Top 95 Node.js interview questions and answers in 2021.

You can check all 95 Node.js interview questions here 👉 https://devinterview.io/dev/nodejs-interview-questions

🔷 1. What is npm?

Answer:

`npm` stands for Node Package Manager. npm provides following two main functionalities:

- 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.

Source: tutorialspoint.com

🔷 2. What is Node.js?

Answer:

Node.js is a web application framework built on Google Chrome's JavaScript Engine (V8 Engine).

Node.js comes with runtime environment on which a Javascript based script can be interpreted and executed (It is analogus to JVM to JAVA byte code). This runtime allows to execute a JavaScript code on any machine outside a browser. Because of this runtime of Node.js, JavaScript is now can be executed on server as well.

Node.js = Runtime Environment + JavaScript Library

Source: tutorialspoint.com

🔷 3. What are the two types of API functions in Node.js?

Answer:

The two types of API functions in Node.js are: a) Asynchronous, non-blocking functions b) Synchronous, blocking functions

Source: lazyquestion.com

🔷 4. What is an error-first callback?

Answer:

Error-first callbacks are used to pass errors and data. The first argument is always an error object that the programmer has to check if something went wrong. Additional arguments are used to pass data.

```
fs.readFile(filePath, function(err, data) {
  if (err) {
    //handle the error
  }
  // use the data object
});
```

Source: tutorialspoint.com

🔷 5. What is Callback Hell?

Answer:

The asynchronous function requires callbacks as a return parameter. When multiple asynchronous functions are chained together then callback hell situation comes up.

Source: codeforgeek.com

🔷 6. What is control flow function?

Answer:

It is a generic piece of code which runs in between several asynchronous function calls is known as control flow function.

Source: lazyquestion.com

🔷 7. What are Event Listeners?

Answer:

Event Listeners are similar to call back functions but are associated with some event. For example when a server listens to http request on a given port a event will be generated and to specify http server has received and will invoke corresponding event listener. Basically, Event listener's are also call backs for a corresponding event.

Node.js has built in event's and built in event listeners. Node.js also provides functionality to create Custom events and Custom Event listeners.

Source: lazyquestion.com

🔷 8. If Node.js is single threaded then how it handles concurrency?

Answer:

Node provides a single thread to programmers so that code can be written easily and without bottleneck. Node internally uses multiple POSIX threads for various I/O operations such as File, DNS, Network calls etc.

When Node gets I/O request it creates or uses a thread to perform that I/O operation and once the operation is done, it pushes the result to the event queue. On each such event, event loop runs and checks the queue and if the execution stack of Node is empty then it adds the queue result to execution stack.

This is how Node manages concurrency.

Source: codeforgeek.com

🔷 9. Could we run an external process with Node.js?

Answer:

Yes. *Child process module* enables us to access operating system functionaries or other apps. Scalability is baked into Node and child processes are the key factors to scale our application. You can use child process to run system commands, read large files without blocking event loop, decompose the application into various “nodes” (That’s why it’s called Node).

Child process module has following three major ways to create child processes –

- spawn - child_process.spawn launches a new process with a given command.
- exec - child_process.exec method runs a command in a shell/console and buffers the output.
- fork - The child_process.fork method is a special case of the spawn() to create child processes.

Source: codeforgeek.com

🔷 10. What are the key features of Node.js?

Answer:

Let's look at some of the key features of Node.js.

- **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.

Source: techbeamers.com

🔷 11. What is the difference between Nodejs, AJAX, and jQuery?

Answer:

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.

Source: techbeamers.com

🔷 12. What are the core modules of Node.js?

Answer:

- EventEmitter
- Stream
- FS
- Net
- Global Objects

Source: github.com/jimuyouyou

🔷 13. What is global installation of dependencies?

Answer:

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

Source: tutorialspoint.com

🔷 14. What do you mean by Asynchronous API?

Answer:

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.

Source: tutorialspoint.com

🔷 15. What are the benefits of using Node.js?

Answer:

Following are main benefits of using Node.js

- **Aynchronous and Event Driven** - 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.
- **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.

Source: tutorialspoint.com

🔷 16. What is libuv?

Answer:

libuv is a C library that is used to abstract non-blocking I/O operations to a consistent interface across all supported platforms. It provides mechanisms to handle file system, DNS, network, child processes, pipes, signal handling, polling and streaming. It also includes a thread pool for offloading work for some things that can't be done asynchronously at the operating system level.

Source: nodejs.org

🔷 17. What is V8?

Answer:

The V8 library provides Node.js with a JavaScript engine (a program that converts Javascript code into lower level or machine code that microprocessors can understand), which Node.js controls via the V8 C++ API. V8 is maintained by Google, for use in Chrome.

The Chrome V8 engine :

- The V8 engine is written in C++ and used in Chrome and Nodejs.
- It implements ECMAScript as specified in ECMA-262.
- The V8 engine can run standalone we can embed it with our own C++ program.

Source: nodejs.org

🔷 18. What is the difference between returning a callback and just calling a callback?

Answer:

```
return callback();
//some more lines of code; -  won't be executed
callback();
//some more lines of code; - will be executed
```

Of course returning will help the context calling async function get the value returned by callback.

```
function do2(callback) {
log.trace('Execute function: do2');
return callback('do2 callback param');
}


var do2Result = do2((param) => {
log.trace(</span><span class="token cString">print </span><span class="token
interpolation"><span class="token interpolation-punctuation cBase">${</span>param<span
class="token interpolation-punctuation cBase">}</span></span><span class="token template-
punctuation cString">);
return </span><span class="token cString">return from callback(</span><span class="token
interpolation"><span class="token interpolation-punctuation cBase">${</span>param<span
class="token interpolation-punctuation cBase">}</span></span><span class="token
cString">)</span><span class="token template-punctuation cString">; // we could use that
return
});


log.trace(</span><span class="token cString">print </span><span class="token
interpolation"><span class="token interpolation-punctuation
cBase">${</span>do2Result<span class="token interpolation-punctuation cBase">}</span>
</span><span class="token template-punctuation cString">);
```

Output:

```
C:\Work\Node>node --use-strict main.js
[0] Execute function: do2
[0] print do2 callback param
[0] print return from callback(do2 callback param)
```

Source: stackoverflow.com

🔷 19. List out the differences between AngularJS and NodeJS?

Answer:

AngularJS is a web application development framework. It's a JavaScript and it is different from other web app frameworks written in JavaScript like jQuery. NodeJS is a runtime environment used for building server-side applications while AngularJS is a JavaScript framework mainly useful in building/developing client-side part of applications which run inside a web browser.

Source: a4academics.com

🔷 20. Is Node a single threaded application?

Answer:

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

Source: tutorialspoint.com

🔷 21. What's the difference between operational and programmer errors?

Answer:

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

Source: blog.risingstack.com

🔷 22. How you can monitor a file for modifications in Node.js ?

Answer:

We can take advantage of File System `watch()` function which watches the changes of the file.

Source: codingdefined.com

🔷 23. How to make Post request in Node.js?

Answer:

Following code snippet can be used to make a Post Request in Node.js.

```
var request = require('request');
request.post('http://www.example.com/action', {
form: {
key: 'value'
}
}, function(error, response, body) {
```

```
if (!error && response.statusCode == 200) {
console.log(body)
}
});
```

Source: techbeamers.com

🔷 24. What is Callback?

Answer:

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 supports callbacks.

For example, a function to read a file may start reading file and return the control to execution environment immediately 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 request without waiting for any function to return result.

Source: tutorialspoint.com

🔷 25. What is Chaining in Node?

Answer:

Chanining is a mechanism to connect output of one stream to another stream and create a chain of multiple stream operations. It is normally used with piping operations.

Source: tutorialspoint.com

🔷 26. What are the global objects of Node.js?

Answer:

These objects are available in all modules:

- **process** - The process object is a global that provides information about, and control over, the current Node.js process.
- **console** - Used to print to stdout and stderr.
- **buffer** - Used to handle binary data.

Source: github.com/jimuyouyou

🔷 27. How to use Buffer in Node.js?

Answer:

Buffer is used to process binary data, such as pictures, mp3, database files, etc. Buffer supports a variety of encoding and decoding, binary string conversion.

Source: github.com/jimuyouyou

🔷 28. How can you avoid callback hells?

Answer:

To do so you have more options:

- **modularization**: break callbacks into independent functions
- use *Promises*
- use `yield` with *Generators* and/or *Promises*

Source: tutorialspoint.com

🔷 29. What is N-API in Node.js?

Answer:

N-API (pronounced N as in the letter, followed by API) is an API for building native Addons. It is independent from the underlying JavaScript runtime (ex V8) and is maintained as part of Node.js itself. This API will be Application Binary Interface (ABI) stable across versions of Node.js. It is intended to insulate Addons from changes in the underlying JavaScript engine and allow modules compiled for one version to run on later versions of Node.js without recompilation.

Source: medium.com

🔷 30. Are you familiar with differences between Node.js nodules and ES6 nodules?

Answer:

The modules used in Node.js follow a module specification known as the **CommonJS** specification. The recent updates to the JavaScript programming language, in the form of ES6, specify changes to the language, adding things like new class syntax and a module system. This module system is different from Node.js modules. To import ES6 module, we'd use the ES6 `import` functionality.

Now ES6 modules are incompatible with Node.js modules. This has to do with the way modules are loaded differently between the two formats. If you use a compiler like Babel, you can mix and match module formats.

Source: stackoverflow.com

🔷 31. What is the purpose of setTimeout function?

👉 Check all 95 answers

🔷 32. How do you debug Node.js applications?

👉 Check all 95 answers

🔷 33. What is purpose of Buffer class in Node?

👉 Check all 95 answers

🔷 34. How Node prevents blocking code?

👉 Check all 95 answers

🔷 35. What's the event loop?

👉 Check all 95 answers

🔷 36. How to avoid callback hell in Node.js?

👉 Check all 95 answers

🔷 37. Explain how does Node.js work?

👉 Check all 95 answers

🔷 38. How does Node.js handle child threads?

👉 Check all 95 answers

🔷 39. What is the relationship between Node.js and V8?

👉 Check all 95 answers

🔷 40. Explain the concept of Domain in Node.js

👉 Check all 95 answers

🔷 41. What is REPL in context of Node?

👉 Check all 95 answers

🔷 42. What is stream and what are types of streams available in Node.js?

👉 Check all 95 answers

🔷 43. What are streams?

👉 Check all 95 answers

🔷 44. What is Event Loop?

👉 Check all 95 answers

🔷 45. What is Event Emmitter?

👉 Check all 95 answers

🔷 46. What is the preferred method of resolving unhandled exceptions in Node.js?

👉 Check all 95 answers

🔷 47. What is a blocking code?

👉 Check all 95 answers

🔷 48. When should we use Node.js?

👉 Check all 95 answers

🔷 49. When should I use EventEmitter?

👉 Check all 95 answers

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

👉 Check all 95 answers

🔷 51. What are the use cases for the Node.js "vm" core module?

👉 Check all 95 answers

🔷 52. Rewrite promise-based Node.js applications to Async/Await

👉 Check all 95 answers

🔷 53. How to gracefully Shutdown Node.js Server?

👉 Check all 95 answers

🔷 54. Why to use Buffers instead of binary strings to handle binary data ?

👉 Check all 95 answers

🔷 55. How can you listen on port 80 with Node?

👉 Check all 95 answers

🔷 56. How the V8 engine works?

👉 Check all 95 answers

🔷 57. Does Node.js support multi-core platforms? And is it capable of utilizing all the cores?

👉 Check all 95 answers

🔷 58. Is it possible to use "Class" in Node.js?

👉 Check all 95 answers

🔷 59. What is LTS releases of Node.js why should you care?

👉 Check all 95 answers

🔷 60. Is Node.js entirely based on a single-thread?

👉 Check all 95 answers

🔷 61. When to not use Node.js?

👉 Check all 95 answers

🔷 62. What is Piping in Node?

👉 Check all 95 answers

🔷 63. What is the purpose of __filename variable?

👉 Check all 95 answers

🔷 64. What's the difference between dependencies, devDependencies and peerDependencies in npm package.json file?

👉 Check all 95 answers

🔷 65. How would you handle errors for async code in Node.js?

👉 Check all 95 answers

🔷 66. Can Node.js work without V8?

👉 Check all 95 answers

🔷 67. What are async functions in Node? Provide some examples.

👉 Check all 95 answers

🔷 68. What are the timing features of Node.js?

👉 Check all 95 answers

🔷 69. Explain usage of NODE_ENV

👉 Check all 95 answers

🔷 70. What's a stub? Name a use case.

👉 Check all 95 answers

🔷 71. Name some of the events fired by streams.

👉 Check all 95 answers

🔷 72. Is Node.js entirely based on a single-thread?

👉 Check all 95 answers

🔷 73. What tools can be used to assure consistent code style?

👉 Check all 95 answers

🔷 74. How does the cluster module work? What's the difference between it and a load balancer?

👉 Check all 95 answers

🔷 75. How does libuv work under the hood?

👉 Check all 95 answers

🔷 76. Explain what is Reactor Pattern in Node.js?

👉 Check all 95 answers

🔷 77. Can Node.js use other engines than V8?

👉 Check all 95 answers

🔷 78. Why Node.js devs tend to lean towards the Module Requiring vs Dependency Injection?

👉 Check all 95 answers

🔷 79. How to solve "Process out of Memory Exception" in Node.js ?

👉 Check all 95 answers

🔷 80. How would you scale Node application?

👉 Check all 95 answers

🔷 81. What is the difference between process.nextTick() and setImmediate() ?

👉 Check all 95 answers

🔷 82. Explain some Error Handling approaches in Node.js you know about. Which one will you use?

👉 Check all 95 answers

🔷 83. What is the purpose of using hidden classes in V8?

👉 Check all 95 answers

🔷 84. Why do we need C++ Addons in Node.js?

👉 Check all 95 answers

🔷 85. Why should you separate Express 'app' and 'server'?

👉 Check all 95 answers

🔷 86. What is V8 Templates?

👉 Check all 95 answers

🔷 87. How V8 compiles JavaScript code?

👉 Check all 95 answers

🔷 88. How many threads does Node actually create?

👉 Check all 95 answers

🔷 89. Provide some example of config file separation for dev and prod environments

👉 Check all 95 answers

🔷 90. How do you convert an existing callback API to promises?

👉 Check all 95 answers

🔷 91. Consider following code snippet

👉 Check all 95 answers

🔷 92. Explain the result of this code execution

👉 Check all 95 answers

🔷 93. Rewrite the code sample without try/catch block

👉 Check all 95 answers

🔷 94. What will happen when that code will be executed?

👉 Check all 95 answers

🔷 95. Explain the result of this code execution

👉 Check all 95 answers

Thanks 🙌 for reading and good luck on your next tech interview!
Explore 3800+ dev interview question here 👉 Devinterview.io