Javascript webdev

Front-end Javascript backend javascript

(MongoDB)

<script>

alert(“Hello world”)

</script>

Semicolon is optional in javascript;

**To link -** <script src =”java.js”> </script>

Console.log(“hey I am console”)

Var a= prompt(“enter your no”)

* var IsTrue= confirm(“are you sure”)

if isTrue{

console.log(“computer is blasting”)

}

Else{

console.log(“not blasting”)

console.log(“your name “ + a)

Variable data types and objects

Javascript automatically detects what kinda variable it is no data types is to give

**Var a = 5;**

**Var b = 4;**

**Var c =”sid”;**

**Console.log(a+b+8)**

**Typeof Operator**

Console.log(typeof a, typeof b, typeof c)

**Var vs Let**

Var is a globally available where as let is only functional till the block.

Always use let because if we use it bhar it will automatically become a global variable.

**Constant**

Const a1=6 ( nothing can change this now )

**DATA TYPES**

**Primitive**

Null number string symbol undefined

Boolean Bigint

**Object**

let o = {

    "name" : "sid",

    "job title" : "web developers"

}

console.log(o)

*/ to change some value inside object*

o.salary = "500 crore"

console.log(o)

this will add salary section to the object it is like dictionary I guess

IF else statements

If else statements works same as in other languages.

let age = 115;

if (age > 19) {

    console.log("siddhu mossewaala")

}

else {

    console.log("not siddhu mosselwala")

}

**Operators :**

**Arthmatic operator**

Addition +

Subtraction –

Multiplication \*

Exponent or power \*\*

Division /

Modulus % tells reminder

Incriment ++

Dicriment –

**Assignment operator**

**=**

**+=**

**-=**

**\*=**

**/=**

**%=**

**Comparsion operator:**

**== equal to “3”==3** is same

**!= not equal to**

**=== equal value and type “3”==3 is not same**

**!== not equal value and type**

**><**

**>=**

**<=**

**? ternary operator**

**Logical operator**

&& logical and

|| logical or

! logical not

And bitwise operator

Loops

**For loop**  loop a block of code no of times

**For in loop** loop through keys of a object

**For of loop** loop through the values of object

**While loop**  loop a block based on condition

**Do while loop** while loop variant which run atleast once

**For loop example:**

for (let i = 1; i <=100; i++) {

    console.log(i)

}

**For in and for of loops example:**

let obj = {

    first:"kiss",

    second:"love",

    third:"idk"

}

for (const key in obj) {

    console.log(key)

}

for (const element of "Siddharth") {

    console.log(element)

}

Functions

function greeting(name) {

    console.log("Hey"+ name +"how are you?")

    console.log(name +"how you been?")

}

greeting(" sid ")

function sumof(a,b) {

    return a+b;

}

console.log("sum of these two number is  "+ sumof(10,20))

arrow function -

where we can store function as a variable idk

const func1 = (x) =>{

    console.log("i am a arrow function", x)

}

func1(2);

func1(4)

func1(5)

Strings

Read this in pdf

Backstrick property for strings

And double and single strings with each other with one go.

let your\_name = "siddharth"

console.log(`hey how are you ${your\_name} and what can i do for you`)

Arrays

Arrays are variable that can hold multiple and different values simultaneously.

let arr = [1,2,3,4,5,6];

console.log(arr)

console.log(arr.length)

Arrays are mutable and strings are not mutable

arr[3] = 555;

console.log(arr[3])

Type of array is object

console.log(typeof(arr))

**To convert to a string**

console.log(arr.toString())

**Join()**

console.log(arr.join(" and "))

1 and 2 and 3 and 555 and 5 and 6 output

**Concat()**

let arr1 = [1,2,3]

let arr2 = [4,5,6]

let arr3 = [7,8,9]

console.log(arr1.concat(arr2,arr3));

this method doesn’t change array it just gives the output.

**LOOPS  
  
for loop**

let a = [2,55,5,77,4]

for (let index = 0; index < a.length; index++) {

    const element = a[index];

    console.log(element)

}

**for each**

a.forEach((value,index,array) => {

    console.log(value,index,array)

});

**For in –** used for object

let obj = {

    "1" : "sid",

    "2" : "tannu",

    "3" : "maggie",

}

for (const key in obj) {

    if (Object.prototype.hasOwnProperty.call(obj, key)) {

        const element = obj[key];

        console.log(element)

    }

}

**For of –** used for array

for (const element of array) {

    console.log(element)

}

**Array.map() –** create same copy of the array

**First simple way**

let arr = [1,2,3,4,5,6,7,8,9,10]

let newarr = []

for (let index = 0; index < arr.length; index++) {

    const element = arr[index];

    newarr.push(element\*\*2)

}

console.log(newarr)

**with using .map()**

let arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

let narray = arr.map((e) => {

    return e \*\* 2

})

console.log(narray)

**arr.filter()**

const GreaterThanSeven = (e)=>{

    if(e>7){

        return true;

    }

    else{

        return false;

    }

}

console.log(arr.filter(GreaterThanSeven))

**arr.reduce() –** it 1 \* 2 \*3\*4\*5

console.log(arr.filter(GreaterThanSeven))

let arr2 = [1,2,3,4,5,6]

const red = (a,b)=>{

    return a + b

}

console.log(arr.reduce(red))

Document Object model

A diagram of a document

AI-generated content may be incorrect.

Dom represent as the page content in HTML in short you can use html using javascript with it

Mapping of entire document into a Object.

      document.title = "Sid is supreme "

        document.body.style.backgroundColor="aqua"

        console.log(document.title)

        console.log(document.body)

**Browser Object Model (BOM)**

The browser Object Model represents additional object provided by browsers (host environments) for working with everything except the document

The functions  **alert/confirm/prompt** are also part of BOM

Location href = “https://sidissupreme.com”

**Window**

Window is global object inside that document and bom exist we don’t write

Window.document.body because it’s a global object and we can work without it easily too.

Dom Nodes

**document.body.childNodes** for ALL the child nodes that exists

**document.body.childNodes[1]**

you can access child nodes using array like symbol[]

**cont.firstChild –** this will show the first child in the nodes and that will be the **a text node** ( a text node is enter or space)

**cont.firstElementChild –** this will skip the text node and give us the html css node that we want like boxes or something.

**cont.firstElementChild.parentElement –** this will get the parent element

**document.body.firstElementChild.children**

HTMLCollection(5) [div.box, div.box, div.box, div.box, div.box]

Children will show without text nodes and childnodes show textnodes

**document.body.firstElementChild.children[3].nextElementSibling**

children 3 => box4 nextElementSibling=> box5

**document.body.firstElementChild.children[3].previousElementSibling**

children 3 => box4 previousElementSibiling=>box3

selectors

**By Class name**

let boxes = document.getElementsByClassName("box");

console.log(boxes)

**By Id**

document.getElementById("red\_colour").style.backgroundColor="aqua";

**By query selector**

document.querySelector(".box").style.backgroundColor="aqua";

this will act on first class element. Only the first one.

document.querySelectorAll(".box").forEach((e)=>{

    e.style.backgroundColor="aqua"

})

To access all the elements use queryselectorall but that will return html group like array is html group to use query selector use foreach loop with arrow function.

Factorial

let a = 6;

function factorial (number) {

    let arr = Array.from(Array(number + 1).keys()) *// for making array without loop*

    console.log(arr.slice(1,))                  *// This will print array from 1 to 6*

    let c = arr.slice(1,).reduce((a,b)=>{

        return a\*b;

    })

    console.log(c)

}

factorial(3)

**this is with reduce.**

**Now with for loop ;**

first with for loop

var num = prompt("Enter the number you want factorial of")

num = parseInt(num)

let fact = 1;

for (let i = 1; i <= num; i++) {

    fact = fact\*i;

}

console.log(fact)

Insert delete using Dom

**document.querySelector(".box").innerHTML –** returns inner html of the selected element

**document.querySelector(".container").innerText –** return inner text

example - 'Hey i am not a box i supoose\nHey i am not a box i supoose\nHey i am not a box i supoose'

**document.querySelector(".container").outerHTML –**

'<div class="container">\n <div class="box">Hey i am not a box i supoose</div>\n <div class="box">Hey i am not a box i supoose</div>\n <div class="box">Hey i am not a box i supoose</div>\n </div>'

You get container as well in outerHtml

**Read notes for this if you need something Sid!!!!**

**Document.designMode=”on”** – now you can add or remove text in any website aram se I did with Netflix it was very fun.

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

        div.innerHTML="sid is here <strong> i am </strong>"

        div.setAttribute("class","created");

        document.querySelector(".container").append(div)

Little manupilation by DOM nice hai

**document.querySelector(".box").classList.add("kuchi") -**  to add class into the box

**document.querySelector(".box").classList.add("kuchi")-**  it is like a light switch if exist it will remove if not it’ll add.

Events

Diffrents events exist in javascript like what will happen when you click the mouse add event and that event will listen when mouse is clicked

Many events not just mouse , keyboard events too

let button = document.getElementById("btn")

button.addEventListener("click", ()=>{

    document.querySelector(".box").innerHTML="Thanks for clicking "

})

**To get the key value and number this helps in making of games**

button.addEventListener("keydown",(e)=>{

    console.log(e.key,e.keyCode)

})

Event Bubbling

   document.querySelector(".child").addEventListener("click",()=>{

            alert("child is clicked")

        })

        document.querySelector(".childcontainer").addEventListener("click",()=>{

            alert("childcontainer is clicked")

        })

        document.querySelector(".container").addEventListener("click",()=>{

            alert("container is clicked")

        })

So when we’ll click on child => child container and container would also click that is called event bubbling to stop that we use

**e.stopPropagation() –** this will stop event from going to others this will stop event right then and there

**setInterval**

function randomcolour() {

            let val1 = Math.ceil(0+Math.random()\*255)

            let val2 = Math.ceil(0+Math.random()\*255)

            let val3 = Math.ceil(0+Math.random()\*255)

            return`rgb(${val1},${val2},${val3})`

        }

        setInterval(() => {

            document.querySelector(".child").style.background=randomcolour();

        }, 1000);

        setInterval(() => {

            document.querySelector(".childcontainer").style.background=randomcolour();

        }, 1000);

        setInterval(() => {

            document.querySelector(".container").style.background=randomcolour();

        }, 1000);

Set interval will do task in certain interval and that interval is in **Miliseconds.**

**setTimeout()**

this will run only **once** after certain interval. Like a timebomb

clearsetinterval (number of sec eg 1)and clearsetTimeout(1) to stop them

Asynchronous, Callbacks and Promises

**Asynchronous in nature ( Taal dena )**

Javascript is asynchronous in nature meaning it will delay the task which are asynchronous in nature

Like setTimeout will delay and be executed at last

console.log("sid is supreme!!!!!")

setTimeout(() => {

    alert("hogya set timeout bhai")

    console.log("chal chal Timeout kar")

}, 2000);

console.log("chal chal ke dikha")

setTimeout will run at last first suprme and then chal chal ke dikha.

**Promise**

let prom1 = new Promise((resolve, reject) => {

    setTimeout(() => {

        console.log("Yes I am done");

        resolve("sid")

    }, 3000);

})

prom1.then((a) => {

  console.log(a);

}

)

**Callback**

Its like pizza shop telling you that once pizza is ready we’ll tell you

const callback = (arg) => {

    console.log(arg);

}

const loadScript = (src, callback) => {

  let sc = document.createElement("script");

  sc.src = src;

  sc.onload = callback("SID");

}

loadScript("https://cdn.jsdelivr.net/npm/prismjs@1.30.0/prism.min.js",callback);

when we have callbacks inside callbacks we get pyramid of doom,

**promise**

let prom1 = new Promise((resolve, reject) => {

    let rand = Math.random()

    if (rand > 0.5) {

        reject("no random no didnt support you")

    } else {

        setTimeout(() => {

*// console.log("Yes I am done");*

            resolve("Resolved babe")

        }, 1000);

    }

})

prom1.then((a) => {

    console.log(a);

}).catch((err) => {

    console.log(err);

**promise is like** if we madee a pizza in 30 min we’ll give it to you otherwise will tell you that we cant make your pizza sorry!

Async and await

*async* function getData() {

    return new Promise((resolve, reject) => {

        setTimeout(() => {

            resolve(420)

        }, 4000);

    })

}

*async* function main() {

    console.log("transferring data");

    console.log("making changes");

    let data = await getData();

    console.log(getData());

    console.log("data is transferring bla bal");

    console.log("DONE!");

}

main()

await can only be used inside the async function It for delaying the things that are delayed and running what can be runned rightnow got it?

**Its substitute for then and catch just use async await**

**Fetch API**

*async* function getData() {

   let api = fetch('https://jsonplaceholder.typicode.com/todos/1')

   let data = (await api).json()

   console.log(data);

   return(55)

}

First we’ll fetch API then we’ll convert it to json

**Settle** means resolve or Reject

**Resolve** means promise has been settled successfully

**Reject** means promise has not been settled successfully.

**🧪 Real-life Analogy:**

Imagine you're making Maggi.

* fetch() = you send your friend to buy the Maggi pack.
* await = you wait until your friend comes back.
* Then you cook it and eat it (console.log(data)).

Without await, you'd try to cook without the Maggi being there yet 🤦

TRY Catch error

let a = prompt("ENTER the first number.")

let b = prompt("ENTER the second number.")

if (isNaN(a) || isNaN(b)) {

    throw new Error("syntax error");

}

let c = parseInt(a) + parseInt(b);

console.log("The sum is ", c)

try {

    console.log(x + b);

} catch (error) {

    console.log("what is x")

}

To mess with errors we use try catch statement

**Finally**

function main() {

    try {

        console.log(x + b);

        return b

    } catch (error) {

        console.log("what is x")

        return x;

    }

    finally{

        console.log("chal chal ke dikha")

    }

}

let d = main()

finally we use cuz in function it will stop after return but finally still runs after return.