# Interview question and their answer  for the MERN + nodeJs

- **NodeJS** is one of the most popular runtime environments, known for its efficiency, scalability, and ability to handle asynchronous operations.
- It is built on **Chrome's V8 JavaScript engine** for executing JavaScript code outside of a browser.

**1. What is NodeJS?**

NodeJS is an open-source, cross-platform JavaScript runtime environment engine used for executing JavaScript code outside the browser. It is built on Google Chrome's V8 JavaScript engine. Some of the key features of the NodeJS are mentioned below:

- Single-threaded
- Non-Blocking, Asynchronous I/O
- Cross-platform
- Fast Execution (V8 Engine)
- Real-Time Data Handling

2. What is NPM?

NPM stands for the Node Package Manager. It is the package manager for the NodeJS environment. It is used to install, share, and manage dependencies (libraries, tools, or packages) in JavaScript applications. Below are the following key points about the NPM:

- NPM uses a package.json file in NodeJS projects to track project dependencies, versions, scripts, and metadata like the project's name and version.
- NPM is accessed by a command-line interface (CLI). Common commands include npm install to install packages, npm update to update them, and npm uninstall to remove them.

**3. Why** NodeJS is single-threaded?

NodeJS is single-threaded because it's based on the asynchronous, non-blocking nature of JavaScript. This design makes it simpler to develop and maintain, and it allows NodeJS to handle many concurrent requests efficiently.

**4. What kind of API function is supported by NodeJS?**

There are two types of API functions supported by NodeJS:

Synchronous: These API functions are used for blocking code.

Asynchronous: These API functions are used for non-blocking code.

**6. What is a module in NodeJS?**

In NodeJS Application, a Module can be considered as a block of code that provide a simple or complex functionality that can communicate with external application. Modules can be organized in a single file or a collection of multiple files/folders. They are useful because of their reusability and ability to reduce the complexity of code into smaller pieces. Examples of modules are. http, fs, os, path, etc.

**5. What is the difference between Synchronous and Asynchronous functions?**

Here we have difference table between Synchronous and Asynchronous functions

| **Feature** | **Synchronous Functions** | **Asynchronous Functions** |
|---|---|---|
| Execution Blocking | Blocks the execution until the task completes. | Does not block the execution; allows other tasks to proceed concurrently. |
| Waiting for Completion | Executes tasks sequentially; each task must complete before the next one starts. | Initiates tasks and proceeds with other operations while waiting for completion. |
| Return Value | Returns the result immediately after completion. | Typically returns a promise, callback, or uses event handling to handle the result upon completion. |
| Error Handling | Errors can be easily caught with try-catch blocks. | Error handling is more complex and often involves callbacks, promises, or async/await syntax. |
| Usage Scenario | Suitable for simple, sequential tasks with predictable execution flow. | Ideal for I/O-bound operations, network requests, and tasks requiring parallel processing. |

**7. What is the purpose of the 'require' keyword in NodeJS?**

The require keyword in NodeJS is used to include and import modules (external or built-in) into a NodeJS application.

**Example**

```
const http = require('http')   //imports the HTTP module to create a server.
```

**7. What is the purpose of the 'require' keyword in NodeJS?**

The require keyword in NodeJS is used to include and import modules (external or built-in) into a NodeJS application.

Example

const http = require('http')   //imports the HTTP module to create a server.

### 10. What is control flow in NodeJS?

Control flow in NodeJS refers to the sequence in which statements and functions are executed. It manages the order of execution, handling asynchronous operations, callbacks, and error handling to ensure smooth program flow.

### 11. What do you mean by event loop in NodeJS?

The event loop in NodeJS is a mechanism that allows it to handle multiple asynchronous tasks concurrently within a single thread. It continuously listens for events and executes associated callback functions.

### 12. What is the order in which control flow statements get executed?

The order in which the statements are executed is as follows:

- Execution and queue handling
- Collection of data and storing it
- Handling concurrency
- Executing the next lines of code

### 14. What is REPL in NodeJS?

REPL in NodeJS stands for Read, Evaluate, Print, and Loop. It is a computer environment similar to the shell which is useful for writing and debugging code as it executes the code in on go.

- **Read:** It reads the input provided by the user (JavaScript expressions or commands).
- **Eval:** It evaluates the input (executes the code).
- **Print**: It prints the result of the evaluation to the console.
- **Loop**: It loops back, allowing you to enter more code and get immediate results.

## 15. How to import a module in NodeJS?

We use the require module to import the External libraries in NodeJS. The result returned by require() is stored in a variable which is used to invoke the functions using the dot notation.

### 16. What is the difference between NodeJS and Angular?

| **Feature** | **NodeJS** | **Angular** |
|---|---|---|
| **Type** | Server-side runtime environment | Front-end framework |
| **Language** | JavaScript (or TypeScript) | TypeScript (primarily, but supports JavaScript) |
| **Execution** | Runs on the server to handle requests and responses. | Runs on the client side (browser) to build user interfaces. |
| **Performance** | Highly efficient for I/O operations due to its non-blocking nature. | Performance-focused with optimization for large-scale single-page applications. |
| **Usage** | Primarily used for server-side JavaScript execution and backend services. | Used for building interactive, dynamic, and responsive front-end applications. |

## 17. What is package.json in NodeJS?

package.json in NodeJS is a metadata file that contains project-specific information such as dependencies, scripts, version, author details, and other configuration settings required for managing and building the project.

**Example:**

```
{
    "name": "app",
    "version": "1.0.0",
    "main": "index.js",
    "scripts": {
        "test": "echo \"Error: no test specified\" && exit 1"
    },
    "keywords": [],
    "author": "",
    "license": "ISC",
    "description": "",
    "dependencies": {
        "express": "^4.21.2"
    }
}
```

### 18. How to create the simple HTTP server in NodeJS?

You can create a simple HTTP server in NodeJS using the built-in http module:

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



const server = http.createServer((req, res) => {

  res.writeHead(200, { 'Content-Type': 'text/plain' });

  res.end('Hello, World!');

});

server.listen(3000, () => {

  console.log('Server is running at http://localhost:3000/');

});
```

### 19. What is the most commonly used libraries in NodeJS?

There are the two most commonly used libraries in NodeJs:

- **ExpressJS:** ExpressJS is a minimal and flexible web application framework for building robust APIs and web apps. It simplifies routing, middleware handling, and request/response management.
- **Mongoose:** An Object Data Modeling (ODM) library for MongoDB and NodeJS, it helps in managing data relationships, schema validation, and business logic.

## 20. What are promises in NodeJS?

A promise is basically an advancement of callbacks in NodeJS. In other words, a promise is a JavaScript object which is used to handle all the asynchronous data operations. While developing an application you may encounter that you are using a lot of nested callback functions which causes a problem of callback hell. Promises solve this problem of callback hell.

## 21. What is event-driven programming in NodeJS?

Event-driven programming is used to synchronize the occurrence of multiple events and to make the program as simple as possible. The basic components of an Event-Driven Program are:

- A callback function ( called an event handler) is called when an event is triggered.
- An event loop that listens for event triggers and calls the corresponding event handler for that event.

## 22. What is a a buffer in NodeJS?

The Buffer class in NodeJS is used to perform operations on raw binary data. Generally, Buffer refers to the particular memory location in memory. Buffer and array have some similarities, but the difference is array can be any type, and it can be resizable. Buffers only deal with binary data, and it can not be resizable. Each integer in a buffer represents a byte. console.log() function is used to print the Buffer instance.

## 23. What are streams in NodeJS?

In NodeJS, streams are a powerful way to handle data in chunks rather than loading the entire data into memory. Streams allow for the efficient processing of large volumes of data, especially in situations where the data size is too large to fit into memory all at once.

**There are the four types of the Streams:**

- **Readable Streams:** These streams allow you to read data. For example, reading data from a file or receiving HTTP request data. **Example:**

fs.createReadStream() or http.IncomingMessage.

- **Writable Streams:** These streams allow you to write data. For example, writing data to a file or sending HTTP response data. **Example:**

fs.createWriteStream() or http.ServerResponse.

- **Duplex Streams:** These are both readable and writable. You can both read and write data using the same stream. **Example**: A TCP socket.
- **Transform Streams:** These are a type of duplex stream where the data is transformed as it is read and written. **Example:** A zlib stream to compress or decompress data.

## Explain crypto module in NodeJS

The crypto module is used for encrypting, decrypting, or hashing any type of data.

**25. What is callback hell?**

Callback hell is an issue caused due to a nested callback. This causes the code to look like a pyramid and makes it unable to read To overcome this situation we use promises.

**26. Explain the use of timers module in NodeJS**

The Timers module in NodeJS contains various functions that allow us to execute a block of code or a function after a set period of time. The Timers module is global, we do not need to use require() to import it.

It has the following methods:

- setTimeout() method
- setImmediate() method
- setInterval() method

**28. What are the different types of HTTP requests?**

The different types of the HTTP requests are mentioned below:

- **GET:** Retrieve data.
- **POST:** Create new resource.
- **PUT**: Update an entire resource.

- **PATCH**: Partially update a resource.
- **DELETE:** Remove a resource.

**32. What are the three methods to avoid callback hell?**

The three methods to avoid callback hell are:

- Using async/await()
- Using promises
- Using generators

## 34. What is CORS in NodeJS?

The word CORS stands for "Cross-Origin Resource Sharing". Cross-Origin Resource Sharing is an HTTP-header based mechanism implemented by the browser which allows a server or an API to indicate any origins (different in terms of protocol, hostname, or port) other than its origin from which the unknown origin gets permission to access and load resources. The cors package available in the npm registry is used to tackle CORS errors in a NodeJS application.

**40. How can we implement authentication and authorization in NodeJS?**

Authentication is the process of verifying a user's identity while authorization is determining what actions can be performed. We use packages like Passport and JWT to implement authentication and authorization.

**43. How to Connect NodeJS to a MongoDB Database?**

To connect to the MongoDB database write the following code after installing the Mongoose package:

```
const mongoose = require("mongoose");



mongoose.connect("DATABASE_URL_HERE", {

  useNewUrlParser: true,

  useUnifiedTopology: true
```

});

**46. What is web socket?**

Web Socket is a protocol that provides full-duplex (multiway) communication i.e. allows communication in both directions simultaneously. Web Socket is a modern web technology in which there is a continuous connection between the user's browser (client) and the server. In this type of communication, between the web server and the web browser, both of them can send messages to each other at any point in time.

---

## React Js

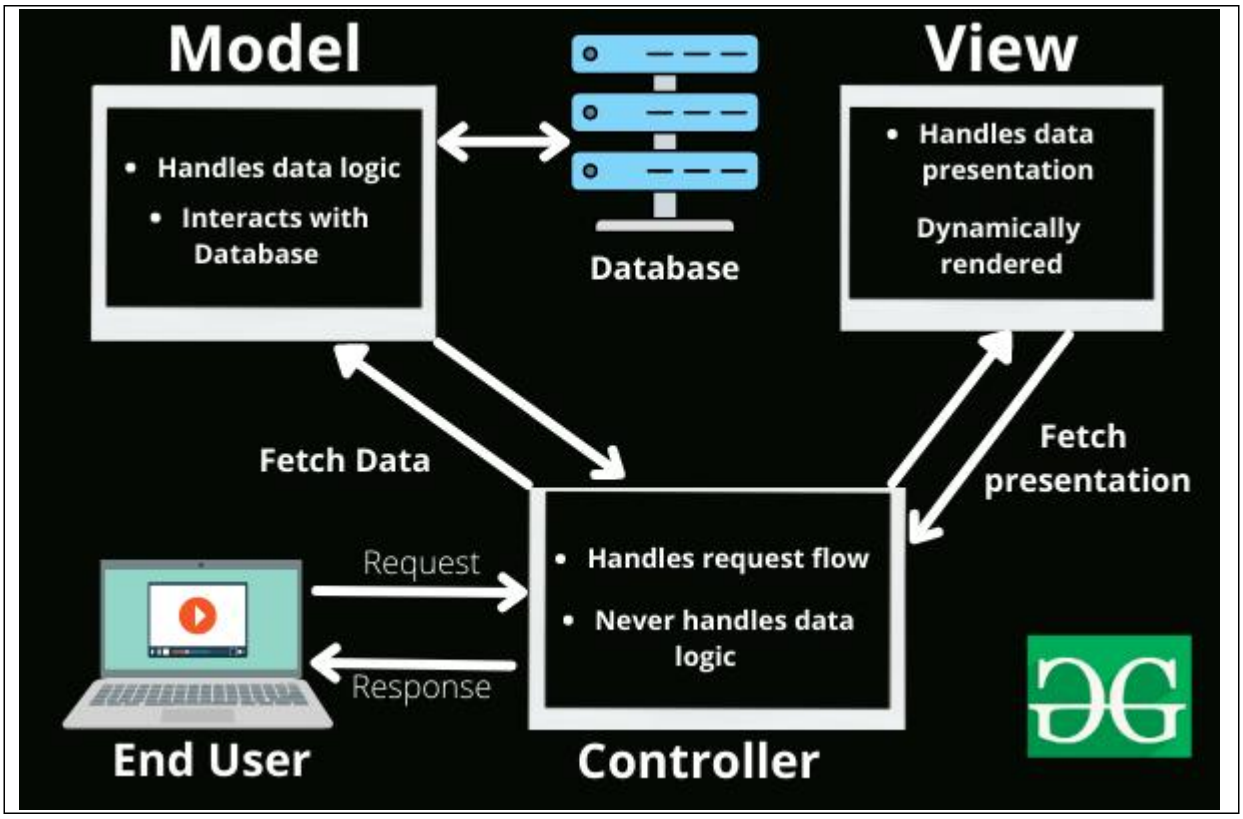
Question:- What is the react Js?

React is an efficient, flexible, and open-source JavaScript library that allows developers to create simple, fast, and scalable web applications.

- **Virtual DOM:** React uses a virtual DOM to efficiently update and render components, ensuring fast performance by minimizing direct DOM manipulations.
- **Component-Based Architecture:** React builds UI using reusable, isolated components, making code more modular, maintainable, and scalable.
- **Hooks**: React Hooks allow functional components to manage state and side effects, making them powerful and more flexible.
- **Server-Side Rendering (SSR)**: React can be used for server-side rendering, where HTML content is generated on the server and sent to the client. This improves the app's performance, especially for SEO.
- **React Router:** React Router enables navigation in a React application. It allows you to define different routes for different views in a single-page application (SPA).

### 2. Explain the MVC architecture.

The Model-View-Controller (MVC) framework is an architectural/design pattern that separates an application into three main logical components: Model, View, and Controller. Each architectural component is built to handle specific development aspects of an application. It isolates the business, logic, and presentation layers from each other



**3. Explain the building blocks of React.**

The five main building blocks of React are

- **Components:** These are reusable blocks of code that return HTML.
- **JSX:** It stands for JavaScript and XML and allows you to write HTML in React.
- **Props and State:** props are like function parameters and State is similar to variables.
- **Context:** This allows data to be passed through components as props in a hierarchy.
- **Virtual DOM:** It is a lightweight copy of the actual DOM which makes DOM manipulation easier.

## . Explain props and state in React with differences

Props are used to pass data from one component to another. The state is local data storage that is local to the component only and cannot be passed to other components.
Here is the difference table of props and state In react

| PROPS | STATE |
|---|---|
| The Data is passed from one component to another. | The Data is passed within the component only. |
| It is Immutable (cannot be modified). | It is Mutable ( can be modified). |
| Props can be used with state and functional components. | The state can be used only with the state components/class component (Before 16.0). |
| Props are read-only. | The state is both read and write. |

**What is virtual DOM in React?**

The **Virtual DOM** in React is an in-memory representation of the actual DOM. It helps React efficiently update and render the user interface by comparing the current and previous virtual DOM states using a process called **diffing**.

**How Virtual DOM Works**

- **Efficient Rendering**: The Virtual DOM is an in-memory representation of the actual DOM that React uses to optimize the process of updating and rendering UI changes.
- **Diffing Algorithm**: React compares the current and previous versions of the Virtual DOM using a diffing algorithm, identifying the minimal set of changes required to update the real DOM.
- **Batch Updates**: Instead of updating the real DOM immediately, React batches multiple changes to reduce unnecessary re-renders, improving performance.
- **Faster Updates**: Since updating the real DOM is slow, React minimizes direct DOM manipulations by only making updates where necessary after comparing the Virtual DOM.
- **Declarative UI**: With the Virtual DOM, React allows developers to write code in a declarative style, letting React handle when and how to efficiently update the UI.

## What is JSX?

JSX is basically a syntax extension of regular JavaScript and is used to create React elements. These elements are then rendered to the React DOM. All the React components are written in JSX. To embed any JavaScript expression in a piece of code written in JSX we will have to wrap that expression in curly braces {}.

**Example of JSX:** The name written in curly braces { } signifies JSX

```
const name = "Learner";
const element = (
    <h1>
        Hello,
        {name}.Welcome to GeeksforGeeks.
    </h1>
);
```

### What are components and their type in React?

A Component is one of the core building blocks of React. In other words, we can say that every application you will develop in React will be made up of pieces called components. Components make the task of building UIs much easier.

**In React, we mainly have two types of components:**

- **Functional Components:** Functional components are simply JavaScript functions. We can create a functional component in React by writing a JavaScript function.

- **Class Components:** The class components are a little more complex than the functional components. The functional components are not aware of the other components in your program whereas the class components can work with each other.

### 8. How do browsers read JSX?

- In general, browsers are not capable of reading JSX and only can read pure JavaScript. The web browsers read JSX with the help of a transpiler. Transpilers are used to convert JSX into JavaScript. The transpiler used is called Babel.
- We can pass data from one class component to another class component.

## 14. Explain the difference between React and Angular?

| *Features* | *React* | *Angular* |
|---|---|---|
| **Used as** | React is a JavaScript library. As it indicates react js updates only the virtual DOM is present and the | Angular is a framework. Angular updates the Real DOM and the data flow is |

| *Features* | *React* | *Angular* |
|---|---|---|
| | data flow is always in a single direction. | ensured in the architecture in both directions. |
| **Architecture** | React is more simplified as it follows MVC ie., Model View Control. | The architecture is complex as it follows MVVM models ie., Model View-ViewModel. |
| **Scalability** | It is highly scalable. | It is less scalable than React JS. |
| **Data Binding** | It supports Uni-directional data binding which is one-way data binding. | It supports Bi-directional data binding which is two way data binding. |
| **DOM** | It has a virtual DOM. | It has regular DOM. |

**16. What is state in React?**

The state is an instance of React Component Class that can be defined as an object of a set of observable properties that control the behaviour of the component. In other words, the State of a component is an object that holds some information that may change over the lifetime of the component.

**17. Explain props in React?**

React allows us to pass information to a Component using something called props (which stands for properties). Props are objects which can be used inside a component

We can access any props inside from the component's class to which the props is passed. The props can be accessed as shown below:

this.props.propName;

**19. Explain the difference between functional and class component in React?**

| Features | Functional Components | Class Components |
|---|---|---|
| Definition | A functional component is just a plain JavaScript pure function that accepts props as an argument | A class component requires you to extend from React. Component and create a render function |
| Rendering | No render method used | It must have the render() method returning JSX |
| Stateless or Stateful | Also known as Stateless components | Also known as Stateful components |
| Lifecycle Methods | React lifecycle methods (for example, componentDidMount) cannot be used in functional components. | React lifecycle methods can be used inside class components (for example, componentDidMount). |
| Constructor | Constructors are not used. | Constructor is used as it needs to store state. |
| State Management | Uses hooks like useState for managing state. | Uses this.state and this.setState for state management. |

**20. Explain one way data binding in React?**

ReactJS uses one-way data binding which can be Component to View or View to Component. It is also known as one-way data flow, which means the data has one, and only one way to be transferred to other parts of the application. In essence, this means child components are not able to update the data that is coming from the parent component. It is easy to debug and less prone to errors.

**20. Explain one way data binding in React?**

ReactJS uses one-way data binding which can be Component to View or View to Component. It is also known as one-way data flow, which means the data has one, and only one way to be transferred to other parts of the application. In essence, this means child components are not able to update the data that is coming from the parent component. It is easy to debug and less prone to errors.

**24. What are Pure Components in React?**

A Pure Component is a type of React component that only re-renders if the props or state it receives change. React provides React.PureComponent, which is a base class that automatically performs a shallow comparison of props and state to determine if a re-render is necessary.

**25. What is the significance of setState() in React?**

setState() is a method used to update the state of a component. When the state is updated, React re-renders the component and its child components to reflect the changes.

**27. What is react router?**

React Router is a standard library for routing in React. It enables the navigation among views of various components in a React Application, allows changing the browser URL, and keeps the UI in sync with the URL.

To install react router type the following command.

npm i react-router-dom

**28. Explain the components of a react-router.**

The main components of a react-router are:

1. **Router(usually imported as BrowserRouter):** It is the parent component that is used to store all of the other components. Everything within this will be part of the routing functionality

2. **Switch:** The switch component is used to render only the first route that matches the location rather than rendering all matching routes.

3. **Route:** This component checks the current URL and displays the component associated with that exact path. All routes are placed within the switch components.

4. **Link:** The Link component is used to create links to different routes.

**29. Explain the lifecycle methods of components**

A React Component can go through four stages of its life as follows.

- **Initialization:** This is the stage where the component is constructed with the given Props and default state. This is done in the constructor of a Component Class.

- **Mounting:** Mounting is the stage of rendering the JSX returned by the render method itself.

- **Updating:** Updating is the stage when the state of a component is updated and the application is repainted.

- **Unmounting:** As the name suggests Unmounting is the final step of the component lifecycle where the component is removed from the page.

## 31. What is this.setState function in React?

We use the setState() method to change the state object. It ensures that the component has been updated and calls for re-rendering of the component. The state object of a component may contain multiple attributes and React allows using setState() function to update only a subset of those attributes as well as using multiple setState() methods to update each attribute value independently.

### 32. What is the use of ref in React?

Refs are a function provided by React to access the DOM element and the React element that you might have created on your own. They are used in cases where we want to change the value of a child component, without making use of props and all. They have wide functionality as we can use callbacks with them.

The syntax to use ref is :

const node = this.myCallRef.current;

### 33. What are hooks in React?

Hooks are a new addition in React 16.8. They let developers use state and other React features without writing a class. Hooks doesn't violate any existing React concepts. Instead, Hooks provide a direct API to react concepts such as props, state, context, refs and life-cycle

### 34. Explain the useState hook in React?

The most used hook in React is the useState() hook. Using this hook we can declare a state variable inside a function but only one state variable can be declared using a single useState() hook. Whenever the useState() hook is used, the value of the state variable is changed and the new variable is stored in a new cell in the stack.

**Syntax:**

const [state, setState] = useState(initialState);

- **state**: The current state value.
- **setState:** A function used to update the state value.
- **initialState**: The initial value of the state.

### 35. Explain the useEffect hook in React?

The useEffect hook in React eliminates the side effect of using class based components. It is used as an alternative to componentDidUpdate() method. The useEffect hook accepts two arguments where second argument is optional.

useEffect(function, dependency)

The dependency decides when the component will be updated again after rendering.

**36. What is React Fragments?**

when we are trying to render more than one root element we have to put the entire content inside the 'div' tag which is not loved by many developers. So since React 16.2 version, Fragments were introduced, and we use them instead of the extraneous 'div' tag. The following syntax is used to create fragment in react.

```
<React.Fragment>
   <h2>Child-1</h2>
   <p>                                                                    Child-2</p>
</React.Fragment>
```

**37. What is a react developer tool?**

React Developer Tools is a Chrome DevTools extension for the React JavaScript library. A very useful tool, if you are working on React applications. This extension adds React debugging tools to the Chrome Developer Tools. It helps you to inspect and edit the React component tree that builds the page, and for each component, one can check the props, the state, hooks, etc.

**50. Explain CORS in React?**

In ReactJS, Cross-Origin Resource Sharing (CORS) refers to the method that allows you to make requests to the server deployed at a different domain. As a reference, if the frontend and backend are at two different domains, we need CORS there.

We can setup CORS evironment in frontend using two methods:

- axios
- fetch

**51. What is axios and how to use it in React?**

Axios, which is a popular library is mainly used to send asynchronous HTTP requests to REST endpoints. This library is very useful to perform CRUD operations.

- This popular library is used to communicate with the backend. Axios supports the Promise API, native to JS ES6.
- Using Axios we make API requests in our application. Once the request is made we get the data in Return, and then we use this data in our project.

To install aixos package in react use the following command.

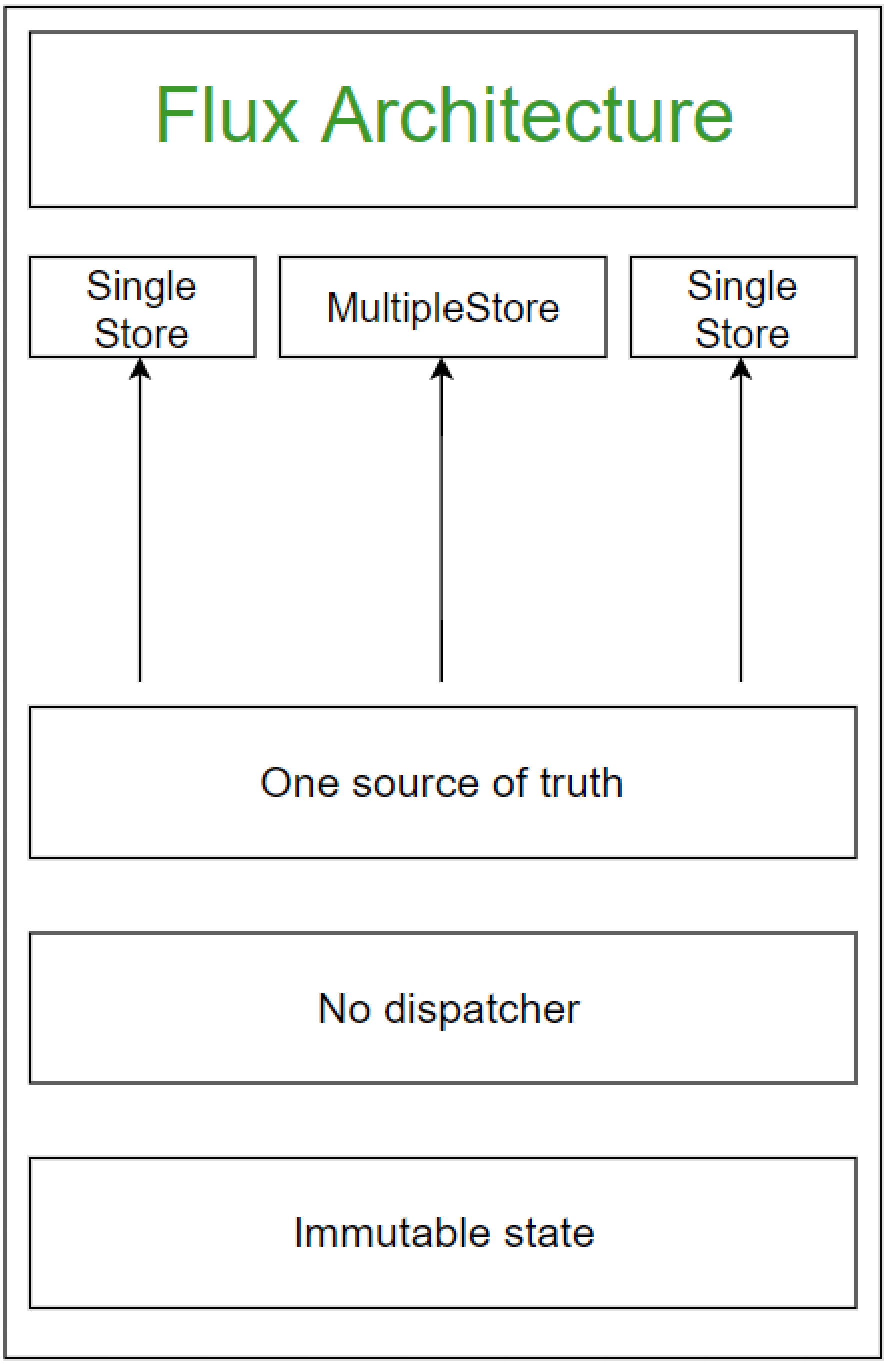
```
npm i axios
```

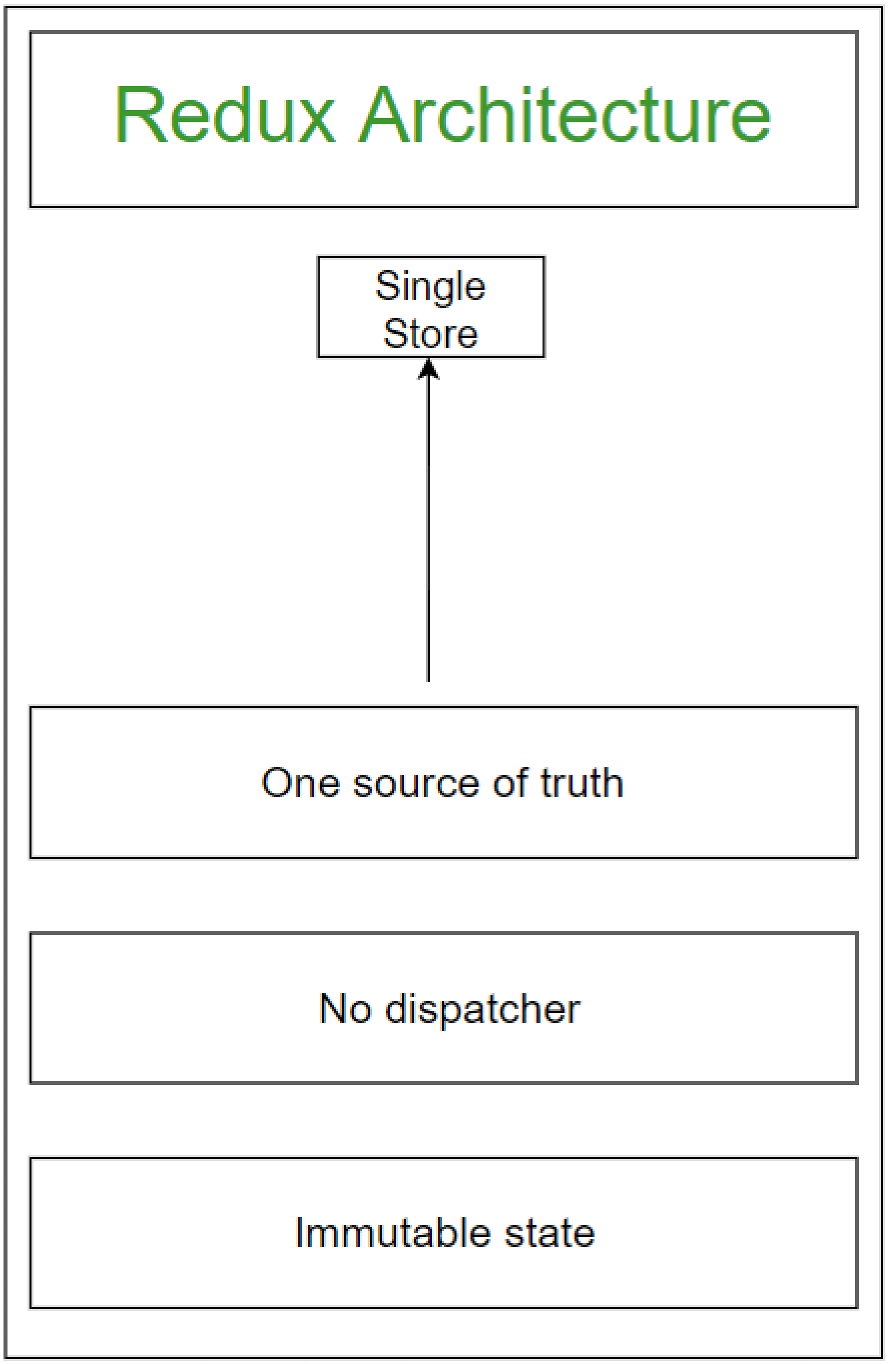
## 54. What is React-Material UI?

React Material UI is an open-source React component library, offering prebuilt components for creating React applications. Developed by Google in 2014, it’s compatible with JavaScript frameworks like Angular.js and Vue.js. Renowned for its quality designs and easy customization, it’s favored by developers for rapid development.

## 55. What is flux architecture in redux?

Flux architecture in Redux is a design pattern used for managing application state in a unidirectional data flow. In this architecture, actions are dispatched to modify the store, which holds the entire application state. The store sends the updated state to the view (UI), and the cycle repeats when new actions are triggered. Redux follows this structure to ensure a predictable and maintainable state management system for large applications.


Flux Architecture
Single Store
MultipleStore
Single Store
One source of truth
No dispatcher
Immutable state



---

NextJs

**1- What is Next.js, and how is it different from React?**

Interview question and their answer  for the MERN + nodeJs

Next.js is a React-based open-source framework that helps developers build server-side rendered React applications.

The key difference between React and Next.js is the way they handle routing. React uses client-side routing, meaning the page transitions are handled entirely on the client-side using JavaScript.

In contrast, Next.js provides server-side routing, which means that the server handles the routing and sends the pre-rendered pages to the client, resulting in faster page loads and better SEO.

Next.js also provides additional features like automatic code splitting, static site generation, and dynamic imports.

**2- What are the advantages of using Next.js over React?**

Next.js offers several advantages over React, including server-side rendering, automatic code splitting, static site generation, dynamic imports, optimized performance, and easy deployment. Additionally, Next.js supports built-in SEO and analytics, making it easier to optimize your application for search engines and track user engagement.

**4- What is server-side rendering, and why is it important?**

Server-side rendering (SSR) is the process of rendering a web page on the server before sending it to the client's browser. SSR is important because it allows search engines to crawl and index your website's content, which can improve your website's SEO. Additionally, SSR can improve the initial page load time and improve the user experience for users with slow internet connections or devices.

**What is client-side rendering, and how does it differ from server-side rendering?**

Client-side rendering (CSR) is the process of rendering a web page on the client's browser using JavaScript after receiving the initial HTML, CSS, and JavaScript from the server. The key difference between SSR and CSR is that SSR sends a fully rendered HTML page to the client's browser, while CSR sends an empty HTML page that is populated by JavaScript.

**6- What is static site generation, and how does it differ from server-side rendering?**

Static site generation (SSG) is the process of generating a static HTML, CSS, and JavaScript file for each page on your website at build time. The key difference between SSG and SSR is that SSG generates a static file that can be served from a content delivery network (CDN), while SSR generates the HTML dynamically on the server and sends it to the client's browser.

**7- How do you configure routing in a Next.js application?**

Next.js uses file-based routing, which means that you can create a page by creating a new file in the pages directory with the corresponding URL path. For example, to create a page with the URL path /about, you would create a file called about.js in the pages directory.

**8- What is the purpose of the getStaticProps function in Next.js?**

The getStaticProps function is used to fetch data at build time for static site generation. This function is called during the build process and can be used to fetch data from an external API or database. The data returned by getStaticProps is then passed as props to the page component.

**9- How do you pass data between pages in a Next.js application?**

Next.js provides several ways to pass data between pages in a Next.js application, including URL query parameters, the Router API, and state management libraries like Redux or React Context. You can also use the getServerSideProps function to fetch data on the server and pass it as props to the page component.

**6. What do you mean by SSR?**

SSR stands for **Server-Side Rendering**. It's a technique used in web development where the server processes the React or other JavaScript framework code and generates the HTML on the server side, sending the fully rendered HTML to the client's browser.

Here's a brief overview of the SSR process:

1. **Request from Client:** When a user makes a request to a server for a web page, the server receives the request.
2. **Server-Side Processing:** Instead of sending just a blank HTML shell or a minimal document, the server executes the JavaScript code associated with the requested page, fetches data if needed, and renders the complete HTML content on the server side.
3. **Sending Rendered HTML to Client:** The fully rendered HTML, along with any necessary CSS and JavaScript, is sent as a response to the client's browser.
4. **Client-Side Hydration:** Once the HTML is received by the browser, any JavaScript code needed for interactive elements or further client-side rendering is executed. This process is known as "**hydration**."

**5. Mention some features of Next.js.**

Next.js is a powerful React framework that offers various features to simplify and enhance the development of web applications. Here are some key features of Next.js:

- **Server-Side Rendering (SSR):** Next.js allows server-side rendering, improving initial page load performance by rendering HTML on the server and sending it to the client.

- **Static Site Generation (SSG):** Next.js supports static site generation, enabling the pre-rendering of pages at build time, resulting in faster loading times and better SEO.
- **File System-Based Routing:** The routing system is based on the file structure of the "pages" directory, making it intuitive and easy to organize code.
- **Automatic Code Splitting:** Next.js automatically splits code into smaller chunks, loading only what's necessary for each page. This enhances performance by reducing initial bundle sizes.
- **API Routes:** Easily create serverless functions by defining API routes alongside your pages, simplifying the development of server-side logic.

**7. What are the benefits of using Next.js?**

Next.js is a popular React framework that brings several benefits to web development. Here are some of the key advantages of using Next.js:

- **Server-Side Rendering (SSR):** Next.js supports server-side rendering out of the box. This means that pages can be rendered on the server and then sent to the client, providing better performance and SEO as search engines can crawl the fully rendered content.
- **Static Site Generation (SSG):** Next.js allows for static site generation, where pages can be pre-built at build time. This can significantly improve performance by serving static files directly from a CDN, reducing the load on servers and improving the user experience.
- **Automatic Code Splitting:** Next.js automatically splits the code into smaller chunks, allowing for efficient loading of only the necessary code for a particular page. This results in faster initial page loads and improved overall performance.
- **Built-in CSS Support:** Next.js provides built-in support for styling solutions, including CSS modules, styled-jsx, and support for CSS-in-JS libraries. This allows developers to choose their preferred styling approach without the need for additional configuration.
- **API Routes:** Next.js allows you to create API routes easily, enabling the development of serverless functions. This can be useful for handling backend logic without the need for a separate server.

## Difference between the pre-rendering types available in Next.js.

| | **Static Generation (SG)** | **Server-Side Rendering (SSR)** |
|---|---|---|
| **Generation Timing** | HTML is generated at build time. | HTML is generated on each request. |
| **Reuse of HTML** | The pre-generated HTML can be reused on every request. | HTML is generated anew for each request. |
| **Recommendation** | Recommended for performance and efficiency. | Suitable for cases where content changes frequently or cannot be determined at build time. |
| **Export Methods:** | Export the page component or use '**getStaticProps**' | Export '**getServerSideProps**' |
| **Build Time Dependency:** | Less dependent on server resources during runtime. | Depends on server resources for generating content dynamically. |
| **Performance** | Typically faster as HTML is pre-generated. | Can introduce higher server load due to on-the-fly HTML generation. |
| **Caching** | Easily cache static HTML. | Requires server-side caching mechanisms. |
| **Scalability** | Scales well as static content can be served efficiently. | May require additional server resources to handle dynamic content generation. |

Interview question and their answer for the MERN + nodeJs

# ExpressJs

## 1. What is ExpressJS?

Express is a small framework that sits on top of NodeJS’s web server functionality to simplify its APIs and add helpful new features. It makes it easier to organize your application’s functionality with middleware and routing. It adds helpful utilities to NodeJS’s HTTP objects and provides the rendering of dynamic HTTP objects.

Express is a part of MEAN stack, a full-stack JavaScript solution for building fast, robust, and maintainable production web applications.

## 2. Why use ExpressJS?

ExpressJS is a lightweight NodeJS framework that allows us to create server-side web applications faster and smarter. The main reason for choosing Express is its simplicity, minimalism, flexibility, and scalability characteristics. It provides an easy setup for middleware and routing.

## 3. Write a 'Hello World' ExpressJS application.

To create a simple ExpressJS application, first, we need to install Express in our NodeJS application.

**Initialize a NodeJS project**

```
npm init -y
```

**Install ExpressJS**

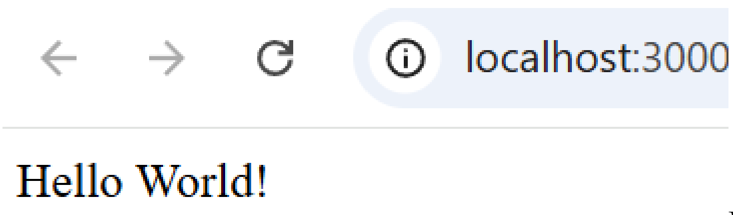
```
npm install express
```

**Create the index.js file and write the below code**

```
// Import the express module

const express = require('express');


const app = express();


app.get('/', (req, res) => {

  res.send('Hello World!');

});


const PORT = 3000;
```

```
// Start the server
app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
```

**Output**



Hello World!

Hello world in Express.JS

**4. Differentiate between NodeJS and ExpressJS?**

| **Feature** | **NodeJS** | **ExpressJS** |
|---|---|---|
| **Definition** | A runtime environment that allows JavaScript to be executed outside the browser. | A web framework built on top of NodeJS to simplify server-side development. |
| **Type** | A server-side JavaScript runtime. | A backend web framework based on NodeJS. |
| **API Development** | Can be used to create APIs but requires extra effort. | Provides an easy and structured way to develop RESTful APIs. |
| **Performance** | Slightly faster as it's minimal and doesn't include extra abstractions. | Adds a slight overhead due to additional features but is still highly efficient. |
| **Use Case** | Ideal for low-level operations, real-time applications, microservices, and command-line tools. | Ideal for web applications, APIs, RESTful services, and middleware-based projects. |

**5. Is ExpressJS a front-end or a back-end framework?**

ExpressJS is a JavaScript backend framework. It is mainly designed to develop complete web applications and APIs. Express is the backend component of the MERN stack which stands for MongoDB, ExpressJS, React.js, NodeJS.

**6. Mentions few features of ExpressJS.**

Few features of the ExpressJS includes

- **Routing:** Express provides a simple way to define routes for handling HTTP requests. Routes are used to map different URLs to specific pieces of code, making it easy to organize your application's logic.
- **Middleware:** Express uses middleware functions to perform tasks during the request-response cycle. Middleware functions have access to the request, response, and the next middleware function.
- **HTTP Utility Methods:** Express mainly used for handling HTTP methods like GET, POST, PUT, and DELETE. This makes it easy to define how the application should respond to different types of HTTP requests.
- **Static File Serving:** It can also serve static files, such as images, CSS, and JavaScript, with the help of built-in express.static middleware.
- **Security:** It includes features and middleware to strengthen the security of your web applications, such as the helmet middleware to secure your app.

**7. Explain the structure of an ExpressJS application?**

The structure of an ExpressJS application can vary depending on its complexity and the specific needs of the project. However, here is a basic approach that is commonly used:

- **Entry point:** This is the starting point of the application where you set up your server, connect to your database, add middleware, and define the main routes.
- **Routes directory:** This directory contains files for the app's routes.
- **Controllers directory:** This directory contains files that define the logic to handle requests for a specific route.
- **Models directory:** This directory is used for creating the schema models for the different data.
- **Middleware directory:** This directory contains custom middleware functions that you can use in your routes.
- **Views directory:** If you're using a templating engine, this directory contains your view templates.
- **Public directory:** This directory contains static files that are served directly by the server such as images, CSS files, and JavaScript files.

**8. What are some popular alternatives to ExpressJS?**

There are several popular alternatives to ExpressJS which includes:

- Koa.js
- Hapi.js
- Sails.js
- Fastify

**9. Which major tools can be integrated with ExpressJS?**

There are many tools and libraries that can be integrated with ExpressJS such as:

- **Database tools:** MongoDB, MySQL, PostgreSQL.
- **Template Engines:** EJS, Pug, Mustache.
- **Authentication libraries:** Passport.js.
- **Logging libraries:** Morgan, Winston.
- **Validation libraries**: Joi, express-validator.
- **ORM libraries:** Sequelize, Mongoose.

**10. What is .env file used for?**

The .env file is used for storing sensitive information in a web application which we don't want to expose to others like password, database connection string etc. It is a simple text file where each line represents a key-value pair, and these pairs are used to configure various aspects of the application.

**11. What are JWT?**

JSON Web Tokens are mainly a token which is used for authentication and information exchange. When a user signs in to an application, the application then assigns JWT to that user. Subsequent requests by the user will include the assigned JWT. This token tells the server what routes, services, and resources the user is allowed to access. Json Web Token includes 3 part namely- Header, Payload and Signature.

**12. Create a simple middleware for validating user.**

```
// Simple user validation middleware

const validateUser = (req, res, next) => {

  const user = req.user;


  if (!user) {

    return res.status(401).json({ error: 'Unauthorized - User not found' });
  }

  next();
};

app.get('/profile', validateUser, (req, res) => {
  const user = req.user;
  res.json({ message: 'Profile page', username: user.username });
});
```

**13. What is Bcrypt used for?**

Bcrypt is a password hashing function which is used to securely hash and store user passwords. It is designed to be slow and computationally intensive, making it resistant to brute-force attacks and rainbow table attacks. Bcrypt is a key component in enhancing the security of user authentication systems.

**14. Why should you separate the Express app and server?**

In ExpressJS, it is recommended to separate the Express App and the server setup. This provides the modularity and flexibility and makes the codebase more easier to maintain and test. Here are some reasons why you should separate the Express app and server:

- **Modularity:** You can define routes, middleware, and other components in the Express app independently of the server configuration.
- **Ease of Testing:** Separation makes it easier to write unit tests for the Express app without starting an actual server. You can test routes, middleware, and other components in isolation.
- **Reusability:** You can reuse the same Express app in different server configurations.
- **Configuration Management:** Separating the app and server allows for cleaner configuration management.
- **Scalability:** It provides a foundation for a scalable code structure. As your application grows, it will easier to maintain the code.

**15. What do you understand about ESLint?**

EsLint is a JavaScript linting tool which is used for automatically detecting incorrect patterns found in ECMAScript/JavaScript code. It is used with the purpose of improving code quality, making code more consistent, and avoiding bugs. ESLint is written using NodeJS to provide a fast runtime environment and easy installation via npm.

**16. Define the concept of the test pyramid.**

The Test Pyramid is a concept in software testing that represents the distribution of different types of tests. It was introduced by Mike Cohn, and it suggests that a testing strategy should be shaped like a pyramid, with the majority of tests at the base and fewer tests as you move up. The Test Pyramid consists of three levels: Unit Tests, Integration Tests, and End-to-End (E2E) Tests.

**17. Differentiate between res.send() and res.json().**

| **Feature** | **res.send()** | **res.json()** |
|---|---|---|
| **Purpose** | Sends a response of any type (string, object, array, buffer, etc.). | Specifically sends a JSON response. |
| **Data Handling** | Converts objects/arrays into JSON automatically, but also supports sending other data formats. | Converts objects/arrays into JSON format explicitly. |
| **Response Type** | Can send text, HTML, JSON, or any other data type. | Only sends JSON-formatted responses. |
| **Use Case** | Used when sending various types of responses, including HTML pages, strings, or JSON data. | Used specifically for sending JSON responses in APIs. |
| **Example Usage** | res.send('Hello World!') | res.json({ message: 'Success' }) |

**ExpressJS Interview Questions and Answers - Intermediate Level**

**18. What is meant by Scaffolding in ExpressJS?**

Scaffolding in ExpressJS refers to the process of generating a basic project structure automatically. This can speed up the initial setup and help maintain consistency in the way projects are structured, especially in large teams.

**19. How would you install an Express application generator for scaffolding?**

Express application generator are used for quickly setting up a new Express application with some basic structure. You can install it using Node Package Manager (npm), which comes with NodeJS.

**To install it globally:**

npm install -g express-generator

**20. What is Yeoman and how to install Yeoman for scaffolding?**

Yeoman is a scaffolding tool for web applications that helps developers to create new projects by providing a generator-based workflow.

**To install Yeoman run the following command:**

npm install -g yo

**Yeoman works with generators, which are packages that define the structure and configuration of a project. You can install a generator like this:**

npm install -g generator-express

**Once installed, you can use Yeoman to create a new application:**

yo appname

**21. What is CORS in ExpressJS?**

CORS (Cross-Origin Resource Sharing) is a security feature implemented by web browsers to control how web pages in one domain can request and interact with resources hosted on another domain.

In the context of ExpressJS, CORS refers to a middleware that enables Cross-Origin Resource Sharing for your application. This allows the application to control which domains can access your resources by setting HTTP headers.

**22. What are Built-in Middlewares?**

ExpressJS, includes a set of built-in middlewares that provide common functionality. These built-in middlewares are included by default when you create an Express application and can be used to handle various tasks. Here are some of the built-in middlewares in Express:

- **ExpressJSon():** This middleware is used to parse incoming JSON requests. It automatically parses the request body if the Content-Type header is set to application/json.
- **express.Router():** The express.Router() function is often used to create modular route handlers. It allows you to group route handlers together and then use them as a middleware.
- **express.static():** This middleware is used to serve static files, such as images, CSS, and JavaScript files, from a specified directory.

**23. How would you configure properties in ExpressJS?**

In ExpressJS, you can configure properties using the app.set() method. This method allows you to set various properties and options which affects the behavior of the Express application.

app.set(name, value);

Here, name represents the name of the property you want to configure, and value is the value you want to assign to that property. Express provides a wide range of properties that you can configure based on your application's requirements.

**24. Which template engines do Express support?**

ExpressJS supports any template engine that follows the (path, locals, callback) signature.

**25. Elaborate on the various methods of debugging on both Linux and Windows systems?**

The debugging is the vital need at the time of software development to identifying issues in the application's logic, handling of HTTP requests, middleware execution, and other aspects specific to web development. Here are some methods commonly used for debugging an ExpressJS application on both Linux and Windows:

- **Console.log:** The simplest way to debug an ExpressJS application is by using console.log(). You can output messages to the console which can be viewed in the terminal.
- **Node Inspector:** This is a powerful tool that allows you to debug your applications using Chrome Developer Tools. It supports features like setting breakpoints, stepping over functions, and inspecting variables.
- **Visual Studio Code Debugger:** VS Code provides a built-in debugger that works on both Linux and Windows. It supports advanced features like conditional breakpoints, function breakpoints, and logpoints.
- **Utilizing debug module:** The debug module is a small NodeJS debugging utility that allows you to create debugging scopes.

**26. Name some databases that integrate with ExpressJS?**

ExpressJS can support a variety of the databases which includes:

- MySQL
- MongoDB
- PostgreSQL
- SQLite
- Oracle

**27. How would you render plain HTML using ExpressJS?**

In ExpressJS, you can render plain HTML using the res.send() method or res.sendFile() method.

**Sample code:**

JavaScriptJavaScript

```
//using res.send


const express = require('express');

const app = express();

const port = 8000;


app.get('/', (req, res) => {

   const htmlContent = '<html><body><h1>Hello, World!</h1></body></html>';

   res.send(htmlContent);

});



app.listen(port, () => {

   console.log(`Server is listening on port ${port}`);

});
```

**28. What is the use of 'Response.cookie()' function?**

The response.cookie() function in ExpressJS is used to set cookies in the HTTP response. Cookies are small pieces of data sent from a server and stored on the client's browser. They are commonly used to store information about the user or to maintain session data.

res.cookie(name, value, [options]);

**29. Under what circumstances does a Cross-Origin resource fail in ExpressJS?**

When a Cross-Origin Resource Sharing request is made, the browser enforces certain security checks, and the request may fail under various circumstances:

- **No CORS Headers:** The server doesn't include the necessary CORS headers in its response.
- **Mismatched Origin:** The requesting origin does not match the origin specified in the Access-Control-Allow-Origin header.

- **Restricted HTTP Methods:** The browser enforces restrictions on which HTTP methods are allowed in cross-origin requests.
- **No Credentials:** The browser makes restrictions on requests that include credentials (such as cookies or HTTP authentication).

**30. What is Pug template engine in ExpressJS?**

Pug is a popular template engine for ExpressJS and other NodeJS frameworks. You can use Pug to render dynamic HTML pages on the server side. It allows you to write templates using a syntax that relies on indentation and concise tags.

**31. What is meant by the sanitizing input process in ExpressJS?**

Sanitizing input in ExpressJS application is an important security practice to prevent various types of attacks, such as Cross-Site Scripting (XSS) and SQL injection. It involves cleaning and validating user input before using it in your application so that it does not contain malicious code or can be a security risk.

**32. How to generating a skeleton ExpressJS app using terminal command?**

To generate a skeleton for an ExpressJS application using the terminal, you can use the Express application generator which is a command-line tool provided by the ExpressJS framework. This generator will setup a basic directory structure which includes necessary files, and installs essential dependencies.

**Steps to generate:**

**Step 1:** Open your terminal and install the Express application generator globally using the following command:

```
npm install -g express-generator
```

**Step 2:** After that you can use the express command to generate your ExpressJS app.

```
express my-express-app
```

**Step 3:** Now go to the app directory and install the dependencies and start the app by running-

```
npm install
```

```
npm start
```

**33. What are middlewares in ExpressJS?**

Middleware functions are those functions that have the access to request and response object and the next middleware or function. They can add functionality to an application, such as logging, authentication, and error handling.

**ExpressJS Interview Questions and Answers - Advanced Level**

**34. What are the types of middlewares?**

There are mainly five types of Middleware in ExpressJS:

- Application-level middleware
- Router-level middleware
- Error-handling middleware
- Built-in middleware
- Third-party middleware

**35. List the built-in middleware functions provided by Express.**

ExpressJS comes with several built-in middleware functions. Few of them are:

- **ExpressJSON:** This is used for parsing incoming requests with JSON payloads.
- **express.static:** This is used to serve static files like images, CSS files, and JavaScript files.
- **express.urlencoded:** This is used for parsing incoming requests with URL-encoded payloads.
- **express.raw**: This is used for parse incoming requests with a raw body.
- **express.text**: This is used for parse incoming requests with a text body.

**36. Mention some third-party middleware provided by ExpressJS.**

ExpressJS allows you to use third-party middleware to extend and enhance the functionality of your web application.

**Here are some commonly used third-party middleware in ExpressJS**

- **body-parser:** This middleware is used to parse incoming request bodies, allowing you to access form data or JSON payloads on req.body.
- **cors:** This module provides middleware to enable Cross-Origin Resource Sharing (CORS) in your Express application.
- **morgan:** Morgan is a middleware module that provides request logging functionality.
- **helmet:** Helmet helps to secure Express apps by setting various HTTP headers.
- **express-session:** This middleware is used for managing user sessions in your Express application.
- **passport:** This middleware is used for implementing authentication and authorization in Express applications.

**37. When application-level Middleware is used?**

Application-level middlewares are bound to an instance of the Express application and are executed for every incoming request. These middlewares are defined using the app.use()

method, and they can perform tasks such as logging, authentication, setting global variables, and more.

**38. Explain Router-level Middleware.**

Router-level middlewares are specific to a particular router instance. This type of middleware is bound to an instance of express.Router(). Router-level middleware works similarly to application-level middleware, but it's only invoked for the routes that are handled by that router instance. This allows you to apply middleware to specific subsets of your routes, keeping your application organized and manageable.

**39. How to secure ExpressJS application?**

It is very important to secure your application to protect it against various security threats. We can follow few best practices in our ExpressJS app to enhance the security of our application.

- Keep Dependencies Updated: Regularly update your project dependencies, including ExpressJS and other npm packages.
- Use Helmet Middleware: The helmet middleware helps secure your application by setting various HTTP headers. It helps prevent common web vulnerabilities.
- Set Secure HTTP Headers: Configure your application to include secure HTTP headers, such as Content Security Policy (CSP), Strict-Transport-Security (HSTS), and others.
- Use HTTPS: Always use HTTPS to encrypt data in transit. Obtain an SSL certificate for your domain and configure your server to use HTTPS.
- Secure Database Access: Use parameterized queries or prepared statements to prevent SQL injection attacks. Ensure that your database credentials are secure and not exposed in configuration files.

**40. What is Express router() function?**

The express.Router() function is used to create a new router object. This function is used when you want to create a new router object in your program to handle requests.

express.Router( [options] )

**41. What are the different types of HTTP requests?**

The primary HTTP methods are commonly referred to as CRUD operations, representing Create, Read, Update, and Delete. Here are the main HTTP methods:

- **GET:** The GET method is used to request data from a specified resource.
- **POST**: The POST method is used to submit data to be processed to a specified resource.
- **PUT:** The PUT method is used to update a resource or create a new resource if it does not exist.

- **PATCH**: The PATCH method is used to apply partial modifications to a resource.
- **DELETE:** The DELETE method is used to request that a specified resource be removed.

**42. Do Other MVC frameworks also support scaffolding?**

The Scaffolding technique is supported by other MVC frameworks also which includes- Ruby on Rails, OutSystems Platform, Play framework, Django, MonoRail, Brail, Symfony, Laravel, CodeIgniter, YII, CakePHP, Phalcon PHP, Model-Glue, PRADO, Grails, Catalyst, Seam Framework, Spring Roo, ASP.NET, etc.

**43. Which are the arguments available to an ExpressJS route handler function?**

In ExpressJS route handler function, there are mainly3 arguments available that provide useful information and functionality.

- **req:** This represents the HTTP request object which holds information about the incoming request. It allows you to access and manipulate the request data.
- **res:** This represents the HTTP response object which is used to send the response back to the client. It provides methods and properties to set response headers, status codes, and send the response body.
- **next**: This is a callback function that is used to pass control to the next middleware function in the request-response cycle.

**44. How can you deal with error handling in ExpressJS?**

ExpressJS provides built-in error-handling mechanism with the help of the next() function. When an error occurs, you can pass it to the next middleware or route handler using the next() function. You can also add an error-handling middleware to your application that will be executed whenever an error occurs.

**45. What is the difference between a traditional server and an ExpressJS server?**

| **Feature** | **Traditional Server (PHP, Java, .NET)** | **ExpressJS Server (NodeJS)** |
|---|---|---|
| **Language** | Uses languages like PHP, Java, C#, Python. | Uses JavaScript (NodeJS). |
| **Architecture** | Multi-threaded, blocking I/O. | Single-threaded, non-blocking I/O. |

| **Feature** | **Traditional Server (PHP, Java, .NET)** | **ExpressJS Server (NodeJS)** |
|---|---|---|
| **Performance** | Slower due to thread-based handling. | Faster due to event-driven, async processing. |
| **Routing Mechanism** | Routing is predefined and handled differently for each language. | ExpressJS provides a built-in and flexible routing system. |
| **Used For** | Enterprise applications, legacy systems, large-scale applications. | APIs, SPAs, microservices, real-time applications. |

**46. What is the purpose of the next() function in ExpressJS?**

The next() function is used to pass control from one middleware function to the next function. It is used to execute the next middleware function in the chain. If there are no next middleware function in the chain then it will give control to router or other functions in the app. If you don't call next() in a middleware function, the request-response cycle can be terminated, and subsequent middleware functions won't be executed.

**47. What is the difference between app.route() and app.use() in ExpressJS?**

| **Feature** | **app.route()** | **app.use()** |
|---|---|---|
| **Purpose** | Defines multiple HTTP methods (GET, POST, PUT, etc.) for a single route. | Mounts middleware or routers to handle requests. |
| **Middleware Support** | Does not apply middleware; only handles route-specific logic. | Used to apply middleware functions like authentication, logging, or parsing request bodies. |
| **Routing Scope** | Specific to a single route. | Can apply to multiple routes or all requests. |
| **Example Usage** | javascript app.route('/user') .get((req, res) => res.send('GET User')) | javascript app.use('/user', (req, res, next) => { |

| **Feature** | **app.route()** | **app.use()** |
|---|---|---|
| | .post((req, res) => res.send('POST User')) .put((req, res) => res.send('PUT User')); | console.log('Middleware for /user'); next(); }); |
| **Used For** | When multiple HTTP methods need to be handled for the same path. | When applying middleware globally or to a group of routes. |

**48. Explain what dynamic routing is in ExpressJS.**

Dynamic routing in ExpressJS include parameters, which allows you to create flexible and dynamic routes in your web application. This parameters are used in your route handlers to customize the behaviour based on the data provided.

In Express, dynamic routing is achieved by using route parameters, denoted by a colon (:) followed by the parameter name.

**Here's a simple example:**

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


// Dynamic route with a parameter
app.get('/users/:userId', (req, res) => {
  const userId = req.params.userId;
  res.send(`User ID: ${userId}`);
});


// Start the server
const port = 8000;
app.listen(port, () => {
  console.log(`Server is listening on port ${port}`);
});
```

**49. How to serve static files in ExpressJS?**

In ExpressJS, you can serve static files using the built-in express.static middleware. This middleware function takes the root directory of your static files as an argument and serves them automatically.

**50. What is the use of app.use() in ExpressJS?**

app.use() is used to add middleware functions in an Express application. It can be used to add global middleware functions or to add middleware functions to specific routes.