1) What is node.js?

Node.js is a Server side scripting which is used to build scalable programs. Its multiple advantages over other server side languages, the prominent being non-blocking I/O.

2) How node.js works?

Node.js works on a v8 environment, it is a virtual machine that utilizes JavaScript as its scripting language and achieves high output via non-blocking I/O and single threaded event loop.

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.

4) What does event-driven programming mean?

In computer programming, event driven programming is a programming paradigm in which the flow of the program is determined by events like messages from other programs or threads. It is an application architecture technique divided into two sections 1) Event Selection 2) Event Handling.

5) Where can we use node.js?

Node.js can be used for the following purposes.

- Web applications (especially real-time web apps)
- Network applications
- Distributed systems
- General purpose applications

6) 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

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

The two types of API functions in Node.js are

- Asynchronous, non-blocking functions
- 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.

11) Does node run on windows?

Yes – it does. Download the MSI installer from http://nodejs.org/download/

12) Can you access DOM in node?

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

14) Why node.js is quickly gaining attention from JAVA programmers?

Node.js is quickly gaining attention as it is a loop based server for JavaScript. Node.js gives user the ability to write the JavaScript on the server, which has access to things like HTTP stack, file I/O, TCP and databases.

15) What are the two arguments that async.queue takes?

The two arguments that async.queue takes

- Task function
- 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.

17) 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

18) What are the pros and cons of Node.js?

Pros:

- If your application does not have any CPU intensive computation, you can build it in Javascript top to bottom, even down to the database level if you use JSON storage object DB like MongoDB.
- Crawlers receive a full-rendered HTML response, which is far more SEO friendly rather than a single page application or a websockets app run on top of Node.js.

Cons:

- Any intensive CPU computation will block node.js responsiveness, so a threaded platform is a better approach.
- Using relational database with Node.js is considered less favourable.

19) 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.

20) What is the difference between Node.js vs Ajax?

The difference between Node.js and Ajax is that, Ajax (short for Asynchronous Javascript and XML) is a client side technology, often used for updating the contents of the page without refreshing it. While,Node.js is Server Side Javascript, used for developing server software. Node.js does not execute in the browser but by the server.

21) What are the Challenges with Node.js ?

Emphasizing on the technical side, it's 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.

23) What is the command that is used in node.js to import external libraries?

Command "require" is used for importing external libraries, for example, "var http=require ("http")". This will load the http library and the single exported object through the http variable.

24) Mention the framework most commonly used in node.js?

"Express" is the most common framework used in node.js.

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 don't 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.

SAP Training	Python	Excel	ASP Net	HBase

Project Management	Test Management	Business Analyst	Ethical Hacking	PMP

Live Project	SoapUI	Photoshop	Manual Testing	Mobile Testing

Data Warehouse	R Tutorial	Tableau	DevOps	AWS

Jenkins	Agile Testing	RPA	JUnit	Software Engineering

Selenium	CCNA	AngularJS	NodeJS	PLSQL