JAVASCRIPT

- **JAVASCRIPT**
  Javascript Is a synchronous single-thread language.
  Everything in javascript happens in the execution context, there are two components in the execution context, **memory or variable environment** where the key value is stored and the **code or thread of execution**. This is where the code is executed line by line
  Javascript is used **to create dynamic and interactive web content like applications and browsers.**
    - **Web Applications, Mobile Applications, Games**

- **Execution Context -** Everything in JavaScript happens inside a global execution context.

- **Call Stack -** The Call stack maintains the order of execution context.

- **Undefined Datatypes** - A variable has been declared but not initialized with a value

- **Not defined** - Variable doesn't exist

- **Primitive -** primitive data structure is predefined by the programming language - int, float, char, boolean

- **Non-primitive** - is a data structure that is built up from primitive data structures. - arrays, objects, classes and functions

- **JSON (JavaScript Object Notation)** is a data-interchange format that is easy for humans to read and write, and easy for machines to parse and generate. JSON is often used for representing structured data and is commonly used in web applications to transmit data between a server and a client, as well as for configuration files and data storage.
  - **JSON METHODS**
    - **JSON.stringify()**: This method is used to convert a JavaScript object or value into a JSON string. It takes the object as an argument and returns the JSON representation as a string.
    - **JSON.parse()**: This method is used to parse a JSON string and convert it into a JavaScript object or value. It takes the JSON string as an argument and returns the corresponding JavaScript object.

- **Set** is a built-in data structure that represents a collection of unique values. Unlike arrays, which can contain duplicate values and are ordered, sets do not allow duplicates, and their elements have no particular order. Sets can be useful when you need to maintain a collection of distinct values and perform operations like adding, removing, or checking for the existence of values efficiently.

- **DOM** - Dom is an object-oriented representation of a webpage.
  getElement by id
  getElement by classname
  getElement by name
  getElement by selector

- **CDN vs Download** - CDN is fast because it stores content in different formats

- **Event Listener -** An event listener is a procedure in JavaScript that waits for an event to occur.

- **Event Loop** - **to monitor the Call Stack and the Callback Queue**. If the Call Stack is empty, the Event Loop will take the first event from the queue and will push it to the Call Stack.

- **Event delegation -** The concept of event delegation is based on the fact that events bubble up the DOM tree from the target element to its ancestors. By capturing the event at a higher level, you can handle events for multiple elements with a single event listener. This approach is particularly useful when you have a large number of elements that need event handling or when dynamically adding or removing elements from the DOM.

- **Event propagation** - also known as event bubbling, is a mechanism in JavaScript where events triggered on a nested element will also trigger on all its ancestor elements in the DOM tree unless the event is explicitly stopped from propagating further. This allows for events to be handled at different levels of the DOM hierarchy.

- **Event Capturing** - an event triggers from the outermost element to the target.

- **Event Bubbling** - the event movement begins from the target to the outermost element in the file. It first runs the handlers on it, then on its parent, and then all the way up on other ancestors. To stop event bubbling, you can use the **stopPropagation.**

- **Memory heap** - stores and writes information where memory is allocated, used, and removed. In other words, it is the space where all the variables and functions are assigned to memory.

- **Run time environment** - A runtime environment is where your program will be executed.

- **Variable Environment** - variables created within the scope of the provided function within the lexical environment.

- **Lexical Environmen**t - Lexical environment is the local memory along with reference to the lexical environment of its parent.

- **Shadowing** - when a variable declared in a certain scope has the same name as a variable in an outer scope.

- **Illegal shadowing**

```
inside lexical scope let a=10;
    {
       var a=20;
    }
```

- **Use strict** - The purpose of "use strict" is to indicate that the code should be executed in "strict mode". For example, With strict mode, you cannot use undeclared variables.

- **Types of function** - function declaration, function expression, an anonymous function

- **First Class Function** - A first-class function is a function passed as an argument

- **Pure Function** - Pure functions are functions that give a single output for each input despite any change to the extended environment of the function. A pure function does not depend on any other variable other than the input variable of the function it sets.

- **High Order Function** - a function that takes another function as an argument or returns a function. **Examples** - map, filter, reduce, foreach

- **Arrow function** - It allows you to create functions in a cleaner way compared to regular functions.

- **Generator function -** used to generate a sequence of values, one at a time, as you need them. Unlike regular functions, generator functions can be paused and resumed multiple times, which makes them useful for creating iterators or implementing asynchronous code. You can pause the execution of a generator function without executing the whole function body. For that, we use the **yield** keyword. GENERATOR FUNCTION

- **IIFE** (Immediately Invoked Function Expression) - is a function that runs as soon as it is defined.

- **Factory function** - A factory function is a design pattern in JavaScript where you create and return objects from a function
- **Hoisting** - Accessing Variables and Function Before Initialization.
- **Scope** - Defines the Accessibility of Variables and Functions
- **Block scope**: This scope restricts the variable declared inside a specific block, from access by the outside of the block.
- **Var** - can be accessed from outside the block.
- **Let** - cannot be accessed from outside the block.
- **Const** - cannot be accessed from outside the block
- **Function scope**: Variables defined inside a function are not accessible from outside the function.
- **Global scope**: Anything that is not inside a function is called global space.
- **Scope chain -** The scope chain in JavaScript is the order in which the interpreter looks for variables in different scopes. When a variable is referenced, the interpreter first checks the current local scope, and if it is not found, it continues to check each outer (enclosing) scope until it reaches the global scope. If the variable is still not found, a ReferenceError is thrown. The scope chain determines the accessibility and visibility of variables in the nested scopes.
- **Temporal dead zone** - Temporal dead zone is the area of a block where a variable is inaccessible until the moment it is initialised with a value.
- **Cross-Origin Resource Sharing (CORS)** - is an HTTP-header based mechanism that allows a server to indicate any origins (domain, scheme, or port) other than its own from which a browser should permit loading resources.
- **Reference Error -** The ReferenceError object represents an error when a variable that doesn't exist (or hasn't yet been initialised) in the current scope is referenced.
- **Syntax Error -** An exception caused by the incorrect use of a pre-defined syntax.
- **Type Error** - The TypeError object represents an error when an operation cannot be performed, typically when a value is not of the expected type.
- **Trim** - The trim() method removes whitespace from both sides of a string.
- **Type of** - Used to check and return the data type.

- **Instanceof** () **-** Used to check the type of an object at run time. it returns a Boolean value (true or false). If the returned value is true, then it indicates that the object is an instance of a particular class and if the returned value is false then it is not.

- **Shallow copy vs deep copy** - In Shallow copy, a copy of the original object is stored and only the reference address is finally copied. In Deep copy, the copy of the original object and the repetitive copies both are stored.

- **API (**application programming interface)
    - **Types of API**
        - **DOM API** - DOM API allows to manipulation of HTML and CSS
        - **Fetch API** - The fetch() method starts the process of fetching a resource from a server.

- **Spread operator** - The JavaScript spread operator (...) allows us to quickly copy all or part of an existing array or object into another array or object.

- **Statically typed vs Dynamically typed** - Dynamically typed languages perform type checking at runtime, while statically typed languages perform type checking at compile time.

- **Single thread vs Multi-thread** - **Single-threaded** processes contain the execution of instructions in a single sequence. One command is processed at a time. **Multi Thread** allows the execution of multiple parts of a program at the same time.

- **Synchronous vs Asynchronous** - Synchronous means the code runs in a particular sequence of instructions given in the program, whereas asynchronous code execution allows to execution of the upcoming instructions immediately. Because of asynchronous programming, we can avoid the blocking of tasks due to the previous instructions.

- **Blocking and Non-blocking** - Blocking refers to operations that block further execution until that operation finishes while non-blocking refers to code that doesn't block execution

- **Prototype -** In JavaScript, prototypes are objects that serve as templates for creating instances of similar objects. Every object in JavaScript has a prototype, which is essentially its "parent" object from which it inherits properties and methods. If a property or method is not found on an object, JavaScript will look for it on its prototype.

  The **prototype** is a property of a Function object. It is the prototype of objects constructed by that function. **__proto__** is an internal property of an object, pointing to its prototype

- **Conditional statement** - Conditional statements are used to perform different actions based on different conditions.

- **Implicit type coercion** - the process of automatic or implicit conversion of values from one data type to another.
    - const num = 5;
    - const str = "10";
    - const result = num + str;
    - console.log(result); // Output: "510"

- **External or explicit type casting** - This involves the developer explicitly converting a value from one data type to another. It is typically done using built-in functions or casting operators provided by the programming language. **EX** - **parseInt**, **parseFloat**

- **Template literals -** Template literals provide an easy way to interpolate variables and expressions into strings. The method is called string interpolation.

- **Back tick -** Although backticks are mostly used for HTML or code embedding purposes, they also act similarly to single and double quotes. Besides, using backticks makes it easier for string operations.

- **Types of conversion**
    - Converting Strings to Numbers
    - Converting Numbers to Strings
    - Converting Dates to Numbers
    - Converting Numbers to Dates
    - Converting Booleans to Numbers
    - Converting Numbers to Booleans

- **Memoization** - it is where the results of expensive function calls are cached and returned from the cache instead of recalculating the results each time the function is called.

- **Currying** - the transformation of the function of multiple arguments into several functions of a single argument in sequence.

- **Closure** - A closure is **the combination of a function bundled together (enclosed) with references to its surrounding state (the lexical environment)**.

A closure allows a function to "remember" its surrounding scope even after the outer function has finished executing. This enables the function to access and manipulate variables from its containing scope, even if those variables are no longer in scope.

- **Callback** - A callback is a function passed as an argument to another function.

- **Callback queue** - After the timer expires, the callback function is put inside the Callback Queue

- **Callback hell** - When we nest multiple callbacks within a function it is called callback hell.

- **Mark and Sweep Algorithm** - The mark and sweep algorithm is a garbage collection technique used in computer programming to automatically free up memory that is no longer in use by a program.

- **Copy elision -** Copy elision is to prevent extra copies of objects.

- **Ternary operators** - The conditional (ternary) operator is the only JavaScript operator that takes three operands: a condition followed by a question mark (?), then an expression to
- execute if the condition is truthy followed by a colon (:), and finally the expression to execute if the condition is falsy. This operator is frequently used as an alternative to an if...else statement.

- **Local Storage vs Session Storage -** the data in localStorage doesn't expire, and data in sessionStorage is cleared when the page session ends.

- **Window** - A window is a global object that is created along with the global execution context.

- **Optional chaining** - The optional chaining (?.) operator accesses an object's property or calls a function. If the object accessed or function called using this operator is undefined or null, the expression short circuits and evaluates to undefined instead of throwing an error.

- **The nullish coalescing (??) operator** is a logical operator that returns its right-hand side operand when its left-hand side operand is null or undefined and otherwise returns its left-hand side operand.

- **Object destructuring -** JavaScript object destructuring allows us to extract data from arrays, and objects and set them into new distinct variables.

- **inner HTML** - Innerhtml is a property of HTML elements in JavaScript that allows you to get or set the HTML content within an element.

- **Query selecter vs get element**

querySelector is the newer feature.getElementById is better supported than querySelector.

querySelector is better supported than getElementsByClassName but querySelector gives you a static node list while getElementsByClassName gives you a live node list.

querySelector lets you find elements with rules that can't be expressed with getElementById and getElementsByClassName.

- **ARRAY METHODS** - Join -The join() method returns an array as a string.
  toString() - It converts an object or value to its string representation.
  Arr.shift()- delete the first element in an array
  Arr.unshift()- adding an element to first
  Arr.push()- add value to the array at last
  Arr.pop ()- delete the last element in an array
  Arr. splice()- changes the array by removing or replacing an element from an array.
  Arr.slice() - ARRAY CLONING Can be done through Slice()
  Arr.reverse()- to reverse an array

- **foreach** - method calls a function for each element in an array.

- In an **entry check loop**, such as the **while loop**, the condition is checked before the execution of the loop's code block. The loop checks the condition first, and if it evaluates to true, the code block is executed. If the condition is false, the loop is not executed at all. The loop may not execute if the condition is initially false, as the condition is checked before entering the loop.

- In an **exit check loop**, such as the **do-while loop**, the condition is checked after the execution of the loop's code block. The loop will execute the code block first, and then evaluate the condition to determine if it should continue iterating or exit the loop.The loop is guaranteed to execute at least once because the condition check occurs after the code block.

- **Map**() - is used to create a new array from an existing array.

- **Filter()** - method takes each element in an array and applies a conditional statement against it. If this conditional returns true, the element gets pushed to the output array. If the condition returns false, the element does not get pushed to the output array.

- **Reduce**() -method reduces an array of values down to just one value. To get the output value, it runs a reducer function on each element of the array.

- **Call** - The call method is used to invoke a function with a specific **this value** and individual arguments.

- **Apply** - The apply method is similar to call, but it accepts arguments as an array or an array-like object.

- **Bind** - The bind method creates a new function with a specific **this value** and any initial arguments.

- **String methods**
  Sk.charAt() - used to call using index
  Sk.charCodeAt()
  Sk.concat(text)- concatenate
  Sk.toUpperCase()
  Sk.toLowerCase()
  Sk.slice() - slice a string
  Sk.split() - is used to convert a string into an array

- **Promise, promise methods**
  Promise - is used to handle asynchronous operations in javascript, they are easy to manage when dealing with multiple asynchronous operations
    - **Three states**
      - **Pending** - initial state neither fulfilled nor rejected
      - **Fulfilled** - meaning that the operation was a success
      - **Rejected** - meaning that the operation failed

  Methods
  1. Resolve - returns a new promise object that is resolved with the given value
  2. Then () - appends the resolved handler call back
  3. Finally () - This method allows you to specify a callback that will be executed regardless of whether the Promise is fulfilled or rejected. It's often used for cleanup operations.
  4. Catch - used for handling Promise rejections.
  5. All -

- **Object.freeze()** is a JavaScript method that freezes an object, making it immutable. When an object is frozen, its properties cannot be added, modified, or deleted. Any attempt to change the object's properties or its prototype will fail silently or throw an error in strict mode.

- **Representational State Transfer (REST)** is a software architectural style that defines the constraints to create web services. The web services that follow the REST architectural style are called RESTful Web Services.

- **Session Timeout** - Session timeout refers to the period of inactivity after which a user's session on a web application or system is automatically terminated. In other words, if a user doesn't interact with the application for a certain duration, their session is considered inactive and they are automatically logged out or prompted to log in again when they try to perform an action.

- **Debouncing** is a technique used in web development to optimize the performance of functions or event handlers that are triggered in response to user input, such as scrolling, resizing, or typing. It prevents a function from executing repeatedly in a short period of time, especially when the event fires rapidly, like when a user types quickly or scrolls quickly. Debouncing is

achieved by delaying the execution of the function until after a certain amount of time has passed since the last occurrence of the event.

- **Use Cases**
  - **Search Suggestions**: When users are typing a search query, you might want to fetch search suggestions from a server. Debouncing ensures that the request is only sent after the user has paused typing.
  - **Infinite Scrolling**: When a user scrolls through a list of items, you can use debouncing to delay loading more items as they scroll, preventing excessive requests.

- **Docker**
  Docker is a platform for developing, shipping and running applications inside containers. Containers are lightweight, standalone, and executable packages that contain everything needed to run a piece of software, including the code, runtime, system tools, libraries, and settings. Docker provides a standardized way to package and distribute applications and their dependencies, ensuring consistency and reproducibility across different environments
-
- **REST** operator in JavaScript is denoted by three dots (. . .) and is used in function definitions and destructuring assignments. It allows you to work with an arbitrary number of arguments or elements as an array-like structure.

- **XSS SCRIPTING -** Cross-site scripting (XSS) is a type of security vulnerability that occurs when a web application includes untrusted or malicious data on its web pages. This can allow an attacker to execute malicious scripts in the context of an innocent user's browser. JavaScript is commonly used for such attacks, as it runs directly in the browser and can manipulate the Document Object Model (DOM) of a web page.