**JavaScript Notes**

1. **Important Methods of Strings in JS:**
2. .length method: To find the length of a string we use .length method.

For eg: let firstname = "Harshithdhdhdhkdhkhdkwhdkwkdhkwdkwhdkhwdk"

console.log(firstname.length)

To find the last letter of the string we use :

console.log(firstname[firstname.length-1])

1. Trim(): If you are having a string and there are some spaces in starting and some in end of the string then we can use trim() method. In this method either we can assign the trim() value to a new object or we can reiterate the same value.

For eg: let firstname = "      ankit     "

console.log(firstname.length)

 lastname = firstname.trim()

 console.log(lastname)

 console.log(lastname.length)

1. toUppercase (): This converts all the characters in a string to uppercase letters. These methods will not change the existing string but will give a entirely new string.

For eg: let firstname = "ankit"

firstname = firstname.toUpperCase()

console.log(firstname)

1. toLowercase(): This converts all the characters in a string to lowercase. These methods will not change the existing string but will give a entirely new string.

For eg: let firstname = "ankit"

firstname = firstname.toLowerCase()

console.log(firstname)

1. slice(): If we want to get selected portion of the string as per our requirements, then we can use the slice() method . In this first we write (start index , stop index).

For eg: let firstname = "ankit"

first\_name = firstname.slice(0,5)

console.log(first\_name)

1. To convert a number to a string we can add the number with an empty string.

For eg: let var1 = 23;

var2 = var1 + "";

console.log(var2);

console.log(typeof var2);

1. Agar aapko string ko number mein convert karna he toh string se pehle “+” laga dena he wo string number mein convert ho jaayegi.

For eg: let var1 = +"123";

console.log(var1);

console.log(typeof var1);

**Template Strings: Template Literals** use back-ticks (``) rather than the quotes ("") to define a string:

With **template literals**, you can use both single and double quotes inside a string:

**Template literals** provide an easy way to interpolate variables and expressions into strings.

The method is called string interpolation.

The syntax is:

${...} For eg :

let age = 22

let name = "Harshit"

console.log (`My name is ${name} and my age is ${age}`)

# Type of undefined is undefined but type of null is object and this is a bug in js.

#BigInt expression: There are two ways to convert a number or integer to a bigInt .

1. Number1 = BigInt(23);
2. Number\_2 = 23n;

#**Falsy Values:**

1. These values will always give false when treated. These are of different types:
2. False
3. Null
4. Undefined
5. “ ”
6. 0

For eg: let firstname = "";

if(firstname){

    console.log(firstname);

}

else{

    console.log("This is a falsy value")

}

# **Ternary Operator:**

It is very useful and similar like if and else statements but is very short in code.

For eg:

let age = 2

let drink = age>=5 ? "beer" : "milk"

console.log(drink)

**# And / Or operator:**

1. For and (&&) condition to be true left-hand-side as well as right-hand-side should be equal.
2. For or (||) condition to be true left-hand-side or right-hand-side should be equal, any of one condition should be true.

**# Nested-if-else:**

let winning\_number = 19

let guess\_number = +prompt("Enter your guess number")

if (guess\_number === winning\_number){

    console.log("Your guess is right")

}

else{

    if(guess\_number<winning\_number){

    console.log("too low!!!!")

    }

    else{

    console.log("too high")

    }

}

**#** Multiple variables aapas mein linked hein unko object kehte hein. Object ke pass dono behaviour or properties hoti hein.

# Falsy or truthy values mein hamesa truthy value jeetti he.

# **OR** Operator – Short circuiting:

1. It is a concept in js in or operator where suppose we have a falsy value and then truthy value and then truthy value so as soon the or operator gets the first truthy value it will stop the execution and ignores the remaining truthy values.

For eg: ( false || 5 || “ankit”)

Output : 5

1. If in And operator (&&) we get a falsy value and a truthy value, it will give always falsy value as a output if.

**#Important Notes:**

1. Function ek esa program he jo ki ek single well defined task karta he.
2. Agar aap koi function object ke andar create karte hein toh usko method kehte hein.

**#Object-Creation How?**

1. Factory function: Factory function ek esa function hota he jo ki ek object banata he or usko return karta he.

For eg : function createrec(x,y){

    return rectangle = {

        length : x,

        breadth : y,

    draw() {

        console.log("drawing rectangle")

        }

    };

}

let rectobj = createrec(5,3);

let rec2 = createrec(3,4);

let rec3 = createrec(7,8)

console.log(rec2)

console.log(rec3)

console.log(createrec)

console.log(rectobj)

# **This** keyword represents our current object. Jis bhi aap current object par kaam kar rhe hote ho uske ye refer karta he.

# **New** ek esa keyword he jo ki ek empty object create karta he.

1. **Constructor Function**:
2. This function is represented in Pascal notation, where first letter of every word is capital.
3. In constructor function “this” keyword is used where “this” keyword represents the object on which we are working currently.
4. Constructor function humare properties ya methods ko define/ initialize karta he.
5. Yahan par object ko return karne ki jarurat ni padti he.
6. function Circle(r,d){
7. this.radius = r,
8. this.diameter = d,
9. this.area = function () {
10. console.log("Find area of circle")
11. }
12. }
13. let circleobj = new Circle(2,3);
14. console.log(circleobj)

**#Dynamic Nature of Objects:**

1. Kisi bi object ke andar insertion or deletion dono possible he.

For eg: function Createrec2(len,bre){

    this.length = len,

    this.breadth = bre,

    this.draw = function(){

        console.log("")

    }

}

let createrect3 = new Createrec2(3,4);

console.log(createrect3)

// Dynamic bature of objects

// Adding new properties to a Object

createrect3.color = "yellow"

console.log(createrect3)

delete circleobj.color

//  console.log(circleobj)

**# Functions are Objects:**

1. Function bi ek object hote hein cz uski bi property hoti he or method hote hein.
2. Primitive data types mein copy banti he jabki reference types mein same address par point karte hein.

**#Note:**

1. Primitive data types are copied by there value.
2. **“Primitive wale case mein ek alag se copy create hoti he.”**
3. Reference data types are copied by there address/references.
4. **“Reference wale case mein same address ke upar multiple variables point kar rhe hote hein.”**
5. Jab kisi primitive data type ko aap kisi function mein pass karte hein toh uski copy banti he.
6. Jab kisi reference data type ko aap kisi function mein pass karte hein toh uska address update ho jaata he.

For eg: let a = 10

function inc(a){

    a++

}

inc(a)

console.log(a)

**Output** – 10 ( in this scenario a ki nayi copy banti he jisme a ki value store hoti he isliye iski value same rahegi value mein koi change ni hoga).

let a = {value:10}

function inc(a){

    a.value++;

}

inc(a)

console.log(a)

Output – 11 (In this scenario a usi same address par jakr update ho jaata he isliye uski value ek increment ho jaati he because koi copy banti ni he).

**#For-In Loop:**

1. With the help of for-in loop we can access the keys, values as well as the key-value pairs from a object.

For eg: let obj1 = {

    length : 10,

    breadth : 20,

}

for (let key in obj1){

    console.log(key,obj1[key])

}

**#For-Of Loop:**

1. For-of loop can only be used with the iterables. Iterables are arrays, objects etc.

For eg: let obj1 = {

    length : 10,

    breadth : 20,

}

for(let key of Object.entries(obj1)){

    console.log(key)

}

**# To check whether a property is Present in an Object or Not:**

1. To check if a property is present in an object or not we use if- else statements with (in) keyword.

For eg: let obj1 = {

    length : 10,

    breadth : 20,

}

if ('breadth' in obj1){

    console.log("Present")

}

else(

    console.log("Absent")

)

**\*\*\*Very Important Qusetions For Interview:**

In JS we can have three methods for object cloning. They are as follows:

1. Iteration

For eg: let src = {

    a : 10,

    b : 20,

    c : 30,

}

dest = {}

for (let key in src){

    dest[key] = src[key]

}

console.log(dest)

src.a++

console.log(dest)

Source ki agr hum a value ko increment karte hein toh destination par koi fark ni pad rha he toh humlog object ko clone karne mein successful ho gye hein.

1. **Assign operator:**
2. Mutliple src or object ki properties humlog single object mein assign kar sakte hein using assign operators.

For eg:

let src = {

    a:10,

    b:20,

    c:30

}

let src2 = {value:25}

let src3 = {value1:30}

let dest = Object.assign({}, src, src2, src3)

console.log(dest)

src.a++

console.log(dest)

3.Spread operator: ( …)

let src = {

    a:10,

    b:20,

    c:30

}

let dest = {...src}

console.log(dest)

src.a++

console.log(dest)

**Garbage Collection**:

1. It finds us about variables/constant which are not in use or inki memory ko wo deallocate kar deta he or inki memory ko free karwa deta h. Jahan par automatic memory free ho jaati he humein alag se nah karni pade.
2. We have no control over garbage collector when to start or when to stop.
3. It runs in the background.

**#In-Bulit Objects And Arrays:**

1. **Math:** It is a inbuit object which contains static method and properties in js which does mathematical calculations, introduced in ES6.

For eg: Math.random(): It generates a random number.

**Math.max(2,34,5);**

**Math.round(23.45)**

**Math.abs(2.56)**

1. **String:** In JS there are two types of string, one is primitive string and other is object string.

Let name = “Babbar” – Primitive String

Let name = new String(“Babbar’) – Object String

For eg: let lastname = "Babbar"

console.log(lastname)

console.log(typeof(lastname))

let firstname = new String ("love")

console.log(firstname)

console.log(typeof(firstname))

\*\* Jaise hi hum string ke saath .dot lagate hein js ise string object ki tarah treat karne lag jaata he.

For eg: lastname.length – 6

lastname[0] – “B”

lastname.includes("Ba") – true

lastname.startsWith("Babb")

true

lastname.toUpperCase();

'BABBAR'

lastname.toLowerCase();

'babbar'

lastname.trim(); - trim() this method removes all the spaces from a given string.

'Babbar'

lastname.replace("Babb","car") : This method replaces a given piece of characters from a string to desired string.

' carar '

**#Template Literal:**

1. This is done with the help of backticks ` ` , the message written in these backticks will be displayed as it is written in the order.

For eg: let message = "This is my first message"

console.log(` This ${lastname}

is

a

template

literal`)

Here ${} this is a placeholder where we can call any variable which we want.

**#Date and Time:**

There are 5 methods to set date and time in Js, They are as follows:

For eg: let date = new Date();

console.log(date)

let date2 = new Date("June 20 1998 07:15")

console.log(date2)

let date3 = new Date(1998, 11, 20, 7)

console.log(date3)

1. Individually we can change respective year, month, date, time as per our needs using the setmethod.

For eg: date3.setFullYear(1947)

console.log(date3)

date3.setMonth(8)

console.log(date3)

1. Individually we can get respective year, month, date, time as per our needs using the getmethod.

For eg: let date = new Date();

console.log(date)

let date2 = new Date("June 20 1998 07:15")

console.log(date2)

let date3 = new Date(1998, 11, 20, 7)

console.log(date3)

date3.setFullYear(1947)

console.log(date3)

date3.setMonth(8)

console.log(date3)

date3.getMonth

console.log(date3)

# **Arrays:**

1. Array ek prakar ka object/ reference type he jisme hum kisi bi prakar ka data store kar sakte hein.
2. Hum array mein index number use karke kisi bi element ko fetch kar sakte hein.

@**Array Insertion:**

Array mein insertion ke 3 methods hote hein:

1. **End**: Agar humein kisi array mein koi element last mein add karna hoga to hum push() method ka use karte hein. For eg: a = [1,2,3,4] a.push(5)

A= [1,2,3,4,5]

1. **Begin**: Agar humein kisi array mein koi element starting mein add karna hoga to hum begin() method ka use karte hein. For eg: b = [1,2,3,4] a.unshift(5)

B = [5,1,2,3,4]

1. **Middle:** Agar humein array mein koi element beech mein add karna he toh hum splice() method ka use karte hein. Isme humein starting mein index batana padta he jispar aako element add karna he + kitne element delete karne hein + Last mein konse elements add karne hein.

For eg: c = [1,2,3,4] c.splice(2-index no. ,0 -delete count ,”a”,”b”,”c”- values to be added)

**@Searching**

**#To check whether a number is present in the array:**

1. We can use include() to tell if a number is present in the given array or not.

For eg: numbers = [1,23,3,4]

Console.log(numbers.include(4)) – True

1. We can use indexOf to check if a number is present in the given array or not.

For eg: if(numbers.indexOf(3) != -1)

    console.log("present")

But this method only works with primitive data types.

let courses = [

    {no:1, naam:"Ankit"},

    {no:2, naam:"Love"}

]

console.log(courses)

console.log(courses.indexOf({no:1,naam:"Ankit"}))

\*\*This does not works with object or reference data types, because in primitive data types values are compared but in case of object data type there is comparison of address and both of these objects are at different addresses. Ye alag object he pr ye alag object he.

**#Callback Functions:**

1. Mereko ek method execute karna he but usko execute karne se phle aapko kuch values chaiye hongi jo ek function/method se milengi, jis function se ye data milega use hum callback function kehte hein.
2. A callback function is a function which is passed in other function as a parameter.
3. A **callback function** is a function passed into another function as an argument, which is then invoked inside the outer function to complete some kind of routine or action.
4. Object mein agar humlog ko kuch find karna he toh hum find() function ka use karte hein. Ye find function ka syntax he - (arrayname.find(\_callbackfunction,predicatefunction\_\_) or ye humko return karta he wo object jo humlog find kar rhe hote hein.

For eg: let courses = [

    {no:1, naam:"Ankit"},

    {no:2, naam:"Love"}

]

   let course= courses.find(function(course){

        return course.naam === "Ankit";

    })

console.log(course)

**#Arrow functions:**

Newer way to right functions in ES6.

1. Phle hum log function keyword() hatayenge or uski jagah ( => ) arrow laga do.
2. Then agar single parameter he toh hum brackets hatayenge.
3. Phir humlog return hatakar curly brackets hata denge {}.
4. Phir jo keywords bache hein unko ek single line mein likh kar arrow laga denge wo humara arrow function kehlata he.
5. Agar humare pass koi bi parameter ni hota toh phir humlog () => \_\_\_; or agar humare pass multiple parameters hein toh phir multiple brackets lagakar likh sakte hein, For eg: (p1)(p2)(p3) => \_\_\_\_\_;

**For eg:** let courses = [

    {no:1, naam:"Ankit"},

    {no:2, naam:"Love"}

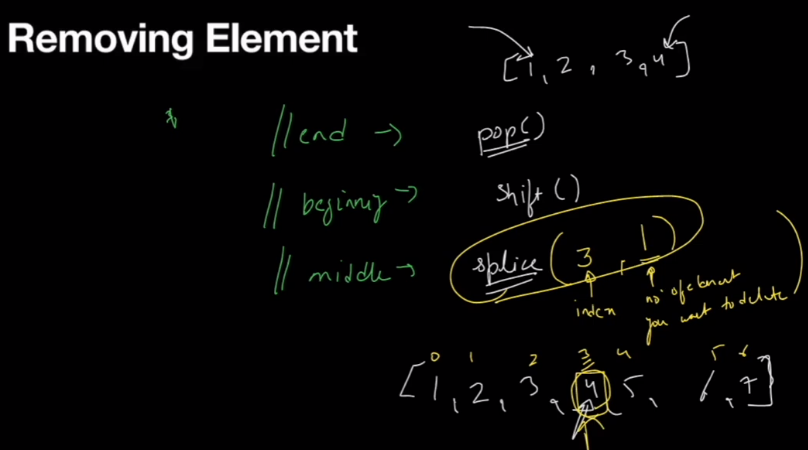
]

   let course= courses.find(course => course.naam === "Ankit");

console.log(course)

**# Removing element in an Array:**

1. End: Agar humlog koi element end se remove karana chaahte hein toh hum log pop() method ka use karte hein.
2. Beginning : Agar humlog koi element starting se remove karana chaahte hein toh hum log shift() method ka use karte hein.
3. Middle: Agar humlog koi element middle se remove karana chaahte hein toh hum log splice() method ka use karte hein.



#**Emptying an array:**

There are two methods of emptying an array[ ] . They are

1. Assign the array to an empty element.
2. Use .length() method.
3. .splice() method
4. There is a problem when we assign the array to an empty element. For eg:
5. let numbers = [1,2,3,4,5]
6. let numbers2 = numbers
7. numbers = []
8. console.log(numbers)
9. console.log(numbers2)

In this case since we are having an object so the array will point to its address but numbers2 will not be an empty array because its having a different memory address. Hence this method is not suitable for emptying an array.

1. .length() method: This is the method we use for emptying an array.

For eg: let numbers = [1,2,3,4,5]

let numbers2 = numbers

numbers.length = 0

console.log(numbers)

console.log(numbers2)

1. .splice () method: With the help of this method also we can empty an array.
2. First is the index no.
3. Second is the no. of elements to be deleted.

For eg: .splice(0, no.of elements to be deleted)

let numbers = [1,2,3,4,5]

let numbers2 = numbers

numbers.splice(0,numbers.length)

console.log(numbers)

console.log(numbers2

**# Combining and Slicing Arrays:**

1. To combine two arrays we use the concate() method.

For eg: let numbers = [1,24,4,4,5]

let numbers2 = [1,3,4.4,4]

let combined = numbers.concat(numbers2)

console.log(combined)

1. To slice an array we use .slice() method. We have to give the (start index) and then (end index) to slice the given array. The end index is one excluded.
2. let numbers = [1,24,4,4,5]
3. let numbers2= numbers.slice(2,5)
4. console.log(numbers2)
5. let numbers3 = numbers.slice(2)
6. console.log(numbers)
7. let numbers4 = numbers.slice()  // Full slicing
8. console.log(numbers4)

**# Spread Operator:**  We can combine two arrays with the help of spread operator too. For eg:

let numbers = [1,34,4,5]

let numbers2 = [3,5.556,5]

let numbers3 = [...numbers,...numbers2]

console.log(numbers3)

We can also add characters between the spread operator;

For eg: let numbers = [1,34,4,5]

let numbers2 = [3,5.556,5]

let numbers3 = [...numbers,"a",...numbers2,"b"]

console.log(numbers3)

We can also create copy of an element using spread(…) operator. For eg:

let numbers = [1,34,4,5]

let numbers2 = [3,5.556,5]

let numbers3 = [...numbers,"a",false,...numbers2,"b",true]

console.log(numbers3)

let combined = [...numbers3]

console.log(combined)

**# Iterating An Array:**

We have to traverse over an array using a loop. For iterables object we use (for-of) loop.

**# For-of Loop:**

For eg: let arr = [10,20,30,40,50]

for(let value of arr){

    console.log(value)

}

**# For-each loop:**

1. In for-each loop we have to give a callback function for its execution.

For eg: let arr = [10,20,30,40,50]

arr.forEach(function(numbers){

    console.log(numbers)

})

Syntax : 1. Sabse pehle array ka naam.

1. Then for each loop likhna he.
2. Then callback function likhna he.

For eg: let arr = [10,20,30,40,50]

arr.forEach(numbers=> console.log(numbers)

)

**# Joining An array:**

1. To join an array we use join () method. In brackets we have to give the symbol with which we want to join.
2. let numbers1 = [10,20,30,40]
3. let numbers2  = numbers1.join("+")
4. console.log(numbers2)

**#Splitting An Array:**

1. To split an array we use .split() method. In Brackets we have to give the keyword with which we want to split.
2. let para = " This is a paragraph"
3. let para2 = para.split(" ")
4. console.log(para2)

**# Sort an Array: It gives us string as an output**

1. To sort an array we use .sort() method.
2. let numbers = [30,40,20,10]
3. let num2= numbers.sort()
4. console.log(num2)

**# Reverse an Array:**

1. To reverse an array we use .reverse() method.

For eg: let numbers = [30,40,20,10]

// let num2= Number.parseInt(numbers.reverse())

// console.log(num2)

// console.log(typeof(num2)

**# Filter an array:**

1. To filter an array we use filter() function. We can do this with the help of a callback function. This function used to filter values based on some condition.

**For eg:** let numbers = [1,2,-1,-4]

let num2 = numbers.filter(function(value){

    return value<0;

})

console.log(num2)

**# Mapping Arrays:**

1. **We can map each element of an array to a different element with the help of map() function. Map function is used to map the values based on index value or some condition applied.**

For eg: let numbers = [1,23,4,5,6]

 let items=numbers.map(value =>"student-no" + value)

console.log(items)

**# Mapping with Objects:**

let numbers = [1,2,-1,-2]

let filtered = numbers.filter(function(value){

    return value<=0

})

console.log(filtered)

let items = filtered.map(function(num){

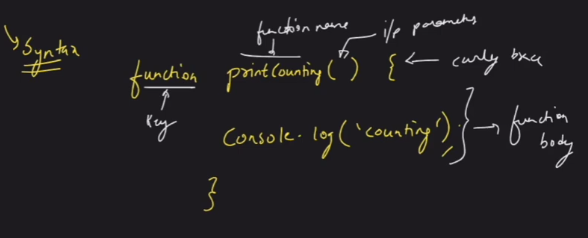
    return {value : num}

})

console.log(items)

**# Functions:**

1. A block of code that fulfills a specific task.



**#Hoisting:**

1. Hoisting is a process of moving function declaration to the top of file and this is done automatically by JS engine.

Jab bhi humlog javascript code run karte hein toh sare ke sare functions side mein aa jaate hein , toh sare ke sare code ko run karnse se phle functions file ke top par chle jaate hein or sara code run karne se pehle functions upar chle jaate hein, toh top se neeche jaate waqt sare functions ko execute karta huye aata he.

run()

For eg: function run(){

    console.log("running")

}

run()

**# Function Assignment:** This is the other way of declaring a function . In this we assign the value of a function to a variable.

For eg: let stand= function walk(){

    console.log(walking)

}

\*\* Hoisting srf function declaration ke time kaam karta he function assignment ke time nhi agar humein function assignment ke time function call karna he toh humein function ko niche hi call karna padega.

stand()

let stand= function walk(){

    console.log("walking")

}

**# Anonymous function assignment:**

This is also a way of function assignment but we did not give the function name in this case.

For eg: let stand2 = function(){

    console.log("bouncing")

}

stand2()

**# Dynamic Function call using Arguments:**

1. We have a special object in which we can pass n number of arguments that is arguments.
2. So we can pass any no. of arguments while function call and we will get the result and the function will become dynamic.
3. Arguments are object.

For eg; function sum(a,b){

    let total =0

    for(let value of arguments)

    total = total + value

    return total

}

let stand= sum(1,2,3,4,5)

console.log(stand);

**#Rest Operator:** The **rest parameter** syntax allows a function to accept an indefinite number of arguments as an array, providing a way to represent [variadic functions](https://en.wikipedia.org/wiki/Variadic_function) in JavaScript.

1. Jab humlog rest (…) operator ka use karte hein to uske baad humlog kuch bhi nahi likha sakte hein, its not allowed.
2. Rest parameter is the last formal parameter.
3. Rest parameter ke andar hum log sare ke sare varying parameters ko iske andar store kara lete hein in an array form.

**For eg:**

function sum(num1,num2,...args){

    console.log(args)

}

console.log(sum(1,3,34.4,5,6))

**# Default Parameters:**

1. There are scenearios jab user function mein values pass ni karta he toh humein default values ka use karna padta he.

For eg: function interest(p,r=6,y=10){

    return p\*r\*y/100

}

console.log(interest(1000))

Yahan par agar hum r or y mein koi values pass ni karte hein toh ye default values 6 or 10 consider kar leta he agar hum values pass karte hein toh phir wo use consider kar lega.

**\*\* Imp. Rule:**

1. Default parmeters mein ek rule hota he agr humne kisi value ko default parameter set kiya he toh uske right side se sari values ko humein default parameter banana padega or sabko values deni padegi nhi toh phir error dega.

For eg: function interest(p,r=6,y){

    return p\*r\*y/100

}

console.log(interest(1000))

Output : Nan

**# Getter And Setter:**

1. Getter 🡪 It is used to access the properties.
2. Setter 🡪 It is used to change and mutate the properties.

Reason to use getter and setter is with getter we can return the value of a function but we can’t change it, its only read only function. So with the help of setter we set or change the values.

**Rules:** a. We define get and set inside the functions.

1. Then we define the getter function to get the values.
2. But since this is a read only- function so we have if update the values so we have to use the set function.
3. let person = {
4. fname : "Ankit",
5. lname : "Jha",
6. get name(){
7. return `${person.fname} ${person.lname}`
8. },
9. set name(value){
10. let parts = value.split(" ")
11. this.fname = parts[0]
12. this.lname = parts[1]
14. }
15. }
16. console.log(person.name)
17. person.name = "Rahul Kumar"
18. console.log(person.name)

**# Error Handling:**

**$ Try & Catch Block:**

1. Try block ke andar hum apna code likhenge, jis bi code ke andar humko lagata he ki exception aa sakta he ya hum define karte hein ki isme exception aayega us code ko hum hamesa try block ke andar likhte hein or agar wahan par exception aaya toh phir ye jaayega catch block ke andar use catch kar liya jayega or phir us error ko humlog throw kar denge jahan par hum custom error message show kar denge ki ye error aaya tha or or ye error kis wajah se aaya tha.

For eg: let person = {

    fname : "Ankit",

    lname : "Jha",

    get name(){

        return `${person.fname} ${person.lname}`

    },

    set name(value){

        if(typeof value !== String){

            throw new Error("You have not sent a string")

        }

        let parts = value.split(" ")

        this.fname = parts[0]

        this.lname = parts[1]

    }

}

console.log(person.name)

try{

    person.name = 1

}

catch (e){

 alert(e)

}

console.log(person.name)

**#Scope:**

Ek variable ka jo lifetime hota he ki yahan se yahan tak wo jinda rahega use hum scope kehte hein.

1. Let keyword jis bi scope ya nearest codeblocks ke beech mein aap define karte ho wahin tak uska lifespan hota he.
2. Var keyword se aap jo bi variable define karte ho wo jis bi function ke beech mein define he y ajis bi file ke andar aap define karte utna uska scope rehta he.

**# Reducing an array:**

**## DOM –**

1. **Window:**
2. Ye ye global object hota he, isko koi bi access karta he, iska scope global hota he.
3. This object is created by browser.
4. It represents a browser window.
5. Is window object ke andar teen object hote hein.
6. This can be controlled by using window() object.

DOM BOM Js Core functions

1. Ye humko browser window ko control karne ka option deta h.
2. **DOM -**  Jo bi humlog HTML ka code likhte hein wo sare ka sare code ko JS ke object mein convert kar deta he use hum Document kehte hein. Agar hum log ko kisi webpage ke document ka object chaiye toh use hum log (document) likh kar print kara sakte hein.
3. **BOM:**  Page ke content ko chodkar baki kisi bhi chahein wo kuch bi location ho ya history ho, in sare contents ko hum BOM se control karte hein. Agar humari js browser se communicate karna chaahti hein other than content toh phir hum wo BOM se control karte hein.

**Making Changes in HTML using JS:**

**DOM:**

1. It is a tree like structure.
2. It converts characters 🡺 tags 🡺 tokens 🡺 nodes 🡺 DOM. Tokens are converted by tokenizer.

**# How to fetch Html element or perform operations(add, update) on HTML elements.**

1. **GetElementbyID(“Idname”) –** Agar koi bhi element fetch karna he jiski ID humein pta he toh phir hum usko getelementbyID se fetch kar sakte hein.
2. Agar humlog kisi ese element ki id daalte hein jo ki present ni he code mein phir wo ( null ) return karta he.

**For eg: document.getElementById("root")**

**Imp Points To Remember:**

1. It is called on Document Object.
2. It always returns a unique object.
3. It returns HTML collections.

# **GetElementsbyClassName**()

1. It returns multiple objects whose class name is same.
2. It returns collections as a object.

For eg: document.getElementsByClassName("section-content")

**#GetelementbyTagName():**

1. It returns multiple objects whose tage name is same.
2. It returns collections as a object.

For eg: document.getElementsByTagName("p")

**#Rules to remember while using GetElementsbyTagName() and GetElementsbyClassname():**

1. Both method use Document Object.
2. Bot return multiple items.
3. The list returned is not an array() it is a collection of items.

\*\* If we select any html element or tag then we can equate it with the help of ($0).

**# Methods to Access Elements:**

1. GetElementByID()
2. GetElementByClassName()
3. GetElementByTagName()

# **QuerySelector**: We can use queryselector method to select elements just like in css. We use this method and pass this it to string just like a css selector. It always return a single object. IN case of multiple objects it will return the first object.

1. QuerySelector(“#header”) – We are having a id name as a (header) so we can use to fetch the query. Header naam ki koi id he usko lekar aayo.

For eg: document.querySelector("#root")

1. QuerySelector(“.header”) – header naam ki koi class he us class ke ek element ko fetch karke laana he.

For eg: document.querySelector(".jss222")

1. QuerySelector(“header”) – header koi tag he us tag ke jitne bi objects hein uske pehle object ko fetch karke laana he.

For eg: document.querySelector("p")

**#QuerySelectorALL():**  When we have to fetch multiple elements then we have to use QuerySelectorAll() method.

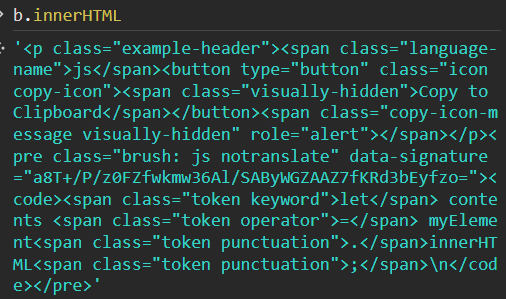
1. QuerySelectorAll(“.hidden”): . represents class name we are fetching all the class names that have hidden as a class name on the webpage.

**# Update the Existing Content:**

There are four methods with which we can update the content of a webpage.

1. .innerHTML () – get/set HTML property of a document
2. .outerHTML () - The **outerHTML** attribute of the [Element](https://developer.mozilla.org/en-US/docs/Web/API/Element) DOM interface gets the serialized HTML fragment describing the element including its descendants.
3. .textContext () - get/set text property of a document
4. .innerText() - get/set text property of a document
5. .**innerHTML** – This has two works to do. Jitna bi us div ya class ke andar HTML code hota he wo sara ka sara display ho jaata he.
6. Get an HTML element/ all of its descendant of HTML content.
7. Set an HTML element content.

For eg: 

 This innerhtml tags gives us all the html tags that are present inside that tag which we are calling with the help of getelement or queryselector method.

**# Difference Between TextContent and InnerHTML :**

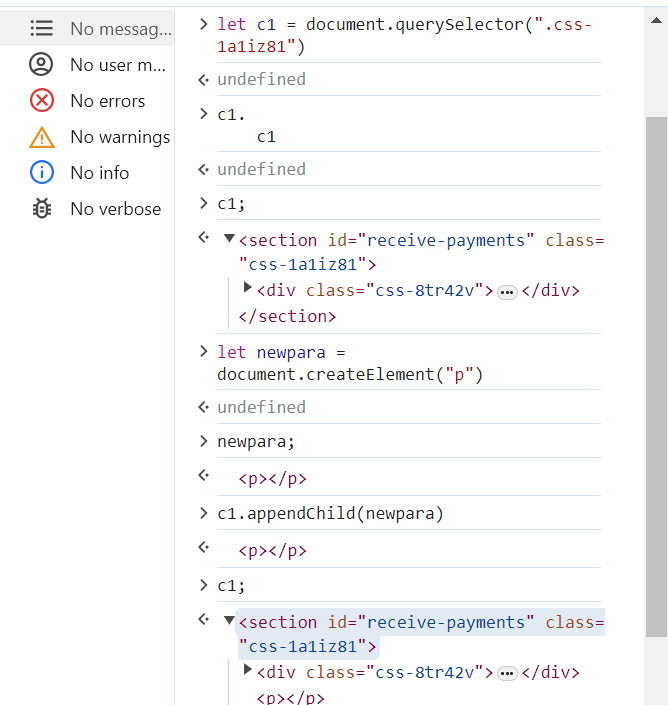
1. In TextContent() method we gets the actual text while in case of innerHTML we gets actual code that is rendered.
2. Inner html jab hum log use karenge toh tags to render kiya jaayega.
3. In case of TextContent() methods in tags ko read kiya jaata he.

**# Difference between TextContent and InnerText():**

1. IN case of Innertext jab wo text display: hidden kiya hota he toh wo humein innertext mein show ni hota he jabki Textzcontent ke andar sara ka sara text humein show hota he.
2. Jab display hidden hota he to us case mein innertext ke andar text nahi aata he or text content ke andar wo text aa jaata he.

**# Adding New Element/Content through JS:**

1. We use CreateElement() method to create a new element in JS. Now as we have just created the intent to add this element we use .appendChild() method.
2. Humlog ko element add karne se pehle use create karna padega .
3. Jab humlog append karte hein child ko toh wo humesa main content ke neeche hi append hota he.

For eg: 

**# Creating Text Node:** Abhi jo humne pehle ek paragraph add kiya tha usme ab text add karna he to hum ab wo hum textnode tag se add karenge.

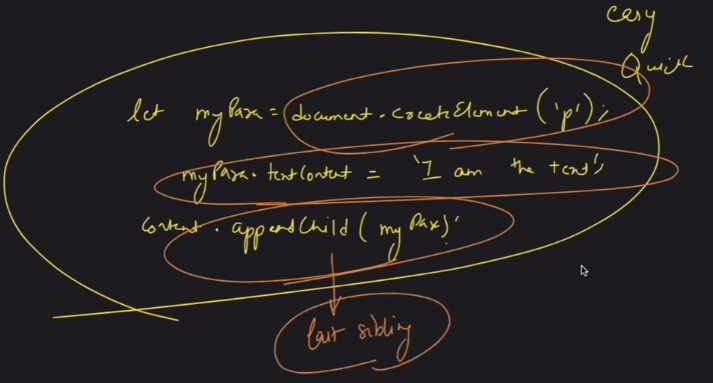
1. First we have to create a new element.
2. Then we have to create a new text node.
3. Then append the new textnode variable to the created new element.

**A black board with writing on it

Description automatically generated**

**# Easier way to Add a new Textnode element:**

1. First create a new Element.
2. Then use TextContent Tag to change the text.
3. Then append this Textcontent tag to the existing createELement Tag.

For eg: 

**# Method to position newly created element:**

**insertAdjacentHTML**():

1.It has to be called with two arguments.

a. location/position at which you want to insert the tag.(kahan insert karna he) 🡺 where

There are 4 positions at which we can insert the tag.

1. beforebegin
2. afterbegin
3. beforeend
4. afterend
5. HTMl/content to be inserted( kisko insert karna he). 🡺 what

For eg: let doc = document.querySelector(".classname")

let para = document.createElement("div")

para.textContent("My name is Ankit")

doc.appendChild("para")

doc.insertAdjacentElement("afterbegin",newtext)

**# Remove Child():**

1. This is just the opposite of appendchild() method.

Imp. Points to remember:

1. We must know the parent element ki wo kon he.
2. We must know which child element to remove.

For eg: let content = document.querySelector(".text")

undefined

content

<h3 class=​"text">​Data Tagger: EVC Ventures ​</h3>​

let add = document.createElement("div")

undefined

add;

<div>​</div>​

add.textContent("Hello")

let p = document.createElement("p")

undefined

p;

<p>​</p>​

p.innerText = "Hello"

'Hello'

add.appendChild(p)

<p>​Hello​</p>​

content.appendChild(add)

<div>​<p>​Hello​</p>​</div>​

add.removeChild(p)

<p>​Hello​</p>​

add;

<div>​</div>​

**# To remove a Element without knowing its parent:**

const node = document.getElementById("child");

if (node.parentNode) {

node.parentNode.removeChild(node);

}

**# Making Changes in CSS using JS:**

We can change the css content with the help of js without changing the CSS code. But with this strategy you can only one element at a time.

**Different properties to set in CSS using JS:**

1. .style()
2. .cssText()
3. .setAttribrute()
4. .className()
5. .classList()
6. **.style():** With this method we can style the text color or background color of the page.

For eg: A screenshot of a computer program

Description automatically generated

1. .cssText(): Ye property use karke hum multiple properties ko change kar sakte hein at a given time.

For eg: let content = $0

undefined

content;

content.style.cssText = "color:green; background-color:yellow; font-size:4rem";

1. .setAttribrute(): This method we can use to set a particular styling or assign a id or element to the tag and we can also assign multiple properties at a time.

For eg: let content = $0

undefined

content;

content.setAttribute("style", "color: green; background-color:blue");​

We can also add a Id to the given attribrute.

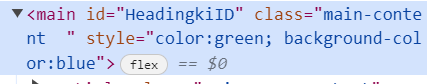
For eg:

let content = $0

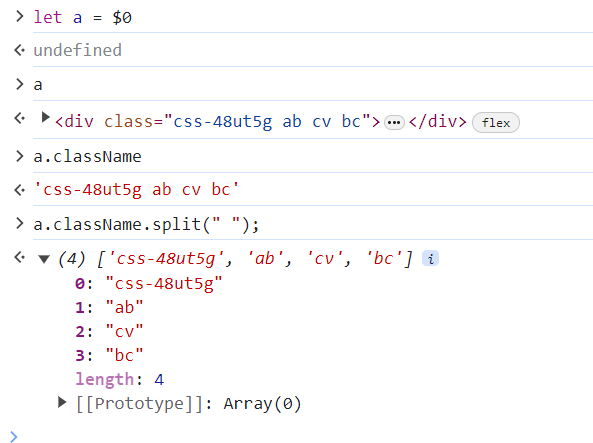
undefined

content;

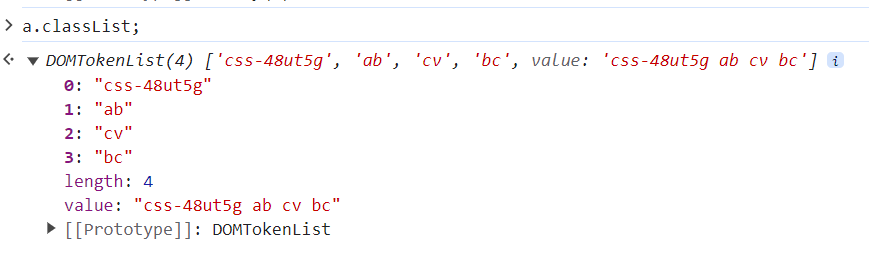
content.setAttribute("Id","HeadingkiID");​

Output: 

1. .className(): This method gives us all the classes that are present inside an element separated by space.

For eg: 

1. .classList(): This method is more used than className() because with this we get output as a array of list of items. Then we can iterate it with the help of for-of loop.

For eg: 

Then when it is converted into an array of lists we can use further methods.

1. Add(): This method is used to add an element.
2. Remove(): to remove an element.
3. Toggle(): agar koi element present nahi he to usko add kardega or agar wo element present he to usko delete kar dega.
4. Contains(): Agar koi element present he us array ke andar to wo true return kardega.

**#Browser Events:**

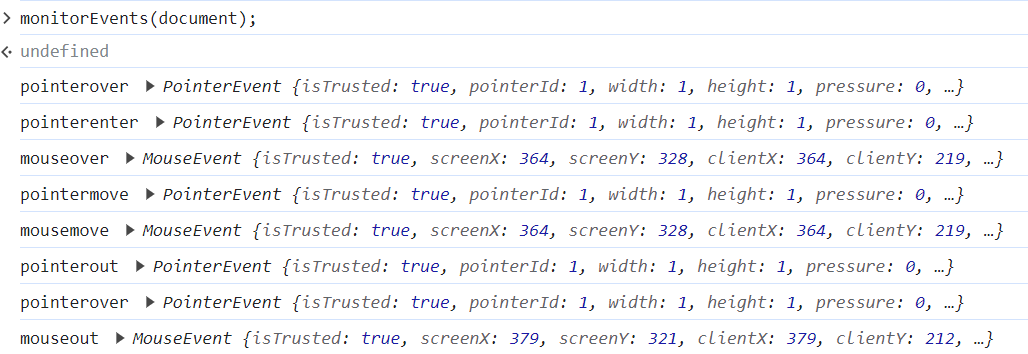
They are basically announcements made by the browser that this task is completed or in progress and you can perform this action now.

**$What you will learn in this video:**

1. What are Events
2. How to respond to an event
3. Data stored in an Event
4. Stop an Event
5. Lifecycle of an Event

#Events ki duniya jo hoti he wo invisible hoti he agar hum chaahein toh usko dekh sakte hein, uske liye humare pass ek method available he **monitorEvents()** jisse koi bi event trigger hota he to hum use dekh paayenge.

For eg: monitorEvents(document); - As soon as when we click anything on the window then we will get the list of all events that are happening on the webpage.



1. We also have a method with which we can turn off the events **unmonitorEvents()**.

**#Eventlistner:**

**1.Event Target:**  It is an interface implemented by Object that can receive events and may have listeners for them. This is the parent-level entity.

**Listener** ka work hota he wo events ko receive karta he and uske baad respond kaise karna he ya mein us event ke saath kya karna chaahta hun wo listner decide karta he. Event hone ke baad jo action define karta he wo listner karta he.

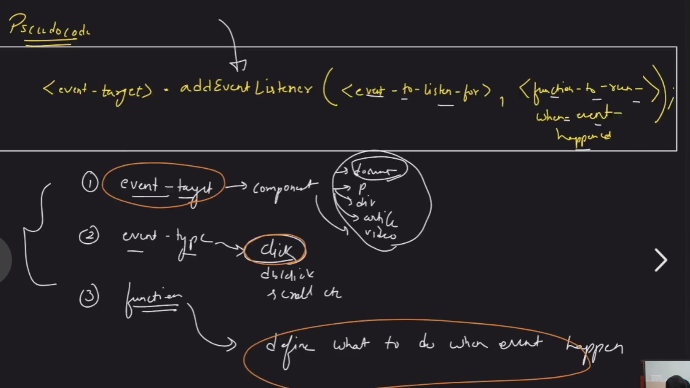
EventTarget has three methods:

1. addEventListner()
2. removeEventlistner()
3. .dispatchEvents()
4. **Node:** It is inherited from Eventtarget. All the properties which are possessed by Eventtarget will be possessed by Node.
5. **Element:**  It is inherited by node. All the properties which are possessed by Eventtarget will be possessed by Node.

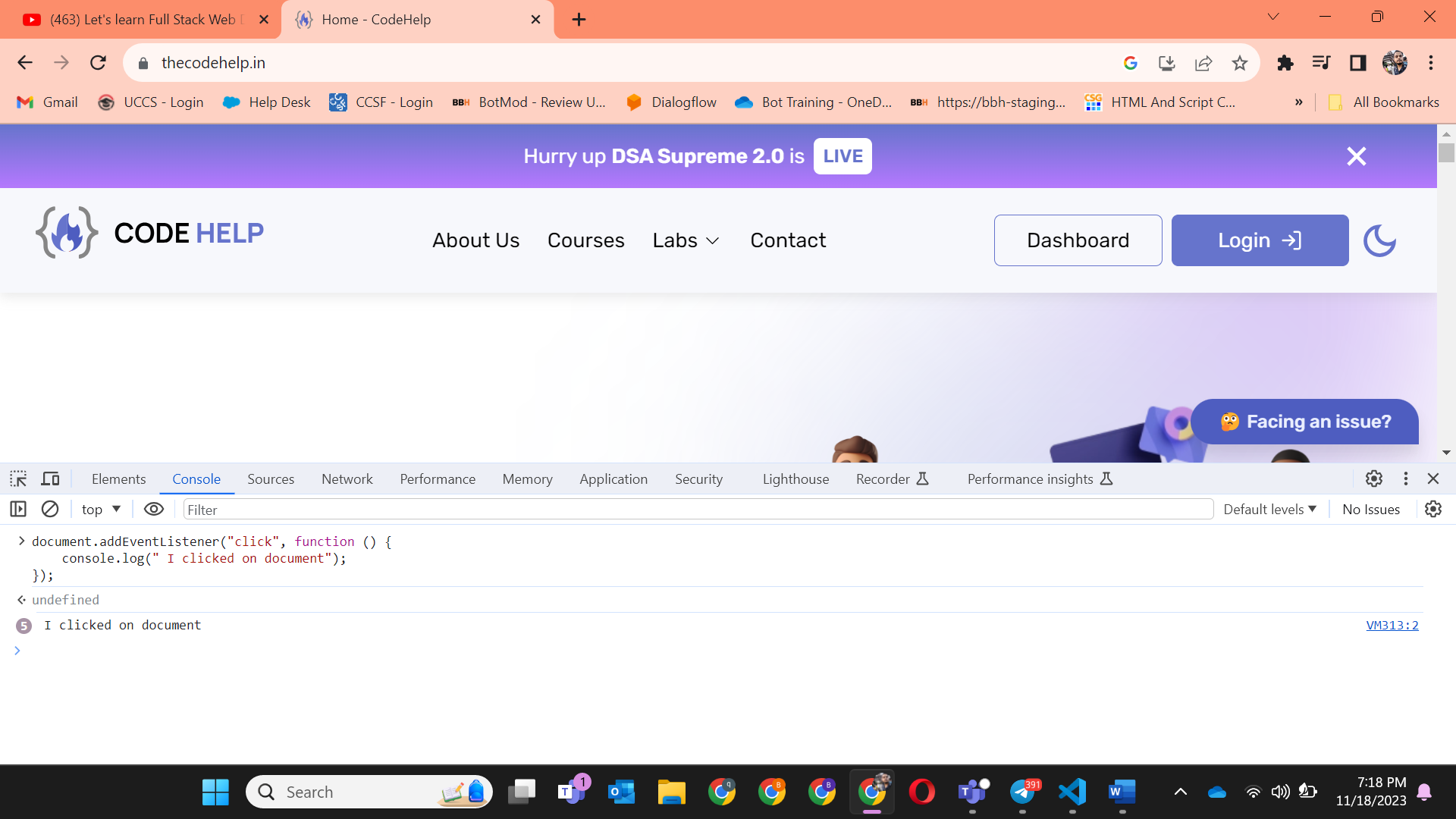


1. **addEventListner**:As the name suggests we can add listner to an event and tell the event what action to perform on it. It basically listen to an event, respond to an event, hook into events. Since this is a top level entity so all the elements come inside this for eg: document, para, article, video.

Steps to follow while adding eventlistner-



1. Event-target: It tells us on which element the listner is to be applied.
2. Event\_type: It tells us which type of event we are performing. For eg: click,scroll
3. Function: It tells us what will happen when the event happen.

For eg: 

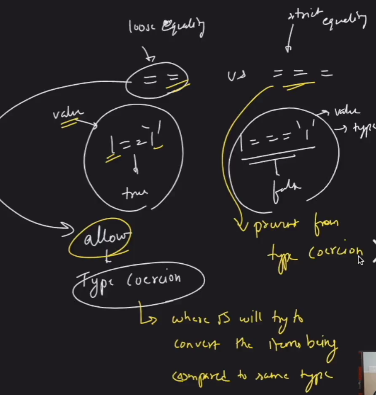
In the above picture we have applied an eventlistner on to a document with event type “click” and the action we are performing with the help of function is telling the browser I clicked on the document.

Ex- A screenshot of a computer

Description automatically generated

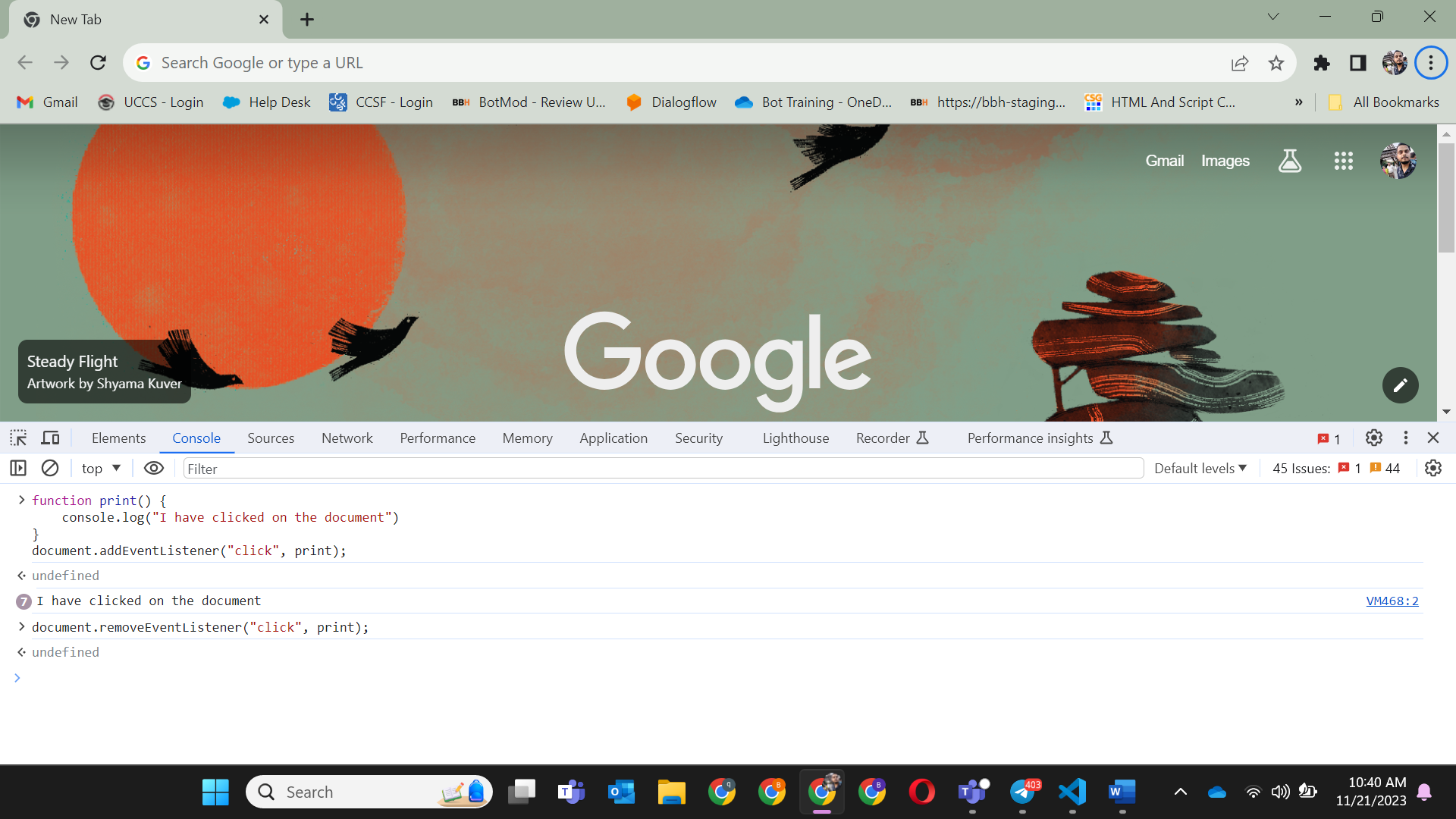
In this example we have applied an event to an H1 tag that on clicking it the background-color turns red.

1. With the help of eventlistner we can track how much time a user has spent on the website or have clicked which links or tell me how much time we have spent on that specific tab.
2. **Typecoercion**: Where Js will try to convert the item being compared to same type. Suppose you have a string “1” and int 1 so js type coercion will convert both into the same data type.

For eg: Loose Equality will allow type coercion but StrictEquality will not allow type coercion. 

1. **RemoveEventListner():**

As we addEventlistner with the help of creating one function so we can remove with the help of same function but we cant create two different functions because otherwise it will not work because two functions with same name will create a different objects so js can’t do type coercion.

For eg: 

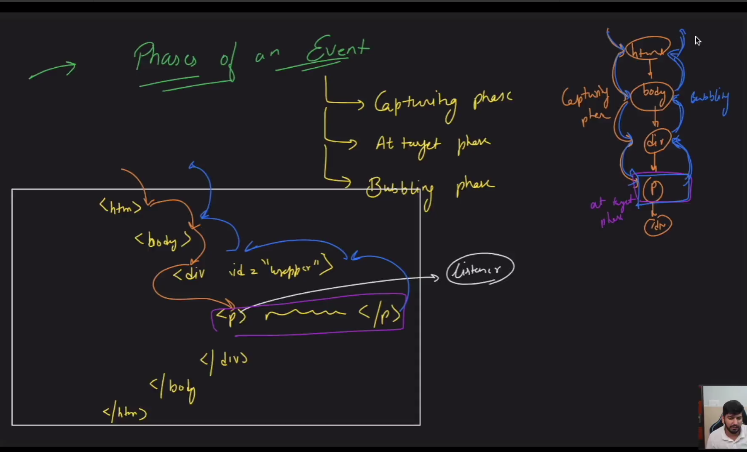
In the above example we have created a function print() and that we have called in the addeventlistner and that same function we have called in the removeeventlistner method to envoke the changes.

Remove listner will only work when the above three conditions satisfy:

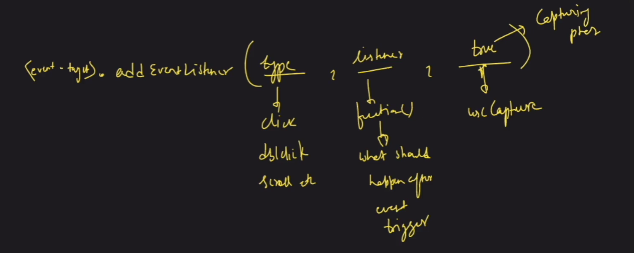
1. Same eventtarget.
2. Same type of data.
3. Same function.

**# Phases of an Event:** In this we want to get what is the flow of the events.

1. **Capturing Phase**
2. **At target Phase**
3. **Bubbling Phase**
4. **Capturing phase:** Is phase ke andar jakar aap dhundte ho ki wo para ya eventtarget kahan he jiske upar aapko listner lagana he.

****

1. **Target Phase:** As soon as we have reached to that particular eventarget where we want to apply the listner or perform the action is called target phase.
2. **Bubbling Phase:** After reaching the at the target phase we have to move backwards out from the code that is known as bubbling phase.

****

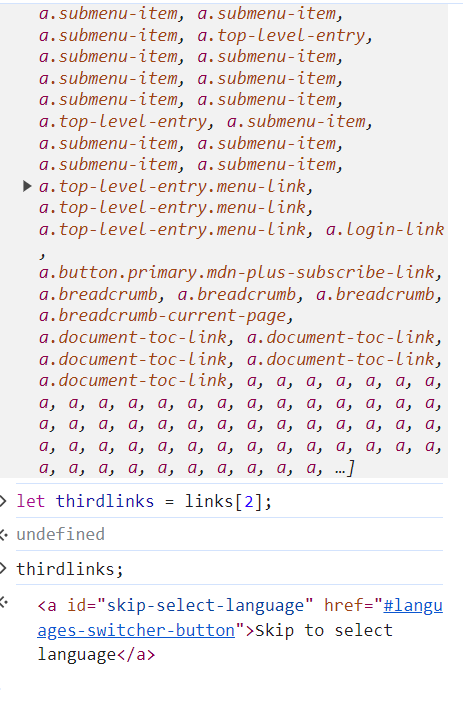
In the above diagram to apply eventlistner we have to pass three arguments.

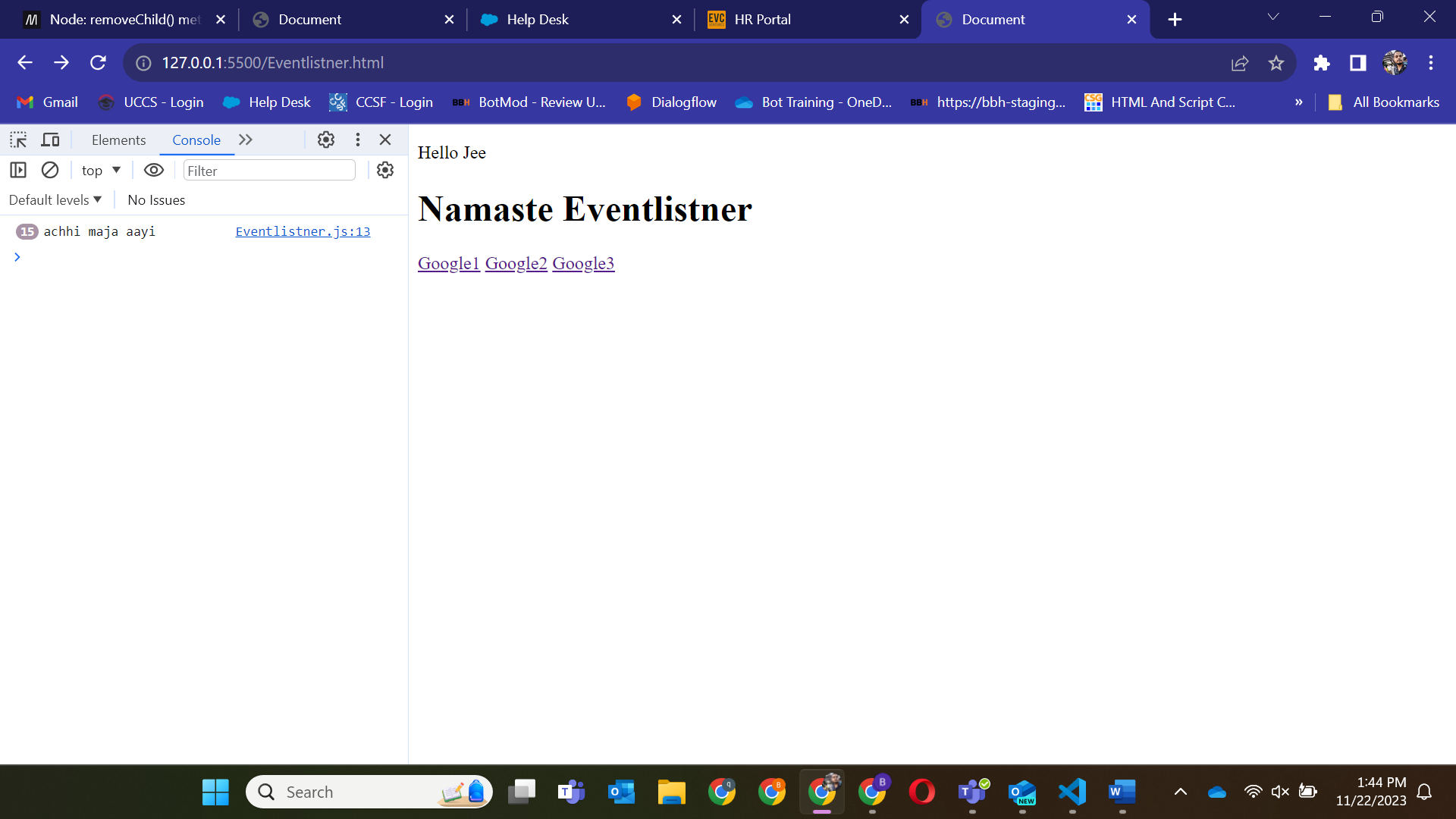
1. Type: For eg- click, double click, scroll
2. Listner: For eg- function() what should happen after when the event is triggered.
3. True: If we set this to true which means it is in the capturing phase and we have to use Capture.
4. By default addEventlistner is applied to the Bubbling phase.

#**The Default Actions**:

1. Anchor tag 🡪 link open window.
2. .preventDefault() : This method will allow us to prevent the default actions that are performed by the tags.

If you want to select a specific link to be selected from the given set of links. We can do this with the help of –



<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <p>Hello Jee</p>

    <h1 id ="wrapper">Namaste Eventlistner</h1>

    <a href = "https://www.google.co.in/">Google1</a>

    <a href = "https://www.google.co.in/">Google2</a>

    <a href = "https://www.google.co.in/">Google3</a>

    <script src="Eventlistner.js"></script>

</body>

</html>

let links = document.querySelectorAll("a");

let third = links[2];

third.addEventListener("click",function(event){

    event.preventDefault();

    console.log("achhi maja aayi")

});

In the above example we have added an event listner to the third link and then we have applied prevent default method to stop it from clicking to another page by passing event as an argument.

\*\* We have to pass event in function while using preventDefault() method because we are applying prevent default method to event only so it is mandatory to pass.

**# How to avoid Too many Events:**

1. Event.Target Property: With this property we can apply individual listners to events within a certain tag or element.

For eg: let myDiv = document.createElement("div")

function parastatus(event){

    console.log("Para" + event.target.textContent)

}

myDiv.addEventListener("click", parastatus);

for (let i =0; i<100 ; i++){

    let newElement = document.createElement("p");

    newElement.textContent = "This is para" + i;

    myDiv.appendChild(newElement);

}

document.body.appendChild(myDiv)

Output : 

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

        <article id = "wrapper">

    <p>XYZ <span>Article</span></p>

    <p>XYZ <span>Article</span></p>

    <p>XYZ <span>Article</span></p>

    <p>XYZ <span>Article</span></p>

    </article>

    <script src="Eventlistner.js"></script>

</body>

</html>

let element = document.querySelector("#wrapper")

element.addEventListener("click", function(event){

    console.log("Span par click karna he" + event.target.textContent)

})

A screenshot of a computer

Description automatically generated With the above code when we are clicking on the <span> tag then also the listner is working and when we are clicking on the <p> tag then also the eventlistner is working, SO we have to avoid this situation and this we can avoid with the help of <Node> property.

**#Node Property: (When we explicitly tell that listner only works on the given tag)**

let element = document.querySelector("#wrapper")

element.addEventListener("click", function(event){

    if(event.target.nodeName === "SPAN"){

    console.log("Span par click karna he" + event.target.textContent)

    }

})

**Output:**

**A screenshot of a computer

Description automatically generated**

In the above code we are explicitly telling that the eventListner must be applied only to <Span> tag and not on the <p> tag with the help of property (event.Target.nodeName === “SPAN”) but one thing we have to keep in mind that the Tag Name must all be in **UPPERCASE.**

**##** We have to always keep the <script> src at the end of the body tag when all the content is loaded so that to avoid any error. But how we will come to know that all the HTML content is loaded. This we come to know with the help of property DOMContentLoaded i.e in the network tab.

A screenshot of a computer

Description automatically generated

**#Dom Lecture3:**

**#Performance:**

1. Measure speed of code.
2. How to write efficient and performing code.
3. Event Loop.
4. Performance.now() – It tells us how much time the code takes to run by giving the timestamp.

For eg: const t1 = performance.now()

for (let i=1; i<=100;i++){

    let newElement = document.createElement("p");

    newElement.textContent ="This is para" +i;

    document.body.appendChild(newElement);

}

const t2 = performance.now()

console.log("This took" + (t2-t1) + "ms")

// Optimising code

const t3 = performance.now()

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

for (let i =1; i>=100; i++){

    let element = document.createElement("p")

    element.textContent="This is para" + i;

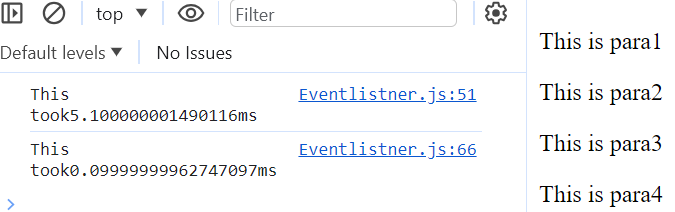
   mydiv.appendChild(element);

}

document.body.appendChild(mydiv)

const t4 = performance.now();

console.log("This took" + (t4-t3) + "ms")

 This tells us how much time the code takes to run and tells its performance.

**# Reflow and Repaint():**

**Reflow:** It tells us the mathematical computations happening in the background.

**Repaint:** After the mathematical computations are performed the image is repainted and it is displayed pixel by pixel and shows on display.

**Best Practice: Minimise the no. of reflow and repaint.**

**#Document Fragment:** It is a lightweight document object which does not have reflows and repaint. It is run when we add this to document and only 1 reflow and 1 repaint happens.

For eg: let fragment = document.createDocumentFragment();

for (let i=1; i<=100;i++){

        let newElement = document.createElement("p");

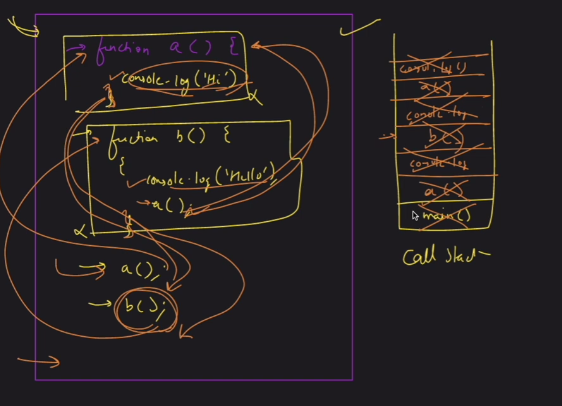
        newElement.textContent ="This is para" +i;

        fragment.appendChild(newElement);

    }

    document.body.appendChild(fragment) // 1 reflow and 1 repaint

**# Call stack:**  Ek esi list he jiske andar aap track rakhte hein ki konsa function ya code kab execute huya he. Jabbhi koi function execute hota he toh uski ek entry call stack mein banti he or jaise hi wo pura execute ho jaata he toh uski entry call stack se remove ho jaati he.

For eg: 

**Single Threading:** Running one command at a time. JS is a single threaded language.

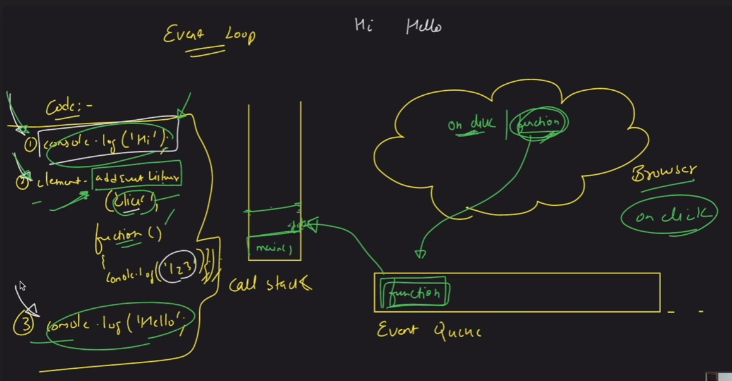
1. Js is a run- to completion nature of code that whichever line or function is executing it will complete its full execution and then it will got to other lines of code.
2. Js does not execute multiple lines/ function at the same time it will allow only one line or function to execute at a given amount of time,

**\*\* Veri Imp.**

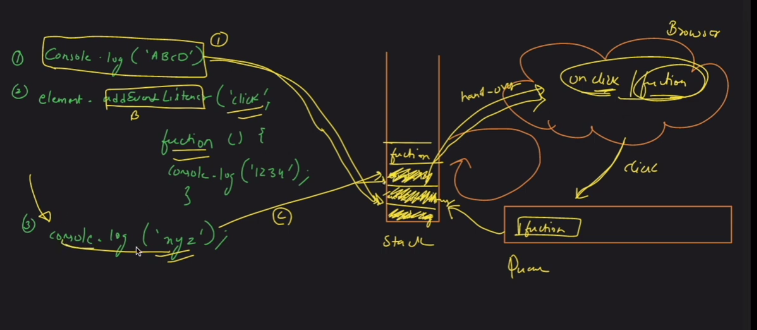
**EVENT LOOP:**

It has three things :

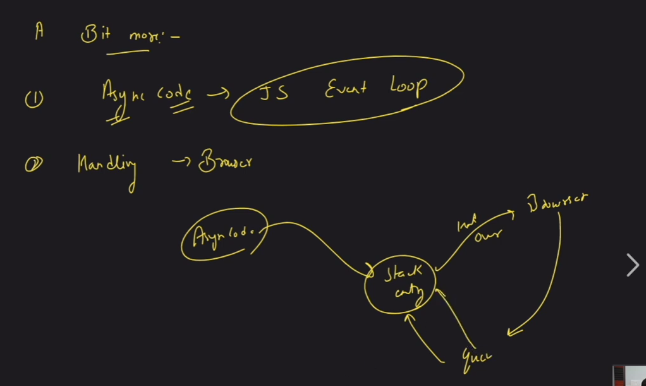
1. Call Stack.
2. Browser
3. Event Queue



In the above diagram first console.log(hi) line will execute and a entry is created in the call stack, then after its execution the entry is removed from call stack, Then the eventlistner is executed and the entry is created in call stack, eventlistner tells the browser that only when it is clicked then only the function will execute otherwise not, Then the console.log(hello) is executed and an entry is created and after completion entry is removed and now suppose the eventlistner function is clicked now its start executing and the entry of a function is stored in the Eventqueue but the imp. thing is to remember that this eventqueue will only execute the function when the call stack is empty i.e is baat ki guarantee nahi he ki on the click wo eventlistner execute ho jaayega.



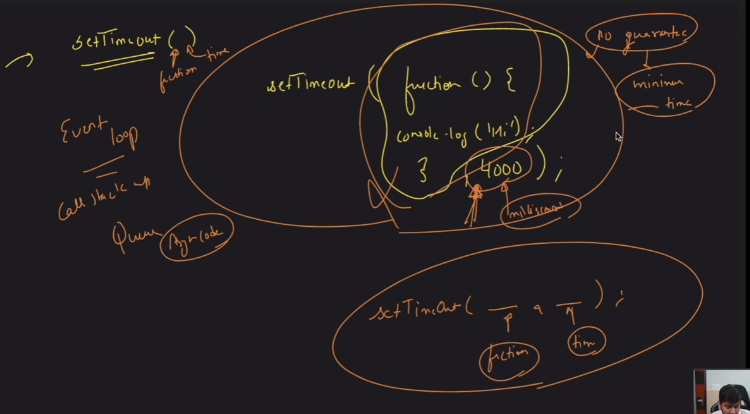
1. Sabse pehle isme console.log(“ABCD”) ki entry bani ye function execute huya or iski entry call stack se hat gayi.
2. Phir eventlistner wala function execute huya wo call stack ke pass aaya or phir usne apnea ap ko borwser ko handover kar diya or wo chala gya.
3. Phir console.log(“xyz”) call stack mein enter huya or phir wo execute huya or uski entry call stack se hat gya.
4. Phir evenlisnter wala function execute huya click karne par browser ke pass se wo entry Event Queue ke andar gayi or queue aapki entry tabhi execute karega jab humara call stack free hoga phir wo call stack check karke aaya ki kya call stack khali he agar khali huyi toh usne eventlistner ko execute kara diya or agar koi synchronous execution phele se chal rha he toh phir uske liye aapko wait karna padega uske khatam hone ka.

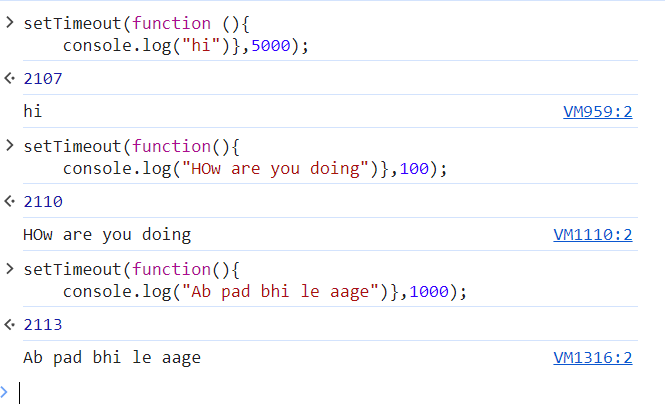


**## setTimeout():**

**It is a async() method**

1. Is method mein hum ek function likhte hein or usko batate hein ki is code ko run hone se pehle itne millisecond wait karna padega. Mandatory to provide the time in this method.
2. Or iski koi guarantee nahi he ki jo timeout time humne diya he ki is code ko itne time ke baad execute karna he wo utne time baad execute hi ho jaayega wo time minimum time he ki itna time toh lagna hi lagna he uske upar bhi time lag sakta he.
3. This method will run code at a later amount of time.
4. It has two important parameters –
5. Function – what we have to do.
6. Time – time required after which the code will run.

For eg: 

For eg: 

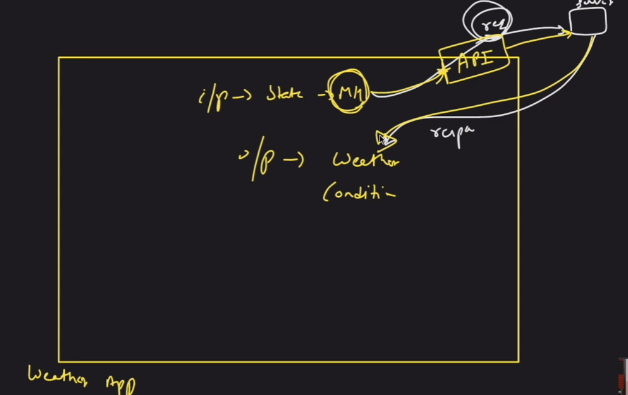
This is the settimeout function in which we have given what the function will do and then we have given the time interval for which the settimeout function will work.

**\*\*Very Very Imp.**

**Dom Lect4**

1. **API:** Application programming interface.

It is a interface between two software components which establishes the connection between two of them.

For eg: 

In the above example suppose we are making a weather app and as soon as we give the input as state name eg (Maharashtra) the API request the server which is place far away and then the server fulfills the request and sends back the response to the client.

So API is basically our naukar jo humara kaam karke deta he.

**#Features of Async code:**

1. Clean and concise.
2. Better error handling.
3. Easier debugging.
4. Better performance it helps reduce wait time.

**Very imp**

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

1. It is used for running processes in the background parallely.
2. It is a async code.
3. In promise we pass a callback function.
4. It has two input parameters ( resolve,reject).

Case1:

For eg: let meraPromise = new Promise(function(resolve,reject){

    console.log("I am inside promise");

    resolve(1998)

})

console.log("Pehla pyaar")

Output :

A screenshot of a computer

Description automatically generated

Case2: W hen we give a async function inside a promise.

For eg: let prom = new Promise(function(resolve,reject){

    setTimeout(function(){

        console.log("Pehla")

    },5000);

    resolve(1998)

})

console.log("dusra")

Output:

A close-up of a website

Description automatically generated

In the above example first the normal function is executed dusra is printed and then after the async function is executed after 5sec and first is printed.

To check the status of promise simply we have to type the variable name with which the promise is created.

For eg: A screenshot of a computer code

Description automatically generated If the promise is resolved it will give status as fulfilled and will return the value.

If status resolved 🡪 fulfilled

If status reject 🡪 rejected.

For eg: A screenshot of a computer program

Description automatically generated

Syntax for Promise: A blackboard with white text

Description automatically generated

1. If we want to run things parallely then we have the best option to run the code using promise.

For eg: let prom3 = new Promise(function(resolve,reject){

    setTimeout(function(){

        console.log("Chautha")

    },8000)

})

let prom2 = new Promise(function(resolve,reject){

    setTimeout(function(){

        console.log("Teesra")

    },3000)

})

let prom = new Promise(function(resolve,reject){

    setTimeout(function(){

        console.log("Pehla")

    },5000);

    // resolve(1998)

    // reject("error aaye hein")

})

console.log("dusra")

**# There are two methods in Promise:**

1. **Then() 🡪 value**
2. **Catch() 🡪 error**
3. Then() – Toh fulfilled wale case mein jo bhi value aayi hogi output mein agar humein use use karna he toh then() wale method ka use karte hein.

**For eg:** let prom = new Promise(function(resolve,reject){

    setTimeout(function(){

        console.log("Pehla")

    },5000);

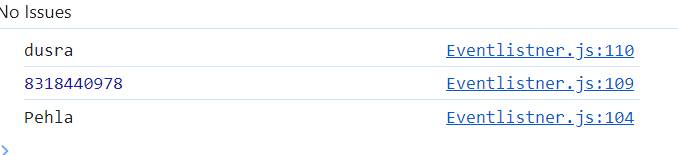
    resolve(8318440978)

    // reject("error aaye hein")

})

prom.then((value) => {console.log(value)});

console.log("dusra")

****

1. Catch() – Agar koi bi error aayi hogi rejected wale state mein to hum use catch() wala method use karte hein.

**For eg:** let prom = new Promise(function(resolve,reject){

    setTimeout(function(){

        console.log("Pehla")

    },5000);

    // resolve(8318440978)

    reject("error aaye hein")

})

prom.catch((error) => {console.log("Bhaiyya error he")});

console.log("dusra")

\*\*Imp:

1.Agar koi promise fulfill ho gya he ya khatam ho gya he uske baad tum koi action lena chaahte ho basis on the completion of the task tab hum then wale method ko use karte hein. If we have multiple promises dependant to each other then we can use then() method, as soon as one promise is completed the next promise start its execution.

For eg: let waada1 = new Promise(function(resolve,reject){

    setTimeout(function(){

        console.log("set Timeout1 started")

    },5000)

    resolve(true)

})

let output = waada1.then((value)=> {

    let waada2 = new Promise(function(resolve,reject){

        setTimeout(function(){

            console.log("set Timeout2 started")

        },8000)

        resolve("Waada2 resolved")

    })

    return waada2;

})

output.then ((value) => console.log(value));

In the above example first we have created a new promise named Waada1 and inside that we have created a new promise and we have marked it as resolved and then we have created an another promise inside waada1 named output using then() method and we have created a setTimeout function which will be called by the browser when the time is completed and stored in the queue and then we have return the vale of second promise.

**\*\*Very IMP: If we are having 50 promises then we can not use then() method:**

1. **Async and Await():**
2. It is a special syntax used to work with promises.
3. Async function always returns a promise.
4. Await is used to stop a promise until the first promise is executed successfully. For eg: we have two promises P1 and P2 so we can use await function on P1 or can say P1 jab tak P1 complete nah ho jaaye mein P2 start hi nahi kar rha, mein P2 ko rok ke rakh rha ki P2 tu start hi nahi hoga.
5. Suppose aapke do async code chal rhe hein tab hum await() ka use kar sakte hein ki jab tak pehla code nah chal jaaye tab tak dusra code nahi chalna chaiye.
6. Hum log kisi bhi function ko async or await bana sakte hein.

**#How we can create an aysnc function:**

async function abcde(){

    return ("kya huya tera")

}

console.log(abcde())

**A computer code with red and black text

Description automatically generated**

**We can make a function await and stop the execution of one process until another is finished just by adding await() keyword and we don’t want to use then() keyword and we want a safer way to execute things then we use the keyword await().**

**For eg:** async function utility() {

let DelhiMausam = new Promise (function(resolve,reject){

    setTimeout(function(){

        console.log("Delhi mein bahut garmi he")

    },3000)

    resolve(true)

})

let HyderabadMausam = new Promise (function(resolve,reject){

    setTimeout(function(){

        console.log("Hyderabad is cool")

    },9000)

    resolve(true)

})

let dm = await DelhiMausam

let hm = await HyderabadMausam

return(dm,hm)

}

**Very Imp;**

**\*\* Fetch API:**

1. If we want to send or receive data over a network than we use the method Fetch API().
2. If we want to use fetch api then we must know the url.

For eg: let content = fetch("https://jsonplaceholder.typicode.com/posts/1")

1. Fetch API method returns promises:

A screenshot of a computer code

Description automatically generated

1. Fetch API method when we want to GET some data from an API:
2. **#GET call using Fetch API():**

So in order to get data from an api we must know the URL of the api, which we put in the fetch() and then we store the value of it in a variable in json format. We used await method to tell it jab tak humara data nhi aata he tab tak usko wait kara ke rakhenge. Data humara json format mein aata he jo ki ek object hota he.

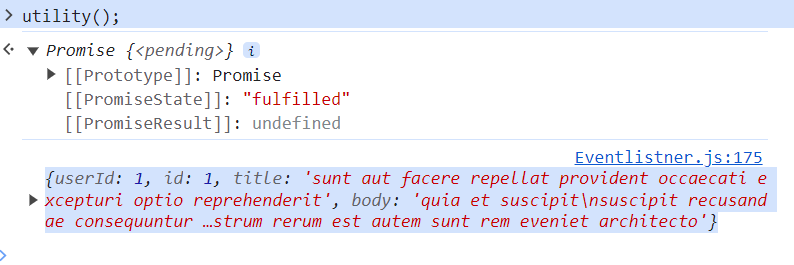
For eg: async function utility(){

let api = await fetch("https://jsonplaceholder.typicode.com/posts/1")

let output = await api.json();

console.log(output)

}

Output: 

1. Jab bhi hum fetch API ke dwara GET call karte hein (data ko retrieve karte hein ya fetch kar rhe hote hein in json() format.

For eg: async function utility(){

let api = await fetch("https://jsonplaceholder.typicode.com/posts/1")

// let output = await api.json();

// let output = await api.text();

let output = await api.ok();

console.log(output)

}

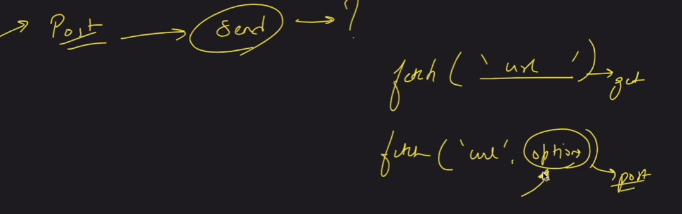
1. Some API’s also have a authentication mechanism in which we have to also pass a key to access that API due to security reasons.

**#Fetch () method:**

1. In Fetch method we have to pass two things.
2. URL
3. Options- In options we can pass the secret key or the information we want to send specific to the user for the information we want to fetch from it.

**#Post call:**

1. If we want to send some data to a network then we use the post method.
2. So in order to make a post call in fetch() function we have to make option as such parameter so that we can make that request a POST call.

For eg: 

1. To humein options wale object ko ese likhan hoga ki wo ek post call ho.

For eg: let options = {

    method: 'POST',

  body: JSON.stringify({

    title: 'foo',

    body: 'bar',

    userId: 1,

  }),

  headers: {

    'Content-type': 'application/json; charset=UTF-8',

  },

};

let content = fetch("https://jsonplaceholder.typicode.com/posts", options)

Upar code mein hum jo ye teen field hein (title, body, userid) usko hum is url (<https://jsonplaceholder.typicode.com/post>) par bhej rhe hein ki isko store karlo using Post method.

1. Now we want to send some additional data in the POST method so we can do this in the following manner and if we want to know what all things are there in the header we can do this way.
2. async function helper(){
3. let options = {
4. method: 'POST',
5. headers: {
6. 'Content-type': 'application/json; charset=UTF-8',
7. },
8. body: JSON.stringify({
9. title: 'Ankit',
10. body: 'Welldone Ankit',
11. userId: 1996,
12. weight: "90kg"
13. }),
14. };
15. let content = await fetch("https://jsonplaceholder.typicode.com/posts", options)
16. let response = await content.json();
17. return response
18. }
19. async function utility(){
20. let ans = await helper();
21. console.log(ans);
22. }
23. console.log(utility())

**#Json.stringify():**

This method converts json object to a json string.

V.Imp:

**#Closures:**

Closure ek esi prakar ki cheez he jisme hum function ko uske required data ke sath bind karte hein. So we can say closure ek top level entity he jisme function ka data pada hoga + sare variables ya paremeters jo function ko required hein us function ko execute

1. In other words, a closure gives you access to an outer function's scope from an inner function. In JavaScript, closures are created every time a function is created, at function creation time.

For eg: function init() {

    var name = "Mozilla"; // name is a local variable created by init

    function displayName() {

      // displayName() is the inner function, that forms the closure

      console.log(name); // use variable declared in the parent function

    }

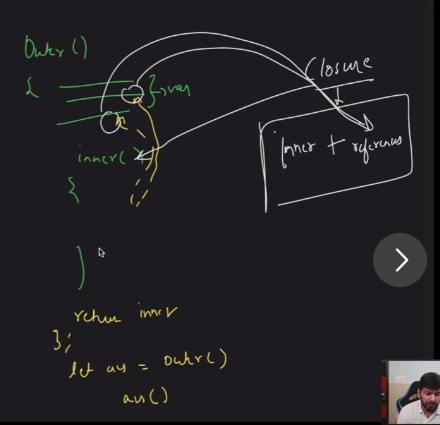
    displayName();

  }

  init();

1. Closure in short aapke pass ek outer function hoga usme kuch variables honge or uske anadr tumhara inner function honga usme outer function ke variables ko use kiya jaa rha hoga or last mein aap inner wale function ko return kara rahenge or last mein aapne outer function ko ek variable mein store kar liya he or last mein us variable ko aapne print kara liya he. Toh aapka closure kya hoga ?

Aapka inner wala function + jo outer function mein variables store huye hein unke references.

For eg: 

**\*\* Very Imp: Closure🡺 Nested Function 🡺 Copy**

1. Jab bhi aapke pass nested function hoga to humein closure ko yadd karna he or nested function mein references wala scene hota he copy ni hoti he.
2. Jitne bi aapke pass nested function honge un sabhi ke liye closure bante jaayenge , toh agar kisi function ko kisi or surrounding state, variable ya environment jo aapke parent ke andar pade honge wo bi access uske pass available hoga or jo access available hoga wo reference ke form mein available hoga.

H.W: Classes and Export module in JS

\*\* We can use this line in HTML code to use the Tailwind code.

    <script src="https://kit.fontawesome.com/35a992e52f.js" crossorigin="anonymous"></script>

Imp: Main point to remember is that we have to paste the line of code for the icon imported from font-awesome under the specific tag under which we want that icon.

Project1: Simple Counter App

Button – We have a special method named OnClick() which tells the button what to do when it is clicked. As soon as the button is clicked we can set the function to be called and it will tell us what to do with that click.

**#Second Project:**

**Unwrap Class:**  It basically means humare pass ek code pada huya he or hum us code ko bta rhe hein ki ye code ye kaam kar rha he or ye code ye with the help of unwrap class method.

**Reason To do It:**

1. To make the code industry ready.
2. Kisi or ke code ko smajhna pdta he isme or uske baad humein aage code likhna hota he so pehle toh wo code dhang se likha hona chaiye phir humein usko breakdown karke samjhna padta he.
3. So it is basically reverse engineering means humare pass pehele se hi code available he usko hum samjh rhe hote hein ki is code ne kiya he or is code ne ye kiya he.

**Overlay:** This is the transparent property that we seen in a website