forEach(show);
function show(value,index,arr)
{
   console.log(index +" -----  > "+value);
}
```

```
D:\july 2024\JavaScript>node forapp.js
0 -----  > 10
1 -----  > 20
2 -----  > 30
3 -----  > 40
4 -----  > 50

D:\july 2024\JavaScript>
```

Or

```
const nums=[10,20,30,40,50];
nums.forEach(function(value,index,arr){
    console.log(value+"----->"+index);
});
```

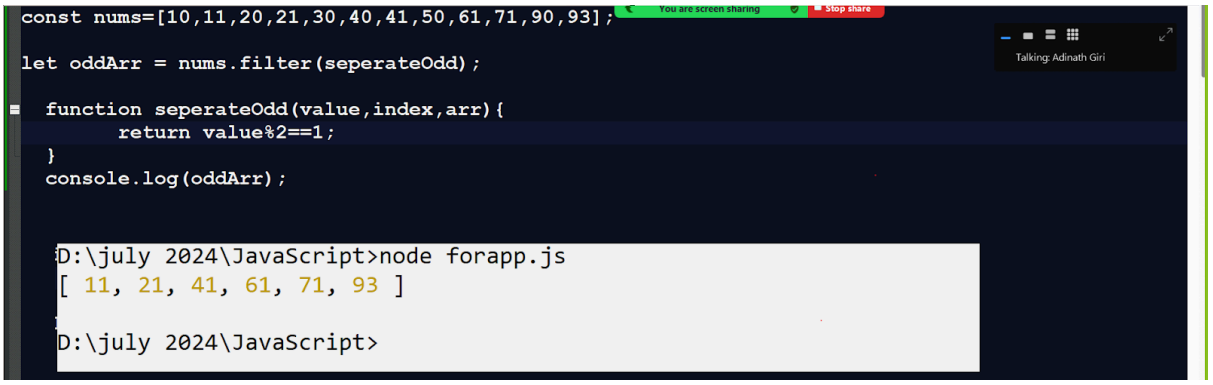
```
D:\july 2024\JavaScript>node forapp.js
10----->0
20----->1
30----->2
40----->3
50----->4

D:\july 2024\JavaScript>
```

map(): the map function creates a new array by performing a function/some operation on every element in array and map function does not change original array

Example: Suppose we have array with values 10 20 30 40 50 and we want to calculate square of every value in array and store in another array

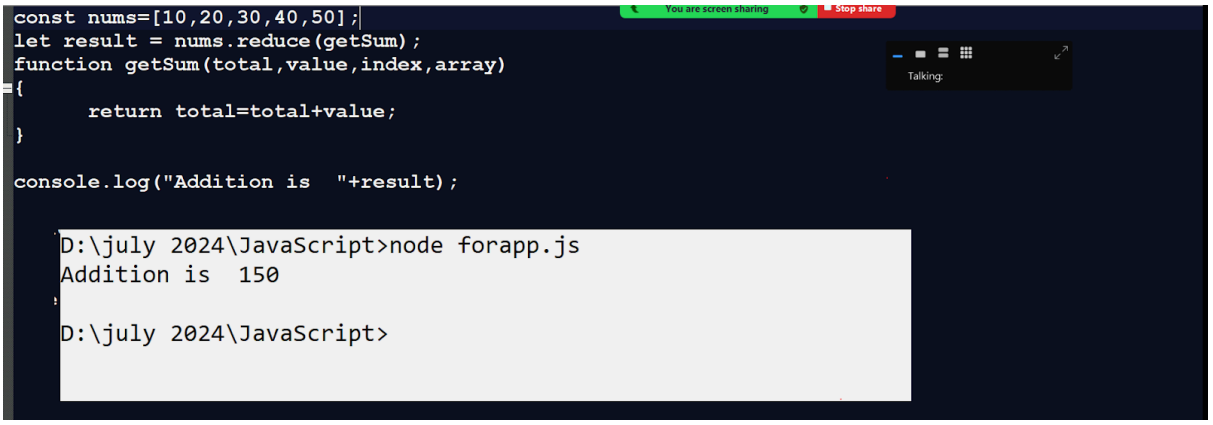
```
const nums=[10,20,30,40,50];

let dummyArr = nums.map(square);
function square(value,index,nums)
{
        return value*value;
}
console.log(dummyArr);
```

Filter function : filter function is used for separate data by using some condition or filter data by using some condition and store in a new array.

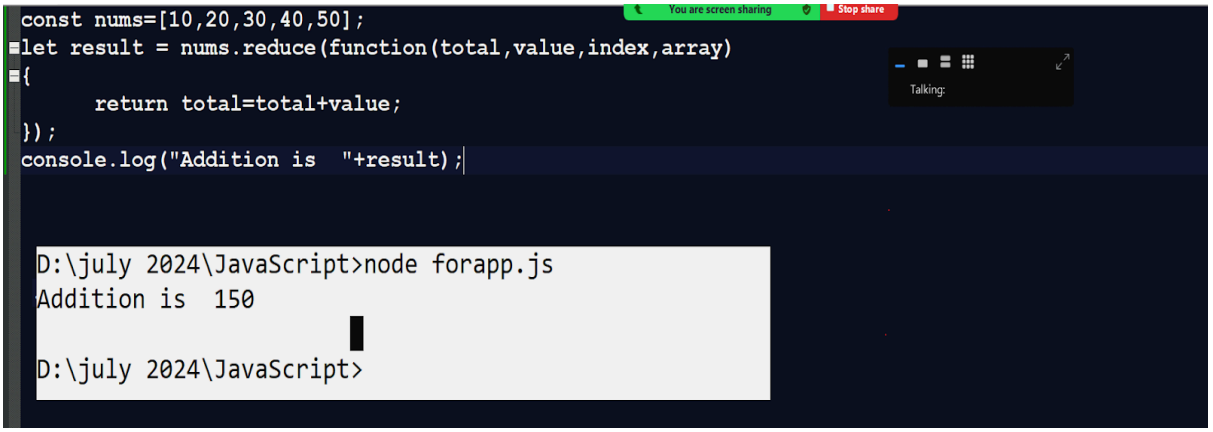


reduce(): reduce() function help us to perform operation all array elements and return single value as result

Example:suppose consider we have array with five values and calculate its sum



Or

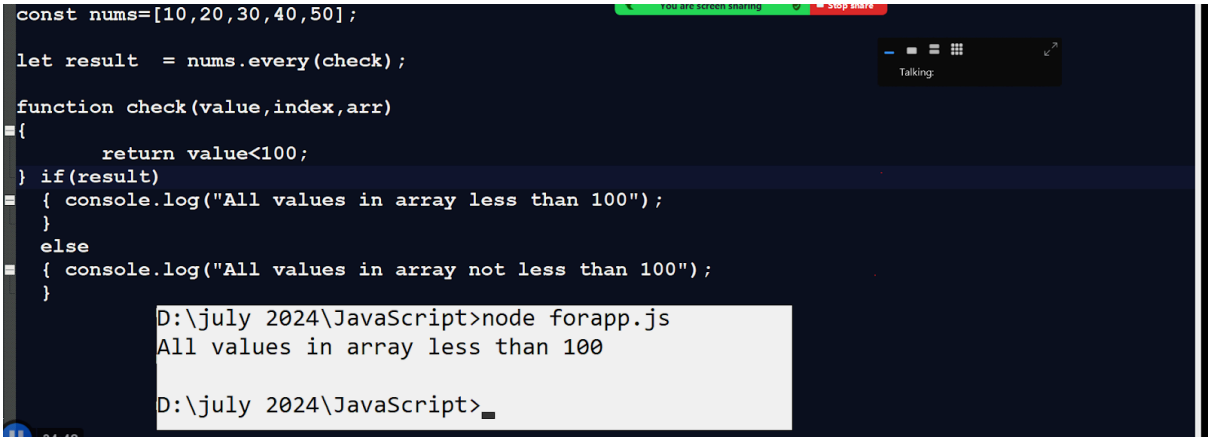


Example: Suppose we have an array with 5 values and perform multiplication on every element in the array.

```
const nums=[10,20,30,40,50];
let result = nums.reduce(function(total=1,value,index,array)
{
        return total=total*value;
});
console.log("Addition is  "+result);
```

every() function : every function used to apply condition or check condition with every element in array and if condition satisfies with all elements then return true otherwise return false.

Example: suppose consider we have values in array and we want to check every value in array must be less than 100



some():some function is used to apply conditions on every element in an array and if any one condition is true return true if all conditions are false return false.

```
const nums=[10,200,300,400,500];

let result  = nums.some(check);
```

```
function check(value,index,arr)
{
        return value<100;
} if(result)
  { console.log("any one value less than 100");
  }
  else
  { console.log("all values greater than 100");
  }
```

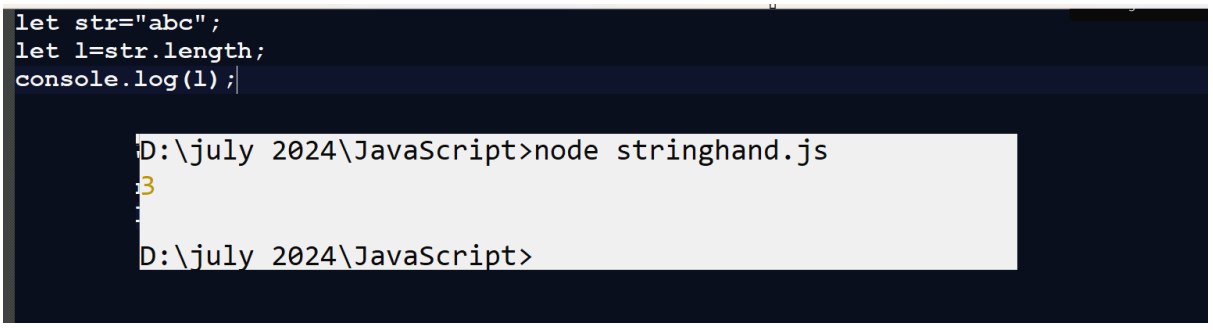
String Handling using JavaScript

String is an immutable object in JavaScript we can say immutable means we cannot modify value once we initialize it called as immutable. Means if we perform any operation on string then string generates new value every time or new string object every time.

Syntax: let var=”value”;

Methods of string class

String length : this property help us to return the length of string



String charAt(index): this method helps us return character from string using some specified index.

```
let str="abc";
let l=str.length;
for(var i=0;i<l;i++)
{
      let  singleChar=str.charAt(i);
      console.log(i+"---->"+singleChar);
}
```

```
D:\july 2024\JavaScript>node stringhand.js
0---->a
1---->b
2---->c

D:\july 2024\JavaScript>
```

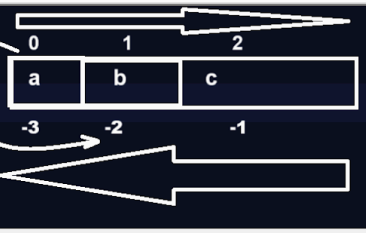
charCodeAt(index): this method returns ascii code of character using its index.

```
let str="abc";
let l=str.length;
for(var i=0;i<l;i++)
{
      let  asciiCode=str.charCodeAt(i);
      console.log(i+"---->"+asciiCode);
}
```

```
D:\july 2024\JavaScript>node stringhand.js
0---->97
1---->98
2---->99

D:\july 2024\JavaScript>
```

at(): this method returns a character from a string using its index.

```
let str="abc";
let schar=str.at(-2);
console.log(schar);
```



Note: at() method can return character using positive index as well as negative index but charAt() method only return character using positive index only.

```
D:\july 2024\JavaScript>node stringhand.js
b
```

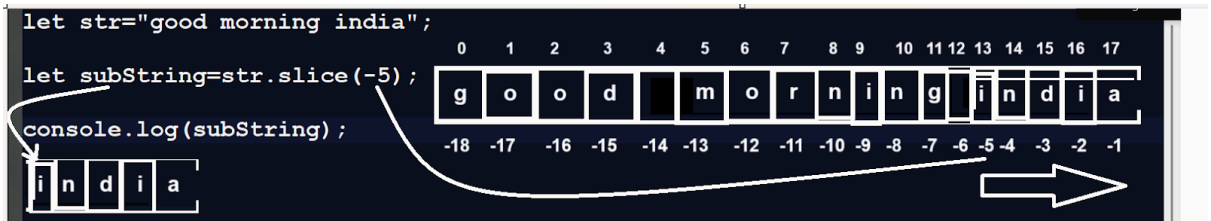
substring(start,end): extract a string data between two specified indexes and create a new string object at the left hand side.

```
let str="good morning india";

let subString=str.substring(5,12);

console.log(subString);
```

```
D:\july 2024\JavaScript>node stringhand.js
morning

D:\july 2024\JavaScript>
```

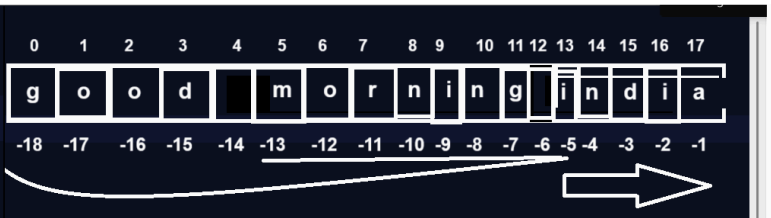
slice(start,end): this method can extract string portion like as substring not mandatory to provide second parameter

```
let str="good morning india";

let subString=str.slice(-5);

console.log(subString);
```




```
D:\july 2024\JavaScript>node stringhand.js
india

D:\july 2024\JavaScript>
```

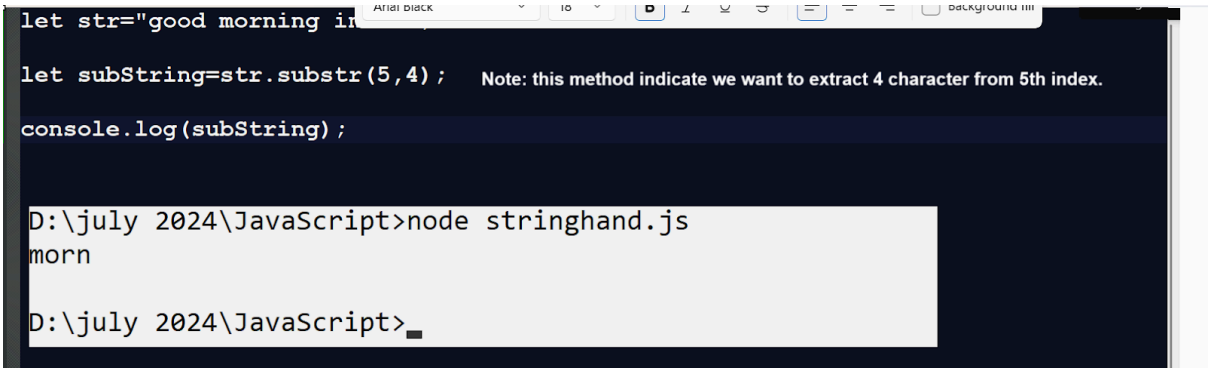
```
let str="good morning india";

let subString=str.slice(-13,-5);

console.log(subString);
```




```
D:\july 2024\JavaScript>node stringhand.js
morning
```

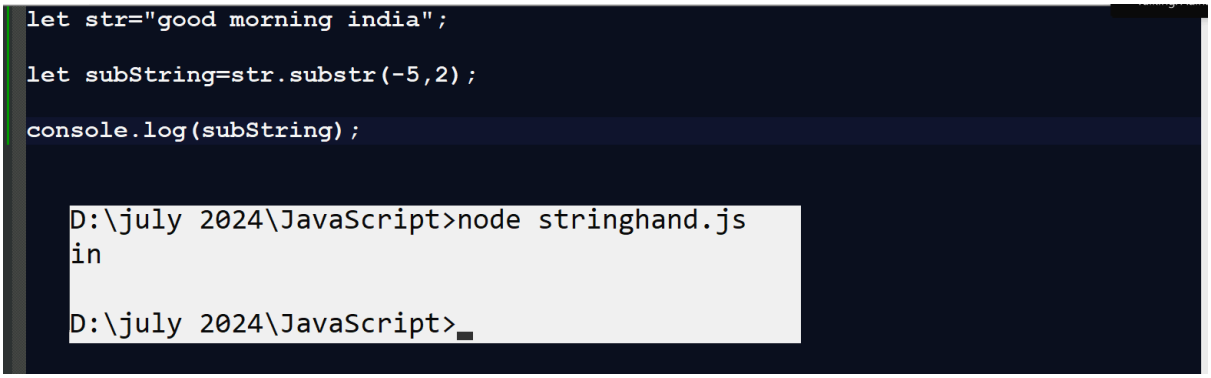
Note: substring() not support negative index and slice() support to negative index and both method not required or mandatory to use second parameter when method not use second parameter then we get substring from start index to length-1 of string

substr(index,length): if we substr function then second parameter is not end index it is length or decide how much characters from string



Note: second parameter can skip from substr means it is not mandatory to give second parameter and when we not provide second parameter then substr extract all string data from index to length-1

Note:if the first parameter is negative then position count from the end of the string or right hand side of string



```
let str="good morning india";

let subString=str.substr(-5,2);

console.log(subString);
```

```
D:\july 2024\JavaScript>node stringhand.js
in

D:\july 2024\JavaScript>
```

toUpperCase(): this method help us convert lower case string to upper case string

```
let str="good morning india";

let str1=str.toUpperCase();

 console.log(str1);
```

```
D:\july 2024\JavaScript>node stringhand.js
GOOD MORNING INDIA

D:\july 2024\JavaScript>
```

toLowerCase(): this method can help us to convert uppercase to lower case string

```
let str="GOOD MORNING INDIA";

let str1=str.toLowerCase();

 console.log(str1);
```

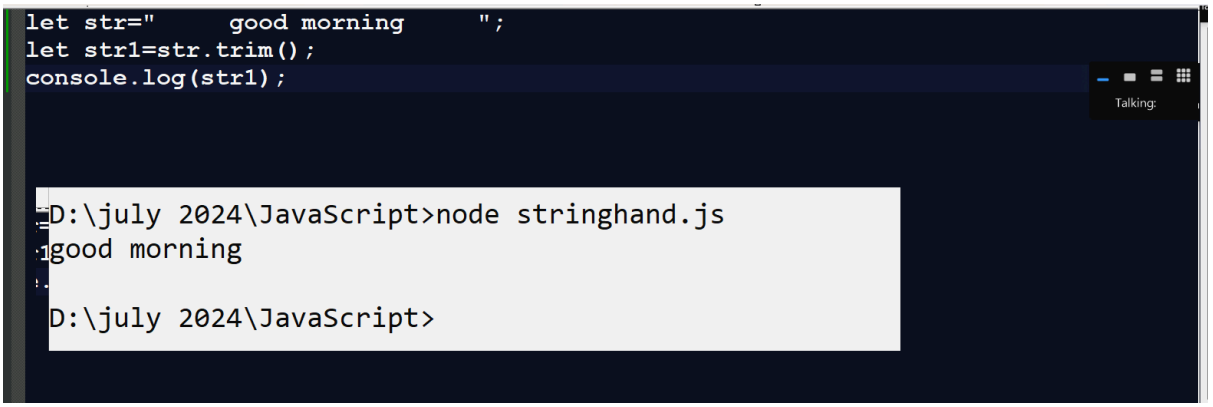
```
D:\july 2024\JavaScript>node stringhand.js
good morning india

D:\july 2024\JavaScript>
```

concat(): this method can combine two or more strings and generate new strings and return at the left hand side.

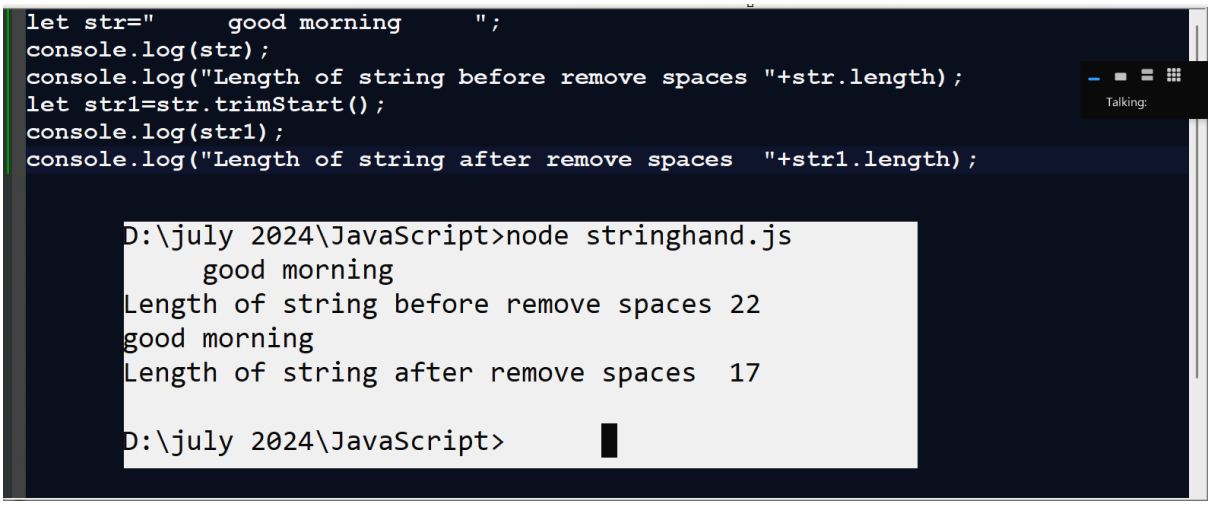
```
let s="abc ";
let s1="mno ";
let s2="stv ";
let s5="xyz ";
let s6="kst ";


let s4 =s.concat(s1,s2,s5,s6);

console.log(s4);
```

```
D:\july 2024\JavaScript>node stringhand.js
abc mno stv xyz kst

D:\july 2024\JavaScript>
```

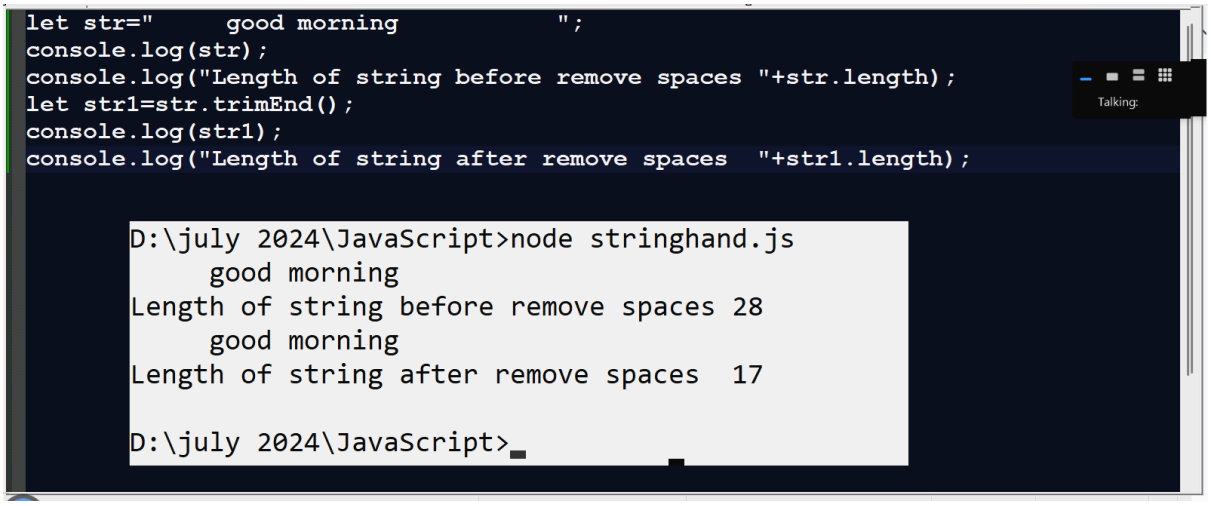
trim():remove the white spaces from the starting and ending of the string.



trimStart(): this method can remove white spaces at beginning of string



trimEnd(): this method can remove white spaces at ending of string



padStart(): we add some data in string the logic of padStart() is if we give some value as first parameter then value must be greater than length of string so padStart() perform subtraction operation between first value and length of string means first value- string length = remain value means add second parameter starting string according to remaining value.

split(): split() method is used for split the string by some specified character and return string array.

replace(): method can change content of string and create new string not perform change in original string

Syntax: replace(“oldstring”,”newstring”);

Example:

```
let str="Hello JavaScript";
console.log("Before replace "+str);

let str1=str.replace("Hello","hi");

console.log("After Replace  "+str1);
```

```
D:\july 2024\JavaScript>node splittest.js
Before replace Hello JavaScript
After Replace  hi JavaScript

D:\july 2024\JavaScript>
```

String searching methods

---

indexOf(): this method help us for search string data from original string and return its index and data not found return -1

```
let str="good morning india";

let index=str.indexOf("morning");

if(index!=-1)
{  console.log("Data found  "+index);
}
else
{  console.log("Data not found  "+index);
}
```

```
D:\july 2024\JavaScript>node splittest.js
Data found  5
```

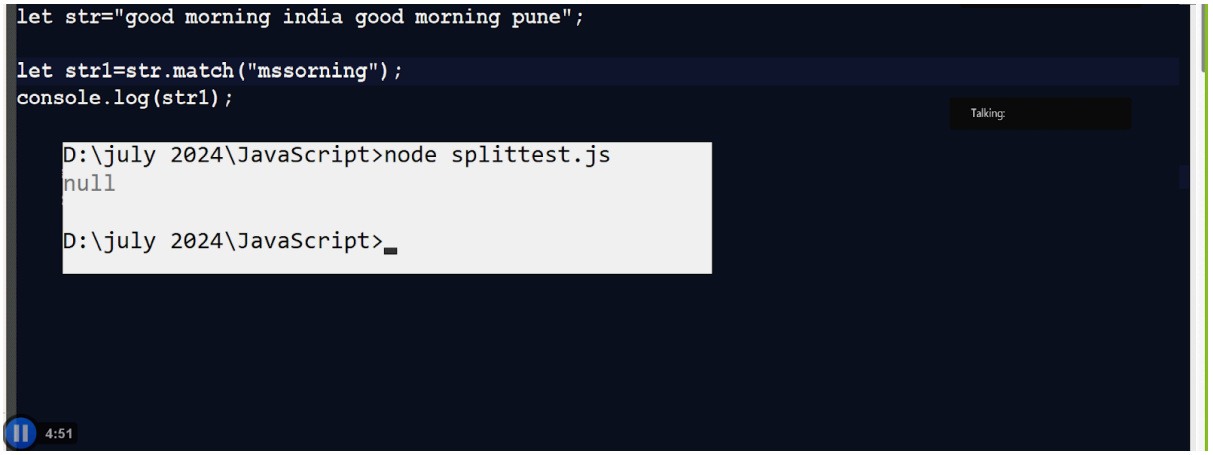
lastIndexOf(): this method help return last occurrence element index and if not found return -1

```
let str="good morning india good morning pune";

let index=str.lastIndexOf("morning");

if(index!=-1)
{  console.log("Data found  "+index);
}
else
{  console.log("Data not found  "+index);
}
```

```
D:\july 2024\JavaScript>node splittest.js
Data found  24

D:\july 2024\JavaScript>
```

search(): this method can search data using string as well as can search data using regular expression

match(): return an array containing the result of matching a string against string (for regular expression)
This method returns null if data is not found.



Validation using JavaScript

Validation means cross verify data before enter or submit in application called as validation

Q. Why we need to perform validation or what is importance of validation

---

1. Avoid wrong input or inappropriate input or cross verify input value
2. To avoid different types of attack on web application

Example: SQL Injection Attack , Cross Site Scripting Attack , CSRF Attack (Cross site Request Forgery) etc

We want to implement following types of validation

---

1. Name validation
2. Mobile number validation
3. Email validation
4. Strong password validation
5. Confirm password validation
6. Adhar Number Validation
7. Pan Card Validation
8. Driving Licences validation
9. Passport number validation
10. Age validation

Etc

Now we want to implement name validation

Rules of name validation

---

1. Digit not allowed in textbox
2. Special symbol not allowed except space
3. First Letter must be capital

Steps to implement above scenario

---

1. **Design web page with one textbox**

```
<html>
 <head>
   <title>name validation page</title>
</head>
<body>
<input type='text' name='name' value=''
style='width:400px;height:40px;'/>
```

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

2. Perform event on textbox and access value of textbox in JavaScript function

```
<html>
 <head>
  <title>name validation page</title>
  <script type='text/javascript'>

     function acceptName(str) //abc123mno
         {

            let flag=true;
                for(var i=0; i<str.length; i++)
                {       //1
                    let ch =str.charCodeAt(i);
                        if(!((ch>=65 && ch<=90) || (ch>=97 &&
ch<=122) || ch==32)){
                            flag=false;
                                break;
                        }
                }
                if(flag)
                {   let spanEle=document.getElementById("s");
                   spanEle.style.color="white";
                       spanEle.innerHTML="";
                }
                else
                {  let spanEle=document.getElementById("s");
                   spanEle.style.color="red";
                       spanEle.innerHTML="invalid name "+flag;
                }
         }
```

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

Mobile Number Validation

```
<html>
 <head>
  <title>name validation page</title>
  <script type='text/javascript'>

    function acceptName(str) //9999
        {

          let flag=true;
              for(var i=0; i<str.length; i++)
              {       //1
                   let ch =str.charCodeAt(i);
                        if(!(ch>=48 && ch<=57) || str.length!=10)
                        { flag=false;
                          break;
                        }
              }
              if(flag)
              {   let spanEle=document.getElementById("s");
                  spanEle.style.color="white";
                       spanEle.innerHTML="";
              }
              else
              {  let spanEle=document.getElementById("s");
                 spanEle.style.color="red";
```

```
                    spanEle.innerHTML="invalid name "+flag;
                }
            }


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

Email validation

Rules of email validation

---

1. @ must be present in email but not more than once
2. Dot (.) may be occur more than one time but at single dot must be after @ symbol and after dot there must be two or three letter
3. Email should not start with @
4. Capital letters should not be present in email addresses.
5. No special symbol allowed except underscore character

Example with source code

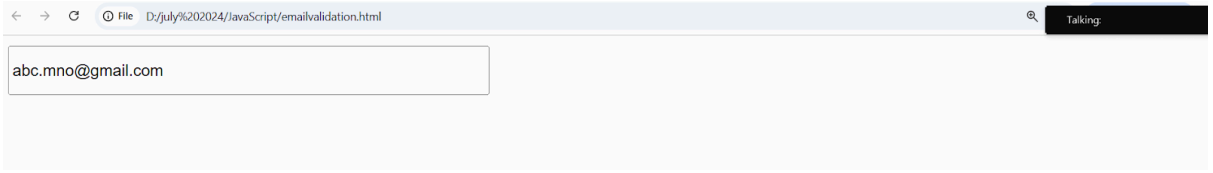
```
<html>
 <head>
  <title>email validation app</title>
  <script type='text/javascript'>
    function emailValidate(str)
        {
        let spanEle=document.getElementById("s");
            let index=str.indexOf("@");
                let index1=str.lastIndexOf("@");
```

```
                if(index<=0 || index!=index1)
                { spanEle.innerHTML="Invalid email";
                  spanEle.style.color="red";
                }
                else{
                  let s="@";
                  let ascii1=s.charCodeAt(0);
                       console.log("ASCII OF @" +ascii1);
                   let strData=str.substring(index);
                        index=strData.indexOf(".");
                         if((index==(strData.length-4)) || (index
==(strData.length-3)) )
                         {spanEle.innerHTML="";
                       spanEle.style.color="red";
                         }
                      let flag=false;
                              for(var i=0; i<str.length;i++)
                              {
                                  let ascii=str.charCodeAt(i);
 if(!(ascii>=48 && ascii<=57 || ascii>=65 && ascii<=90 ||
ascii>=97&&ascii<=122 || ascii==64 ||ascii==46 || ascii==95))
                                       {  flag=true;
                                           break;
                                       }
                                      if(ascii>=65 && ascii<=90)
                                      { flag=true;
                                        break;
                                      }
                              }
                              if(flag)
                              {
                                 spanEle.innerHTML="Invalid email";
                           spanEle.style.color="red";
                              }
                              else{
                                spanEle.innerHTML="";
                           spanEle.style.color="red";
```

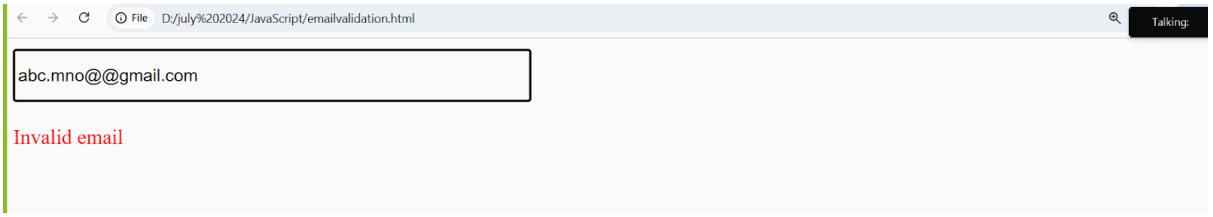
```
                    }

                }

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

Output

Valid email



Invalid email

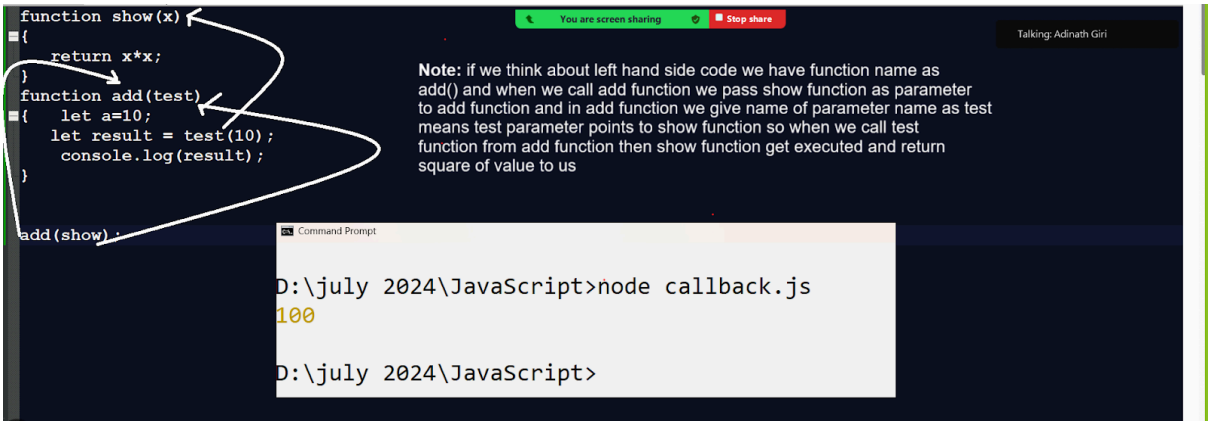


Call back function

Call back function means function passed as an argument to another function called as call back function

This technique allow us call one function from another function

Means callback function execute after his parameter function



setInterval and setTimeout

---

setInterval() function : setInterval is function from JavaScript Which is used for execute some task repeat again and again using some specified time period called as setInterval

Syntax:
setInterval(function,time limit):

There are two parameter in setInterval function

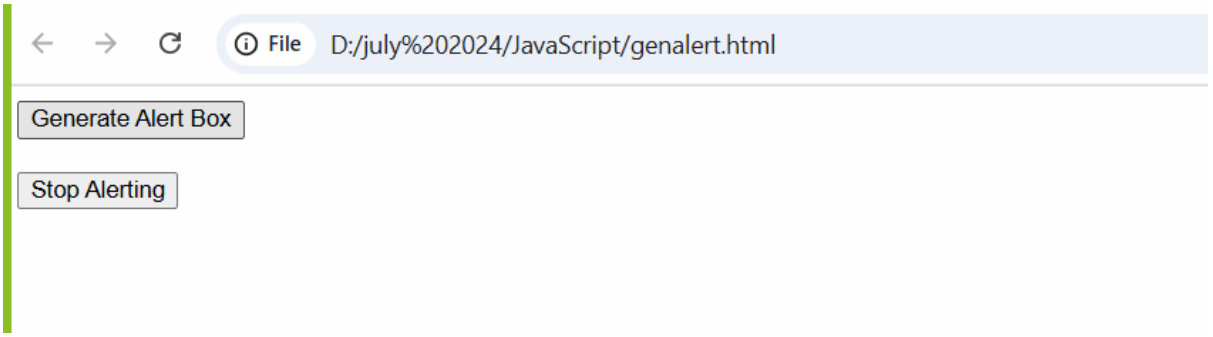
Function : here we need pass user defined where we normally write logic or we can use anonymous function means function contain logic which we want to execute after some specified time period
Time limit : this parameter decide cycle of re-rexecution means setInterval call user defined function according to time values pass in it

Example with source code

```
<html>
  <head>
    <title>example of setInterval function</title>
        <script type='text/javascript'>
          let result;
          function genAlert()
          {
```

```
            result=setInterval(show,5000);
          }

              function show()
              {  alert("i am alert box");
              }

              function stop()
              {  alert(result);
                 clearInterval(result);
              }
        </script>
  </head>
  <body>
   <input type='button' name='s' value='Generate Alert Box'
onclick='genAlert()'/>
   <br><br>
    <input type='button' name='s' value='Stop Alerting'  onclick='stop()'/>

  </body>
</html>
```

Output



Example: we want to create digital clock using setInternval()

```
<html>
  <head>
    <title>example of setInterval function</title>
        <script type='text/javascript'>
```

```
        function digiClock()
        {
          setInterval(function(){
                let d = new Date();
                    let time=d.toLocaleTimeString();
                    document.getElementById("s").innerHTML=""+time;
       },1000);
          }

       </script>
  </head>
  <body>
   <input type='button' name='s' value='Generate Alert Box'
onclick='digiClock()'/>
   <br><br>
       <span id="s"></span>
  </body>
</html>
```

setTimeout() function

---

setTimeout() is used to execute the task or logic after some specified time period but execute logic only once.

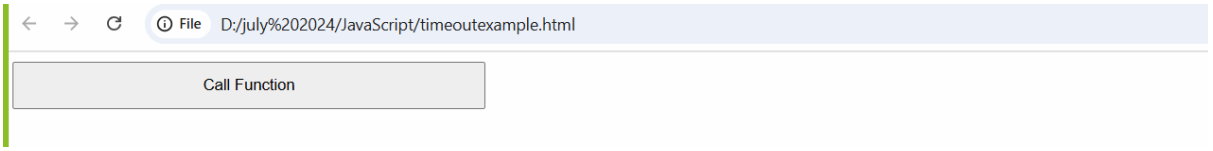
```
Example:
setTimeout(show,5000);
function show()
{  alert("Good");
}
or
Example
setTimeout(function(){
   alert("Good");
},5000);
```

Example with source code

```
<html>
 <head>
   <title>example of set time out</title>
```

```
  <script type='text/javascript'>
    function test()
        { setTimeout(show,5000);
        }
        function show()
        {alert("good");
        }
  </script>
 </head>
 <body>

  <input type='button' name='s' value='Call Function'
  style='width:400px;height:40px;' onclick='test()'/>
 </body>
</html>
```

Output



Advanced JavaScript & EcmaScript 6 Feature

---

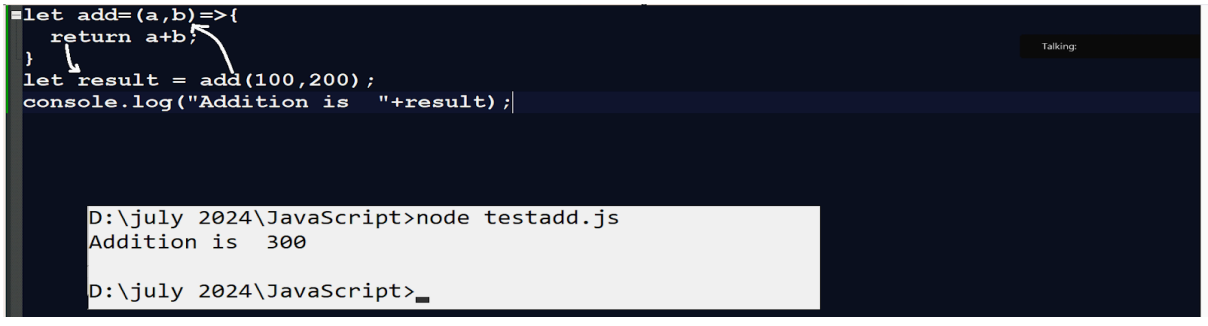
1. Arrow function
2. Rest operator
3. Spread operator & Array destructing
4. Template string
5. Call back
6. JSON & REST API
7. Async and await
8. Promises
9. fetch() function
10. AJAX
11. Event loop
12. Javascript generators
13. Closure

14. JavaScript memoization
15. JavaScript currying
16. Local storage and session storage
17. Regular expression
18. Mini project on javascript

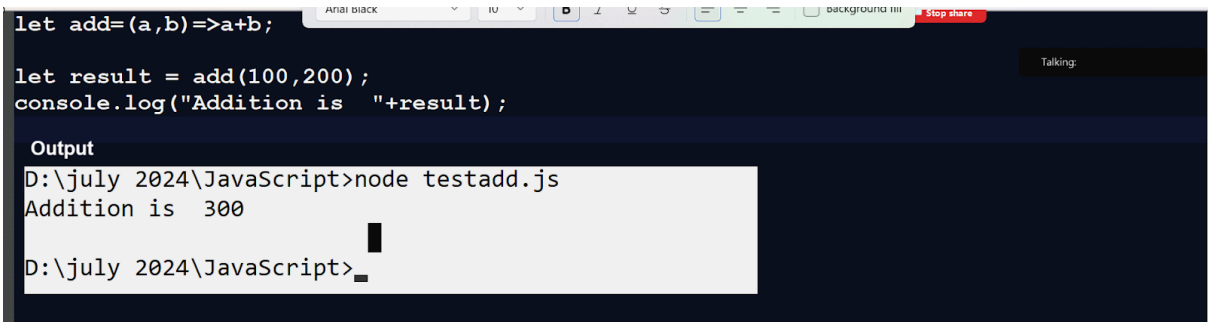
Arrow function: arrow function is a facility to execute the small logics like as lambda expression in JAVA.
Basically it is a function writing technique in JavaScript.

```
let functionname=(parameter)=>{
   write here your logics
}
or
const functionname=(parameter)=>{
    write here your logics
}
or
var functionname=(parameter)=>{
   write here your logics
}
```

Example: WAP to create a function name as add() with two parameters and calculate its addition.



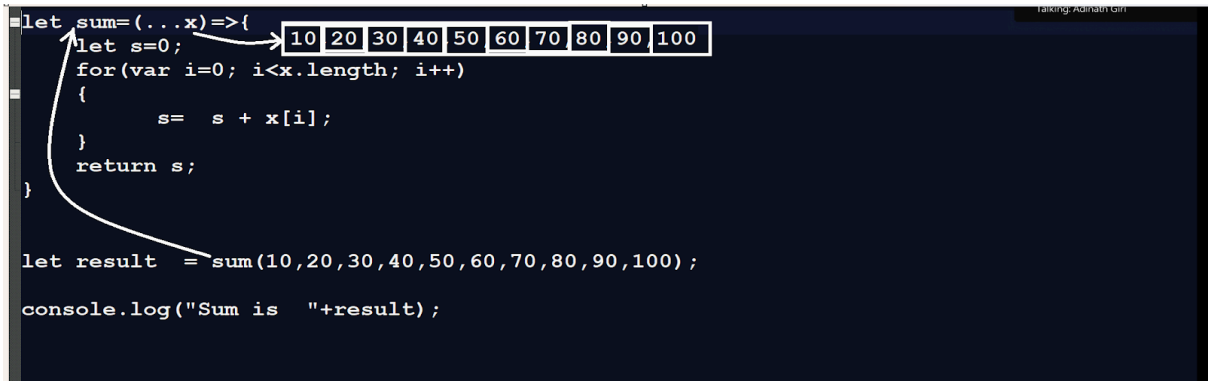
Or



Example: WAP to create function name as table() and print table of specified number.

Rest Operator:

Rest Operator is denoted by … in JavaScript and which is used for accept infinite number of parameter as function parameter like as variable argument in JAVA



Spread operator

Spread operator is used for array destructing purpose

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

let [b,c,d,e,f]=[...a];

console.log("B is   "+b);
console.log("C is   "+c);
console.log("D is   "+d);
console.log("E is   "+e);
console.log("F is   "+f);
```

```
D:\july 2024\JavaScript>node testarrdes.js
B is   10
C is   20
D is   30
E is   40
F is   50
```

Template string

Template string is a concept in JavaScript which is denoted by backtick operator (`) and which is used for avoid string concatenation operator Means some time we want to add in string we required perform concatenation operator by using + symbol so better we can use template string for avoid this problem and we can write the ${expression} in template string for passing run time data

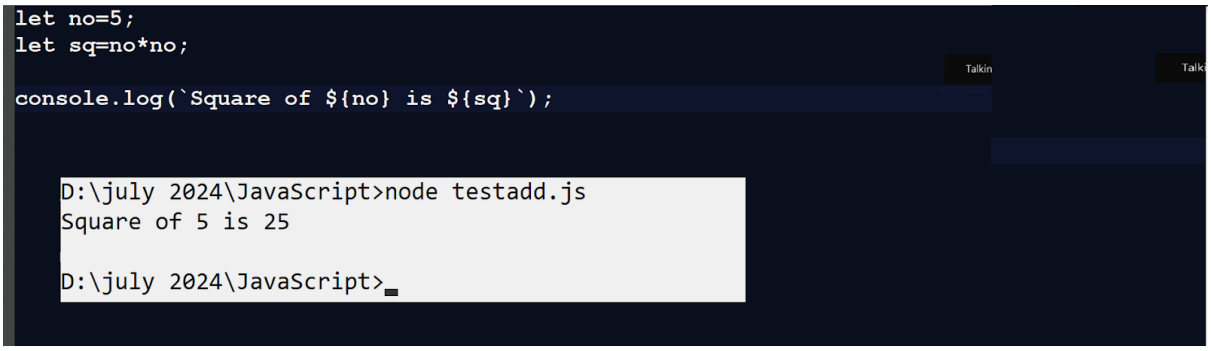
```
let a=100;
let b=200;

let c= a+b;

console.log(`${a} + ${b} = ${c}`);
```

Output

```
D:\july 2024\JavaScript>node testadd.js
100 + 200 = 300

D:\july 2024\JavaScript>
```

Example: WAP to declare variable name as no=5 and calculate its square and display message like as

Output: Square of 5 is 25



API

Q. What is an API?

API stands for application programming interface before API we want to discuss about the web services .

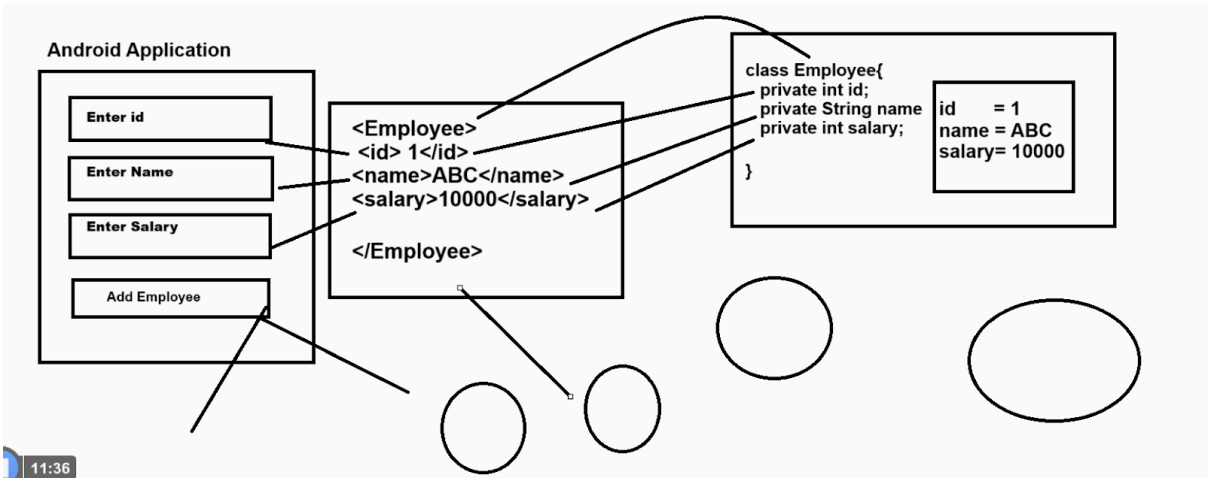
Q. What are web services?

Web services are the type of internet software system that enable to provide communication between two application via internet may be developed using same tech stack or different tech stack means the feature of web services is language interoperability means web service able to share data between two application may be develop using same technology or different technology.

Types of web services?

SOAP(Simple Object Access Protocol)

A protocol for exchanging structured information in a platform independent way and use XML for message format and operate or use standard web protocols like as HTTP , SMTP etc
SOAP has limitation it require strict compliance with standards , making it heavier but very secure
Typically used in enterprise application



REST : (Representational State Transfer)

REST is not protocol it is a design pattern and use by using standard http methods like as GET, POST,PUT,DELETE etc
REST is very light weight as compared with SOAP and modern approach to compare SOAP.
If we use REST then we can send requests using any format like JSON,XML ,String etc as well as we can get responses using string,JSON objects or maybe using XML.
It is faster and easier to implement as compared to SOAP so it is very popular for developing mobile applications as well as web applications or communicating with IOT Devices.
It is stateless meaning is each request from the client to server is independent with no need for storing information about previous request

Q. What is the benefit of web services?

---

1. **Interoperability :** this feature of web services able to provide communication between two systems may be using the same platform or different platform as well as provide communication applications may be developed in the same technology or may be different technology or also able to provide communication between IOT devices and software applications.

2. **Scalability :** They allow systems to scale because you can connect different services with different applications using a different platform and enabling ne functionalities

3. **Modularity :** web services can be used to expose specific functionality or data that can be used by other services or applications.
4. **Standardised communication:** the protocolo and formats like HTTP methods or XML and JSON) used by web services to provide communication between two systems or applications.

Feature of REST API?

---

1. **Stateless :** Each request from client to server must be contain all information the server needs to understand the process the request and server does not store request information so previous request information is not present at server side

2. **Client server architecture :** the client i.e browser or may be any other application like mobile app and server i.e may be apache or database etc operate independently means client can send request to server and server send response to client using REST API.

3. **Uniform interface :** REST API use a standard set of conversions for interaction typically using HTTP methods like as GET ,POST,PUT,DELETE etc
   Means if we implement the REST API using any technology we have same methods for sending request and getting response method syntax may be different dependent on the langage
4. **Representation :** when we interact with resource then REST API provide standard data representation to us like as JSON

5. **Layered system :** REST Allow for use of intermediator like as gateways or proxies between client and server for the better scalability and security

Common Http Methods in REST API

GET : get method normally use by developer for retrieve information or get response data from server
POST : send data to the server to create new resources at the server side.
PUT : update an existing resource on the server side.
DELETE : Remove resource from a server
PATCH : apply parietal modification of resource

Can we give example of REST API uses

---

1. **Social media integration :** a mobile app might use REST API for fetching user data from facebook or twitter or we can integrate social media login credentials in our website also or we can fetch user history in our application from social media using REST API.
2. **Weather App:** weather services provide REST API that all developer to integrate weather data in to their application
3. **Payment Gateway Integration:** developer can integrate payment gateway in his application for online payment by using third party payment broker company
4. **Google Map integration:** user can integrate google map in his application by using REST API.
   Etc

How to transfer data using REST API between Two Application

REST API can send data using any format like using string, object, json , xml etc but 99% people use JSON as standardised format for data sharing between two applications using REST API.

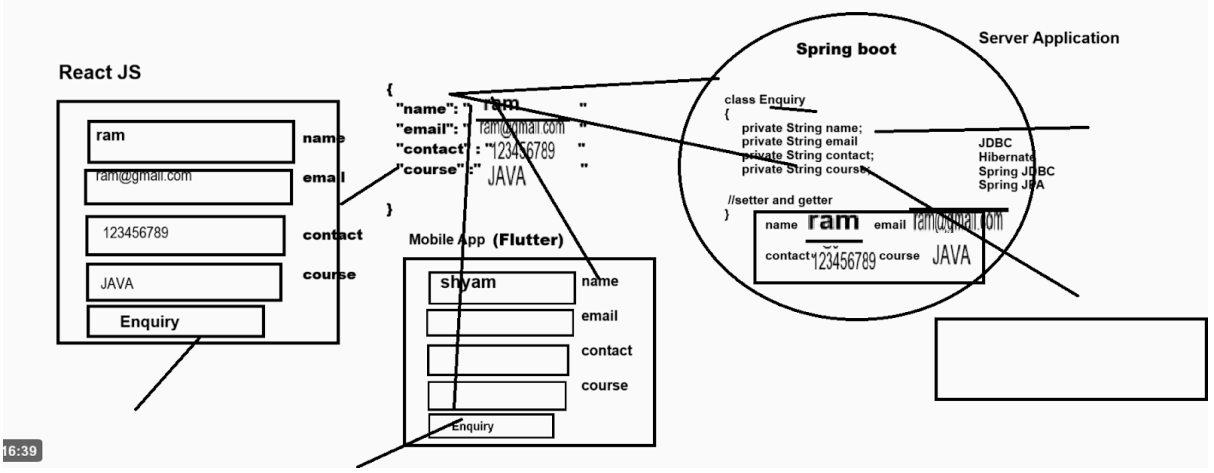
Q. What is JSON?

JSON stands for JavaScript object Notation
JSON is text format for storing and transporting data and JSON store data in the form of key and value pairs i.e name and value pair
Key is always in the form of string and value may be any kind.

Generalize format of JSON Object

```
let variablename={
                    "key/name":value
                };
```

Example:



Q. Why people use JSON format in REST API if we have some other format also like as XML or string etc

---

1. **Human Readable :** JSON format is text based format and easy for humans to read and write
2. **Light weighted:** JSON is compact compared to formats like XML , making it more efficient for transmitting data over networks. It has minimal overhead , which is important for

applications that need to optimize for speed and bandwidth usage.

3. **Language independency:** JSON almost work with all languages like as we can use JSON in JAVA, Python,.NET etc

4. **Easy to parse and generate :** All programming language offer built in support or external library for parsing JSON data into native object means according to that language Example: JSON.parse(),JSON.stringify() etc
5. **Structured data representation :** JSON provide standard structure of data representation as well as allow nested object structures for data representation like as we can integrate array, objects or any kind of data in json object

```
{
  "name":"ABC",
  "id":1,
  "marks":[60,60,60,60,60,60],

  "personalInfo": {
        "address":"PUNE",
        "pan":"ASDS"
    }
}
```

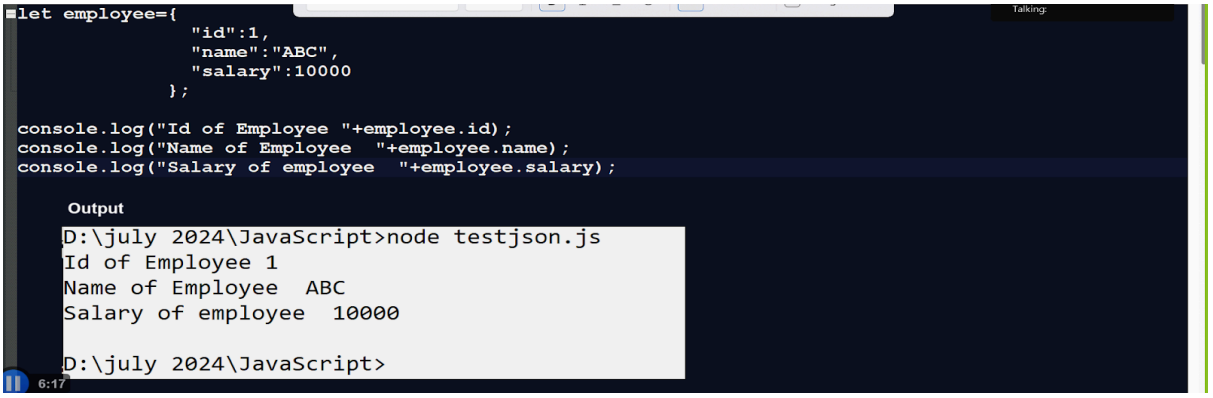
6. **Good for asynchronous communication :** JSON is widely used in AJAX (Asynchronous JavaScript and XML) to transmit data between client and server. It is fast and lightweight and perfectly works with asynchronous data processing in modern web applications.
7. **Interoperability:** JSON is language independent format so we can share data between two applications that are developed by using different technology or may be the same technology without any complicated process so when we share data between two different technologies called as language interoperability we can achieve it using JSON.

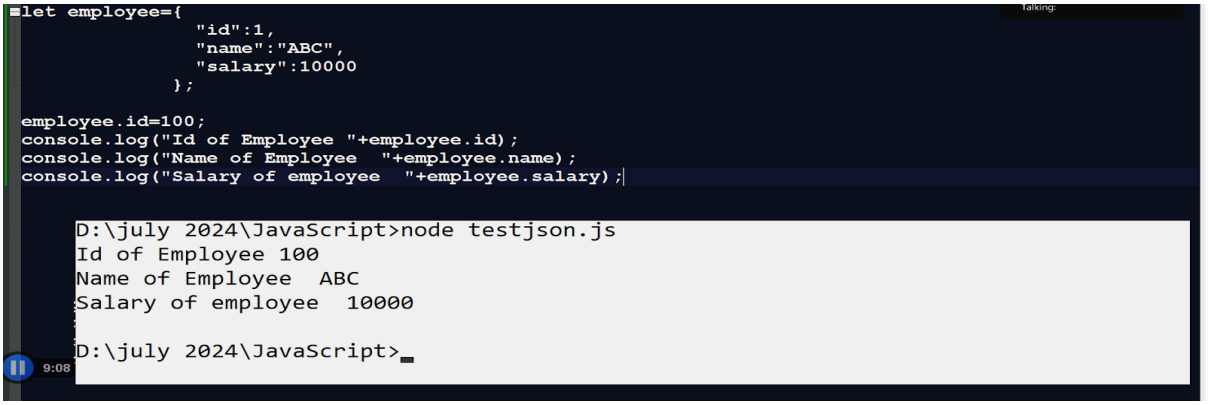
Note: JSON key is always in the form of string and values may be following types.

1. String
2. Number
3. Object
4. Array
5. Boolean
6. Null

   In JavaScript values can be all of the above plus any valid JavaScript expression including

   a. Function
   b. Date
   c. Undefined





Example: We want to design form with field name,email and contact with two buttons

1. **Store data in JSON**
2. **Display JSON Data**

Example with source code

---

test.js

```
let reg={
   name:"",
   email:"",
   contact:""
}

let save=()=>{

        let nm=document.getElementById("n").value;
        let em=document.getElementById("e").value;
        let ct=document.getElementById("c").value;
        reg.name=nm;
        reg.email=em;
        reg.contact=ct;
        alert("Data Store Successfully....");
}
let show=()=>{
        document.getElementById("hn").innerHTML=reg.name;
        document.getElementById("he").innerHTML=reg.email;
        document.getElementById("hc").innerHTML=reg.contact;

}
```

Test.css

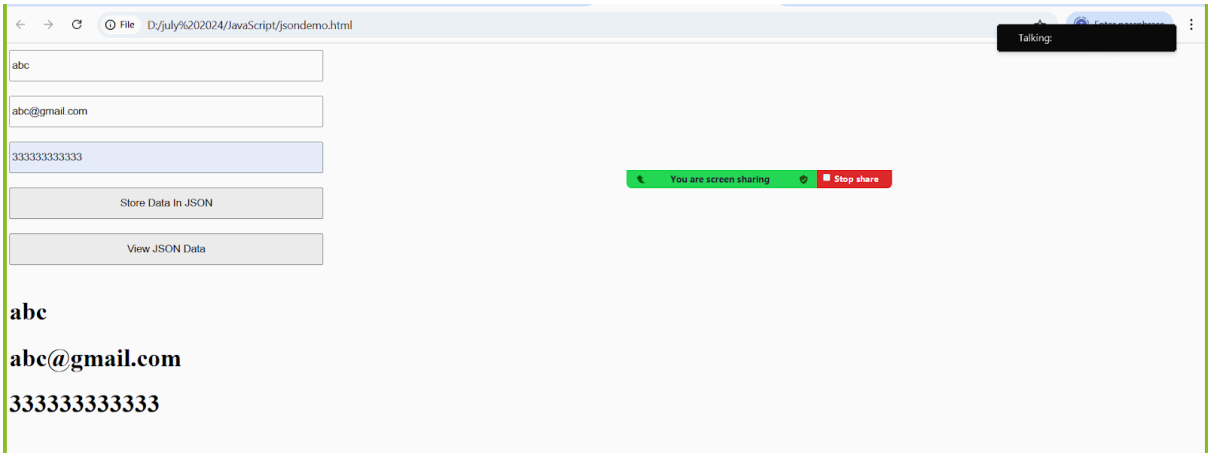
```
input{
  width:400px;
  height:40px;

}
```

Jsondemo.html

```
<html>
```

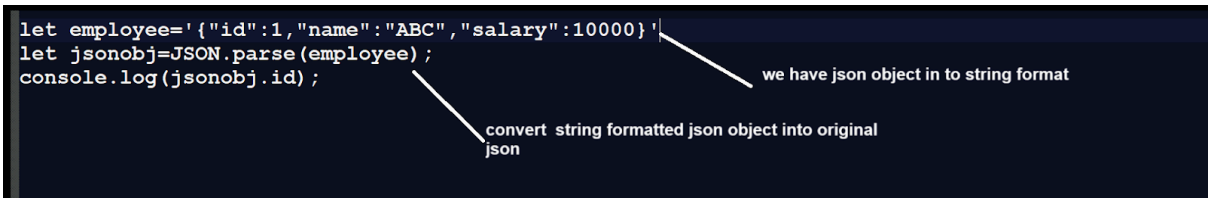
```
 <head>
  <title>json data example</title>
  <link rel="stylesheet" href='test.css' />
  <script type='text/javascript' src='test.js'></script>
 </head>
 <body>

  <input type='text' name='name' value='' id='n'/> <br><br>
  <input type='text' name='email' value='' id='e'/><br><br>
  <input type='text' name='contact' value='' id='c'/><br><br>
  <input type='button' name='s' value='Store Data In JSON'
onclick='save()'/><br><br>
  <input type='button' name='s1' value='View JSON Data'
onclick='show()'/><br><br>
  <h1 id="hn"></h1>
  <h1 id="he"></h1>
  <h1 id="hc"></h1>
 </body>
</html>
```

Output

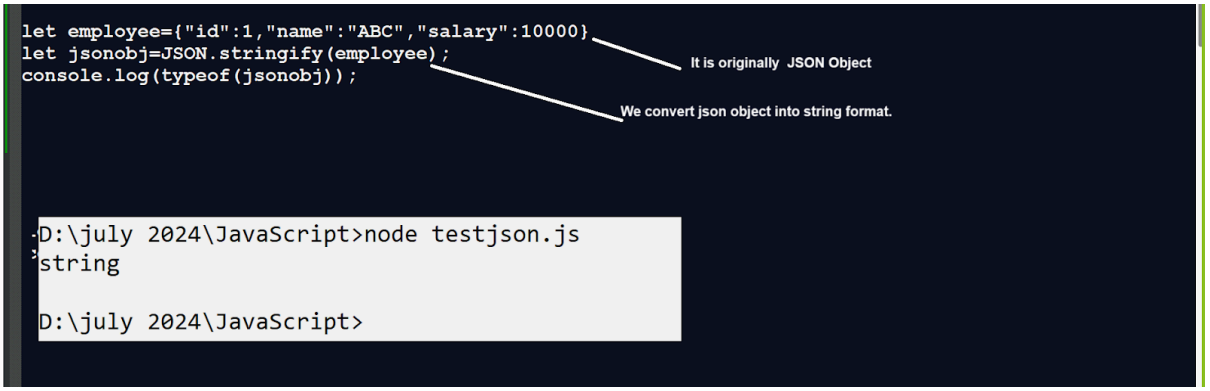


Methods of JSON

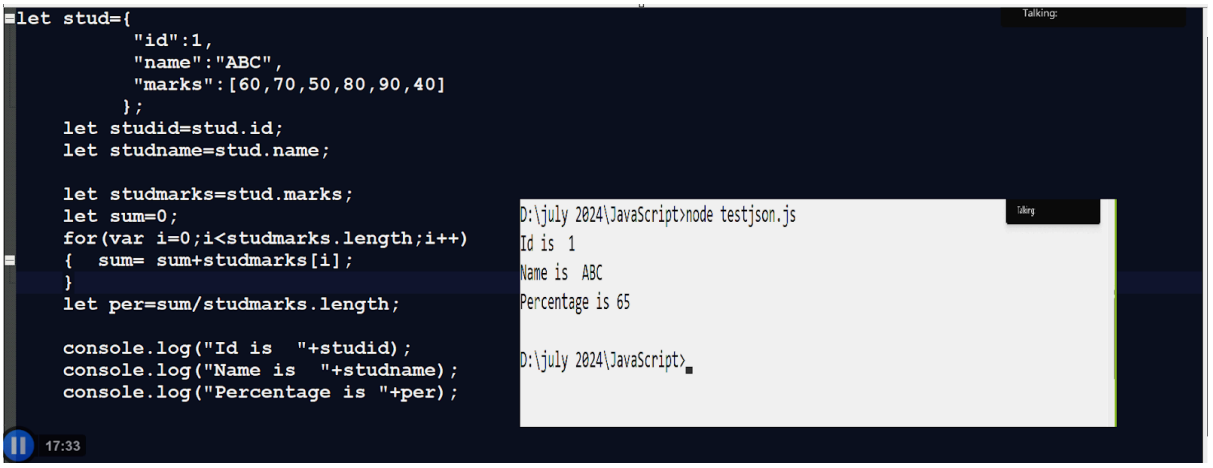
JSON.parse(): JSON.parse() method is used for convert string to json Object because when we get data from a server we get in the form string format so when we convert that data into json object then we can use JSON.parse() method



JSON.stringify(): this method converts json objects into string format.



Example: we want to create student json object which contain id,name ,marks of six subject and display the id ,name and percentage of student



Example: Suppose consider we multiple students

```
let studarr=[{
                    "id":1,
                    "name":"ABC",
                    "marks":[60,60,60,60,60,60]
          },
                   {
                    "id":2,
                    "name":"MNO",
```

```
                    "marks":[50,50,50,50,50,50]
                   },
                   {
                    "id":3,
                    "name":"XYZ",
                    "marks":[70,70,70,70,70,70]
                   }
          ];

let studCount=studarr.length;
 for(var i=0;i<studCount; i++)
 {
          let m=studarr[i].marks;
          let agg=0,per;
          for(var j=0;j<m.length; j++)
          { agg=agg+m[j];
          }
          per=agg/m.length;
          console.log(studarr[i].id+"\t"+studarr[i].name+"\t"+per);
 }
```

Output

```
D:\july 2024\JavaScript>node testjson.js
1       ABC     60
2       MNO     50
3       XYZ     70

D:\july 2024\JavaScript>
```

Promises in JavaScript

The promise object represents the eventual completion or failure of an asynchronous operation and its resulting value.

Note: before learning the promise we need to know synchronous operation and asynchronous operation in JavaScript.

Synchronous operation : synchronous operation means execute code line by line means when we first operation successfully completed after that second operation get executed and JavaScript is by default synchronous language as well as JavaScript is single thread language

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

When should promises be used?

---

1. Producing code is code that can take some time
2. Consuming code : is code that must wait for the result.

Example: we can use promises for fetch API data from server etc

How to use promises practically in JavaScript

If we think about promise then JavaScript provide one class to us name as Promise and Promise class has one constructor and constructor contain call back function as parameter and call function contain two parameters resolve ,reject

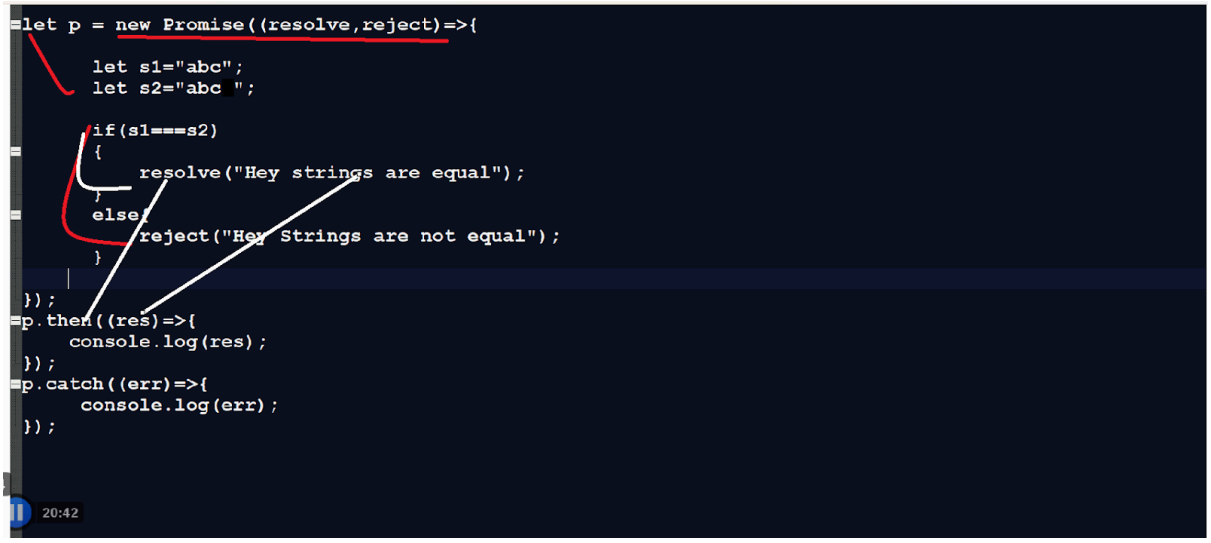
```
syntax:
let variablename = new Promise(function(resolve,reject){


});
```

1. The promise constructor take only one parameter argument which is callback function
2. The callback function take two parameters resolve and reject Perform operation inside the callback function and if everything went well then call resolve and if desired operation not meet then call reject

Note: JavaScript promise object can be

1. **Pending:** while promise object pending (working) the result is undefined
2. **Fulfilled:** when a promise object is fulfilled the result is value.
3. **Rejected :** when a promise object is rejected the result is an error object.

Example: we want to compare two strings with each other using promises.



Example: Write promise object which check number is even or odd if number is even then promise object should generate result success or resolve and if number is number is odd then promise object should generate reject

Event loop in JavaScript

JavaScript is a single threaded language , which means it can execute a piece of code/onecommand or statement at a time using sequential direction means one after another.
But Asynchronous operation or asynchronous callback to perform non-blocking operation like network request, file I/O or times etc Asynchronous operation if a particular operation requires some time for execution then waiting period of that operation can execute other task called as synchronous and asynchronous operation in JavaScript can handle by using event loop concept.

Event loop is the mechanism that handles the following things.

1. **Call Stack:** Call stack is a place where synchronous operation get executed means single threaded logics execute in call stack.

2. **Web API/browser:** when any asynchronous activity or code found then Call Stack handover the code to the browser like as setTimeout or DOM events etc
3. **Callback Queue :** where asynchronous call back maintenance means if we have multiple asynchronous operations then maintain in the callback queue.

4. **MicroStack Queue :** for promises and other microstack has higher priority than callback queue

   Event Loop can check following things continuously

   ---

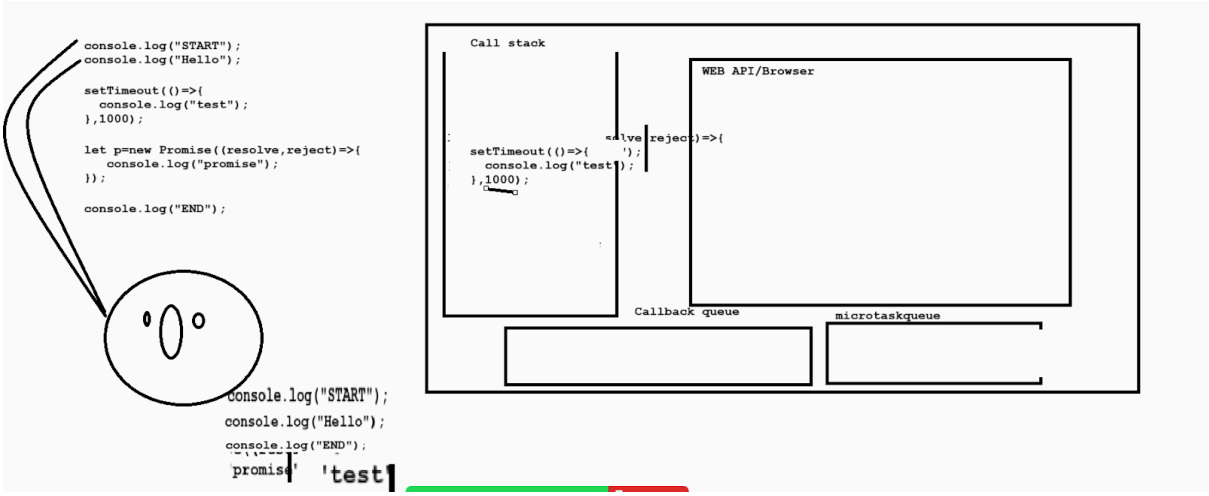
   1. If the call stack is empty
   2. Then pushes task from the microstack queue first
   3. Then from call back queue into the call stack execution

Example with source code

```
console.log("START");
console.log("Hi");

setTimeout(()=>{
  console.log("i am set timeout");
},100);

let p=new Promise((resolve,reject)=>{
        setTimeout(()=>{
  console.log("I am promise");
        },500);
});

console.log("END");
```

```
D:\july 2024\JavaScript>node testeventloop.js
START
Hi
END
i am set timeout
I am promise
```

Following diagram shows the working of event loop



OOP concept using JavaScript

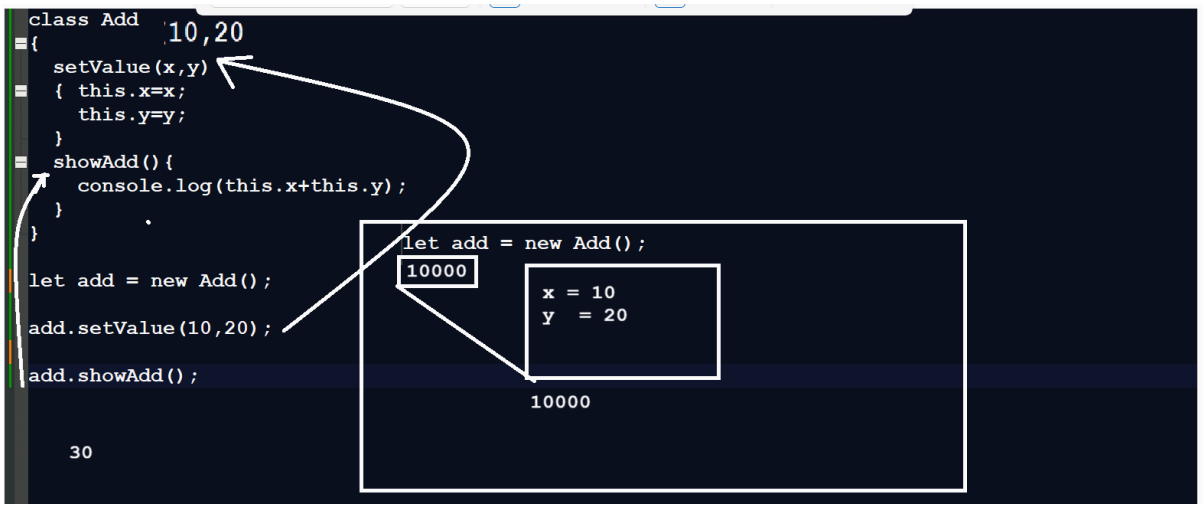
How to write a class in JavaScript?

Syntax:

```
class classname
{
    functionname(localvariable){
      this.instancevariable=localvariable;
    }
}
```

How to create object of class in JavaScript

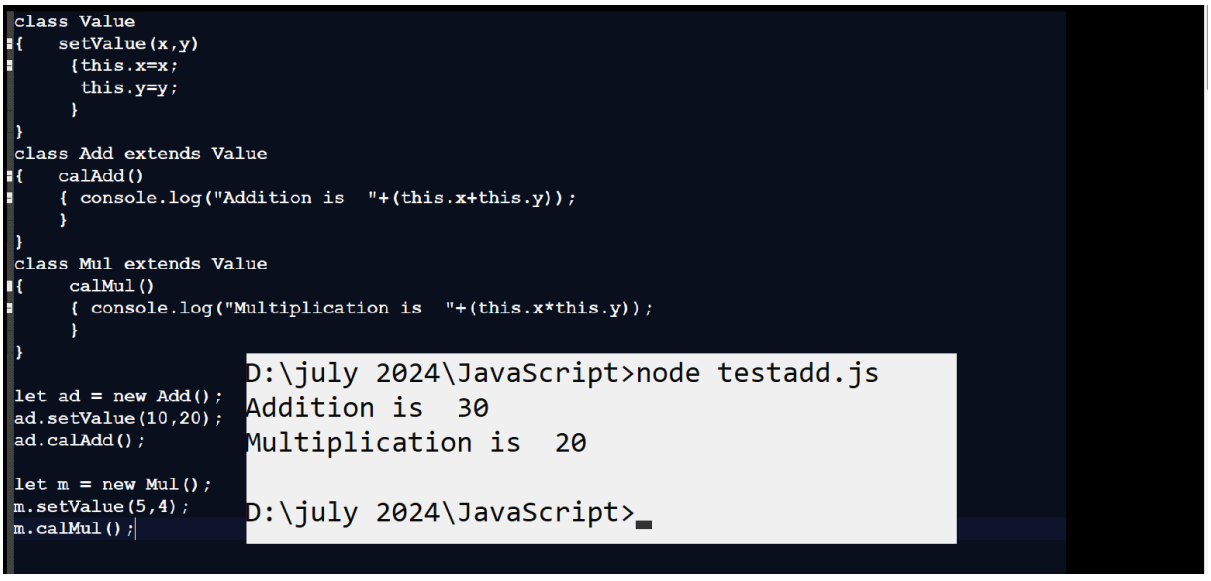
Syntax: let variablename= new classname();



How to write constructor in class

If we want to write a constructor in class we have a special function provided by JavaScript name as constructor and this function is called automatically when we create an object of the class.

How to perform inheritance using Script



Constructor using inheritance

```
class Value
{   constructor(){
        console.log("Hey i am parent constructor");
  }
}
class Add extends Value
{
  constructor(){
```

```
        super();
       console.log("I am child constructor");

  }
}
let ad= new Add();
```

Example : parent class with parameterized constructor

```
class Value
{    constructor(x,y){
        console.log("Hey i am parent constructor "+(x+y));
  }
}
class Add extends Value
{
  constructor(){
        super(10,200);
       console.log("I am child constructor");

  }
}
let ad= new Add();
```

Fetch API In JavaScript

Fetch API provides an interface for fetching resources like JSON data across the web.
Means using Fetch API we can retrieve data from REST API design by using any technology using JavaScript.
Means may be backend technology is Java Spring boot or Express etc
It offers more powerful and alternative to traditional XMLHttpRequest

Key Feature of Fetch API

---

1. **Promise based :** Simplifies asynchronous operation with JavaScript promise.

2. **Cleaner syntax : use feature for the request handling**
3. **Modern feature :** including streaming responses and CORS feature
4. **Browser Support:** widely support major browsers.

How to use JavaScript Fetch API

JavaScript fetch API uses the fetch() function and its core. The Fetch method take and mandatory argument

1. The URL of the resource that you want to get
2. It accept optional parameter like as HTTP Methods ,headers, body and more etc

```
let promiseobject = fetch(url,[options]);
promiseobject.then((response)=>{
write here your logics
});

promiseobject.catch((error)=>{
  write here your logics
});
```

main.js

//fetch method use GET method
//GET method is used for retrive or access data from server

```
let promise =fetch("https://jsonplaceholder.typicode.com/todos/");
promise.then((response)=>response.json()).then(json=>{
        let createTable=document.createElement("table");

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

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

        th=document.createElement("th");
```

```
        th.innerHTML="title";
        tr.appendChild(th);

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


        createTable.appendChild(tr);
        createTable.style.border="2px solid red";
        createTable.style.width="50%";
        createTable.style.marginLeft='200px';

        for(var i=0;i<json.length;i++)
        {     tr=document.createElement("tr");
              let td=document.createElement("td");
              td.innerHTML=""+json[i].id;
              tr.appendChild(td);
              td=document.createElement("td");
              td.innerHTML=""+json[i].title;
              tr.appendChild(td);

              td=document.createElement("td");
              td.innerHTML=""+json[i].completed;
              tr.appendChild(td);

              createTable.appendChild(tr);

        }
        document.getElementById("grid").appendChild(createTable);

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

Fetchapi.html

```
<html>
 <head>
  <title>i am fetch api demo</title>
  <script type='text/javascript' src='main.js'> </script>
 </head>
 <body>
  <div id="grid">
  </div>
 </body>
</html>
```

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 by holding many pieces of code in it we wait for return value.

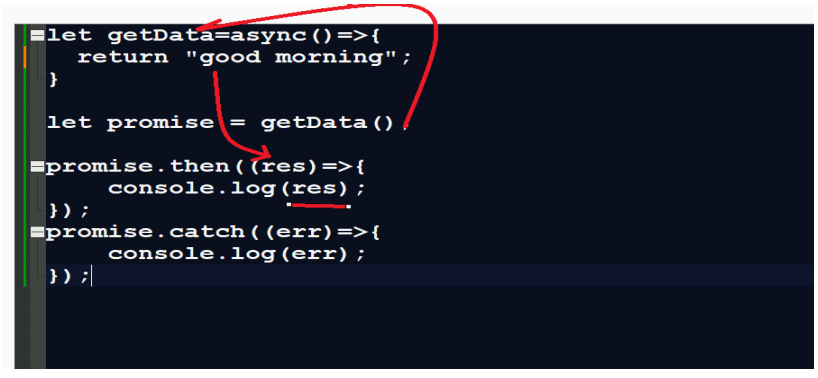
Means we can say use async keyword with a function and function always return promise object and using promise object can get result from async function and await keyword is used for hold the promise execution means you can use await keyword in async function.

Async function simply allows us to write the promise based code as if it where synchronous and it checks that we are not breaking the execution thread async function will return value.

Async syntax

```
async function functionname(){
}
```

Note: when we use any function on async we do not need to create an external object of promise.



Q. Why async keyword return promise?

For understanding this question we will discuss await keyword

Q. What is the await keyword?

Await is used to wait for the promise.it could be used within an async block only . it makes the code wait until promise return a result

Syntax of await

---

let value= await promise object;

Example with source code

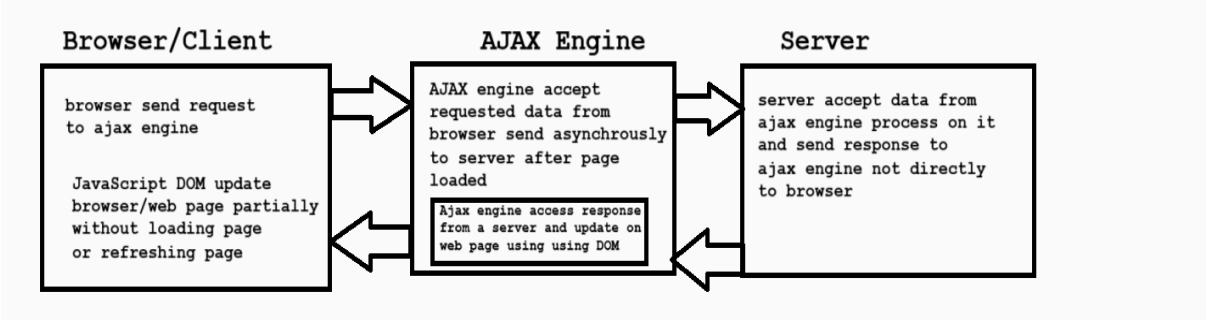
```
let tempCal=async()=>{

        let csnT=new Promise(function(resolve,reject){
                setTimeout(()=>{
                        resolve("44");
                },1000);
        });
        let pt=new Promise(function(resolve,reject){
                 setTimeout(()=>{
                        resolve("34");
                },7000);
        });

   let sT=await csnT;
   let pT=await pt;
   return [sT,pT];
}
 let result =tempCal();
  result.then((res)=>{
          console.log(res);
  });
```

AJAX

AJAX stands for asynchronous JavaScript and XML called as AJAX
Basically AJAX technology develop by google in 2005 and the main goal of this technology is to perform partial updation of web page means update the web page content without reloading complete web page
The main goal of Ajax is to improvise the web page performance

Why use Ajax or What are the benefits of Ajax?

1. Update a web page without reloading the web page
2. Request data from a server -after the page has loaded
3. Receive data from a server -after the page has loaded
4. Send data to server - in the background.



How use Ajax Practically using JavaScript

Steps to use the AJAX practically using JavaScript

1. **Create object of XMLHttpRequest**

   XMLHttpRequest objects can be used to exchange data with a server behind the scenes. This means that it is possible to update parts of a web page without reloading the whole page.

   Syntax of creating object of XMLHttpRequest

   let var= new XMLHttpRequest();

Methods of XMLHttpRequest

---

abort(): cancel the current request
getAllReponseHeaders(): return the header information
open(method,url,async): this method help us to send request to server using some specified url

Parameter details

Method: method decide how we submit form GET/POST
Url : url parameter means we provide here server side web page or url where we want to send requests.
Async: async decides asynchronous behaviour and when set true then AJAX send request using asynchronous format and if we set false then AJAX send request using synchronous format.

send(): send the request to server using GET method
send(String): send the request to the server using the POST method.

XMLHttpRequest object properties

---

onreadystatechange : defines a function to be called when the readystate property changes.
readState: hold the status of XMLHttpRequest object

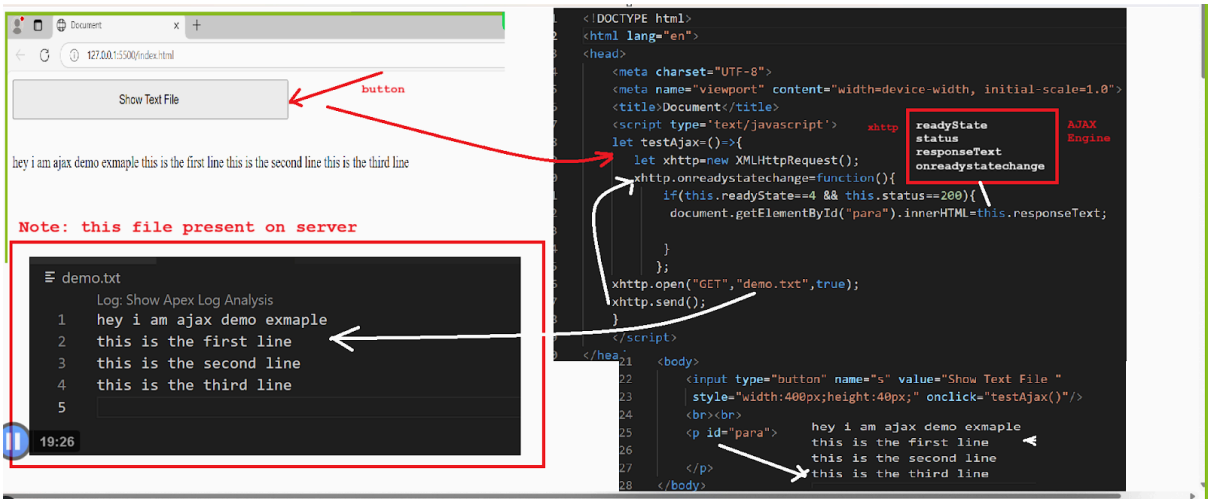
Code	**Meaning of request**
0	Request not initialized
1	Server connection established
2	Request received
3	Processing request
4	Request finished and response is ready

responseText: return the response data as string send by server

status: return the status number of request. Normally status help us to identity response from a server

Status code	meaning
200	OK
403	Forbidden
404	Not found

statusText: return the status text e,g OD or Not Found etc



Example with source code

Abc.txt

```
hey i am ajax demo exmaple
this is the first line
this is the second line
this is the third line
```

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Document</title>
    <script type='text/javascript'>
    let testAjax=()=>{
       let xhttp=new XMLHttpRequest();
       xhttp.onreadystatechange=function(){
           if(this.readyState==4 && this.status==200){

document.getElementById("para").innerHTML=this.responseText;

           }
          };
    xhttp.open("GET","demo.txt",true);
    xhttp.send();
    }
    </script>
</head>
<body>
    <input type="button" name="s" value="Show Text File "
     style="width:400px;height:40px;" onclick="testAjax()"/>
    <br><br>
    <p id="para">

    </p>
</body>
</html>
```

Closure concept in JavaScript

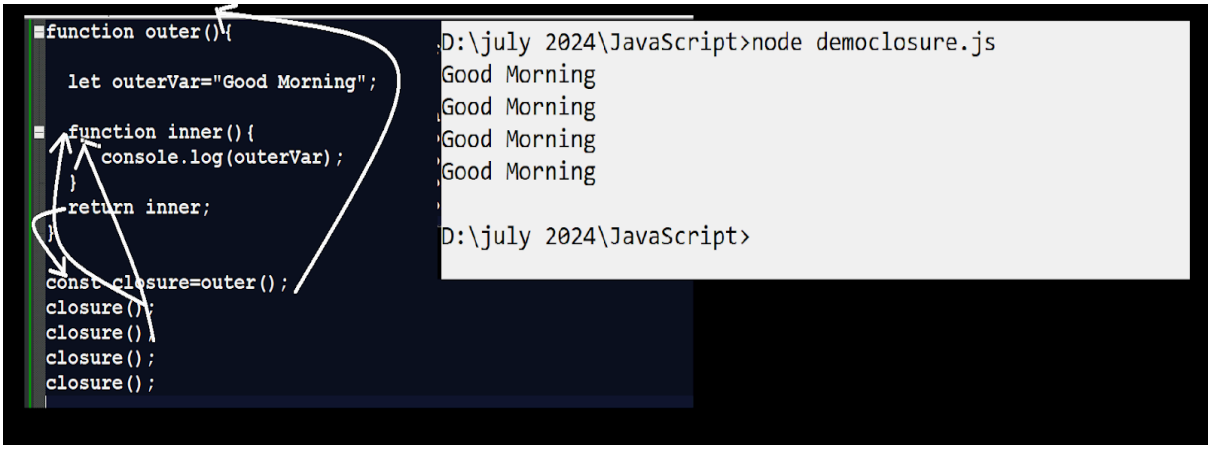
Q. What is closure in JavaScript?

A closure is a function that allows access to variables from its outer function and global variables from its outer function and mark it as global arable event after the outer function has finished executing.
This enables functions to remember their environment.

Important points related with closure

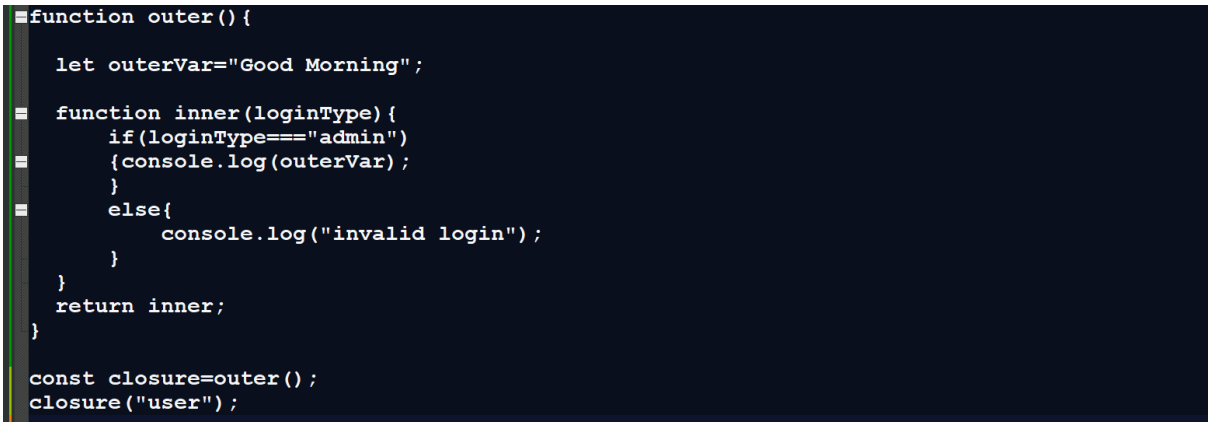
---

1. A closure allows a function to access variables from its outer function even after it has finished executing
2. Global variable can be made private within a function using closure ,ensure they cannot be access or modified directly from outside
3. Closure provides a way to encapsulate private data and create public methods to interact with it.
4. Closure help to retain reference to variables that would otherwise be lost after the execution other outer function



Note: if we think about above example

1. We have outer() function and it contain local variable outerVar
2. In outer() we define inner function which logs the value of outerVar
3. outer() function return value of inner function and hold its reference in closure variable means here closure work as inner because address of inner function stored in closure reference
4. So when we call closure then indirectly inner get execute and we get value of outerVar but outerVar is member of outer function but we not executing outer function but get value of outer variable using inner function means inner function hold outer variable value so if outer function not in execution but we can use its variable it called as closure concept.



```
function outer(){

  let outerVar="Good Morning";

  function inner(loginType){
      if(loginType==="admin")
      {console.log(outerVar);
      }
      else{
          console.log("invalid login");
      }
  }
  return inner;
}

const closure=outer();
closure("user");
```

Lexical scoping

Lexical scoping means that a function scope is determined by where the function is defined , not where it is executed
This allow inner functions to access variables from their outer function

```
function outer(){

  let outerVar="Good Morning";

  function inner(loginType){
     if(loginType==="admin")
         {console.log(outerVar);
         }
         else{
                 console.log("invalid login");
         }
  }
  return inner;
}

const closure=outer();
closure("user");
```

How to declare private variable using closure in JavaScript

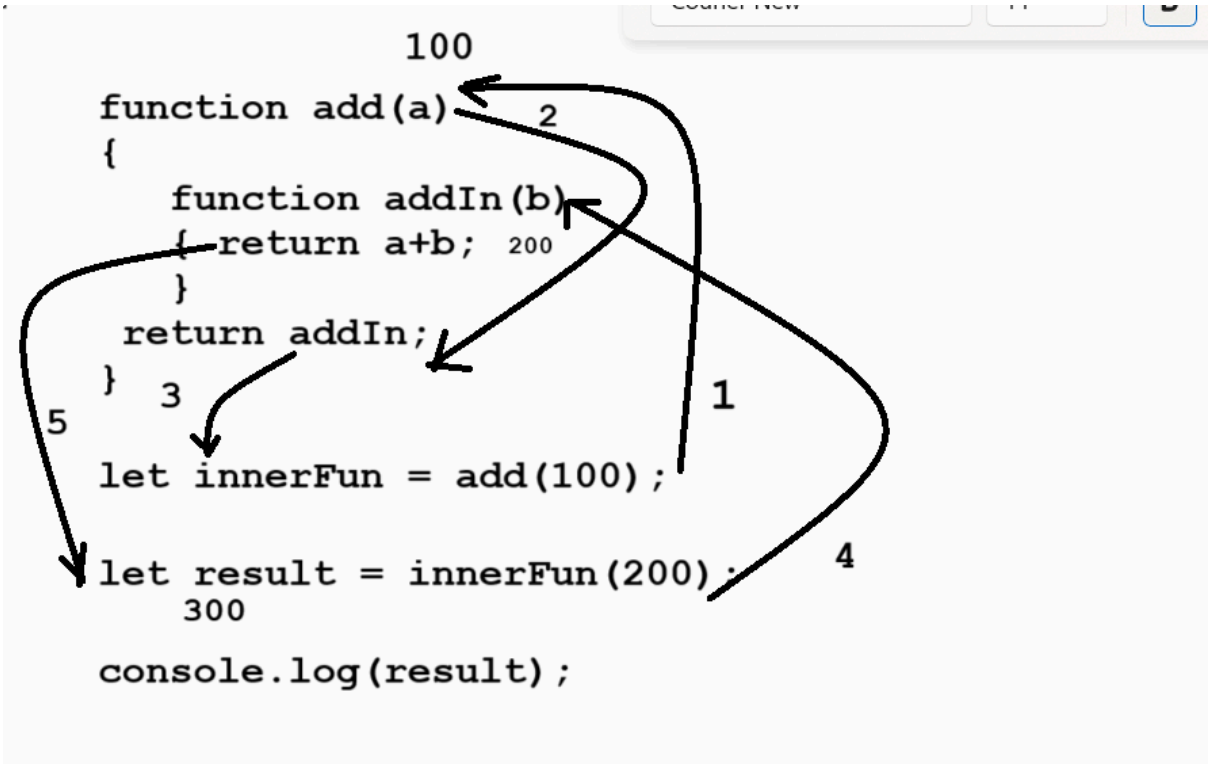
Closure allows a function to keep variables hidden and only accessible within that function. This is often used when we create modules in a project and need to protect data from being accessed or modified by other modules.

Function Currying in JavaScript (Example of closure)

Function Currying is JavaScript technique to transform a function that takes multiple arguments into a series of functions one argument at a time. Currying relies on closure because each of the intermediate function has access to the argument passed previously

Or

Simple word we can say Currying is facility if we have multiple nested function then each and every function take single argument at time and inner function can access outer function parameter and we can call inner function separately for passing parameter called as function currying



Or

```
function add(a)
{
  return function (b){
    return a+b;
  }

}

let innerFun=add(100);

let result =innerFun(200);

console.log(result);
```

JavaScript Object literals

JavaScript object literal is a concept where we can declare javascript object with constant value using key and value pair like as JSON object but the major difference Json object and JavaScript object is JSON object key always present in the form of string and JavaScript object key

always is not present in the form of string they consider as normal variable or state variable.