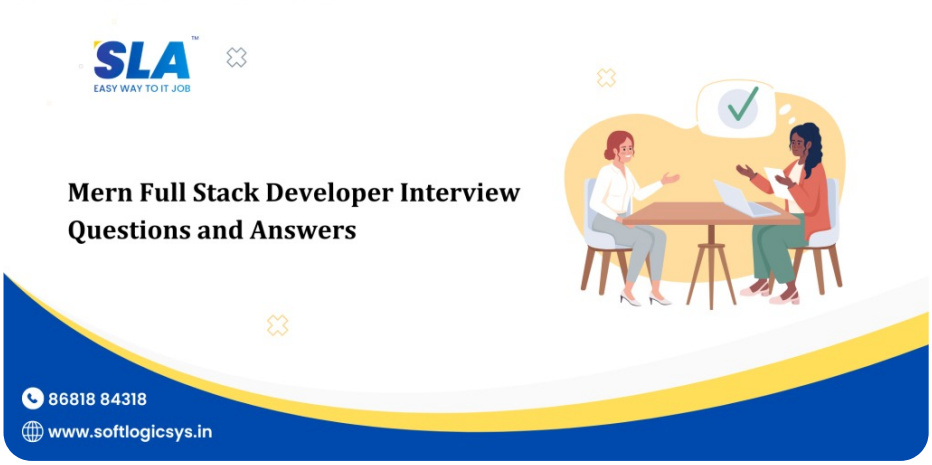

Share on your Social Media

Top 40+ Mern Full Stack Developer Interview Questions and Answers

Published On: April 8, 2024

Mern Full Stack Developer Interview Questions and Answers

MERN is an acronym for Node.js, React, Express, and MongoDB. The well-liked framework for creating mobile and web applications is called the MERN stack. As you know well, the MERN stack consists of Express and Node as middleware, React as the client-side front-end web framework, and MongoDB as the database back end.

Download Mern Full Stack Interview Questions PDF

1. What is the MERN stack used for?

Web applications are developed using a set of JavaScript-based technologies called the MERN stack. MongoDB, Express, React, and Node.js make up the stack. Data in JSON documents can be easily stored and retrieved with MongoDB, a highly scalable document database.

2. What is replication in MongoDB?

The practice of synchronizing data across several MongoDB servers for redundancy, availability, and scalability is called replication. One server serves as the primary server and the rest serve as secondary servers in a replicated MongoDB configuration.

Replication in MongoDB provides data redundancy and fault tolerance, which are essential components of database management.

3. Does MongoDB allow constraints using foreign keys?

No, MongoDB does not provide constraints using foreign keys. However, it has

a feature called manual references, which is designed to mimic the operations of foreign key constraints.

4. What is sharding in MongoDB?

Data partitioning over several servers in a MongoDB cluster is known as sharding. Spreading the data over several servers aids in horizontal database scaling. Sharding is the process of dividing data into smaller units called shards, each of which is kept on a separate server.

Based on the value of the sharding key, MongoDB uses it to identify which shard the data should be stored on.

5. In MongoDB, when is it OK to embed one document inside another?

When two documents in MongoDB have a parent-child relationship and the child document can always be accessed through the parent document, then the two documents should be embedded within each other.

Because the order is always retrieved from the user document, you can embed the order document within the user document if it contains a list of orders. As it cuts down on round trips to the server, embedding documents improves query performance.

6. What are the advantages of BSON over JSON in MongoDB?

Binary JSON, or BSON, is a binary-encoded serialization format for documents that resemble JSON. In MongoDB, BSON provides the following advantages over JSON:

- Additional data types supported by BSON include date, timestamp, and binary data.
- As BSON is more compact than JSON, serialization and deserialization processes run more quickly.
- Faster data access results from BSON's faster traversal speed.

7. In MongoDB, how can transactions be implemented?

MongoDB versions 4.0 and higher support multi-document transactions. The following actions can be used to implement transactions:

- Use the startSession() function to begin a transaction.
- Use the withTransaction() function to perform one or more operations inside the transaction.
- Use the abortTransaction() function to roll back the transaction as a whole if any of its operations fail.
- Use the commitTransaction() function to commit the transaction if all of its activities are successful.

8. How does one query MongoDB using the like operator?

Regular expressions are used in MongoDB to implement the like operator. In an aggregate query, you can use the $regex operator in the $match pipeline stage to use the like operator.

For example, all documents in the myCollection collection where the name

Top 20 AWS DevOps Interview Questions and Answers

Published On: June 19, 2024

AWS DevOps Interview Questions and Answers The integration of AWS and DevOps is doing magic...

Top 14 MEAN Stack Interview Questions and Answers

Published On: June 19, 2024

MEAN Stack Interview Questions and Answers Since MEAN Stack combines several other applications as part...

field begins with the string “Ruth” are found by running the following query:

db.myCollection.aggregate([

{ $match: { name: { $regex: “^Ruth” } } }

])

9. Describe the Aggregation Pipeline in MongoDB.

A MongoDB architecture called the Aggregation Pipeline enables you to process data records and provide computed results. Filters, projections, grouping, and sorting are some of the steps that the pipeline uses to process data. Up until the final output is obtained, each step receives an input, processes the data, and then transfers the output to the subsequent stage.

10. How can the value of one field be used to update another in a MongoDB database?

To refer to the value of the other field, use the $set operator and the $ notation. For instance, you can use the following update statement to update the field “field1” with the value of “field2” multiplied by two:

```
db.collection.update(

{ },

{ $set: { field1: { $multiply: [ “$field2”, 2 ] } }
},

{ multi: true }

)
```

11. When is Redis a better option than MongoDB, or vice versa?

Applications that need fast data access, such as real-time messaging systems or cache layers, are usually better served by Redis, whereas applications that need more sophisticated queries and support for a larger variety of data types are better served by MongoDB.

12. In MongoDB, why is a Covered Query important?

In MongoDB, a covered query is crucial since it can greatly enhance query efficiency by delivering only the necessary fields and obviating the need to retrieve the complete document. This can speed up reaction times and lessen network traffic.

13. In MongoDB, how do you locate a document that has an array with a particular value in it?

In MongoDB, you can locate a document with an array that has a particular value by using the $elemMatch operator. For example, you can use the following query to locate documents in the “collection” collection that have

an array field “field1” with the value “value1” in it:

db.collection.find({ field1: { $elemMatch: { $eq: “value1” } } })

14. In the CAP theorem, where does MongoDB stand?

MongoDB belongs to the CAP theorem’s AP (Availability-Partition tolerance) category. This means that in some scenarios—like network partitions or node failures—MongoDB gives availability and partition tolerance precedence over consistency.

On the other hand, MongoDB does provide developers with configurable consistency choices, enabling them to modify the consistency level under their particular use case requirements.

15. What does it mean to separate an Express application from a Node.js server?

A Node.js application can have greater flexibility and scalability if the Express app and server are kept apart. You can construct additional instances of the server to manage greater traffic by separating the server and the application, or you can use a load balancer to divide traffic across numerous instances.

Separating the application from the server can also facilitate testing and maintenance since server modifications can be made without affecting the logic of the application.

16. How do React Hooks work?

React version 16.8 added a feature called React Hooks, which lets developers leverage React capabilities like state in functional components. State and other React features were previously limited to usage in class components. Hooks offers a mechanism for managing lifecycle functions and component states without requiring the use of class components. UseState, useEffect, useContext, and useReducer are a few examples of hooks.

17. What advantages does utilizing ReactJS offer?

Using ReactJS has several advantages, such as:

- Reusable parts that help cut down on code duplication and ease development.
- Virtual DOM rendering, which is efficient.
- Declarative programming, which facilitates reasoning through code, has a large community and ecosystem with a wealth of third-party libraries and tools.
- Server-side rendering and lazy loading are examples of performance enhancements.
- Cross-platform interoperability: React is suitable for desktop, mobile, and online apps.

18. In React, what are Higher-Order components (HOC) and how do they function?

React’s Higher-Order Components (HOC) pattern makes it possible to reuse components. These are functions that return a new component after accepting an argument in the form of a component. The new component, which has more capability added to it, is referred to as an expanded component. HOCs manage cross-cutting issues, including permission,

logging, and caching. They function by enclosing the original component in a function that either adds more props or changes the original component's behavior.

19. What are the advantages of using React Hooks?

React hooks have many advantages. Some of them are:

- Enhanced reusability and structuring of the code.
- Component logic was made simpler with less boilerplate code.
- Simpler component testing and debugging.
- Improved output through a decrease in the requirement for class materials.
- The capacity to distribute stateful logic among several parts.
- Hooks allow functional components to access state and lifecycle functions that were previously only available to class components.

20. What are the disadvantages of React?

React has certain drawbacks, including:

- The challenging learning curve for newcomers.
- Client-side rendering results in limited support for SEO.
- Performance problems in complicated and large-scale applications.
- Restricted support for rendering on the server.
- Requires further libraries, such as Redux for state management, to function fully.
- Additionally, React is not backward compatible, which means that modifications made to the codebase while updating to a newer version of React may not work as intended.

21. What distinguishes functional components from class components?

React's class and functional components are different in a few ways:

Syntax: While functional components use function syntax, class components use class syntax.

State: Before React Hooks, functional components lacked state; class components, however, possessed state.

Lifecycle Methods: Class components can utilize lifecycle methods with React Hooks, whereas functional components can use lifecycle methods with componentDidUpdate and componentDidMount.

Performance: Because they do not have to worry about the overhead of creating and maintaining a class instance, functional components are typically faster than class components.

22. What is ReactJS's reconciliation?

When a component's state or props change, React updates the DOM (Document Object Model). This process is known as reconciliation. React performs this process by comparing a component's prior and current states and updating the DOM only when required. This procedure helps in lowering

the quantity of DOM updates, enhancing the application's efficiency.

23. What are the pure components in React?

React components classified as pure components only render when the input props have changed. These elements are performance-optimized because they avoid needless re-renders when the props remain unchanged.

A pure component can be either a functional component that makes use of the higher-order *'React.memo'* component or a class component that extends the *'React.PureComponent'* class.

24. What are the advantages and significance of utilizing a key in lists?

A unique attribute called "key" in React is used to uniquely identify each element in a list. React uses the key to determine whether elements have changed, been added, or been removed.

React can speed up the rendering process by reusing preexisting elements rather than producing new ones by assigning a unique key to each list item.

Advantages of listing keys in lists:

- **Efficient rendering:** When the list is refreshed, React can rapidly ascertain which elements require updating.
- **Improved user experience:** The user experience is enhanced by minimizing needless re-renders.
- **Easy debugging:** Debugging is made easier because React can utilize the key to pinpoint the exact component that is generating an error.

25. What is React in the MERN stack?

User interfaces are made with the React JavaScript library. It efficiently manages user interface updates by just re-rendering the modified portions of the DOM, due to the use of a virtual DOM (Document Object Model).

Each component in React's component-based architecture is in charge of rendering a particular area of the user interface. Combining and reusing the parts allows for the creation of more complex user interfaces.

26. What distinguishes virtualDOM from shadowDOM?

- A web standard called ShadowDOM makes it possible to encapsulate and isolate HTML and CSS code from the rest of the page. It enables programmers to design unique HTML elements with unique behaviors and styles.
- VirtualDOM, on the other hand, is a condensed memory representation of the real DOM. React uses it to minimize DOM changes to maximize rendering performance.

27. What distinguishes React's useRef from createRef?

React elements and components in class components are referenced using createRef, while React elements and components in functional components are referenced using useRef. Furthermore, useRef is used for elements that can be mounted more than once, while createRef is used to create a reference to an element that is mounted only once.

28. Do hooks take the place of render properties and higher-

order components (HOC)?

Indeed, hooks can take the place of higher-order components and render properties. Developers no longer need to use render properties or HOCs when creating bespoke, reusable logic in functional components due to React's Hooks feature.

29. How can you use ReactJS validation on props?

Using propTypes, you may apply validation to props in ReactJS. PropTypes is a type-checking library that lets you define what kind of props a component expects. It assists in making sure the appropriate props are supplied to a component and offers an error message if the prop type is not proper.

30. What does React's super (props) function do?

In a React component, super(props) is used to execute the parent class's constructor and give the props object as an argument. If you wish to access the props object within your component's constructor, you must do this.

31. Why would event handlers in React need to be bound to this?

To guarantee that a custom function gets access to the state and props of a React component, you must bind it to this in the component's constructor when specifying it as an event handler.

32. How does Node.js's libuv function underneath?

A library called libuv gives Node.js cross-platform asynchronous I/O capabilities. It provides Node.js with a uniform API by abstracting away the variations in how various operating systems execute I/O activities. Under the hood, it makes effective use of operating system-specific technologies like kqueue, IOCP, and epoll to manage I/O events.

33. When is Node.js not the right choice?

Node.js is meant to handle I/O-intensive operations, so it might not be the best option for CPU-intensive tasks like data processing or video encoding. Furthermore, it might not be the ideal option for applications that demand a lot of memory or require real-time performance.

34. Is it feasible to use the Node.js class?

Yes, you can use classes in Node.js. The ES6 syntax, which includes classes and allows for the creation of objects with shared properties and functions, is supported by Node.js. It is important to remember, though, that the primary purpose of Node.js is to use the functional programming paradigm found in JavaScript.

35. In Node.js, what is a blocking code?

A blocking code stops the program from running until a particular task is completed. Because this code waits for a response before continuing to the next line of code, it impedes the speed of the program. Non-blocking I/O operations are used in Node.js to avoid writing blocking code.

36. Explain what a stream is and the many kinds of streams that Node.js offers.

A stream in Node.js is an uninterrupted data flow from a source to a

destination. Using streams, processing massive volumes of data in small portions is effective and eliminates the need to load the complete dataset into memory.

Node.js has four different kinds of streams:

Readable stream: A file or network socket that allows data to be read is known as a readable stream.

Writable stream: A file or network socket that allows data to be written to it.

Duplex stream: A stream that can be read and written to, like a TCP socket, is called a duplex stream.

Transform stream: A unique kind of duplex stream known as a "transform stream" can alter or transform data as it moves through it. The crypto module, which encrypts and decrypts data, and the zlib module, which compresses and decompresses data, are two examples.

37. What unique characteristics does Node.js offer?

There are numerous noteworthy aspects of Node.js, including:

Asynchronous and event-driven: Node.js is lightweight and efficient because it employs an event-driven, non-blocking I/O paradigm.

Server-side rendering: JavaScript applications may be rendered server-side with Node.js, which improves SEO and speeds up load times.

Cross-platform: Node.js is compatible with a variety of operating systems, including Linux, macOS, and Windows.

Extensible: Packages found in the npm registry can be used to extend Node.js.

Scalable: Node.js has high throughput and can manage several concurrent connections.

38. Convert promise-based Node.js apps to async/await applications.

The procedures listed below can be used to change a promise-based Node.js application to async/await:

- To define an asynchronous function, place the async keyword before the function name.
- Inside the async function, swap out the promise chain for '*await*' expressions.
- '*Try-catch*' blocks are used to handle the errors.

Example

> *async function getUsers() {*

```
try {
const response = await fetch('https://jsonplaceholder.typicode.com/users');
const data = await response.json();
console.log(data);
} catch (error) {
console.error(error);
}
}
```

39. How can callback hell be avoided in Node.js?

The term "callback hell" describes the state in which a code becomes challenging to read and update due to several nested callback levels. The following strategies can help you stay out of callback hell with Node.js:

- Instead of using anonymous functions, use named functions.
- Divide more complicated functions into more manageable ones.
- To manage asynchronous operations, use control flow tools like Promises or Async.js.

40. How are child threads handled in Node.js?

Node.js is an event-driven architectural single-threaded environment. Node.js uses the child_process module to manage child threads. The four methods in this module to create child threads are fork(), spawn(), exec(), and execFile(). Every method starts a fresh child process that has its own execution environment and memory.

41. How does Node.js concurrency work?

Despite being a single-threaded environment, Node.js's event-driven

architecture allows for concurrency. Node.js handles incoming requests asynchronously through the use of an event loop. Upon receiving a request, Node.js adds it to the queue of the event loop and proceeds to handle the subsequent request. Node.js sends a callback to the event loop upon completion of the request, which in turn calls the relevant function.

42. Explain the asynchronous API.

An application programming interface (API), known as an asynchronous API, enables software to carry out its operations without waiting for the conclusion of a prior job. It offers an event-driven programming architecture that is non-blocking and appropriate for managing lengthy processing activities like database access, file and network input/output, and so on.

43. Can you explain what causes callback hell?

An instance of nested callbacks becoming excessively complex and challenging to read and manage is referred to as "Callback Hell." When a callback function is called inside another callback function, the resulting code becomes more intricate and multilayered. This happens when callback nesting gets too deep in code due to improper planning, rendering the code illegible.

44. Differentiate between calling and returning a callback?

A callback function can be called to indicate that it is being executed and supplied as an argument to another function, or it can be returned as a value from another function. While in the second scenario, the callback function is called in the context of the function receiving it as a return value, in the first scenario, it is executed in the context of the calling function.

45. What does prop drilling mean and how can it be avoided?

- The process of sending data through multiple intermediary components from a parent component to a deeply nested child component is known as prop drilling.
- A lot of repetitive and needless code may result from this, which may be challenging to maintain. Redux or context can be used to avoid prop drilling.
- Context offers an alternative to manually passing props down at each level of the component tree to transfer data through it.
- You can store and manage the state of your application in one place using Redux, a state management library.

46. Describe the outcome of running a certain code sample.

Explaining the outcome of running it is impossible without the code sample to refer to.

47. How is JavaScript code compiled in V8?

V8 uses a just-in-time (JIT) compiler to translate JavaScript code into machine code. Before producing optimized machine code, it parses the JavaScript code into an abstract syntax tree (AST) and applies several optimizations.

Softlogic Academy

Softlogic Systems

KK Nagar [Corporate Office]

No.10, PT Rajan Salai, K.K. Nagar, Chennai – 600 078.
Landmark: Karnataka Bank Building
Phone: +91 86818 84318
Email: enquiry@softlogicsys.in
Map: Google Maps Link

OMR

No. E1-A10, RTS Food Street
92, Rajiv Gandhi Salai (OMR),
Navalur, Chennai - 600 130.
Landmark: Adj. to AGS Cinemas
Phone: +91 89256 88858
Email: info@softlogicsys.in
Map: Google Maps Link

Navigation

- About Us
- Blog Posts
- Careers
- Contact
- Placement Training
- Corporate Training
- Hire With Us
- Job Seekers
- SLA's Recently Placed Students
- Reviews
- Sitemap

Important Links

- Disclaimer
- Privacy Policy
- Terms and Conditions

Courses

- Python
- Software Testing
- Full Stack Developer
- Java
- Power BI
- Clinical SAS
- Data Science
- Embedded
- Cloud Computing
- Hardware and Networking
- VBA Macros
- Mobile App Development
- DevOps

Social Media Links

Review Sources

- Google
- Trustpilot
- Glassdoor
- Mouthshut
- Sulekha
- Justdial
- Ambitionbox
- Indeed
- Software Suggest
- Sitejabber