Top 75 NodeJs Interview Questions and Answers

1. What is NodeJS?

Node.js is an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside of a browser. Node JS was created by Ryan Dahl, Ryan Dahl is a software engineer and the original developer of the Node.js JavaScript runtime.

2. How can you avoid callback hells?

There are lots of ways to solve the issue of callback hells:

1. modularization: break callbacks into independent functions,
2. use a control flow library, like async.
3. use generators with Promises,
4. use async/await (note that it is only available in the latest v7 release and not in the LTS version

3. When are background or worker processes useful?

Worker processes are extremely useful if you'd like to do data processing in the background, like sending out emails or processing images.

There are lots of options for this like RabbitMQ or Kafka.

4. Why is NodeJS Single threaded?

Node.js is single-threaded for async processing. By doing async processing on a single-thread under typical web loads, more performance and scalability can be achieved as opposed to the typical thread-based implementation.

5. Name the types of API functions in Node?

There are two types of functions in Node.js.

1. **Blocking functions** - In a blocking operation, all other code is blocked from executing until an I/O event that is being waited on occurs. Blocking functions execute synchronously.
2. **Non-blocking functions** - In a non-blocking operation, multiple I/O calls can be performed without the execution of the program being halted. Non-blocking functions execute asynchronously.

6. Explain chaining in Nodejs?

Chaining is a mechanism whereby the output of one stream is connected to another stream creating a chain of multiple stream operations.

7. What are streams in Nodejs Explain the different types of streams present in Nodejs?

Streams are objects that allow the reading of data from the source and writing of data to the destination as a continuous process.

There are four types of streams.

to facilitate the reading operation.

to facilitate the writing operation.

to facilitate both read and write operations.

is a form of Duplex stream that performs computations based on the available input.

8. What is package json?

The **package.json** file in Node.js is the heart of the entire application. It is basically the manifest file that contains the metadata of the project where we define the properties of a package.

9. Explain the purpose of module exports?

A module in Node.js is used to encapsulate all the related codes into a single unit of code which can be interpreted by shifting all related functions into a single file

10. List down the major security implementations within Nodejs?

Major security implementations in Node.js are: *Authentications*, *Error Handling*

11. Explain the concept of URL module?

The URL module splits up a web address into readable parts

12. What is middleware?

Middleware comes in between your request and business logic. It is mainly used to capture logs and enable rate limit, routing, authentication, basically whatever that is not a part of business logic. There is many third-party middleware also such as body-parser and you can write your own middleware for a specific use case.

13. Explain libuv?

Libuv is a multi-platform support library of Node.js which majorly is used for asynchronous I/O. It was primarily developed for Node.js, with time it is popularly practiced with other systems like as Luvit, pyuv, Julia, etc. Libuv is basically an abstraction around libev/ IOCP depending on the platform, providing users an API based on libev. A few of the important features of libuv are:

- Full-featured event loop backed
- File system events
- Asynchronous file & file system operations
- Asynchronous TCP & UDP sockets
- Child processes

14. List down the two arguments that async.queue takes as input?

Below are the two arguments that async.queue takes as input - Task Function & Concurrency Value

15. Differentiate between spawn and fork methods in Nodejs?

In Node.js, the spawn() is used to launch a new process with the provided set of commands. This method doesn't create a new V8 instance and just one copy of the node module is active on the processor. When your child process returns a large amount of data to the Node you can invoke this method.

16. Explain the purpose of ExpressJS package?

Express.js is a framework built on top of Node.js that facilitates the management of the flow of data between server and routes in the server-side applications. It is a lightweight and flexible framework that provides a wide range of features required for the web as well as mobile application development. Express.js is developed on the middleware module of Node.js called connect. The connect module further makes use of http module to communicate with Node.js. Thus, if you are working with any of the connect based middleware modules, then you can easily integrate with Express.js.

17. Explain the usage of a buffer class in Nodejs?

Buffer class in Node.js is used for storing the raw data in a similar manner of an array of integers. But it corresponds to a raw memory allocation that is located outside the V8 heap. It is a global class that is easily accessible 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.

18. How does Nodejs handle the child threads?

In general, Node.js is a single threaded process and doesn't expose the child threads or thread management methods. But you can still make use of the child threads using spawn() for some specific asynchronous I/O tasks which execute in the background and don’t usually execute any JS code or hinder with the main event loop in the application. If you still want to use the threading concept in your application you have to include a module called ChildProcess explicitly.

19. Explain stream in Nodejs along with its various types?

Streams in Node.js are the collection of data similar to arrays and strings. They are objects using which you can read data from a source or write data to a destination in a continuous manner. It might not be available at once and need not to have fit in the memory. These streams are especially useful for reading and processing a large set of data. In Node.js, there are four fundamental types of streams:

- **Readable**: Used for reading large chunks of data from the source.
- **Writeable**: Use for writing large chunks of data to the destination.
- **Duplex**: Used for both the functions; read and write.
- **Transform**: It is a duplex stream that is used for modifying the data.

20. Describe the exit codes of Nodejs?

In Node.js, exit codes are a set of specific codes which are used for finishing a specific process. These processes can include the global object as well. Below are some of the exit codes used in Node.js:

- Uncaught fatal exception
- Unused
- Fatal Error
- Internal Exception handler Run-time failure
- Internal JavaScript Evaluation Failure

21. Is cryptography supported in Nodejs?

Yes, Node.js does support cryptography through a module called Crypto. This module provides various cryptographic functionalities like cipher, decipher, sign and verify functions, a set of wrappers for open SSL’s hash HMAC etc.

```
const crypto = require('crypto');
const secret = 'akerude';
const hash = crypto.createHmac('swaEdu', secret).update('Wel
come to Edureka').digest('hex');
console.log(hash);
```

22. Explain the reason as to why Express app and server folder must be kept separate?

Express ‘app’ and ‘server’ must be kept separate as by doing this, you will be separating the API declaration from the network related configuration which benefits in the below listed ways:

- It allows testing the API in-process without having to perform the network calls
- Faster testing execution
- Getting wider coverage metrics of the code
- Allows deploying the same API under flexible and different network conditions
- Better separation of concerns and cleaner code

23. What is the role of asset module in nodejs?

The assert module provides a set of assertion functions for verifying invariants

24. What is the role of async_hooks module in nodejs?

The async_hooks module provides an API to track asynchronous resources. It can be accessed using:

```
const async_hooks = require('async_hooks');
```

25. What are buffer objects in nodejs?

In Node.js, Buffer objects are used to represent binary data in the form of a sequence of bytes. Many Node.js APIs, for example streams and file system operations, support Buffers, as interactions with the operating system or other processes generally always happen in terms of binary data

26. What are the different ways of implementing Addons in NodeJS?

There are three options for implementing Addons:

- N-API
- nan direct use of internal V8
- libuv
- Node.js libraries

27. How can we spawn the child process asynchronously without blocking the Nodejs event loop?

- **child_process.spawn()** method spawns the child process asynchronously, without blocking the Node.js event loop, the child_process.
- **spawnSync()** function provides equivalent functionality in a synchronous manner that blocks the event loop until the spawned process either exits or is terminated

28. How can we take advantage of multi-core system in Nodejs as nodejs works on single thread?

We can use node js cluster to use multicores in the hardware, the cluster module allows easy creation of child processes that all share server ports

```
const cluster = require("cluster");
const http = require("http");
const numCPUs = require("os").cpus().length;

if (cluster.isMaster) {
  console.log(`Master ${process.pid} is running`);

  // Fork workers.
  for (let i = 0; i < numCPUs; i++) {
    cluster.fork();
  }

  cluster.on("exit", (worker, code, signal) => {
    console.log(`worker ${worker.process.pid} died`);
  });
} else {
  // Workers can share any TCP connection
  // In this case it is an HTTP server
  http
    .createServer((req, res) => {
      res.writeHead(200);
      res.end("hello world\n");
    })
    .listen(8000);

  console.log(`Worker ${process.pid} started`);
}
//Running Node.js will now share port 8000 between the
workers:
```

$ node server.js

```
Master 3596 is running
Worker 4324 started
Worker 4520 started
Worker 6056 started
Worker 5644 started
```

29. What is the datatype of console?

The datatype of console is an object

30. Which are the different console methods available?

There are around 21 inbuilt console methods, we can also build our own prototypes using new Console constructor function

here are a few popular one's

1. **console.clear()** will clear only the output in the current terminal viewport for the Node.js binary.
2. **console.error([data][, ...args])** Prints to stderr with newline. Multiple arguments can be passed, with the first used as the primary message and all additional used as substitution
3. **console.table(tabularData[, properties])** a table with the columns of the properties of tabularData (or use properties) and rows of tabularData and log it.

31. Can node js perform cryptographic functions?

Yes,The crypto module provides cryptographic functionality that includes a set of wrappers for OpenSSL's hash, HMAC, cipher, decipher, sign, and verify functions.

Use `require('crypto')` to access this module.

32. How can we read or write files in node js?

The fs module provides an API for interacting with the file system in a manner closely modeled around standard POSIX functions. To use this module:

```
const fs = require('fs');
```

There are a few methods like

```
fs.readFile(file, data[, options], callback)
fs.writeFile(file, data[, options], callback)
```

33. Which are the global objects in Node JS?

__dirname

__filename

clearImmediate(immediateObject)

clearInterval(intervalObject)

clearTimeout(timeoutObject)

console

exports

global

module

process

queueMicrotask(callback)

require()

setImmediate(callback[, ...args])

setInterval(callback, delay[, ...args])

setTimeout(callback, delay[, ...args])

TextDecoder

TextEncoder

URL

URLSearchParams

WebAssembly

34. How can we perform asynchronous network API in Node JS?

The net module provides an asynchronous network API for creating stream-based TCP or IPC servers (net.createServer()) and clients (net.createConnection()).

It can be accessed using:

```
const net = require('net');
```

35. What are the utilities of OS module in NodeJS?

The os module provides operating system-related utility methods and properties. It can be accessed using:

```
const os = require('os');
```

36. Which are the areas where it is suitable to use NodeJS?

I/O bound Applications

Data Streaming Applications

Data Intensive Real-time Applications (DIRT)

JSON APIs based Applications

Single Page Applications

37. Which are the areas where it is not suitable to use NodeJS?

it’s not suitable for heavy applications involving more of CPU usage

38. What Are the Key Features of NodeJs?

Asynchronous event driven IO helps concurrent request handling – All APIs of Node.js are asynchronous. This feature means that if a Node receives a request for some Input/Output operation, it will execute that operation in the background and continue with the processing of other requests. Thus, it will not wait for the response from the previous requests.

Fast in Code execution – Node.js uses the V8 JavaScript Runtime engine, the one which is used by Google Chrome. Node has a wrapper over the JavaScript engine which makes the runtime engine much faster and hence processing of requests within Node.js also become faster.

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.

Node.js library uses JavaScript – This is another important aspect of Node.js from the developer’s point of view. The majority of developers are already well-versed in JavaScript. Hence, development in Node.js becomes easier for a developer who knows JavaScript.

There is an Active and vibrant community for the Node.js framework – The active community always keeps the framework updated with the latest trends in the web development.

No Buffering – Node.js applications never buffer any data. They simply output the data in chunks.

39. Explain REPL In NodeJs?

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.

40. Can you write CRUD operations in Node js without using frameworks?

Yes, we can use inbuilt http library for that, here is a simple code for the same:

```
var http = require('http');//create a server object:
http.createServer(function (req, res) {
res.writeHead(200, {'Content-
Type': 'text/html'}); // http headervar url = req.url;
if(url ==='/about'){
  res.write('<h1>about us page<h1>'); //write a response
  res.end(); //end the response
}else if(url ==='/contact'){
  res.write('<h1>contact us page<h1>'); //write a response
  res.end(); //end the response
}else{
  res.write('<h1>Hello World!<h1>'); //write a response
  res.end(); //end the response
}}).listen(3000, function(){
console.log("server start at port 3000"); //the server objec
t listens on port 3000
});
```

41. What Is The Difference Between Nodejs AJAX And JQuery?

The one common trait between Node.js, AJAX, and jQuery is that all of them are the advanced implementation of JavaScript. However, they serve completely different purposes.

Node.Js :

It is a server-side platform for developing client-server applications. For example, if we've to build an online employee management system, then we won't do it using client-side JS. But the Node.js can certainly do it as it runs on a server similar to Apache, Django not in a browser.

AJAX (Aka Asynchronous Javascript And XML) :

It is a client-side scripting technique, primarily designed for rendering the contents of a page without refreshing it. There are a no. of large companies utilizing AJAX such as Facebook and Stack Overflow to display dynamic content.

JQuery :

It is a famous JavaScript module which complements AJAX, DOM traversal, looping and so on. This library provides many useful functions to help in JavaScript development. However, it’s not mandatory to use it but as it also manages cross-browser compatibility, so can help you produce highly maintainable web applications.

42. What Is EventEmitter In NodeJs?

Events module in Node.js allows us to create and handle custom events. The Event module contains “EventEmitter” class which can be used to raise and handle custom events. It is accessible via the following code.

```
// Import events module
var events = require('events');

// Create an eventEmitter object
var eventEmitter = new events.EventEmitter();
```

When an EventEmitter instance encounters an error, it emits an “error” event. When a new listener gets added, it fires a “newListener” event and when a listener gets removed, it fires a “removeListener” event.

EventEmitter provides multiple properties like “on” and “emit”. The “on” property is used to bind a function to the event and “emit” is used to fire an event. .

43. What Is A Child_process Module In NodeJs?

Node.js supports the creation of child processes to help in parallel processing along with the event-driven model.

The Child processes always have three streams <child.stdin>, child.stdout, and child.stderr. The stream of the parent process shares the streams of the child process.

Node.js provides a <child_process> module which supports following three methods to create a child process.

EXEC – <CHILD_PROCESS.EXEC> METHOD RUNS A COMMAND IN A SHELL/CONSOLE AND BUFFERS THE OUTPUT.

SPAWN – <CHILD_PROCESS.SPAWN> LAUNCHES A NEW PROCESS WITH A GIVEN COMMAND.

FORK – <CHILD_PROCESS.FORK> IS A SPECIAL CASE OF THE SPAWN() METHOD TO CREATE CHILD PROCESSES.

44. What do you mean by Asynchronous API?

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.

45. What are the benefits of using Node.js?

Following are main benefits of using Node.js

- **Asynchronous and Event Driven** All APIs of Node.js library are asynchronous 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.
- **Very Fast** Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
- **Single Threaded but highly Scalable** − Node.js uses a single threaded model with event looping. Event mechanism helps server to respond in a non-bloking ways and makes server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and same program can services much larger number of requests than traditional server like Apache HTTP Server.
- **No Buffering** − Node.js applications never buffer any data. These applications simply output the data in chunks.

46. Is Node a single threaded application?

Yes! Node uses a single threaded model with event looping.

47. What is global installation of dependencies?

Globally installed packages/dependencies are stored in <user-directory>/npm directory. Such dependencies can be used in CLI (Command Line Interface) function of any node.js but cannot be imported using require() in Node application directly. To install a Node project globally use -g flag.

```
npm install express -g
```

48. What is local installation of dependencies?

By default, npm installs any dependency 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. Locally deployed packages are accessible via require(). To install a Node project locally following is the syntax.

```
npm install express
```

49. How to check the already installed dependencies which are globally installed using npm?

```
npm ls -g
```

50. Name some of the attributes of package.json?

Following are the attributes of Package.json

- **name** – name of the package
- **version** – version of the package
- **description** – description of the package
- **homepage** – homepage of the package
- **author** – author of the package
- **contributors** – name of the contributors to the package
- **dependencies** – list of dependencies. npm automatically installs all the dependencies mentioned here in the node_module folder of the package.
- **repository** – repository type and url of the package
- **main** – entry point of the package
- **keywords** – keywords

51. How to uninstall a dependency using npm?

Use following command to uninstall a module.

```
npm uninstall dependency-name
```

52. How to update a dependency using npm?

Update package.json and change the version of the dependency which to be updated and run the following command.

```
npm update
```

53. What is Callback?

Callback is an asynchronous equivalent for a function. A Callback function is called at the completion of a given task. Node makes heavy use of callbacks. All APIs of Node are written is such a way that they support callbacks. For example, a function to read a file may start reading file and return the control to execution environment immidiately so that next instruction can be executed. Once file I/O is complete, it will call the callback function while passing the callback function, the content of the file as parameter. So, there is no blocking or wait for File I/O. This makes Node.js highly scalable, as it can process high number of requests without waiting for any function to return result.

54. What is a blocking code?

If application has to wait for some I/O operation in order to complete its execution any further then the code responsible for waiting is known as blocking code.

55. How Node prevents blocking code?

By providing callback function. Callback function gets called whenever corresponding event triggered.

56. What is Event Loop?

Node js is a single threaded application but it support concurrency via concept of event and callbacks. As every API of Node js are asynchronous and being a single thread, it uses async function calls to maintain the concurrency. Node uses observer pattern. Node thread keeps an event loop and whenever any task get completed, it fires the corresponding event which signals the event listener function to get executed.

57. What is Event Emitter?

EventEmitter class lies in events module. It is accessible via following syntax –

```
//import events module
var events = require('events');

//create an eventEmitter object
var eventEmitter = new events.EventEmitter();
```

When an EventEmitter instance faces any error, it emits an 'error' event. When new listener is added, 'newListener' event is fired and when a listener is removed, 'removeListener' event is fired.

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.

58. What is purpose of Buffer class in Node?

Buffer class is a global class and can be accessed in application without importing buffer module. A Buffer is a kind of an array of integers and corresponds to a raw memory allocation outside the V8 heap. A Buffer cannot be resized.

59. What is Piping in Node?

Piping is a mechanism to connect output of one stream to another stream. It is normally used to get data from one stream and to pass output of that stream to another stream. There is no limit on piping operations. Consider the above example, where we've read test.txt using readerStream and write test1.txt using writerStream. Now we'll use the piping to simplify our operation or reading from one file and writing to another file.

60. Which module is used for buffer-based operations?

buffer module is used for buffer-based operations.

```
var buffer = require("buffer")
```

61. What is difference between synchronous and asynchronous method of fs module?

Every method in fs module has synchronous as well as asynchronous form. Asynchronous methods take a last parameter as completion function callback and first parameter of the callback function is error. It is preferred to use asynchronous method instead of synchronous method as former never block the program execution where the latter one does.

62. What are streams?

Streams are objects that let you read data from a source or write data to a destination in continuous fashion.

63. How many types of streams are present in Node.

In Node.js, there are four types of streams.

- **Readable** – Stream which is used for read operation.
- **Writable** – Stream which is used for write operation.
- **Duplex** – Stream which can be used for both read and write operation.
- **Transform** – A type of duplex stream where the output is computed based on input.

64. Name some of the events fired by streams.

Each type of Stream is an Event Emitter instance and throws several events at different instance of times. For example, some of the commonly used events are:

- **data** – This event is fired when there is data is available to read.
- **end** – This event is fired when there is no more data to read.
- **error** – This event is fired when there is any error receiving or writing data.
- **finish** – This event is fired when all data has been flushed to underlying system.

65. What is a first-class function in JavaScript?

When functions can be treated like any other variable then those functions are first-class functions. There are many other programming languages, for example, scala, Haskell, etc which follow this including JS. Now because of this function can be passed as a param to another function(callback) or a function can return another function(higher-order function). map() and filter() are higher-order functions that are popularly used.

66. What is Node.js and how it works?

Node.js is a virtual machine that uses JavaScript as its scripting language and runs Chrome's V8 JavaScript engine. Basically, Node.js is based on an event-driven architecture where I/O runs asynchronously making it lightweight and efficient. It is being used in developing desktop applications as well with a popular framework called electron as it provides API to access OS-level features such as file system, network, etc.

67. How do you manage packages in your node.js project?

It can be managed by a number of package installers and their configuration file accordingly. Out of them mostly use npm or yarn. Both provide almost all libraries of javascript with extended features of controlling environment-specific configurations. To maintain versions of libs being installed in a project we use package.json and package-lock.json so that there is no issue in porting that app to a different environment.

68. How is Node.js better than other frameworks most popularly used?

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

It runs on a chrome v8 engine which is written in c++ and is highly performant with constant improvement.

Also since we will use Javascript in both the frontend and backend the development will be much faster.

And at last, there are ample libraries so that we don't need to reinvent the wheel.

69. Explain the steps how "Control Flow" controls the functions calls?

1. Control the order of execution
2. Collect data
3. Limit concurrency
4. Call the following step in the program.

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

- **setTimeout/clearTimeout** – This is used to implement delays in code execution.
- **setInterval/clearInterval** – This is used to run a code block multiple times.
- **setImmediate/clearImmediate** – This is used to set the execution of the code at the end of the event loop cycle.
- **process.nextTick** – This is used to set the execution of code at the beginning of the next event loop cycle.

71. What are the advantages of using promises instead of callbacks?

The main advantage of using promise is you get an object to decide the action that needs to be taken after the async task completes. This gives more manageable code and avoids callback hell.

72. What is fork in node JS?

A fork in general is used to spawn child processes. In node it is used to create a new instance of v8 engine to run multiple workers to execute the code.

73. Why is Node.js single-threaded?

Node.js was created explicitly as an experiment in async processing. This was to try a new theory of doing async processing on a single thread over the existing thread-based implementation of scaling via different frameworks.

74. List down the two arguments that async.queue takes as input?

- Task Function
- Concurrency Value

75. How does Node.js overcome the problem of blocking of I/O operations?

Since the node has an event loop that can be used to handle all the I/O operations in an asynchronous manner without blocking the main function.

For example, if some network call needs to happen it will be scheduled in the event loop instead of the main thread (single thread). And if there are multiple such I/O calls each one will be queued accordingly to be executed separately (other than the main thread).

Thus, even though we have single-threaded JS, I/O ops are handled in a nonblocking way.