1. what is Node.js?

Node.js is an open-source Javascript run-time built on the Chrome V8 Javascript engine. It
allows developers to execute Javascript code on the server-side, enabling them to build
scalable, efficient, and real-time applications.

2. How does Node.js handle asynchronous operations?

Node.js uses an event-driven, non-blocking I/O model. Asynchronous operations are managed
using callbacks, promises, or async/await, allowing the server to handle multiple requests
concurrently without blocking the execution thread.

3. What is the Node.js package manager?

The Node.js package manager is called npm (Node Package Manager). It's used to install,
manage, and share packages (libraries) that extend Node.js functionality.

4. How can you include external modules in Node.js?

you can include external modules using the require() function. For example, to include the
express module: const express = require('express');

5. what is the purpose of the fs module in Node.js?
The fs module, short for "file system," provides methods for interacting with the file system, such as reading and writing files, creating directories, and managing paths.

6. what is the difference between require() and import in Node.js?
require() is the commonJS ways of importing modules in Node.js, whereas import is part of the ECMAScript module system introduced in modern JavaScript (ES6+). Node.js supports both, but import requires certain configuration and might not work in all cases without transpilation.

7. How can you create a simple HTTP server using Node.js?
you can create a simple HTTP server using the built-in http module in Node.js

8. what is the purpose of the global object process in Node.js?
The process object provides information and control over the Node.js process running on the system. It contains properties like argv (command-line arguments), env (environment variables), and methods like exit() to terminate the

process.

9. How can you handle errors in Node.js?

Errors in Node.js can be handled using try...catch blocks, or by attaching error event listeners to
asynchronous operations. Additionally, Node.js provides the process.on('uncaughtException')
event to catch unhandled exceptions.

10. What is middleware in the context of Express.js?

Middleware in Express.js are functions that are executed in the request-response cycle. They
can modify the request and response objects, handle authentication, logging, and other
common tasks. Middleware can be added globally or to specific routes using app.use().

11. What is npm and how is it used?

npm (Node Package Manager) is the default package manager for Node.js. It's used to install,
manage, and share JavaScript packages (libraries) from the npm registry. You can use
commands like npm install, npm uninstall, and npm update to manage packages in your Node.js

projects.

12. How can you handle routes and URLs in Node.js using Express.js?
Express.js provides a routing system that allows you to define routes for different URLs and
HTTP methods. You can use the app.get(), app.post(), app.put(), and app.delete() methods to
define routes and their corresponding handlers.

13. What is middleware in the context of Express.js?
Middleware in Express.js are functions that execute during the request-response cycle. They
can perform tasks like logging, authentication, data parsing, and more. Middleware can be
added globally or to specific routes using the app.use() method.

14. What is the purpose of the next() function in Express.js middleware?
In Express.js middleware, the next() function is used to pass control to the next middleware in
the stack. If not called, the request-response cycle may get stuck. It's crucial to call next() to
ensure the middleware chain

continues.

15. How can you read environment variables in Node.js?

You can access environment variables using the process.env object in Node.js. For example, to
read the value of an environment variable named
DATABASE_URL, you would use
process.env.DATABASE_URL.

16. How does Node.js handle modules and dependencies?

Node.js uses the CommonJS module system to manage modules. Each file is treated as a
separate module. You can use the require() function to import modules, and you can create
your own modules using the module.exports object.

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

The Buffer class is used to work with binary data in Node.js, including reading from or writing to
streams and handling network protocols. It's particularly useful for dealing with file I/O and
binary data manipulation.

18. How can you perform asynchronous operations in a loop in Node.js?

To perform asynchronous operations in a loop, you can use techniques like callbacks, promises,
or async/await. Promises and async/await are especially helpful for writing more readable and
maintainable code when dealing with async loops.

19. What is the event loop in Node.js?
The event loop in Node.js is responsible for managing the execution of asynchronous code. It
constantly checks the call stack and the message queue, moving tasks from the queue to the
stack when the stack is empty, ensuring non-blocking behavior.

20. What is a module bundler, and why might you use one in Node.js?
A module bundler like Webpack or Parcel is used to bundle JavaScript modules and their
dependencies into a single file (or multiple files) for deployment. It's useful for optimizing
performance by reducing the number of network requests and managing assets.

21. What is the purpose of the require.resolve() method in Node.js?
The require.resolve() method is used to determine the full path of a

module without actually
loading the module into memory. It's commonly used when you need
to know the location of a
module's file.

22. How can you handle file uploads in Node.js with Express.js?
To handle file uploads in Express.js, you can use middleware like
multer. Multer simplifies
handling multipart/form-data, which is typically used for file
uploads. It allows you to access
uploaded files and fields in your route handlers.

23. What is a Promise in Node.js?
A Promise is a built-in object in Node.js that represents the
eventual completion or failure of an
asynchronous operation. It simplifies handling asynchronous code by
providing methods to
handle success and failure scenarios.

24. What is the async keyword in JavaScript functions?
The async keyword is used to define an asynchronous function in
JavaScript. An asynchronous
function always returns a Promise, and you can use the await
keyword within it to pause
execution until a Promise is

resolved.

25. What is the purpose of the child_process module in Node.js?
The child_process module in Node.js is used to create and manage child processes. It allows
you to execute commands or other Node.js scripts as separate processes, enabling parallel
execution and interaction with external programs.

26. How can you handle cross-origin resource sharing (CORS) in Node.js?
To handle CORS in Node.js, you can use middleware like cors to configure your Express.js
application to allow or restrict cross-origin requests. This helps control which origins are
permitted to access your resources.

27. What is the difference between process.nextTick() and setTimeout()?
process.nextTick() schedules a callback to be executed on the next iteration of the event loop,
immediately after the current operation completes. setTimeout() schedules a callback to be
executed after a specified delay, allowing other operations to run in the

meantime.

28. How can you deploy a Node.js application to a server?

You can deploy a Node.js application by:

Setting up a server environment (e.g., using a cloud service like AWS, Heroku, or DigitalOcean).

Uploading your application files to the server.

Installing Node.js and the necessary dependencies on the server.

Running your application using a process manager like PM2 or systemd.

29. What is the purpose of the os module in Node.js?

The os module in Node.js provides methods to interact with the operating system. It offers
information about the server's CPU, memory, network interfaces, and more.

30. What is an EventEmitter in Node.js?

An EventEmitter is a built-in class in Node.js that allows you to create objects that emit named
events. you can attach event listeners to these objects to respond t specific events when they
occur.

31. How can you read and write files asynchronously in Node.js?

Node.js provides the fs (file system) module for reading and writing files asynchronously. you
can use methods like fs.readFile() and fs.writeFile() to perform file operations without blocking
the event loop.

32. what is 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 URLs,
extract components like hostname and pathname, and create URL objects.

33. How do you handle environment-specific configuration in a Node.js application?
Environment-specific configuration can be managed using environment variables. For example,
you can use the process.env object to access environment variables like database credentials,
API keys, and configuration settings.

34. What is the role of a reverse proxy in Node.js deployments?
A reverse proxy acts as an intermediary server that forwards client requests to the appropriate
backend server (Node.js application). It can enhance security, load

balancing, and provide
additional features like caching and SSL termination.

35. How can you handle sessions and cookies in a Node.js application?
you can use middleware like express-session in Express.js to manage sessions and cookies.
This middleware handles session data and can store it in memory, or disk, or in a database.

36. What is the purpose of the cluster module in Node.js?
The cluster module allows you to create multiple child processes (workers) for a Node.js
application. This helps utilize multiple CPU cores, improving performance and allowing better
handling of concurrent requests.

37. How can you handle real-time communication in Node.js applications?
For real-time communication, you can use libraries like socket.io that provide WebSocket-based
communication between the server and clients. WebSocket enables bi-directional
communication for features like real-time chats and live updates.

38. What is the process.argv property in Node.js?

The process.argv property is an array containing command-line arguments passed to a Node.js
script. The first element is the path to the Node.js executable, the second element is the script
file's path, and subsequent elements are arguments passed when running the script.

39. How can you securely store sensitive information like API keys in a Node.js
application?
you can use environment variables to store sensitive information outside of your codebase.
Libraries like dotenv can help manage environment variables in development environments. In
production, use the environment provided by your hosting platform.

40. What is the purpose of the crypto module in Node.js?
The crypto module provides cryptographic functionality for generating secure hashes,
encryption, decryption, digital signatures, and more. It's often used to enhance the security of
data in Node.js applications.

41. How can you handle asynchronous errors in Node.js?

Asynchronous errors can be handled using the try...catch construct within asynchronous
functions. Additionally, using error-first callbacks, promises wit hcatch(), and try...catch with
async/await can help manage errors effectively.

42. what is a stream in Node.js?
A stream is a way to handle reading from or writing to data sources in a continuous manner,
piece by piece, rather than loading the entire data into memory. Streams are crucial for efficient
I/O operations, especially when dealing with large files or network data.

43. How do you manage database operations in Node.js applications?
Node.js can interact with databases using various libraries like mongoose (for MongoDB),
sequelize (for SQL databases), and pg-promise (for PostgreSQL). These libraries provide
abstractions and methods for querying, inserting, updating, and deleting data in databases.

44. what is middleware in the context of Express.js routing?
Middleware in Express.js can be thought of as functions that sit

between the initial request and
the final response. They can perform tasks like logging,
authentication, and validation before
passing the request to the route handler.

45. How can you improve the performance of a Node.js application?
Performance improvements can include optimizing database queries,
using caching
mechanisms (e.g., Redis), implementing load balancing with multiple
instances, using a reverse
proxy, minifying code, and enabling compression.
46. What is the purpose of the http module in Node.js?
The http module in Node.js provides a way to create and manage
HTTP servers and clients. It
enables you to create web servers, handle requests, and send
responses using HTTP methods
like GET, POST, PUT, and DELETE.

47. How can you handle authentication in Node.js applications?
Authentication can be handled using libraries like passport or
jsonwebtoken. These libraries
help manage user authentication and session management, enabling
secure access to
resources.

48. What are template engines in Node.js?
Template engines are used to generate dynamic HTML content by combining data with
template files. Popular template engines for Node.js include EJS, Handlebars, and Pug
(formerly known as Jade).

49. How can you schedule tasks to run at specific intervals in Node.js?
You can use the built-in setInterval() function to repeatedly execute a function at specified
intervals. For more advanced scheduling, you can use libraries like node-schedule or nodecron.

50. What is the purpose of the util module in Node.js?
The util module provides utility functions that are helpful for working with JavaScript objects and
functions. It includes functions for inheritance, debugging, and formatting.

51. What is a template literal in Node.js?
A template literal is a string literal that supports embedded expressions using backticks (`). It
allows you to include variables and expressions directly within the

string.

52. what is the purpose of the assert module in Node.js?
The assert module provides assertion testing to verify that values meet expected conditions. It's
often used for writing unit tests to ensure code correctness.

53. How can you handle promises that need to be resolved or rejected in a specific
order?
you can use Promise.all() to handle multiple promises concurrently and wait for all of them to
resolve or reject. For sequential execution, you can use chaining or async/await.

54. What is the Node.js event loop and how does it work?
The Node.js event loop is a fundamental part of the runtime that handles asynchronous
operations. It continually processes events from the event queue, executing callbacks and
resolving promises when their conditions are met.

55. How can you secure a Node.js application against common vulnerabilities?
To secure a Node.js application, you should:

- Regularly update dependencies to patch security vulnerabilities.
- Implement input validation to prevent injections and XSS attacks.
- use libraries like helmet to set security-related HTTP headers.
- use authentication and authorization mechanisms for user access.
- Validate and sanitize user inputs to prevent malicious actions.

56. How does error handling in async/await differ from using callbacks or promises?

In async/await, error handling resembles synchronous code with try...catch blocks. Errors thrown within an async function can be caught using surrounding try...catch blocks.

57. What are WebSocket connections, and how can you implement them in Node.js?

WebSocket connections provide full-duplex communication between a client and a server over a single connection. Libraries like ws allow you to create WebSocket servers and clients in Node.js for real-time bidirectional communication.

58. How can you manage environment-specific configurations using .env files?

Libraries like dotenv allow you to store environment-specific

configuration variables in a env
file. These variables can be loaded using process.env to keep
sensitive information separate
from your codebase.

59. How can you profile and debug Node.js applications?
you can use built-in tools like the Node.js Debugger (node inspect or
node --inspect) for
debugging. Profiling can be done using the --prof flag with node
and analyzing the
generated cpuprofile or heapprofile files.

60. How do you handle memory leaks in long-running Node.js
applications?
- use tools like the --inspect flag or Node.js built-in heapdump to diagnose leaks.
- Avoid global variables and circular references.
- unsubscribe from event listeners when they're no longer needed.
- Release resources and close connections properly.
- use heap snapshots and monitoring tools to detect trends over time.

61. What is Authentication?
Authentication is the process of identifying someone's identity by
assuring that the person is the

same as what he is claiming for. (Identification) It is used by both servers and clients.

- client ⇒ The client uses it when he wants to know that it is the same server that it claims to be.
- server ⇒

- The server uses it when someone wants to access the information, and the server needs to know who is accessing the information.
- It is done mostly by using the username and password.
- Other ways of authentication can be done using cards, retina scans, voice recognition, and fingerprints.
- Most common Authentication used
- Signup/Register/Create Account
- Login.
- If the Email and Password are correct only then Login.
- we have to match it with the data that is there in the database.
- For Authentication, we give them an access token that is unique when they are logged in.For that, we use JWT(JSON Web Token) Authentication

62. what is Authorization?

Authorization is the process of granting someone to do something. It means it is a way to check
if the user has permission to use a resource or not.

- It usually works with authentication so that the system could know who is accessing the information.
- In authorization, data is provided through the access tokens.

63. How do you do role-based authentication?

- Role-based authentication is a common technique used to manage access control in systems with multiple users. In this approach, each user is assigned a specific role or set of roles that determine their level of access to various resources within the system.
- Identify the different roles that will be needed to access different resources within the system. For example, a system might have roles such as "teachers" "students", or "admin".

64. What is hashing?

- Hashing is a process of transforming plain text into a fixed-length sequence of

characters, known as a hash.

- Hashing is used to store passwords securely in a database.

65. What is encryption?

- Encryption is the process of converting plain, readable data or information into an encoded form so that it can only be accessed and understood by authorized parties who have the decryption key.
- The purpose of encryption is to protect data from unauthorized access, theft, or modification.
- The coded message is called cipher text

65. What is encryption?

- Encryption is the process of converting plain, readable data or information into an encoded form so that it can only be accessed and understood by authorized parties who have the decryption key.
- The purpose of encryption is to protect data from unauthorized access, theft, or modification.
- The coded message is called cipher

tept

67. what is salt?

- salt is a random sequence of characters that is added to the password before it is hashed.
- The purpose of the salt is to add an extra layer of security to the password hash.

68. what is JWT?

- JWT stands for JSON Web Token. It is a compact, URL-safe means of representing claims to be transferred between two parties. JWTs are used for authentication and authorization in web applications and APIs.
- They contain JSON objects which have the information that need to be shared. Each JWT is also signed using hashing to ensure that the JSON contents (also known as JWT claims) cannot be altered by the client or a malicious party.

69. How is JWT different and list the pros and cons of using JWT tokens?

- JWT is a format for securely transmitting data between parties, while hashing and

encryption are cryptographic techniques for transforming data.
JWT is used for
authentication and authorization in web applications, while hashing and encryption are
used for data validation and data confidentiality

70. What are the different ways to manage authentication?
There are several ways to manage authentication.

- MFA(Multi-factor Authentication) -
- It requires at least two methods of authentication. This is becoming more common
because passwords alone are no longer considered secure.
- With multi-factor authentication, a user is asked to enter a username and
password. If it matches, a code is sent to an authentication app, phone number,
email, or another resource then only the user is supposed to have access. The
user will then enter that code on the login page.

71. What is cookie-based auth?

- Cookie-based auth is a method of managing user authentication in web applications
where a cookie (pieces of data used to identify the user and their

preferences) is
used to store user authentication information.

- The browser returns the cookie to the server every time the page is requested. Specific
cookies like HTTP cookies are used to perform cookie-based authentication to maintain
the session for each user.

72. what is session management?

- Session management is the process of securely managing and maintaining the state of a
user's interaction with a web application or system.
- The purpose of session management is to keep track of a user's actions across multiple
pages or requests and to persist the data for the duration of the user's session

73. What is OAuth?

- OAuth (Open Authorization) is an open standard and authorization protocol that
allows third-party applications to access user data from a resource server (such as a
social media platform like Facebook, Gmail, etc., or cloud storage service) on behalf of
the user, without requiring the user to share their login credentials

with the third-party
application.

- The OAuth process involves the client (the third-party application that wants to
access the user's data) obtaining an access token from the resource server (the server
that hosts the user's data and is responsible for authenticating the user)

74. How does Express work?

- Express is just a framework, that can help us in creating the server in a very easy way.
- Four stages :
- creating an express application
- creating a new route
- starting an HTTP server on a given port number
- handling a request once it comes in

75. What are Middlewares?

- Middleware is the layer that exists between the application components, tools, and
devices.
- Middlewares are the functions that get executed before the request reaches the route
handler and after the response is sent to the client.

10pt;"> • It sets between request and response.

76. what is the MVC framework?

MVC is an acronym for Model-View-Controller. It is a design pattern for software projects.
Model: The model represents the structure of data, the format, and the constraints with which it
is stored. It maintains the data of the application. Essentially, it is the database part of the
application.
View: View is what is presented to the user. Views utilize the Mode and present data in a form
that the user wants. A user can also be allowed to make changes t the data presented to the
user. They consist of static and dynamic pages which are rendered or sent to the user when the
user requests them.
controller: The controller controls the requests of the user and then generates an appropriate
response which is fed to the viewer. Typically, the user interacts with the view, which in turn
generates the appropriate request, this request will be handled by controller. The controller
renders the appropriate view with the model data as a

response.

77. How do you do static routing?

- Static routing is a method of routing network traffic by manually configuring the network
devices to use a fixed path or route to reach a particular network or destination.
- With static routing, we must manually provide/configure the above functions on each
router of the network. Since all configurations are manual, a routing protocol is not
required and is not used in static routing.

78. What are common libraries you work with Express?

CORS - Allow or Restrict Requested Resources on a Web Server

- CORS is a node.js package that provides a Connect/Express middleware for enabling
CORS with a variety of options.
- CORS stands for Cross-Origin Resource Sharing. Without prior consent, it prevents other
websites or domains from accessing your web resources directly from the browser.
- Features of CORS
- Supports GET, POST, or HEAD HTTP methods.
- Allows web programmers to use regular XML HttpRequest, which

handles errors
better.
- Allows websites to parse responses to increase security.

79. How do you manage sessions in Express?

• In Express, sessions are typically managed using middleware that stores session data on
the server and sends a session ID to the client in a cookie.
• The client then sends the session ID with subsequent requests, allowing the server to
retrieve the session data.
• The middleware will create a session object for each client and store it on the server. It
will also send a session ID cookie to the client, which will be used to identify the session
on subsequent requests.

80. What is CORS?

CORS (Cross-Origin Resource Sharing) -
• CORS stands for Cross-Origin Resource Sharing. Without prior consent, it prevents other
websites or domains from accessing your web resources directly from the browser.

81. How do you manage cookies with Express?

- Cookies are small pieces of data that are stored on the client side (usually in the
browser) and are sent back to the server with each subsequent request. They are
commonly used to store user preferences, session data, and other information that needs
to persist across multiple requests.

82. What are Models?

- A constructor is a blueprint using which we can create objects. Similarly, using the Model
we create documents (collection of a particular database of MongoDB).
- So, the model is a constructor function.

Syntax ⇒

mongoose.model(Nameofthecollection, Schemaofthecollection)

83. What are aggregation pipelines?

- It is a way provided by MongoDB to perform aggregation.
- The aggregation pipeline has various stages and each stage transforms the document.

84. Explain why a mongoose does not return a promise but has .then?

- Mongoose provides a way to define and interact with MongoDB collections in a more structured and consistent manner than using MongoDB directly.
- It does not return Promises directly, but it does provide support for Promises through its underlying Promise library, and its fluent API and chaining syntax aim to provide a more consistent way to work with MongoDB collections.

85. What is REST API?

- A REST (Representational State Transfer) API (Application Programming Interface) is a type of web service that enables different software systems to communicate with each other over the internet. RESTful APIs rely on the HTTP protocol, which is built on top of TCP/IP (Transmission Control Protocol/Internet Protocol).
- REST suggests creating an object of the data requested by the client and sending the values of the object in response to the user.
- RESTful APIs use TCP (low-level networking protocol that provides a reliable, ordered, and error-checked delivery of packets between two applications running

on different hosts) to establish a connection between a client and a server and to
transfer data between them.

86. What is gRPC?

- gRPC is a high-performance RPC framework/technology built by Google. It uses
Google's own "Protocol Buffers", which is an open-source message format for data
serialization, as the default method of communication between the client and the server.
- It allows clients to call remote methods on a server as if they were local methods, which
makes it easier to build distributed systems and microservices.

88. What is HTTP?

- HTTP stands for Hypertext Transfer Protocol. It's a set of rules which govern the
exchange of data over the Internet.
- It is an application-layer protocol that is used to transmit data between web servers and
web browsers.

89. What is a web socket?