**Table of Contents**

* [Node.js Interview Questions](https://hackr.io/blog/node-js-interview-questions#Nodejs_Interview_Questions)
  + [Question: Explain the basics of Node.js.](https://hackr.io/blog/node-js-interview-questions#Question_Explain_the_basics_of_Nodejs)
  + [Question: How are ‘Child Threads’ handled in Node.js?](https://hackr.io/blog/node-js-interview-questions#Question_How_are_Child_Threads_handled_in_Nodejs)
  + [Question: State some differences between Angular JS and Node.js.](https://hackr.io/blog/node-js-interview-questions#Question_State_some_differences_between_Angular_JS_and_Nodejs)
  + [Question: State the primary uses of Node.js.](https://hackr.io/blog/node-js-interview-questions#Question_State_the_primary_uses_of_Nodejs)
  + [Question: What is Event-Driven programming?](https://hackr.io/blog/node-js-interview-questions#Question_What_is_Event-Driven_programming)
  + [Question: In the context of Node.js, what is REPL?](https://hackr.io/blog/node-js-interview-questions#Question_In_the_context_of_Nodejs_what_is_REPL)
  + [Question: What is a test pyramid in Node.js?](https://hackr.io/blog/node-js-interview-questions#Question_What_is_a_test_pyramid_in_Nodejs)
  + [Question: What is libuv?](https://hackr.io/blog/node-js-interview-questions#Question_What_is_libuv)
  + [Question: Is Node.js the best platform for CPU-heavy applications?](https://hackr.io/blog/node-js-interview-questions#Question_Is_Nodejs_the_best_platform_for_CPU-heavy_applications)
  + [Question: What is the purpose of the Express JS Package?](https://hackr.io/blog/node-js-interview-questions#Question_What_is_the_purpose_of_the_Express_JS_Package)
  + [Question: What are LTS versions of Node.js?](https://hackr.io/blog/node-js-interview-questions#Question_What_are_LTS_versions_of_Nodejs)
  + [Question: Explain the working of assert in Node.js](https://hackr.io/blog/node-js-interview-questions#Question_Explain_the_working_of_assert_in_Nodejs)
  + [Question: What is callback hell?](https://hackr.io/blog/node-js-interview-questions#Question_What_is_callback_hell)
  + [Question: What is stub in Node.js?](https://hackr.io/blog/node-js-interview-questions#Question_What_is_stub_in_Nodejs)
  + [Question: What is Event Loop?](https://hackr.io/blog/node-js-interview-questions#Question_What_is_Event_Loop)
  + [Question: What is stream in Node.js? What are its types?](https://hackr.io/blog/node-js-interview-questions#Question_What_is_stream_in_Nodejs_What_are_its_types)
  + [Question: List and explain the timing features of Node.js.](https://hackr.io/blog/node-js-interview-questions#Question_List_and_explain_the_timing_features_of_Nodejs)
  + [Question: Highlight the differences between process.nextTick() and setImmediate().](https://hackr.io/blog/node-js-interview-questions#Question_Highlight_the_differences_between_processnextTick_and_setImmediate)
  + [Question: Explain readFile and createReadStream in Node.js.](https://hackr.io/blog/node-js-interview-questions#Question_Explain_readFile_and_createReadStream_in_Nodejs)
  + [Question: Does Node.js provide a Debugger?](https://hackr.io/blog/node-js-interview-questions#Question_Does_Nodejs_provide_a_Debugger)
  + [Question: Describe the exit codes in Node.js.](https://hackr.io/blog/node-js-interview-questions#Question_Describe_the_exit_codes_in_Nodejs)
  + [Question: Why is NODE\_ENV used?](https://hackr.io/blog/node-js-interview-questions#Question_Why_is_NODE_ENV_used)
  + [Question: What is Event Emitter in Node.js?](https://hackr.io/blog/node-js-interview-questions#Question_What_is_Event_Emitter_in_Nodejs)
  + [Question: What is Punycode?](https://hackr.io/blog/node-js-interview-questions#Question_What_is_Punycode)
  + [Question: Explain the concept of JIT and highlight its relation with Node.js.](https://hackr.io/blog/node-js-interview-questions#Question_Explain_the_concept_of_JIT_and_highlight_its_relation_with_Nodejs)
  + [Question: Why is buffer class used in Node.js?](https://hackr.io/blog/node-js-interview-questions#Question_Why_is_buffer_class_used_in_Nodejs)
  + [Question: What is the difference between fork () and spawn () methods in Node.js?](https://hackr.io/blog/node-js-interview-questions#Question_What_is_the_difference_between_fork_and_spawn_methods_in_Nodejs)
  + [Question: State the steps to write an Express JS application.](https://hackr.io/blog/node-js-interview-questions#Question_State_the_steps_to_write_an_Express_JS_application)

**Spread the Knowledge**

Node.js was initially developed by Ryan Dahl and saw its first release in May 2009. It is a JS environment used to execute JavaScript code outside the browser. Based on Chrome’s V8 engine, it represents the JavaScript anywhere and everywhere programming paradigm, unifying the web app development to a single language rather than different languages for server and client-side scripting.

Many corporate such as LinkedIn, IBM, GoDaddy, Groupon, Netflix, and Paypal use Node.js. Its increasing popularity has sparked a demand for Node.js developers.

**Node.js Interview Questions**

If you wish to make a career in Node.js, below are the top Node.js interview questions with answers which will make you interview-ready.

**Question: Explain the basics of Node.js.**

**Answer:** In its elemental sense, Node.js is a JS runtime built using Chrome’s V8 JavaScript engine as a platform. Node.js has gained popularity as it is lightweight and efficient, majorly due to its event-driven and non-blocking I/O model. Built with performance as its primary focus, Node.js is responsible for processing the JavaScript code into the native machine code, which can then be used by your computer to execute the processes. Even though it is based on the V8 engine, which is used in Google Chrome and other chromium based browsers (Vivaldi, Brave and Opera), it does not run in the browser itself. During development, various features such as file system API, HTTP library and OS utility methods were added to the engine, so that Node.js can be executed as a program on a computer.

**Question: How are ‘Child Threads’ handled in Node.js?**

**Answer:** In its core sense, Node.js is a single thread process. Also, it does not expose any child threads and the modes of thread management to the developer. However, child threads may be generated in Node.js in a variety of processes, one of them being asynchronous I/O. Although the child threads spawned through these processes run in the backdrop, they don’t block the main code or execute any application code. But if you require threading support in an application powered by Node.js, multiple tools are available for utilization.

**Question: State some differences between Angular JS and Node.js.**

**Answer:** Below are some differences between Angular JS and Node.js

|  |  |
| --- | --- |
| **Angular JS** | **Node.js** |
| Angular JS is written in TypeScript | Node.js is written in a variety of languages, such as C, C++, and JavaScript |
| It is ideal while creating highly interactive web pages | It is suited for small-scale projects and applications |
| It is an open-source framework for web application development | It is a run-time environment based on multiple platforms |
| Used to create single-page applications for client-side | Use to create server-side networking applications |
| Helps in splitting an application into model-view-controller (MVC) components | Helps in generating queries for databases |
| Appropriate for the development of real-time applications | It is appropriate in situations when something faster and highly scalable is required |
| Angular itself is a web application framework | Node.js has many frameworks, including Express.js, Partial.js and more |

**Question: State the primary uses of Node.js.**

**Answer:**

* **Complex single-page applications:** Node.js is ideal for creating single-page applications which are complex in nature, such as online drawing tools, mail solutions, and social networking. These types of applications are limited to one-page and the UX is similar to that of a desktop application. Node.js can be used here due to its asynchronous data flow happening at the backend.
* **Real-Time Applications (RTA):** We use a hefty number of Real-Time applications in our day-to-day life. For example, Google Docs, Slack, Skype, WhatsApp and many more. Node.js’ event API, WebSockets, and asynchronous data flow can ensure a faultless server operation, which will update the data instantly.
* **Chat Rooms:** This may be clubbed under RTA, but since instant messaging and chatting has emerged as one of the top real-time application models, it needs a special focus. If your product is in this realm, you are looking at requirements such as lightweight, high traffic capacity and substantial data flow. All these requirements can be fulfilled using Node.js and some [JavaScript framework](https://hackr.io/blog/10-best-javascript-frameworks-2019) at the backend. The aforementioned web sockets come in handy in receiving and sending messages in a chat room environment.
* **Browser-Based Games:** The above-mentioned chat rooms can also be integrated into browser-based games, where Node.js is a perfect choice. Combining the Node.js technology with HTML5 and even JS tools will help you create real-time browser-based games.
* **Data Streaming Applications:**Another type of application where Node.js is perfect is data streaming applications. The key selling point of these applications is that they can process data in the unloading phase. Through this, while some parts can be downloaded upfront to keep the connection and can download the other parts if and when necessary. In this context, Node.js streaming applications deal with both audio and video data.
* **Representational State Transfer (REST) Application Programming Interfaces (APIs):** APIs based on REST hold a key position in the construction of modern commercial software architecture due to the wide usage of the HyperText Transfer Protocol (HTTP). The Express.js framework of the Node.js ecosystem can help in building fast and light REST APIs.
* **Server-Side Web Applications:** While Node.js and its frameworks can help in creating server-side web applications, it should be noted that CPU-heavy operations should not be expected.
* **Command Line Tools:** Node.js’ expansive ecosystem is an advantage when it comes to building a command line tool, and there are a variety of tutorials available online which can help you build your own CLT.

**Question: What is Event-Driven programming?**

**Answer:** Event-Driven programming approach profoundly uses events to trigger various functions. In this scenario, an event can be anything, such as the pressing of a key or clicking of a mouse button. Whenever an event occurs, a call-back function already registered with the element is executed, following the ‘publish-subscribe’ pattern. Due to this programming approach, Node.js is faster than other comparable technologies.

**Question: In the context of Node.js, what is REPL?**

**Answer:** In the context of Node.js, REPL is Read, Eval, Print and Loop. REPL is a computer environment (similar to a Windows console or Linux shell) where any command that is entered is met with a system responded output. The REPL environment is bundled with Node.js by default, and it performs the following tasks:

* **Read:** Reads the inputs from the user and converts it into JavaScript data-structure, finally storing it in memory
* **Eval:** It receives the data structure and evaluates it
* **Print:** It prints the final output
* **Loop:** It loops the provided command till CTRL + C is pressed two times

**Question: What is a test pyramid in Node.js?**

**Answer:** In Node.js, a test pyramid is a figure which explains the proportion of unit tests, integrations tests, and end-to-end tests are required for the fruitful development of a project. The components of a test pyramid are given below:

* **Unit Tests:** They test the individual units of code in isolation. They are fast and you might perform a lot of these tests
* **Integrations Tests:** They test the integration among dissimilar units.
* **End-to-End (E2E) Tests:** They test the system as a whole, right from the User Interface to the data store, and back.

**Question: What is libuv?**

**Answer:** Libuv, a support library of Node.js, is used for asynchronous Input/output. While it was initially developed just for Node.js, it now witnesses practice with other systems such as Luvit, Julia, Pyuv and more. Some of the features of Libuv are:

* File System Events
* Child Processes
* Full-featured event loop backed
* Asynchronous TCP & UDP sockets

**Question: Is Node.js the best platform for CPU-heavy applications?**

**Answer:** Even though Node.js can help in creating server-side applications, CPU-incentive applications are not the strong suit of Node.js. The CPU-heavy operations pave the way for the blockage of incoming requests and push the thread into critical situations.

**Question: What is the purpose of the Express JS Package?**

**Answer:** Built on top of Node.js, ExpressJS is a JS framework used to manage the flow of information between the routes and server in server-side apps. Being lightweight, flexible and filled with a variety of relevant features, it is apt for mobile and web application development.

**Question: What are LTS versions of Node.js?**

**Answer:** Long Term Support or LTS version/releases of Node.js are the releases which receive all the critical fixes, performance step ups and security updates. These versions receive support for at least 1.5 years and have a focus on security and stability of the application.

**Question: Explain the working of assert in Node.js**

**Answer:** Assert is used to write tests in Node.js. The feedback is provided only if any of the test cases that are running fails. To test invariants, the module gives you a set of assertion tests. It is used internally by Node.js, but if you use require (‘assert’) code, you will be able to use it in other applications as well.

**Question: What is callback hell?**

**Answer:** In Node.js, callback hell is also known as the Pyramid of Doom. Caused by intensively nested, unreadable and unmanageable callbacks, making the code harder to read and debug. It is caused due to improper execution of the asynchronous logic.

**Question: What is stub in Node.js?**

**Answer:** Stubs are programs or functions used to stimulate component behavior. Stubs provide the answers to the functions during test cases.

**Question: What is Event Loop?**

**Answer:** All the asynchronous callbacks are handled by an event loop in Node.js. It is the foundation of the non-blocking input/output in Node.js, making it one of the most vital features of the environment. Due to the nature of Node.js being event-driven, a listener can be attached to an event for the callback to be executed by the former when the event occurs. Node.js executes the event loop and then moves on to the rest of the code, without having to wait for the output. Once the whole operation culminates, it receives the output and the callback function is executed. Once it receives the response, the functions are executed one by one.

**Question: What is stream in Node.js? What are its types?**

**Answer:** In Node.js, streams are the collection of data similar to strings and arrays. Moreover, streams are objects through which you can read source data or write destination data continuously. These streams are particularly helpful for reading and processing large amounts of information. There are four types of streams in Node.js, which are:

* **Readable**: Used to read large amount of data from source
* **Writeable**: Used to write data to destination
* **Duplex**: Used for both read and write
* **Transform**: A duplex stream used for data modification

**Question: List and explain the timing features of Node.js.**

**Answer:** A timer module containing multiple functions for the execution of the code after a specific time period is provided by Node.js. Some of the functions provided in this module are:

* **process.nextTick:** This function schedules a callback function which is required to be invoked in the next iteration of the event loop
* **setTimeout/clearTimeout:** This function schedules code execution after the assigned amount of time (in milliseconds)
* **setImmediate/clearImmediate:** This functions executes code at the conclusion of the existing event loop cycle
* **setInterval/clearInterval:** This function is used to execute a block of code a number of times

**Question: Highlight the differences between process.nextTick() and setImmediate().**

**Answer:** Both process.nextTick() and setImmediate() are functions of the Timers module, but the difference lies in their execution.

* The process.nextTick() function waits for the execution of action till the next pass around in the event loop or when the event loop culminates, then only the callback function is invoked.
* The setImmediate() function is used for callback method execution on the next cycle of the event loop, which returns it to the event loop for the execution of the input/output operations.

**Question: Explain readFile and createReadStream in Node.js.**

**Answer:** Both readFile and createReadStream are ways to read and execute files provided by the Node.js.

The readFile process is fully buffered which returns response(s) only if the complete file is entered into the buffer and can be read. This process is highly memory intensive and can become slow in case the file size is large.

The createReadStream process is partially buffered, treating the entire process as a series of events. In this process, the whole files are split into chunks that are processed and then sent as a response individually in succession. Unlike readFile, createReadStream is effective when it comes to reading and processing large files.

**Question: Does Node.js provide a Debugger?**

**Answer:** A built-in TCP protocol and the debugging client is provided by Node.js. If you wish to debug your file, you can use the following argument before the name of your JS file which you wish to debug.

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

**Question: Describe the exit codes in Node.js.**

**Answer:** Exit codes in Node.js are a specific group of codes which finish off processes, which can include global objects as well. Some of the exit codes in Node.js are:

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

**Question: Why is NODE\_ENV used?**

**Answer:** When any Node.js project is in the stage of production, Node.js promotes the principle to use NODE\_ENV variable to flag it. When the NODE-ENV is set to production, your application will perform at a speed 2 to 3 times faster than usual. The variable also improves judgment during the development phase of projects.

**Question: What is Event Emitter in Node.js?**

**Answer:** Node.js has an EventEmitter class which holds all the objects which can emit events. These objects hold a function called eventEmitter.on() using which multiple functions can be attached to the event emitted by the object. Whenever an object from the EventEmitter class throws an event, all the attached functions to the vent are called upon synchronously.

**Question: What is Punycode?**

**Answer:** Punycode can be defined as an encoding syntax in Node.js which is helpful for converting the Unicode string of characters into ASCII. This is done as the hostnames can only comprehend ASCII codes and not Unicode. While it was bundled up within the default package in recent versions, you can use it in the previous version using the following code:

**punycode = require(‘punycode’);**

**Question: Explain the concept of JIT and highlight its relation with Node.js.**

**Answer:** A JIT or Just-in-time compiler sends bytecode (consisting of interpretable instructions) to the processor by converting it into instruction. Once you are finished with the writing part of a program, the source language statements are compiled into bytecode by the compiler, rather than the code that carries the data which is similar to the destination hardware platform processor.

Node.js employs JIT compilation which improves the speed of code execution to a great extent. It takes the source code and converts it into machine code in runtime. Through this, functions which are called regularly are compiled to machine code, increasing the overall speed of code execution.

**Question: Why is buffer class used in Node.js?**

**Answer:** In Node.js, the buffer class stores the raw data, in a manner similar to that of an array of integers. However, it communicates to a raw memory dump, allocated outside the V8 heap. The buffer class is a global class and can be accessed in an application without having to import the buffer module. It is majorly used as pure JavaScript code is not attuned with binary data.

**Question: What is the difference between fork () and spawn () methods in Node.js?**

**Answer:** In Node.js, spawn () launches a new process with the available set of commands. This doesn’t generate a new V8 instance only a single copy of the node module is active on the processor. This method can be used when your child process returns a large amount of data to the node.

On the other hand, fork () is a particular case of spawn () which generates a new V8 engines instance. Through this method, multiple workers run on a single node code base for multiple tasks.

**Question: State the steps to write an Express JS application.**

**Answer:**To set up an ExpressJs application, you need to follow the following steps:

* Create a folder with the project name
* Create a file named package.json inside the folder
* Run the ‘npm install’ command on the command prompt to install the libraries present in the package file\
* Create a file named server.js
* Create the ‘router’ file inside the package consisting of a folder named as index.js
* The application is created inside the package containing the index.html file

With this, we come to an end of the top Node.js interview questions. Do you think we missed an important question or have any doubts that need to be cleared? Let us know in the comments below. You can also check our best [Node.js tutorials](https://hackr.io/tutorials/learn-node-js?ref=blog-post), curated by community members.