Node.js Interview Questions

A list of top frequently asked **Node.js interview questions** and answers are given below.

---

1) What is Node.js?

Node.js is Server-side scripting which is used to build scalable programs. It is a web application framework built on Google Chrome's JavaScript Engine. It runs within the Node.js runtime on Mac OS, Windows, and Linux with no changes. This runtime facilitates you to execute a JavaScript code on any machine outside a browser.

---

2) Is Node.js free to use?

Yes. It is released under MIT license and is free to use.

---

3) Is Node a single threaded application?

Yes. Node is a single-threaded application with event looping.

---

4) What is the purpose of Node.js?

These are the following purposes of Node.js:

- ○ Real-time web applications

- ○ Network applications

- ○ Distributed systems

- ○ General purpose applications


5) What are the advantages of Node.js?

Following are the main advantages of Node.js:

- Node.js is asynchronous and event-driven. All API?s of Node.js library are non-blocking, and its server doesn't wait for an API to return data. It moves to the next API after calling it, and a notification mechanism of Events of Node.js responds to the server from the previous API call.

- Node.js is very fast because it builds on Google Chrome?s V8 JavaScript engine. Its library is very fast in code execution.

- Node.js is single threaded but highly scalable.

- Node.js provides a facility of no buffering. Its application never buffers any data. It outputs the data in chunks.

---

6) Explain Node.js web application architecture?

A web application distinguishes into 4 layers:

- **Client Layer:** The Client layer contains web browsers, mobile browsers or applications which can make an HTTP request to the web server.

- **Server Layer:** The Server layer contains the Web server which can intercept the request made by clients and pass them the response.

- **Business Layer:** The business layer contains application server which is utilized by the web server to do required processing. This layer interacts with the data layer via database or some external programs.

- **Data Layer:** The Data layer contains databases or any source of data.
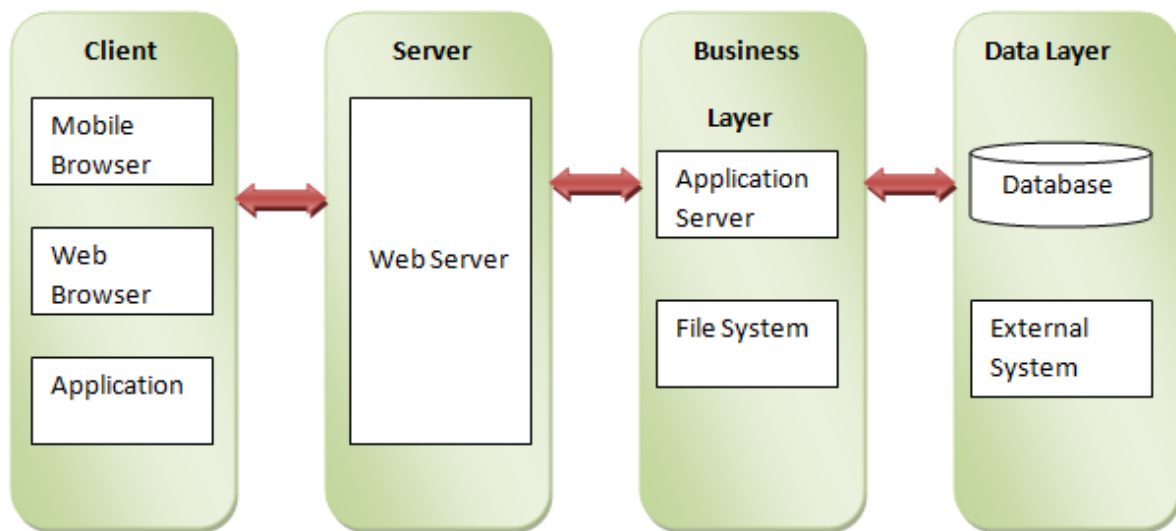
7) What do you understand by the term I/O?

The term I/O stands for input and output. It is used to access anything outside of your application. The I/O describes any program, operation, or device that transfers data to or from a medium or another medium. This medium can be a physical device, network, or files within a system.

I/O is loaded into the machine memory to run the program once the application starts.

8) How many types of API functions are available in Node.js?

There are two types of API functions in Node.js:

- Asynchronous, Non-blocking functions

- Synchronous, Blocking functions

9) What do you understand by the first class function in JavaScript?

When functions are treated like any other variable, then those functions are called first-class functions. Apart from JavaScript, many other programming languages, such as Scala, Haskell, etc. follow this pattern. The first class functions can be passed as a param to another function (callback), or a function can return another function (higher-order function). Some examples of higher-order functions that are popularly used are map() and filter().

---

10) What is the difference between JavaScript and Node.js?

Difference between JavaScript and Node.js

The following table specifies the crucial differences between JavaScript and Node.js:

Comparison features	JavaScript	Node.js
Type	JavaScript is a programming language. More precisely, you can say that it is a scripting language used for writing scripts on the website.	Node.js is an interpreter and run time environment for JavaScript.
Utility	JavaScript is used for any client-side activity for a web application.	Node.js is used for accessing or performing any non-blocking operation of any operating system.
Running Engine	The running engine for JavaScript is Spider monkey (Firefox), JavaScript Core (Safari), V8 (Google Chrome), etc.	The running engine for Node.js is V8 (Google Chrome).

Browser compatibility	JavaScript can only be run in browsers.	The Node.js code can be run outside the browser.
Platform dependency	JavaScript is basically used on the client-side and is used in frontend development.	Node.js is mostly used on the server-side and is used in server-side development.
HTML compatibility	JavaScript is capable enough to add HTML and play with the DOM.	Node.js is not compatible enough to add HTML tags.
Examples	Some examples of the JavaScript frameworks are RamdaJS, TypedJS, etc.	Some examples of the Node.js modules are Lodash, express, etc. We have to import these modules from npm.
Written in	JavaScript is the upgraded version of ECMA script that uses Chrome's V8 engine and is written in C++.	Node.js is written in C, C++, and Javascript.

11) Explain the working of Node.js?

The workflow of a Node.js web server typically looks like the following diagram. Let us see the flow of operations in detail:
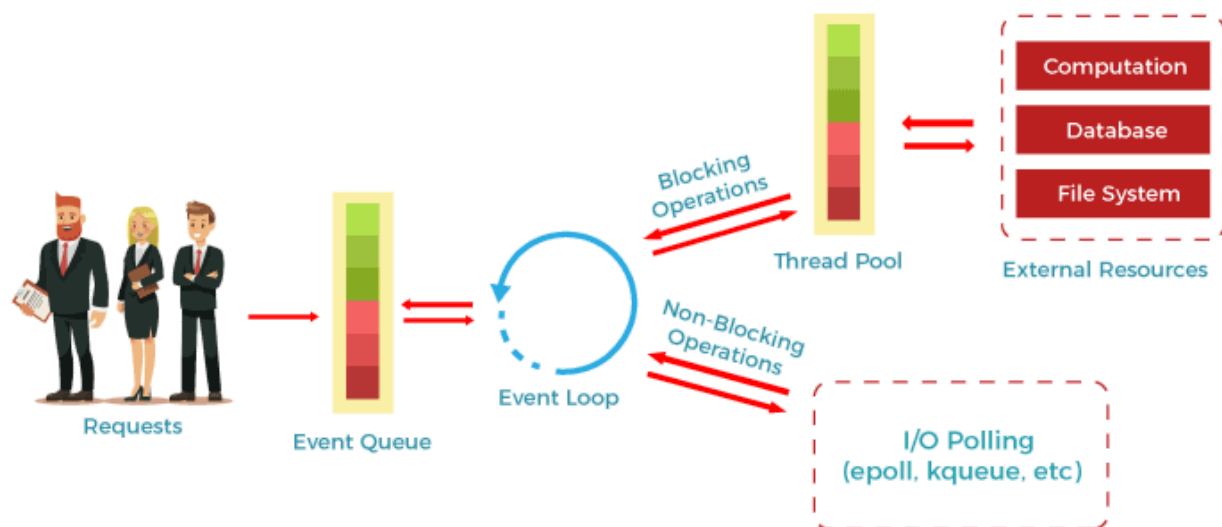
- ○ According to the above diagram, the clients send requests to the webserver to interact with the web application. These requests can be non-blocking or blocking and used for querying the data, deleting data, or updating the data.

- ○ js receives the incoming requests and adds those to the Event Queue.

- ○ After this step, the requests are passed one by one through the Event Loop. It checks if the requests are simple enough not to require any external resources.

- ○ The event loop then processes the simple requests (non-blocking operations), such as I/O Polling, and returns the responses to the corresponding clients.

- ○ A single thread from the Thread Pool is assigned to a single complex request. This thread is responsible for completing a particular blocking request by accessing external resources, such as computation, database, file system, etc.

- ○ Once the task is completed, the response is sent to the Event Loop that sends that response back to the client.

---

12) How can you manage the packages in your Node.js project?

We can manage the packages in our Node.js project by using several package installers and their configuration file accordingly. Most of them use npm or yarn. The npm and yarn both provide almost all libraries of JavaScript with extended features of controlling environment-specific configurations. We can use package.json and package-lock.json to maintain versions of libs being installed in a project. So, there is no issue in porting that app to a different environment.

13) Why is Node.js Single-threaded?

Node.js is a single-threaded application with event looping for async processing. The biggest advantage of doing async processing on a single thread under typical web loads is that you can achieve more performance and scalability than the typical thread-based implementation.

14) What do you understand by callback hell in Node.js?

Callback hell is a phenomenon that creates a lot of problems for a JavaScript developer when he tries to execute multiple asynchronous operations one after the other. A function is called an asynchronous function when some external activity must complete before processing a result. It is called asynchronous because there is an unpredictable amount of time before a result becomes available. These functions require a callback function to handle errors and process the result.

Example:

1. getData(function(a){
2.   getMoreData(a, function(b){
3.   getMoreData(b, function(c){
4.   getMoreData(c, function(d){
5.   getMoreData(d, function(e){
6.   ...
7.   });
8.   });

```
9.        });
10.   });
11. });
```

---

15) How is Node.js better than other most popular frameworks?

Based on the following criteria, we can say that Node.js is better than other most popular frameworks:

- js makes development simple because of its non-blocking I/O and even-based model. This simplicity results in short response time and concurrent processing, unlike other frameworks where developers use thread management.

- js runs on a chrome V8 engine which is written in C++. It enhances its performance highly with constant improvement.

- With Node.js, we will use JavaScript in both the frontend and backend development that will be much faster.

- js provides ample libraries so that we don't need to reinvent the wheel.

---

16) In which types of applications is Node.js most frequently used?

Node.js is most frequently and widely used in the following applications:

- Internet of Things

- Real-time collaboration tools

- Real-time chats

- Complex SPAs (Single-Page Applications)

- Streaming applications

- Microservices architecture etc.

---

17) What are some commonly used timing features of Node.js?

Following is a list of some commonly used timing features of Node.js:

- **setTimeout/clearTimeout:** This timing feature of Node.js is used to implement delays in the code execution.

- **setInterval/clearInterval:** The setInterval or clearInterval timing feature is used to run a code block multiple times in the application.

- **setImmediate/clearImmediate:** This timing feature of Node.js is used to set the execution of the code at the end of the event loop cycle.

- **nextTick:** This timing feature sets the execution of code at the beginning of the next event loop cycle.

---

18) What do you understand by the term fork in Node.js?

Generally, a fork is used to spawn child processes. In Node.js, it is used to create a new instance of the V8 engine to run multiple workers to execute the code.

---

19) Which is the best tool we can use to assure consistent code style in Node.js?

ESLint tool is one of the best tools we can use with any IDE to ensure a consistent coding style. It also helps in maintaining the codebase.

---

20) What is the main difference between front-end and back-end development?

The following table specifies the key differences between a front-end and back-end development:

Front-end Development	Back-end Development
The front-end development in an application refers to the client-side of an application.	The back-end development in an application refers to the server-side of an application.
As the name specifies, the front-end development is the part of a web application where users can see and interact.	As the name specifies, the back-end development consists of everything that happens behind the scenes and users cannot see and interact with.
The front-end development includes everything that attributes to the visual aspects of a web application.	The back-end development generally includes a web server that communicates with the database to serve the users' requests.
HTML, CSS, Bootstrap, jQuery, JavaScript, AngularJS, and React.js are essential front-end development technologies.	Java, PHP, Python, C++, Node.js, etc., are the technologies required for back-end development.
Examples of some front-end frameworks are AngularJS, React.js, jQuery, Sass, etc.	Examples of some back-end frameworks are Express, Django, Rails, Laravel, Spring, etc.

21) Give an example to demonstrate how can we use async await in Node.js?

See the following example of using async-await pattern:

```
1.  function wait (timeout) {
2.   return new Promise((resolve) => {
3.  setTimeout(() => {
4.    resolve()
5.  }, timeout);
6.   });
7.  }
8.  async function requestWithRetry (url) {
9.   const MAX_RETRIES = 10;
10. for (let i = 0; i <= MAX_RETRIES; i++) {
11. try {
12.  return await request(url);
13. } catch (err) {
14.  const timeout = Math.pow(2, i);
15.  console.log('Waiting', timeout, 'ms');
16.  await wait(timeout);
17.  console.log('Retrying', err.message, i);
18. }
19. }
20. }
```

22) What are the modules in Node.js? Which are the different modules used in Node.js?

In Node.js applications, modules are like JavaScript libraries and include a set of functions. To include a module in a Node.js application, we must use the require() function with the parentheses containing the module's name.

Node.js has several modules which are used to provide the basic functionality needed for a web application. Following is a list of some of them:

Core Modules	Description
HTTP:	The HTTP module includes classes, methods, and events to create a Node.js HTTP server.
util:	The util module includes utility functions required in the application and is very useful for developers.
url:	The url module is used to include the methods for URL parsing.
fs:	The fs module includes events, classes, and methods to handle the file I/O operations.
stream:	The stream module is used to include the methods to handle streaming data.
query string:	The query string module is used to include the methods to work with a query string.
zlib:	The zlib module is used to include the methods to compress or decompress the files used in an application.

23) What are buffers in Node.js?

In general, a buffer is a temporary memory mainly used by the stream to hold on to some data until it is consumed. Buffers are used to represent a fixed-size chunk of memory allocated outside of the V8 JavaScript engine. It can't be resized. It is like an array of integers, which each represents a byte of data. It is implemented by the Node. js Buffer class. Buffers also support legacy encodings like ASCII, utf-8, etc.

24) What is error-first callback?

Error-first callbacks are used to pass errors and data. If something goes wrong, the programmer has to check the first argument because it is always an error argument. Additional arguments are used to pass data.

1. fs.readFile(filePath, function(err, data) {
2. if (err) {
3. //handle the error
4. }
5. // use the data object
6. });

25) What is an asynchronous API?

All the API's of Node.js library are asynchronous means non-blocking. A Node.js based server never waits for an API to return data. The Node.js server moves to the next API after calling it, and a notification mechanism of Events of Node.js responds to the server for the previous API call.

26) How can you avoid callbacks?

To avoid callbacks, you can use any one of the following options:

○ You can use **modularization**. It breaks callbacks into independent functions.

- You can use **promises**.

- You can use **yield** with Generators and Promises.

---

27) Does Node.js provide Debugger?

Yes, Node.js provides a simple TCP based protocol and built-in debugging client. For debugging your JavaScript file, you can use debug argument followed by the js file name you want to debug.

Syntax:

1. node debug [script.js | -e "script" | **<host>**:**<port>**]

---

28) What is a control flow function?

Control flow function is a generic piece of code that runs in between several asynchronous function calls.

---

29) How "Control Flow" controls the functions calls?

The control flow does the following job:

- Control the order of execution

- Collect data

- Limit concurrency

- Call the next step in a program

---

30) Is it possible to access DOM in Node?

No, it is not possible to access DOM in Node.

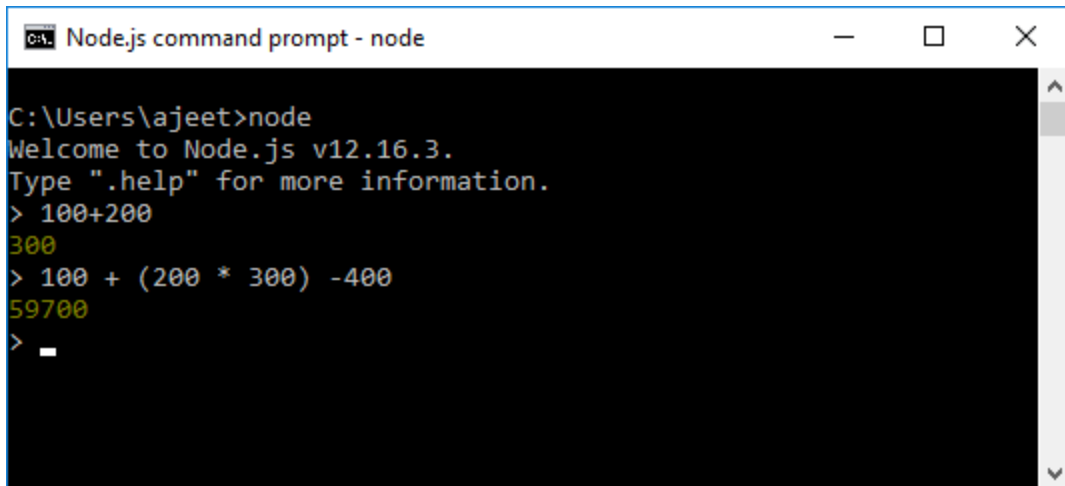## 31) What types of tasks can be done asynchronously using the event loop?

- ○ I/O operations
- ○ Heavy computation
- ○ Anything requiring blocking

---

32) What is REPL in Node.js?

REPL stands for Read Eval Print Loop. It specifies a computer environment like a window console or Unix/Linux shell where you can enter a command, and the computer responds with an output. It is very useful in writing and debugging the codes. REPL environment incorporates Node.js.

See the Example:

1. $ node
2. > 100 + 200
3. 300
4. > 100 + (200 * 300) - 400
5. 59700
6. >

```
C:\Users\ajeet>node
Welcome to Node.js v12.16.3.
Type ".help" for more information.
> 100+200
300
> 100 + (200 * 300) -400
59700
> ▄
```

33) Explain the tasks of terms used in Node REPL.

Following are the terms used in REPL with their defined tasks:

Read: It reads user's input; parse the input into JavaScript data-structure and stores in memory.

Eval: It takes and evaluates the data structure.

Print: It is used to print the result.

Loop: It loops the above command until user press ctrl-c twice to terminate.

34) Is it possible to evaluate simple expressions using Node REPL?

Yes. You can evaluate simple expressions using Node REPL.

35) What is the use of the underscore variable in REPL?

In REPL, the underscore variable is used to get the last result.

1. C:\Nodejs_WorkSpace>node

2. **>** var x = 10
3. undefined
4. **>** var y = 20
5. undefined
6. **>** x + y
7. 30
8. **>** var sum = _
9. undefined
10. **>** console.log(sum)
11. 30
12. undefined
13. **>**

---

36) Does Node.js supports cryptography?

Yes, Node.js Crypto module supports cryptography. It provides cryptographic functionality that includes a set of wrappers for open SSL's hash HMAC, cipher, decipher, sign and verify functions. For example:

1. const crypto = require('crypto');
2. const secret = 'abcdefg';
3. const hash = crypto.createHmac('sha256', secret)
4.   .update('Welcome to JavaTpoint')
5.   .digest('hex');
6. console.log(hash);

---

37) What is npm? What is the main functionality of npm?

npm stands for Node Package Manager. Following are the two main functionalities of npm:

- Online repositories for node.js packages/modules which are searchable on search.nodejs.org

- Command line utility to install packages, do version management and dependency management of Node.js packages.

---

38) What tools can be used to assure a consistent style in Node.js?

Following is a list of tools that can be used in developing code in teams, to enforce a given style guide and to catch common errors using static analysis.

- JSLint

- JSHint

- ESLint

- JSCS

---

39) What is the difference between operational and programmer errors?

Operational errors are not bugs, but create problems with the system like request timeout or hardware failure. On the other hand, programmer errors are actual bugs.

---

40) What is the difference between the global installation of dependencies and local installation of dependencies?

- Global installation of dependencies is stored in /npm directory. While local installation of dependencies stores in the local mode. Here local mode refers to the package installation in node_modules directory lying in the folder where Node application is present.

- Globally deployed packages cannot be imported using require() in Node application directly. On the other hand, locally deployed packages are accessible via require().

- To install a Node project globally -g flag is used.
    1. C:\Nodejs_WorkSpace>npm install express ?g

- To install a Node project locally, the syntax is:
    1. C:\Nodejs_WorkSpace>npm install express

---

41) What is the use of a buffer class in Node.js?

The Node.js provides Buffer class to store raw data similar to an array of integers but corresponds to a raw memory allocation outside the V8 heap. It is a global class and can be accessed in an application without importing a buffer module. Buffer class is used because pure JavaScript is not compatible with binary data. So, when dealing with TCP streams or the file system, it's necessary to handle octet streams.

---

42) What is the role of assert in Node.js?

The Node.js Assert is a way to write tests. It provides no feedback when running your test unless one fails. The assert module provides a simple set of assertion tests that can be used to test invariants. The module is intended for internal use by Node.js, but can be used in application code via require ('assert'). For example:

1. var assert = require('assert');
2. function add (a, b) {

3.   return a + b;

4. }

5. var expected = add(1,2);

6. assert(expected === 3, 'one plus two is three');

---

43) What are the streams in Node.js?

The Streams are the objects that facilitate you to read data from a source and write data to a destination. There are four types of streams in Node.js:

- **Readable:** This stream is used for reading operations.

- **Writable:** This stream is used for write operations.

- **Duplex:** This stream can be used for both reading and write operations.

- **Transform:** It is a type of duplex stream where the output computes according to input.

---

44) What is event-driven programming in Node.js?

In Node.js, event-driven programming means as soon as Node starts its server, it initiates its variables, declares functions and then waits for an event to occur. It is one of the reasons why Node.js is pretty fast compared to other similar technologies.
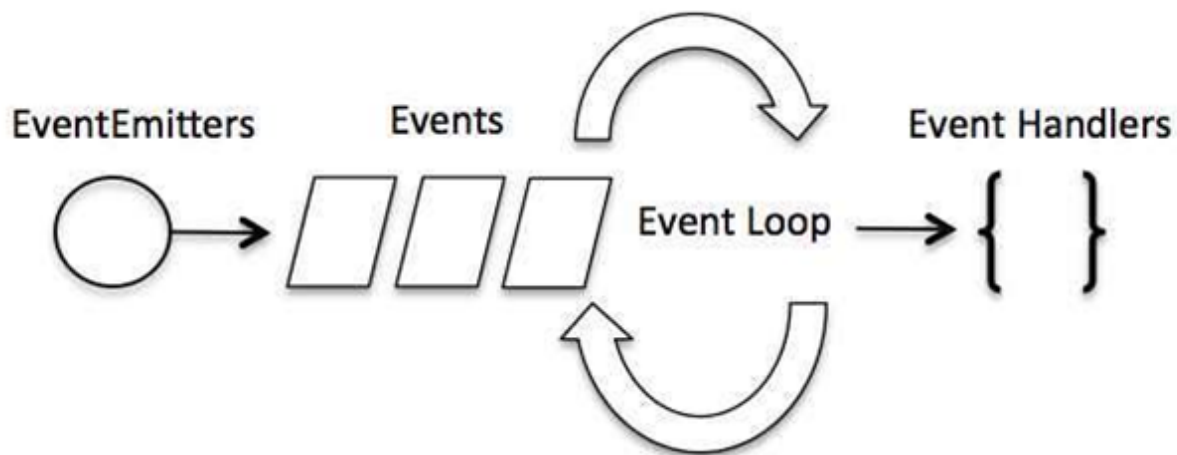
45) What is the difference between events and callbacks in Node.js?

Although, Events and Callbacks look similar the differences lies in the fact that callback functions are called when an asynchronous function returns its result whereas event handling works on the observer pattern. Whenever an event gets fired, its listener function starts executing. Node.js has multiple in-built events available through the events module and EventEmitter class which is used to bind events and event listeners.

46) What is the Punycode in Node.js?

The Punycode is an encoding syntax which is used to convert Unicode (UTF-8) string of characters to ASCII string of characters. It is bundled with Node.js v0.6.2 and later versions. If you want to use it with other Node.js versions, then use npm to install Punycode module first. You have to used require ('Punycode') to access it.

Syntax:

1. punycode = require('punycode');

47) What does Node.js TTY module contains?

The Node.js TTY module contains tty.ReadStream and tty.WriteStream classes. In most cases, there is no need to use this module directly. You have to used require ('tty') to access this module.

Syntax:

1. var tty = require('tty');

---

48) What are the key differences between Angular and Node.js?

Key differences between Angular and Node.js:

Angular	Node.js
Angular is a structural front-end development framework for developing dynamic web apps.	Node.js is a cross-platform, run-time, server-side environment for applications written in JavaScript language.
Angular is entirely written in TypeScript language.	Node.js is written in C, C++, and JavaScript languages.
Angular is used for building single-page, client-side web applications.	Node.js is used for building fast and scalable, client-side, and server-side networking applications.
Angular is easy to use. The developers need to add the Angular file to use it in their applications.	Node.js is slightly complicated to use. Here, the developers need to install Node.js on their computer system.

Angular split a web application into MVC components. Here, the models and views are much simpler than what is found in other JavaScript client-side frameworks.	Node.js generates database queries and uses the event-driven nature of JavaScript to support non-blocking operations, making the platform efficient.
Angular is based on the model-view-controller design pattern and follows that pattern completely.	Node.js is single-threaded. It means the web requests and processing runs on the same thread.
Angular is a Web Framework.	Node.js provides different Web Frameworks like Socket.io, Hapi.js, Meteor.js, Express.js, and Sails.js, etc.
Angular is ideal for creating highly active and interactive web apps.	Node.js is the best for developing small-size projects.
Angular requires a deep understanding of prototyping, scope, and various other JavaScript aspects.	Node.js facilitates developers to use JavaScript on the client as well as the server-side. So, they can focus on learning one language.

49) What are the main differences between operational and programmer errors?

The most crucial difference between operational and programmer errors is that the operational errors are not bugs but problems with the system such as to request timeout or hardware failure. On the other hand, the programmer errors are actual bugs in the application.

50) What do you understand by an EventEmitter in Node.js?

In Node.js, an EventEmitter is a class that includes all the objects capable of emitting events. This can be achieved by attaching named events that are emitted by the object using an eventEmitter.on() function. Thus whenever this object throws an event, the attached functions are invoked synchronously.

Example:

1. const EventEmitter = require('events');
2. class MyEmitter extends EventEmitter {}
3. const myEmitter = new MyEmitter();
4. myEmitter.on('event', () => {
5.   console.log('an event occurred!');
6. });
7. myEmitter.emit('event');

51) What is the difference between readFile and createReadStream in Node.js?

In Node.js, there are two ways to read and execute files: readFile and CreateStream.

○ The readFile() process is a fully buffered process that returns the response only when the complete file is pushed into the buffer and is read. This process is called a memory-intensive process, and in the case of large files, the processing can be very slow.

○ On the other hand, the createReadStream() is a partially buffered process that treats the entire process as an event series. The entire file is split into chunks and then processed and sent back as a response one by one. After completing this

step, they are finally removed from the buffer. Unlike the readFile process, the createReadStream process is effective for the processing of large files.

---

52) What is the concept of Punycode in Node.js?

In Node.js, the concept of Punycode is used for converting one type of string into another type. Punycode is an encoding syntax used for converting Unicode (UTF-8) string of characters into a basic ASCII string of characters. Now, the hostnames can only understand the ASCII characters so, after the Node.js version 0.6.2 onwards, it was bundled up with the default Node package.

To use it with any previous versions, you have to use the following code:

Syntax:

1. punycode = require('punycode');

---

53) How can you enhance the Node.js performance through clustering?

Just because the Node.js applications run on a single processor, they don't take advantage of a multiple-core system by default. Clustering is used to overcome this issue. The cluster mode is used to start up multiple node.js processes, thereby having multiple instances of the event loop. When we start using clusters in a Node.js app, it creates multiple node.js processes. But there is also a parent process called the cluster manager, which is responsible for monitoring the health of the individual instances of the application.

---
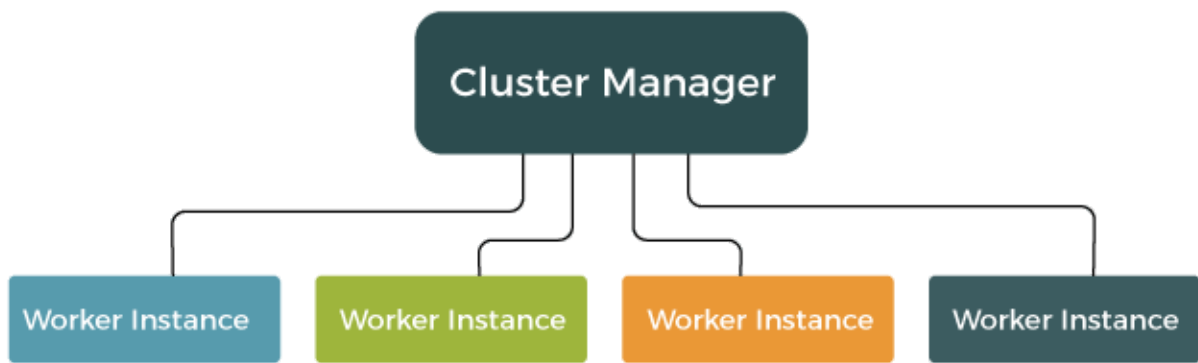
54) What is a thread pool in Node.js? Which library handles it?

In Node.js, the libuv library is used to handle the Thread pool. The libuv library is a multi-platform C library that supports asynchronous I/O-based operations such as file systems, networking, and concurrency.