\*Template literals => Instead of using + or curly braces to seprate strings and var in console.log use "``"

example= console.log("Sum Of" + a "+" +b "is =" C);

instead of writing this we can write this.

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

\*deafult parameters or arguments =>when we assign a value to the parameters in the function definition.

example = function sum(a,b=5)this is called default parameter.

\*Fat arrow function(Imp) => we use "FUNCTION" keyword to create a function in JS but now using fat arrow function we

can create it very quickly and with more features.

exapmle =function sum()

{

let a=5 b=3;

return a+b;

}

lets convert it into fat arrow function

const sum = () =>

{

let a=5 b=3;

return a+b;

}

\*\*\*Some importrtant points in fat arrow function

1.when we create a function using "FUNCTION" Keyword we don't need to care about where we Initialized the function we can use

it anywhere(before initialization or after initialization) but in arrow fat function we first need to initialize it and

only then we can use it.

ex= sum()

function sum()

{

let a=5 b=3;

return a+b;

}

we can use traditonal function before initializing .

const sum = () =>

{

let a=5 b=3;

return a+b;

}

sum()

but in fat arrow function we first need to initialize it then we can use it.

2. if the line of code is in 1 line we don’t need to write it inside the curly braces we just return it like this

const sum = () => `sum of a + b is ${(a=5)+(b=5)}`

\*array in javascript

when we need to store multiple data in a single variable.

In java script array is a class.

Array is a prototype of a class.

Syntax

Var names = [“Rahul”,”Sameer”,34]

-data inside each box of array is called elements.

-lowest element or first element of array is known as lower Index or lower boundary.

-highest element or last element of array is known as upper index or higher boundary.

-index number in array start from 0.

-Index number of last element is array length -1.

Methods of array

---traversal of an array---

Navigate through the array

Or change data

Ex- var names = [“Rahul”,”Sameer”,34]

Console.log(names[1]);

Output – Sameer

Check the length or elements of an array

Syntax

Console.log(name.length);

-for in loop and for of loop

For in loop is used to print or display the index numbers of all the elements of an array.

Ex= var names = [‘sameer’,’rahul’,’rajan’,’rajat’];

For(let elements in names)

{

Console.log(elements)

}

Output-

0

1

2

3

For of loop is used to print all the elements of an array.

Ex= Ex= var names = [‘sameer’,’rahul’,’rajan’,’rajat’];

For(let elements of names)

{

Console.log(elements)

}

Output=

Sameer

Rahul

Rajan

Rajat

\*\*\*\*\*\*for each function \*\*\*

We can get output of index number , data inside an array in which we are working on at same time using for each function.

Syntax

Arrayname.foreach(function(element,index,array){

Console.log(element + index + array)

}

Java Script array methods

Convert an array into a astring

1.toString() => It is used to convert an array into a string using the comma seprator. Syntax console.log(arrayname.toString());

2. join() => it is used to convert an array into the string but you can also specify the separator between the array elements.

Syntax => console.log(arrayname.join(“\*”)

\*Push and pop

1. Push is used to Add the Element at the end of an array.

Syntax=> arrayname.push();

It returns the Length of a new array.

2. Pop is used to remove the last element of an array.

Syntax => arrayname.pop

It returns the removed value from the array

\*Shifting

1. Shift => it is used to remove the first element of an array.

Syntax => arrayname.shift()

It returns the Value of Removed item.

2.unshift => it is used to add element at the first index number and shift all elements upward.

Syntax => arrayname.unshift(‘hello’)

It returns the new length of an array.

Changing elements in array

1.we can also change the items of an array using its index number.

For ex=> arrayname[4] = “Hello”

In this the 4rth index number item is going to remove and replaced by the word hello.

2. We can also add the new item at the end of an array using length method

Syntax => arrayname[arrayname.length] = “hello”

In this case new item is going to add at the end of an array

NOTE=> But instead of using these Methods we can use unshift and push methods because they are more easy to use.

3.DELETE – it is used to delete the elements from an array but it can also cause us some problem so we use pop and push only.

4.concate- it is used to merge to or more arrays together

Syntax- array1.concate(array2);

It does not change the existing array it returns new array.

\*SPLICE AND SLICE

1. Splice – It is used to add or remove the items from an array.

Syntax- var newarrayname = array.splice(p1,p2,”e1”,”e2”)

P1- is the index no. of the element which we have to remove.

P2-is the used to define how many items we need to after the index no. p1 in the array.

E1 and e2 – elements which we need to add or replace.

\*Splice returns the array with a deleted items.

\*It modify the original array.

2.SLICE-> this method is used to slice the parts on an array.

Syntax-var newarrayname = array.slice(p1)

P1- this parameter defines from which index number we need to slice the elements on an array.

Ex- var names = ['DELHI','HARAYANA','BARELLY','DEHRADUN','HARIDWAR','RISHIKESH']

var next = names.slice(1)

console.log(names)

console.log(next)

in this Example all elements from index no. 1 to last index number is going to be sliced out from an array and going to be stored in a new array.

\*slice method does not change the existing array it return the new array with removed or sliced elements.

If we need don’t need to slice full array from p1 to last element we need to add another parameter

Syntax-var newarrayname = array.slice(p1,p2)

From this p1 element array is going to remove the elements till p2 but it not include the p2 elements in the returned array.

// var names = ['DELHI','HARAYANA','BARELLY','DEHRADUN','HARIDWAR','RISHIKESH']

// var next = names.slice(1,3)

// console.log(names)

// console.log(next)

As in the array is going to slice from p1=1=harayana from p2=3=Dehradun

But Dehradun is not going to add in the array.

JAVASCRIPT SORTING OF ARRAYS

1.Sort() => this method is used to sort the strings inside the array in ascending order .

Syntax=> arrayname.sort()

1. Reverse() => this method is used to reverse the array.

Syntax- arrayname.reverse()

3.Compare Function- if we need to sort numbers as strings like this “2” then we need to use compare function because “100” “2” in this 2 is 100 is greater than 2 but when we sort these as a string only 1st alphabet of 100 and 2 is used to determine the highest so 2 is high in compare to 100 if we sort integer as a string that’s why we need to use a compare function to sort the string integers .

Syntax- arrayname.sort(function(a,b){a-b})

Example-

let number = ["45","87","12","67","346","6567","47","-40","-4","0"]

console.log(number.sort(function(a,b){return a-b}))

1. Array filter

Syntax- arrayname.filter();

It returns the new array after performing the operations on a array.

Example-

// let number = [45,87,12,67,346,6567,47,-40,-4,0]

// let number1 = number.filter(function(elem,index){return elem % 2 == 0})

// console.log(number)

// console.log(number1)

4.indexof() =>it is used the find the index number of first available element in an array that you searched.

\*Returns the index number of searched element.

\*If nothing is present that you searched than it return -1.

Syntax- arrayname.indexof(‘name of the string’)

1. lastIndexof => it is also same as indexof but it will search the last available object of the array that you searched.

Syntax- arrayname.lastIndexof(‘name of the string’)

\*Returns the index number of lastly occurred searched element.

\*If nothing is present that you searched than it return -1.

\*MAP AND REDUCE

1.MAP

it is used too call every element of an array using a function.

Syntax- var newarray = oldarray.map(function(Elements,Index NO. ,arrayname){return “expression”})

\*it returns the new array as a return value.

\*this is same as a for each function but it returns the new array without touching the old array and it can be used as a chainable function.

Ex-

let arr = array.map(function(currentelem){return Math.sqrt(currentelem)}).map(function(currentelem){return Math.round(currentelem)}).filter(function(currentelem){return currentelem > 2}).reduce(function(accumulator,currentelem){return accumulator += currentelem})

2.Reduce.

It is used to convert 2d and 3d array into a single dimensional array.

But then is ECMA script 2020 they introduced the new features. That we can perform to find percentage ,average , addition, multiplication etc.

It takes accumulator as a new parameter it accumulate all data again and again as the function is called.

STRINGS IN JAVASCRIPT

Topics

1. Escape Character

2. Finding a string in a string

3. Searching for a string in a string

4. Extracting string parts

5. Replacing String content.

6. Extracting string Characters

7. Other useful methods

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S | A | M | E | E |  | S | I | N | G |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

INDEX NUMBERS OF STRING

1. Escape character or backlash character(\).

\*.length is used to find the length of a string.

\*length is not a function it is a property.

Escape character -> if we need to write a double quotes in any string word then we cannot use double quote inside the double quote it gives us error so we use escape character to use double quote inside console.log(“”)double quotes.

Ex-> console.log(“My name is \“Sameer\””)

Output-> My name is “Sameer”

But there is also a simple way to use double quotes inside a double quotes.

\*Using inverted quotes.

\*If we need to use double quotes in the output string then we can single quotes on for creating the string.

Or vice versa

Ex-

1.Console.log(‘My name is “Sameer” ’)

Output – My name is “Sameer”

1. Console.log(“My name is ‘Sameer’ ”)

Output- My name is ‘Sameer’

2.Find A string in A string.

* Indexof method

This method is also same as array index of it also search the string inside an string to the first occurrence of string.

Syntax- let newvar = oldvar.indexof(“string name”,p1(optional))

P1-> is the parameter from which index no. we need to start out search going forward in the string.

\*returns the index number of the string first letter position.

\*if not found the similar string than returns --1.

\*case sensitive.

* lastIndexof method

This method is also same as array index of it also search the string inside an string to the last occurrence of string.

Syntax- let newvar = oldvar.lastIndexof(“string name”,p1(optional))

P1-> is the parameter from which index no. we need to start out search from last index no. to first index no. of the string

\*returns the index number of the string first letter position.

\*if not found the similar string than returns -1.

\*case sensitive.

3.Searching in strings.

\* search method

It searches for a value(string) and returns

The position of the string.

\*if this did not found any string similar it returns -1.

\*it did not take parameters like indexof and lastindexof.

Syntax-> array = newarray.search(“string”)

1. Extracting Strings part.

There are 3 methods for extracting strings in javascript.

1.slice(start,end)

2.substring(start,end)

3.substr(start,length)

1. Slice(start,end)

It is used to extract a part of an string and return the extracted part of an string.

Syntax-> new var = oldstring.slice(p1,p2)

P1->from where we want to start the extraction.

P2->where we want to end the extraction of an string but this part is not included in the returned extracted value.

Ex->

let MYintro = "My name is sameer singh i am doing nothing right now sameer"

let myname = MYintro.slice(11,16)

console.log(myname)

in this index 16 is not included in the return value which is r of string Sameer.

\*in this original string does not affected by the operation.

\*returns the extracted part of an string.

BUT THERE ARE SOME CASES IN THIS STRING SLICE.

Case1- if we put any negative value in the

P2 then it will remove the value

From the last index number of the

String.

Example->

let MYintro = "My name is sameer singh i am doing nothing right now sameer"

let myname = MYintro.slice(0,-4)

let myname2 = MYintro.slice(8,-10)

console.log(myname)

console.log(myname2)

output->

My name is sameer singh i am doing nothing right now sa

is sameer singh i am doing nothing right

case2-> if we don’t want to write a p2 then its ok .it is optional.

2.substring->

It is similar to slice but it does’nt take any negative values in the p1 .but if you give the negative value in p2 then it will return the string from the starting of the string.

3.substr=>this is also similar as slice and substring but it takes p2 as a length of string and return the new string as a result.

Syntax->old str = newstr.substr(4,10)

4.Replacing String Content.

\*Replace method

It is used to replace the strings with another string.

Syntax-> str = newstr.replace(p1,p2)

P1= The string we want to place.

P2=new data which we need to add in the

Replaced string.

\*it returns a new string it does not effect the source string.

\*by default it only replace the first found similar string

\*it is case sensitive.

\*You can use replaceAll() method to replace all the similar string in the string.

1. Extracting String characters.

There are 3 methods to extract string characters.

1. charAt().
2. charcodeAt().
3. Property acces[].

1.charAt=>this is used to extract the character from a string for the specified

Index no.

Syntax- newstr.charAt(index number)

\*it returns the extracted character.

\*it only takes single parameter.

2.charCodeAt()-> this is used to extract the Unicode of a character for the specified index number in the string.

Syntax- newstr.charCodeAt(index number)

\*it returns the extracted character.

\*it only takes single parameter.

\*it return the Unicode value of U-16.

3.property access []

Syntax- stringname[index number]

\*when we use “[]”these in array these are called property access.

2.other usefull methods

1. Stringname.toUpperCase()- to convert full string into upper case.
2. Stringname.toLowerCase()-to convert full string to Lower case.
3. Concat()-str1.concat(str2)or {str1 + str1}
4. Trim()-> trim remove white spaces from bothsides of the string.
5. Split()->convert string into an array. Str.split()we can use comma (,) | and “”

DATE & TIME IN JAVASCRIPT

1. Date Methods(get and set)

2. Time methods(get and set)

\*Java script date objects represent single moment in time in a platform independent format.

\*Date object contains a number that represents milliseconds since

1 January 1970.

Date Methods (get and set)

\*\*\*\*\*\*\*\*\*Creating date objects\*\*\*\*\*\*

There are 4 ways to create date objects.

1.new Date()

2.newDate(year,month,day,hours,minute,seconds,miliseconds)//it takes 7 arguments.

3.new Date(Milliseconds)//we cannot avoid month section.

4.new Date(Date String)

1.new Date()-> This object Creates the present date and time from the 1 January 1970 in the standard TZ standard.

“2022-06-02T14:43:07.630Z” like this.

To convert this into the readable or Indian standard time zone we can use two methods.

1.toLocaleString() -> “2/6/2022, 8:15:04 pm”

2.toString()->Thu Jun 02 2022 20:15:51 GMT+0530 (India Standard Time)

3.there is one more method for to get the milliseconds passed since 1 january 1970

By using Date.now().

2.newDate(year,month,day,hours,minute,seconds,miliseconds)//it takes 7 arguments.//we cannot avoid month section.

This is used when we do not need the present running time. Using this method we can create an time and date object that in which we can set our own time.

Syntax-> new Date(year,month,day,hours,minutes,seconds,milliseconds)

var hd = new Date(2000,11,12,8,50,45);

console.log(hd.toLocaleString())

output-

A:\java script>node p1.js

12/12/2000, 8:50:45 am

\*month parameter must be included in this method if we do not pass month in the parameter than it will give the date 1 January 1970.

\*java script counts the months from 0 to 11.

3.new Date(Date String)->in this we can convert the string into the date and time.

Syntax-new Date(“October 1”)

Ex-

var hd = new Date("October 1,2022 11:46:50")

console.log(hd.toLocaleString())

output-

A:\java script>node p1.js

1/10/2022, 11:46:50 am

4.new Date(milliseconds)

We get the date by passing milliseconds as a parameter.

Ex-

var hd = new Date(5646887646487)

console.log(hd.toLocaleString())

output-

10/12/2148, 5:24:06 pm

\*\*\*\*\*\*\*\*\*GET AND SET METHODS\*\*\*\*\*\*\*

1.get and set methods for date.

\*varname.getFullYear

\*varname.getMonth().

\*varname.getDay().

Set methods

\*varname.setFullYear(2005). ().//you can also pass the month and date in this otherwise it can set it automatically. (optional)

\*varname.setMonth(6).

\*varname.setDay(25).

\*\*\*\*\*\*\*\*\*GET AND SET METHODS\*\*\*\*\*\*\*

1.get and set methods for time.

\*varname.getTime()//it returns time in milliseconds from 1 january 1970.

\*varname.getHours().

\*varname.getMinutes().

\*varname.getSeconds().

\*varname.getMilliseconds().

Set methods

\*varname.setTime()//it returns time in milliseconds.

\*varname.setHours().

\*varname.setMinutes().

\*varname.setSeconds().

\*varname.setMilliseconds().

if we need only time as an out put we use

toLocaleTimeString(). Method

if we need only date as an output we use

toLocaleDateString(). method

Math Objects in java script

1. Pi=> Math.pi//returns Value of Pi.
2. Power-> Math.pow(x,y)// returns x to the power y.
3. Square root-Math.sqrt(x)//returns the value of squareroot of x.
4. Roundoff-> Math.round(x.xxx)//returns the nearest integer value of decimal value.
5. Absolute –Math.abs(x)// always return the positive value of x if value is wheather positive or negative.
6. Math.ceil(x.xxx)//returns the Incremented Rounded value to the nearest integer value of decimal value.in Math.round() it determine the value of nearest integer depends on if the decimal valur is greater or equal to five. But in ceil method it doesn’t what the decimal value is it returns the incremented value example- Math.round(4.11) is 4. But in Math.ceil(4.11) is 5.
7. Math.floor(x.xxx)- this is vice versa of Math.ceil() it returns the decremented value of roundoff method.
8. Math.max or min()- Returns the minimum and maximum numbers in intergers.
9. Math.random()-returns the random value between 0 and 9.//Multiply by zero first.
10. Math.trunc(x.xx)-it returns the integer value of decimal value(before decimal value)//it doesn’t matter positive or negative.

Window object, DOM, BOM

DOM-DOCUMENT OBJECT MODEL

\*Windows vs Documents.

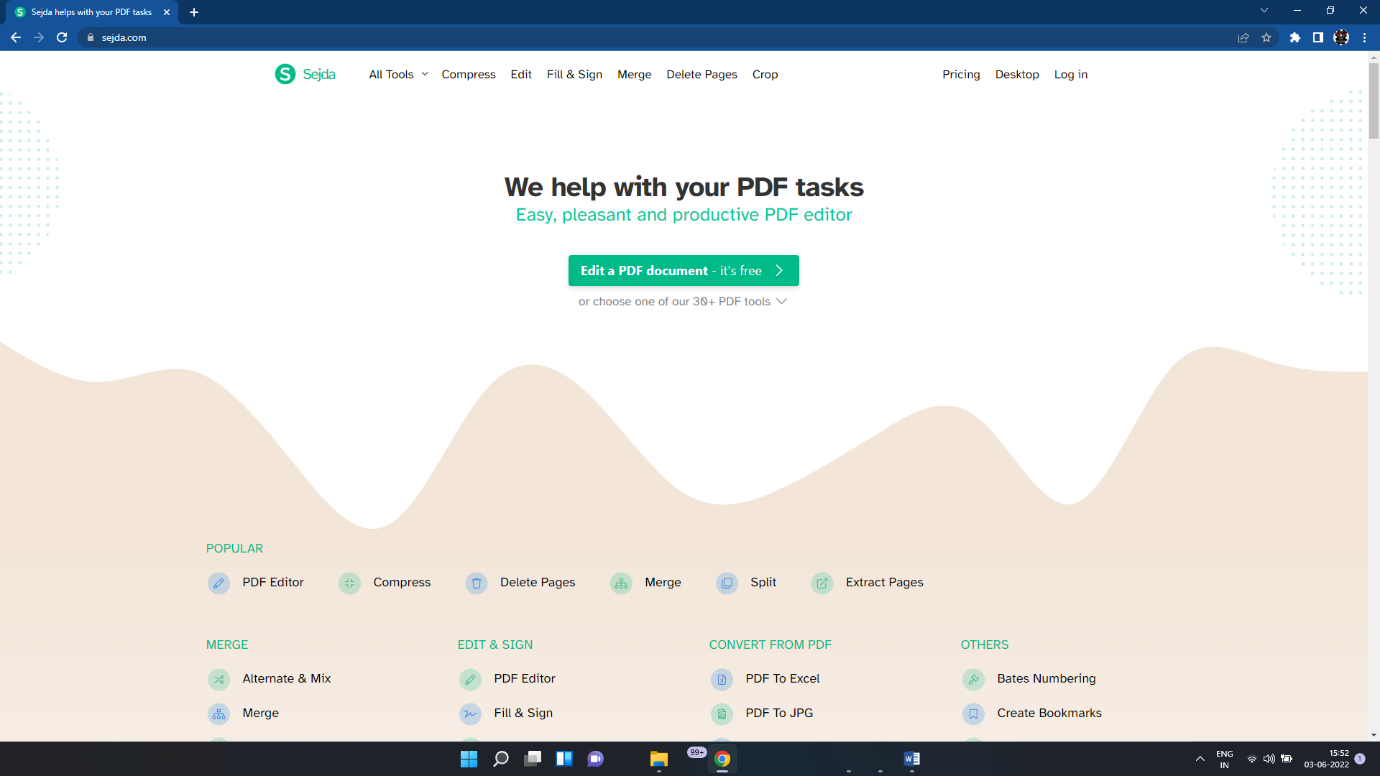
\*DOM vs BOM.

\*DOM Navigation

\*Searching and getting elements reference.

Window object vs DOM.

Window Object->



This whole area is called window object.

\*it contain tabs section, refresh button, backward , forward , etc.

\*Window object is a main container or a global object. Any task related to entire browser window is a part on window object.

\*ex-url, history , navigation , backward button , refresh button etc.

\*it is a parent of DOM .DOM is a child of a Windows Object.

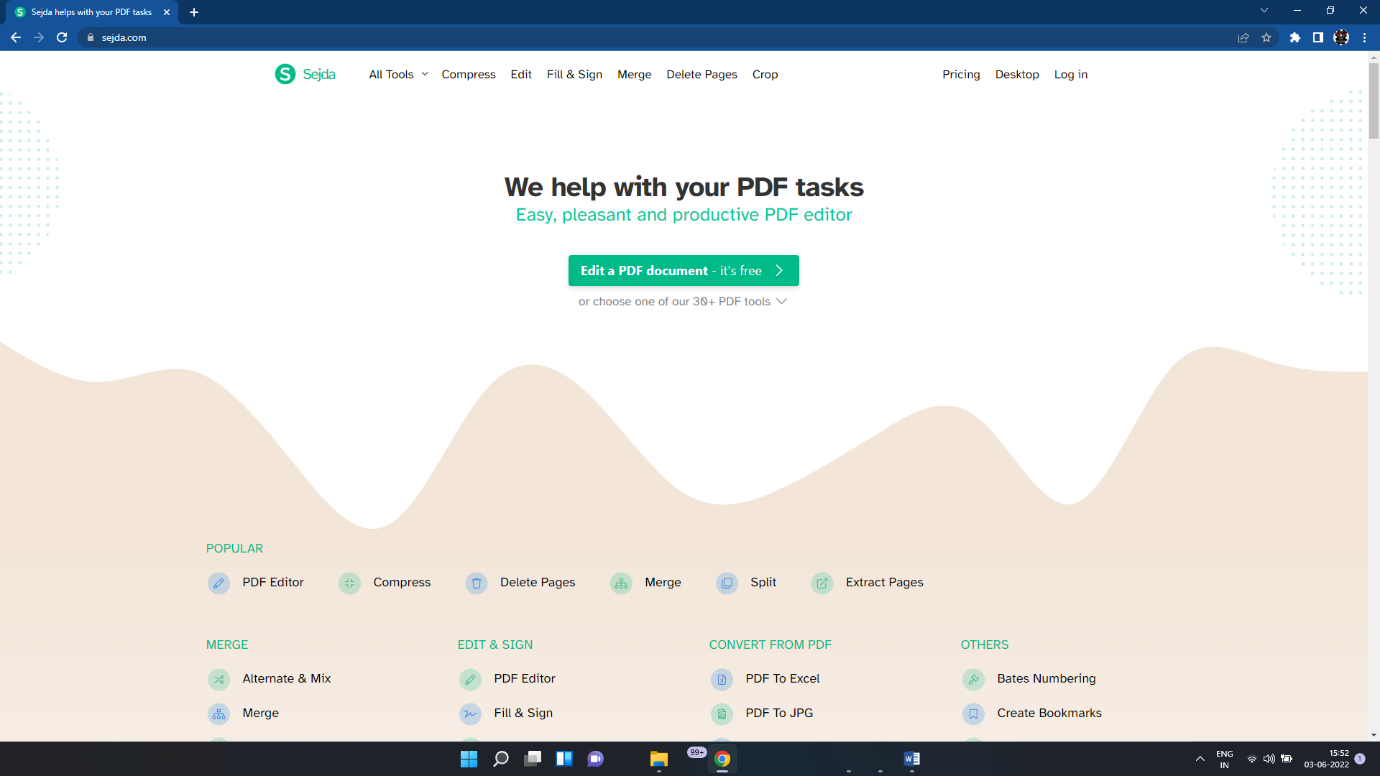
\*All the members like Objects, Methods and properties if they are a part of windows object then we do not need to call windows object.

Ex- we can write Location.href also we do not need to write Windows.Location.href.

\*windows has methods , properties and objects.

\*Document is also the object of window and it has a screen object describing the physical display.

DOM- Document Object Model.



This part Is called DOM.

\*DOM is the child of windows object.

\*It only deals with the html elements

\*We need to refer to the document object if we want to use the methods ,objects and properties.

EX- document.getElementsById()

Document.getElementsByName()

Etc.

BOM –browser Object Model.

\*it deals with the browser related content like history, navigation, screen , frames, location, etc.

\*DOM NAVIGATION

\*root element of DOM is HTML.

1. document.html ->returns the full html part of webpage.

2.document.title->returns the nodes of title tag.

3.document.body->returns the nodes of body tag .

4.document.head-> returns the nodes of the head tag.

5.document.body.childnodes-> returns the nodes inside the body tag. // also returns text,whitespaces , enter as a node.

6.document.body.children->returns the nodes inside the body tag without returning text ,whitespaces , enter.

7.document.body.firstchild-returns the first child of body tag //if text , whitespace , enter is used after body tag the it returns it as a first child.

8.document.body.lastchild-> returns the last child of body tag //if text , whitespace , enter is used at last body tag the it returns it as a first child.

9.document.body.firstelementchild->returns the first element tag// it doesn’t return if text , whitespace , or enter is used.

10. document.body.lastelementchild->returns the last element tag// it doesn’t return if text , whitespace , or enter is used.

11. haschildnodes()-> it returns if any tag has a child nodes or not.

How to Search the element and the reference of the elements.

1. (#) is denoted for ID by a tag.
2. (.) is denoted by class.
3. Tag name.

\*Search and change Data Using ID->document.getElementsById(“idname”)

\*search and change data using class->document.getElementsByClassName(“classnanme”)

\*search and change data using tag name->

document.getElementsByTagName(“tagname”).

\*search and change data using Name attribute->

Document.getElementsByName(‘name’)

\*query selectors.

1.Query selector – syntax –

Document.querySelector(“# or . or classname”)

In this we don’t need to right full syntax for particular class , id , or tag name .using this we can directly access the html code using (.)

For class , (#) for id , and tag name directly.

\*it return or change the first matched element that you searched.

1. Query selector all

Syntax -> document.querySelectorAll((“# or . or classname)

sss

It returns all the values that is in the same class or by same tag name.

Events In java Script

\*4ways of writing Events in javascript.

\*What is Event Object.

\*MouseEvent Is javascript.

\*keyboardEvenet in java script.

\*InputEvents In java SCRIPT.

4 ways of writing Events In java script.

What is events?

Events are the things that Happen To “HTML”

Elements . when javascript is used in html pages and javascript react to them.

4 ways

1. Using inline events alert();
2. By calling a Function
3. Using inline events
4. Using event listeners

1.Using inline events alert().

In this we don’t need to write another script for using this alert event it only alert us about something like popping up notification.

Example->

<html>

    <body>

        <a href="#" onclick="alert('YO YO HONEY SINGH')">

            <span>

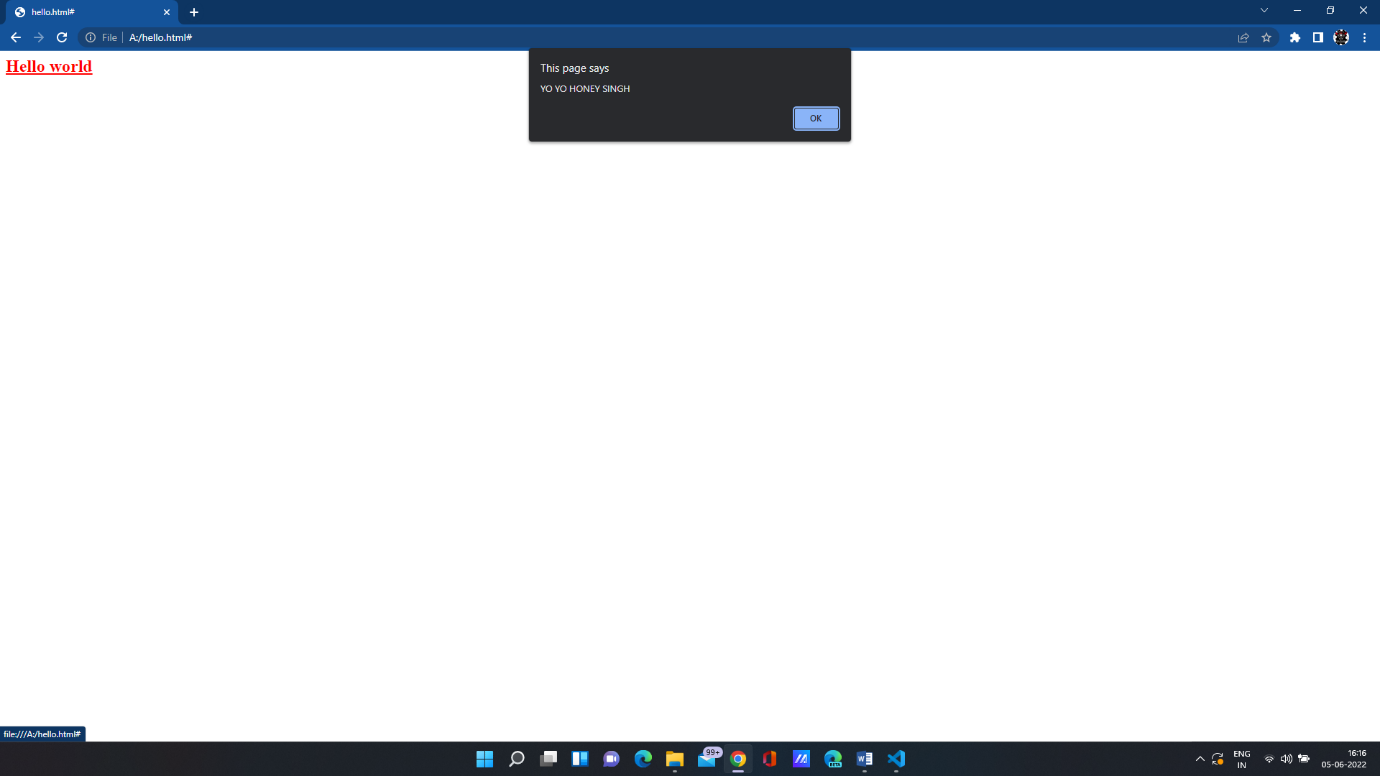
                <h2>Hello world</h2>

            </span>

        </a>

    </body>

</html>



* This alert event is only used for alerts.

2.By calling a Function

In this we can use event by using function.

<a href="#" onclick=onfunctioncall()>

            <h2>

                2nd way

            </h2>

         </a>

And in the script we write the script code of onfunctioncall()

 <script>

             function onfunctioncall(){

                 alert("2nd way of using events by using function")

             }

         </script>

1. Using inline events

In this we use the reference of the id , class , tag name for firing events using DOM searches.

Html code-

<a href="#" id="thirdway" class="third\_Way">

            <h2>

                3rd way

            </h2>

         </a>

Js-

// by using class id

            // const thirdwayy = document.getElementById('thirdway')

            //by using class name

            const thirdwayy = document.querySelector('.third\_way')

            thirdwayy.onclick = function(){

                alert("3nd way of using events by using function")

            }

1. Using event listeners.

Using this event listener we can add the listener it detects that the that the button is clicked or not if it is clicked then it execute the code that we want to write.

Html code->

<button  href="#" id="Fourthwayid" class="third\_Way">

            <a>

                <h2>

                   4th way

                </h2>

             </a>

         </button>

Js code-

const fourthway = document.getElementById('Fourthwayid')

            fourthway.addEventListener('click', function(){

                alert("4rth way of using events by using function")

            })

\*What is Event objects?

Event Objects Are the parents Of event objects. Such as MouseEvents , Keyboard Events, focus Events.

* All the events that occurred on the html using javascript is the children of the Event object.

\*MouseEvents

1. Onmouseup-> when the left button of the mouse is up then it will be excuted.

Syntax-> varname.onmouseup = (any way which you want to write the event)

 mouse1.onmouseup = function(){

              mouse1.style.color = "Blue"

1. Onmousedown-> when the right button is clicked then what happen.
2. Onmouseenter-> when mouse is inside the frame of reference this event is triggered.
3. Onmouseleave-> when the mouse is outside the frame of refrence.
4. Onmousemove->

\*KeyboardEvents

1.onkeyPress() -> Event fire when the key is pressed.

2.KeyUp() -> Event fire when the key is up.

3.keyDown() -> Event fire when the key is down.

\*Onchange events

\*Onchange event occurs when the value of the elements has been changed

\*for radio boxes and checkboxes the onchange event occurs when the check state has been changed.

\*\*\*This is important\*\*

Time events In javascript

* Settimeout(Function,milliseconds)
* setInterval(function,milliseconds)
* clearTimeout()
* clear(interval)

\*SetTimeout(function,Milliseconds)

It executes the function in a time that you specified in milliseconds.

Ex- if we need to execute some function in 1 hour after present time than we use settimeout event.

const Button = document.querySelector('#BTN')

        const Myname = document.querySelector('#Names')

        const Load = document.querySelector('#Loading')

        const ShowMyName = () => {

             Load.innerHTML= "Loading..."

             setTimeout(()=>{Myname.innerHTML ="Sameer Singh"},1000)

        }

        Button.addEventListener('click',ShowMyName);

\*cleartimeout(p1)

To clear time out we just need to take the reference of the SetTimeout function From html Element at put reference into the function.

OOPS IN JAVA SCRIPT

\*What is “Object Literals”?

\*What is “This”objects?

What is “Object Literals”?

\*Object Literals are the key:value data structure.

\*storing variable and functions together.

\*we can also refer this as an object.

How To create an object

1=> 1st way.(no one use this much after es6)

Let person = {

\*\*\*variables\*\*\*

(\*key)F\_Name : “Sameer”(value),

L\_Name : “Singh”,

\*\*\*\*\*methods\*\*\*\*\*\*

(\*key)Getdata : function(){

Console.log(person.F\_Name) ”(value) }

}

\*we need to separate key and value by using (:) this operator.

\*Functions defined inside object is called Methods.

\*variable and function we defined are always separated by (,) comma.

2=> 2nd way.(No need to write functions names after es6).

Let person = {

\*\*\*variables\*\*\*

(\*key)F\_Name : “Sameer”(value),

L\_Name : “Singh”,

\*\*\*\*\*methods\*\*\*\*\*\*

(\*key)Getdata (){

Console.log(person.F\_Name) ”(value) }

}

\*this is same as the 1st way but the only difference is we do not need to create a anonymous function in the value after the key method variable assignment.

Ex- getdata(){//your methods}

\*Can we create the new object inside the object?

Yes we can.

Let person = {

Name:{

//FNAME : “Sameer” },

Age : 45

}

* we just need to use curly braces in the value of the key than new object is created inside the object.
* To access the object inside the object we use- let say person is a parent object and Name is the child object inside the person then we only need to write =>Person.Name.FNAME

\*What is “This” Object?

\*It contains the Current Context.

\*Its Value depends on where it is placed.

Means-> if “This” object is inside any object than we do not need to write the object name to access its values we just use “This” object to access the value of any key inside the current object. For example=>

If we want to access the F\_NAME variable inside the object named by person then if we need to access the F\_NAME by using “person.F\_NAME” we can also access this first name by using “THIS.F\_NAME” but we can only access F\_NAME using “This” object inside the object.

\*if we use “This” Object outside the Scope of Any object then it will access properties and methods of Windows Object.

\*we cannot use fat arrow functions if we need to use “this” object.

\*if we use this inside the function that is not in the object then This object is return the windows object.

\*But we can also access the Variables that is defined in the global scope (not inside any object )

\*Destructuring

1. Destructuring Of array.

2. Destructuring of object Properties(or we can say object variables).

What is the meaning of destructuring?

Destructuring Means Unpacking the values of arrays and object properties.

1.Destructuring Of Arrays.

Syntax-> Array = [“Sameer”,”Singh”,21]

/////Destructuring///

Var [Fname,Lastname,age]=array[]

This Code is Just like=>

Var Fname = Array[0]

Var Lastname = Array[1]

Var Age = Array[2]

\*Using destructuring we can assign the values inside an array to variables in some line of code. Instead of assigning values 1 by 1 we just do it in 1 line of code.

\*we can also add the values during destructuring ex-

Var [Fname,Lastname,age,class=9]=array[]

2.Destructuring of Object properties.

Objects properties destructuring is same as array destructuring but difference is we need to replace [] square brackets by curly braces.

\*and also the variable name of the objects properties is also need to be exact same as detructuring definition.

For example=>

Var person ={

Fname:”sameer”,

Lname:”singh”

}

Let {Fname,Lname} = person

Console.log(Fname)

Console.log(Lname)

Output

Sameer

singh

Objects Properties

1.We can Use Dynamic properties.

2.No need to write key and value if both are same.

We can Use Dynamic properties

What is Dynamic Values?

Any variable which is not inside the object but the name of the variable is same as the key inside the object this is called dynamic data.

\*using square brackets around the key inside the object we can get the data outside the objects if both the key and outside defined variables has same names.

Example.

Var Fname = “Sameer”

Var person ={

[Fname]:”is my name.”,

Lname:”singh”

}

Console.log(person)

Output

Sameer is my name,singh

\*we are Assigning the data of Fname in the key(Fname) inside the Object using Dynamic Property([]).

\*we can also use arithmetic operations inside the objects key using dynamic property

Ex-

[20+6]:”age”

Console.log(person.age)

Output-

26 age

No need to write key and value if both are same.

\*if we have same key and value ex-fname:fname, and the data of the value(right fname) is defined outside the object then we do not need to write both of them we can also write just{fname,}

\*when we put only key value inside the object then the value is also need to be defined outside the object with the same name as key value if not then it will show error.

Example

let Fname = "Sameer"

let Lname = "Singh"

let person = {

   Fname,

   Lname

}

console.log(person)

output-

{ Fname: 'Sameer', Lname: 'Singh' }

As you can see it automatically assign the outside objects data in the values of keys (Fname,Lname) inside the objects.

\*Spread operator(…)

\*let say we have an array called fruit-

Const fruits = [“apple”,”orange”,”mango”];

And we have an another array called favfruits-

Let favfruits = [“banana”,”papaya”,”lichi”]

Let say if we want to add fruits array items in favfruits array then we need to write it like this=>

Let favfruits = [“banana”,”papaya”,”lichi”,“apple”,”orange”,”mango”]

But we can also use spread operator for calling fruit array directly into the favfruit array by justr using the array name like this=>

Let favfruits = […fruits,“banana”,”papaya”,”lichi”]

\*by using …arrayname we can call items of another array in the present array without writing it again and again.

let fruits = ["mango","banana","orange"]

let favfruits = ["lichi","papaya",...fruits]

console.log(favfruits)

output-

[ 'lichi', 'papaya', 'mango', 'banana', 'orange' ]

ECMA SCRIPT 2016/ES6 FEATURES

1:Array.includes.

2:Exponential operator.

\*Array.include()=>

It returns the value of true or false on the basis of the item we search is present or not inside an array.

Syntax=>arrayname.include(//searched item)

let fruits = ["mango","banana","orange"]

console.log(fruits.includes("banana"))

output

true

2.exponential operator (\*\*)

it returns the power value ex-

2\*\*2 is equal to 2\*2

2\*\*4 is equal to 2\*2\*2\*2

6\*4 is equal to 6\*6\*6\*6

ECMA SCRIPT 2017 /EC8

1. String padding
2. Object.values()
3. Object.entries()
4. Async//we cover in further
5. Rest/spread operators for objects.(ECMAScript 2018/ES9)

\*String padding

It is used for padding in the string in start and end

Syntax->varname.padstart(//padding length) ////string length also get included in padding length ex- consider we have a string “Sameer”and we add start padding of 3 then result will not be reflected because length of the string is 5 if we need to add the padding of 3 from start in string we need to give padding length of 8(character of the strings also included in the padding length).

Then we have pdding from end it is same as padding from start

Syntax – varname.padend(//padding length)

const names = "sameer"

console.log(names.padEnd(20));

object.values()

it extract the values inside an object and convert the values into an array.

Syntax->

Let names = Objects.values(objectname)

let person={

   name:"sameer",

   class:"12th",

   age:45

}

console.log(Object.values(person))

output-

[ 'sameer', '12th', 45 ]

Object.entires()

It extract the key and values of an object and convert it into the w dimensional array like

[[key1,value1],[key2,value2],[key3,value3]]

Syntax->object.entries(objectname)

let person={

   name:"sameer",

   class:"12th",

   age:45

}

console.log(Object.entries(person))

output-

[ [ 'name', 'sameer' ], [ 'class', '12th' ], [ 'age', 45 ] ]

Spread operator for objects. .(ECMAScript 2018/ES9)

This is same as arrays if we need to add properties of 1 object into another then we use (…) spread operator.

Example-

let person={

   name:"sameer",

   class:"12th",

   age:45

}

let person2 = {

   name1 :"rahul",...person

}

console.log(person2)

output-

{ name1: 'rahul', name: 'sameer', class: '12th', age: 45 }

ECMASCRIPT 2019/ES10

1:flat()

2:Objects.fromEntries()

3.Stringname(trimStart,TrimEnd)

1.Flat()

It is used to convert any dimensional array into single dimensional by just using flat() method.

\*We can also convert any array into single dimensional using reduce function also but it is very lengthy process.

\*by using reduce function we cannot convert multi level array into single dimensional for ex- arr = =[["A1","A2"],["B1","B2"],["C1",["D1","D2"]]]

let arr =[["A1","A2"],["B1","B2"],["C1",["D1","D2"]]]

let arr1 = arr.reduce(function(Accum,elem){

   return Accum.concat(elem)

})

console.log(arr1)

output-

[ 'A1', 'A2', 'B1', 'B2', 'C1', [ 'D1', 'D2' ] ]

Like this it will not convert array levels

\*this is why we use the flat function to convert the array of arrays into single dimensional.

Ex-

let arr =[["A1","A2"],["B1","B2"],["C1",["D1","D2"]]]

let arr1 = arr.flat()

console.log(arr1)

output-

[ 'A1', 'A2', 'B1', 'B2', 'C1', [ 'D1', 'D2' ] ]

\*as you can see the output is same as reduce function. nut if we pass parameter inside the flat object then it will convert to that level and if we pass infinity it doesn’t matter how many levels array has it simply convert into 1darray.

Objects.fromentries()

This is Vice Versa of Objects.entries()

\*Objects.Entries() convert objects into 2d arrays and Objects.fromEntries() convert 2d arrays into objects.

Syntax- objectname.Object.Fromentries(arrayname)

Stringname(trimStart,TrimEnd)

\*it is used to trim the padding of the string.

ECMASCRIPT 2020

1.Bigint Data Type

\*Maximun value of a integer that we can use in java script is 9007199254740991 . js cannot handle the value above this.

\*Because of this problem Bigint is introduced in the java script.

\*if we simply put (n) after the integer than it will become the Bigint datatype

Ex – let a = 4564n;

Then this a became the BigInt data type

let a = 45n;

console.log(typeof(a))

output-

bigint

using bigint datatype we can handle the value above the max interger calue in js

let a = Number.MAX\_SAFE\_INTEGER

console.log(9007199254740991n + 4000n)

output-

9007199254744991n

ADVANCE JAVASCRIPT PART=1

1. Event propagation(Event Bubbling and Event Capturing)
2. Higher Order function
3. Callback function
4. Function currying(we will se after Async JS section)
5. CallBack Hell
6. AJAX call using XML Httprequest
7. BONUS Section JSON
8. Fetch API
9. Promises
10. Async – Await
11. Error Handling in JS.

Event propagation(Event Bubbling and Event Capturing)

\*What is Event Propagation?

=>Event propagation determines in which order the event is Handled by the elements.

(From Top to bottom or from bottom to top)

\*what is event capturing?

=> when the button is clicked it propagates event from outermost element(HTML) to innermost element.

Or

When the event is first captured by the outermost element(html) and then propagates to the innermost element

\*What is Event Bubbling?

=>When the button is clicked it propagates the event from innermost element to outermost element(HTML)

Or

Event is first captured and handled by the innermost element and propagates to the outermost element(html)

\*What is bubble phase?

=>event propagates from target phase to windows phase.

\*what is capture phase?

=>event propagates from windows to target phase.

\*what is target phase?

=>element where the event is fired.

Example of Event Bubbling

<html>

    <body>

         <div id="Div1">

             "hello"

             <div id="div2">

                 Hello2

             </div>

         </div>

    </body>

    <script>

        let ParentDiv = document.querySelector('#Div1')

        let ChildDiv = document.querySelector('#div2')

///////////event bubbling////////

        ParentDiv.addEventListener('click',()=>{alert("Parent")})

        ChildDiv.addEventListener('click',()=>{alert("child div")})

    </script>

</html>

\*In this when we click on the Hello2 then first the event of child div is fired then event of div1 parent div is fired and hello2 is out target phase.

\*The order of propagation is Innermost Element (div2) < Outtermost elemet(div1) < O.E(body) < O.E(Html)

\*as you can see we did not done anything different in the event listener function because event bubbling is default if we did not pass third argument in the addeventlistner.

\*as you can see the event bubbling is set as a default property so if we need to stop event bubbling we simply use event.stopPropagation()

Example of Event Capturing

<html>

    <body>

         <div id="Div1">

             "hello"

             <div id="div2">

                 Hello2

             </div>

         </div>

    </body>

    <script>

        let ParentDiv = document.querySelector('#Div1')

        let ChildDiv = document.querySelector('#div2')

///////event Capturing/////////

        ParentDiv.addEventListener('click',()=>{alert("Parent")},true)

        ChildDiv.addEventListener('click',()=>{alert("child div")},true)

    </script>

</html>

\*in this as you can see there is 3rd argument we passed on addeventlistner which is true.

\*the third argument of addeventlistner is true or false.

\*in false state it triggers event bubbling(by default it is always set to false and this is why event bubbling is default)

\*in true state it triggers the event capturing.

\*in this example when we click the target phase which is hello2 , first the event of outermost event is handled(div1)and then it will go to innermost element event (div2)

Higher Order function

\*What is Higher Order Function?

=> function which takes another functions as an argument is called higher order function.

Ex- we have a two functions add() and sum() and we have another function call all()

If we pass function add() as an argument to the function all like all(add) the this all() function is called higher order function.

Callback function

* Function which is passed as a argument in the another function is known as callback function.

In the above example all() is a higher order function and add() and sum() which is passed as an argument in the another function is called callback function.

const Add = (a,b)=>{return a+b}

const Sub = (a,b)=>{return a-b}

const Mul = (a,b)=>{return a\*b}

const Div = (a,b)=>{return a/b}

const Calc = (num1,num2,opperator)=>{return opperator(num1,num2)}

console.log(Calc(5,78,Add))

console.log(Calc(5,78,Sub))

console.log(Calc(5,78,Mul))

console.log(Calc(5,78,Div))

Calc is a higher order function because it is taking add(),sub() etc as a argument.

Add(),sub() etc are call back functions because they are passing as a argument inside a function.

We will learn another subtopics after learning asyncroneous javascript.

How javascript works and Asyncronus java script. (1.1)

* Hoisting in java script.
* Scope chain
* Lexical Scoping in java script
* Use Strict Mode
* This Keyword or This Object (we already done it previously)
* Closures in javascript
* What is Asynchronous js programming
* What is Event loop?

Hoisting in java script.

* There are two phases in javascript creation phase and execution phase
* Creation phase = in which we create our code.
* Execution phase = In which the code is executed.
* Hoisting = is a mechanism where all the variables and function declaration comes to top of the scope during execution.

Ex-

Creation phase- console.log(name)

Var name;

Name = “sameer”

Output – undefined

So during execution face it will execute like this- line 1= var name;(all variables and fuction declarations comes at the top)

Line 2 = Console.log(name)

Line 3 => Name = “sameer”

Output-

Undefined

First line 1 declaration comes at top because of hoisting.

Then line 2 and 3 will be executes.

* To tackle the problem of hoisting we use let and const instead of var.
* In let and const we cannot perform any task without declaring variable this will give us error.

Scope Chain and lexical scoping in javascript.

* Scope chain = scope chain is used to resolve the value of variable names in js.
* Lexical Scoping In javaScript.

Lexical scoping in java script means child function of the parent function can acces the variables of parent function but not vice versa.

Or

Inner functions can access the variables inside their parent function but vice versa is not true.

Example-> let a = 45;

const Parent = ()=>{

   let b = 78;

   const child = ()=>{

           let c =45;

           console.log(a+b+c)

   }

   child()

}

Parent()

* In this we can access the variable b in child function which in inside the parent but we cannot access variable c in parent function which is declared inside child function.
* If we do so then it will give error it is called lexical scoping in javascript.

Closure in java Script.

* This is same as the lexical scoping
* The innermost functions can access to the variables to the outermost function variables and parameters but vice versa is not true.
* Closures in java script is created when the function is created.

Another example of closures.

const outermost = (a) =>{

        const innermost = ()=>{

                let b = 5;

                console.log(a+b)

        }

        return innermost // return the function means it goes out from the parent function or now not the child of the parent function

}

const checkcloser = outermost(6)

checkcloser()

output

11

* When we use return on the function then it is not the part of the parent function.
* Then why after getting out from the parent functions this innermost function still accessing the variables and parameters of parent function.
* It is because when we create functions the closure is also created and the values of parent functions are getting stored inside the closure of the child object.

Question – Difference Between Synchronous and Asynchronous In java Script?

Ans=> Synchronous Functions = > The Code is executed one by one .if 2nd line of code didn’t execute till 1st line of code did not completed.

No matter how much time it take.

Example-const Synch2 = ()=>{

       console.log("Work 2 ")

}

const Synch = ()=>{

        console.log("Work 1")

        Synch2()

        console.log("Work 3")

}

Synch()

Output-

Work 1

Work 2

Work 3

Asynchronous Function => It executed the code immediately didn’t wait to execute line 1 if it taking some time it execute line 2 and wait for line 1 in backend.

Example-

const Synch2 = ()=>{

        setTimeout(function(){

                console.log("work 2")

        },2000)

}

const Synch = ()=>{

        console.log("Work 1")

        Synch2()

        console.log("Work 3")

}

Synch()

Output-

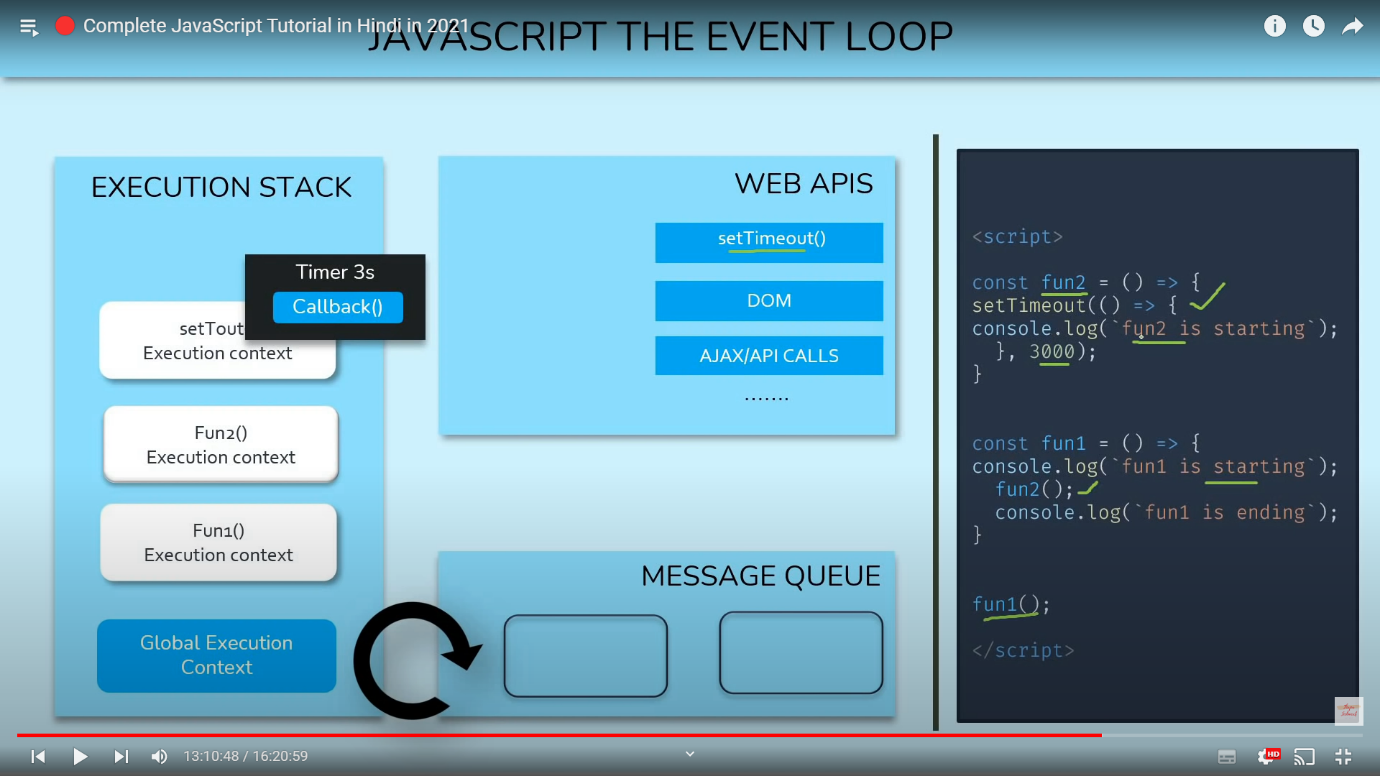
Work 1

Work 3

work 2

(We will learn about what javascript do in backend in the event loop)

Event Loop



Execution stack=>when the function is called it will go to the execution stack until work of the called function is completed.

WEB APIs=> when setTimeout Function is called it first go to the Execution stack and then set the callback timer of 3 seconds and then stored inside the web api and removed from the execution stack for further operations.

Message queue=>when 3 seconds is completed then the call back function is moved into the message queue and then if execution stack is empty then This small Rounded arrow called event loop is executed and pass the call back function in execution stack and then settimeout function is executed and removed from execution stack this is how asynchronous functions work .

ADVANCE JAVASCRIPT PART=1 again.

1. Function currying(we will se after Async JS section)
2. CallBack Hell
3. AJAX call using XML Httprequest
4. BONUS Section JSON
5. Fetch API
6. Promises
7. Async – Await

Function Currying=>Currying is a technique to Evaluate or convert Functions with multiple arguments into the sequence of functions taking single arguments ex-

Sum(4,5,6,7) into Sum(4)(5)(6)(7)

In other words the instead of taking all arguments at ones first one take argument and return the new function to second and then second take argument and return the new function to third it goes on until all arguments are fulfilled.

Using Traditional Way of function

function Sum(num1){

        return function (num2){

               return function(num3){

                       return function(num4){

                           console.log(num1+num2+num3+num4)

                       }

               }

        }

}

Sum(4)(5)(6)(7)

Using Fat arrow function

const Add = (A)=>(B)=>(C)=>(D)=>console.log(A+B+C+D)

Sum(8)(9)(10)(11)

Fat arrow is more easy to write I preffer fat arrow.

CallBack Hell

When we call a high number of functions inside a function then it will look very confusing and not look so good is called callback hell to deal with this type of problem we use promises we study about this latter

Ex-

function Sum(num1){

        return function (num2){

               return function(num3){

                       return function(num4){

                           console.log(num1+num2+num3+num4)

                       }

               }

        }

}

It is shaped like a triangle what happen if we have 20 or 30 function inside it .

AJAX Call Using XMLhttprequest

Fetch api using Ajax call.

API- Application programming interface.

* First WE need to create a instance like const varname = new XMLHttpRequest()
* Then we need to call , varname.open(‘GET’,”API LINK”)

GET is used to fetch the api.

In this We Created The request for the api to server.

* Then by using varname.send();

Using this send we send the request if if everything is write don’t get anny error at our console in chrome.

* Inside any event we use. This.responsetext to get the data inside an api.
* If the data inside the api is in JSON(Javascript Object Notation) form then we convert this into javascript readable form using JSON.parse(this.responseText)
* Then this data is converted into Javascript readable format.

Fetch API using Fetch() and Handle It using Promises.

* What is promises?

When we use fetch() function to call an API it returns Us a Promise in this promise we get 2 things. 1. If the data that we need from API is get to us Than it returns fulfilled means we get the data. 2. If we didn’t receive the data because of anything we get reject

Example-

<script>

        const Jokes = document.querySelector('#Jokes')

        const JokesBtn = document.querySelector('#JokesBTN')

        const JokesGenerate = ()=>{

            const setheaders = {

                headers : {

                    Accept : "application/json"

                }

            }

            fetch('https://icanhazdadjoke.com', setheaders)

            .then((response)=>response.json()).then((data)=>{

                Jokes.innerHTML = data.joke;

            }).catch((error)=>{

                console.log(error)

            })

        }

       JokesBtn.addEventListener('click',JokesGenerate)

       JokesGenerate()

    </script>

* In this fetch() function we don’t need to write get it automatically get the api
* Then this fetch() function returns us fulfilled(means api data recieved) or Rejected (means api not received due to some internet or api issue)
* After this we use .then() function and than we create anonymous function inside the then function and pass response(this variable store the data of promises we get by fetch()) then if our api is received without any error then we continue our code further.
* To detect the error in the api we just need to add .catch() function and the end of our program and pass the argument error(this variable stores because of ehich error out code is not run) in it and then inside catch anonymous function we just use console.log(error) to get error details.

Fetch Api using Async and Await.

 const Jokes = document.querySelector('#Jokes')

        const JokesBtn = document.querySelector('#JokesBTN')

        const JokesGenerate = async ()=>{

            const setheaders = {

                headers : {

                    Accept : "application/json"

                }

            }

          try{

     const response = await fetch('https://icanhazdadjoke.com',setheaders)

            const Data = await response.json()

            Jokes.innerHTML = Data.joke

            }catch(err){

            console.log(err)

           }

}

       JokesBtn.addEventListener('click',JokesGenerate)

       JokesGenerate()

* We just need to write async key word in front of the function definition to convert it into the asynchronous function – ex

In traditional function

async function\_name(){//code};

In fat arrow function

Const function\_name = async ()=>{//code}

* we also use fetch in this async function and it also returns the promises.
* But instead of passing it as an argument in then() function we just wait for the api to receive the promises and then we store the response inside the user defined variable and for wait we just need to write await keyword in beginning of the fetch function(means wait until the response is received)ex=

Const res = await fetch(“API LINK”)

* Then we convert the data into json format using res.json() and store it into the variable we also use await keyword in this also.

Local Storage in browser,

Use

localStorage.set() to set data

localStorage.get() to get data

<https://mikkegoes.com/javascript-projects-for-beginners/>

<https://skillcrush.com/blog/projects-you-can-do-with-javascript/#clock>