Discuss Blocking and Non-Blocking I/O strategy with respect to Node JS considering Restaurant scenario.

→ In computer programming, blocking and non-blocking I/O refers to two different strategies for handling data access.

Blocking I/O is a synchronous approach in which a process or thread requesting data must wait for the data to be available before it can continue executing. For example, if a process or thread makes a request to read data from a file it will block until the data has been read and is available to be processed. This can be an effective strategy in certain situations but it can also lead to delay and bottlenecks if the data is not immediately available or if the process or thread is waiting for data from multiple sources.

Non Blocking I/O on the contrary is an asynchronous approach in which a process or thread can continue executing even if the data it has requested is not yet available when the case, the process or thread will register a callback function that will be executed when the data becomes available. This allows the process or thread to continue executing while it is waiting for data rather than being blocked.

Taking a scenario of a restaurant, non blocking I/O might be used to manage the flow of orders and requests from the customers. For example when a customer places an order the restaurants customer order system might register a callback function to handle the order once it is ready to be prepared. In the mean time, the system can continue processing order requests

and orders rather than being blocked until the current order is ready. This can help in improving the efficiency and responsiveness of the restaurants operation.

Node JS is a Javascript runtime that uses non blocking I/O to handle requests and responses asynchronously. This makes it well suited for building scalable network applications that can handle a high volume of concurrent requests.

4. Discuss Node JS Modules in detail - NPM, GLOBALS, FILE SYSTEM, CALLBACKS, EVENT and HTTP.

→

NPM

NPM stands for Node Package Manager It is the default package manager for Node JS and is written entirely in javascript NPM is developed by Issac Z Schluter. NPM manages all the modules and packages for Node JS and consists of command line client npm. It gets installed in the system with the installation of node JS. NPM can install all the dependencies of a project through the package.json file. It can also update and uninstall packages.

GLOBAL

Global modules are node packages that are installed on your system rather than your project directory. They allow us to use the package as a tool anywhere on local computer. By saying global, we are talking about the scope of usage of these module. In general, modules are scoped in the project directory only, it means you can't use them outside the project. Since global modules are installed in the computer it can be used anywhere in our system. Global modules get installed in the standard directory.

FILE SYSTEM

The Node JS file system module allow us to work with the file system on our computer. Node JS provides an inbuilt module called FS to handle file operations like creating, reading, deleting, etc. Node JS gives the functionality of the file I/O by providing wrappers around the standard POSIX function.

CALLBACK

Callback modules are those modules that use callback functions to handle asynchronous operations. Callback functions are functions that are passed as arguments to other functions are executed when the operation is complete. A callback is a function which is called when a task is completed thus helps in preventing any kind of blocking and a callback function allows other code to run in the mean time.

EVENT

Event modules are modules that allow Node JS module for working with events, program to handle events. 'Events' module is the built in node JS module for working with events.

HTTP

HTTP modules are modules that allow Node JS program to make HTTP requests and also handle HTTP responses. The 'http' module is the built in Node JS module for working with http. The HTTP module create an HTTP server that listens to server ports and gives a respons to the client.

1. What is Node JS discuss its features in detail.

→ Node JS is an open source, cross platform runtime environment used for development of server side web applications. All the applications in Node JS are written in Java Script and since we know that javascript is platform independent, therefore we can run Node JS in a wide variety of operating system.

Node JS is developed based on an event driven architecture and a non blocking Input / Output API that is designed to optimize an application throughput and scalability for real time web applications. It is to be noted that in Node JS we have a real time, two way connection, where both the client and server can initiate communication thus allowing them to exchange data freely.

FEATURES :

The features of Node JS are as follows :

1. Asynchronous and Event Driven

The libraries are all asynchronous in nature. A server built with Node JS server waits for data from an API. After accessing an API, the server moves on to the next one. In order to receive and track responses of previous API requests, it uses a notification mechanism called events.

2. Single Threaded

Node JS employs a single threaded architecture which uses event looping and thus making it very scalable. In contrast to typical

servers, which create limited threads to process requests, the event mechanism allows the node JS server to reply in a non-blocking manner and makes it more scalable Node JS uses a single threaded program that can handle a large number of requests.

3. Cross Platform compatibility

Node JS may use variety of systems including windows, unix, linux, Mac OS and mobile devices. It can be paired with the appropriate package to generate a self sufficient executable.

4. Uses Java Script

Java Script is used by node JS library which is an important aspect of Node JS from an engineers perspective. Most of the engineers are familiar with java script and hence will find working with node JS is much easier.

5. Fast data streaming

When data is transmitted in multiple streams, processing them takes a long time. Node JS processes data at a very fast rate. It processes and upload a file simultaneously thereby saving a lot of time.

6. NO Buffering

2 What is multithreaded execution and Event driven execution - Explain the limitations of the existing structure and how it overcome by Event driven structure in detail by considering UBER STORY as a case study.

→ Multithreaded execution is a way of running multiple threads concurrently within a single process. Each threads represents a single separate flow of execution and can run concurrently with other threads within the same process. This can be used to improve the performance and responsiveness of a program by allowing different tasks to be run in parallel.

Event driven execution on other hand is a programming paradigm in which the flow of execution is determined by events or triggers. In a event driven system, a program will have a typical main loop that waits for events to occur and then trigger a response to those events. This allows the program to be more responsiveness to external input and to perform tasks so they become necessary rather than in a predetermined order.

LIMITATIONS :

Limitation of multi-threaded execution structure is that it can be difficult to both manage and synchronise the different threads especially if they are accessing shared resources. This can lead to problems such as race conditions and deadlocks.

Taking the case study of UBER, the company uses a combination of multithread execution and event driven execution to manage the complex system of drivers, passengers and rides. The UBER

application uses multithreaded execution to perform various tasks on parallel such as searching for nearby drivers, calculating fares and processing, updating payments.

At the same time, the application uses an event driven structure to respond to events such as a passenger requesting a ride or a driver accepting a drive request. When the passenger requests a ride, the application triggers a service of events that includes searching for a nearby driver, matching the driver with the passenger and updating the status of the ride. This event driven and updating structure allows the UBER application to be responsive to external input and to perform tasks as needed rather than following a predetermined sequence.

Thus from the above explanation, we can say that the overall use of both multithreaded and event driven execution in the UBER application allows the company to manage the complex system of rides and drivers effectively and efficiently.