1- What is JavaScript? What is the role of JavaScript engine?

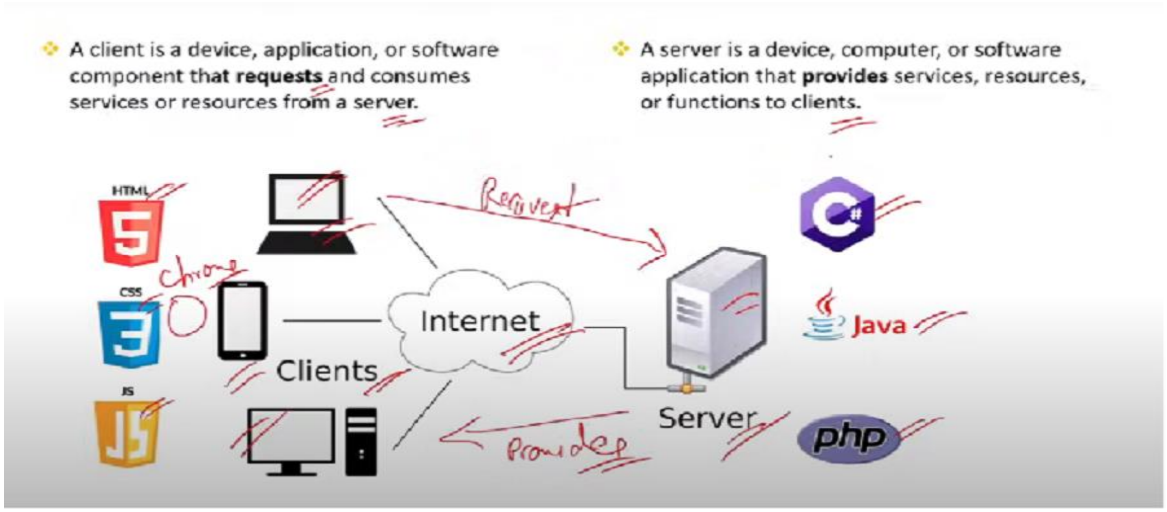
JavaScript is a programming language that is used for converting static web pages to interactive and dynamic web pages.

now the question is we receive only the JavaScript code from the server but how is the browser understanding that code and executing it that is because every browser has a JavaScript engine inside it example:

[V8 (chrome) , spider Monkey (firefox) , Javascript-Core (Safari) , Chakra (Edge , internet exploere)]

A JavaScript engine : is a program present(موجود) in web browsers that executes JavaScript code.

2- what are Client side and Server side?



A client is a device, application, or software component that **requests** and consumes services or resources from a server.
A server is a device, computer, or software application that **provides** services, resources, or functions to clients.
HTML
CSS
JS
chrome
Request
Internet
Clients
Server
Provides
Java
php

3- What is Scope in JavaScript?

Scope determines where variables are defined and where they can be accessed.

There are three types of Scopes:

1. **global scope:** where we Define the variable at top of the program like:

```
let globalVariab1e =" global " ; // global -accessible anywhere
tom();
```

2. **function Scopes** : variables are defined inside the function

```
function test(params) {
 var functionvariable="function " // Function - accessbile inside function only
}
```

3. **local or block Scope**: where the variable is defined inside an if block or elseblock or any other block

```
function test(params) {
 var functionvariable="function " // Function - accessbile inside function only
 if (true) {
   let localvariable="local or block" //Block- accessbile inside block only
 }
}
```

4- what is JSON?

JSON (JavaScript Object Notation) is a lightweight data interchange format.

5- **What is Hoisting in JavaScript?**

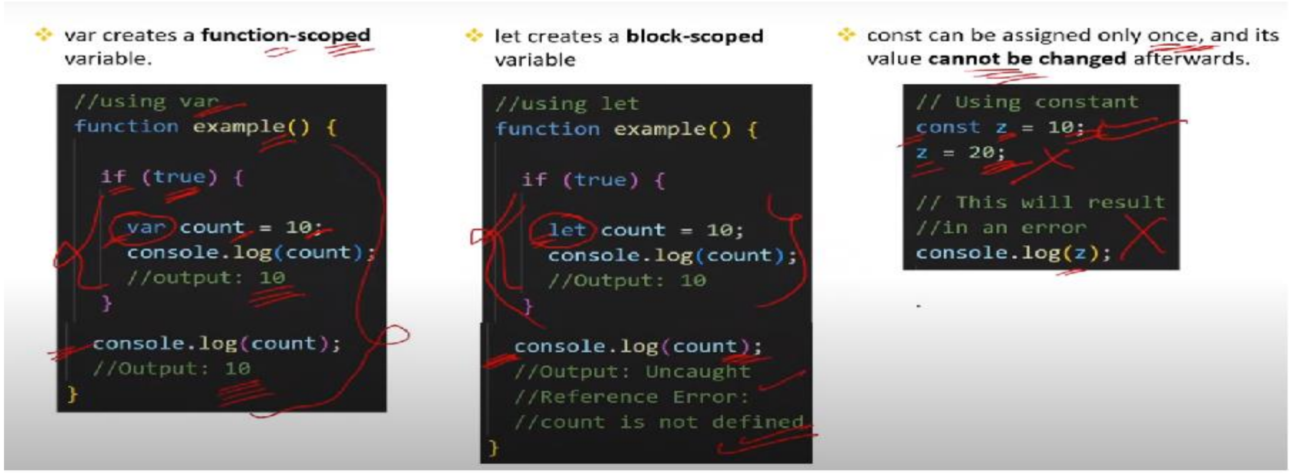
Hoisting is a JavaScript behavior where functions and variable declarations are moved to the top of their respective scopes during the compilation phase.

```
// Function hoisting
myFunction();
function myFunction() {
console. log( "Hello ! ")
}
// Output: Hello
```

The code here function is called first and then we are declaring and defining. Now the question is how is it even possible to call the function first even before declaring and defining it. This is possible because JavaScript code will be executed in top down approach then this should give an error exactly but this is possible because of Hosting. When you run this code the JavaScript engine will automatically move the method declaration to the top of the scope.

6- What is different between var , let ,const ?

item	var	let	const
Scope	function	block	block



- var creates a **function-scoped** variable.

```
//using var
function example() {

  if (true) {

    var count = 10;
    console.log(count);
    //output: 10
  }

 console.log(count);
 //Output: 10
}
```

- let creates a **block-scoped** variable

```
//using let
function example() {

  if (true) {

    let count = 10;
    console.log(count);
    //Output: 10
  }
 console.log(count);
 //Output: Uncaught
 //Reference Error:
 //count is not defined
}
```

- const can be assigned only once, and its value **cannot be changed** afterwards.

```
// Using constant
const z = 10;
z = 20;

// This will result
//in an error
console.log(z);
```

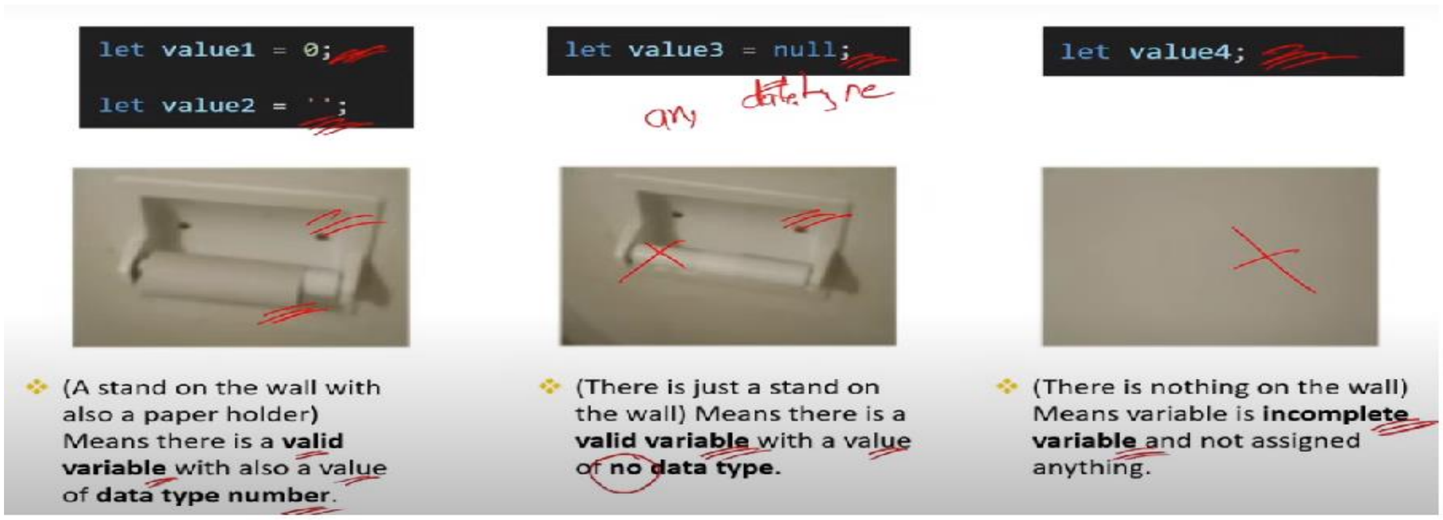
7- What are datatypes in JS?

A data type determines the type of variable. And type of datatypes:

primitive	**Non-primitive**
Number - String – Undefined – Null-Boolean	Object – Array – RegExp – Function – Date
Primitive data types can hold only single value.	Non-primitive data types can hold multiple values and methods.
Primitive data types are immutable (ثابت) and their values cannot be changed.	Non-primitive data types are mutable and their values can be changed.
Primitive data types are simple data types.	Non-primitive data types are complex data types.

8- **What is the difference between null and undefined in JS?**

Null	**Undefined**
null variables are intentionally(عمدا) assigned the null value (absence of data).	When a variable is declared but has not been assigned a value, it is automatically initialized with undefined.
Null can be used, when you are sure yo do not have any value for the particular variable.	Used when you don't have the value right now, but you will get it after some logic or operation.



```
let value1 = 0;

let value2 = '';
```

```
let value3 = null;
```

```
let value4;
```

- (A stand on the wall with also a paper holder) Means there is a **valid variable** with also a value of **data type number**.
- (There is just a stand on the wall) Means there is a **valid variable** with a value of **no data type.**
- (There is nothing on the wall) Means variable is **incomplete variable** and not assigned anything.

9- **What is type coercion (type casting) in JS?**

```
var numVar=10
    var strVar="10"
    var boolVar=true
    var nullVar=null
    console.log(numVar + strVar) // string concatenate number convert to string 1010
    console.log(numVar - strVar) // 0
    console.log(numVar == strVar) // true
    console.log(numVar + boolVar) // 11
    console.log(nullVar + boolVar) // 1
```

Type coercion is the automatic conversion of values from one data type to another during certain operations or comparisons and can be use in:

1. Can be used during String and Number concatenation.
2. Can be used while using comparison operators .

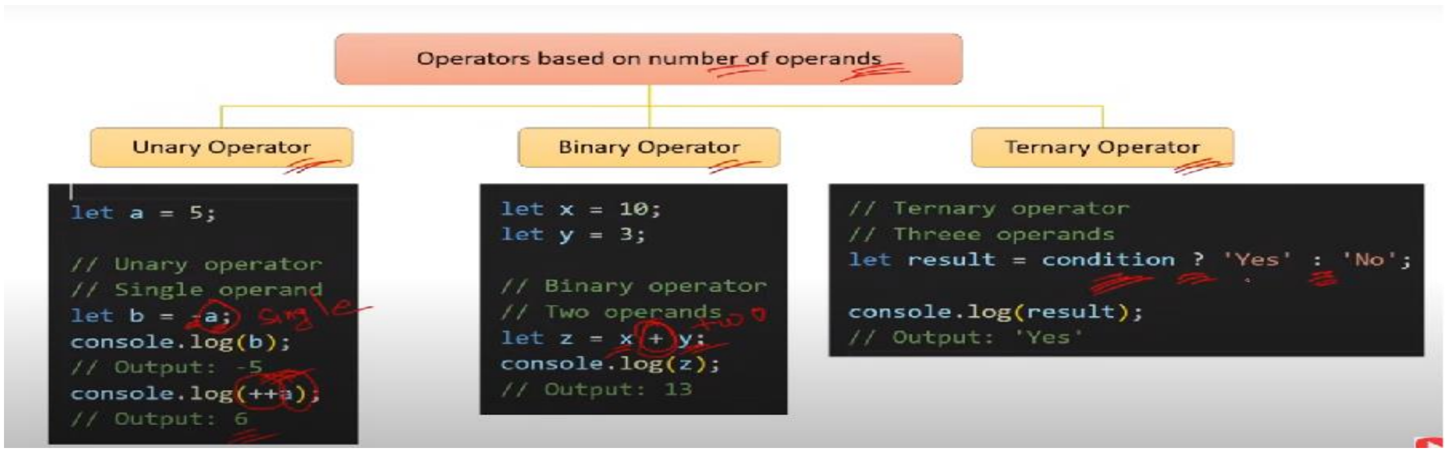
10- **What is operator precedence (الأولوية)?**

As per operator precedence, operators with higher precedence are evaluated first.

```
let a=6
let b=3
let c=2
/* bracketOf - Division - Multipliction - add - sub */
let result=a+b*c+(a-b) //6+3*2+(3)=15
console.log(result)
```

11- What is the difference between unary, binary and Ternary Operator?

- **(+) unary plus** ➔return number if its not number
- **(-) unary Negative** ➔retrun number if its not number + negative it



```
let a = 5;

// Unary operator
// Single operand
let b = -a;
console.log(b);
// Output: -5
console.log(++a);
// Output: 6
```

```
let x = 10;
let y = 3;

// Binary operator
// Two operands
let z = x + y;
console.log(z);
// Output: 13
```

```
// Ternary operator
// Threee operands
let result = condition ? 'Yes' : 'No';

console.log(result);
// Output: 'Yes'
```

12- What is the difference between double equal to and triple equal (== and ===)?

```
 // Loose Equality
console. log(1== "1")
console. log(true == "1")
console. log(true == 1)
// Output: true
```

1. **Loose Equality (==)** compares two values for equality after performing type coercion

```
  // strict Equality
console. log(1=== "1")
console. log(true === "1")
console. log(true === 1)
// Output: false
```

2. **Strict Equality (===)** operator compares two values for equality without performing type coercion.

13- What is the difference between Spread (نشر) and Rest operator in JS?

```
 //spread operator
  let arr=[1,2,3]
  console.log(arr)  //[1,2,3]
  console.log(...arr)  // 1 2 3
```

The spread operator(...) is used to expand(توسيع) or spread elements from an iterable (such as an array, string, or object) into individual elements.

The spread operator used in:

1. Coping an Array:

```
//coping an Array
  let arr=[1,2,3]
  let copingarr=[...arr]
  console.log(copingarr)  //[1,2,3]
```

2. Merging Arrays:

```
//merging Arrays
  let arr1=[1,2,3]
  let arr2=[4,5]
  let margeArr=[...arr1,...arr2]
  console.log(margeArr)  //[1,2,3,4,5]
```

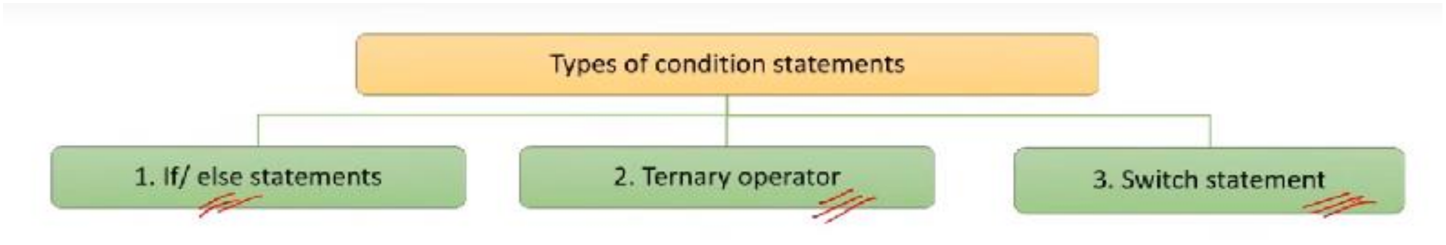
3. Passing multiple arguments to a function:

```
  //3.  Passing multiple arguments to a function
  let arr1=[1,2,3,4,5,15]
  sum(...arr1)
  function sum(...params) {
    let result=0
    params.forEach((i)=>{
         result=result + i
    })
    console.log(result)
  }
```

The rest operator is used in function parameters to collect all remaining arguments into an array.

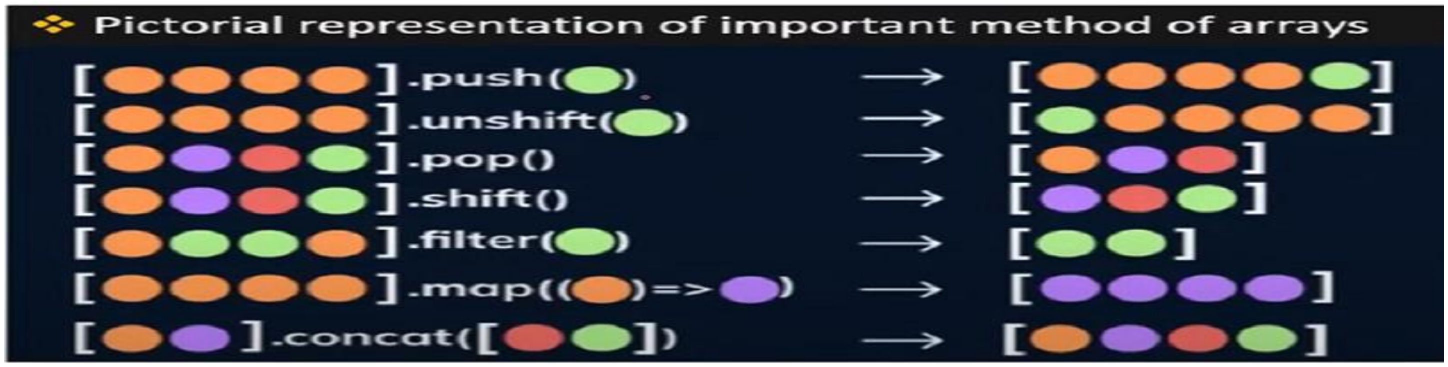
```
sum(1,2,3,4,5)
 function sum(one,two,...params) {
   console.log(one)  //1
   console.log(two) //2
   console.log(params) //[3,4,5]
 }
```

14- what are the types of conditions statements in JS?



15- What are Arrays in JS? How to get, add & remove elements from arrays?

An array is a data type that allows you to store multiple values in a single variable**.**



16- What is the indexOf() method of an Array?

IndexOf () method gets the index of a specified element in the array.

17- What is the difference between slice () find () and filter () methods of an Array?

1. **Find ():** method get the first element that satisfies a condition,return undefined if not found any ele.
2. **Filter ():** method get an array of elements that satisfies a condition and if the one element satisfies the condition will return array of one element , return undefined if not found any ele..
3. **Slice ():** method get a subset from start index to end index (end not included).

```
const numbers = [5, 12, 8, 130, 44];
// find method()
const firstEven = numbers.find(number => number % 2 === 0);
console.log(firstEven); // Output: 12
// filter method
const allEven = numbers.filter(number => number % 2 == 0);
console.log(allEven); // Output: [12, 8, 130, 44]
//---------------------------------------------
const allEven2= numbers.filter(number => number % 2 !== 0);
console.log(allEven2); // Output: [5]
// slice method
const silceEven=numbers.slice(1,4)
console.log(silceEven); // Output: [12, 8, 130]
```

18- What is the splice () method of an Array?

The splice () method is used to add, remove, or replace elements in an array.

```
// array. splice(startlndex, deleteCount, ...itemsToAdd);
   /* let test = ["a" ,"b" , "c"] */
   // add x and y to array
   test.splice(1,0,"x","y")
   console.log(test)  // ["a" ,"x","y","b" , "c"]
   // remove b from array
   let test = ["a" ,"b" , "c"]
   test.splice(1,1)
   console.log(test) //["a" , "c"]
   // replace b by kero in array
   let test = ["a" ,"b" , "c"]
   test.splice(1,1,"kero")
   console.log(test)  //["a" ,"kero", "c"]
```

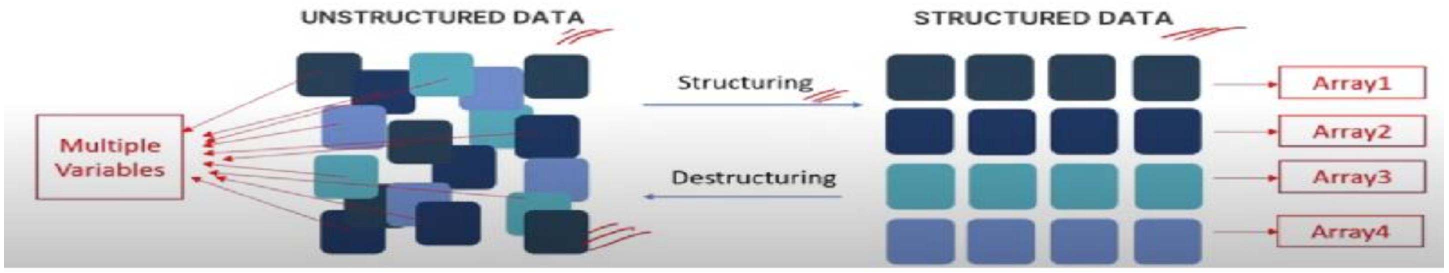
19- What is the difference between the map() and forEach () methods of an Array?

1. **Map ():** method is used when you want to modify each element of an array and create a new array with the modified values.
2. **forEach()**: method is used when you want to perform some operation on each element of an array without creating a new array.

```
//    Using Map
let maparr=[1,2,3]
   let newArr=maparr.map((m)=>m * 2)
   console.log(newArr) //[2,4,6] return new array
 console.log(maparr) //[1,2,3] return origin array
   //using forEach
   let forarr=[1,2,3]
   forarr.forEach((e)=>{ //does not return anything
    console.log(e * 2)}) //2 4 6
    console.log(forarr) // [1,2,3]
```

20- what is Array Destructuring in JS?

array destructuring allows you to extract elements from an array and assign them to individual variables in a single statement.



```
     let fruits=["banana" , "apple" ,"orange"]
 //array destructuring
 let [a,b,c]=fruits
 console.log(a) //"banana"
 console.log(b)  //"apple"
 console.log(c)  //"orange"
```

21- What is a loop? What are the types of loops in JS?

A loop is a programming way to run a piece of code repeatedly until a certain condition is met.

22- What is the difference between break and continue statement?

1. **Break: The "break" statement is used to terminate (إنهاء) the loop.**

```
  for (let index = 1; index <=5; index++) {
     if(index === 3) {
        break;
     }
     console.log(index) //  1 2
```

2. **Continue: The "continue" statement is used to skip the current iteration of the loop and move on to the next iteration.**

```
for (let index = 1; index <=5; index++) {
     if(index === 3) {
        continue;
     }
     console.log(index) //  1 2 4 5
```

23- When to use for..of loop and when to use forEach method in applications?

```
let arr=[1,2,3]
for (const iterator of arr) {
console.log(iterator)
if (iterator == 2) {
break; // 1 2 } }
arr.forEach((item)=>{
console.log(item)
if (item == 2) {
break; //Illegal break statement
}})
```

- **for...of loop is** suitable when you need more control over the loop, such as using break statement or continue statement inside.
- **forEach** method iterate (يكرر / يدور) over each element of the array and perform some action on each element.

24- What is function in js ? what are types of function?

A function is a reusable block of code that performs a specific task**.**

(Named, Anonymous, Function Expression, Arrow, IIFE , Callback, higher-Order)

25- What is the difference between named and anonymous? When to use in applications?

Named functions have a name identifier

```
function sum(a, b){
    return a + b;
}
console.log(sum(2,3))// Output: 5
```

- Use named functions for big and complex logics.
- Use when you want to reuse one function at multiple places.

Anonymous functions do not have a name identifier and cannot be referenced or called directly by name.

```
mybtn.addEventListener("click",function () {
        console.log("click")
        })
```

- Use anonymous functions for small logics.
- Use when want to use a function in a single place.

26- What is Function Expression?

A function expression is a way to define a function by assigning it to a variable.

```
   let test=function sum(a,b){
    return a+b
}
console.log(test(1,2)) // 3
console.log(sum(1,2))  //error sum is not function
```

27- What is arrow function?

Arrow functions, also known as fat arrow functions, is a simpler and shorter way for defining function in JS.

```
      let add =(x,y)=>{return  x+y}  or  let add =(x,y)=> x+y

     console.log(add(3,12))
```

28- What is callback function?

A callback function is a function that is passed as an argument to another function.

```
    function add(x,y) {return x + y}
     function subtract(x,y) {return x - y}
     function multply(x,y) {return x * y}
     function display(x , y, opertion) {
         let result=opertion(x,y)
         console.log(result)}
     display(10,5,add) //15
     display(10,5,subtract) //5
     display(10,5,multply)  //50
     setTimeout(() => {
        console.log("i am callback")
      }, 1000);
```

29- What is Higher-order function In JS?

Take one or more functions as arguments (callback function) OR return a function as a result.

30- What is the difference between arguments and parameters?

Parameters are the placeholders (العناصر النائبة) defined in the function declaration.

Arguments are the actual values passed to a function when it is invoked or called.

```
 // a and b are parameters
      function add(a ,b) {
        console.log(a+b)
      }
      // 3 and 4 are arguments
      add(3,4)
```

31- What is default parameters in functions?

In JavaScript, default parameters allow you to specify default values for function parameters.

```
function test(name="kerolos") {
       console.log("hello " + name)
     }
     test()  //hello kerolos
     test("ahmed")  //hello ahmed
```

32- What are Pure(نقية) and Impure functions in JS?

- **A pure function** is a function that always produces the same output for the same input.
- **Pure functions** cannot modify the state.

```
    // pure function
     function add(x ,y ) {return x + y}
     console.log(add(2,3))  //5
     console.log(add(2,3))  //5
```

- **An impure function**, can produce different outputs for the same input.
- **An impure functions** can modify the state (change the value of total in next example).

```
  let total=0
     function addToTotal(value) {
        total+=value
        return total}
       console.log(addToTotal(5))  //5
       console.log(addToTotal(5))  //10
```

33- What is Function Currying in JS?

Currying in JavaScript transforms a function with multiple arguments into a nested series of functions, each taking a single argument.

```
//  normal function
    function add(a, b, c) {
        return a + b + c;
    }
    console.log(add(1,2,3)); // Output: 6
    // curring function
function curriedAdd(a) {
  return function(b) {
    return function(c) {
      return a + b + c;
    };
  };
}
// Using the curried function
console.log(curriedAdd(1)(2)(3)); // Output: 6
```

34- What are call, apply and bind methods in JS?

1. **The call method** invokes a function with a specified this (object context) value and arguments provided individually**.**

```
function greet(greeting, punctuation) {
  console.log(greeting + ', ' + this.name + punctuation);
}
const person = { name: 'Alice' };
greet.call(person, 'Hello', '!'); // Output: Hello, Alice!
```

2. **The apply method** is similar to call, but it takes the arguments as an array.

```
function greet(greeting, punctuation) {
  console.log(greeting + ', ' + this.name + punctuation);
}
const person = { name: 'Alice' };greet.apply(person, ['Hello', '!']); // Output: Hello,Alice!
```

3. **The bind method** creates a new function that, when called, has its this value set to the provided value, with a given sequence of arguments preceding any provided when the new function is called.

```
function greet(greeting, punctuation) {

  console.log(greeting + ', ' + this.name + punctuation);
}
const person = { name: 'Alice' };
const kero =greet.bind(person)
kero("hello" ,"!") // Output: hello, Alice! */
```

- call and apply invoke the function immediately.
- bind returns a new function that can be invoked later.

35- What are template literals and string interpolation in strings?

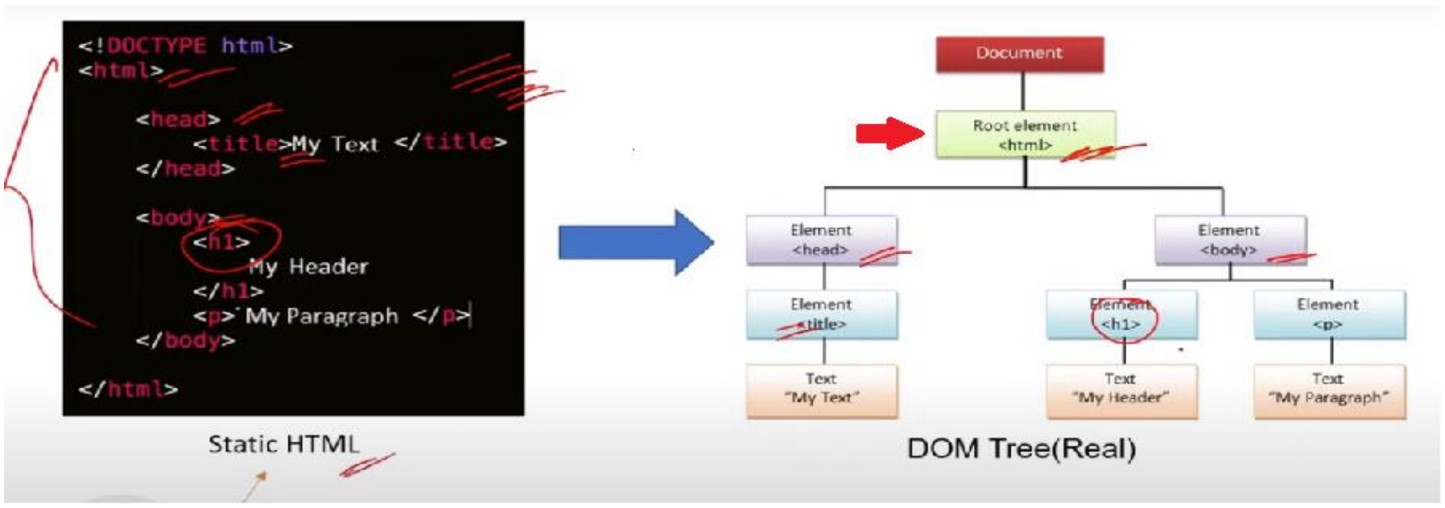
A template literal, also known as a template string, is a feature introduced in ECMAScript 2015 (ES6) for string interpolation and multiline strings in JavaScript.

```
 // templete literal with interpolation (backtick)
   let name ='kerolos'
   console.log(`hello ${name}`)  //hello kerolos
   // ----------------------------------------------
   // templete literal with multiline string (backtick)
   let multi=`
   hello sir
   my name is kerolos
   `
```

36- How many ways you can concatenate strings?

```
let s1="hello" ; let s2="kerolos"
    // + operator
    console.log(s1 + s2)  //hello kerolos
   //  concat method
   console.log(s1.concat(s2))  //hello kerolos
   //templete literal
   console.log(`${s1} ${s2}`)  //hello kerolos
   //join
   let s3=[s1,s2]
   console.log(s3.join(" ")) //hello kerolos
```

37- What is DOM? What is the difference between HTML and the DOM?



```html
<!DOCTYPE html>
<html>

    <head>
        <title>My Text </title>
    </head>

    <body>
        <h1>
            My Header
        </h1>
        <p> My Paragraph </p>
    </body>

</html>
```

The DOM(Document Object Model) represents the webpage as a tree-like structure that allows JavaScript to dynamically access and manipulate the content and structure of web page.

in short HTML is just a markup language for reading and writing purposes for developers and users in reality memory plays with Dom.

38- What is selector in JS?

Selectors in JS help to get specific elements from DOM based on IDs, class names, tag names.
(getElementById() - getElementsByClassName() - getElementsByTagName()-querySelector() –querySelectorAll())

39- What are the methods to modify elements properties and attributes?

(textContent – innerHTML -setAttribute()-removeAttribute() - style.setProperty-classList.add())

40- What is different between textContent and innerHTML?

```javascript
let element = document.getElementById("example");
element.innerHTML="Hello <strong>kerolos</strong>" //in browser Hello kerolos
element.textContent="Hello<strong>kerolos</strong>"//in browser Hello<strong>kerolos</strong>
element.textContent="Hello<strong>kerolos</strong>"//in browser Hello<strong>kerolos</strong>
```

41- How to add or remove properties of HTML elements in the DOM ?

We can add or remove properties using setAttribute or removeAttribute method.

```javascript
   let element = document.getElementById("example");
   element.setAttribute("data-info","new value")
   element.removeAttribute("data-info")
```

42- How to add or remove style of HTML elements in the DOM ?

We can add or remove style using setProperty or classList method.

```javascript
 let element = document.getElementById("example");
   element.style.setProperty("background-color" , "red")
   element.classList.add("active")
  element.classList.remove("active")
```

43- What is cloneNode() method?

```
<div id="parent">

        <h1>child</h1>
    </div>
   <script>
   let element = document.getElementById("parent");
   let cloneEle=element.cloneNode(true)
   cloneEle.textContent='i am clone '
   document.body.append(cloneEle)
   //the output in DOM is <div id="parent"> i am clone   </div>
```

44- What is Error handling in JS? Error handling is the process of managing errors.

```
   try {
       let result=somevarible + 10
 console.log(result)
 } catch (error) {
console.log("error in code is "+error.message) //error in code is somevarible is not defined}
```

45- What is the role of finally block in JS?

Finally, block is used to execute some code irrespective (بغض النظر) of error.

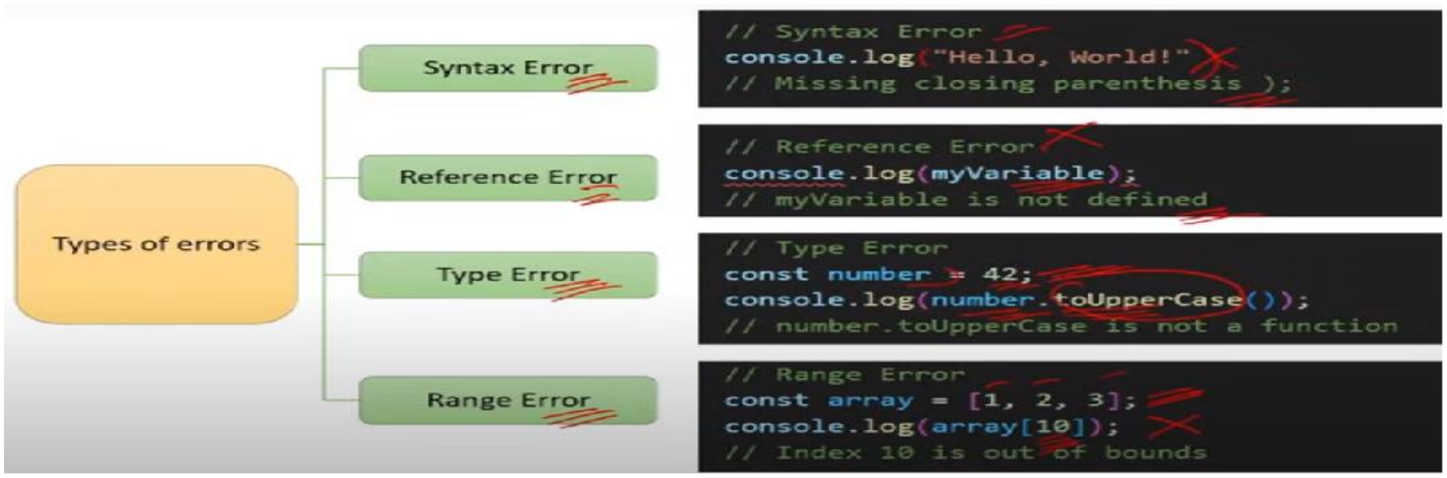
46- What is the purpose of the throw statement in JS?

The throw statement stops the execution of the current function and passes the error to the catch block of calling function.

```
function userData() {
    try {
        validateAge(10)
        validateAge("10")
        validateAge(25)
    } catch (error) {
        console.log(error.message)
    }

}
function validateAge(age) {
    if (typeof age !== "number") {
        throw new Error("this age not valid")
    }
    console.log("this age valid "+ age)
}
userData()
//output
// this age valid 10
//  this age not valid
```

47- what are the different types of errors In JS?



```
// Syntax Error
console.log("Hello, World!"
// Missing closing parenthesis );

// Reference Error
console.log(myVariable);
// myVariable is not defined

// Type Error
const number = 42;
console.log(number.toUpperCase());
// number.toUpperCase is not a function

// Range Error
const array = [1, 2, 3];
console.log(array[10]);
// Index 10 is out of bounds
```

48- what is object in JS?

An Object is a data type that allows you to store key-value pairs.

49- What is the difference between an array and an object?

Array	Object
Arrays are collection of value	Objects are collections of key-value pairs.
Arrays are denoted by square brackets ()	Objects are denoted by curly braces {}.
Elements in array are ordered (zero index).	Properties in objects are unordered.

50- how do you add or modify or delete properties of an object

```
let myobj={}
   //add properties
   myobj.name="kerolos"
   //modify properties
   myobj.name="amr"
   //delete properties
   delete myobj.name
```

51- Explain the difference between dot notation and bracket notation?

- Both dot notation and bracket notation are used to access properties or methods of an object.
- Dot notation is more popular and used due to its simplicity.

```
 let myobj={
         name:"kerolos",
         age:30}
         console.log(myobj.name) // dot notation kerolos
         console.log(myobj["name"]) //bracket notation kerolos
```

Limitation of dot notation - In some scenarios bracket notation is the only option, such as when accessing properties when the name is stored in a variable.

```
let myobj={
        name:"kerolos",
        age:30}
        let myvarible="age"
        console.log(myobj.myvarible) // dot notation undefined
        console.log(myobj[myvarible]) //bracket notation 30
```

52- What are some common methods to iterate over properties in Object?

```
let myobj={
     name:"kerolos",
     age:30}
    //   using for... in loop
    for (const key in myobj) {
       console.log(key + " "+ myobj[key]) //name kerolos age 30
    }
   // using keys() method
   Object.keys(myobj).forEach((param)=>{
     console.log(param + " "+ myobj[param]) //name kerolos age 30
   })
   // using values() method
   Object.values(myobj).forEach((values)=>{
     console.log(values) // kerolos 30
   })
```

53- How do you check if a property exists in an object?

```
let myobj={
     name:"kerolos",
     age:30}
    //   using the in Operator
    console.log("name" in myobj) // true
    console.log("city" in myobj) //false
    //   using hasOwnProperty method
    console.log(myobj.hasOwnProperty("name")) // true
    console.log(myobj.hasOwnProperty("city")) // false
```

54- how do you clone or copy an object?

```
let myobj={
      name:"kerolos",
      age:30}
     //   using spread syntax (shallow copy نسخة سطحية)
     const copyObject={...myobj}
    console.log(copyObject)
    //    using assign method() (shallow copy)
    const copyObject2=Object.assign({},myobj)
    console.log(copyObject2)
    // using JSON.parse() and JSON.stringify()  (deep copy)
    const copyObject3=JSON.parse(JSON.stringify(myobj))
    console.log(copyObject)
```

55- What is the difference between deep copy and shallow copy?

```
let myobj={
      name:"kerolos",
      address:{
          city:'assuit',
          street:"3"
      }}
 const shalloecopy=Object.assign({},myobj)
  shalloecopy.address.city="cairo"
  console.log(shalloecopy.address.city) //cairo
  console.log(myobj.address.city)  //cairo
 //==============================
 const deepcopy=JSON.parse(JSON.stringify(myobj))
  deepcopy.address.city="Alex"
  console.log(deepcopy.address.city) //Alex
  console.log(myobj.address.city) //assuit
```

Shallow copy in **nested objects** case will **modify** the parent object property value, if cloned object property value is changed. But deep copy will not modify the parent object property value.

56- What is Set Object in JS?

The Set object is a collection of **unique** values meaning that duplicate values are not allowed.

```
const uniqueVal=new Set()
uniqueVal.add(5)
uniqueVal.add(10)
uniqueVal.add(5) // ignore duplicate values
console.log(uniqueVal.size) // 2 (10 , 5)
console.log(uniqueVal.has(10)) //true
uniqueVal.delete(10)
console.log(uniqueVal.size) // 1 (5)
//remove duplicate value in array
let arr=[1,4,2,4]
const myobj=new Set(arr)
let newarr=[...myobj]
console.log(newarr) // [1,4,2]
```

57- What are Events ?

Events are actions that happen in the browser, such as a button click, mouse movement, or keyboard input.

58- What are Event Object in JS ?

Whenever any event is triggered, the browser automatically creates an event object and passes it as an argument to the event handler function.

The event object contains various properties and metheds that provide information about the event, such as the type of event , the element that triggered the event etc.

```
let mybtn=document.getElementById("mybtn")
mybtn.addEventListener("click" , handleEvent)
function handleEvent(event) {
    console.log(event.target) // <button id="mybtn">click</button>
    console.log(event.type)  //click
}
```

59- what is Event Delegation in JS?

Event delegation in JavaScript is a technique where you attach a single event handler to a parent element to handle events on its child elements.

```
 <ul id="mylist">
        <li>item1</li>
        <li>item2</li>
        <li>item3</li>
    </ul>
<script>
let mybtn=document.getElementById("mylist")
mybtn.addEventListener("click" , handleEvent)
function handleEvent(event) {
    console.log(event.target.textContent)
}
```

When click in item1 …item1 will appearance in console and When click in item2 …item2 will appearance in console ……

60- What is Event Bubbling In JS?

Event bubbling is the process in JavaScript where an event triggered on a child element propagates up the DOM tree, triggering event handlers on its parent elements.

```
    <div id="outer">
        <div id="inner">
            <button id="mybtn">dasd</button>
        </div>
    </div>
<script>
let mybtn=document.getElementById("mybtn")
let outer=document.getElementById("outer")
let inner=document.getElementById("inner")
outer.addEventListener("click" , handleEvent)
inner.addEventListener("click" , handleEvent)
mybtn.addEventListener("click" , handleEvent)
function handleEvent(event) {
    console.log(this.id) // mybtn inner outer
}
```

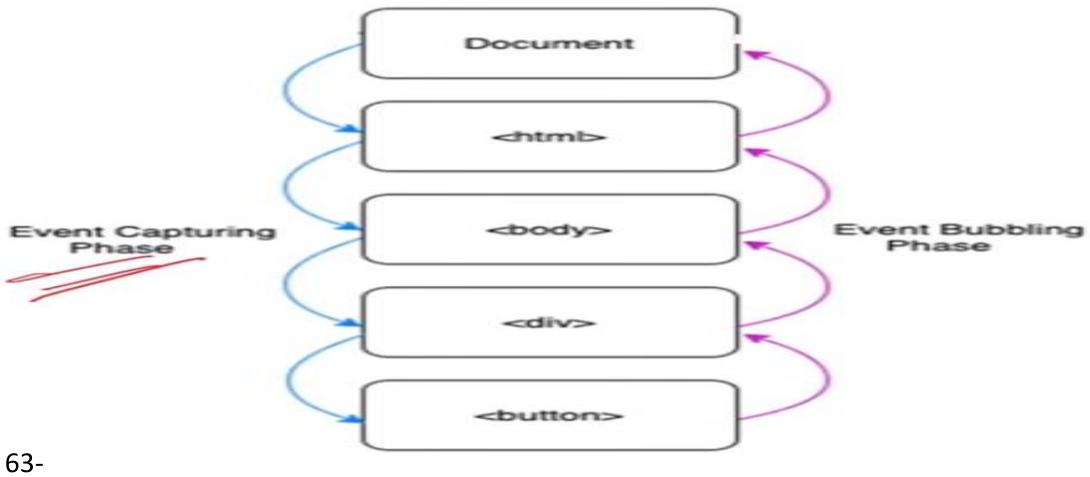
61- How can you stop event propagation (إنتشار) or event Bubbling in js?

```
function handleEvent(event) {
       console.log(this.id)
       event.stopPropagation() //mybtn
}
```

62- What is Event capturing In JS?

Event capturing is the process in JavaScript where an event is handled starting from the highest-level ancestor (the root of the DOM tree)and moving down to the target element (opposite of event bubbing).

```
    <div id="outer">
        <div id="inner">
            <button id="mybtn">dasd</button>
        </div>
    </div>
<script>
let outer=document.querySelector("#outer")
let inner=document.querySelector("#inner")
let mybtn=document.querySelector("#mybtn")
outer.addEventListener("click",kkk,true)
inner.addEventListener("click",kkk,true)
mybtn.addEventListener("click",kkk,true)
function kkk(event) {
    console.log(this.id) //outer inner mybtn
}
```



63-