Node.js Interview Questions for Freshers

Preparing You for Success in Node.js Interview

TOTAL 50 QUESTION AND ANSWER

Overview & Purpose

Navigating Node.js Interview Questions for Freshers. helping them prepare for job interviews

1. What is node.js ?

Node.js is a runtime environment built on Chrome's V8 JavaScript engine. It allows you to run JavaScript code outside of a web browser, making it possible to build server-side applications using JavaScript. Node.js provides an event-driven, non-blocking I/O model, which makes it lightweight and efficient for handling concurrent requests

2. What is single threaded ?

Node. js is known to be a single-threaded runtime environment, meaning that a program's code is executed line after line and there can't be two lines of a program running at the same time

.

3. What is asynchronous and synchronous ?

Synchronous : - means the code runs in a particular sequence of instructions given in the program. Each instruction waits for the previous instruction to complete its execution.

Asynchronous :- code execution allows executing next instructions immediately and doesn't block the flow because of previous instructions.

4. What is the promise ?

- promise is an object that represents the eventual completion or failure of an asynchronous operation. It is used to handle asynchronous operations in a more organized and readable way.
- With promises, you can write asynchronous code that looks more like synchronous code, making it easier to understand and maintain.
- Promises have three states: pending, fulfilled, and rejected.

  When a promise is pending, it means the asynchronous operation is still in progress.

  When a promise is fulfilled, it means the operation was successful and the promise returns a value.

  When a promise is rejected, it means the operation encountered an error and the promise returns an error.

- Promises provide methods like `.then()` and `.catch()` to handle the fulfillment or rejection of the promise.
  The `.then()` method is used to handle the successful completion of a promise.
  while the `.catch()` method is used to handle any errors that occur during the promise execution.

- Promises are commonly used when working with asynchronous tasks like making HTTP requests, reading and writing files, or querying databases.

5. What is a callback ?

In JavaScript and Node.js, a callback is a function that is passed as an argument to another function and is executed later, usually after an asynchronous operation has completed. The purpose of a callback is to handle the result or response of the asynchronous operation.

Callbacks are commonly used in scenarios like making API requests, reading files, or performing database queries, where the result may not be immediately available. Instead of blocking the execution and waiting for the operation to complete, a callback function is provided to be called once the operation is finished. This allows the code to continue executing other tasks while waiting for the asynchronous operation to complete.

6. What is the event loop ?

The event loop is a fundamental concept in Node.js that handles asynchronous operations and ensures that the program remains responsive. It is responsible for managing the execution of code and handling events in a non-blocking manner.

In Node.js, the event loop continuously checks for pending events and executes the associated callback functions when those events occur. It follows an event-driven architecture, where events are emitted by various sources like user input, network requests, timers, and file system operations. The event loop listens for these events and triggers the corresponding callbacks when the events are ready to be processed.

The event loop allows Node.js to handle multiple requests concurrently without blocking the execution of other tasks. It ensures that the program remains

efficient and responsive by efficiently managing the order of execution for different events and their associated callbacks.

7. key features of node.js ?

NodeJS Features

The following are the NodeJS features which make it popular for developers

- Cross-Platform : Node.js is cross-platform, which means you can develop and run applications on various operating systems, including Windows, macOS, and Linux.

- Scalability: Node.js is known for its ability to handle a large number of concurrent connections with high efficiency, making it suitable for building scalable applications.

- High Performance : Node.js uses an event-driven, non-blocking I/O model, which allows it to handle requests in a non-blocking manner. This results in faster response times and improved performance.

- JavaScript Runtime : Node.js uses JavaScript on both the server-side and client-side, which allows developers to use the same language and codebase for both ends of the application, leading to increased productivity and code reusability.

- NPM (Node Package Manager) : Node.js has a rich ecosystem of open-source packages available through npm (Node Package Manager). These packages provide ready-to-use solutions for various functionalities, saving development time and effort.

- Real-time Applications : Node.js is well-suited for building real-time applications, such as chat applications or collaborative tools, thanks to its event-driven architecture and support for WebSockets.

- Community Support : Node.js has a large and active community of developers, which means there are plenty of resources, tutorials, and libraries available to help with development challenges.

8. Data types in node.js ?

In Node.js, and JavaScript in general, data types can be categorized into two main groups: primitive and non-primitive (also known as reference) data types.

- Primitive Data Types:
- Number: Represents numeric data, both integers and floating-point numbers.
- String: Represents text data, enclosed in single (') or double ("") quotes.
- Boolean: Represents true or false values.
- Undefined: Represents a variable that has been declared but has no assigned value.
- Null: Represents the intentional absence of any object or value.
- Symbol(ES6): Represents a unique and immutable value, often used as object property keys.
- BigInt (ES11): Represents large integers that cannot be accurately represented with regular JavaScript numbers.

Primitive data types are immutable, meaning their values cannot be changed once they are created.

- Non-Primitive (Reference) Data Types:
- Object: Represents a collection of key-value pairs, like a dictionary or hash map. It includes arrays, functions, and custom objects.
- Array: Represents a collection of values stored in an ordered list.
- Function: Represents a reusable block of code that can be called with arguments.
- Date: Represents date and time values.
- Buffer (Node.js-specific): Used for handling binary data, such as reading and writing files or network streams.

Non-primitive data types are mutable, meaning their values can be changed after they are created.

9. What is Event-driven in Node.js ?

Event-driven in Node.js means that the execution of code is based on events and event handlers. It allows for better scalability and responsiveness in applications.

Event-driven in Node.js means responding to actions or occurrences (events) as they happen, without blocking the program's flow, using callbacks or event listeners to handle these events efficiently.

10. What are core modules in node.js ?

In Node.js, core modules are built-in modules that provide essential functionality for various tasks.

- FS (File System) : Provides file I/O operations, allowing you to read, write, and manipulate files and directories.
- Http : Enables you to create HTTP servers and make HTTP requests, making it essential for building web applications.
- Https : An extension of the `http` module that adds support for secure HTTP (HTTPS) connections.
- Url : Helps in parsing and formatting URLs, making it useful for working with web addresses.
- Path : Provides utilities for working with file and directory paths, ensuring compatibility across different platforms.
- OS(Operating System) : Gives information about the operating system on which Node.js is running, such as CPU architecture and memory usage.

- Events : Provides an event-driven architecture for handling events and creating custom event emitters.

- Util : Contains various utility functions, including those for object and function manipulation.

- Crypto : Offers cryptographic functions for hashing, encryption, and decryption.

- Querystring : Helps in parsing and formatting query strings commonly used in web applications.

- Buffer : Allows you to work with binary data directly, useful for I/O operations.

- Stream : Provides a framework for reading from and writing to streams of data, making it efficient for handling large datasets.

11. What is the callback queue in node.js ?

- In Node.js, the callback queue, also known as the "task queue" or "callback queue", is a data structure that holds the callback functions that are ready to be executed. When an asynchronous operation completes, its associated callback function is placed in the callback queue.

- The event loop in Node.js continuously checks the callback queue for any pending callbacks. If there are callbacks in the queue, the event loop picks them up one by one and executes them in the order they were added. This ensures that the callbacks are executed in a sequential and non-blocking manner.

- The callback queue plays a crucial role in handling asynchronous operations in Node.js. It allows the program to continue executing other tasks while waiting for the completion of asynchronous operations, resulting in improved efficiency and responsiveness.

12. What is V8 in JS ?

V8 is an open-source JavaScript engine developed by Google. It is written in C++ and is used to execute JavaScript code in web browsers and server-side environments like Node.js.

13. What is the LIBUV library ?

LIBUV is a multi-platform C library used by Node.js to handle asynchronous tasks, manage events, and provide cross-platform support for efficient I/O operations. It's a crucial part of Node.js that enables its non-blocking, event-driven architecture to work consistently on different operating systems.

14. What is callback hell ?

Callback hell, in Node.js and JavaScript, refers to the situation where you have multiple layers of nested callback functions within your code, making it hard to read and manage due to excessive indentation and complexity.

To avoid callback hell, developers often use techniques like modularization, Promises, or async/await to write cleaner and more readable asynchronous code.

15. Explain the FS module ?

The FS (File System) module in Node.js is a built-in module that provides functions for working with the file system. It allows you to perform various operations on files and directories, including reading and writing files, creating and deleting directories, and managing file metadata. Here are some key features and functions of the FS module.

16. What is REPL in node js ?

REPL stands for Read Evaluate Print Loop, and it is a programming language environment (basically a console window) that takes a single expression as user input and returns the result back to the console after execution. The REPL session provides a convenient way to explore the Node.js features in a quick way.

Here's how REPL works:

- Read: You enter JavaScript code or expressions one line at a time. The REPL reads and interprets your input.
- Eval: The entered code is evaluated or executed by the Node.js interpreter.
- Print: The result of the evaluated code is printed to the console.
- Loop: The REPL then goes back to the beginning, allowing you to enter more code or expressions.

17. What is the stream in node.js ?

The Node.js stream feature makes it possible to process large data continuously in smaller chunks without keeping it all in memory. In other words, you can use streams to read from or write to a source continuously instead of using the traditional method of processing all of it at once.

One benefit of using streams is that it saves time, since you don't have to wait for all the data to load before you start processing. This also makes the process less memory-intensive.

There are namely four types of streams in Node.js.

- Writable: We can write data to these streams.
  Example: fs.createWriteStream().
- Readable: We can read data from these streams.
  Example: fs.createReadStream().
- Duplex: Streams that are both Writable as well as Readable.
  Example: net.socket.
- Transform: Streams that can modify or transform the data as it is written and read. Example: zlib.createDeflate.

18. What is Piping in Node.js ?

In Node.js, "pipe" is a method that allows you to easily transfer data from one readable stream to another writable stream. It acts like a data connector,

simplifying the process of moving data between different sources and destinations, such as reading from a file and writing to a network connection.

19. What is jwt token in node.js ?

A JSON Web Token, popularly known as JWT, is an open standard that defines a compact way for securely sharing information between two parties: a client and a server.

20. What is Piping in Node.js ?

In Node.js, "pipe" is a method that allows you to easily transfer data from one readable stream to another writable stream. It acts like a data connector, simplifying the process of moving data between different sources and destinations, such as reading from a file and writing to a network connection.

21. Explain microservices in node.js ?

Microservices in Node.js refers to a software architectural approach where a large application is divided into smaller, independent services, each running its own Node.js server. These services communicate over a network, allowing for scalability, flexibility, and easier development and deployment. Node.js's event-driven nature and lightweight characteristics make it a good fit for building responsive and efficient microservices.

22. What is the purpose of the Node Package Manager?

Node Package Manager (NPM) is a command line tool that installs, updates or uninstalls Node. js packages in your application. It is also an online repository for open-source Node. js packages.

23. How can you include external libraries in your Node.js application?

To include external libraries in a Node.js application:

- Use NPM (Node Package Manager).
- Run `npm install library-name` to install the library.
- Require the library in your code with `require('library-name')`.
- Manage dependencies in `package.json`.
- Update dependencies with `npm update`.

24. What is the difference between require and import in Node.js?

`require` is the traditional method for importing modules. It uses CommonJS syntax and is compatible with both Node.js and browsers (when using tools like Browserify). `require` is dynamic, allowing conditional loading of modules based on runtime logic. It primarily deals with object-based exports.

`import` is the ES6 way of importing modules, using a more modern syntax. While Node.js has added support for ES6 modules, `import` is considered static, expecting module names to be string literals. It explicitly supports both named and default exports.

25. What is the global object in Node.js?

In Node.js, the global object is an object that provides access to global variables and functions throughout the Node.js runtime environment. It is similar to the window object in the browser when working with JavaScript on web pages.

global Object: In Node.js, the global object is named global. You can access it directly without needing to declare global explicitly. For example, you can use global.console to access the console object.

26. What is the purpose of the package.json file in a Node.js project?

The package.json file in a Node.js project is used to manage project dependencies, scripts, and metadata. It includes information about the project, such as its name, version, and dependencies.

27. Can you explain the role of modules in Node.js and how to use them?

Modules in Node.js are reusable blocks of code that encapsulate related functionality. They can be created using the module.exports or exports object and can be imported using the require function.

28. What is Express.js and how is it used in Node.js?

Express.js is a popular web application framework for Node.js. It provides a set of features and tools to build robust and scalable web applications..

29. How do you handle file uploads in Node.js?

File uploads in Node.js can be handled using libraries like multer. Multer allows you to handle multipart/form-data and supports various storage options.

30. Can you explain the concept of middleware in Node.js and provide an example?

Middleware in Node.js is a function that sits between the server and the route handler. It can perform tasks like logging, authentication, error handling, etc. An example of middleware in Express.js is the body-parser middleware for parsing request bodies.

31. How do you deploy a Node.js application to a production environment?

Deploying a Node.js application to a production environment involves steps like setting up a server, installing dependencies, configuring environment variables, and running the application using a process manager like PM2.

32. What is the purpose of the "process.argv" property in Node.js and how can you use it?

The "process.argv" property in Node.js provides access to the command-line arguments passed to the Node.js process. It is an array where the first element is the path to the Node.js executable, and the subsequent elements are the command-line arguments. You can use it to retrieve and process the values passed as arguments when running a Node.js script from the command line. For example, if you run a script with the command "node myScript.js arg1 arg2", you can access "arg1" and "arg2" using "process.argv[2]" and "process.argv[3]".

33. How can you handle authentication and authorization in a Node.js application?

Authentication and authorization in a Node.js application can be handled using middleware like passport.js or by implementing custom authentication logic

using libraries like bcrypt for password hashing and JWT for token-based authentication.

34. What are the differences between callbacks, promises, and async/await in Node.js?

Callbacks are a traditional way of handling asynchronous operations in Node.js. Promises provide a more structured approach to handling asynchronous operations, while async/await is a syntax sugar built on top of promises to write asynchronous code in a more synchronous-looking manner..

35. What is the purpose of the util module in Node.js?

The purpose of the `util` module in Node.js is to provide various utility functions that can be helpful in different scenarios. It offers functions for debugging, formatting, and working with objects. For example, you can use `util.format()` to format strings, `util.inspect()` to inspect objects, and `util.promisify()` to convert callback-based functions into Promise-based functions. The `util` module provides a range of useful functions that can simplify development tasks.

36. What is the purpose of the net module in Node.js?

The `net` module in Node.js provides an asynchronous network API for creating servers and clients. It allows you to create TCP servers and clients, enabling you to build network applications like chat servers, file transfer protocols, and more. It provides functions and classes for creating and managing sockets, handling data streams, and listening for incoming connections. The `net` module is a core module in Node.js, so you don't need to install any additional packages to use it.

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

The setImmediate() and setTimeout() functions in Node.js are used to schedule the execution of code, but they have some differences:

- setImmediate() : schedules the code to be executed immediately after the current phase of the event loop, before any other I/O events or timers. It is typically used to defer the execution of a callback function to the next iteration of the event loop.
- setTimeout() : schedules the code to be executed after a specified delay, measured in milliseconds. It allows you to delay the execution of a callback function by a certain amount of time.
- setImmediate()` executes the code immediately after the current phase of the event loop, while setTimeout() delays the execution of the code by a specified amount of time.

38. How can you secure your Node.js application?

To secure your Node.js application, you can follow these best practices:

- Keep dependencies updated: Regularly update your application's dependencies to ensure you have the latest security patches.
- Implement input validation : Validate and sanitize all user inputs to prevent common security vulnerabilities like SQL injection and cross-site scripting (XSS) attacks.
- Use secure authentication : Implement strong password hashing algorithms, enforce password complexity requirements, and use techniques like token-based authentication or OAuth for secure user authentication.

- Protect against cross-site scripting (XSS) : Use output encoding or template engines that automatically escape user-generated content to prevent XSS attacks.
- enable secure communication : Use HTTPS and SSL/TLS certificates to encrypt data transmitted between the client and server, protecting it from eavesdropping and tampering.
- Implement access control and authorization : Restrict access to sensitive resources and ensure that users have appropriate permissions to perform certain actions.
- Log and monitor security events : Implement logging and monitoring to detect and respond to security incidents in real-time.
- Regularly test for vulnerabilities : Conduct security audits, penetration testing, and code reviews to identify and fix potential security vulnerabilities.

39. What is the purpose of the querystring module in Node.js?

The `querystring` module in Node.js provides utilities for working with query strings. It allows you to parse and stringify URL query strings, which are commonly used to pass data between the client and server in web applications.

The `querystring` module provides functions like `querystring.parse()` to parse a query string into a JavaScript object, and `querystring.stringify()` to convert a JavaScript object into a query string. This module also provides other utility functions for encoding and decoding query strings.

By using the `querystring` module, you can easily handle and manipulate URL query strings in your Node.js applications.

40. What is a callback function hell? How can it be avoided?

Callback hell refers to a situation in asynchronous programming where multiple nested callback functions make the code difficult to read, understand, and maintain. It occurs when there are many asynchronous operations that depend on each other, leading to deeply nested callbacks.

To avoid callback hell, you can use the following techniques:

- Use Promises : Promises provide a more structured and readable way to handle asynchronous operations. They allow you to chain multiple asynchronous operations together using methods like `then()` and `catch()`, resulting in cleaner and more maintainable code.
- Use Async/Await : Async/await is a modern syntax introduced in JavaScript that allows you to write asynchronous code in a synchronous style. It simplifies the handling of asynchronous operations by using the `async` keyword to define an asynchronous function and the `await` keyword to wait for the result of a promise.
- Modularize Code : Break down complex logic into smaller, reusable functions. This helps to reduce the nesting of callbacks and makes the code more modular and easier to understand.
- Use Control Flow Libraries : Libraries like `async.js` and `q.js` provide control flow mechanisms that help manage asynchronous operations. They offer functions like `parallel()`, `series()`, and `waterfall()` to handle multiple asynchronous tasks in a more organized way.

41. Explain the purpose of the url module in Node.js

The `url` module in Node.js provides utilities for working with URLs. It allows you to parse, format, and manipulate URL strings.

- With the `url` module, you can parse a URL string into its individual components like protocol, hostname, port, path, query parameters, and fragment identifier. This can be done using the `url.parse()` function.
- You can also format and construct URL strings from their individual components using the `url.format()` function.
- Additionally, the `url` module provides other utility functions like `url.resolve()` to resolve relative URLs, `url.resolveObject()` to resolve URLs as objects, and `url.parseQuery()` to parse query strings.
- By using the `url` module, you can easily work with URLs in your Node.js applications.

42. What is the difference between a module and a package in Node.js?

- In Node.js, a module refers to a single file or a set of related files that contain reusable code. It encapsulates functionality and provides an interface to interact with that functionality. Modules can be created by developers and can be used in other parts of the application by importing and requiring them.
- On the other hand, a package in Node.js is a directory that contains one or more modules along with a `package.json` file. The `package.json` file holds metadata about the package, including its name, version, dependencies, and other configuration details. Packages can be published and shared with others through package registries like npm. They can also be installed and used in other Node.js applications to add additional functionality or third-party libraries.
- a module refers to a single file or a set of related files that provide a specific functionality, while a package is a directory that contains one or more modules along with metadata and can be published, shared, and installed using a package manager like npm.

43. What is WebSocket in node.js?

- WebSockets in Node.js are a communication protocol that enables real-time, bidirectional communication between a client and a server over a single, long-lived connection. Unlike traditional HTTP requests, which are stateless and require the client to initiate a new request for each interaction, WebSockets allow for continuous, two-way communication.
- In Node.js, you can implement WebSockets using the `ws` module. This module provides a WebSocket server and client implementation. With the `ws` module, you can create a WebSocket server that listens for incoming connections and handles messages from clients.
- WebSockets are commonly used in applications that require real-time updates, such as chat applications, collaborative tools, and real-time analytics.

44. What is the purpose of the Buffer class in Node.js?

- The Buffer class in Node.js is used to handle binary data, such as reading from or writing to streams, working with file systems, or interacting with network protocols. It provides a way to manipulate and store raw data in memory.
- Buffers are instances of the Buffer class and can be created in various ways, such as from strings, arrays, or directly from binary data. They can be used to represent data in different encodings, such as UTF-8, ASCII, or Base64.

45. What is a first class function in Javascript?

In JavaScript, a first-class function means that functions are treated as values. This allows functions to be assigned to variables, passed as arguments to other functions, and returned from other functions. It gives JavaScript the ability to use functions as data, making it a versatile and powerful language.

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

- In Node.js, there are two types of API functions: synchronous (blocking) and asynchronous (non-blocking) functions.
- Synchronous functions block the execution of code until the function call is complete. They can be easier to understand and use, but they can also cause the program to become unresponsive if the operation takes a long time to complete.
- Asynchronous functions, on the other hand, allow the program to continue executing while waiting for the function call to complete. They are often preferred for handling I/O operations or any tasks that may take a significant amount of time. Asynchronous functions typically use callbacks, promises, or async/await to handle the asynchronous behavior.

47. Where is Node.js used?

Node.js Used for.

- Building server-side web applications and APIs.
- Developing real-time web applications, including chat applications and online gaming platforms.
- Creating single-page applications (SPAs) using frameworks like Express.js and Nest.js.
- Server-side rendering (SSR) with frameworks like Next.js.

48. Why does Google use the V8 engine for Node.js?

Google makes use of the V8 engine because it can easily convert JavaScript into a low-level language. This is done to provide high performance during the execution of an application and also to provide users with real-time abilities to work with the application.

49. What is the purpose of the crypto module in Node.js?

The crypto module in Node.js provides cryptographic functionality, allowing you to securely encrypt and decrypt data, generate secure hashes, sign and verify digital signatures, and more. It offers a wide range of cryptographic algorithms and methods to ensure data integrity, confidentiality, and authenticity. With the crypto module, you can implement secure communication, password hashing, token generation, and other security-related features in your Node.js applications. It's a powerful tool for safeguarding sensitive information and protecting the integrity of your data.

50. What is a passport in Node.js?

In Node.js, Passport is a popular authentication middleware that provides a simple and flexible way to authenticate users in web applications. It supports various authentication strategies, such as username/password, social media logins (like Facebook or Google), and more. Passport simplifies the authentication process by abstracting the underlying details and providing a consistent interface for authentication across different providers. It also integrates well with frameworks like Express.js, making it easy to implement authentication in Node.js applications.

Important note

Node.js is a JavaScript runtime environment.

If any question is missing then see it in JavaScript PDF. Thank you……!

Created by Durgesh Bisen

Preparing You for Success in Node.js Interview

CONNECT FOR MORE