**NODE JS INTERVIEW BIT QUESTION**

1. **What is a first class function in JavaScript?**

In JavaScript, a first-class function refers to the concept of treating as first-class citizens. This means that functions can be assigned to variables, passed as arguments to other functions, and returned as values from other functions.

Ex- FirstClassfunctn.js file in javaScript interview folder me

1. **What is Node.js and how it works?**

Node.js is an open-source (free), server-side JavaScript runtime environment which built on Chrome's V8 JavaScript engine. It allows developers to run JavaScript code on the server, provides an event-driven, non-blocking I/O model, making it lightweight and efficient for building scalable network applications.

Node.js works by using an event loop, which allows it to handle multiple requests simultaneously without blocking the execution of other code. When a request is received, Node.js registers a callback function to be executed once the request is completed

Node.js also has a built-in module system that allows developers to easily import and use external libraries and modules to use. Ex- npm module ,http module , external mongoose librabry, jwt token librabry , cloudinary etc . This makes it easy to extend the functionality of Node.js applications.

Overall, Node.js provides a powerful platform for building server-side applications, APIs, real-time applications, and more using JavaScript.

**12)What is REPL?**

REPL stands for "Read-Eval-Print Loop." It is an interactive programming environment that allows user to enter the code and execute commands and immediately see the results without installing any software applicaation. The REPL reads your input first, evaluates it (i.e interprets the entered code wch has writtten by user), prints the result, and then repeats the process, That’s y its called REPL.

It provides a quick and iterative way to experiment with code, test functionality, and explore language features. without the need for compiling or running complete programs.

**Ex-**

**> 2 + 2**

**4**

**> const message = "Hello, REPL!";**

**undefined**

**> console.log(message);**

**Hello, REPL!**

**Undefined**

**4)How is Node.js better than other frameworks most popularly used?**

* Node.js provides simplicity in development because of its non-blocking I/O and even-based model results in short response time and concurrent processing, unlike other frameworks where developers have to use thread management.
* It runs on a chrome v8 engine which is written in c++ and is highly performant with constant improvement.
* Also since we will use Javascript in both the frontend and backend the development will be much faster.
* And at last, there are ample libraries so that we don’t need to reinvent the wheel.

**5) Explain the steps how “Control Flow” controls the functions calls?**

* Control the order of execution
* Collect data
* Limit concurrency
* Call the following step in the program.

**6) What is fork in node JS?**

In Node.js, the fork() function is a method provided by the child\_process module. It is used to create a new child process that is a replica or clone of the parent process that runs a separate instance of the Node.js event loop.

The fork() function is allowing for inter-process communication (IPC) between the parent and child processes.

When you call fork(), it creates a new Node.js process that runs the specified module. This module can be a JavaScript file that you want to execute as a separate process. It can open any app by exec method use . The child process created by fork() inherits the standard input/output streams (stdin, stdout, stderr) from the parent process, allowing for communication between the parent and child processes.

EX 1)

const cp= require('child\_process');

//open chrome (when right button click to run code )

cp.exec(“start chrome”)

Ex -2

const { fork } = require('child\_process');

// Fork a new child process

const child = fork('child.js');

// Listen for messages from the child process

child.on('message', (message) => {

console.log('Received message from child:', message);

1});

// Send a message to the child process

child.send('Hello from parent!');

) **What tools can be used to assure consistent code style?**

ESLint can be used with any IDE to ensure a consistent coding style which further helps in maintaining the codebase.

**14) What is the purpose of module.exports?**

The primary purpose of module.exports is to export functions, objects, or values from a module, making them accessible to other modules that require or import them.

Here are a few key points about the purpose of module.exports:

Exporting Functions and Objects: By assigning a function or an object to module.exports, you make it available for other modules to use. This allows you to encapsulate code and provide a well-defined interface for interaction with the module.

Reusability and Code Sharing: module.exports facilitates the sharing of code between different modules in your application. By exporting functions, objects, or classes, you can reuse them in multiple files, promoting code reuse, and reducing redundancy.

**9)Why is Node.js single-threaded?**

Node.js is often described as single-threaded because it uses a single thread to handle incoming requests and execute JavaScript code. And also beacause it uses an event-driven, non-blocking I/O model. This means that it operates on a single thread, handling multiple concurrent requests asynchronously.

The single-threaded nature of Node.js can be acheives by leveraging its underlying event loop mechanism. When a request is received, Node.js registers a callback function to handle that request and continues to process other incoming requests also. As the I/O operations, such as reading from files or making network or API requests, the thread is not blocked while waiting for these operations to complete(is operation k pura hone ka wait krte hue), Instead, Node.js continues executing other tasks too. Once an I/O operation is finished, the callback associated with that operation is pushed into the event loop for execution .

**10) How do you create a simple server in Node.js that returns Hello World ?**

To create a simple server in Node.js that returns "Hello World" as the response, you can follow these steps:

Step 1: Set up a new Node.js project by creating a directory for your project and navigating to it in the command line.

Step 2: Initialize the project by running the following command in terminal:

npm init -y

Step 3: Install the express package, which is a popular Node.js web framework, by running the following command:

npm install express

Step 4: Create a new file called server.js (or any other name you prefer) in the project directory.

Step 5: Open server.js in a text editor and add the following code:

Javascript code

//express instance create

const express = require('express');

const app = express();

//port define

const port = 3000;

app.get('/', (req, res) => {

res.send('Hello World');

});

app.listen(port, () => {

console.log(`Server running on port ${port}`);

});

Step 6: Save the file.

Step 7: In terminal the following command to start the server:

node server.js

Step 8: The server should now be running. Open a web browser and navigate to http://localhost:3000. You should see the message "Hello World" displayed in the browser.

OUTPUT - "Hello World"

**24) What is middleware?**

check notebook for answer of this question

**22)What is node.js streams?**

Node.js Streams are instances of EventEmitter which can be used to work with streaming data in Node.js. They can be used for handling and manipulating , moving streaming large files(videos, mp3, etc) over the network. They use buffers as their temporary storage.

Streams is a sequence(collection of data jo ek sequence क्रम me Rakha hua hai) of data that is being moved from one computer to another computer over the internet Or trasfered data one file to another within the same computer.

Process of streams In chunks(large data converted into piece, portion, or "टुकड़ा) form instead of waiting for the entire data to be available before processing

Ex: watching a video on YouTube

The data arrives in chunks and you watch in chunks while the rest of the arrives over time , jaise jaise ap watch krte jate ho. And if you want to download then its doesn’t load whole data .

Ex: transferring file contents from fileA

(You don’t wait to entire content save in a temporary memory before its move into a file B).

The contents arrive in chunks and you transfer in chunks while the remaining

contents arrive over time

it Prevents unnecessary data downloads and memory usage.

There are mainly four types of the stream:

* Writable: the streams to which data can be written (for example, fs.createWriteStream()).
* Readable: the streams from which data can be read (for example, fs.createReadStream()).
* Duplex: this streams are used for both Readable and Writable (for example, net.Socket).
* Transform: Duplex streams that can modify or transform the data as it is written and read (for example, zlib.createDeflate()). is called Transform stream .

**23. What are node.js buffers?**

In Node.js, a buffer is a temporary storage area for holding and manipulating binary data until it consumed. It is a fixed-size chunk of memory ,If there is data already processed , Node puts the arriving data in a buffer.

It is an intentionally small area that Node maintains in the runtime to process a stream of data dataBuffers are commonly used in scenarios where data needs to be handled in its binary form, such as reading files, working with network streams, or interacting with binary protocols.

Ex: streaming a video online

If your internet connection is fast enough, the speed of the stream will be fast enough to

instantly fill up the buffer and send it out for processing

That will repeat till the stream is finished

If your connection is slow, after processing the first chunk of data that arrived, the video

player will display a loading spinner which indicates it is waiting for more data to arrive

Once the buffer is filled up and the data is processed, the video player shows the video

While the video is playing, more data will continue to arrive nd wait in the buffer.

Buffers also provide methods for copying, slicing, and concatenating buffers, as well as converting between different character encodings and object .

Once a buffer is created, you can read from and write to it using methods such as buffer.write(), buffer.toString(), buffer.slice(), and more. These methods allow you to manipulate the data at the byte level.

Ex – index.js file in js interview folder.

**21) How can we use async await in node.js?**

In Node.js, you can use async/await to write asynchronous code that appears more synchronous and easier to read and understand.

async/await is built on top of promises and allows you to write asynchronous code using a more sequential and linear style.

Here's how you can use async/await in Node.js:

async function fetchData() {

try {

const response = await fetch('https://api.example.com/data');

const data = await response.json();

console.log(data);

} catch (error) {

console.error('Error:', error);

}

}

fetchData();