1=>
what is javascript ? what is role of js engine? (V imp)
Ans=>
js is programming language that is used for converting static web pages to intractive and dynamic web pages

a js engine is a program present in web browsers that executes js code.
crome(v8)   firefox(spider monkey) safari(javascript core) chakra (edge)

2=>
what is client side and server side?
Ans=>
A Client is device application or software components that requests and consumes services or resources from a server.

A Server is a device computer or software application that provides services resources or function to client.

3=>
what is variable? what is difference between var let and const?
Ans=>
variable are used to store data.

var creates a function scoped variable
let creates a block scoped variable
const can assigned only once and its value connot be changed afterwards.

i. Reaassignment:
:VAR allows reassignment and can be redeclared within the same scope
:LET allows reassignment but cannot be redeclared within the same scope
:const doesnot allow reassignment or redeclaration after the intial assignment

ii. Hoisting:
:var is hoisted and initialized with undefined
:let and const are hoisted but not initialized. tring to access them before the declaration results in a reference error

iii. global object property:

:variable declared with var become properties of the global object

:variable declare with let and const do not become properties of the global object

ex=

```
var globalVar=60;
log(window.globalVar) //60
var globalLet=65;
log(window.globalLet) //undefined
```

4=>

what are some important string operation in js?

Ans=>

substr() substring() replace() search() toLocaleLowerCase() toLocaleUpperCase() index of() includes() slice() concat() lastIndex() charCodeAt() trim() valueOf() split() toString() match()

5=>

what is DOM? what is difference between HTML and DOM?

Ans=>

the DOM(document Object Model) represents the web pages as a tree-like-structure that allows javascript to dynamically access and manipulate the content and structure of web page.

6=>

what are selectors in js?

Ans=>

selectors in js help to get specific elements from DOM based on ID's class name and tag name

1 getElementById("") => select a single element

2 getElementByClassName("") => select multiple elements that share the same class name

3 getElementByTagName("") => select multiple elements based on their tag name

4 querySelector("")

5 querySelectorAll("")

7=>

what are data types in js?

Ans=> a data type determines the type of variables
    Primitive
    Number
    string
    Boolean
    undefined
    null

   Non-Primitive
   object
   array
   function
   date
   regExp

8=>
   what are operators? what are the types of operator in js?
Ans=>
   operators are symbols or keywords used to perform operations on operands.
   x+y=z (here x,y=operandds +,= operators z=result)
   1. Arithmetic Operator (+,-,*,/,%,**(exponentiation))
   2. Assignment Operator (=,+=,*=)
   3. Comparison Operator (==,===,>=,<=,!==)
   4 Logical Operators   ([ANT &&][OR ||] [NOT !x])
   5 String Operators   (cocatenation "shubh" + "baranwal")

9=>
   what are the types of conditions satetement in js?
Ans=>
    1. if/else statement if(){}else if(){}
    2. Ternary Operator   5>3 ? 5 : 3
    3. Switch Statement let a=5;

```
switch(a){
  case 1:
   log(1)
```

```
                 break;
                case 2:
                 log(2)
                 break;
                .
                .
                .
                default:
                 log(default)

              }
```

10=>
  what are loops? what are the types of loops in JS?
Ans=>
  a loop is programming way to run a piece of code repeatedly until a certain condition is met.
  Ex= for,while,do-while,for...of,for...in

```
  for(let i=0(initialization); i < 5(condition); i++(updation) ){
     log(i)
  }
```

11=>
  what are funtion in js ? types of function
Ans=>
  a function is a resuable block of code that perform a specific task.
  Types of function
  Named function, anonymous function , function expression, arrow function, iife, callback function
  high order function

 * in javascript are object
 *they already have some predefined properties like name,length
 *you can also make more properties on function(but generally it's not requried exept for constructer function)

```
 function sayHii(person){
     return console.log(person);
 }
 sayHii('shubh')
 console.log(sayHii.length); //no of argument
 console.log(sayHii.name); //name of function
```

*function can be called as constructor

```
function Person(name){
    this.name=name;
}
const p=new Person('john') //constructor
```

*named function expression (nfe)

```
let sayhi=function fx(user){
    if(user){
        return "hello " + user
    }else{
        return fx("anonimous");
    }
}
console.log(sayhi('shubh'));  //hello shubh
console.log(sayhi()); hello anonimous
```

12=>
    what are arrow function in js ? what is it use?
Ans=>
    arror function also known as fat arrow function is a simpler and shorter way for defining function is javascript

```
    (parameter list)=>{function body}
    const add=(a,b)=>x+y;
```

log(add(5,3))

13=>
what are arrays in js? how to get add and remove elements from array?

Ans=>
an array is a data types that allows you to store multiple values in a single variable
let fruits=["apple","banana","orange"]
Array Methods:=
get Methods:=>
indexOf(),find(),filter(),Slice()

Add Methods:=>
push() concat()

Remove Methods:=>
pop() shift() splice()

Modify Methods:=>
map() forEach()

Others Method:=>
join() length() sort() reverse() reduce() some() every()

14:=>
What are Objects in JS?

Ans=>
an object is a data type that allows you to store key-value pairs

```
let person={
  name:"shubh",
  hobbies:["teaching","footballs"],
  greet:function(){
    console.log(this.name)
  }
}
access  console.log(person.name)
```

15=>
 what is scope in javascript?
Ans=>
 Scope determines where variables are defined and where they can be accessed.
 globar=>accessbile anywhere
 function=>accessbile inside function only
 block=>accessbile inside block only

16=>
 what is hoisting in javascript?
Ans=>
 Hoisting is a javascript behavior where function and variable declarations are moved to the top of their respective scopes during the compilation phase.
 Function=>

```
myFunction()
function myFunction(){
   console.log("hello")
}
```

 ex2
 error dega function experrasion pahle access nahi kiya ja sakta hai
ye function experassion hota hai

```
 sayHello()
 let sayHello=function(){
     console.log("shubh");
 }
```

variable=>
 EX=1

```
x=10;
console.log(x)
var x;
```

```
ex=
console.log(a);
var a=10; //not give error but give undifined
```

17=>
what are error handling in js?
Ans=>
error handling is the process of managing error using try{}catch{} function

try blocks contains the code that might throw an error

```
try{
  const result=someUndefinedVar+10;
  console.log(result)
}catch(error){
  console.log("an error occured", error.message)
}
output=> an error occured: someUndefinedVar is not defined
```

18=>
what is asynchronous programming? what is it use?
Ans=>
*Asynchronous programming allows multiple tasks or operation to be initiated and executed concurently.
*Anychronous operation do not block the execution of the code.

Use of asynchronous operation
1 fetching data from api
2 downloading files
3 uploading files
4 animations and transitions
5 time taking operations...

19=>
difference between primitive and not primitive data types?
Ans=>
Primitive Data Type=>

1 Number,string boolean undefined null are primitive data tye.
2 PDT data types can hold only single value.
3 PDT are immutable and their values cannot be changed
 let age=25;
 age=30;
 isme phala age dusre memory acces karega dusra age dusra memory access karenga
4 PDT are simple data types

Non-Primitive-Data-Type
1 Object array function date regexp are NPDT
2 NPDT can hold multiple values and method
3 NPDT are mutable and their values can be changed
4 NPTD are complex Data types

20=>
Difference between null and undefined ?
Ans=>
1=>Undfined: when a variable is declared but has not been assigned a value, it is automatically initialized with undefined.
2=> undefined can be used when you dont have the value right now but you will get it after some logic or operation

```
let a;
log(a)//undifined
```

1=> null variables are intentionally assigned the null value.
2=> null can be used when you are sure you do not have any value for the particular variable.

```
let a=null;
log(a) //null
```

21=>
what is the use of typeof operator?
Ans=>
1=> typeof operator is used to determine the type of each variable
2=> real application use typeOf operator can be used to validate the data received from external sources (api)

22=>
  what is type coercion in js?
Ans=>
  type coercion is the automatic conversion of values from one data type to another during certain operation or comparisons.
  Use of coercion=>
    1=>
      type coercion can be used during string and number concatenation
    2=>
      type coercion can be used while using comparison operators.

```
let string="42"
let number=42
let boolean=true;
let nullValue=null;

type coercion -automatic conversion
log(string+number) //4242
log(number+boolean) //43
log(nuber == string) true
log(boolean == 1) //true
log(boolean + nullVakue) //1
```

23=>
  what is the differnce between unary binary and ternary operators?
Ans=>
  Unary operators

```
let a=5;
let b=-a;
log(++a);
```

  Binary operator

```
let x=10;
let y=20;
let z=x+y;
```

Ternary Operator

```
let result=condition ? "yes" : "No"
```

24=> what is short circuit evaluation in js?

Ans=>

Short circuit evaluation stops the execution as soon as the result can be determined without evaluating the remaining sub expressions.

Ex:=>

```
    SC with logical AND
    let result1=false && someFunction();
    log(result1) //false

    let result2=true || someFunction();
    log(result2) //true
```

25=> what is copping of object?

shallow copy=>

shallow copy deplicates the top level structure of an object but it does not created copies of nested object

1=>using spread operator.

```
let originalObj={
  key1:"value1",
  key2:"value2",
  key3:"value3",
}
const shallow1={...originalObj}
```

2=>using Object.assign()

```
const shallow2=Object.assign({},originalObj)
```

deepCopy using json.parse() and json.stringify()

json.stringify() converts an object to a json string and json.parse() converts a json string back to an object effectively creating a deep copy

```
        let originObj={
            key1:"value 1",
            key2:"value 2",

          nested:  {key3:"value 3"},
        }

const deepC=JSON.parse(JSON.stringify(originObj));
```

26=>
   what is immutability concept
   immutability involves not modifying the original object, instead creating and returning a new object with the desired changes

```
    const original={
        key1:"value"
    }

// create new object with modification
const modificationObj={...original,key2:"value2"}
console.log(original,modificationObj);
```

27=>
   what is object destructuring ?
   oject dest allows you to extract values from objects and assign them to variables

```
let person={
    name:"shubh",
    age:23,
    country:"india"
}
let {name,age}=person;
console.log(name);
console.log(age);
```

28=>
how can we check the property of object?

```
let person={
 key:"value"
}
console.log(person.hasOwnProperty('key')) //true
```

29=>
what is method is javascript?
we can also defined function as value to properties of object these will be called methods. methods are just function but it means they have been called in 'reference' on an object

```
let person={
   name:"shubh",
   sayHii:function(){
      console.log("say helo")
   }
}
```

29=>
what is this keyword in javascript?
the this keyword refers to the current execution context. it's value depends on how a function is called

1=> in a global context, this refers to the global object(eg window)
2=> in a method of object this refers to the that Object
3=> in a function invoked with the new keyword (constructor function) this refers to the newly created instance
4 in event handler, this often refers to the element that triggered the event
5 in a function invoked using the call or apply method this is explicitly set to the first orgument passed

30=>
what is Map keyword in javascript?
the map object is a collection of key-values pairs where key and value can be any data type. it provides an easy way to iterate over the keys and values and it maintains the order of insertion

map keyword

```
let map=new Map();
let person={name:'shubh'};
let personAccount={balance:5000};
```

set properties in Map

```
map.set('1','str1');
map.set(1,'num1');
map.set(true,'bool1');
map.set(person,personAccount)
```

get value

```
 console.log(map.get(1));
console.log(map.get('1'));
console.log(map.get(true));
console.log(map.get(person));
```

size

```
console.log(map.size);
```

iterable of keys

```
console.log(map.keys());
```

iterable of values

```
console.log(map.values());
```

iterable og key value pair

```
console.log(map.entries());
```

key exist

```
console.log(map.has(1));
console.log(map.has(14));
```

31=>
what is operator precedance?
Ans=>
As per operator precendence operators, with high precedence are evaluatted first.

32=> when to use which type of condition statement in real application?
Ans=>
if...elese for complex different and multiline execution.
Benifits: cover all scenarios.

Ternary Operator:=>
for simple condition & single value evaluation.
Benefit: short one line syntax

Switch stetement:
for same left side values.
Benefit: more structured code

33=>
what is difference between == and ===?
Ans=>
==(loose equality) operator compares two values for equality after performing type coercion
=== strict equality operator compares two values for equality without performing type coercion.

(Normally === is perferred in use to get more accurate comparisons)

34=>
what is difference between spread and rest operator in js?
Spread Operator=>

the spread operator(...) is used to expand or spread elements from an iterable (such as an array string or object) into individual elments.
use of S.O.=>
Copying an array   Merging array   passing multiple argument to a function

ex=>

```
    const array=[1,2,3];
    console.log(...array) //1,2,3

    coping of object     let originArray=[1,2,3];
                         let coppiesArray=[...originArray]

    merging array
                        let array1=[1,2,3];
                        let array2=[4,5,6];
                        let mergeArray=[...array1,...array2]

    passing multiple arguments to a function
                       const number=[1,3,4,5];
                       sum(...number);
                       function sum(a,b,c,d){
                        console.log(a+b+c+d)
                       }
```

Rest Operator=> the rest operator is used in function paramenters to collect all remaining arguments into in array.

```
display(1,2,3,4,5,6);
function display(first,second,...restArguments){
   console.log(second)
   console.log(first)
   console.log(remaining)
}
```

35=>

what is the indexof() method of an array?
indexof() method gets the index of a specified elements in the array.

```
let array=[1,4,6,7,8,5,3];
let a=array.indexOf(4)
console.log(a) //1
```

36=>
what is difference between find() and filter() method of an array?
both method for getting elements (filter(),find(),slice())
1=>find() method get the first element that satiesfies a condition

```
let array=[1,2,3,4,5,6,7,8];
let c=array.find((num)=>{
  return num%2===0
})
console.log(c) //2
```

2=> filter() method get array of elements that satisfies a condition

```
let array=[1,2,3,4,5,6,7,8];
let c=array.filter((num)=>{
  return num%2===0
})
console.log(c) //2,4,6,8
```

36=>
what is the slice() method of an array?
slice() method get a subset of the array from start index to end index(end not included)

```
let letters=["a","b","c","d","e","f"];
let e=array.slice(1,4);
console.log(e) //['b','c','d']
```

37=>
what is the difference between push() and concat() method of an array?

array method for adding elements(push(),concat())
push() will modify the original array itself.

```
let array1=[1,2];
array1.push(3,4);
console.log(array1) //[1,2,3,4]
```

concat method will create the new array and not modify the original array

```
let a1=[1,3];
let a2=a1.concat([5,6])
```

38=> what is difference pop() and shift()

array methods for removing elements(pop(),shift())
pop() will remove the last elements of the array.

```
let arr=[1,3,4,5];
let popped=arr.pop();
log(popped) //5
log(arr) //[1,3,4]
```

shift() will remove the first elements of the array.

```
let arr=[1,3,4,5];
let shifted=arr.shift();
log(shifted) //1
log(arr) //[3,4,5]
```

39=> what is the splice() method of an array?
the splice method is used to add remove or replace elements in an array
syntax array.splice(start Index,deletecount,...items to add)

40=> what is difference between the slice() and splice() method of an array?

slice() method is used to get a subset of array from start index to end index(end not included)

splice() method is used to add remove and replace elements in an array

41=>
what is diff between map() and foreach()

array methods for modification and iteration

map() method is used when you want to modify each element of an array and create a new array with modified value.

foreach() method is used when you want to perform some operation on each element of an array without creating a new array

42=>
what is sort and reverse method?
array can be sor and reverese by using sort() and reverse() methods of an array.

43=>
what is array destructuring in js?
array destructuring allows you extract elements from an array and assign them to individual variable in a single statement

AD introduced in ecmascript 6 (es60)

```
let fruits=["apple","banana","orange"];
const [fr1,fr2,fr3]=fruits
```

43=>
what are array like object in javascript?
array like object are object that have indexed elements and a length property similar to arrays, but they may not have all the methods of arrays like push() and pop() and others
ex=> string,arguments,html collection

argument

```
sum(1,2,4);
function sum(){
   console.log(arguments)
   console.log(arguments.length)
   console.log(arguments[0])
   }

   string
   let string="hello";
   console.log(string.length)
   console.log(string[0])

  accessing html collection
  let boxes=d.GebCN('box')
  console.log(boxes[0])
```

44=>

how to convert array like to array?

method to convert an array like object into array

Array.from()   spread syntax   array.prototype.slice.call()

example array-like object

```
var arraylike={0:'a',1:'b',2:'c',length:3}
```

using array.from()

```
let array1=Array.from(arraylike);
```

using spread operator

```
let array2=[...arraylike]
```

```
let array3=array.prototype.slice.call()
```

45=>

what is difference between while and for loop?

for loop allows to iterate a block of code a specific number of time.
for loop is better for condition with initialization and with incriment because all can be set in just one line of code.

while loop executes a block of code while certain condition is met
while loop is better when there is only condition no initialization no incriment

46 what is difference between while and do while loop?

while loop=>
while loop execute a block of code while certain codition is true

```
  while (condition) {
  // Code to be executed as long as the condition is true
}
```

do while loop
the do while loop is similar to the while loop except that the block of code is executed at least one time even if the condition is false

```
      do {
         statement
} while (condition);
```

48=>
what is difference between break and continue stement in javascript

The break statement is used to terminate the execution of a loop prematurely

```
for (let i = 0; i < 5; i++) {
  if (i === 3) {
    break; // exit the loop when i is 3
  }
  console.log(i);
}
// Output: 0, 1, 2
```

the continue statement is used to skip the current iteration of the loop and move on to the next iteration

```
for (let i = 0; i < 5; i++) {
  if (i === 2) {
    continue; // skip the rest of the loop for i equal to 2
  }
  console.log(i);
}
// Output: 0, 1, 3, 4
```

49=>

what is difference between for and for of loop

for loop is slightly more complex having more lines of code whereas for of much simpler and better for itteration arrays.

```
   const array = [1, 2, 3, 4, 5];

for (const value of array) {
  console.log(value);
}
// Output: 1, 2, 3, 4, 5
```

50=>

what is difference between for of and for in loop?

for...of

for of loop is used to loop through the values of an object like array and string

it allows you to access each value directly without having to use an index

for...in

for..in loop is used to loop through the properties of an object.

it allows you to iterate over the keys of an object and access the values associated by using keys as the index

```
const obj = { a: 1, b: 2, c: 3 };
for (const key in obj) {
  console.log(key, obj[key]);
}
// Output: a 1, b 2, c 3
```

51=>
    when to use for...of loop and when to use foreach method in application?

for of loop is suitabl when you need more control over the loop. such as using break and continue statement inside

foreach method iterate over each element of the array and perform some action on each elements

52=>
    what is difference between named and anonymous function?

    *named function have a name identifier
    *use named function for big and complex logics.
    *use when you want to reuse one function at multiple places?

```
function multiply(x, y) {
  return x * y;
}

console.log(multiply(5, 3)); // Output: 15
```

*anonymous function do not have name idectifier and cannot be refferenced directly by name
*use anonymous function for small logic
*use when want to use a function in single use

```
console.log(function(a,b){
   return a*b
}(4,5))
```

53=>
  what is function expression in js?

a function expressing is a way to define a function by assigning it to a variable.

```
  const add=function(a,b){
   return a+b;
  }
console.log(add(5,3))
```

54=>
  what is callback function . what is it use

"A callback function in JavaScript is a function that is passed as an argument to another function "and is executed after the completion of a specific task. Callback functions are commonly used in asynchronous operations, event handling, and to manage the flow of control in certain situations.

```
function add(x,y){
   return x+y
}
function multiply(x,y){
   return x*y
}

function display(x,y,operation){
   var result=opration(x,y);
   console.log(result)
}
display(2,3,add)
display(2,3,multiply)


// Example 1: Asynchronous Operation
function fetchData(url, callback) {
  // Simulating an asynchronous operation (e.g., AJAX request)
  setTimeout(() => {
    const data = "This is the fetched data";
    callback(data);
  }, 2000);
}

fetchData("https://example.com/api/data", function(result) {
  console.log(result);
});
// Output (after 2 seconds): "This is the fetched data"

// Example 2: Event Handling
document.getElementById("myButton").addEventListener("click", function() {
  console.log("Button clicked!");
});
```

55=>
what is higher order function in js

A higher-order function either takes one or more functions as arguments or returns a function as its result.

```
// Higher-order function taking a function as an argument
function multiplyBy(factor) {
  return function (number) {
    return number * factor;
  };
}
```

```
const double = multiplyBy(2);
console.log(double(5)); // Output: 1
```

56=>
   what is difference between arguments and parameters?

parameters are the placeholders defined in the function declaration

```
function add(a,b){

}
```

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

```
add(4,3)
```

57=>
     what is higher order function?

a hof is take one or more function as arguments(cb function) which return a function as a result.

```
function hof(func){
   func()
}
hof(sayHello);
function sayhello(){
   console.log("hello")
}

function createAddar(number){
   return function(value){
      return value+number
   }
}
```

```
let addFive=createAddar(5);
console.log(addFive(2))
```

58=>
   how many ways can you pass arguments to a function?

   *positional arguments

```
functio add(a,b){
   console.log(a+b)
}
add(2,3)
```

*Named Arguments

```
  var person={
   name:"shubh"
  }

funcation greet(person){
console.log(person.name)
}
greet(person)
```

*Argument Object

```
sum(1,2,3)
function sum(){
   console.log(arguments[0])
   console.log(arguments[1])
   console.log(arguments[2])
   console.log(arguments.length)
}
```

59=>
  how do use default parameter in a function

in js default parameters allow you to specify default values for functional parameters.

```
   function greet(name="amit"){
      console.log(name)
   }
   greet()
```

60=>
  what is the use of event handling in js?
  *event handling is the process of responding to user action in a web page.
  *the addEventListener method of js allows to attach an event name and with the function you want to perform on that event
  Ex=> click mouseover   keydown change load

61=>
  what is first class function in js?
  a programming language is said to have first class function if function in that language are treated like other variable
1=> assignment

```
const myFunction=function(){
   console.log("hello")
}
```

2-> passable as argument

```
function double(number){
   console.log(number*2)
}

function performOperation(double,value){
   return double(value)
}
console.log(performOperation(double,4))
```

3->

```
   returnable as values
   function simpleFun(){
      console.log("this is value")
   }
   const simpleVar=simpleFun();
   simpleVar()
```

62=>

what are pure and impure function in js?

pure function
*a pure function is a function that always produces the same output for the same input.
*pure function cannot modify the state
*pure function cannot have sideEffect

```
   function add(a,b){
      return a+b
   }
   add(2,4)
```

impure function
*a impure function is a function can produces the different output for the same input.
*impure function modify the state
*impure function have sideEffect

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

63=>
   what is function currying in js?

curring in js transforms a functio with multiple arguments into a nested series of function each taking single argumenrs.

advantage=>
    reusability , modularity and specialization big complex functions with multiple arguments can be broken down into small reusable functions with fewer arguments.

```
// Non-curried function
function add(x, y, z) {
  return x + y + z;
}

// Curried version
function curryAdd(x) {
  return function(y) {
    return function(z) {
      return x + y + z;
    };
  };
}

// Usage of the curried function
const curriedAdd = curryAdd(1)(2)(3);
console.log(curriedAdd); // Output: 6
```

64=>
  "What are 'call,' 'apply,' and 'bind' methods in JavaScript?"

Certainly! In JavaScript, call, apply, and bind are methods used to manipulate the this context and passed to a function

The call method is used to invoke a function with a specified this value and arguments provided individually.

```
function sayHello(message) {
  console.log(`${message}  ${this.name}`);
}

const person = { name: "John" };

sayHello.call(person,"hello"); // Outputs: Hello, John
```

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

```
function sayHello(greeting) {
  console.log(greeting + ", " + this.name);
}

const person = { name: "John" };

sayHello.apply(person, ["Hola"]); // Outputs: Hola, John
```

The bind method creates a new function that, when called, has its this keyword set to the provided value and initial arguments fixed

```
function sayHello() {
  console.log("Hello, " + this.name);
}

const person = { name: "John" };
const greetJohn = sayHello.bind(person);

greetJohn(); // Outputs: Hello, John
```

65=>
what is string?
a string is a primitive data type that represents a sequence of characters. Strings are used to store and manipulate textual data.

```
let singleQuotedString = 'Hello, World!';
let doubleQuotedString = "Hello, World!";
```

66=>
what are template literals and string interpolation in string

a templete literal also known as templete string is a featuree introduced in es6(2015) for string interpolation and multiline string in js.

Template literals provide a cleaner and more readable syntax for string concatenation.

```
let name = "John";
let age = 30;

// Using template literals
let greeting = `Hello, my name is ${name} and I am ${age} years old.`;

console.log(greeting);
// Output: Hello, my name is John and I am 30 years old.
```

In JavaScript, string interpolation can be achieved using concatenation (+ operator) or template literals.

```
let firstName = "Alice";
let lastName = "Smith";

// Using concatenation for string interpolation
let fullNameConcatenation = "Full Name: " + firstName + " " + lastName;

// Using template literals for string interpolation
let fullNameTemplateLiteral = `Full Name: ${firstName} ${lastName}`;
```

```
console.log(fullNameConcatenation);
console.log(fullNameTemplateLiteral);

// Both outputs should be the same:
// Full Name: Alice Smith
```

67=>
difference between single quotes and double quotes and backticks in js

Single Quotes ('):

Strings enclosed in single quotes are a way to represent text in JavaScript. Example: let text = 'Hello, World!';

Double Quotes ("):

Double quotes are another way to define strings in JavaScript, and they work similarly to single quotes. Example: let text = "Hello, World!";
Backticks (``):

Backticks, introduced with ES6, are used for template literals, a more powerful way to create strings. They allow multiline text and easy variable embedding. Example: let text = `Hello, ${name}!`;

68=>
what is string immutability?
string in js are considered immutable because you cannot modify the content of an existing string directly.

69=>
how many ways you can concatenate string

+operator
cancat() method
templete literls
join()method

70=>
  how do you select modify create and remove dom elements?

dom method
1=>Selecting dom elements=>
 getElementById......

2=>modifying elements properties and attributes
textContent,innerHTML,setAttribute(name,value)
removeAttribute(name),style.property

3 creating and appending elements
createElements(tag)
appendChild(node)
cloneNode(deep)

4=> removing elements
  remove()
  removeChild(none)

5=> adding and removing event listener

addEventListerer(type,listener)
removeEventListerer(type,listener)

71=>
  what is difference between querrySelector() and querrySelectorAll()

querySelector() returns the first matching element.
querySelectorAll() returns a NodeList containing all matching elements.

72=>
what are the method to modify elements properties and attributes?

textContent,innerHTML,setAttribute(),removeAttribute(),style.setProperty
classlist.add

73=>
difference between innerHTML and textContent?
text content property used to assign plain text to elements

innerHTML property the browser interprets the content as html and renders it accordingly

74=>
how to add and remove properties properties of html elements in dom using js?

setAttribute method is used to add property
```
const myElement = document.getElementById('exampleElement');
myElement.setAttribute('newAttribute', 'New Value');
```

removeAtt method is used to remove property
```
const myElement = document.getElementById('exampleElement');
myElement.removeAttribute('oldAttribute');
```

75=>
how to remove and add style using dom

Adding Styles:

Modify the style property directly or use setProperty to add or modify styles.
Example: element.style.color = 'red'; or element.style.setProperty('color', 'blue');

Use removeProperty to remove a specific style.
Set a style property to an empty string to effectively remove it.
Example: element.style.removeProperty('color'); or element.style.color = '';

add new class to elements
e.cL.add
e.cL.remove
e.cL.toggle

76=>
how to create new element in dom using js? what is difference between createElement() and cloneNode()?

createElement() create new Elements.
```
var newDiv=document.createElement('div')
newDiv.textContent="newly created elements";
document.body.appendChild(newDiv)
```

cloneNode copy the exicting element with all attributes.
```
<div id="originalElement" class="original">Original Element</div>

  <script>
    // Get the original element
    var originalElement = document.getElementById('originalElement');

    // Clone the element without its descendants
    var clonedElement = originalElement.cloneNode(false);
    clonedElement.textContent = 'Cloned Element';
    // Modify the class of the cloned element
    clonedElement.classList.add('cloned');

    // Append the cloned element to the body
    document.body.appendChild(clonedElement);
  </script>
```

77=>
    what is difference between createElement() and createTextNode()

    createElement() create new Elements.

```
var newDiv=document.createElement('div')
newDiv.textContent="newly created elements";
document.body.appendChild(newDiv)
```

    createTextNode() new text node with the content

```
 <div id="myDiv"></div>

  <script>
    // Get the div element
    var myDiv = document.getElementById('myDiv');

    // Create a text node with content
    var textContent = 'This is a new text node.';
    var textNode = document.createTextNode(textContent);

    // Append the text node to the div element
    myDiv.appendChild(textNode);
  </script>
```

78=>
   what is the role of finally block in js

finally block is used to execute some code irrespective of error
it contains code that is guaranteed to be executed

```
try {
  // Code that may throw an exception
} catch (error) {
  // Code to handle the exception
} finally {
  // Code that will always be executed
}
```

79=>
   what is the purpose of the throw statement in js?

```
function userData(){
   try{
      validateUserAge(25)
      validateUserAge("invalid") //this will throw
      validateUserAge(29)
   }catch(error){
      cosole.error("error",error.message)
   }
}

function validateUserAge(age){
   if(typeof age !== "number"){
      throw new Error("age must be a number")
   }
   console.log("user age is valid")
}
```

the throw statements stops the execution of the current function and passed the error to catch block of calling function.

80=>
 what is the error propogation in js?

error propogation refers to the process of passing or propogating an error from one part of code to another by using the throw statement with try catch

81=>
 what are the best practices for error handling?
 1:=use try catch and handle errors appropriately

 2:= use descriptive error messages
 throw new Error("")

3:=> avoid swallowing errors

4:= log error

Wrap in Try...Catch: Put risky code in a try...catch block to catch errors and prevent crashes.

Specify Error Types: Identify the type of error in catch blocks for precise handling.

Custom Error Messages: Create custom error messages for clarity and easier debugging.

Log Errors: Always log errors to the console for troubleshooting.

81=>
 what are the different types of error in js?

SyntaxError:

Occurs when there's a mistake in the syntax of your code.
javascript
Copy code
let x = 10;

if (x > 5 // Missing closing parenthesis

ReferenceError:

Arises when trying to access an undeclared variable or function.
javascript
Copy code
console.log(y); // 'y' is not defined

Happens when an operation is performed on an inappropriate data type.
javascript
Copy code
let num = 'Hello';
num.toFixed(2); // num is not a number, TypeError

RangeError:

Occurs when a numeric value is not within the expected range.
javascript
Copy code
let arr = new Array(-1); // Invalid array length, RangeError

EvalError:

Not commonly encountered; historically used for errors in the eval() function (now deprecated).
javascript
Copy code
eval('alert("Hello")'); // Not recommended due to security risks

82=>
  in how many ways we can create an object?

Object Literal:

The simplest way to create an object is by using an object literal notation.
javascript
Copy code
```
let person = {
  name: 'John',
  age: 30,
  gender: 'male'
};
```

Object Constructor:

You can create an object using the Object constructor.
javascript
Copy code
```
let person = new Object();
person.name = 'John';
person.age = 30;
person.gender = 'male';
```

Object.create():

The Object.create() method creates a new object with the specified prototype object.
javascript
Copy code
```
let personPrototype = {
  name: '',
  age: 0,
  gender: ''
};
```

```
let person = Object.create(personPrototype);
person.name = 'John';
person.age = 30;
person.gender = 'male';
```

Function Constructor:

You can use a constructor function to create objects with shared methods and properties.
javascript
Copy code

```
function Person(name, age, gender) {
  this.name = name;
  this.age = age;
  this.gender = gender;
}

let person = new Person('John', 30, 'male');
```

83=>
  what is difference between array and object?

array
*array are collection of values
*array are denoted be square braces
*element in array are ordered

Object
*object are collection of key values pairs.
*object are denoted by curly braces
*properties in object are unordered

84=>

how to add or modify or delete properties of an object?

```
let person={}
```

adding

```
person.name="shubh"
```

modifying

```
person.name="shubh ji"
```

delete

```
delete person.name
```

85=> explain the difference between dot notation and braces notation?
both dot notation and bracket notation are used to access properties of method of an object

*dot notation is more popular and used due to its simplicity.

*limitation of dot notation in some scenarios brackets notation is the only option such as when sccessing properties when the property name is stored in a variable

```
let person = {
  'first name': 'John',
  'last name': 'Doe'
};

console.log(person['first name']);
```

85=>
what are some common method to iterate over the properties of an object?

For...In Loop:

```
let person = {
  name: 'John',
  age: 30,
  gender: 'male'
};

for (let key in person) {
  console.log(key, person[key]);
}
```

2. Object.keys():

```
Object.keys(person).forEach(key => {
  console.log(key, person[key]);
});
```

Object.values():

```
Object.values(person).forEach(value => {
  console.log(value);
});
```

86=>
   how to check if a property exists in an object?

Using the "in" Operator:

```
let person = {
  name: 'John',
  age: 30
};

if ('name' in person) {
  console.log('The property "name" exists in the object.');
}
```

using hasownproperty

```
if (person.hasOwnProperty('name')) {
  console.log('The property "name" exists in the object.');
}
```

Using "!== undefined" Directly:

```
if (person.name !== undefined) {
  console.log('The property "name" exists in the object.');
}
```

87=>what is set object in js?
the set object is a collection of unique values meaning that duplicates values are not allowed

set provides method for adding deleting and checking the existence of values in the set

set can be used to remove duplicates values from arrays

```
// Creating a Set with unique values
let uniqueNumbers = new Set([1, 2, 3, 2, 1]);

// Adding elements to the Set
uniqueNumbers.add(4);
uniqueNumbers.add(5);

// Checking for existence
console.log(uniqueNumbers.has(3)); // true
console.log(uniqueNumbers.has(6)); // false
```

```
// Iterating through the Set
uniqueNumbers.forEach(number => {
  console.log(number);
});
// Output: 1, 2, 3, 4, 5

// Size of the Set
console.log(uniqueNumbers.size); // 5

// Deleting an element
uniqueNumbers.delete(2);
console.log(uniqueNumbers);
// Output: Set { 1, 3, 4, 5 }

// Clearing all elements
uniqueNumbers.clear();
console.log(uniqueNumbers);
// Output: Set {}
```

88=>
what is the difference between map and object in js?

map
*Any data type (including objects and functions)
*Maintains insertion order of keys

Object
*Strings or symbols
*No guaranteed order; depends on the engine

89=>
what are events in js? how are event triggered?

event are action that happen in the browser such as a button click mouse movement or keyboad input

```
const myButton = document.getElementById('exampleButton');

myButton.addEventListener('click', function() {
  console.log('Button clicked!');
});
```

90=>
   what are the types of events in js?
click,museover,keydown,keyup,submit,focus,blur,change,load,resize

91=>
   what is event object in js?

In JavaScript, the event object is a built-in object that holds information about events when they occur. It's automatically passed to an event handler function.

```
// Adding a click event listener to a button
const myButton = document.getElementById('exampleButton');

myButton.addEventListener('click', function(event) {
  // Accessing properties of the event object
  console.log('Event type:', event.type);           // Click
  console.log('Target element:', event.target);      // The button that was clicked
  console.log('Mouse coordinates:', event.clientX, event.clientY); // Mouse position
  event.preventDefault();  // Preventing the default behavior of the click event
});
```

92=>
   what is delegation in js?

Event delegation in JavaScript is a technique where a single event listener is placed on a parent element to manage events for its child elements.

```
<ul id="myList">
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>
```

Without delegation, you might attach a click event listener to each list item individually:

javascript
Copy code
```
const items = document.querySelectorAll('#myList li');

items.forEach(item => {
  item.addEventListener('click', function() {
    console.log(`Clicked on ${item.textContent}`);
  });
});
```

```
const myList = document.getElementById('myList');

myList.addEventListener('click', function(event) {
  if (event.target.tagName === 'LI') {
    console.log(`Clicked on ${event.target.textContent}`);
  }
});
```

93=>
   what is event bubbling in js?
event bubbling is the process in js where an event triggered up to the dom tree triggering event handlers on its parent elements

```
<div id="outer">
```

```
  <p id="middle">
    <span id="inner">Click me!</span>
  </p>
</div>

<script>
document.getElementById('outer').addEventListener('click', function(event) {
  console.log('Outer div clicked!');
  console.log('Event target:', event.target.id);
  console.log('Event phase:', event.eventPhase);
});

document.getElementById('middle').addEventListener('click', function(event) {
  console.log('Middle p clicked!');
  console.log('Event target:', event.target.id);
  console.log('Event phase:', event.eventPhase);
});

document.getElementById('inner').addEventListener('click', function(event) {
  console.log('Inner span clicked!');
  console.log('Event target:', event.target.id);
  console.log('Event phase:', event.eventPhase);
});
</script>
```

94=>
 how can you stop event propogation or event bubbling in js?

event bubbling can be stopped by calling stopPropagation() method on event

```
<div id="outer">
  <button id="myButton">Click me!</button>
</div>
```

```
<script>
document.getElementById('outer').addEventListener('click', function() {
  console.log('Outer div clicked!');
});

document.getElementById('myButton').addEventListener('click', function(event) {
  console.log('Button clicked!');
  event.stopPropagation(); // Stops the event from propagating further
});
</script>
```

95=>
   what is event capturing in js?
 event capturing is process in js where an event is handled starting from the highest level ancestor(the root of the dom tree) moving down to the target elements.

```
 <div id="outer">
  <p id="middle">
    <span id="inner">Click me!</span>
  </p>
</div>

<script>
document.getElementById('outer').addEventListener('click', function() {
  console.log('Capturing Phase - Outer div clicked!');
}, true);

document.getElementById('middle').addEventListener('click', function() {
  console.log('Capturing Phase - Middle p clicked!');
}, true);

document.getElementById('inner').addEventListener('click', function() {
  console.log('Capturing Phase - Inner span clicked!');
}, true);
</script>
```

96=>
  what is the purpose of the event.preventDefault() method in js?

the event.preventDefault() method is used to prevent the default behavior of an event and link will be prevented

```
<form id="myForm">
  <label for="username">Username:</label>
  <input type="text" id="username" name="username">
  <br>
  <input type="submit" value="Submit">
</form>

<script>
document.getElementById('myForm').addEventListener('submit', function(event) {
  // Prevent the default form submission behavior
  event.preventDefault();

  // Custom handling of form submission
  const username = document.getElementById('username').value;
  console.log('Form submitted with username:', username);
  // Additional processing or sending data to the server can be done here
});
</script>
```

97=>
   what is the use of this keyword in the context of event handling in js?

this keyword refers to the elements that the event handler is attached to

```
   <button class="myButton">Button 1</button>
<button class="myButton">Button 2</button>
<button class="myButton">Button 3</button>

<script>
// Select all elements with class 'myButton'
const buttons = document.querySelectorAll('.myButton');

// Add a click event listener to each button
buttons.forEach(function(button) {
  button.addEventListener('click', function() {
    // 'this' refers to the button that was clicked
    console.log('Button clicked:', this.textContent);

    // Change the text content of the clicked button
    this.textContent = 'Clicked!';
  });
});
</script>
```

98=>
  how to remove an event handler from an element in js?

removeEventListener() method is used to remove event handler from elements

```
<button id="myButton">Click me!</button>

<script>
// Function to be executed when the button is clicked
function buttonClickHandler() {
  console.log('Button clicked!');

  // Remove the click event handler after the first click
  document.getElementById('myButton').removeEventListener('click', buttonClickHandler);
}

// Attach the click event handler to the button
document.getElementById('myButton').addEventListener('click', buttonClickHandler);
</script>
```