Backend Developer Interview Questions

node.js is a super popular server-side platform that more and more organizations are using. If you are preparing for a career change and have an upcoming job interview, it's always a good idea to prepare and brush up on your interview skills beforehand. Although there are a few commonly asked Node.js interview questions that pop up during all types of interviews, we also recommend that you prepare by focusing on exclusive questions to your specific industry.

We have compiled a comprehensive list of common Node.js interview questions that come up often in interviews and the best ways to answer these questions. This will also help you understand the fundamental concepts of Node.js.

1. What is Node.js? Where can you use it?

Node.js is an open-source, cross-platform JavaScript runtime environment and library to run web applications outside the client's browser. It is used to create server-side web applications.

Node.js is perfect for data-intensive applications as it uses an asynchronous, event-driven model. You can use I/O intensive web applications like video streaming sites. You can also use it for developing: Real-time web applications, Network applications, General-purpose applications, and Distributed systems.

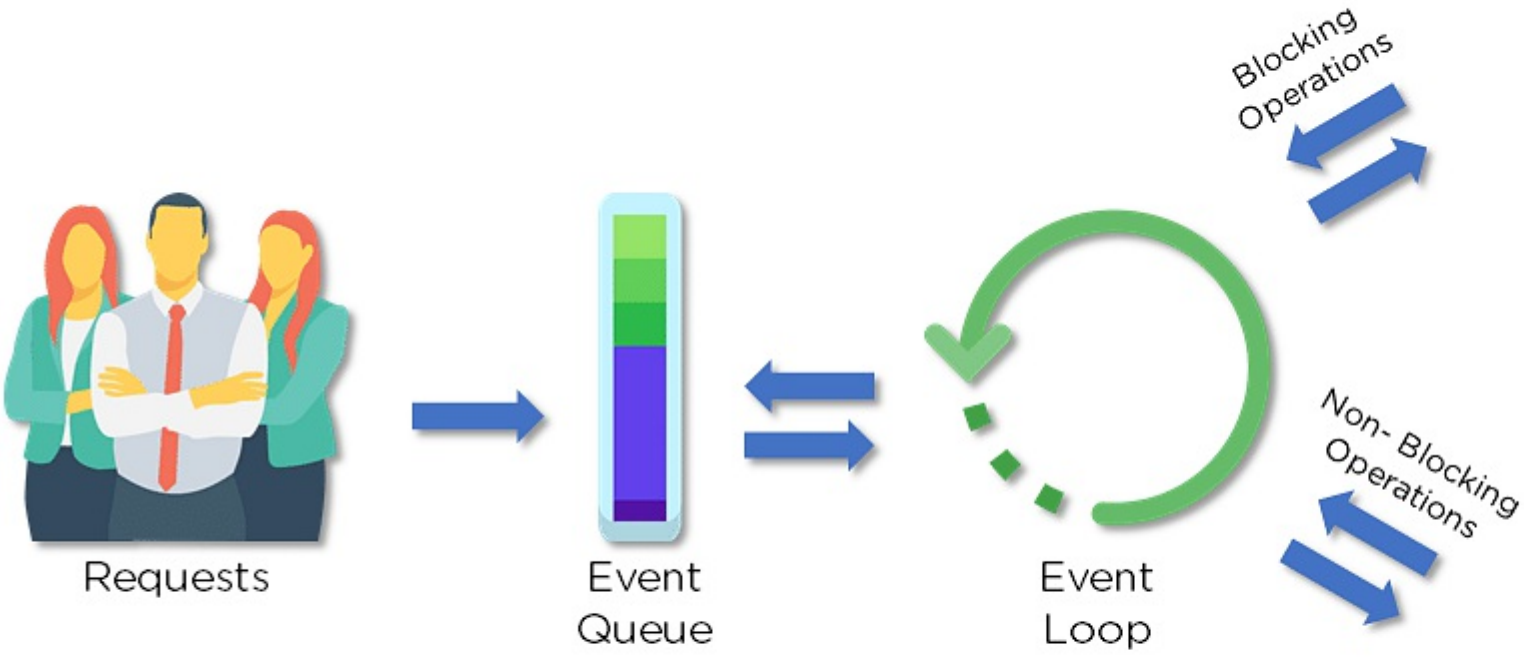
2. Why use Node.js?

Node.js makes building scalable network programs easy. Some of its advantages include:

- It is generally fast
- It rarely blocks
- It offers a unified programming language and data type
- Everything is asynchronous
- It yields great concurrency

3. How does Node.js work?

A web server using Node.js typically has a workflow that is quite similar to the diagram illustrated below. Let's explore this flow of operations in detail.



- Clients send requests to the webserver to interact with the web application. Requests can be non-blocking or blocking:
- Querying for data
- Deleting data
- Updating the data
- Node.js retrieves the incoming requests and adds those to the Event Queue
- The requests are then passed one-by-one through the Event Loop. It checks if the requests are simple enough not to require any external resources
- The Event Loop processes simple requests (non-blocking operations), such as I/O Polling, and returns the responses to the corresponding clients

A single thread from the Thread Pool is assigned to a single complex request. This thread is responsible for completing a particular blocking request by accessing external resources, such as computation, database, file system, etc.

Once the task is carried out completely, the response is sent to the Event Loop that sends that response back to the client.

4. Why is Node.js 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 instead of the typical thread-based implementation.

5. If Node.js is single-threaded, then how does it handle concurrency?

The Multi-Threaded Request/Response Stateless Model is not followed by the Node JS Platform, and it adheres to the Single-Threaded Event Loop Model. The Node JS Processing paradigm is heavily influenced by the JavaScript Event-based model and the JavaScript callback system. As a result, Node.js can easily manage more concurrent client requests. The event loop is the processing model's beating heart in Node.js.

6. Explain callback in Node.js.

A callback function is called after a given task. It allows other code to be run in the meantime and prevents any blocking. Being an asynchronous platform, Node.js heavily relies on callback. All APIs of Node are written to support callbacks.

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

- The control flow of asynchronous logic is more specified and structured.
- The coupling is low.
- We've built-in error handling.
- Improved readability.

8. How would you define the term I/O?

- The term I/O is used to describe any program, operation, or device that transfers data to or from a medium and to or from another medium
- Every transfer is an output from one medium and an input into another. The medium can be a physical device, network, or files within a system

9. How is Node.js most frequently used?

Node.js is widely used in the following applications:

1. Real-time chats
2. Internet of Things
3. Complex SPAs (Single-Page Applications)
4. Real-time collaboration tools
5. Streaming applications
6. Microservices architecture

10. Explain the difference between frontend and backend development?

Front-end	Back-end
Frontend refers to the client-side of an application	Backend refers to the server-side of an application
It is the part of a web application that users can see and interact with	It constitutes everything that happens behind the scenes
It typically includes everything that attributes to the visual aspects of a web application	It generally includes a web server that communicates with a database to serve requests
HTML, CSS, JavaScript, AngularJS, and ReactJS are some of the essentials of frontend development	Java, PHP, Python, and Node.js are some of the backend development technologies

11. What is NPM?

NPM stands for Node Package Manager, responsible for managing all the packages and modules for Node.js.

Node Package Manager provides two main functionalities:

- Provides online repositories for node.js packages/modules, which are searchable on search.nodejs.org
- Provides command-line utility to install Node.js packages and also manages Node.js versions and dependencies

12. What are the modules in Node.js?

Modules are like JavaScript libraries that can be used in a Node.js application to include a set of functions. To include a module in a Node.js application, use the require() function with the parentheses containing the module's name.

Node.js has many modules to provide the basic functionality needed for a web application. Some of them include:

Core Modules	Description
HTTP	Includes classes, methods, and events to create a Node.js HTTP server
util	Includes utility functions useful for developers

fs	Includes events, classes, and methods to deal with file I/O operations
url	Includes methods for URL parsing
query string	Includes methods to work with query string
stream	Includes methods to handle streaming data
zlib	Includes methods to compress or decompress files

13. What is the purpose of the module .Exports?

In Node.js, a module encapsulates all related codes into a single unit of code that can be parsed by moving all relevant functions into a single file. You may export a module with the module and export the function, which lets it be imported into another file with a needed keyword.

14. Why is Node.js preferred over other backend technologies like Java and PHP?

Some of the reasons why Node.js is preferred include:

- Node.js is very fast
- Node Package Manager has over 50,000 bundles available at the developer's disposal
- Perfect for data-intensive, real-time web applications, as Node.js never waits for an API to return data
- Better synchronization of code between server and client due to same code base
- Easy for web developers to start using Node.js in their projects as it is a JavaScript library

15. What is the difference between Angular and Node.js?

Angular	Node.js
It is a frontend development framework	It is a server-side environment
It is written in TypeScript	It is written in C, C++ languages
Used for building single-page, client-side web applications	Used for building fast and scalable server-side networking applications
Splits a web application into MVC components	Generates database queries

Also Read: What is Angular?

16. Which database is more popularly used with Node.js?

MongoDB is the most common database used with Node.js. It is a NoSQL, cross-platform, document-oriented database that provides high performance, high availability, and easy scalability.

17. What are some of the most commonly used libraries in Node.js?

There are two commonly used libraries in Node.js:

- **ExpressJS** - Express is a flexible Node.js web application framework that provides a wide set of features to develop web and mobile applications.
- **Mongoose** - Mongoose is also a Node.js web application framework that makes it easy to connect an application to a database.

18. What are the pros and cons of Node.js?

Node.js Pros	Node.js Cons
Fast processing and an event-based model	Not suitable for heavy computational tasks
Uses JavaScript, which is well-known amongst developers	Using callback is complex since you end up with several nested callbacks
Node Package Manager has over 50,000 packages that provide the functionality to an application	Dealing with relational databases is not a good option for Node.js

Best suited for streaming huge amounts of data and I/O intensive operations	Since Node.js is single-threaded, CPU intensive tasks are not its strong suit

19. What is the command used to import external libraries?

The "require" command is used for importing external libraries. For example - **"var http=require ("HTTP")."** This will load the HTTP library and the single exported object through the HTTP variable.

Now that we have covered some of the important beginner-level Node.js interview questions let us look at some of the intermediate-level Node.js interview questions.

```
var http = require('http');
```

Node.js Interview Questions and Answers For Intermediate Level

20. What does event-driven programming mean?

An event-driven programming approach uses events to trigger various functions. An event can be anything, such as typing a key or clicking a mouse button. A call-back function is already registered with the element executes whenever an event is triggered.

21. What is an Event Loop in Node.js?

Event loops handle asynchronous callbacks in Node.js. It is the foundation of the non-blocking input/output in Node.js, making it one of the most important environmental features.

22. Differentiate between process.nextTick() and setImmediate()?

The distinction between method and product. This is accomplished through the use of nextTick() and setImmediate(). next Tick() postpones the execution of action until the next pass around the event loop, or it simply calls the callback function once the event loop's current execution is complete, whereas setImmediate() executes a callback on the next cycle of the event loop and returns control to the event loop for any I/O operations.

23. What is an EventEmitter in Node.js?

- EventEmitter is a class that holds all the objects that can emit events
- Whenever an object from the EventEmitter class throws an event, all attached functions are called upon synchronously

```
const EventEmitter = require('events');
class MyEmitter extends EventEmitter { }
const myEmitter = new MyEmitter();
myEmitter.on('event', () => {
console.log('an event occurred!');
});
myEmitter.emit('event');
```

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

The two types of API functions in Node.js are:

- Asynchronous, non-blocking functions
- Synchronous, blocking functions

25. What is the package.json file?

The package.json file is the heart of a Node.js system. This file holds the metadata for a particular project. The package.json file

is found in the root directory of any Node application or module

This is what a package.json file looks like immediately after creating a Node.js project using the command: npm init

You can edit the parameters when you create a Node.js project.

```
{
  "name": "node-npm",
  "version": "1.0.0",
  "description": "A demo application",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Taha",
  "license": "ISC"
}
```

26. How would you use a URL module in Node.js?

The URL module in Node.js provides various utilities for URL resolution and parsing. It is a built-in module that helps split up the web address into a readable format.

```
var url = require('url');
var adrs = 'http://localhost:8080/default.htm?year=2020&
month=march';
var que = url.parse(adrs, true);
console.log(que.host); //returns 'localhost:8080'
console.log(que.pathname); //returns '/default.htm'
console.log(que.search); //returns '?year=2020 and month
=march'
var quedata = que.query; //returns an object: { year: 2020,
 month: 'march' }
console.log(quedata.month); //returns 'march'
```

27. What is the Express.js package?

Express is a flexible Node.js web application framework that provides a wide set of features to develop both web and mobile applications

28. How do you create a simple Express.js application?

- The request object represents the HTTP request and has properties for the request query string, parameters, body, HTTP headers, and so on
- The response object represents the HTTP response that an Express app sends when it receives an HTTP request

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

app.get('/', function (req, res) {
  res.send('Hello World');
})

var server = app.listen(8081, function () {
  var host = server.address().address
  var port = server.address().port

  console.log("Example app listening at http://%s:%s", host, port)
})
```

29. What are streams in Node.js?

Streams are objects that enable you to read data or write data continuously.

There are four types of streams:

Readable – Used for reading operations

Writable − Used for write operations

Duplex − Can be used for both reading and write operations

Transform − A type of duplex stream where the output is computed based on input

30. How do you install, update, and delete a dependency?

31. How do you create a simple server in Node.js that returns Hello World?

- Import the HTTP module
- Use createServer function with a callback function using request and response as parameters.
- Type “hello world."
- Set the server to listen to port 8080 and assign an IP address

32. Explain asynchronous and non-blocking APIs in Node.js.

- All Node.js library APIs are asynchronous, which means they are also non-blocking
- A Node.js-based server never waits for an API to return data. Instead, it moves to the next API after calling it, and a notification mechanism from a Node.js event responds to the server for the previous API call

33. How do we implement async in Node.js?

As shown below, the async code asks the JavaScript engine running the code to wait for the request.get() function to complete before moving on to the next line for execution.

34. What is a callback function in Node.js?

A callback is a function called after a given task. This prevents any blocking and enables other code to run in the meantime.

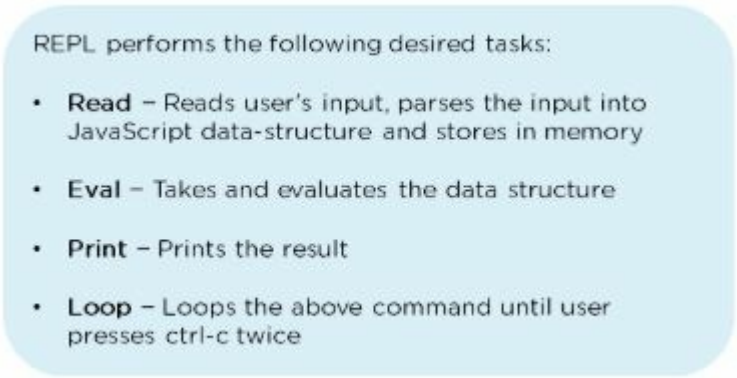
In the last section, we will now cover some of the advanced-level Node.js interview questions.

Node.js Interview Questions and Answers For Experienced Professionals

This section will provide you with the Advanced Node.js interview questions which will primarily help experienced professionals.

35. What is REPL in Node.js?

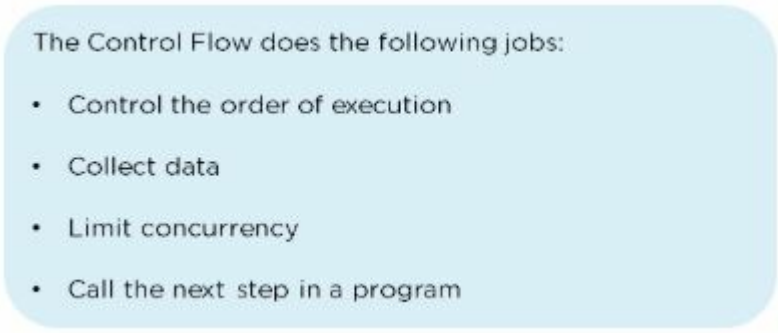
REPL stands for Read Eval Print Loop, and it represents a computer environment. It's similar to a Windows console or Unix/Linux shell in which a command is entered. Then, the system responds with an output



36. What is the control flow function?

The control flow function is a piece of code that runs in between several asynchronous function calls.

37. How does control flow manage the function calls?



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

fork()	**spawn()**
`child_process.fork(modulePath[, args][, options])`	`child_process.spawn(command[, args][, options])`
fork() is a particular case of spawn() that generates a new instance of a V8 engine.	Spawn() launches a new process with the available set of commands.
Multiple workers run on a single node code base for multiple tasks.	This method doesn't generate a new V8 instance, and only a single copy of the node module is active on the processor.

39. What is the buffer class in Node.js?

Buffer class stores raw data similar to an array of integers but corresponds to a raw memory allocation outside the V8 heap.

Buffer class is used because pure JavaScript is not compatible with binary data

40. What is piping in Node.js?

Piping is a mechanism used to connect the output of one stream to another stream. It is normally used to retrieve data from one stream and pass output to another stream

41. What are some of the flags used in the read/write operations in files?

- **r** – Open file for reading. An exception occurs if the file does not exist.
- **r+** – Open file for reading and writing. An exception occurs if the file does not exist.
- **w** – Open file for writing. The file is created (if it does not exist) or truncated (if it exists).
- **w+** – Open file for reading and writing. The file is created (if it does not exist) or truncated (if it exists).
- **a** – Open file for appending. The file is created if it does not exist.
- **a+** – Open file for reading and appending. The file is created if it does not exist.

42. How do you open a file in Node.js?

This is the syntax for opening a file in Node.js

```
fs.open(path, flags[, mode], callback)
```

43. What is callback hell?

- Callback hell, also known as the pyramid of doom, is the result of intensively nested, unreadable, and unmanageable callbacks, which in turn makes the code harder to read and debug
- improper implementation of the asynchronous logic causes callback hell

44. What is a reactor pattern in Node.js?

A reactor pattern is a concept of non-blocking I/O operations. This pattern provides a handler that is associated with each I/O operation. As soon as an I/O request is generated, it is then submitted to a demultiplexer

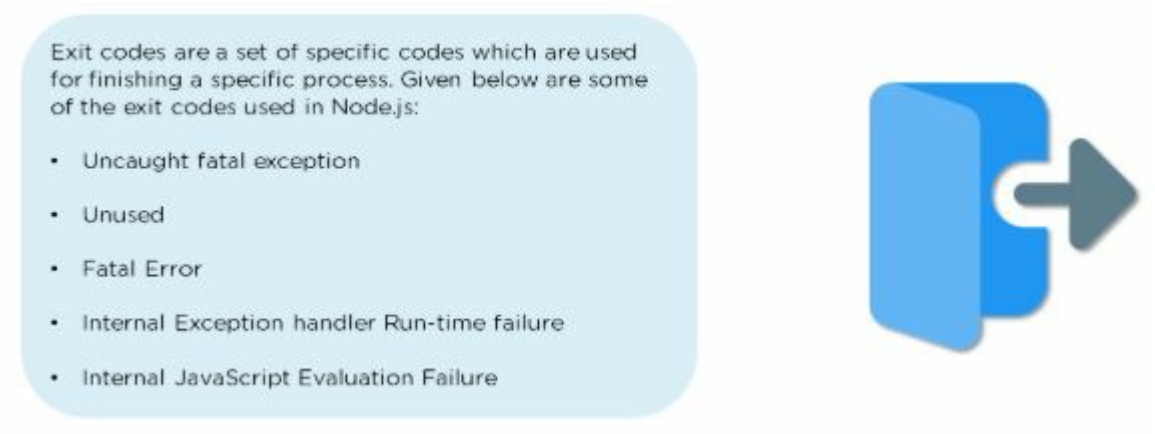
45. What is a test pyramid in Node.js?

- A test pyramid is a figure which explains the proportion of unit tests, integrations tests, and end-to-end tests that are required for the proper development of a project
- The components of a test pyramid are given below:
  - **Unit Tests:** They test the individual units of code in isolation
  - **Integrations Tests:** They test the integration among dissimilar units
  - **End-to-End (E2E) Tests:** They test the whole system, from the User Interface to the data store, and back.

46. For Node.js, why does Google use the V8 engine?

The V8 engine, developed by Google, is open-source and written in C++. Google Chrome makes use of this engine. V8, unlike the other engines, is also utilized for the popular Node.js runtime. V8 was initially intended to improve the speed of JavaScript execution within web browsers. Instead of employing an interpreter, V8 converts JavaScript code into more efficient machine code to increase performance. It turns JavaScript code into machine code during execution by utilizing a JIT (Just-In-Time) compiler, as do many current JavaScript engines such as SpiderMonkey or Rhino (Mozilla).

47. Describe Node.js exit codes.



48. Explain the concept of middleware in Node.js.

Middleware is a function that receives the request and response objects. Most tasks that the middleware functions perform are:

- Execute any code
- Update or modify the request and the response objects
- Finish the request-response cycle
- Invoke the next middleware in the stack

49. What are the different types of HTTP requests?

HTTP defines a set of request methods used to perform desired actions. The request methods include:

GET: Used to retrieve the data

POST: Generally used to make a change in state or reactions on the server

HEAD: Similar to the GET method, but asks for the response without the response body

DELETE: Used to delete the predetermined resource

50. How would you connect a MongoDB database to Node.js?

To create a database in MongoDB:

- Start by creating a MongoClient object
- Specify a connection URL with the correct IP address and the name of the database you want to create

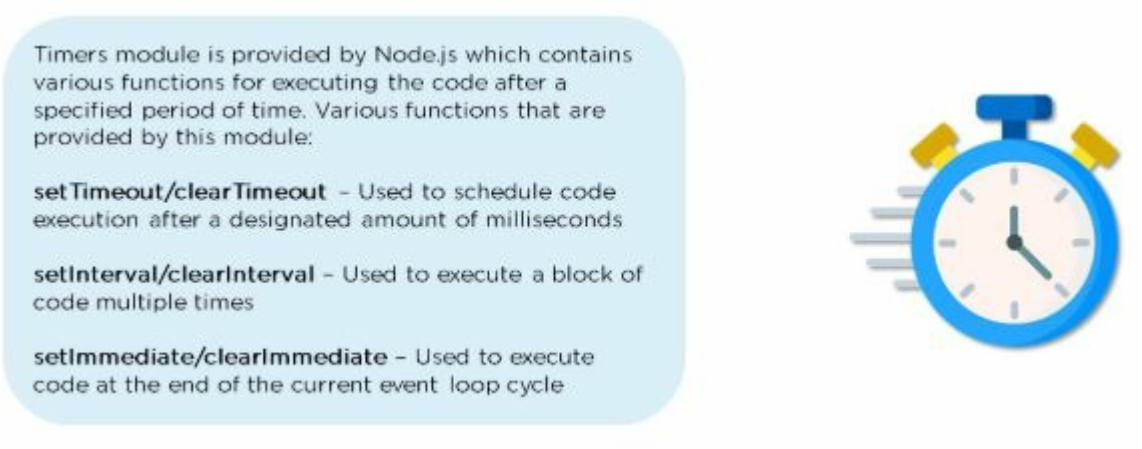
```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/mydb";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  console.log("Database created!");
  db.close();
});
```

51. What is the purpose of NODE_ENV?

52. List the various Node.js timing features.

As you prepare for your upcoming job interview, we hope that this comprehensive guide has provided more insight into what types of questions you'll be asked.



53. What is WASI, and why is it being introduced?

The WASI class implements the WASI system called API and extra convenience methods for interacting with WASI-based applications. Every WASI instance represents a unique sandbox environment. Each WASI instance must specify its command-line parameters, environment variables, and sandbox directory structure for security reasons.

54. What is a first-class function in Javascript?

First-class functions are a powerful feature of JavaScript that allows you to write more flexible and reusable code. In Node.js, first-class functions are used extensively in asynchronous programming to write non-blocking code.

55. How do you manage packages in your Node.Js project?

Managing packages in your Node.js project is done using the Node Package Manager (NPM), which allows you to install and manage third-party packages and create and publish your packages.

56. How is Node.js better than other frameworks?

Node.js is a server-side JavaScript runtime environment built on top of the V8 JavaScript engine, the same engine that powers Google Chrome. It makes Node.js very fast and efficient, as well as highly scalable.

57. What is a fork in node JS?

The Fork method in Node.js creates a new child process that runs a separate Node.js instance and can be useful for running CPU-intensive tasks or creating a cluster of Node.js servers.

58. List down the two arguments that async. First, does the queue take as input?

The async.queue function in Node.js takes two arguments as input: a worker function and an optional concurrency limit. It is used to create a task queue executed in parallel.

59. What is the purpose of the module.exports?

The module. exports object in Node.js is used to export functions, objects, or values from a module and is returned as the value of the require() function when another module requires a module.

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

In summary, several tools can be used in Node.js to ensure consistent code style and improve code quality, including ESLint, Prettier, and Jest.

61. What is the difference between JavaScript and Node.js?

Node.js is a runtime environment for executing JavaScript code outside of a web browser, while JavaScript is a programming language that can be executed in both web browsers and Node.js environments.

62. What is the difference between asynchronous and synchronous functions?

Synchronous functions block the execution of other code until they are complete, while asynchronous functions allow other code to continue executing while they are running, making them essential for writing scalable Node.js applications.

63. What are the asynchronous tasks that should occur in an event loop?

Asynchronous tasks that should occur in an event loop in Node.js include I/O operations, timers, and callback functions. By performing these tasks asynchronously, Node.js can handle a large number of concurrent requests without blocking the event loop.

64. What is the order of execution in control flow statements?

In Node.js, control flow statements are executed in a specific order. The order of execution is determined by the event loop. The event loop is a mechanism in Node.js that allows for the execution of non-blocking I/O operations.

65. What are the input arguments for an asynchronous queue?

An asynchronous queue in Node.js is a data structure that allows for the execution of functions in a specific order. Functions are added to the queue and are executed in the order that they were added. An asynchronous queue is useful when you want to execute a series of functions in a specific order.

66. Are there any disadvantages to using Node.js?

Node.Js is not suitable for CPU-intensive tasks. This is because Node.js is single-threaded, meaning it can only execute one task at a time. Node.js is not suitable for applications that require a lot of memory. This is because Node.js uses a lot of memory for each connection. If you have a large number of connections, it can quickly consume a lot of memory.

67. What is the primary reason for using the event-based model in Node.js?

The main reason to use the event-based model in Node.js is performance. The event-based model allows for non-blocking I/O

operations, which means that Node.js can handle a large number of connections without using a lot of resources.

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

Ajax and Node.js are two different technologies that are used for different purposes. Ajax is a client-side technology that allows for asynchronous communication between the client and the server. It is typically used to update parts of a web page without requiring a full page reload.

Node.js, on the other hand, is a server-side technology that is used for building fast, scalable, and efficient server-side applications. It is typically used for real-time applications, such as chat applications, online games, and streaming services.

69. What is the advantage of using Node.js?

Node.js is fast and scalable. Node.js is easy to learn and use. Node.js is well-suited for real-time applications, such as chat applications, online games, and streaming services. This is because Node.js can handle a large number of connections and can perform non-blocking I/O operations, which makes it ideal for real-time communication.

70. Does Node run on Windows?

Yes, Node.js runs on Windows. Node.js is a cross-platform runtime environment, which means that it can run on a variety of operating systems, including Windows, macOS, and Linux.

71. Can you access DOM in Node?

No, you cannot access the DOM in Node.js. The DOM is a browser-specific API that allows for the manipulation of HTML and XML documents. Since Node.js does not run in a browser, it does not have access to the DOM.

72. Why is Node.JS quickly gaining attention from JAVA programmers?

Node.js is quickly gaining attention from Java programmers because it is fast, scalable, and efficient. Java is a popular server-side technology, but it can be slow and resource-intensive. Node.js, on the other hand, is built on the V8 JavaScript engine, which is known for its speed and performance.

73. What are the Challenges with Node.js?

Node.js is single-threaded, which means that it can only execute one task at a time. Node.js is relatively new compared to other server-side technologies, such as Java and PHP. This means that there needs to be more support and more resources available for Node.js. Node.js is only suitable for applications that require a little memory.

74. What is "non-blocking" in node.js?

In Node.js, non-blocking refers to the ability of the runtime environment to execute multiple tasks simultaneously without waiting for the completion of one task before starting the next. This is achieved through the use of asynchronous I/O operations, which allow Node.js to handle multiple requests concurrently.

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

Node.js uses an event-driven, non-blocking I/O model that allows it to handle I/O operations more efficiently. By using callbacks, Node.js can continue processing other tasks while waiting for I/O operations to complete. This means that Node.js can handle multiple requests simultaneously without causing any delays. Additionally, Node.js uses a single-threaded event loop architecture, which allows it to handle a high volume of requests without any issues.

76. How can we use async await in node.js?

To use async/await in Node.js, you'll need to use functions that return promises. You can then use the async keyword to mark a function as asynchronous and the await keyword to wait for a promise to resolve before continuing with the rest of the code.

77. Why should you separate the Express app and server?

Firstly, separating your app and server can make it easier to test your code. By separating the two, you can test your app logic independently of the server, which can make it easier to identify and fix bugs.

Secondly, separating your app and server can make it easier to scale your application. By separating the two, you can run multiple instances of your app on different servers, which can help to distribute the load and improve performance.

Finally, separating your app and server can make it easier to switch to a different server if necessary. By keeping your app logic separate from your server logic, you can switch to a different server without having to make any major changes to your code.

78. Explain the concept of stub in Node.js.

In Node.js, a stub is a function that serves as a placeholder for a more complex function. Stubs are typically used in unit testing to replace a real function with a simplified version that returns a predetermined value. By using a stub, you can ensure that your unit tests are predictable and consistent.

79. What is the framework that is used majorly in Node.js today?

There are many frameworks available for Node.js, but the two most popular ones are Express and Koa.

80. What are the security implementations that are present in Node.js?

One of the most important security features in Node.js is the ability to run code in a restricted environment. This is achieved through the use of a sandboxed environment, which can help to prevent malicious code from accessing sensitive data or causing any damage to the system.

Another important security feature in Node.js is the ability to use TLS/SSL to encrypt data in transit. This can help to prevent eavesdropping and ensure that sensitive data is protected.

81. What is Libuv?

Libuv is a critical component of Node.js, and it's what makes it possible to handle I/O operations in a non-blocking and efficient manner.

82. What are global objects in Node.js?

Global objects in Node.js are objects that are available in all modules without the need for an explicit require statement. Some of the most commonly used global objects in Node.js include process, console, and buffer.

83. Why is assert used in Node.js?

An assert module is an important tool for writing effective tests in Node.js.

84. Why is ExpressJS used?

Express is a great choice for building web applications in Node.js, and its popularity and active community make it a safe and reliable choice for developers of all levels.

85. What is the use of the connect module in Node.js?

The Connect module can be used to handle different types of middleware, such as error-handling middleware, cookie-parsing middleware, and session middleware. Error-handling middleware is used to handle errors that occur during the request/response cycle. Cookie parsing middleware is used to parse cookies from the request header. Session middleware is used to manage user sessions.

86. What's the difference between 'front-end' and 'back-end' development?

Front-end developers focus on the client side of the application, while back-end developers focus on the server side of the application. Both roles are important for building a successful web application and require different skill sets and expertise.

87. What are LTS releases of Node.js?

LTS stands for Long-term support. LTS releases of Node.js are versions that are supported for an extended period, usually for 30 months from the time of release. These releases are typically more stable and reliable than non-LTS releases and are recommended for production use.

88. What do you understand about ESLint?

ESLint is a popular open-source tool that is used to analyze and flag errors and potential problems in JavaScript code.

89. Define the concept of the test pyramid. Please explain the process of implementing them in terms of HTTP APIs.

The test pyramid is a concept that is often used in software testing to illustrate the ideal distribution of different types of tests. The pyramid consists of three layers: unit tests, integration tests, and end-to-end tests. The idea is that the majority of tests should be at the unit level, with fewer tests at the integration and end-to-end levels.

To implement the test pyramid in terms of HTTP APIs, you can start by writing unit tests for each endpoint in the API. These tests should focus on testing the functionality of the endpoint in isolation without making any external requests or dependencies. Once the unit tests are passed, you can write integration tests that test the interaction between different endpoints and components in the API. Finally, you can write end-to-end tests that test the entire API, from the user interface to the database.

90. How does Node.js handle the child threads?

Node.js handles child threads by creating separate instances of the Node.js runtime environment that can be used to execute code in parallel with the main process.

91. What is an Event Emitter in Node.js?

An Event Emitter is a Node.js module that facilitates communication between objects in a Node.js application. It is an instance of the EventEmitter class, which provides a set of methods to listen for and emit events. In Node.js, events are a core part of the platform, and they are used to handle asynchronous operations.

92. How to Enhance Node.js Performance through Clustering?

Clustering can be used to improve the performance of HTTP servers, database connections, and other I/O operations. However, it is important to note that clustering does not guarantee a linear increase in performance.

93. What is a thread pool, and which library handles it in Node.js?

A thread pool is a collection of threads that are used to execute tasks in parallel. In Node.js, the thread pool is handled by the libuv library, which is a multi-platform support library that provides asynchronous I/O operations.

94. How are worker threads different from clusters?

Worker threads and clusters are two different approaches to leveraging the power of multiple CPUs in Node.js. While clusters create multiple instances of a Node.js process, each running on a separate CPU core, worker threads provide a way to create multiple threads within a single process.

95. How to measure the duration of async operations?

The console.time and console.timeEnd methods allow you to measure the duration of a block of code. The console.time method is used to start the timer and the console.timeEnd method is used to stop the timer and log the duration to the console.

The performance.now method provides a more precise way to measure the duration of async operations. It returns the current timestamp in milliseconds, which can be used to calculate the duration of a task.

96. How to measure the performance of async operations?

There are several tools and techniques you can use to measure performance, including using the built-in --prof flag, using the perf tool, and using third-party libraries like benchmark.js.

97. What are the types of streams available in Node.js?

There are four types of streams available in Node.js, including readable streams, writable streams, duplex streams, and transform streams.

98. What is meant by tracing in Node.js?

Tracing is a technique used in Node.js to profile the performance of an application. It involves recording the function calls and events that occur during the execution of the application and analyzing the data to identify performance bottlenecks.

99. Where is package.json used in Node.js?

The package.json file is located in the root directory of an application and it is used by the npm package manager to install and manage the dependencies of an application.

100. What is the difference between readFile and create Read Stream in Node.js?

Create Read Stream is a better option for reading large files, while the read file is a better option for small files. It is important to choose the right method based on the size of the file and the requirements of the application.

101. What is the use of the crypto module in Node.js?

The crypto module is widely used in Node.js applications to generate secure random numbers, create digital signatures, and verify signatures. It also provides support for various encryption algorithms such as AES, DES, and RSA.

102. What is a passport in Node.js?

Passport is a popular authentication middleware for Node.js. It provides a simple and modular way to implement authentication in Node.js applications. Passport supports many authentication mechanisms, including username/password, social logins like Facebook and Google, and JSON Web Tokens (JWTs).

103. How to get information about a file in Node.js?

In Node.js, the fs module provides methods for working with the file system. To get information about a file, you can use the fs. stat() method. The fs. stat() method returns an object that contains information about the file, such as the file size, creation date, and modified date.

104. How does the DNS lookup function work in Node.js?

In Node.js, the DNS module provides methods for performing DNS lookups. DNS stands for Domain Name System, and it is responsible for translating domain names into IP addresses. The DNS. lookup() method is used to perform a DNS lookup and resolve a domain name into an IP address.

105. What is the difference between setImmediate() and setTimeout()?

The setTimeout() method schedules code execution after a specified delay, measured in milliseconds. On the other hand, the setImmediate() method schedules code execution to occur immediately after the current event loop iteration completes. This means that setImmediate() has a higher priority than setTimeout().

106. Explain the concept of Punycode in Node.js.

Punycode is a character encoding scheme used in the domain name system (DNS) to represent Unicode characters with ASCII characters. It is used to encode domain names that contain non-ASCII characters, such as Chinese or Arabic characters.

107. Does Node.js provide any Debugger?

Yes, Node.js provides a built-in debugger that can be used to debug Node.js applications.

108. Is cryptography supported in Node.js?

Yes, Node.js provides built-in support for cryptography through the crypto module.

109. Why do you think you are the right fit for this Node.js role?

As a Node.js developer, I have experience in building scalable and efficient server-side applications using Node.js. I am a team player and have excellent communication skills. I believe that my experience and skills make me a strong candidate for this Node.js role.

110. Do you have any past Node.js work experience?

Yes, my past Node.js work experience has given me a solid foundation in building scalable and efficient server-side applications using Node.js.