1) What is Javascript.

Ans- 1) Javascript is client side as well as server side scripting language and it allow you to make webpage interactive.

2 ) It is primarily used for web development.

3 ) javascript is dynamically typed language.

Dynamically typed – the type of a variable is determined at runtime.

4 ) It is function oriented,object oriented or multiparadiagm scripting language

multiparadiagm scripting language -A multiparadigm programming language is a language that supports more than one programming paradigm.

2) What is Exceptions

JavaScript Exceptions: In JavaScript, exceptions are unexpected events that occur during program execution and disrupt the normal flow. These events can be fixed by the program using a try-catch block

2) How Browser Execute Javascript.

Parsing:

The first step is to parse the HTML, CSS, and JavaScript files to create the Document Object Model (DOM), the CSS Object Model (CSSOM), and the Abstract Syntax Tree (AST) for the JavaScript code.

The DOM represents the structure of the HTML document, the CSSOM represents the styles applied to the document, and the AST represents the syntactic structure of the JavaScript code.

Compilation:

After parsing, the JavaScript code is compiled into machine code. Modern browsers use a Just-In-Time (JIT) compiler for this purpose. The JIT compiler translates the JavaScript code into native machine code that can be executed by the computer's processor.

Execution:

The compiled code is then executed by the browser's JavaScript engine. Each browser has its own JavaScript engine (e.g., V8 in Chrome, SpiderMonkey in Firefox, JavaScriptCore in Safari).

The execution happens in multiple stages: the code is interpreted initially, and as it runs more frequently, the JIT compiler optimizes certain parts of the code for better performance.

Global Execution Context:

The JavaScript engine creates a global execution context for the code. This context includes the global object (e.g., window in browsers), the this keyword, and a scope chain that allows access to variables and functions.

Function Execution Context:

When a function is called, a new execution context is created for that function. This includes its own scope, local variables, and parameters. The function's execution context is added to the top of the scope chain.

3) What are features of Javascript.

Single-threaded: JavaScript is single-threaded, executing one operation at a time in a synchronous manner, but it supports asynchronous operations through events and callbacks.

DOM Manipulation: It has built-in support for manipulating the Document Object Model (DOM), enabling dynamic changes to web content.

**Cross-platform** : means JavaScript can run on different devices, operating systems, and browsers without changes. It provides a consistent way to build and run web applications everywhere.

**Interpreted Language** : means JavaScript is executed directly by the browser without needing to be compiled first. This makes the development process faster and more flexible, as you can test and modify the code instantly.

4) What is DOM ?

DOM stands for Document Object Model.

Object model of your document is known as dom

The Document Object Model (DOM) is a programming interface for web documents. It represents the structure of a document as a tree of objects, where each object corresponds to a part of the document, such as elements, attributes, and text content.

5) How to perform click event on Dom Using Javascript

There are two ways to perform a click event on DOM using JavaScript:

Using the onclick event attribute

Using the addEventListener() method

The onclick event attribute is a static attribute that can be added to any HTML element. When the element is clicked, the code specified in the onclick attribute is executed.

The addEventListener() method is a dynamic method that can be used to add event listeners to any DOM element. An event listener is a function that is called when a specific event occurs on the element.

Here is an example of how to perform a click event on DOM using the onclick event attribute:

<button onclick="alert('Hello World!')">Click Me</button>

Use code with caution.

Learn more

In this example, when the button is clicked, the alert() function is called, which displays a message box with the text "Hello World!".

Here is an example of how to perform a click event on DOM using the

addEventListener() method:

const button = document.querySelector('button');

button.addEventListener('click', function() {

alert('Hello World!');

});

Use code with caution.

Learn more

In this example, the addEventListener() method is used to add an event listener to the button element. The event listener is a function that is called when the button is clicked. The function displays a message box with the text "Hello World!".

6) What is alert() and confirm() ?

alert Method: The alert method is used to display a simple message to the user in a popup dialog box.

It only has one button (OK), and it does not provide any options for user input.

It is commonly used for displaying information or notifications to the user.

confirm Method:

The confirm method is used to display a message to the user and request confirmation. It typically has two buttons: "OK" and "Cancel."

It returns a boolean value (true if the user clicks "OK" and false if the user clicks "Cancel").

It is commonly used for asking the user to confirm an action before proceeding.

alert("This is an alert message!");

var userConfirmed = confirm("Do you want to proceed?");

if (userConfirmed) {

// User clicked "OK"

// Perform the action

} else {

// User clicked "Cancel" or closed the dialog

// Do something else or abort the action

}

7) What is difference between undefined and null

undefined:

undefined:  
A variable is **undefined** when it is declared but not given any value. It is the default value of uninitialized variables.

* A function without a return statement automatically returns **undefined**

Example:

let variable;

console.log(variable); // Output: undefined

function doSomething() {

// No return statement, so the function returns undefined

}

null:

Null is a special value in programming that means "nothing" or "no value." It is assigned on purpose to show that a variable or property has no value.

Example:

let variable = null;

console.log(variable); // Output: null

8) What is difference between == and ===

In JavaScript, the difference between == and === is that

== compares the value of two variables, but not their data types, while === checks both the value and datatypes of two variables.

console.log("5" == 5); // Output: true (after type coercion)

console.log(false == 0); // Output: true

console.log("5" === 5); // Output: false (string is not equal to number)

console.log(false === 0); // Output: false (boolean is not equal to number)

9) What is function . Can we overload function in javascript.

A function in JavaScript is a block of code that performs a specific task.

JavaScript does not support function overloading and is executed after the completion of some asynchronous operation or task.

10) What is callback ? Explain with suitable example.

callback function is a function which is passed as argument to another function .

function add(a,b,callback){

console.log("A:"+a);

console.log("B:"+b);

var result= a>b?"a is greater":"b is greater";

callback(result);

}

add(10,20,(result)=>{

console.log(result)

});

11) What is promise? What are advantage of promise over callback.

JavaScript Promises make handling asynchronous operations like API calls, file loading, or time delays easier. Think of a Promise as a placeholder for a value that will be available in the future. It can be in one of three states:

1. **Pending**: The operation is still running
2. **Fulfilled**: The operation finished successfully.
3. **Rejected**: The operation failed.

**Advantages of Promises over Callbacks:**

1. **Improved Readability**:
   * Promises make asynchronous code easier to read and understand compared to nested callbacks (callback hell).
2. **Better Error Handling**:
   * Promises handle errors using a catch block, making it easier to manage errors in asynchronous code.
3. **Chaining**:
   * Promises allow chaining multiple .then() calls, making the flow of execution more linear and less complex.
4. **Avoids Callback Hell**:
   * Callbacks often result in deeply nested code, which is hard to debug and maintain. Promises solve this by flattening the structure.

12) What is callback hell.

Callback hell happens when you use many nested callbacks in asynchronous code. This makes the code:

* Hard to **read**.
* Hard to **debug**.
* Hard to **maintain**.

It is also called the **"pyramid of doom"** because the nested structure looks like a pyramid due to excessive indentation.

// Asynchronous function 1

function asyncOperation1(callback1) {

setTimeout(function() {

console.log("Async operation 1 completed");

callback1("Result from operation 1");

}, 1000);

}

// Asynchronous function 2

function asyncOperation2(result1, callback2) {

setTimeout(function() {

console.log("Async operation 2 completed with " + result1);

callback2("Result from operation 2");

}, 1000);

}

// Asynchronous function 3

function asyncOperation3(result2, callback3) {

setTimeout(function() {

console.log("Async operation 3 completed with " + result2);

callback3("Result from operation 3");

}, 1000);

}

// Nested callbacks (Callback hell)

asyncOperation1(function(result1) {

asyncOperation2(result1, function(result2) {

asyncOperation3(result2, function(result3) {

console.log("Final result: " + result3);

});

});

});

13) What is closure ?

A closure is the combination of a function bundled together (enclosed) with the lexical environment. In other words, a closure gives you access to an outer function's scope from an inner function.

function init() {

var name = "Mozilla"; // name is a local variable created by init

function displayName() {

// displayName() is the inner function, that forms the closure

console.log(name); // use variable declared in the parent function

}

displayName();

}

init();

14) What is IIFEs(Immediately Invoked Function expression).

An **IIFE (Immediately Invoked Function Expression)** is a JavaScript function that is **executed immediately after it is defined**. It is wrapped in parentheses to make it an expression and followed by () to invoke it.

**Use Cases of IIFE**

1. **Avoid polluting the global namespace**  
   Encapsulates code within a function, preventing variable leakage into the global scope.
2. **To create closures**  
   Allows the creation of private variables and functions by enclosing them in the IIFE.
3. **Avoid conflicts of variable names**  
   Helps avoid variable name collisions between libraries and programs.
4. **Private and public variables/methods**  
   Enables modular code with both private and public elements.
5. **Execute async/await functions**  
   IIFEs can be used for asynchronous code execution immediately.
6. **JQuery Library**  
   Commonly used in libraries like JQuery for scope management and initialization.
7. **Work with require functions**   
   Helps manage modules and dependencies.

15) What is annonymous function ?

A function without name is called anonymous function .

Anonymous functions are commonly used as arguments to other functions.

var add = function(x, y) {

return x + y;

};

// Anonymous function assigned to the variable 'add'

What is hoisting in javascript.

Hoisting in JavaScript is a behavior where variable and function declarations are moved to the top of their containing scope during the compilation phase, before the code is executed. This allows you to use variables and functions before they are declared in the code.

Variable Hoisting:

console.log(x); // Output: undefined

var x = 5;

console.log(x); // Output: 5

Function Hoisting:

javascript

Copy code

sayHello(); // Output: Hello!

function sayHello() {

console.log("Hello!");

}

sayHi(); // This would result in an error

var sayHi = function() {

console.log("Hi!");

};

16) What is difference between var, let and const keyword.

Ans. -> var let const

var : The scope of a var variableis functional or global scope.

let : The scope of a let variable is block scope.

const : The scope of a const variable is block scope.

var : It can be updated and re-declared in the same scope.

let : It can be updated but cannot be re-declared in the same scope.

const : It can neither be updated or re-declared in any scope.

var : It can be declared without initialization.

let : It can be declared without initialization.

const : It cannot be declared without initialization.

var : It can be accessed without initialization as its default value is “undefined”.

let : It cannot be accessed without initialization otherwise it will give ‘referenceError’.

const : It cannot be accessed without initialization, as it cannot be declared without initialization.

var : These variables are hoisted.

let : These variables are hoisted but stay in the temporal dead zone untill the initialization.

const : These variables are hoisted but stay in the temporal dead zone untill the initialization.

// VAR

function example() {

var x = 10;

if (true) {

var x = 20;

console.log(x); // Output: 20

}

console.log(x); // Output: 20

}

//LET

function example() {

let x = 10;

if (true) {

let x = 20;

console.log(x); // Output: 20

}

console.log(x); // Output: 10

}

17)Explain use strict

"use strict"; is a statement that tells the JavaScript engine to run the code in strict mode. Strict mode is a set of rules that makes JavaScript code more secure and easier to understand.

18)What is event bubbling and event capturing

19)What are the primitive datatype in javascript

String, Boolean, Number, BigInt, Null, Undefined, and Symbol are the six primitive data types available in JavaScript

20)What are different types of popup boxes available in javascript

There are three main types of popup boxes available in JavaScript:

Alert boxes:

These boxes are used to display information to the user. They have only one button, "OK", and the user must click it to close the box.

Confirm boxes:

These boxes are used to get confirmation from the user before performing an action. They have two buttons, "OK" and "Cancel", and the user must click one of them to close the box. If the user clicks "OK", the action is performed; if the user clicks "Cancel", the action is not performed.

Prompt boxes:

These boxes are used to get input from the user.

21)What will happen if an infinite while loop is run in javascript

If an infinite while loop is run in JavaScript, it will cause the program to hang or become unresponsive. An infinite loop is a loop that never exits on its own, meaning the condition specified in the loop always evaluates to true.

22) List HTML DOM mouse events

onclick,ondblclick, onmouseover, onmouseout, onmouseup,onmousemove

Here’s a list of common HTML DOM mouse events that detect user interactions with the mouse:

1. click: Triggered when the mouse button is clicked.
2. dblclick: Triggered when the mouse button is double-clicked.
3. mousedown: Triggered when a mouse button is pressed down.
4. mouseup: Triggered when a mouse button is released.
5. mousemove: Triggered when the mouse pointer moves.
6. mouseover: Triggered when the mouse pointer enters an element.
7. mouseout: Triggered when the mouse pointer leaves an element.
8. contextmenu: Triggered when the right mouse button is clicked, typically used to display a context menu.

23)How to get the last index of a string in javascript lastIndexOf

In JavaScript, you can use the lastIndexOf() method to get the last index of a specific substring within a string. Here's an

**example:**

let myString = "Hello, World! Hello again.";

// Find the last index of the substring "Hello"

let lastIndex = myString.lastIndexOf("Hello");

console.log("Last index of 'Hello':", lastIndex);

24)Describe negative infinity in javascript

when any negative number is divided by zero then it give result as -Infinity or It can also be created by using the Number.NEGATIVE\_INFINITY constant.

25)Explain await and aync ? How to use await and async?

The async keyword is used to declare an asynchronous function. An asynchronous function always returns a promise. Inside an async function, you can use the await keyword to pause the execution of the function until the promise is resolved.

The await keyword is used inside an async function to wait for a promise to settle (either resolve or reject). When await is used with a promise, it pauses the execution of the async function until the promise is resolved, and it returns the resolved value

26)How to handle the excption in javascript.

JavaScript provides the try, catch, finally, and throw statements for exception handling.

27) what is Authentication

Authentication is the process of verifying the identity of a user or system. It ensures that the person or system trying to access a resource **is the right one**.

NODE.JS

What is Node.js? Explaing the advantage of Node.js Over java and php

Node.js is a JavaScript runtime that allows you to run JavaScript on servers.

Node.js uses asynchronous programming!

Node-js uses single threaded javascript event loop module .

What are the limitation of Node.Js

Single-threaded execution:

Node.js uses a single thread to handle all requests. This can be a bottleneck for CPU-intensive tasks.

Memory usage:

Node.js can use a lot of memory, especially if you have a lot of objects in your application.

It is not ideal for CPU-intensive tasks

What is Headers

Http Headers are an importand part of the API request and response as they represent the meta data associated With the API request and response

Headers carry information for the Request and Response Body

React Commoud =>{

* Npm create vite
* Open Folder
* Npm i

}

How Node.Js Works

How Node.js is single threaded

Explain any five built-in package/Dependency name in node.js

What is module in node.js

what is Module.exports

How to create server in node.js

How node.js handle multiple request

How to use url module in node.js

What is setInterval, setTimeout

What is \_\_dirname and \_\_filename

What is synchronous/Blocking and Asynchronous/Non-blocking code in node.js

What is file system in node.js

What are the differenct type of flag using in node.js

What is stream in node.js? Explain the types of stream

How to pipe stream in node.js

What is request and response in node.js

What is package.json and package.lock

What is npm ? How to install dependency/module at application level and Environment elevel

How do you manage packages in your node.js project

How Node.js is better then other framework

What are the some commonly used timing features of Node.js

What is fork in node.js

How do you create a simple server in node.js that return hello world

How many types of API functions are there in node.js

What is REPL and how to use it?

What is purpose of module.exports.

What is an event-loop in Node.js

If Node.js is single threaded then how does it handle multiple request/ concurrency

Differenctiate between process.nextTick() and setImmediate()

What is node.js stream

What is middleware

Explain what a reactor pattern in node.js

Describes the exit code of Node.js

What is an EventEmitter in node.js

What is a thread pool and which library handles it in node.js

What is purpose of NODE\_ENV

How would you connection mongodb database to node application

What are different type of http request

What is difference between get and post

What is query string and how to send the data in get request

What is the use body parser

How to set the path of static file in express

What are types of Middleware in express ? Explain with suitable example

Does order of middleware matters in express.

What is express.js

What are some distinctive features of Express

is Express.js fron-end or backend framework?

Why do we use express.js

What is difference between express.js and node.js

What do you understand by Scaffolding in Express.Js

Which are the argument available to an Express.Js route handler function

How can you allow CORS in Express.jS?

How can you deal with Error handling in Express.js? Explain with an example

Write a code to start serving static file Express.JS

How can we render plain HTML in express

How can we send the data while rendering page in express

How to enable debugging in express app?

What is routing and How routing works in express

How dynamic routing works in express.js

Database

What do you understand by term database ?

Define DBMS

Define RDBMS

What are various type of relationships in database? define them

Explain normalization

What are different types of normalization?

What is SQL ?

How many SQL statement used ? Define them

Enlist some commands of DDL , DML and DCL

Explain the term record, field and table in database

What is view in database

What are advantage and disadvantage of view?

Define Entity , Entity Type and Entity Set

Define a weak entity set

What do you undestand by join ? Explain different types of Join

What is self join

Explain Primary key, Foreign key, unique key, composite key, super key

What is projection in database?

What is difference between delete, truncate and drop command

Based on given table, solve the following query ?

**Employee table**



**a)** Write the SELECT command to display the details of the employee with empid as 1004.

**b)** Write the SELECT command to display all the records of table Employees.

**c)** Write the SELECT command to display all the records of the employee whose name starts with the character ‘R’.

**d)** Write a SELECT command to display id, age and name of the employees with their age in both ascending and descending order.

**e)** Write the SELECT command to calculate the total amount of salary on each employee from the below Emp table.

22.What is mongodb ? What are advantage of mongodb

23.What is difference between SQL and NoSQL database

24.What is document in mongodb?

25.What is collection in mongodb

26.What is mongo shell

27.What are some features of mongodb

28.How to add data in mongodb ?

29.How do you update data in mongodb

30.How do you delete document

31.How to perform query in mongodb

32.What are datatypes in mongodb

33.What is index and how to create index in mongodb

34.Explain the set modifier in mongodb

35.Does mongodb support primary key and foreign key relationship ?

36.Explain the structure of ObjectId in mongodb

Ans :-ObjectID is a 12-byte BSON type. These are:

4 bytes value representing seconds

3 byte machine identifier

2 byte process id

3 byte counter

Is it true that MongoDB uses BSON to represent document structure?

**37.What are Indexes in MongoDB?**

In MondoDB, Indexes are used to execute query efficiently. Without indexes, MongoDB must perform a collection scan, i.e. scan every document in a collection, to select those documents that match the query statement. If an appropriate index exists for a query, MongoDB can use the index to limit the number of documents it must inspect.

**38.By default, which index is created by MongoDB for every collection?**

In-which language mongodb is writtern ?

**39.What will happen when you remove a document from database in MongoDB? Does MongoDB remove it from disk?**

40.What is capped and uncapped collection in mongodb

41.What is lookup in mongodb

42.How to get record of second max age record from mongodb collection

43.What is mongoose ?

44.What is mongoose Schmea

45.How to set reference in mongoose Schema

46.How populate works in mongoose ?

47.List the name of the mongoose method to perform CRUD operation

48.What is default connction pool size in mongoose

49.How to set connection pool size in mongoose

50.What is connection pool and what are advantage of using connection pool size