Top 50

Node.js

Interview Questions & Answers

Knowledge Powerhouse

Top 50 Node.js Interview Questions & Answers

Knowledge Powerhouse

DEDICATION

To our readers!

CONTENTS

4. What are the types of applications that can be built by using Node.js?

5. Why Node.js is based on single threaded architecture?

6. What are the different types of APIs available in Node.js?

7. Can we run Node.js on Windows environment?

8. What is Event Loop in Node.js?

9. What are the disadvantages of Node.js?

10. How can we handle blocking I/O operations in Node.js?

11. What is the difference between

Asynchronous and Non-blocking?

12. What is the difference between Node.js and Ajax?

13. What is the difference between Node.js and AngularJS?

14. How will you import external libraries in Node.js?

15. What happens if we call require() method to load same module multiple times in Node.js?

16. What is REPL in Node.js?

17. What are the popular REPL commands in Node.js?

18. What is NPM in Node.js?

19. Why Node.js application is scalable?

20. What is the purpose of module.exports in Node.js?

21. What is Tracing in Node.js?

22. How will you debug an application in Node.js?

23. What is a Child Process in Node.js?

24. What is a Cluster in Node.js?

25. What is closure in JavaScript?

26. What is a Buffer in Node.js?

27. How will you convert a Buffer to

JSON in Node.js?

28. Why do we use __filename in Node.js?

29. What is the use of Timers is Node.js?

30. What are the important APIs in Timers module in Node.js?

31. What is EventEmitter in Node.js?

32. What is the use of net.Socket in Node.js?

33. What are the important events of net.Socket in Node.js?

34. Can we build a REST service in

Node.js?

35. What is the use of DNS module in Node.js?

36. What are the important command line options in Node.js?

37. How does Node.js work?

38. How can we avoid Callback Hell in Node.js?

39. How do you resolve unhandled exceptions in a Node.js program?

40. What is a Callback function in Node.js?

41. What are the most popular modules of Node.js?

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

43. What is the use of QueryString in Node.js?

44. How will you get the amount of free memory on the server in which Node.js application is running?

45. What is a Global object in Node.js?

46. What are the security mechanisms available in Node.js?

47. What is the use of Zlib in Node.js?

48. How will you convert a Buffer

content into readable String in Node.js?

49. How do you write unit test cases in Node.js?

50. What are the standard Javascript errors?

ACKNOWLEDGMENTS

We thank our readers who constantly send feedback and reviews to

motivate us in creating these useful books with the latest information!

INTRODUCTION

Node.js is one of the most popular Javascript engine in technology world. There is a growing demand for Software Engineer jobs in Node.js.

This book contains technical interview questions that an interviewer asks for Node.js domain. Each question is accompanied with an answer so that you can prepare for job interview in short time.

We have compiled this list after attending dozens of technical interviews in top-notch companies like- Airbnb,

Netflix, Uber etc.

Often, these questions and concepts are used in our daily work.

There is a sample answer with each question. But try to answer these questions in your own words.

After going through this book 2-3 times, you will be well prepared to face Node.js interview for a Software Engineer, Senior Software Engineer and Principal Engineer positions.

Node.js Interview Questions

1. What are the reasons of popularity of Node.js?

Node.js is becoming very popular technology for developing web applications these days. Some of the main reasons of popularity of Node.js are as follows:

1. **Low barrier to entry**: It is very easy to start and run a Node.js application. Developers find it very easy

to use and build application in it.

2. **Big Companies Support**: Google is investing heavily in support of Node.js. This increases confidence level of companies in selecting Node.js as a framework to create applications.

3. **Extremely Fast**: Applications developed in Node.js are extremely fast.

4. **Back End vs. Front End**: One good benefit of Node.js is that we can implement

Front end as well as Back end application in Node.js.

5. **Wide Availability**: Node.js is widely available across the world. This increases its availability.

2. What is Event Driven Programming followed in Node.js?

In Event Driven Programming (EDP), the flow of a software program is based on the events, mostly user actions, like- Mouse clicks, key presses, sensor outputs.

EDP is mainly used in Graphic User Interface (GUI) projects.

In EDP, there is a main loop that keeps running and listening to events. As and when an event is detected, a callback function is triggered. This callback function is related to the event.

3. What are the main advantages of using Node.js?

Some of the main advantages of using Node.js are as follows:

1. **Productivity**: Since Node.js is based on Javascript, we can use same language for front-end as well as back-end. This increases the productivity of development of an application.

2. **Fast:** Node.js is very fast. It

runs on Google's V8 engine that compiles Javascript into machine code and makes it run very fast.

3. **Asynchronous**: Since Node.js is event-driven, we can implement asynchronous processing. This helps in handling a large amount of requests in a short period of time.

4. **Real time applications**: With Node.js we can easily build Real-time applications that require real-time interaction and processing.

E.g. Gaming or Chat applications.

5. **Streaming**: It is very easy to read/write from websockets in Node.js. This makes it very suitable for streaming applications.

4. What are the types of applications that can be built by using Node.js?

Node.js can be used for building a wide-variety of applications. Some of the main types of applications that can be easily built with Node.js are as follows:

1. **Real-time web applications**: We can use Node.js to build

applications for chatting and gaming purpose.

2. **Web applications**: Even we can build web-applications with Node.js.

3. **Payment applications**: We can even use Node.js to build fast payment applications. Even PayPal widely uses Node.js for their system.

4. **Streaming applications**: Node.js provides very good performance for creating and running a streaming

application.

5. Why Node.js is based on single threaded architecture?

Node.js is based on single threaded architecture due to the reason that it runs on Google's V8 engine which is a single threaded architecture.

Over the time, single threaded architecture of Node.js provides much better performance than one thread per request architecture.

6. What are the different types of APIs available in Node.js?

Node.js mainly provides following two types of APIs:

1. **Synchronous API**: These are blocking functions that wait for the response to come back.
2. **Asynchronous API**: These are non-blocking functions that keep on processing in the background.

7. Can we run Node.js on Windows environment?

Yes, it is possible to run Node.js on Windows environment. We have to download windows .msi file to run Node.js applications on Windows.

8. What is Event Loop in Node.js?

Node.js is based on Event Driven programming model. At the heart of Node.js is an Event Loop. This Event Loop listens to events. On each event a call back function is called. This call back function is asynchronous in nature.

When we start Node.js, it internally starts the Event Loop.

9. What are the disadvantages of Node.js?

Although Node.js is getting popular for certain applications, it has some disadvantages as well. Some of these are as follows:

1. **Libraries**: Node.js does not have a large number of libraries as compared with other frameworks like Spring and languages like Java.
2. **Callback**: For some synchronous applications,

event driven callback mechanism is not suitable.

3. **Documentation**: Still it is difficult to find one common place with complete documentation of Node.js.
4. **Talent**: Since Node.js is a new field it is difficult to find and hire talented developers in this area.

10. How can we handle blocking I/O operations in Node.js?

In Node.js we can use asynchronous calls to handle blocking I/O operations. In an asynchronous main thread of our program is not waiting for an I/O operation to finish.

Let say we want to get some information from Database, we just fire and event. Our main program keeps executing the steps after that

event. In the meantime when we receive output from database read, it emits another event. This event leads to further processing on the data retrieved from the database.

Where as our main program keep on executing all the time.

11. What is the difference between Asynchronous and Non-blocking?

Asynchronous and non-blocking look very similar in nature but there is a subtle difference here.

Asynchronous means that our API will return immediately after calling it. In the background it will start a process to fulfill the request. Once that process is complete, our work is done. But our main thread does not wait for

that process to complete.

Non-blocking means if our API cannot complete the work, it returns an error immediately. Based on the response, success of error, we can decide if we want to make another call to same API or continue with the next operation in our main flow. We can create a wait mechanism in a non-blocking operation.

12. What is the difference between Node.js and Ajax?

Both Node.js and Ajax use Javascript in the backend implementation. Due to this, they appear similar in nature. But there are fundamental differences between these two.

Ajax is simply Asynchronous Javascript and XML. It is a client-side technology. It is mainly used for asynchronously handling the page refresh requests. Sites like Facebook, Gmail etc. use Ajax to

implement dynamically self-loading pages. Ajax is executed in client browser.

Node.js is a server-side technology implemented in Javascript. It is used for developing enterprise software. Node.js is not executed in client browser. It executes in server.

13. What is the difference between Node.js and AngularJS?

Both Node.js and AngularJS use Javascript in the implementation. Due to this, they appear similar in nature. But there are fundamental differences between these two.

AngularJS is a web framework that is implemented in Javascript. It is used to build front-end web applications with powerful back-end support. In AngularJS we use

tags to create single page applications. It is based on Model View Controller (MVC) design pattern.

Node.js is a platform to create server-side software applications. It is based on Google's V8 Javascript engine. We can use it to create a highly scalable system with large data usage.

14. How will you import external libraries in Node.js?

We can use require() call to import external libraries in Node.js.

15. What happens if we call require() method to load same module multiple times in Node.js?

Node.js has a built-in caching mechanism to load modules. If we call require() multiple times to load same module, the module will be cached the first time it is loaded. This provides significant performance improvement.

16. What is REPL in Node.js?

REPL stands for Read Eval Print Loop in Node.js. We have to use require('repl') to load this module.

There is a repl.REPLServer class in repl module. This server accepts input from user. This input is evaluated by REPLServer and an appropriate action is taken on that.

In general it supports many functions that are popular in Unix

Command Line Interface. Like- Automatic completion of input, editing in-line, multi-line input, Control C, Control D etc.

17.What are the popular REPL commands in Node.js?

Some of the popular REPL commands in Node.js are as follows:

1. **Ctrl + C**: Terminate current command
2. **Ctrl + D**: Terminate REPL
3. **Ctrl +C twice**: Terminate REPL
4. **Tab**: When Tab is pressed on

a blank line it displays global and local variables. When Tab is pressed while entering input, it displays the relevant auto-completion options.

18. What is NPM in Node.js?

NPM or npm is Node Package Manager in Node.js. There are two main uses of NPM:

1. **Online Repository**: It provides Online repository for node.js packages. We can store and search node.js packages in NPM.

2. **Installation**: It also provides a command line utility to install and uninstall node.js packages. We can do version

management and dependency management with NPM. We can even update the version of a module with NPM.

19. Why Node.js application is scalable?

In a Node.js application we do multiple I/O operations in parallel. Since I/O operations do not share data between threads, we can execute these in parallel. Once I/O is complete, it emits event that can be handled by the Event Loop. Due to handling of multiple I/O in parallel Node.js application can be scaled easily.

Scalability in Node.js is more

applicable to web-services where we have a large amount of I/O operations.

One limitation of Node.js scalability is that we cannot use it for an application with parallel processes that share data.

20. What is the purpose of module.exports in Node.js?

In Node.js module.exports is same as exports. It is an object created by Module system.

We have to do module.exports immediately. It cannot be done in asynchronous mode in a callback.

In a require() call Node.js returns module.exports object.

21. What is Tracing in Node.js?

We can use Tracing to see the events generated by V8 engine, Node core and userspace code.

We can enable Tracing in a Node application by passing the **--trace-events-enabled** flag when starting a Node.js application.

Once this flag is enabled, we can see the log files with the Trace events in a tab of Chrome browser.

22. How will you debug an application in Node.js?

The latest implementation of Node.js includes an out-of-process debugging utility. We can access it via a TCP-based protocol and a built-in debugging client.

We have to start Node.js with the debug argument followed by the path of the script to start debugging.

We can put debugger keyword in a statement to enable a breakpoint. On running the debugger, there will be breakpoint at that location.

We can also watch expressions by using watch().

23. What is a Child Process in Node.js?

There is a child_process module in Node.js that can be used to spawn child processes.

We can use commands like spawn(), exec(), fork() etc to create new Child Processes.

These child processes can be synchronous as well as asynchronous.

For shell script automation,

synchronous child process can be used.

In asynchronous mode, we can specify a callback that will be called when Child process terminates.

24. What is a Cluster in Node.js?

We can use cluster module to take advantage of multi-core system in a Node.js application.

We can create child processes that share server ports by using Cluster module.

These processes communicate via IPC.

Cluster module can distribute the incoming connections in Round

Robin approach or in Interested Worker approach. In Round Robin approach, master distributes connections to workers in a round-robin fachion.

In Interested Worker approach, master distributes connections to interested workers only. Then workers can directly accept the connections.

Based on our application needs we can kill worker threads on need basis, without affecting other workers.

25. What is closure in JavaScript?

A Closure is an inner function that can access variables of outer function.

We use Closures extensively in Node.js. These are the main building blocks of asynchronous and non-blocking architecture of Node.js.

The inner function has access to not only its own variables but also

to the parameters of outer function.

One simple way to create a closure is to include a function inside another function in JavaScript.

E.g.

```
function printName (name) {
var address = “My name is ";
  function addressMe () {
     return address +” “ + name;
   }
return addressMe ();
}

printName(“James”); // My name
```

is James

We even use Closures in Jquery also.

26. What is a Buffer in Node.js?

A Buffer in Node.js is used to read and manipulate binary data streams. It can be used with TCP streams or in file system operations.

As per Node documentation, a Buffer is similar to an array of integers.

A Buffer has raw memory allocated outside the V8 engine heap.

When we create a Buffer, its size is specified and this size cannot be changed later.

Since Buffer class is global, any part of the program can access and use it.

27. How will you convert a Buffer to JSON in Node.js?

Node.js provides utility method buffer.toJson() to convert a buffer data into JSON format. We can use it to convert Buffer contents to JSON format.

28. Why do we use __filename in Node.js?

In Node.js __filename is the name of the current module. It is the absolute path of the current module file.

As per documentation, the __filename is not necessarily the same as the file name used in the command line for a main program in Node.js.

In general, __filename is local to a

module.

29. What is the use of Timers is Node.js?

Node.js provides a useful module timers (aka Timers) to schedule functions at a later point of time.

Timer functions are global. Therefore there is no need to call require('timers').

These API are similar to the APIs in web browser for timers tasks.

30. What are the important APIs in Timers module in Node.js?

Some of the important APIs in Timers module of Node.js are as follows:

1. **Immediate**: We can use setImmediate() to call a function immediately. We can use clearImmediate() to cancel the scheduled action.

2. **Timeout**: This setTimeout() API is used to set timeout for event loop to not exit the program till the timeout period is over. Once timeout period is over, EventLoop will exit.

3. **Interval**: We use setInterval() to call the function when the time interval elapses.

4. **Clear:** All the above three APIs provide clear APIs to clear any Immediate, Timeout or Interval settings.

31. What is EventEmitter in Node.js?

EventEmitter is a class defined in events module in Node.js. This class is used for emitting the ‘newListener’ event when a new listener is added. On removing a listener ‘removeListener’ event is emitted.

EventEmitter also maintains the list of listeners that are added to it.

In Node.js, EventEmitter is an

important class to interact with listeners in a program.

32. What is the use of net.Socket in Node.js?

Node.js provides net.Socket class in net module that can be used for creating a new socket object.

Socket is an abstraction of TCP socket. In this implementation we can use it as a duplex stream.

A client can create net.Socket and use it to connect to a server. Node.js can also create net.Socket and pass it in a connection to

client.

We have to set readable and writable flags to allow reads and writes on this socket.

33. What are the important events of net.Socket in Node.js?

Some of the important events of net.Socket in Node.js are as follows:

1. **Connect**: This event is emitted when a connection is successfully established.
2. **Lookup**: This event is emitted after resolving the hostname but before the

connection is established.

3. **Close**: When server closes, this event is emitted.
4. **End**: When socket sends FIN packet, this event is emitted.
5. **Data**: This event is emitted when data is received.
6. **Drain**: This event is emitted when the write buffer becomes empty.
7. **Timeout**: This event emits when socket timeouts due to inactivity.
8. **Error**: This is an event that emits on occurrence of an error.

34. Can we build a REST service in Node.js?

Yes, we can build a REST (Representational State Transfer) service in Node.js.

We can use ExpressJS framework for this purpose.

We can provide implementation for GET, POST and PUT requests to handle REST services on a resource.

A simple implementation with express is as follows:

```
var express = require('express');
var app = express();

app.get('/', function(req, res){
  res.send('hello Node!’);
});

app.listen(3000);
```

35. What is the use of DNS module in Node.js?

We use DNS module in Node.js to get the utilities for domain name resolution. There are two sets of functions in this module.

1. Functions to do domain name lookup
2. Function to connect to a DNS server and then do the name resolution.

Yes, we can build a REST

(Representational State Transfer) service in Node.js.

36. What are the important command line options in Node.js?

Some of the important command line options in Node.js are as follows:

1. **-i or —interactive:** This option opens REPL.
2. **--no-warnings:** We use it suppress printing of warnings.
3. **--v8-options:** It is used to

print v8 engine command line options.

4. **--preserve-symlinks:** This option is used to instructs the module loader to preserve symbolic links while resolving and caching modules.

37. How does Node.js work?

Node.js is a server side Javascript based platform. In Node.js there is a V8 engine that is like a virtual machine. This engine executes Javascript code written in Node.js. It has a single event loop and non-blocking I/O. This helps in providing very high throughput.

38. How can we avoid Callback Hell in Node.js?

Node.js heavily depends on Callback functions. But at times developers create programs that create heavily nested Callbacks. This leads to unreadable spaghetti code that is difficult to comprehend.

We can use divide and conquer approach to avoid Callback Hell. We have to create loosely coupled modules in our Node.js program

that handle specialized functions.

With each module, we define independent functions in which we can pass parameters.

39. How do you resolve unhandled exceptions in a Node.js program?

We can create a handler for uncaughtexception event in Node.js. This handler can handle the unhandled exceptions.

But this is method is not a professional method to handle unhandled exceptions. In general, we should modularize our program and create domains. Any

unhandledexception coming from a domain has to be handled at that level. This helps in handling the exception before letting it reach at Process level.

40. What is a Callback function in Node.js?

A Callback function is an important part of Event driven programming in Node.js. We use Callabck functions to handle multiple requests made to the server. With a Callback function we can handle events in an Asynchronous way.

Let say we have a large list of numbers to sort. In Node.js we can pass this list to a function with the

callback. The server can keep on handling the next request. Once the sort function finishes its work, it calls the callback function to inform server and emits an event. Based on this event further actions can be taken.

But during the sorting of list, server is not blocked. It keeps on handling other requests.

It is an important concept in creating a scalable application in Node.js.

41. What are the most popular modules of Node.js?

Some of the most popular modules of Node.js are as follows:

1. **Console**: This module provides debugging capabilities in a Node.js program.
2. **Cluster**: This module provided utilities to take advantage of multi-core

systems by a Node.js application.

3. **Crypto**: This module is mainly used in cryptography implementation in Node.js. It is an important security module in Node.js
4. **DNS**: This module is used for Node.js Domain name resolution and lookup.
5. **File System**: This is the module for handling file operations in node.js.
6. **HTTP**: This module helps in dealing with HTTP requests.
7. **REPL**: This module provides Read Eval Print Loop implementation in

Node.js. It is useful in creating programs to accept user input.

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

In readFile function Node will read the file completely and load it in the memory. Whereas in createReadStream function, the file is read as a stream.

By using createReadStream function we can read file from any

random position that is passed as an argument. We can pass start and end parameters to createReadStream function.

43. What is the use of QueryString in Node.js?

We use QueryString module to handle and process URL query strings. This module provides utilities to parse URLs and format URL query strings.

Some of the useful methods are querystring.escape(), querystring.parse(), querystring.stringify(), querystring.unescape() etc.

44. How will you get the amount of free memory on the server in which Node.js application is running?

We can use OS module utilities to get the amount of free memory on the server.

The function to use is os.freemem(). This function gives us the amount of free system

memory in bytes as an integer.

45. What is a Global object in Node.js?

There are Global objects in Node.js that are accessible to all the parts and modules of Node application. Some of the Global objects are as follows:

1. **Buffer**: This object is used to handle binary data in Node.js.
2. **Console**: This object is used to print to stderr and stdout.
3. **Process**: This is the Global

Process object in Node.js. It provides information and control on the current Node process.

46. What are the security mechanisms available in Node.js?

Some of the important security mechanisms available in Node.js are as follows:

1. **TLS**: There is a tls module in Node.js that supports Transport Layer Security (TLS). It is built on top of

OpenSSL.

2. **HTTPS**: Node.js support HTTPS protocol. This protocol support HTTP over TLS/Secure Socket Layer (SSL). It is a separate module in Node.js.

3. **Crypto**: We can also use Crypto module in Node.js to encrypt and decrypt data in a Node application. Encrypted data increases the security of the system against malicious attacks.

47. What is the use of Zlib in Node.js?

Zlib is a module in Node.js that provides compression and decompression utilities based on Gzip and Deflate/Inflate. Since there is a large amount of I/O in a Node.js application, it makes sense to use compression and decompression to save bandwidth and computing time.

48. How will you convert a Buffer content into readable String in Node.js?

We can use string_decoder module APIs to decode buffer objects. This module provides utilities to decode a buffer in a way that preserves encoded multi-byte UTF-8 and UTF-16 characters.

In this way we can convert the

Buffer contents into readable String.

49. How do you write unit test cases in Node.js?

We use Assert module to implement simple unit tests in a Node.js application. Assert module has functions like assert.deepequal() , assert.deepstrictequal() etc functions to write different unit testcases.

50. What are the standard Javascript errors?

Some of the standard Javascript errors are as follows:

1. **EvalError**: This error is thrown when a call to eval() fails.
2. **SyntaxError**: This error is thrown in response to improper JavaScript language syntax.
3. **RangeError**: When a value is not within an expected

range, this error is thrown.

4. **ReferenceError**: It is thrown when undefined variables are used.
5. **TypeError**: If we pass arguments of the wrong type, this error is thrown.
6. **URIError**: When a global URI handling function is misused this error is thrown.

Thanks!!!

We hope you liked our book. Please help us by giving your valuable review on Amazon.com.

Your review will mean a lot to us!!

REFERENCES

https://nodejs.org