**Basic Methods**

* charAt()
* charCodeAt()
* codePointAt()
* concat()
* includes()
* endsWith()
* startsWith()
* indexOf()
* lastIndexOf()
* localeCompare()
* match()
* matchAll()
* normalize()
* padEnd()
* padStart()
* repeat()
* replace()
* replaceAll()
* search()
* slice()
* split()
* substring()
* toLowerCase()
* toUpperCase()
* toLocaleLowerCase()
* toLocaleUpperCase()
* trim()
* trimStart() / trimLeft()
* trimEnd() / trimRight()
* valueOf()

**🧪 Newer/Advanced Methods**

* at() *(like charAt but supports negative indices)*

**📏 Utility Methods (inherited or special)**

* length *(not a method, but a property)*
* toString() *(returns the string itself)*

**Array Methods:** foreach(), map(),flatMap(), filter(), find(), findIndex(), reduce()

1. push() return new length of array
2. pop() return value and array
3. splice(start, deleteCount, value1, value2, …..)
4. indexOf(value, startFrom)
5. lastIndexOf(value, startFrom) backtraking in array.
6. includes(value) return true or false
7. flat(noOfArray) use for removing nested array inside array. **noOfArray** means how much nested array you want to remove.
8. arrayName.sort((a, b)=>{a-b}) in Ascending order and descending order
9. let array = [1,2,4,5,2,1,5,6,0]
10. let newArray = array.sort((a,b)=> {
11. if(a>b) return -1
12. if(b>a) return 1
13. })
14. console.log(newArray)

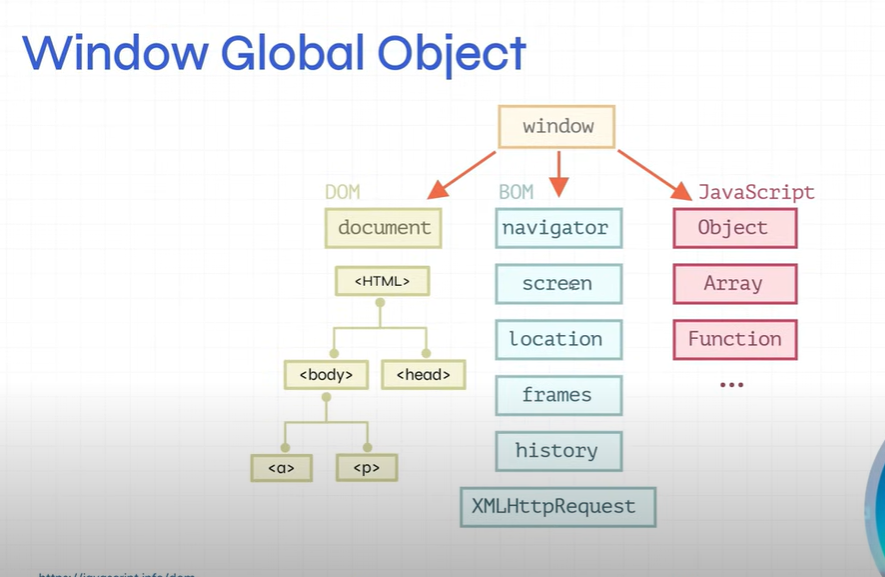
**String Methods:**

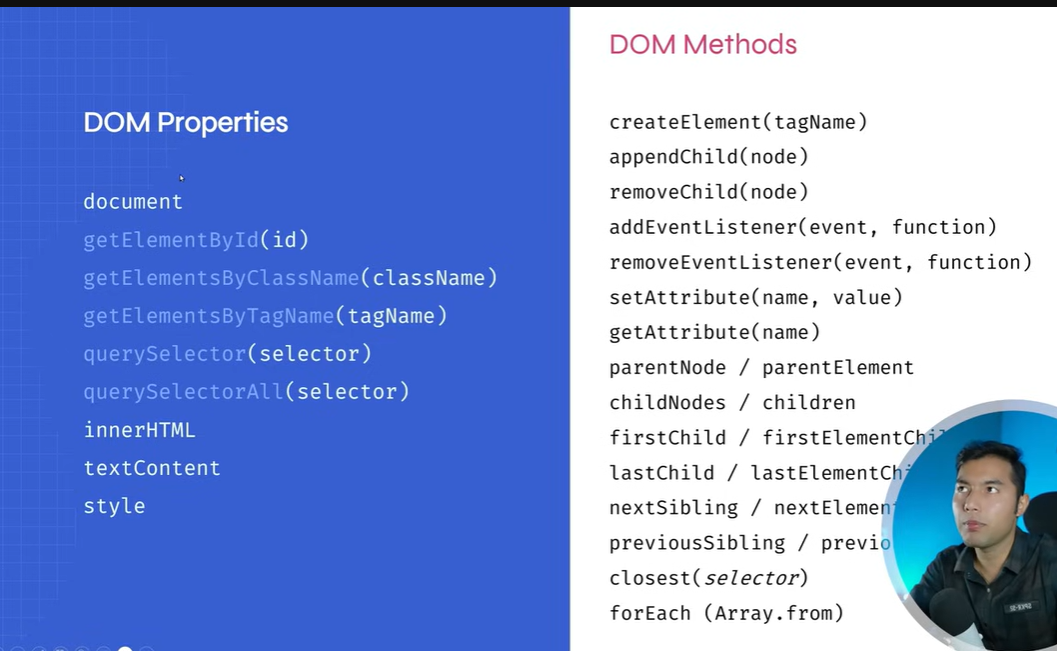
1. indexOf()
2. lastindexOf()
3. Array.from() convert string to array
4. search() return indexValue and also support regular expression
5. match() and matchAll()
6. includes() return true or false and does not support regular expression like /g, /i
7. startsWith() and endsWith()
8. slice(start, end) give you a piece of string and also can take negative values
9. substring(start, end) give you a piece of string and can’t take negative values
10. at() take both positive and negative values
11. replace() and replaceAll()
12. charAt() , charCodeAt()
13. toUpperCase(), toLowerCase()
14. trim() remove extra spaces
15. split() convert string into an array
16. reverse() reverse array
17. join() convert array into string
18. String.fromCharCode()
19. padStart(NoOfSpaceRequired, addSymbol) and padEnd(NoOfSpaceRequired, addSymbol)

**Math object:**

1. Math.round()
2. Math.floor() in posivtive value go down and in negative go up
3. Math.ciel() in posivtive value go up and in negative go down
4. Math.trunc() in both negative and positive go down
5. Math.pow()
6. Math.random()
7. toFixed()
8. Math object does not work with Bigint data type .

**Window Object Model:**





**How we can copy text?**

1. Navigator.clipboard.writeText(para.innertext)
2. Explore its further methods like write , read, readText

**Set:**

1. For removing duplication we can use Set()
2. We will use Set() when we need to remove duplicate values
3. Methods: add(), delete(), has(), clear(), size.

**Date:**

1. ISO DATE Format: ISO formats like "YYYY-MM-DDTHH:mm:ss"
2. Months start from 0 to 11 like January = 0,…… December = 11
3. Related to point 2: Js count months from 0 to 11.
4. We get the millisecond value from January 01 1970.
5. JS stores date as number of milliseconds.
6. Days start from 0 to 6 in JS.

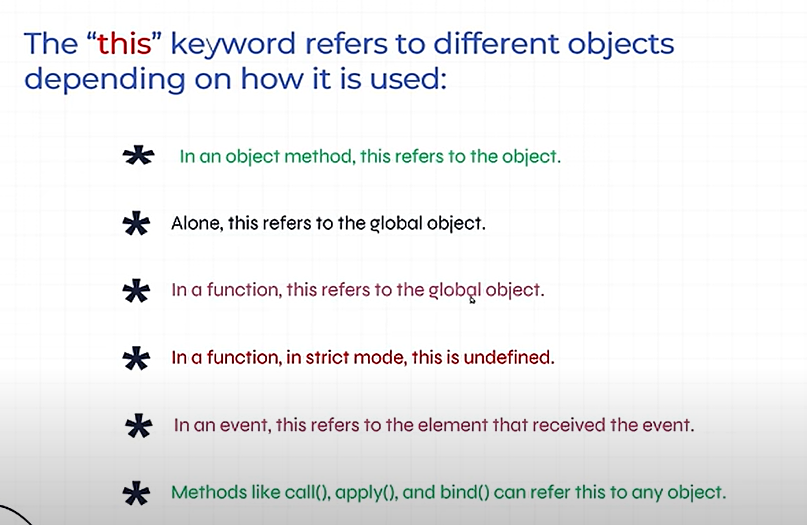
**Typing Casting:**

1. In JS we have two type of casting implicit and explicit

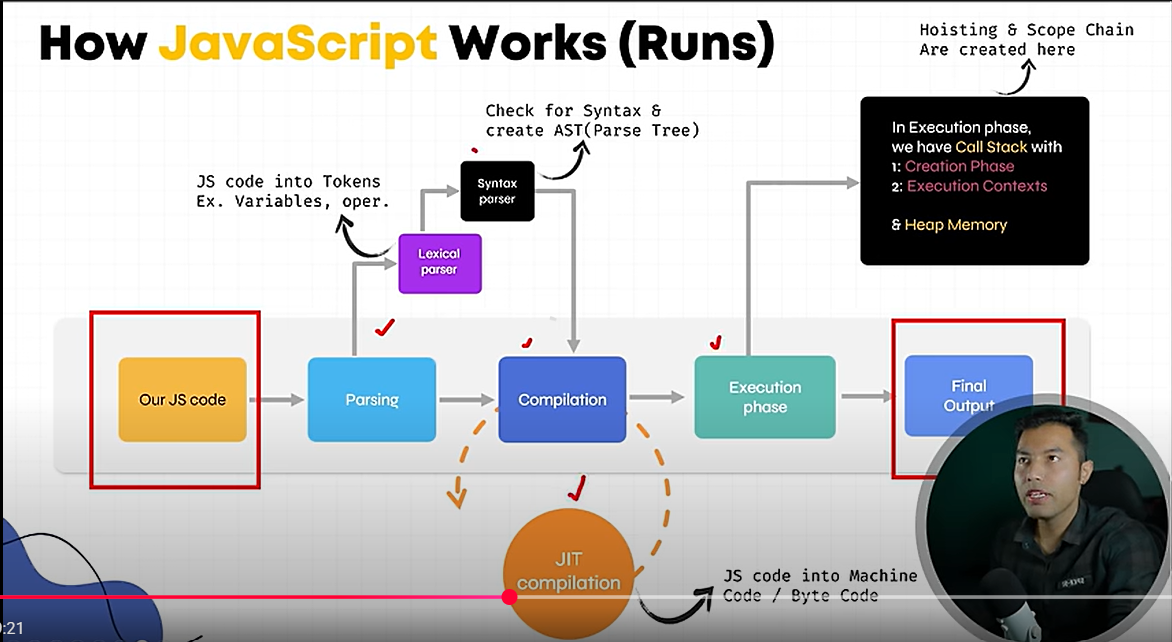
**Problem with object:**

* We cant print object with string like that **{console.log(‘origional object’ + obj)}** it will through error like that **{original object[object object]}** for solving this reason we use that syntax . **{console.log(‘origional object’, obj)}**
* Object.assign() and spread operator.
* We can make a key in object Dynamic by sqare braket [].

**this keyword:**



**HOW JS WORK?**

****

When we run the js code or load the js application the first step is to Lexical analysis/parsing and the second step is Syntex Analysis(AST-Abstract Syntex Tree) and the third step is Compilation and the fourth step is Execution Phase.

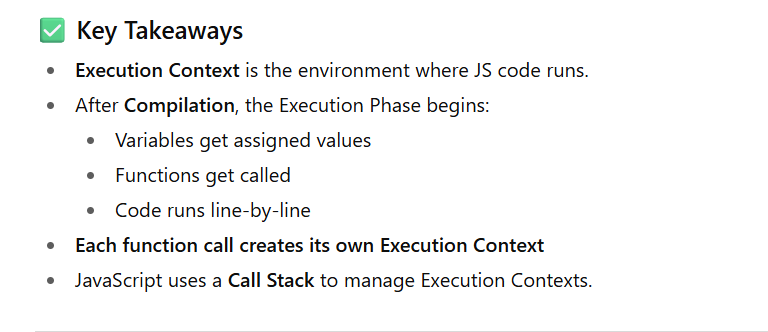
**Execution Phase:**

1. In execution phase the we have to type of context first is **Golobal Execution context** and second **function execution context.**
2. **Golobal Execution context** create when we run javascript at first time and **function execution context** create when we invoke function.
3. Every execution context goes through two main phases:

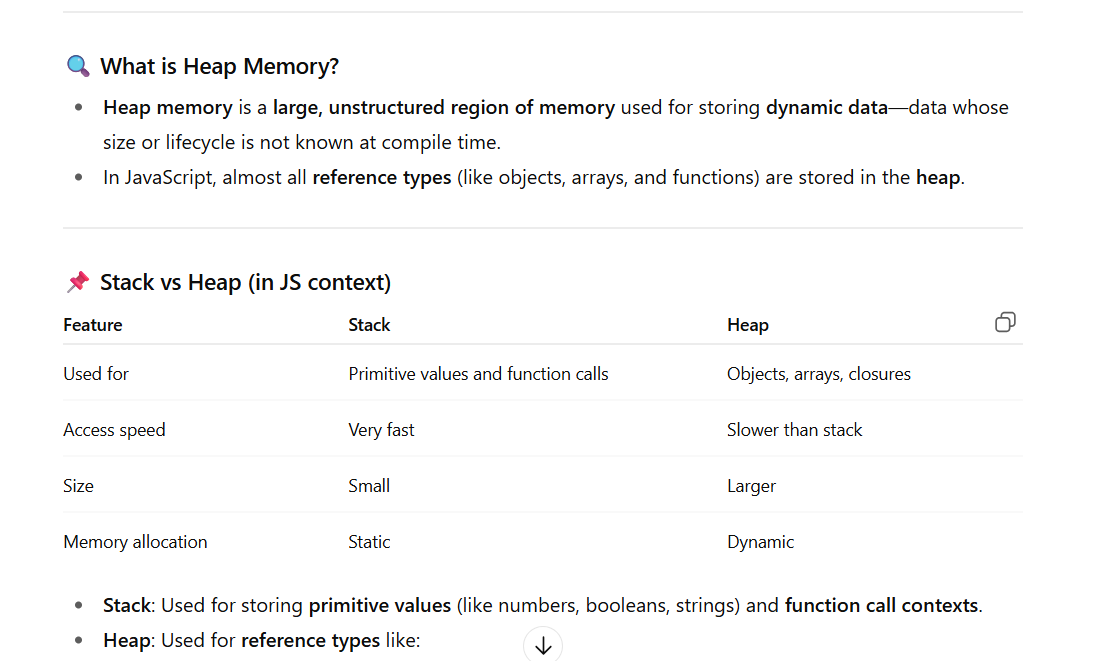
**A: Compilation (Creation) Phase**

**B: Execution Phase**

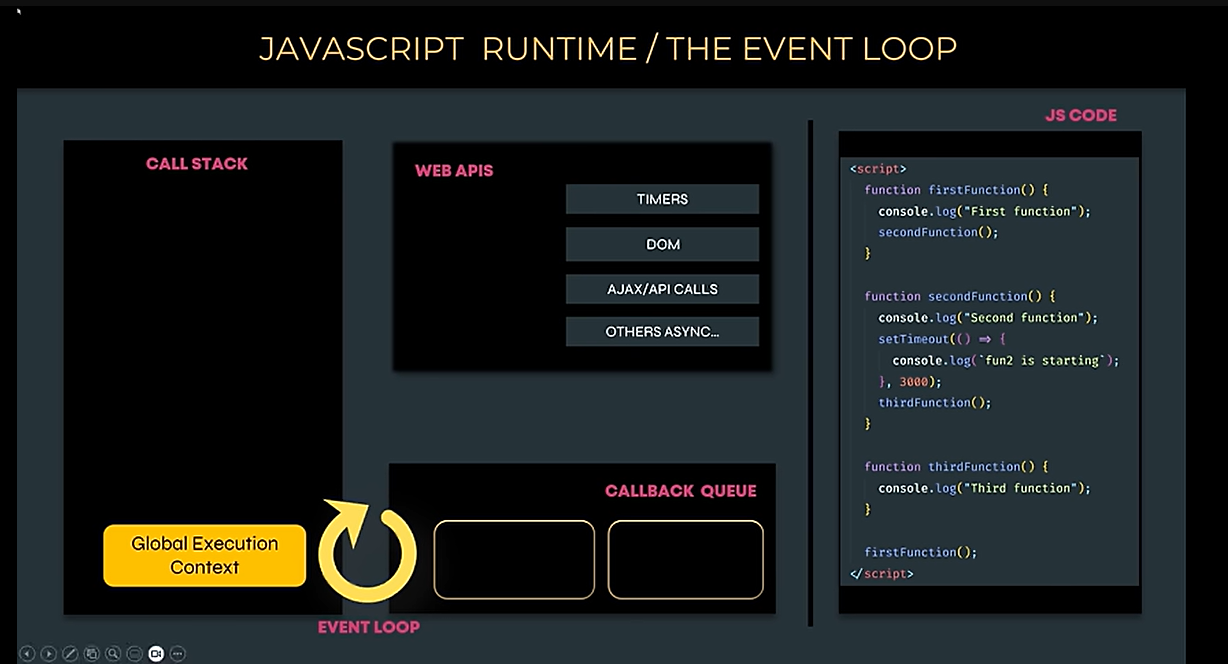
****

****

**What is heap memory in js?**

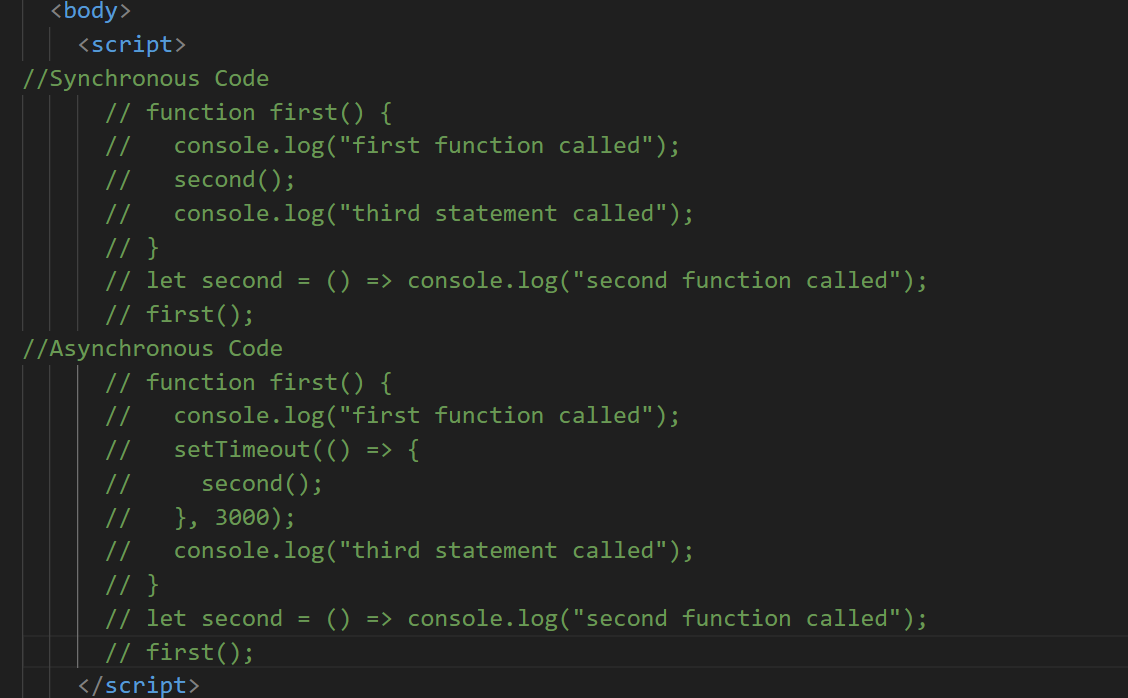
****

**JS Runtime/ Event Loop:**

****

**Note:** JS itself is single threaded but the three things WEB APIS, CALLBACK QUEUE AND EVENT LOOP make it to perform multiple task at a time.

**Synchronous & Asynchronous Code:**

****

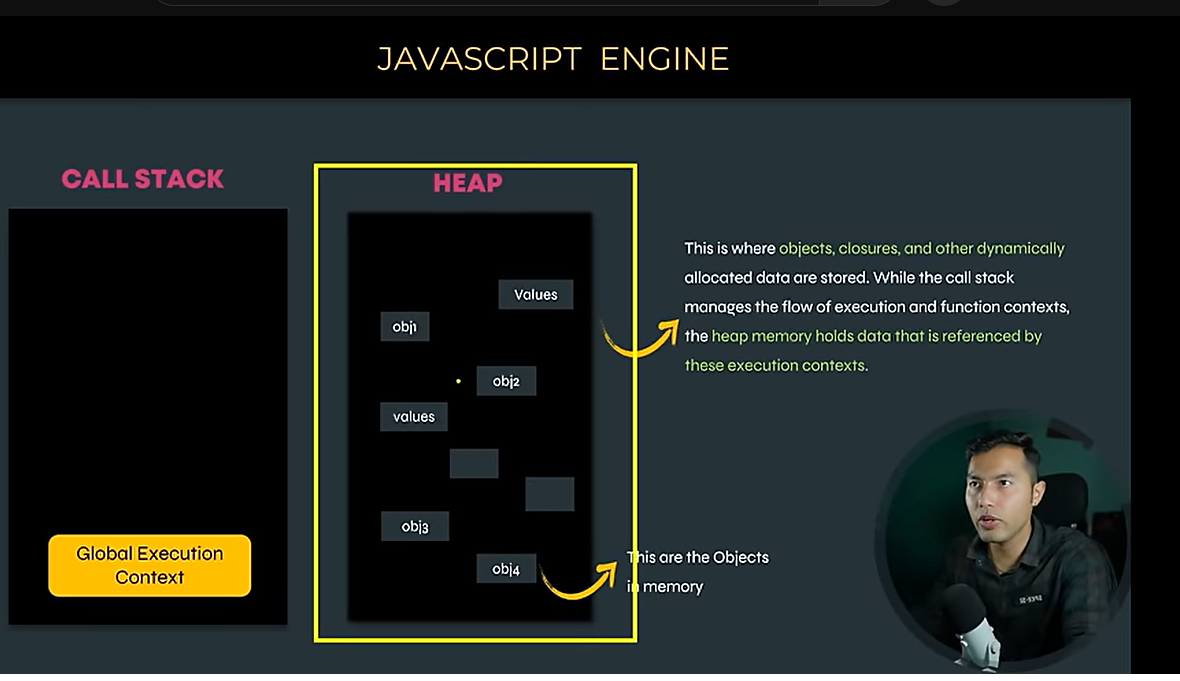
**What is Hosting in JS ?**

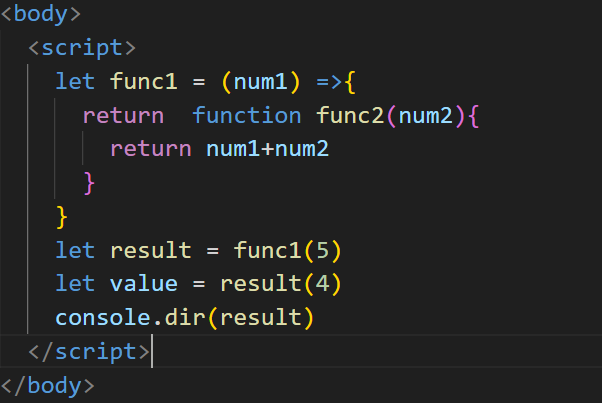
1. All variables and function declaration move on the top. And variables initialize by undefined and function initialized by reference of function which stored in memory.
2. If we call the variable before declaration then we get **undefined** if variable declared by **var** otherwise we get error.(declared by **let** or **const** get error)
3. If we call function before its declaration then we get the actual output but if the function is arrow function then we get error.

**Global Variable & Local Variable & Scope Chaining:**

Local Variable = Function Scope and block scope(like if-else)

**Closures with example code:**





**Understand Destructuring Concept of array and object?**

**Spread Operator:**

* We can use it for copy array or object
* We can use it for concatenation
* We can use it in string.

**Important Note:**

1. We cant use this keyword in arrow function for targeting cur object.
2. In arrow function if we want to access any current object variable we will use objectName.varibaleName.
3. In normal function this keyword point the current object. And can access the cur object variables through this keyword inside a function.

**Rest Parameter:**

1. Rest parameter can only 1 time occur in parameter of function.
2. We always write it at the last of function. Other parameter write first then write rest parameter

**What is Symbol?**

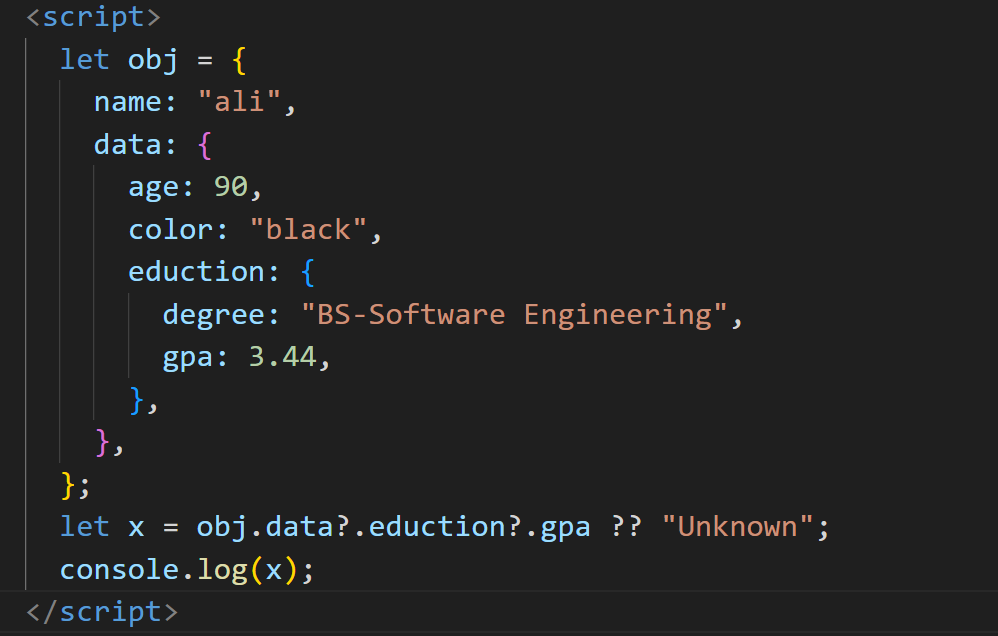
**What is BigInt?**

1. It is a privmitaive data type and store the value more than Max-Sage-Integer.
2. For creating BigInt we write n at the last of number or call BigInt constructor.  
   let num = 3434353456464n OR  
   let num = Bigint(43456456454)
3. Math object does not work with BingInt.
4. First we will convert Bigint into Number data type then Math object can work for it.

**Nullish Colescing Opertor: ??**

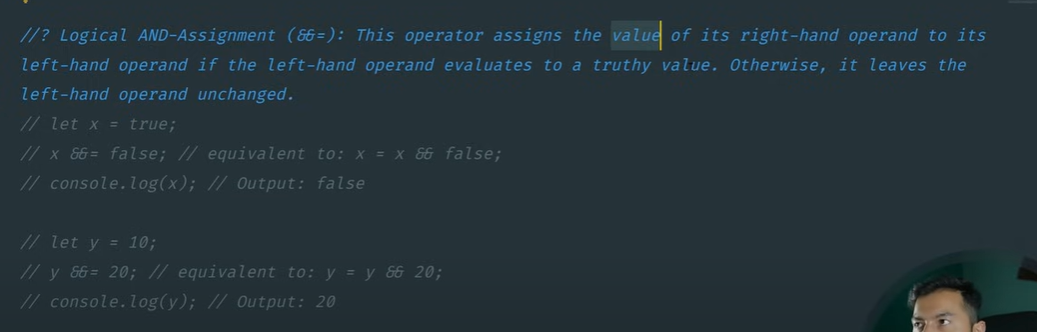
1. This operator is use for handling null or undefined.
2. It will give right hand side value if left hand side value is null or undefined other wise it will give left hand side.

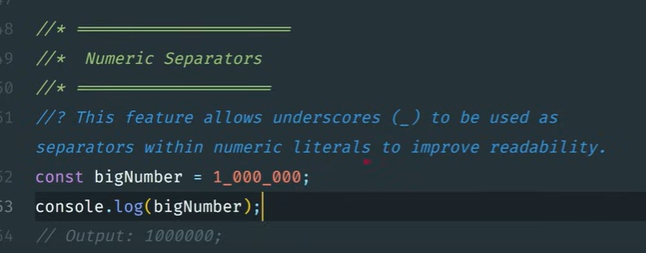
**Optional Chaining : ?.**



**Logical Operator : ||= , &&=**

1. &&=





**objectName.hasOwnProperty() and Object.hasOwn(object name, key):**

1. In both properties there is a big difference.
2. If we create object with Object.create(null) then on this object Object.hasOwnProperty() will not work.

**What is Event delegation in js?**

Event delegation in JavaScript is a pattern of handling events efficiently. Instead of attaching event listeners to multiple child elements individually, you attach a single event listener to their parent element. The parent can then listen for events that bubble up from its children and decide what to do based on the event target.

**What is the First Class Function?**

The function which we treat as a value , can assign to variable , and can pass as an argument in functions.

**What is Higher Order Function and Callback?**

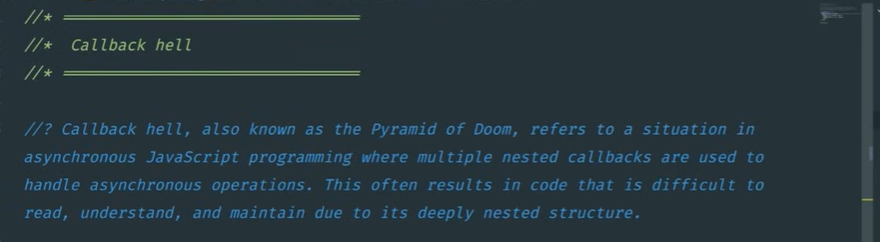
In JavaScript, a Higher-Order Function (HOF) is simply a function that does at least one of these:

* Takes another function as an argument (parameter).
* Returns a function as its result.
* Common Built-in Higher-Order Functions in JS

1. map()
2. filter()
3. reduce()
4. forEach()
5. sort()

A **callback** in JavaScript is simply a function passed as an argument to another function, and then executed later inside that function.

**What is Callback Hell?**



**What is Promises in JS?**

A Promise is a built-in JavaScript object that represents a value that may be available now, later, or never.

So Promise is just a mechanism to handle asynchronous operations.

In promise we have three state:

1. Pending
2. Resolve
3. Reject

Methods in promise is :

1. Promise.all()
2. promiseName.allSetteld()
3. promiseName.race()

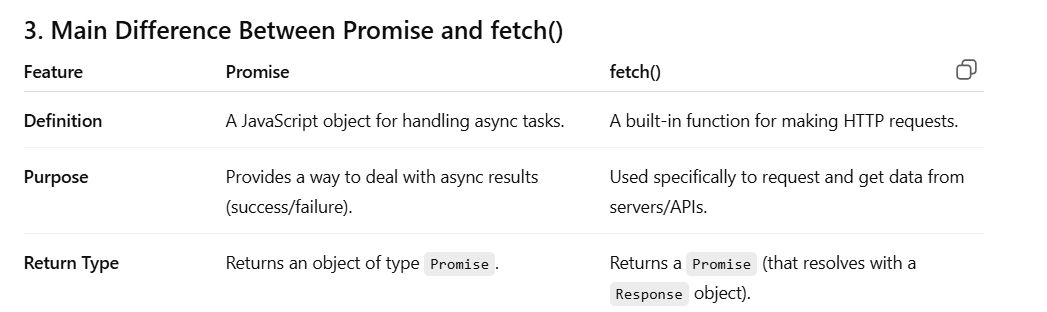
**What is fethch() method in JS?**

fetch() is a function in JavaScript used to make HTTP requests (GET, POST, PUT, DELETE) to servers or APIs.

Very important:

1. fetch() returns a Promise.
2. That promise resolves when the server responds.
3. Then you can chain .then() and .catch() just like any other promise.

**What is the difference between Promise and fetch?**



**What is async await?**

**What is try and catch method?**