Javascript

>EVENT DELEGATION
.instead of attaching an event listener to each child element, you attach a single event listener to a parent element.

Event propagation :Event propagation in JavaScript refers to the process by which events bubble up and/or trickle down through the DOM tree .There are two main phases of event propagation: event capturing and event bubbling

Event Capturing: This is the phase where the event starts from the top of the DOM tree and travels down to the target element.

Event bubbling :When an event occurs on a child element, it bubbles up through its ancestors until it reaches the parent element that has the event listener attached.

Object destructuring: is a feature in JavaScript that allows you to unpack values from objects and assign them to variables more concisely.

Proxy Object
A Proxy object allows you to create a handler for operations performed on another object, known as the target object

---

lastIndexOf : The lastIndexOf method in JavaScript is used to find the last occurrence of a specified value in an array or string.
.If the element is not there it return -1 by default.

includes :The includes() method in JavaScript is used to determine whether a given value is present in an array or a string.
.It returns a Boolean value: true if the value is found, and false otherwise.

this: refers to the executing context of an object

.The Temporal Dead Zone (TDZ) is a concept in JavaScript that refers to the time period between when a variable is declared using let or const and when it is initialized.
.During this time, the variable cannot be accessed or used.
.Attempting to access it will result in a Reference Error.

>POLYFILL
.A polyfill is a piece of code (usually JavaScript) used to provide modern functionality on older browsers or environments that do not support it natively.
.Polyfills are especially useful for ensuring that your web applications work consistently across different

browser versions
.Polyfills enable features from newer versions of JavaScript or other web standards to be used in older browsers that lack support for these features.

>DEBOUNCING
.Debouncing is a technique used in programming to limit the rate at which a function is executed.
.Debouncing ensures that a function is executed only after a certain period of inactivity.
>How Debouncing Works
Delay Execution: When the function is triggered, a timer is started.
Clear Timer: If the function is triggered again before the timer expires, the previous timer is cleared, and a new timer is started.
Execute Function: Once the timer expires without being interrupted, the function is executed.

>THROTTLING
.Throttling ensures that a function is executed at most once in a specified time interval, regardless of how many times the event occurs.
.It essentially limits the number of times a function can be called over time.

>CLOSURE
.A closure occurs when a function is defined inside another function, and the inner function has access to the outer function's variables.
.The key idea is that even after the outer function has finished executing, the inner function still has access to the outer function’s scope.

>APPLICATIONS OF CLOSURE
.Data encapsulation and privacy
.Maintaining State
.Callback functions

>DRAWBACKS
.Memory consumption
.Potential for Memory Leaks:

>Currying is a function programming technique where function with multiple argument is transformed into a sequence of function each taking a single argument.

>OPTIONAL CHAINING
.Optional Chaining allows us to access deeply nested properties of an object without having to explicitly check if each level exists.
.This helps avoid errors when trying to access properties of null or undefined.

>The ternary operator is a concise way to perform conditional evaluations.
.It is a shorthand for an if-else statement and can be used to assign values or execute expressions based on a condition.
.It contain a condition and two expression based on the condition

>A callback is a function that is passed as an argument to another function and is executed after the completion of the latter function.
.Callbacks are a fundamental concept for handling asynchronous operations, event handling, and for creating more modular code.

>The call stack is a data structure that is used of managing function calls and execution context. It is a stack with lastin firstout.

How the Call Stack Works:
.Function Execution: When a function is called, a new execution context is created and pushed onto the stack.
.This context contains information about the function's parameters, local variables, and the code to be executed.
.Function Completion: Once the function completes execution, its context is popped off the stack,

>A lexical environment is a data structure that keeps track of the variables and function declarations within a specific scope.
.It includes the environment record and a reference to its outer environment.

>var
Scope: Function-scoped or globally-scoped if declared outside a function. It’s not block-scoped.
Hoisting: Variables declared with var are hoisted to the top of their scope, meaning you can use them before they are declared, but their value will be undefined until the declaration is encountered.
Re-declaration: You can declare the same variable multiple times within the same scope.

>let
Scope: Block-scoped. This means it’s confined to the block, statement, or expression where it is used.
Hoisting: Variables declared with let are hoisted but not initialized. Accessing them before the declaration will result in a ReferenceError.
Re-declaration: You cannot re-declare a variable within the same scope.
.Red-assigning is possible

>const
Scope: Block-scoped, similar to let.
Hoisting: Variables declared with const are hoisted but not initialized.
.Accessing them before the declaration results in a ReferenceError.
Re-declaration: You cannot re-declare a variable with const in the same scope.
Assignment: Variables declared with const must be initialized at the time of declaration, and their values cannot be reassigned.
.However, if the variable holds an object or array, the contents of the object or array can be modified.

>STATE IN PROMISE
.Promises are objects that represent the eventual completion or failure of an asynchronous operation

>PENDING

. This is the initial state of a Promise.

.It means that the asynchronous operation has not yet completed, and the Promise is neither fulfilled nor rejected.

.At this point, it is waiting for the operation to finish.

FULFILLED

.A Promise enters this state when the asynchronous operation completes successfully.

.When a Promise is fulfilled, it has a result value, which can be accessed using the then method.

REJECTED

.A Promise enters this state when the asynchronous operation fails for some reason.

.When a Promise is rejected, it has a reason (error) for the failure, which can be accessed using the catch method.

.JavaScript provides several methods to handle multiple Promises, each with its own unique behavior.

.These methods are useful for managing asynchronous operations efficiently.

PROMISE.ALL

.Promise.all takes an iterable of Promises and returns a single Promise that resolves when all of the input Promises have resolved, or rejects as soon as one of the Promises rejects.

PROMISE.ANY

.Promise.any takes an iterable of Promises and returns a single Promise that resolves as soon as any of the input Promises resolves, or rejects if all of the Promises reject.

PROMISE.RACE

.Promise.race takes an iterable of Promises and returns a single Promise that settles as soon as any of the input Promises settle (resolve or reject).

PROMISE.ALLSETTLED

.Promise.allSettled takes an iterable of Promises and returns a Promise that resolves after all of the given Promises have either resolved or rejected, with an array of objects that each describe the outcome of each Promise.

>EVENT LOOP

.The event loop is a continuous process that monitors the call stack and the task queues.

.It ensures that tasks are executed in the correct order, allowing JavaScript to handle asynchronous operations efficiently while maintaining a single-threaded execution model.

.When the callstack becomes finished it execution context event loop places the next callback to complete its execution.

MACROTASK QUEUE

.Macrotasks are larger, discrete units of work that are scheduled to run after the current executing

context (the main script or current task).
.Common sources of macrotasks include:
.setTimeout and setInterval
.I/O operations
.UI rendering events
.Event listeners

MACROTASK QUEUE
.The macrotask queue is a queue of tasks that the event loop will process after completing the current execution context (such as a script or a function).
.Once the call stack is empty, the event loop checks the macrotask queue and executes the next task in line.

>MICROTASK
.Microtasks are small, discrete tasks that are scheduled to run after the currently executing script but before any other macrotasks are executed.
.Promise callbacks: .then() and .catch()
.queueMicrotask API: A native API to queue microtasks manually.
.MutationObserver callbacks: Used for DOM change notifications.

>MICROTASK QUEUE
.The microtask queue is a queue that stores all microtasks that are scheduled to be run as soon as the current synchronous execution (including the current macrotask) is complete.
.Unlike macrotasks, which might involve I/O operations and other longer-running tasks, microtasks are intended to execute quickly and are given higher priority.
.Microtask has higher priority than macrotask

BOM
.The Browser Object Model (BOM) is a collection of objects provided by the browser that allow JavaScript to interact with the browser window, document, and other browser-specific functionalities.
.The BOM includes objects such as Window, Navigator, Location, History, and Screen.
.The Window object serves as the entry point to the BOM and provides access to the other objects within

>HIGHER ORDER FUNCTIONS
.Higher Order functions are functions that takes one or more functions as arguments.
.Returns a function as a result

>COMMON HIGHER ORDER FUNCTIONS
.Map:The map() method is a higher-order function that takes a callback function as an argument and applies it to each element of an array, returning a new array with the results.
.Filter:Returns an array with the elements that pass a certain condition
.Reduce: Executes a reducer function on each element of the array, resulting in a single output value.

>REPL
.REPL stands for Read Eval Print Loop.
.It's an interactive programming environment that takes single user inputs, executes them, and returns the result to the user.

REST
.REST, which stands for Representational State Transfer, is an architectural style for designing networked applications.

PRINCIPLES OF REST

1.STATELESSNESS
.Each request from a client to a server must contain all the information needed to understand and process the request.
.This means the server does not store any session information about the client.
.The stateless nature allows the server to treat each request independently, making the system more scalable and easier to manage.

2.CLIENT-SERVER ARCHITECHTURE
.REST follows the client-server architecture, where the client and server are independent entities.
.The client is responsible for managing the user interface and user state, while the server manages the data storage and business logic.
.This separation of concerns allows for greater flexibility and scalability.

3.CACHEABILITY
.Responses from the server must define themselves as cacheable or not.
.This means that data can be cached to improve performance by reducing the need for duplicate processing and network requests.
.Proper use of caching can significantly improve the scalability and performance of a RESTful system.

>EVAL
.The eval() is a built-in function that evaluates or executes a string of code as if it were actual JavaScript code.
.This means you can dynamically execute code that you generate or receive as a string.

>SYMBOL
.Symbol is a unique and immutable primitive value that was introduced in (ES6).
.Symbols are often used as property keys in objects to avoid name collisions, ensuring that each property is unique even if they share the same name.

>LABEL
.A label is an identifier followed by a colon (:) that is used to label a statement, such as a loop or a block of code.

.Labels are typically used in conjunction with control flow statements like break or continue to change the flow of execution in a program more flexibly.

>METAPROGRAMMING
.Metaprogramming is a programming technique in which programs have the ability to treat other programs (or themselves) as data.
.This means that a metaprogram can read, generate, analyze, or transform code.
.Essentially, metaprogramming allows you to write code that writes or modifies other code.
.JavaScript supports metaprogramming through features like:
>Proxies: Proxies allow you to intercept and redefine fundamental operations for objects, such as property access, assignment, function invocation, and more.
>Reflect API: The Reflect API provides methods for performing object operations, which can be used alongside proxies to more precisely control object behavior.
>Eval: The eval() function allows you to execute a string of JavaScript code dynamically, although it's generally discouraged due to security risks.

BITWISE OPERATOR
.Bitwise operator is used to perform operations on individual bits of binary numbers.
.AND (&).OR(|) are examples of bitwise operators

>LITERALS
.Literals are fixed values that are written directly into the code.

DEFAULT CONSTRUCTOR
.A default constructor is a constructor that is automatically created by the compiler if no other constructors are explicitly defined in a class.
.The default constructor does not take any arguments.

>FACTORY FUNCTION
.A factory function is a function in JavaScript that creates and returns objects.
.Return Objects: A factory function explicitly returns an object, typically a new instance each time it's called.
.No new Keyword: Unlike constructors (which use new), factory functions do not require the new keyword to create an object.
.Avoid Issues with this: Unlike constructors, factory functions are not dependent on the new keyword and therefore avoid issues related to this binding.

>IIFE
.An IIFE (Immediately Invoked Function Expression) is a JavaScript function that runs as soon as it is defined.
>ADVANTAGES
.Encapsulation of variables

.Avoiding global name pollution
.Modular code organisation
.Immediate Execution
.Closure support

>GENERATOR FUNCTION
.A generator function in JavaScript is a special type of function that can pause its execution and later resume it.
.Unlike regular functions that run from start to finish and return a single value, generator functions can yield multiple values one at a time.
.Yield: The yield keyword is used to pause the generator function's execution and return a value to the caller.
.The function can be resumed later from where it left off.
.Iterator Object: When you call a generator function, it doesn’t run immediately.
.Instead, it returns an iterator object.
.You can then call the next() method on this iterator to execute the generator function until the next yield.
.next() Method: This method resumes the generator function's execution.

WEAKSET
.A WeakSet is a special type of collection that holds weakly referenced objects.
.This means that the objects stored in a WeakSet can be garbage collected if there are no other references to them, which helps in optimizing memory usage.
.A WeakSet can only contain objects.
.Primitive values like numbers, strings, or booleans cannot be added to a WeakSet.
.Cannot contain duplicates
.Can be iterated.

WEAKMAP
.A WeakMap in JavaScript is a special type of collection that allows you to store key-value pairs where the keys are objects, and the values can be of any data type.
.Like WeakSet, the references to the keys in a WeakMap are weak, meaning that if there are no other references to a key object, the object can be garbage collected, which helps in managing memory efficiently.
.Keys Must Be Objects.
.Primitive data types (like strings, numbers, or booleans) cannot be used as keys.
.Cannot be iterated
.Cannot contain duplicates

Prototype pollution
.A type of security vulnerability in JavaScript where an attacker manipulates the prototype of built-in objects (like Object.prototype) to introduce malicious properties or methods.

CONTINUE

.In programming, the continue statement is used within loops to skip the current iteration and immediately proceed to the next iteration of the loop.

.When the continue statement is encountered, the rest of the code inside the loop for the current iteration is ignored, and the loop's control flow jumps back to the beginning for the next iteration.

>JSON

.JSON stands for JavaScript Object Notation.

.It is a way of storing and exchanging data between client and server

>JSON.stringify()

.Converts a JavaScript object or value to a JSON string.

>JSON.parse()

.Parses a JSON string and converts it back into a JavaScript object.

>DEFAULT PARAMETERS

.Default parameters allow you to define a function with parameters that have default values.

.If the function is called without arguments for those parameters, the default values will be used.

>OBJECT METHODS

>OBJECT.ASSIGN()

.Useful for cloning objects or merging multiple objects into one.

>OBJECT.KEYS()

.Return an array of keys of the objects

>OBJECT.VALUES()

.Return an array of object values

>OBJECT.ENTRIES()

.Return all the key-value pair of the Objects as array

>OBJECT.FROMENTRIES()

.Transforms an array of key-value pairs into Object

Object.FREEZE()

.Used to make an object completely immutable.

.This means that once an object is frozen, you cannot:

.Add new properties to the object.

.Remove existing properties from the object.

.Modify existing property values.

Object.SEAL()

.By using Object.seal() we cannot add or delete object properties but we can modify the properties of

objects

>OBJECT.IS()
.Object.is() returns true if the two values are the same, and false otherwise.
.Like ===

>CROSS SITE REQUEST FROGERY
.CSRF is a type of web security vulnerability that allows an attacker to trick a user into performing actions on a web application on behalf of the attacker, without the user's consent.
.These actions are typically made without the user knowing, using their authenticated session with a web application.

>Preventing CSRF Attacks:
.CSRF Tokens
.Samesite Cookies

>PROXY SERVER
.A proxy server acts as an intermediary between a client and the internet.
.It forwards client requests to the destination server and returns the response to the client.
.A Proxy Server is client-facing, helping the client interact with external servers while hiding the client’s identity.

>REVERSE PROXY
.A reverse proxy is a server that sits in front of web servers and forwards client requests to the appropriate backend servers.
.A Reverse Proxy is server-facing, helping servers manage incoming requests and hiding server infrastructure details from the client.

>PROXY_PASS
.The proxy_pass directive is a key feature in NGINX
.proxy_pass is an NGINX directive that specifies the backend server (or servers) to which the incoming client requests should be forwarded

>STACK MEMORY
.Stack memory is used for static memory allocation.
.It stores local variables, function parameters, and return addresses.
.It operates in a Last-In-First-Out (LIFO) manner.
.The most recently added item is the first to be removed.
.The size of the stack is typically limited and determined by the system or programming environment.
.It's usually smaller compared to the heap.
.Memory allocation and deallocation are automatic and managed by the system.
.When a function is called, its local variables are pushed onto the stack, and they are popped off when the function returns.

>HEAP MEMORY
.Heap memory is used for dynamic memory allocation.
.It is used to store data that needs to persist beyond the scope of function calls, such as objects, dynamic arrays, and other large data structures.
.It operates on First-In-First-Out manner like Queue
.The heap is typically much larger than the stack, allowing for the allocation of large amounts of memory.
.Memory allocation and deallocation are manual and controlled by the programmer (or the garbage collector, in languages with automatic memory management).

>CYCLIC REFERENCE
.A cyclic reference occurs when two or more resources reference each other in a way that creates a closed loop.
.This means that at least one element in the chain of references eventually refers back to itself, either directly or indirectly.

>CALLBACK HELL
.Callback Hell (also known as Pyramid of Doom) refers to a situation in programming where callbacks (functions passed as arguments to other functions) are nested within each other, creating a complex and deeply nested structure.
.This often happens in asynchronous programming where multiple asynchronous operations are performed in sequence or in a nested manner.

>ASYNCHRONOUS FUNCTION
.An asynchronous function is a function in programming that performs operations asynchronously, meaning it can execute tasks that take time without blocking the execution of other code.
.Asynchronous functions allow a program to continue running while waiting for these operations to complete.
.An async function is defined using the async keyword before the function declaration.
.Inside this function, you can use await to handle Promise-based operations.

>SETIMMEDIATE
.setImmediate is a function in JavaScript that is used to execute a piece of code asynchronously, but as soon as possible after the current event loop cycle, but before any I/O operations or timers scheduled for the next event loop iteration.
.This makes it more immediate than setTimeout(..., 0), which schedules the function to run after a minimum delay of 0ms.
.In Node.js, process.nextTick() is even faster and executes before setImmediate.
.process.nextTick() runs callbacks at the end of the current operation, while setImmediate runs them on the next event loop.

>EXCEPTIONS
.Exceptions in JavaScript are errors that occur during the execution of a program.

.When an exception is thrown, the normal flow of the program is interrupted, and control is transferred to an exception handler.

>COMMON TYPES OF EXCEPTIONS ARE
.SyntaxError: When the code is syntactically incorrect.
.ReferenceError: When a variable or object is referenced that does not exist.
.TypeError: When an operation is attempted on an object of an incorrect type.
.RangeError: When a number is outside the valid range for a specific operation.
.EvalError: When an error occurs during the evaluation of a string of code.

HANDLING EXCEPTIONS
.JavaScript provides the try...catch...finally block for handling exceptions.
.try: The code block that might throw an exception is placed here.
.catch: This block is executed if an exception is thrown within the try block.
.The exception object is passed as an argument to the catch block.
.finally: This block is always executed, regardless of whether an exception is thrown.

>ERROR OBJECTS
.In JavaScript, when an exception occurs, an error object is created to represent the details of the error.
.This object provides information such as the error message, the type of error...

>FUNCTION BROWSING
.Function browsing refers to the process of exploring and understanding the behavior of functions within JavaScript code.
.It involves examining the function's definition, parameters, return value, and how it interacts with other parts of the program

>FUNCTION DEFINITION
.Name: The identifier used to reference the function.
.Parameters: The variables that the function accepts as input.
.Body: The code block that defines the function's behavior.
.Return Value: The value that the function returns when it completes its execution.

>FUNCTION INVOCATION
.The process of calling a function to execute its code.
.Involves passing arguments to the function, if any, and receiving its return value.

>FUNCTION SCOPE
.The region of code within which a function's variables and parameters are accessible.
.Functions have their own scope, which is separate from the global scope and the scope of other functions.

>FUNCTION HOISTING
.The behavior where function declarations are hoisted to the top of their scope, allowing them to be

used before they are defined.
.Function expressions, however, are not hoisted.

>FUNCTION EXPRESSION
.Functions that are assigned to variables
.Can be anonymous or named.
.Can be used as arguments to other functions or returned from functions.

>UNDEFINED
.Undefined is used to indicate that a variable has not been assigned a value.
.It is the default value of variables that have been declared but not yet initialized.

>NULL
.Null is a special value that represents the intentional absence of any value.
.It is often used to indicate that a variable should be empty or to signify that an object is missing or not available.

>FUNCTION COMPOSITION
. It refers to the process of combining two or more functions to produce a new function.
.The resulting function is a pipeline of the original functions, where the output of one function becomes the input of the next.

>PIPING
.Piping is a concept used in programming and data processing that involves passing the output of one function or process directly as input to another function or process.

>PARTIALS
.Partials often refers to a concept where a function is pre-filled with some arguments, resulting in a new function that is easier to use or more specific.

>JWT
.JSON Web Token (JWT) is a compact, URL-safe token format used for securely transmitting information between parties as a JSON object.
.JWTs are widely used for authentication and authorization in web applications.
.A JWT consists of three main components:
>HEADER
.The header typically consists of two parts: the type of the token (which is always JWT) and the signing algorithm being used, such as HMAC SHA256 or RSA.
>PAYLOAD
.The payload contains the claims.
.Claims are statements about an entity (typically, the user) and additional data.
>SIGNATURE
.The signature ensures that the token wasn’t altered after it was issued.

.The signature is created by taking the encoded header, the encoded payload, a secret, and the algorithm specified in the header, and signing them.

>WORKING FLOW OF JWT
.The process begins when a user logs into a system (for example, by entering a username and password).
.The client (usually a web browser or mobile app) sends these credentials to the server in a login request.
.Server Verifies Credentials
.If the credentials are valid, the server creates a JWT and sends it back to the client as a response.
.Client Stores the JWT in localstorage,sessionstorage/cookies
.Client Sends JWT in Subsequent Requests
.Server Responds to the Client
.If the JWT is valid, the server processes the request and sends the appropriate response
.JWTs often have an expiration time (exp claim).
.After the token expires, the client must re-authenticate (login again) or use a refresh token system to get a new JWT without requiring the user to log in again.

>JWT VERTIFY
.jwt.verify() is a method provided by the jsonwebtoken library in Node.js used for verifying a JSON Web Token (JWT).
.This method allows you to validate the token's authenticity and integrity.

>SPARSE ARRAY
.A sparse array is an array in which most of the elements have a default value, typically zero or null, and only a few elements contain meaningful data.

CONSTRUTOR FUNCTION
.A constructor function in JavaScript is a special type of function used to create and initialize objects.
.When you invoke a constructor function using the new keyword, it creates a new instance of an object with the properties and methods defined in the constructor.

ARROW FUNCTION
.Arrow function is a concise way of writing function
.Arrow functions do not have their own this. Instead, they inherit this from their enclosing scope.
.Cannot be used as a constructor. Trying to use new with an arrow function will throw an error.
.Does not have its own arguments object. You would need to use rest parameters to achieve similar functionality.
.Cannot be used inside object as method due to the lack of this keyword which will throw unexpected errors.

>INSTANCE OF
.the instanceof operator is used to check whether an object is an instance of a specific class or

constructor function.
.It checks the object's prototype chain to determine if the prototype property of the constructor appears anywhere in the object’s chain.

>FUNCTION BORROWING
.Function Borrowing in JavaScript is a technique where one object uses a method (function) belonging to another object.
.This is typically done using methods like call(), apply(), or bind().

>CALL()
.The call() method calls a function with a specified this value and arguments passed individually.

>APPLY()
.Similar to call(), but instead of passing arguments individually, you pass them as an array.

>BIND()
.The bind() method creates a new function with a specified this value
.Unlike call() and apply(), it does not execute the function immediately but instead returns a new function that can be invoked later.

>TYPE CONVERSION
.type conversion refers to changing one data type to another.
.It can happen automatically (implicit type conversion) or manually (explicit type conversion).
1.Implicit Type Conversion (Type Coercion)
.JavaScript automatically converts data types during certain operations.
.This can sometimes lead to unexpected results because JavaScript tries to "guess" the desired type.
2. Explicit Type Conversion (Type Casting)
Explicit type conversion is when you manually convert a value from one type to another using JavaScript functions or operators.

>METHODS
.In JavaScript, methods are functions that are defined as properties of an object.
.They represent actions that objects can perform.
.The key difference between a method and a regular function is the context in which they are called, which is tied to the object that owns the method.

>CONSTRUCTOR
.A constructor is a special type of function used to create and initialize objects.
.Constructors are invoked using the new keyword, and their primary role is to set up the initial state of a new object by assigning values to properties.

>SHADOWING
.Shadowing in programming refers to the situation where a variable in a local scope has the same name as a variable in a global scope

.In this case, the locally scoped variable "shadows" the outer variable, meaning the inner variable takes precedence, and the outer variable becomes inaccessible within that scope.

>ILLEGAL SHADOWING
.Illegal shadowing: Happens when block-scoped (let/const) variables conflict with function-scoped (var) variables in the same scope, or when trying to redeclare variables within the same scope using different keywords.
.Same variables with different keywords

>SHALLOW COPY
.A shallow copy means copying an object's top-level properties.
.However, if the object has nested objects or arrays (i.e., references), the copy only duplicates the reference to those nested objects, not the actual objects themselves.
.As a result, changes made to the nested objects in the copied object will also affect the original object since both the copy and original still share the same reference to those nested objects.

>DEEP COPY
.A deep copy creates an entirely new object, recursively copying all the nested objects and arrays.
.This means that the new object is completely independent of the original object, and changes made to nested objects in the copy will not affect the original, and vice versa.

>FIRST CLASS FUNCTION
.A first-class function is a concept in programming where functions are treated as first-class citizens.
.This means that functions can be treated like any other variable or data type in the language.
.Functions can be assigned to variables.
.Functions can be passed as arguments to other functions.
.Functions can return other functions.

>PROMISE VS ASYNC/AWAIT
.A Promise is an object representing the eventual completion (or failure) of an asynchronous operation.
.Promises provide a way to chain asynchronous tasks using .then() for success and .catch() for error handling.
.A Promise is created using the new Promise() constructor, which takes two parameters: resolve and reject.
.Promises can be chained using .then() for success and .catch() for errors.
.Promises allow non-blocking asynchronous operations, and when one completes, the next .then() block is executed.

.async/await is a more concise and readable syntax built on top of Promises.
.It makes asynchronous code appear synchronous, which can improve readability and simplify error handling.
.async function: Declaring a function as async allows you to use the await keyword inside it.
.await keyword: Pauses the execution of the async function until the Promise is resolved, making it look like synchronous code.

>OBSERVABLES
.Observables are objects that represent a stream of data or events that can be observed (i.e., subscribed to) over time.
.They are central to Reactive Programming, especially in libraries like RxJS (Reactive Extensions for JavaScript).
.Promise resolves with a single value
.Observables can emit multiple values
.Promise starts immediately when created
.Observables starts when we subscribe to it

>POSTINCREMENT(x++)
.The post-increment operator increments the value of a variable by 1, but returns the original value before the increment.

>PREINCREMENT(++x)
.The pre-increment operator increments the value of a variable by 1 and returns the new, incremented value

---

NULLISH OPERATOR
.The nullish operator (??) in JavaScript is used to provide a default value when the value on the left-hand side is either null or undefined.
. It's particularly useful when you want to handle cases where a variable is either null or undefined, but still consider other "falsy" values (like 0, false, '') as valid.

>ES6 FEATURES
.let and const
.Arrow functions
.Destructuring arguments
.Default parameters
.Spread and Rest Operators
.Modules
.classes
.Promises
.Async await
.Iterators and Generators
.Template Literals

>SPREAD OPERATOR
.The spread operator is used to "spread" or expand the elements of an array or object into individual elements.
.It’s useful for copying arrays, concatenating arrays, and passing multiple elements into functions.

>REST OPERATOR
.The rest operator collects multiple elements into an array.
.It is used in function parameters and destructuring assignments to group remaining elements into an array.

>NEW KEYWORD
.In JavaScript, the new keyword is used to create an instance of an object from a constructor function or class.

>DESIGN PATTERNS
.Design patterns are important to help developers structure their code, manage complexity, and promote code reuse.
.Since JavaScript is an object-oriented, functional, and event-driven language, various design patterns can be applied depending on the context of the problem.

>LIBRARY
.library is a collection of pre-written code (functions, objects, classes, etc.) that provides reusable functionality to help solve common programming tasks.
.By using a library, developers can avoid writing everything from scratch and can leverage tried and tested solutions to perform certain operations more efficiently.

>FRAMEWORK
.A framework in JavaScript (or any programming language) is a collection of pre-written code, tools, and conventions that provides a structured way to build applications.
. Unlike a library, which you call to use its functions, a framework often "calls" your code and dictates the structure and flow of the application, offering a predefined architecture to follow.

>ESCAPE SEQUENCE
.An escape sequence is a series of characters in programming and markup languages that represents a special character or action.
\t
\r...

>HTTP
.HTTP (HyperText Transfer Protocol) is the foundation of data communication on the web.
.It is a protocol used by browsers and servers to communicate and exchange information over the internet.
.HTTP defines how messages are formatted and transmitted, as well as how web servers and browsers should respond to various requests.
.It has a request response cycle
.HTTP: Transmits data in plain text, which can be vulnerable to interception.
.HTTPS: A secure version of HTTP, where data is encrypted to provide more security, often seen on banking sites or websites requiring personal data.

>PASS BY VALUE
.The concepts of pass by value and pass by reference describe how arguments are passed to functions and how changes to those arguments affect the original variables.
.When a variable is passed by value, a copy of the variable's value is made.
.Changes made to the parameter within the function do not affect the original variable.

>PASS BY REFERENCE
.When a variable is passed by reference, a reference (or pointer) to the original variable is passed to the function.
.Changes made to the parameter affect the original variable.

>ACCESSING OBJECTS
1.Dot Notation
2.Bracket Notation
3.Optional chaining
4.Object.keys(), Object.values(), Object.entries()
5.Destructuring
6.For in loop

>FOR IN
.Used to iterate over the enumerable properties of an object.
.It accesses the keys (property names) of the object.

>FOR OF
.Used to iterate over iterable objects like arrays, strings, maps, sets, etc.
.It accesses the values of the iterable, not the keys.

>BINARY LITERALS
.Binary literals allow you to express numbers in binary format

>OBJECT LITERALS
.Object literals are used to define objects in JavaScript.
.Object literals provide an easy way to create objects without the need for constructors or methods.

>TEMPLATE LITERALS
.Template literals are a way to create strings in JavaScript, allowing for easier string interpolation (embedding expressions in strings) and multiline strings without special characters.

PAD START AND PAD END
**.The `padStart()` and `padEnd()` methods in JavaScript are used to add padding to the beginning or end of a string until the resulting string reaches the specified length

THENABLE
.A thenable in JavaScript is any object or function that has a `then` method, which behaves similarly to

the `then` method of a native JavaScript `Promise`.
.If an object or function has a `then` method, it can be treated like a promise by the JavaScript runtime.

WEB APIS
.Web APIs (Application Programming Interfaces) are built-in browser interfaces that allow developers to interact with the browser and perform various tasks

1. **DOM (Document Object Model) API
**To manipulate HTML and CSS, allowing dynamic changes to the webpage content and structure.

2. **Fetch API**
.To make network requests to retrieve resources from a server or send data to a server.

3. **Geolocation API**

.To retrieve the geographical location of a user's device.

4. **LocalStorage and SessionStorage API** .To store key-value data in the browser, either permanently (`localStorage`) or temporarily (`sessionStorage`).

5. **History API
**.To manipulate the browser session history stack.

FLATMAP
.FlatMap is a method available on arrays that combines two operations: mapping and flattening.
.It allows you to apply a function to each element of an array and then flatten the result into a new array, effectively reducing the depth of the returned structure.

HTML SANIZATION
.HTML sanitization is the process of cleaning up HTML content to prevent potentially harmful code from being executed, especially when that HTML is generated from user input or third-party sources.
.This is critical in web development to mitigate security vulnerabilities like Cross-Site Scripting (XSS) attacks.
.Use dompurify-A fast and secure library that cleans HTML and prevents XSS attacks.
