1. What is Node JS?  
Node.js is an open source server framework built on Google Chrome's JavaScript Engine (V8 Engine). It is the server side JavaScript platform. Node.js applications are developed in JavaScript and it can be run on various platforms like Windows, Linux, UNIX, Mac OS X, etc.  
Node.js is not a JavaScript library, but it is a platform to execute JavaScript on server side.  
2. What are the benefits of using Node.js?  
**Following are the benefits of the using Node.js.**

* Web application developed in JavaScript is incredibly fast.
* API of the Node.js library is asynchronous which means server never waits for an API return data.
* Node.js can handle thousands of concurrent connection.
* Node.js is supports the exchange of the data using JSON. This makes it light weight as compared to another framework.
* Large numbers of libraries are available in Node.js which enhance the development time.

3. Where can we deploy node application?  
Node.js application cannot be deployed on our existing host server. We use VPS or other server to install node and run on our web application.  
4. What does event-driven programming means?  
It is a programming paradigm in which the flow of program is determined by user action like mouse clicks, key press. Event driven programming is basically used in graphical user interface and JavaScript web application.  
It is an application architecture technique divided into two sections

* Event Selection
* Event Handling

5. What is the key feature of Node.js?  
**Following are the key feature of Node.js.**

* It uses JavaScript as a programming language for both front-end and back-end.
* Node.js applications are highly scalable which helps to improve our application performance.
* A latest version of Node.js is focused on performance improvements and provides better security.
* Node.js supports the caching features, so it improves load time of the application.
* We can develop Restful services API using Node.js.
* Node.js application supports multiple unit testing frameworks.

6. What are NPM and its need in Node.js?  
NPM stands for Node Package Manager. The NPM package manages and support various command to install and remove from the module. It is the online repository of node package and share open source library like jQuery, Angular JS etc.

7. Explain REPL in Node.js? List out some REPL command in Node.js?  
REPL stands for Read-Eval-Print-Loop. It is an interface which accepts the command, execute it and print the result. Node.js comes bundled with a REPL environment.   
**REPL can perform the following task.**

* Read
* Eval
* Print
* Loop

**Below are the REPL commands in Node.js.**

* **Ctrl + c:** terminate the current command.
* **Ctrl + c twice:** it terminates the REPL.
* **Ctrl + d:** terminates the REPL session.
* **Tab key:** display the list of current available command.
* **.break:**exit from the multiline expression.

8. What is module and what are the core modules in Node.js?  
Module is a set of function that we include in our application. Node.js encapsulates JavaScript related code into the single unit of code. Node.js has set of built-in module. We can include any built-in module in our application using **require()** function.  
**Example:**  
var http = require('http')

9. What are callback and callback hell and how to avoid it?  
Callback is like an asynchronous equivalent for a function. Node.js makes heavy use of callbacks and triggers them at the completion of a given task. All the APIs of Node.js are written in such a way that they support callbacks.  
Node.Js works just like a courier boy. He goes to every house to deliver packets. If a person is not available, he calls the person and asks him to call when available. Similarly, Node.js attends one job at a time and doesn't wait for the processing of the request (delivering of packet) to complete. Instead, it attaches a callback function (call to the owner) to it. Whenever the processing of a request completes, an event gets called, which triggers the associated callback function to send the response.   
Callback hell is heavily nested callbacks which make the code unreadable and difficult to maintain. Node.js uses a single-threaded event loop to process queued events. It uses callbacks to allow the code execution to continue past the long-running task. But more the no. of callbacks, longer the chain of returning callbacks would be. It's hard to debug the code and can cause you a whole lot of time.  
**Ways to solve the issue of callback hells:**  
Modularization: break callbacks into independent functions  
Use a control flow library, like async  
Use generators with Promises  
Use async/await

10.  What is Cluster process module in Node.js?  
Node.js runs single threaded programming. But to take advantage of computer's multi-core systems, the Cluster module allows you to easily create child processes that each runs on their own single thread, to handle the load.

11.  Why Node.js is single threaded?

Node.js was created with the believe that more performance and scalability can be achieved by doing async processing on a single thread than the typical thread based implementation.

12.  What is the package.json file in Node.js project?  
All npm packages contain a file in the project root, called package.json. This file holds various metadata relevant to the project. This file is used to give information to npm that allows it to identify the project as well as handle the project's dependencies.    
Node itself is only aware of two fields in the package.json:

{  
  "name" : "barebones",  
  "version" : "0.0.0",  
}

13.   What are the buffers and how to create the buffers?  
Reading data from secondary storage (Hard disk) is time consuming than reading from main memory. Hence OS brings some data to RAM(main memory) for easier and fast accessing. These chunks of data are called Buffers.  
Node.js introduced the Buffer class to deal with binary data because JavaScript has no mechanism for reading or manipulation of the binary data.  
Each buffer has some raw memory which is allocated outside the V8.  
**Creating buffer:-**  
var buffer = new Buffer(16); buffer.write("Welcome", "utf-8") **Output:** 7

14.  What are streams and what are the different types of stream?  
Just like arrays or strings, Streams are collections of data. The difference is that streams might not be available all at once, and they don't have to fit in memory. This makes streams really powerful when working with large amounts of data, or data that's coming from an external source one chunk at a time.  
A Stream is an abstract interface that is used to work with stream data in Node.js application. It lets you read data from a source or write data to a destination in continuous fashion.  
**There are four types of stream in Node.js**

* **Readable:** It is used to read operation.
* **Writeable:** It is used to write operation
* **Duplex:** Used for both read and write operation.
* **Transform:** This is used to compute the output based on input.

15.  What is EventEmitter In Node.Js?  
Node.js application is the event-driven application. Node.js uses the event module for event handling.  
All objects in Node.js which emit events are the instances of events.EventEmitter.   
EventEmitter class lies in the events module.  
EventEmitter provides multiple properties like on and emit. "on" property is used to bind a function with the event and "emit" is used to fire an event.

16.  What are the streams piping in Node.js?

Stream piping is the way to connect one output stream to another output stream. Node.js provides the pipe() method for this purpose.  
We can pass one readable stream as a parameter in pipe() method and this stream will be write into another stream.   
**e.g**

var fs = require('fs');  
var rs = fs.createReadStream('input.txt');  
var ws = fs.createWriteStream('output.txt');  
rs.pipe(ws)

17.  What is the File system module in Node.js?  
Node.js file system provides the functionality to work with file in our local computer. The require ('fs') function must be included when we are using file module.  
**Following are the common use of File System module.**  
**Read files:** fs.readFile() method is used for reading file.

**Create files:** Below methods are used to create file in Node.js.

* fs.appendFile()
* fs.open()
* fs.writeFile()

**Update files:**  
Following methods are used to update file.

* fs.appendFile()
* fs.writeFile()

**Delete files:**  
We use fs.unlink() method for deleting the file  
**Rename files:**  
fs.rename() is used to rename the file.

18.  What is express and why to use express?  
Express framework facilitates the rapid development of Node based Web applications. Express is the light-weight web application framework used to build single page application, multipage application and hybrid web application.  
It allows dynamic rendering of HTML Pages, based on arguments passed to templates.  
**Reason to use Express:**  
We should use Express because of following features.

* It supports the Routing, session, configuration in the web application.
* It also supports content navigation, error handling etc.
* Express supports multiple databases like MySQL, MS SQL, MongoDB etc.

19.  What is template engine? Advantages of Template engine?  
Template engine helps to create an HTML template with minimal code. It injects data into HTML template at client side and produces the final HTML.  
**Advantages of Template engine in Node.js:**

* It improves developer's productivity with faster performance.
* It improves readability and maintainability.
* It maximizes client side processing.
* Single template for multiple pages.
* Templates can be accessed from CDN.

20.  What is the meaning of "non-blocking" in Node.js?  
Non-blocking code are those which doesn't wait for previous assign task to complete. Non-blocking code means that the IO is not blocking.   
Node.js operates on single-thread, but non-blocking I/O calls allows it to support many concurrent connections.  
21.  What is Test Driven Development (TDD)?  
TDD is the test first development approach.   
Test Driven Development (TDD) is a programming practice that instructs developers to write new code only if an automated test has failed. This avoids duplication of code. The primary goal of TDD is to make the code clearer, simple and bug free  
these are the following sequence we generally followed in TDD approach

* Add a test
* Run all tests and see if the new one fails
* Write some changes
* Run tests
* Refactor code
* Repeat