3) What do you mean by the term I/O ?

I/O is the shorthand for input and output, and it will access anything outside of your application. It will be loaded into the machine memory to run the program, once the application is started.

What is the advantage of using node.js?

It provides an easy way to build scalable network programs, Generally fast, Great concurrency, Asynchronous everything, Almost never blocks

What are the two types of API functions in Node.js ?

The two types of API functions in Node.js are

a) Asynchronous, non-blocking functions

b) Synchronous, blocking functions

8) What is control flow function?

A generic piece of code which runs in between several asynchronous function calls is known as control flow function.

9) Explain the steps how Control Flow controls the functions calls?

Control the order of execution, Collect data, Limit concurrency, Call the next step in program

10) Why Node.js is single threaded?

For async processing, Node.js was created explicitly as an experiment. It is believed that more performance and scalability can be achieved by doing async processing on a single thread under typical web loads than the typical thread based implementation.

No, you cannot access DOM in node.

13) Using the event loop what are the tasks that should be done asynchronously?

I/O operations, Heavy computation, Anything requiring blocking

What are the two arguments that async.queue takes?

The two arguments that async.queue takes

a) Task function

b) Concurrency value

16) What is an event loop in Node.js ?

To process and handle external events and to convert them into callback invocations an event loop is used. So, at I/O calls, node.js can switch from one request to another.

Mention the steps by which you can async in Node.js?

By following steps you can async Node.js

First class functions, Function composition, Callback Counters, Event loops

Single Threaded but Highly Scalable Node.js uses a single thread model for event looping. The response from these events may or may not reach the server immediately. However, this does not block other operations. Thus making Node.js highly scalable. Traditional servers create limited threads to handle requests while Node.js creates a single thread that provides service to much larger numbers of such requests.

9) How Node.js overcomes the problem of blocking of I/O operations?

Node.js solves this problem by putting the event based model at its core, using an event loop instead of threads

21) What are the Challenges with Node.js ?

Emphasizing on the technical side, its a bit of challenge in Node.js to have one process

with one thread to scale up on multi core server.

22) What does it mean non-blocking in node.js?

In node.js non-blocking means that its IO is non-blocking. Node uses libuv to handle its IO in a platform-agnostic way. On windows, it uses completion ports for unix it uses epoll or kqueue etc. So, it makes a non-blocking request and upon a request, it queues it within the event loop which call the JavaScript callback on the main JavaScript thread.

25) What is Callback in node.js?

Callback function is used in node.js to deal with multiple requests made to the server. Like if you have a large file which is going to take a long time for a server to read and if you dont want a server to get engage in reading that large file while dealing with other requests, call back function is used. Call back function allows the server to deal with pending request first and call a function when it is finished.

Q-5. Explain How Does Node.Js Work?

Answer.

A Node.js application creates a single thread on its invocation. Whenever Node.js receives a request, it first completes its processing before moving on to the next request.

Node.js works asynchronously by using the event loop and callback functions, to handle multiple requests coming in parallel. An Event Loop is a functionality which handles and processes all your external events and just converts them to a callback function. It invokes all the event handlers at a proper time. Thus, lots of work is done on the back-end, while processing a single request, so that the new incoming request doesnt have to wait if the processing is not complete.

While processing a request, Node.js attaches a callback function to it and moves it to the back-end. Now, whenever its response is ready, an event is called which triggers the associated callback function to send this response.

Q-6. Explain REPL In Node.Js?

Answer.

The REPL stands for Read Eval Print Loop. It is a simple program that accepts the commands, evaluates them, and finally prints the results. REPL provides an environment similar to that of Unix/Linux shell or a window console, in which we can enter the command and the system, in turn, responds with the output. REPL performs the following tasks.

READ

It Reads the input from the user, parses it into JavaScript data structure and then stores it in the memory.

EVAL

It Executes the data structure.

PRINT

It Prints the result obtained after evaluating the command.

LOOP

It Loops the above command until the user presses Ctrl+C two times.

And the C++ lib UV library takes care of the non-sequential I/O via background workers.

To explain it practically, lets assume there are 100s of requests lined up in Node.js queue. As per design, the main thread of Node.js event loop will receive all of them and forwards to background workers for execution. Once the workers finish processing requests, the registered callbacks get notified on event loop thread to pass the result back to the user.

**How To Avoid Callback Hell In Node.Js?**

Node.js internally uses a single-threaded event loop to process queued events. But this approach may lead to blocking the entire process if there is a task running longer than expected.

Node.js addresses this problem by incorporating callbacks also known as higher-order functions. So whenever a long-running process finishes its execution, it triggers the callback associated. With this approach, it can allow the code execution to continue past the long-running task.

Make Your Program Modular, Use Async Mechanism, Use Promises Mechanism,

**What Are Globals In Node.Js?**

There are three keywords in Node.js which constitute as Globals. These are Global, Process, and Buffer.

Global.

The Global keyword represents the global namespace object.

process object is used to get information on current process. Provides multiple events related to process activities.

**What is setInterval**

The setInterval(cb, ms) global function is used to run callback cb repeatedly after at least ms milliseconds. The actual delay depends on external factors like OS timer granularity and system load. A timer cannot span more than 24.8 days.

This function returns an opaque value that represents the timer which can be used to clear the timer using the function clearInterval(t)

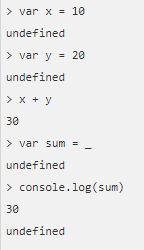
**setTimeout**

The setTimeout(cb, ms) global function is used to run callback cb after at least ms milliseconds. The actual delay depends on external factors like OS timer granularity and system load. A timer cannot span more than 24.8 days.

This function returns an opaque value that represents the timer which can be used to clear the timer.

The \_\_dirname represents the name of the directory that the currently executing script resides in.

The \_\_filename represents the filename of the code being executed. This is the resolved absolute path of this code file. For a main program this is not necessarily the same filename used in the command line. The value inside a module is the path to that module file.

All APIs of Node.js library are aynchronous that is non-blocking. It essentially means a Node.js based server never waits for a API to return data. Server moves to next API after calling it and a notification mechanism of Events of Node.js helps server to get response from the previous API call

npm ls -g

