**What is JSON**

JSON stands for JavaScript Object Notation

JSON is often used when data is sent from a server to a web page

If we want to share or stored and send small amount of data to web server at that time we often used JSON.

let jsonObj={

    name:'shubham',

    age:'27'

}

console.log(typeof(jsonObj),jsonObj);//if want to convert object to json

let parseJSNObj=JSON.parse('{"name":"shubham"}');//if want to convert string to json

console.log(typeof(parseJSNObj),parseJSNObj);

let parseOBJTOJSON=JSON.stringify(jsonObj);//if want to convert obj to JSON string

console.log(typeof(parseOBJTOJSON),parseOBJTOJSON);

**Array Slice and splice Method**

Array slice methods gives selected range array values like

var fruits=["Mango","Banana","pinaple","Kanis"];

var NewArr=fruits.slice(0,2);

Array aplice method used to delete values from array in range

var fruits=["Mango","Banana","pinaple","Kanis"];

var NewArr=fruits.splice(0,2);

**== and === Diff**

**== checks only values without comparing datatype**

**=== checks values as well as its datatype**

var val1="1"

var val2=1

if(val1==val2)

{

    console.log("True");

}

else

{

    console.log("false");

}

if(val1===val2)

{

    console.log("True");

}

else

{

    console.log("false");

}

**Lamda or Arrow function**

var myFunc= () => console.log("My first lamda function");

myFunc();

var myFunc2=(a=10,b=5)=>{

    return a+b;

}

console.log(myFunc2(2,2));

var myFunc3=()=>{

    return({name:"Shubham"});

}

 console.log(myFunc3());

**High order function**

We can call function inside that another function gets called

var myHOFunc=(name)=>{

    if(name=="Shubham")

    {

        return function(behaviour){

            console.log(name + " " + "You are " + behaviour);

        }

    }

    else if(name=="Anil")

    {

        return function(behaviour){

            console.log(name + " " + "You are " + behaviour);

        }

    }

    else

    {

        return "Pass correct Parameter"

    }

}

myHOFunc("Shubham")("Good");

myHOFunc("Anil")("Bad");

**var VS let**

Let is block level scope element

**Hoisting in JS**

If we assign variable or call function before declaration or assign which is possible in javascript that is known as Hoisting

Hoisting is possible with normal functions, hoisting is not possible when we used arrow funtions

myName="shubham";

console.log(myName);

var myName;

getmyAge(26);//hoisting with normal function is ok

function getmyAge(age)

{

    console.log("My age is " + age);

}

myfun4();//hoisting with arrow function is not possible

var myfun4=()=>{

    console.log("This is an arrow function");

}

**Pure and Impure function**

Pure function returns same value with same input every time.

Impure functions returns diff values every time.

var valx=1;

var Valy=1;

var myfun5=()=>{

    valx=valx+1;

    console.log(valx);

}

var myfun6=()=>{

    console.log(Valy);

}

myfun5();

myfun5();

myfun5();

myfun6();

myfun6();

myfun6();

**this keyword**

this keyword is referes to the object it belongs to.

console.log(this);//referes to window object

function myfun7()

{

    console.log("Hello");

}

this.myfun7();//referese to window object it is global liberary space

const myfirstobj={

    name:"shubham",

    myfun6:function(){

        console.log("this is myfirstobj Function");

        console.log(this.name);//refer to the myFIrstobj

    }

}

myfirstobj.myfun6();

**DOM(Document Object Model)**

Whenever our page loads, the Document object model is created accoding to our code.

In the document object all of the info like tags elements, links are here.In the model there we can get a tree wise structure of our code.

**Types of functions in JS**

//Function Declration

function FDexample()

{

    //statements

}

//function expression

var FEval=function(){

    //statements

}

//arrow function

var AFexample=()=>{

    //statements

}

**Array**

var namesarr=["Shubham","Anil","Shinde","Anagha"];

var namesarrCopy=[];

for(var i=0;i<=namesarr.length-1;i++)

{

    console.log(namesarr[i]);

}

namesarr.push("Atharva");//add element in last

namesarr.pop();//removes last element

namesarr.splice(1,1);//removes elements between range

namesarrCopy=namesarr.slice(0,2);//get and put elemnts in another arr

namesarr.shift();//removes first element from arr

namesarr.unshift("Mitali");//add element in starting of array

namesarr.splice(0,2,"1","2","3","4");//removes the array from start index and 2 is the count and adds next elements

console.log(namesarr);

console.log(namesarrCopy);

**Operstors**

// Operators

// Arithmetic

//assignment operators

//comparision operators

//logical operators

//logical AND

console.log (true && true);

console.log (true && false);

console.log (false && true);

console.log (false && false);

//logical or

console.log (true || true);

console.log (true || false);

console.log (false || true);

console.log (false || false);

// logical not

console.log (!false);

console.log (!true);

**DOM Manipulation**

let elem=document.getElementById('click');

console.log(elem);

let elemclass=document.getElementsByClassName('container');

console.log(elemclass);

elemclass[0].style.background = "yellow";

elemclass[1].classList.add("bg-primary");

elemclass[1].classList.add("text");

console.log(elem.innerHTML);

console.log(elem.innerText);

console.log(elemclass[0].innerHTML);

console.log(elemclass[0].innerText);

let getLstTagName=document.getElementsByTagName('div')

console.log(getLstTagName);

createdElement=document.createElement('p');

createdElement.innerText='Hello this is created Para';

createdElement2=document.createElement('b');

createdElement2.innerText='Hello this is created Para 2';

getLstTagName[0].append(createdElement);

getLstTagName[0].replaceChild(createdElement2,createdElement);

getLstTagName[0].removeChild(createdElement2);

sel=document.querySelectorAll('.container');

console.log(sel);

**Events**

var show=()=>{

    document.querySelectorAll('.container')[0].innerHTML="<b>DOM Manipulation Hua</b>";

    alert("clicked hua");

}

var mouseoverfn=()=>{

    alert("mouse over");

}

var mouseoutfn=()=>{

    alert("mouse out");

}

PrintFruits.addEventListener('click',show);

// PrintFruits.addEventListener('mouseover',mouseoverfn);

// PrintFruits.addEventListener('mouseout',mouseoutfn);//mouseup/mousedown

**Interval**

let printTimeOut=()=>{

    console.log("Hi");

}

let setIntervalFn=()=>{

    console.log("Interval called");

}

setTimeout(printTimeOut, 1000);

// setInterval(setIntervalFn,2000);

**Local storage**

localStorage.setItem('name','shubham');

undefined

localStorage

Storage {name: 'shubham', length: 1}

localStorage.getItem('name')

'shubham'

**Async Aswait**

function printresult(res)

{

    console.log(res);

}

async function asyncawait(){

    console.log("Inside async await");

    const response=await fetch('https://api.github.com/users');

    const users=await response.json();

    console.log("Inside async await 2");

    return users;

}

console.log("before async await called");

let res=asyncawait();

console.log(res);

console.log("After async await");

console.log(res.then(data=>printresult(data)));

console.log("Last Line of this JS File");

**call back function**

* We can pass the function as parameter which is a call back function

let users;

async function getstudents(callback){

    const response=await fetch('https://api.github.com/users');

    users=await response.json();

    callback();

}

function printallusers(){

    console.log(users);

}

getstudents(printallusers);

//printallusers();

**Promises**

1. Pending 2.Resolved 3. Reject

function promisesEXample(){

    return new Promise(function(resolve,reject){

        let boolVal=true;

        if(boolVal){

            console.log("Promise resolved");

            resolve();

        }

        else{

            console.log("Promise Reject");

            reject();

        }

    })

}

promisesEXample().then(function(){

    console.log("Success");

}).catch(function(){

    console.log("Failed");

})