By OnlineInterviewQuestions.com

Node Js Interview Questions for Beginners

What is Node Js?

Node Js is one of the most popular and powerful server technologies today. It allows you to built the entire website only in one programming Language i.e Javascript. Node js is a free and open-source server technology that uses Javascript to create complete web software. It runs on various platforms like Windows, Linux, Unix, Mac OS X, etc.

Q1. What is Node js ?

Node Js is one of the most popular and powerful server technologies today.
It allows you built the entire website only in one programming Language i.e Javascript. Node js is free and open source server technology that uses Javascript to create complete web software.It runs on various platforms like Windows, Linux, Unix, Mac OS X, etc.

Q2. Explain CLI in Node.js?

CLI stands for Command Line Interface. It is a utility or program on your computer where users type commands to perform some action or run some script rather than clicking on the screen.
There are different types of command line interfaces depending on which operating system you are using. We have listed some of them below.

- Bash on Linux.
- Terminal of Mac.
- Command Prompt or Powershell on Windows
- Shell/ Command line/terminal on Unix and Ubuntu

Q3. In which Language Node Js is written ?

Node js is written in C, C++,JavaScript.It uses Google's open source V8 Javascript Engine to convert Javascript code to C++.

Q4. Who is the author of Node Js ?

Node Js is written by Ryan Dahl. He is an American software developer, born in San Diego, California, the U.S.

in 1981. Besides Node.js he has developed the JavaScript runtime, Deno Javascript, and TypeScript runtime.

Q5. Explain What is a Javascript Engine ?

A Javascript Engine is a program that converts code written in Javascript to something that computer processor understands.

Q6. Explain V8 Engine ?

V8 is Google's open source high-performance JavaScript engine, written in C++ and used in Google Chrome, the open source browser from Google, and in Node.js, among others. It implements ECMAScript as specified in ECMA-262, and runs on Windows 7 or later, macOS 10.5+, and Linux systems that use IA-32, ARM, or MIPS processors. V8 can run standalone or can be embedded into any C++ application.
Source: https://developers.google.com/v8/

Q7. Explain ECMAScript ?

ECMAScript is the standard on which Javascript is based on. It was created to standardize Javascript. It is commonly used for client-side scripting on the World Wide Web and used by Node Js for writing server applications and services.

Q8. How can you check the installed version of Node Js ?

Use `node -v` command to check the installed version of Node Js.

Q9. Explain What is NPM ?

NPM stands for node package manager. It is default Package Manager for JavaScript programming language. NPM is used for installing/updating packages and modules of Javascript.

Q10. Explain Modules in Node Js ?

Modules are reusable block of code whose existence does not impact other code in any way. It is not supported by Javascript. Modules are introduced in ES6. Modules are important for Maintainability, Reusability, and Namespacing of Code.

Q11. What are CommonJs Modules ?

CommonJS Modules is the Standard how to code modules are structured. It specifies an ecosystem for JavaScript outside on the server or for native desktop applications.

Q12. For what require() is used in Node Js ?

require() is used to include modules from external files in Node Js. It is the easiest way to include a module in Node. Basically require is a function that takes a string parameter which contains the location of the file that you want to include. It reads the entire javascript file, executes the file, and then proceeds to return the exports object.
Syntax:

```
require('path');
```

Q13. Explain module.exports in Node Js ?

The method or variable defined in modules cannot be directly accessible by the outer world, that means you cannot call a module member from the external file. In order to access module member, we need to export the functions or variables of modules using module.exports method.
Syntax and usage:

```
// greet.js
var greet=function(){
console.log("hello World");
}
module.exports=greet;
//In app.js
var greet=require('./greet.js');
greet();
```

Q14. Is Node Js Single-threaded ?

Yes, Node Js is single threaded to perform asynchronous processing. Doing async processing on a single thread could provide more performance and scalability under typical web loads than the typical thread-based implementation.

Q15. What are events ?

An event is an action or occurrence recognized by software/app that is handled by event handler by writing a code that will be executed when the event fired.
Mouse move, Click, file copied or deleted are some examples of events.
In Node Js there are two types of events.
1)System Events: The event that comes from the C++ side.
2)Custom Events: Custom events are user-defined events.

Q16. Explain event loop in Node Js ?

In Node Js processes are single threaded, to supports concurrency it uses events and callbacks. An event loop is a mechanism that allows Node.js to perform non-blocking I/O operations.

Q17. How to create a simple server in Node js that returns Hello World ?

By writing following line of code, you can **create a server in Node Js**.

```
var http =require('http');
http.createServer(function(req,res){
res.writeHead(200,{'Content-Type':'text/plain'});
res.end('Hello World\n');
}).listen(1320,'127.0.0.3');
```

Q18. Difference between cluster and child_process modules?

A **Cluster** is a module of Node.js that contains sets of functions and properties that helps the developers for forking processes through which they can take advantage of the multi-core system.

A **child_process** can be easily spun using Node's child_process module and these child processes can easily communicate with each other with the help of a messaging system.

Q19. How to stop master process without suspending all of its child processes?

With the help of the **Upstart process management system**, you can stop the master process without suspending all of its child processes.

Q20. What does emitter do and what is dispatcher?

An **Emitter** class can be used to raise and handle custom events. It facilitates interaction between objects in Node.

A **Dispatcher** is a service object that is used to ensure that the Event is passed to all relevant Listeners.

Q21. Since node is a single threaded process, how to make use of all CPUs?

You can use the cluster module. Node Js is supporting clustering to take full advantage of your CPU.

Q22. List some features of Express JS.

Some of the main features of Express JS are listed below: –

- It is used for setting up middlewares so as to provide a response to the HTTP or RESTful requests.
- With the help of express JS, the routing table can be defined for performing various HTTP operations.
- It is also used for dynamically rendering HTML pages which are based on passing arguments to the templates.
- It provides each and every feature which is provided by core Node JS.
- The performance of Express JS is adequate due to the presence of a thin layer prepared by the Express JS.
- It is used for organizing the web applications into the MVC architecture.
- Everything from routes to rendering view and performing HTTP requests can be managed by Express JS.

Q23. Write the steps for setting up an Express JS application.

Following are the steps used to set up an express JS application: –

1. A folder with the same name as the project name is created.
2. A file named package.json is created inside the folder created.
3. “npm install” command is run on the command prompt. It installs all the libraries present in package.json.
4. A file named server.js is created.
5. “Router” file is created inside the package which consists of a folder named index.js.
6. “App” is created inside the package which has the index.html file.

This way, an express JS application is set up.

Q24. What do you mean by Express JS?

Express JS is an application framework which is light-weighted node JS. A number of flexible, useful and important features are provided by this JavaScript framework for the development of mobile as well as web applications with the help of node JS.

Q25. Name the type of web applications which can be built using Express JS.

Single-page, multi-page, and hybrid web applications can be built using Express JS.

Q26. What is the use of Express JS?

Express.js is a lightweight web application which helps in organizing the web application into MVC architecture on the server side.

Q27. What function are arguments available to Express JS route handlers?

The arguments which are available to an Express JS route handler-function are-

- Req – the request object
- Res – the response object
- Next (optional) – a function which is used to pass control to one of the subsequent route handlers.

The third argument is optional and may be omitted, but in some cases, it is useful where there is a chain of handlers and control can be passed to one of the subsequent route handlers skipping the current one.

Q28. How to config properties in Express JS?

In Express JS, there are two ways for configuring the properties:

- With process.ENV:
- A file with the name ".env" is to be created inside the project folder.
- All the properties are to be added in the ".env" file.
- Any of the properties can be used in server.js.
- With require JS:
- A file with the name "config.json" is to be created in the config folder inside the project folder.
- The config properties are to be added in the config.json file.
- Now, require should be used to access the config.json file.

Q29. How can models be defined in Express JS?

There is no notion of any database in Express JS. So, the concept of models is left up to third-party node modules, allowing the users to interface with nearly any type of database.

Q30. How to authenticate users in express JS?

Since authentication is an opinionated area which is not ventured by express JS, therefore any authentication scheme can be used in express JS for the authentication of users.

Q31. Which template engine is supported by express JS?

Express JS supports any template engine that conforms to the (path, locals, callback) signature.

Q32. How can plain HTML be rendered in express JS?

There’s no need to render HTML with the res.render () function. If there’s a specific file, then you should use the res.sendFile () function. If any assets are being served from a dictionary, then express.static () middleware function needs to be used.

Q33. Why to use Express.js?

Below are the few reasons why to use Express with Node.js

- Express js is built on top of Node.js. It is the perfect framework for ultra-fast Input / Output.
- Cross Platform
- Support MVC Design pattern
- Support of NoSQL databases out of the box.
- Multiple templating engine support i.e. Jade or EJS which reduces the amount of HTML code you have to write for a page.
- Support Middleware, basic web-server creation, and easy routing tools.

Q34. Explain the difference between readFile and createReadStream in Node js ?

- readFile load the whole file which you had marked to read whereas createReadStream reads the complete file in the parts of the size you have declared.
- The client will receive the data faster in the case of createReadStream in contrast with readFile.
- In readFile, a file will first completely read by memory and then transfers to a client but in later option, a file will be read by memory in a part which is sent to clients and the process continue until all the parts finish.

Q35. List types of Http requests?

Http defines a set of request methods to perform the desired actions. These request methods are:

1. **GET**: The GET method asked for the representation of the specifies resource. This request used to retrieve the data.
2. **POST**: The POST technique is utilized to present an element to the predetermined resource, generally causing a change in state or reactions on the server.
3. **HEAD**: The HEAD method is similar to the GET method but asks for the response without the response body.
4. **PUT**: This method is used to substitute all current representations with the payload.
5. **DELETE**: It is used to delete the predetermined resource.
6. **CONNECT**: This request is used to settle the TCP/IP tunnel to the server by the target resource
7. **OPTION**: This method is used to give back the HTTP strategies to communicate with the target resource.
8. **TRACE**: This method echoes the message which tells the customer how much progressions have been

made by an intermediate server.

9. **PATCH**: The PATCH method gives partial modifications to a resource.

Q36. What is difference between put and patch?

The main difference between the put and the patch is

Put	Patch
The embedded entity is believed to the modified version of the resources that are deposited on the original server. It is requested to the client to replace the stored is substituted.	In this, the information regarding the way of modifying the original server which has the resources to produce a new version is found.
At the time of updating the resource, you need to forward full payload as the request.	At the time of updating the resource, you only need to send the parameter of the resource which you want to update.

Q37. How can you set default node version using nvm?

Run below command on the terminal to set default node version along multiple installed versions of node. You can list all install versions of the node by running `nvm ls`

```
nvm alias default v7.3.0
```

Q38. How to generate unique UUIDs/ guid in Node Js

Use node-uuid package to generate unique UUIDs/ guid in Node Js. Below code demonstrates how to generate it.

```
var uuid = require('node-uuid');
// Generate a v1 (time-based) id
uuid.v1();
// Generate a v4 (random) id
uuid.v4();
```

Q39. Write a simple code to enable CORS in Node js?

CORS stands for Cross-Origin Resource Sharing. It a is a mechanism that uses additional HTTP headers to tell a browser to let a web application running at one origin (domain) have permission to access selected resources from a server at a different origin.

Use below code to enable CORS on NodeJS

```
app.use(function(req, res, next) {
  res.header("Access-Control-Allow-Origin", "*");
```

```
  res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-T
  next();
});
```

Q40. List the types of application you can build using Node Js ?

Using Node Js you can build applications like:

- Internet of Things
- Real-Time Chats Applications
- Complex Single-Page Applications
- Real-Time Collaboration Tools
- Streaming apps
- Microservices / API's

Q41. Explain what is libuv in Node Js ?

libuv is Cross-platform I/O abstraction library that supports asynchronous I/O based on event loops.It is written in C and released under MIT Licence.

libuv support Windows IOCP, epoll(4), kqueue(2), and Solaris event ports. Initially, it was designed for Node.js but later it is also used by other software projects.

Reference : https://en.wikipedia.org/wiki/Libuv

Q42. Why Zlib is used in Node js ?

Zlib is Cross-platform data compression library. It was written by Jean-loup Gailly and Mark Adler. In Node js, you can Zlib for Threadpool, HTTP requests, and responses compression and Memory Usage Tuning. In order to use zlib in node js, you need to install node-zlib package. After installation below is sample code to use Zlib.

```
var Buffer = require('buffer').Buffer;
var zlib = require('zlib');
var input = new Buffer('lorem ipsum dolor sit amet');
var compressed = zlib.deflate(input);
var output = zlib.inflate(compressed);
```

Further Reading https://nodejs.org/api/zlib.html

https://en.wikipedia.org/wiki/Zlib

Q43. Write a program to Print 0 to N element in pyramid shape?

The program to Print 0 to N element in pyramid shape:

```
function generatePyramid() {
var totalNumberofRows = 5;
var arr = new Array();
for (var i = 1; i <= totalNumberofRows; i++) {
for (var j = 1; j <= i; j++) {
arr.push(j);
console.log(j);
}
console.log("\n");
}
}
```

Q44. List out some new features introduced in ES6?

Following are the list of few new Features introduced in ES6

- const and let keywords
- Array helper functions like map, forEach, filter, find, every, some, reduce
- Arrow functions
- Classes and enhanced object literals
- Template strings
- Default function arguments
- Rest and spread operators
- Promises
- Modules
- Multi-line Strings
- Destructuring Assignment

Q45. What is JIT and how is it related to Node JS ?

JIT stands for Just-in-time. A JIT compiler is a program which is used to send bytecode (it consists of instruction that can be interpreted) to the processor by converting it into instruction. After you have done with writing a program, the compiler compiles the source language statements into bytecode instead of compiling it into the code that carries the information which is similar to the specific hardware platform's processor.

Relation of JIT with Node: Virtual machine of Nodejs has JIT compilation which improves the execution speed of the code. The virtual machine takes the source code and converts to machine code in runtime. By this, the hot functions which are called very often are compiled to machine code and, hence increasing speed.

Q46. How to use aggregation in Mongoose?

Aggregations are a set of functions that are used to manipulate the data that has been returned from a MongoDB query. In Mongoose, aggregations work as a pipeline. The aggregate function accepts the array of data transformations which are applied by data using different methods in terms of arguments.

Syntax: db.customers.aggregate([... aggregation steps go here ...]);

In above, aggregation is applied to data of customers.

Q47. How Node js read the content of a file?

Normally NodeJs reads the content of a file in non-blocking, asynchronous way. Node Js uses its **fs** core API to deal with files. The easiest way to read the entire content of a file in nodeJs is with fs.readFile method. Below is sample code to read a file in NodeJs asynchronously and synchronously.

Reading a file in node asynchronously/ non-blocking

```
var fs = require('fs');
fs.readFile('DATA', 'utf8', function(err, contents) {
    console.log(contents);
});
console.log('after calling readFile');
```

Reading a file in node asynchronously/blocking

```
var fs = require('fs');
var contents = fs.readFileSync('DATA', 'utf8');
console.log(contents);
```

Q48. How Promises are better than callbacks?

Promises are considered to be better than callbacks in Node.js for several reasons. these are:

Better error handling: Promises provide a more structured way to handle errors, making it easier to identify and fix bugs in your code. With callbacks, errors are often passed as the first argument, which can be easy to overlook or mishandle.

Improved readability: Promises make it easier to read and understand the flow of asynchronous code. With callbacks, it can be difficult to follow the execution order of multiple nested callbacks, leading to callback hell.

Simplified control flows: Promises allow you to chain multiple asynchronous operations together, making it easier to control the flow of your code. With callbacks, you often have to nest them, which can lead to complex and hard-to-maintain code.

Support for async/await: Promises are the foundation of the new async/await syntax, which makes it easy to write asynchronous code that looks like synchronous code.

Better composability: Promises can be composed together in a more natural way, making it easier to build complex asynchronous code.

Q49. Describe Node.js event loop and event driver architecture?

Event Loop. js is asynchronous and single-threaded, they use asynchronous function calls to maintain concurrency.

Q50. What are Streams? List types of streams available in Node Js ?

Streams are special types of objects in Node that allow us to read data from a source or write data to a destination continuously. There are 4 types of streams available in Node Js, they are

- **Readable** ? For reading operation.
- **Writable** ? For writing operation.
- **Duplex** ? Used for both reading and write operation.
- **Transform** ? A type of duplex stream where the output is computed based on the input.

Q51. What is difference between return and callback in JavaScript functions?

Return statements are used to indicates the end of a given function's execution whereas **callbacks** are used to indicate the desired end of a given function's execution.

Q52. How does Promise and Queue work?

The **Promise** constructor takes an input as a function that will be executed immediately and then it will be passed in two functions resolve and reject.

A **queue** is a data structure used in Node.js that is used to appropriately organize asynchronous operations such as HTTP requests, read or write file operations, streams, and more.

Q53. What's the first argument passed to a Node Js callback handler?

In Node Js all core modules, as well as most of the community-published modules, follows a pattern where the first argument to any callback handler is an error object. this object is optional, if there is no error then in that case null or undefined is passed to the callback.

Example of the callback function

```
function callback(err, results) {
 // usually we'll check for the error before handling results
 if(err) {
  // handle error somehow and return
 }
  // no error, perform standard callback handling
}
```

Q54. What do you understand by middleware? How can you use middleware in Node Js?

The **middleware function** is a function that is called before the route handler and has access to the request object, the response object, and the next ().

Q55. Explain the difference between process.tick() and setImmediate() ?

setImmediate() method is used to invoke its callback function is placed in the check phase whereas setImmediate() method is called in the poll phase.

The **process.nextTick()** method is used to add the callback function at the start of the next event queue, it is called for the first time before the event loop is processed. process.nextTick(callback);

Q56. What is the revealing module pattern?

Revealing module pattern is similar to Module Pattern.IT ExposES only the properties and methods you want via an returned Object.

```
var greeting = 'Hello world'
function greet() {
 console.log(greeting)
}
module.exports = {
  greet: greet
}
```

Q57. what is Closures?

A Closure is a function defined within another scope that has access to all the variables within the outer scope.

Global variables can be made local (private) with closures.