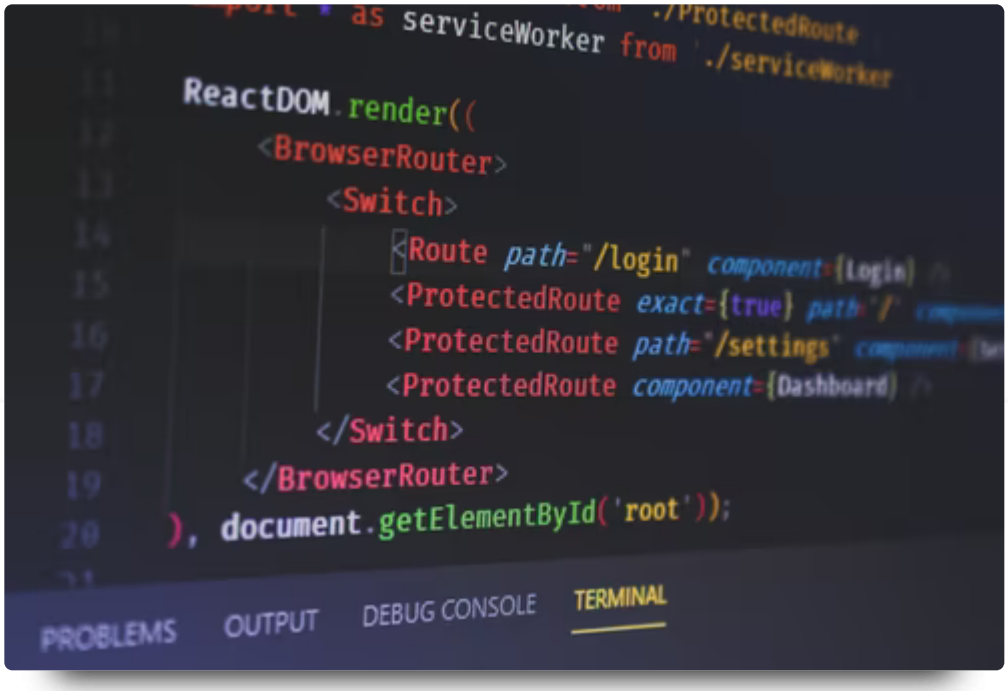


66 MERN Stack Interview Questions (ANSWERED) To Nail Your Next Tech Interview

MongoDB 83 Node.js 88 React 155

> variations of the MEAN stack (MongoDB Express Angular Node), where the traditional Angular.js frontend framework is replaced with React.js. Follow along and check 66 most common MERN Stack Interview Questions you are most likely will be asked on your next MERN Developer interview.

Q1: How does React work?

Entry React 155

Answer

React creates a virtual DOM. When state changes in a component it firstly runs a "diffing" algorithm, which identifies what has changed in the virtual DOM. The second step is reconciliation, where it updates the DOM with the results of diff.

Having Tech or Coding Interview? Check 👉 155 React Interview Questions

Source: github.com/Pau1fitz

Q2: What are the advantages of ReactJS?

Entry React 155

Answer

Below are the advantages of ReactJS:

1. Increases the application's performance with Virtual DOM
2. JSX makes code is easy to read and write
3. It renders both on client and server side
4. Easy to integrate with other frameworks (Angular, BackboneJS) since it is only a view library
5. Easy to write UI Test cases and integration with tools such as JEST.

Having Tech or Coding Interview? Check 👉 155 React Interview Questions

Q3: What is `props` in React?

Answer

Props are inputs to a React component. They are single values or objects containing a set of values that are passed to React Components on creation using a naming convention similar to HTML-tag attributes. i.e, *They are data passed down from a parent component to a child component.*

The primary purpose of props in React is to provide following component functionality:

1. Pass custom data to your React component.
2. Trigger `state` changes.
3. Use via `this.props.reactProp` inside component's `render()` method.

For example, let us create an element with reactProp property,

```
<Element reactProp = "1" />
```

This `reactProp` (or whatever you came up with) name then becomes a property attached to React's native props object which originally already exists on all components created using React library.

```
props.reactProp;
```

Having Tech or Coding Interview? Check 👉 155 React Interview Questions

Source: https://github.com/sudheerj

Q4: Which are the most important features of MongoDB?

Entry MongoDB 83

Answer

- Flexible data model in form of documents
- Agile and highly scalable database
- Faster than traditional databases
- Expressive query language

Having Tech or Coding Interview? Check 👉 83 MongoDB Interview Questions

Source: tutorialspoint.com

Answer

For example, AngularJS (1.x) approaches building an application by extending HTML markup and injecting various constructs (e.g. Directives, Controllers, Services) at runtime. As a result, AngularJS is very opinionated about the greater architecture of your application — these abstractions are certainly useful in some cases, but they come at the cost of flexibility.

By contrast, React focuses exclusively on the creation of components, and has few (if any) opinions about an application's architecture. This allows a developer an incredible amount of flexibility in choosing the architecture they deem "best" — though it also places the responsibility of choosing (or building) those parts on the developer.

Having Tech or Coding Interview? Check 👉 155 React Interview Questions

Source: codementor.io

15+ Ultimate Software Architecture Interview Questions (ANSWERED)

Design Patterns 45 DevOps 44 Software Architecture 70

Q6: What Is Replication In MongoDB?

Junior MongoDB 83

Answer

Replication is the process of synchronizing data across multiple servers. Replication provides redundancy and increases data availability. With multiple copies of data on different database servers, replication protects a database from the loss of a single server. Replication also allows you to recover from hardware failure and service interruptions.

Having Tech or Coding Interview? Check 👉 83 MongoDB Interview Questions

Source: interviewbubble.com

Q7: What are *Higher-Order*

Junior React 155

Answer

A higher-order component **(HOC)** is a function that takes a component and returns a new component. Basically, it's a pattern that is derived from React's compositional nature We call them as **"pure' components"** because they can accept any dynamically provided child component but they won't modify or copy any behavior from their input components.

```
const EnhancedComponent = higherOrderComponent(WrappedComponent);
```

HOC can be used for many use cases as below,

1. Code reuse, logic and bootstrap abstraction
2. Render High jacking
3. State abstraction and manipulation
4. Props manipulation

Having Tech or Coding Interview? Check 155 React Interview Questions

Source: github.com/sudheerj

Q8: What are the differences between a *class component* and *functional component*?

Junior React 155

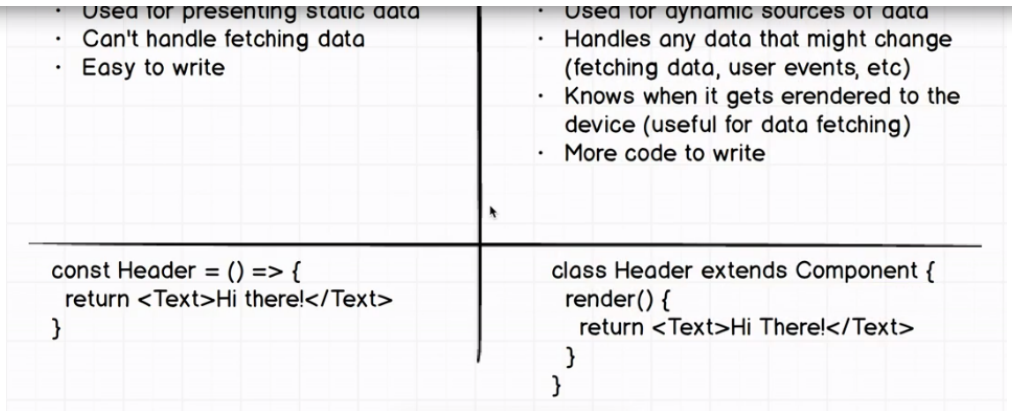
Answer

Class Components

- Class-based Components uses ES6 class syntax. It can make use of the lifecycle methods.
- Class components extend from React.Component.
- In here you have to use this keyword to access the props and functions that you declare inside the class components.

Functional Components

- Functional Components are simpler comparing to class-based functions.
- Functional Components mainly focuses on the UI of the application, not on the behavior.
- To be more precise these are basically render function in the class component.
- Functional Components can have state and mimic lifecycle events using Reach Hooks



Having Tech or Coding Interview? Check 155 React Interview Questions

Source: stackoverflow.com

Q9: What are the key features of Node.js?

Junior | Node.js 88

Answer

Let's look at some of the key features of Node.js.

- **Asynchronous event driven IO helps concurrent request handling –** All APIs of Node.js are asynchronous. This feature means that if a Node receives a request for some Input/Output operation,

- **Fast in Code execution –** Node.js uses the V8 JavaScript Runtime engine, the one which is used by Google Chrome. Node has a wrapper over the JavaScript engine which makes the runtime engine much faster and hence processing of requests within Node.js also become faster.
- **Single Threaded but Highly Scalable –** Node.js uses a single thread model for event looping. The response from these events may or may not reach the server immediately. However, this does not block other operations. Thus making Node.js highly scalable. Traditional servers create limited threads to handle requests while Node.js creates a single thread that provides service to much larger numbers of such requests.
- **Node.js library uses JavaScript –** This is another important aspect of Node.js from the developer's point of view. The majority of developers are already well-versed in JavaScript. Hence, development in Node.js becomes easier for a developer who knows JavaScript.
- **There is an Active and vibrant community for the Node.js framework –** The active community always keeps the framework updated with the latest trends in the web development.
- **No Buffering –** Node.js applications never buffer any data. They simply output the data in chunks.

Having Tech or Coding Interview? Check 👉 88 Node.js Interview Questions

Source: techbeamers.com

40+ ADO.NET Interview Questions (ANSWERED) to Know in 2019

ADO.NET 33 SQL 42

Q10: What are the limitations of React?

Junior React 155

Answer

Below are the list of limitations:

1. React is just a view library, not a full-blown framework
2. There is a learning curve for beginners who are new to web development.
3. Integrating React.js into a traditional MVC framework requires some additional configuration
4. The code complexity increases with inline templating and JSX.
5. Too many smaller components leading to over-engineering or boilerplate

---

Having Tech or Coding Interview? Check 👉 155 React Interview Questions

Source: github.com/sudheerj

Answer

All APIs of Node.js library are aynchronous that is non-blocking. It essentially means a Node.js based server never waits for a API to return data. Server moves to next API after calling it and a notification mechanism of Events of Node.js helps server to get response from the previous API call.

Having Tech or Coding Interview? Check 👉 88 Node.js Interview Questions

Source: tutorialspoint.com

Q12: What is Callback Hell?

Junior Node.js 88

Answer

The asynchronous function requires callbacks as a return parameter. When multiple asynchronous functions are chained together then callback hell situation comes up.

Having Tech or Coding Interview? Check 👉 88 Node.js Interview Questions

Source: codeforgeek.com

Q13: What is *Reconciliation*?

Junior React 155

Answer

When a component's props or state change, React decides whether an actual DOM update is necessary by comparing the newly returned element with the previously rendered one. When they are not equal, React will update the DOM. This process is called **reconciliation**.

Having Tech or Coding Interview? Check 👉 155 React Interview Questions

Source: github.com/sudheerj

Q14: What is the difference between

Junior Node.js 88

callback?

Answer

```
return callback();
//some more lines of code; -  won't be executed

callback();
//some more lines of code; - will be executed
```

Of course returning will help the context calling async function get the value returned by callback.

```
function do2(callback) {
    log.trace('Execute function: do2');
    return callback('do2 callback param');
}

var do2Result = do2((param) => {
    log.trace(`print ${param}`);
    return `return from callback(${param})`; // we could use that return
});

log.trace(`print ${do2Result}`);
```

Output:

```
C:\Work\Node>node --use-strict main.js
[0] Execute function: do2
[0] print do2 callback param
[0] print return from callback(do2 callback param)
```

Having Tech or Coding Interview? Check 👉 88 Node.js Interview Questions

Source: stackoverflow.com

Q15: When should we embed one document within another in MongoDB?

Junior MongoDB 83

Answer

You should consider embedding documents for:

- *contains* relationships between entities

Having Tech or Coding Interview? Check 83 MongoDB Interview Questions

Source: tutorialspoint.com

Q16: Does Mongodb Support Foreign Key Constraints?

Mid | MongoDB 83

Answer

No. MongoDB does not support such relationships. The database does not apply any constraints to the system (i.e.: foreign key constraints), so there are no "cascading deletes" or "cascading updates". Basically, in a NoSQL database it is up to you to decide how to organise the data and its relations if there are any.

Having Tech or Coding Interview? Check 83 MongoDB Interview Questions

Source: interviewbubble.com

Q17: Explain advantages of BSON over JSON in MongoDB?

Mid | MongoDB 83

some cases BSON uses even more space than JSON. The reason for this is another of the BSON design goals: traversability. BSON adds some "extra" information to documents, like length of strings and subobjects. This makes traversal faster.

- BSON is also designed to be fast to encode and decode. For example, integers are stored as 32 (or 64) bit integers, so they don't need to be parsed to and from text. This uses more space than JSON for small integers, but is much faster to parse.
- In addition to compactness, BSON adds additional data types unavailable in JSON, notably the BinData and Date data types.

Having Tech or Coding Interview? Check 👉 83 MongoDB Interview Questions

Source: stackoverflow.com

Q18: Given the code defined above, can you identify two problems?

Mid React 155

Answer

Take a look at the code below:

```
class MyComponent extends React.Component {
  constructor(props) {
    // set the default internal state
    this.state = {
      clicks: 0
    };
  }

  componentDidMount() {
    this.refs.myComponentDiv.addEventListener('click', this.clickHandler);
  }

  componentWillUnmount() {
    this.refs.myComponentDiv.removeEventListener('click', this.clickHandler);
  }

  clickHandler() {
    this.setState({
      clicks: this.clicks + 1
```

```
    let children = this.props.children;

    return (
      <div className="my-component" ref="myComponentDiv">
      <h2>My Component ({this.state.clicks} clicks})</h2>
      <h3>{this.props.headerText}</h3>
    {children}
    </div>
    );
  }
}
```

Given the code defined above, can you identify two problems?

Answer: 1. The constructor does not pass its props to the super class. It should include the following line:

```
constructor(props) {
  super(props);
  // ...
}
```

2. The event listener (when assigned via `addEventListener()`) is not properly scoped because ES2015 doesn't provide autobinding. Therefore the developer can re-assign `clickHandler` in the constructor to include the correct binding to this:

```
constructor(props) {
  super(props);
  this.clickHandler = this.clickHandler.bind(this);
  // ...
}
```

Having Tech or Coding Interview? Check 👉 155 React Interview Questions

Source: codementor.io

Q19: How Node prevents blocking code?

Mid Node.js 88

Answer

By providing callback function. Callback function gets called whenever corresponding event triggered.

Having Tech or Coding Interview? Check 👉 88 Node.js Interview Questions

Source: tutorialspoint.com

Answer

To achieve concepts of transaction and locking in MongoDB, we can use the nesting of documents, also called embedded (or sub) documents. MongoDB supports atomic operations within a single document.

Having Tech or Coding Interview? Check 83 MongoDB Interview Questions

Source: tutorialspoint.com

Q21: How does Node.js handle child threads?

Mid Node.js 88

Answer

Node.js, in its essence, is a single thread process. It does not expose child threads and thread management methods to the developer. Technically, Node.js does spawn child threads for certain tasks such as asynchronous I/O, but these run behind the scenes and do not execute any application JavaScript code, nor block the main event loop.

If threading support is desired in a Node.js application, there are tools available to enable it, such as the ChildProcess module.

Having Tech or Coding Interview? Check 88 Node.js Interview Questions

Source: lazyquestion.com

Q22: How to avoid Callback Hell in Node.js?

Mid Node.js 88

Answer

process finishes its execution, it triggers the callback associated. Sometimes, it could lead to complex and unreadable code. More the no. of callbacks, longer the chain of returning callbacks would be.

There are four solutions which can address the callback hell problem:

- **Make your program modular** - It proposes to split the logic into smaller modules. And then join them together from the main module to achieve the desired result.
- **Use `async/await` mechanism** - Async /await is another alternative for consuming promises, and it was implemented in ES8, or ES2017. Async/await is a new way of writing promises that are based on asynchronous code but make asynchronous code look and behave more like synchronous code.
- **Use promises mechanism** - Promises give an alternate way to write async code. They either return the result of execution or the error/exception. Implementing promises requires the use of `.then()` function which waits for the promise object to return. It takes two optional arguments, both functions. Depending on the state of the promise only one of them will get called. The first function call proceeds if the promise gets fulfilled. However, if the promise gets rejected, then the second function will get called.
- **Use generators** - Generators are lightweight routines, they make a function wait and resume via the `yield` keyword. Generator functions uses a special syntax `function* ()`. They can also suspend and resume asynchronous operations using constructs such as promises or `thunks` and turn a synchronous code into asynchronous.

```
function* HelloGen() {
  yield 100;
  yield 400;
}

var gen = HelloGen();

console.log(gen.next()); // {value: 100, done: false}
console.log(gen.next()); // {value: 400, done: false}
console.log(gen.next()); // {value: undefined, done: true}
```

Having Tech or Coding Interview? Check 👉 88 Node.js Interview Questions

Source: techbeamers.com

Android 113 Java 147 ASP.NET Web API 33

Web Security 58 Ruby on Rails 72 OOP 54

Q23: How to query MongoDB with “like”?

Mid MongoDB 83

Answer

I want to query something as SQL's like query:

```
select *
from users
where name like '%m%'
```

How to do the same in MongoDB?

Answer

```
db.users.find({"name": /.*m.*/})
// or
db.users.find({"name": /m/})
```

You're looking for something that contains "m" somewhere (SQL's `'%'` operator is equivalent to Regexp's `'.*'`), not something that has "m" anchored to the beginning of the string.

Having Tech or Coding Interview? Check 83 MongoDB Interview Questions

Source: stackoverflow.com

Q24: If Node.js is single threaded then how it handles concurrency?

Mid Node.js 88

Node internally uses multiple POSIX threads for various I/O operations such as File, DNS, Network calls etc.

When Node gets I/O request it creates or uses a thread to perform that I/O operation and once the operation is done, it pushes the result to the **event queue**. On each such event, **event loop** runs and checks the queue and if the execution stack of Node is empty then it adds the queue result to execution stack.

This is how Node manages concurrency.

Having Tech or Coding Interview? Check 👉 88 Node.js Interview Questions

Source: codeforgeek.com

Q25: Rewrite promise-based Node.js applications to `Async/Await`

Mid Node.js 88

Problem

Rewrite this code to Async/Await:

```
function asyncTask() {
    return functionA()
        .then((valueA) => functionB(valueA))
        .then((valueB) => functionC(valueB))
        .then((valueC) => functionD(valueC))
        .catch((err) => logger.error(err))
}
```

Answer

```
async function asyncTask() {
    try {
        const valueA = await functionA()
        const valueB = await functionB(valueA)
        const valueC = await functionC(valueB)
        return await functionD(valueC)
    } catch (err) {
        logger.error(err)
    }
}
```

Having Tech or Coding Interview? Check 👉 88 Node.js Interview Questions

Source: stackoverflow.com

(ANSWERED) Developers Must Know in 2020

React 155 React Native 72 Reactive Programming 12

Q26: What are Pure Components?

Mid React 155

Answer

PureComponent is exactly the same as Component except that it handles the `shouldComponentUpdate` method for you. When props or state changes, PureComponent will do a shallow comparison on both props and state. Component, on the other hand, won't compare current props and state to next out of the box. Thus, the component will re-render by default whenever `shouldComponentUpdate` is called.

Having Tech or Coding Interview? Check 👉 155 React Interview Questions

Source: https://github.com/sudheerj

Q27: What are React Hooks?

Mid React 155

Answer

Hooks are a new addition in React 16.8. They let you use state and other React features without writing a class. With Hooks, you can extract stateful logic from a component so it can be tested independently and reused. Hooks allow you to reuse stateful logic without changing your component hierarchy. This makes it easy to share Hooks among many components or with the community.

Having Tech or Coding Interview? Check 👉 155 React Interview Questions

Source: reactjs.org

Q28: What are advantages of using React Hooks?

Mid React 155

Answer

Primarily, hooks in general enable the extraction and reuse of stateful logic that is common across multiple components without the burden of higher order components or render props. Hooks allow to easily manipulate the state of our functional component without needing to convert them into class components.

Having Tech or Coding Interview? Check 155 React Interview Questions

Source: hackernoon.com

Q29: What is Aggregation in MongoDB?

Mid MongoDB 83

Answer

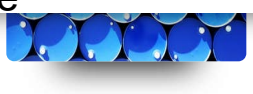
Aggregations operations process data records and return computed results. Aggregation operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result. MongoDB provides three ways to perform aggregation:

- the aggregation pipeline,
- the map-reduce function,
- and single purpose aggregation methods and commands.

Having Tech or Coding Interview? Check 83 MongoDB Interview Questions

Source: tutorialspoint.com

40+ ADO.NET Interview Questions (ANSWERED)



ADO.NET 33 SQL 42

Q30: What is JSX?

Mid React 155

Answer

JSX is a syntax extension to JavaScript and comes with the full power of JavaScript. JSX produces React *elements*. You can embed any JavaScript expression in JSX by wrapping it in curly braces. After compilation, JSX expressions become regular JavaScript objects. This means that you can use JSX inside of if statements and for loops, assign it to variables, accept it as arguments, and return it from functions:

Having Tech or Coding Interview? Check 👉 155 React Interview Questions

Source: github.com/Pau1fitz

Q31: What is ReactDOM?

Mid React 155

Answer

It's a top-level React API to render a React element into the DOM, via the `ReactDOM.render` method.

Having Tech or Coding Interview? Check 👉 155 React Interview Questions

Source: github.com/WebPredict

Q32: What is Sharding in MongoDB?

Mid MongoDB 83

Answer

Sharding is a method for storing data across multiple machines. MongoDB uses sharding to support deployments with very large data sets and high throughput operations.

Having Tech or Coding Interview? Check 👉 83 MongoDB Interview Questions

Source: tutorialspoint.com

Q33: What is *Stream* and what are types of Streams available in Node.js?

Mid Node.js 88

Answer

There is four fundamental type of streams:

- Readable.
- Writeable.
- Duplex.
- Transform.

Readable streams as the name suggest used in reading a large chunk of data from a source. Writable streams are used in writing a large chunk of data to the destination.

Duplex streams are both readable and writable (Eg socket). Transform stream is the duplex stream which is used in modifying the data (eg zip creation).

Having Tech or Coding Interview? Check 👉 88 Node.js Interview Questions

Source: codeforgeek.com

22+ Angular 6 Interview Questions to Stand Out in 2018

Angular 120

Q34: What is *prop drilling* and how can you avoid it?

Mid React 155

Answer

When building a React application, there is often the need for a deeply nested component to use data provided by another component that is much higher in the hierarchy. The simplest approach is to simply pass a prop from each component to the next in the hierarchy from the source component to the deeply nested component. This is called **prop drilling**.

The primary disadvantage of prop drilling is that components that should not otherwise be aware of the data become unnecessarily complicated and are harder to maintain.

To avoid prop drilling, a common approach is to use React context. This allows a `Provider` component that supplies data to be defined, and allows nested components to consume context data via either a `Consumer` component or a `useContext` hook.

Q35: What is `Key` and benefit of using it in lists?

Mid React 155

Answer

A **key** is a special string attribute you need to include when creating lists of elements. Keys help React identify which items have changed, are added, or are removed.

For example, most often we use IDs from your data as keys

```
const todoItems = todos.map((todo) =>
    <li key={todo.id}>
    {todo.text}
    </li>
);
```

When you don't have stable IDs for rendered items, you may use the item index as a key as a last resort:

```
const todoItems = todos.map((todo, index) =>
    <li key={index}>
        {todo.text}
    </li>
);
```

Note:

1. We don't recommend using indexes for keys if the order of items may change. This can negatively impact performance and may cause issues with component state
2. If you extract list item as separate component then apply keys on list component instead li tag.

There will be a warning in the console if the key is not present on list items.

Having Tech or Coding Interview? Check 155 React Interview Questions

Source: github.com/sudheerj

Q36: What is a *Blocking Code*?

Mid Node.js 88

Answer

If application has to wait for some I/O operation in order to complete its execution any further then the code responsible for waiting is known as blocking code.

Q37: What is the alternative of binding `this` in the constructor?

Mid | React 155

Answer

You can use property initializers to correctly bind callbacks. This is enabled by default in create react app. You can use an arrow function in the callback. The problem here is that a new callback is created each time the component renders.

Having Tech or Coding Interview? Check 155 React Interview Questions

Source: github.com/Pau1fitz

Q38: What is the difference between

Mid | React 155

Answer

Virtual DOM

Virtual DOM is about avoiding unnecessary changes to the DOM, which are expensive performance-wise, because changes to the DOM usually cause re-rendering of the page. Virtual DOM also allows to collect several changes to be applied at once, so not every single change causes a re-render, but instead re-rendering only happens once after a set of changes was applied to the DOM.

Shadow DOM

Shadow dom is mostly about encapsulation of the implementation. A single custom element can implement more-or-less complex logic combined with more-or-less complex DOM. An entire web application of arbitrary complexity can be added to a page by an import and `<body><my-app></my-app>` but also simpler reusable and composable components can be implemented as custom elements where the internal representation is hidden in the shadow DOM like `<date-picker></date-picker>`.

Having Tech or Coding Interview? Check 155 React Interview Questions

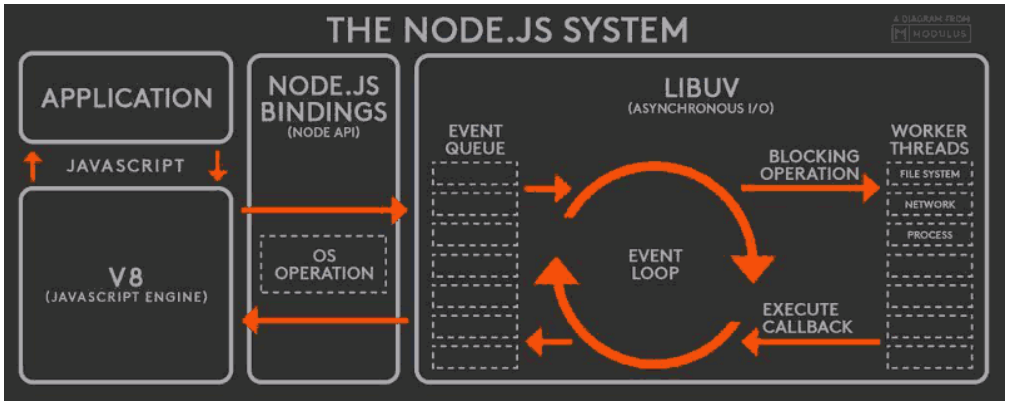
Source: stackoverflow.com

Q39: What's the *Event Loop*?

Mid Node.js 88

Answer

The event loop is what allows Node.js to perform non-blocking I/O operations — despite the fact that JavaScript is single-threaded — by offloading operations to the system kernel whenever possible.



Every I/O requires a callback - once they are done they are pushed onto the event loop for execution. Since most modern kernels are multi-threaded, they can handle multiple operations executing in the background. When one of these operations completes, the kernel tells Node.js so that the appropriate callback may be added to the poll queue to eventually be executed.

Having Tech or Coding Interview? Check 88 Node.js Interview Questions

Q40: What's the difference between `useRef` and `createRef`?

Answer

The difference is:

- `createRef` will always create a new ref. In a class-based component, you would typically put the ref in an instance property during construction (e.g. `this.input = createRef()`). You don't have this option in a function component.
- `useRef` takes care of returning the same ref each time as on the initial rendering.

Having Tech or Coding Interview? Check 👉 155 React Interview Questions

Source: reactjs.org

Q41: What's the difference between a "smart" component and a "dumb" component?

Mid React 155

Answer

- *Smart components* manage their state or in a Redux environment are connected to the Redux store.
- *Dumb components* are driven completely by their props passed in from their parent and maintain no state of their own.

Having Tech or Coding Interview? Check 👉 155 React Interview Questions

Source: github.com/WebPredict

Q42: Do Hooks replace render props and higher-order components?

Senior React 155

a simpler way to serve this use case.

There is still a place for both patterns (for example, a virtual scroller component might have a renderItem prop, or a visual container component might have its own DOM structure). But in most cases, Hooks will be sufficient and can help reduce nesting in your tree.

Having Tech or Coding Interview? Check 👉 155 React Interview Questions

Source: reactjs.org

Q43: How replication works in MongoDB?

Senior MongoDB 83

Answer

A replica set consists of a primary node and a secondary node too. With the help of a replica set, all the data from primary node to the secondary node replicates. Replication is a process of synchronizing the data. Replication provides redundancy and it also increases the availability of data with the help of multiple copies of data on the different database server. It also protects the database from the loss of a single server.

Having Tech or Coding Interview? Check 👉 83 MongoDB Interview Questions

Source: interviewbubble.com

in ReactJS?

Answer

When the application is running in development mode, React will automatically check for all props that we set on components to make sure they must right correct and right data type. For incorrect type, it will generate warning messages in the console for development mode whereas it is disabled in production mode due performance impact. The mandatory prop is defined with isRequired.

The set of predefined prop types are below

1. `React.PropTypes.string`
2. `React.PropTypes.number`
3. `React.PropTypes.func`
4. `React.PropTypes.node`
5. `React.PropTypes.bool`

For example, we define propTypes for user component as below,

```
import PropTypes from 'prop-types';

class User extends React.Component {
  render() {
    return (
      <h1>Welcome, {this.props.name}</h1>
      <h2>Age, {this.props.age}
    );
  }
}

User.propTypes = {
  name: PropTypes.string.isRequired,
  age: PropTypes.number.isRequired

};
```


Q45: Is Node.js entirely based on a single-thread?

Senior Node.js 88

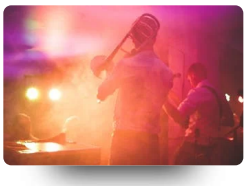
Answer

Moreover, Node.js has an optimized design which utilizes both JavaScript and C++ to guarantee maximum performance. JavaScript executes at the server-side by Google Chrome v8 engine. And the C++ lib UV library takes care of the non-sequential I/O via background workers.

To explain it practically, let's assume there are 100s of requests lined up in Node.js queue. As per design, the main thread of Node.js event loop will receive all of them and forwards to background workers for execution. Once the workers finish processing requests, the registered callbacks get notified on event loop thread to pass the result back to the user.

Having Tech or Coding Interview? Check 👉 88 Node.js Interview Questions

Source: techbeamers.com



22 PWA Interview Questions Every Developer Should Know (2020 REVIEW)

PWA 22

Q46: Is it possible to use `Class` in Node.js?

Senior Node.js 88

Answer

With ES6, you are able to make "actual" classes just like this:

```
class Animal {

    constructor(name) {
        this.name = name;
    }

    print() {
        console.log('Name is :' + this.name);
    }
}
```

You can export a class just like anything else:

```
module.exports = class Animal {

};
```

```
var Animal = require('./Animal');

class Cat extends Animal {
    ...
}
```

Having Tech or Coding Interview? Check 👉 88 Node.js Interview Questions

Source: stackoverflow.com

Q47: Update MongoDB field using value of another field

Senior MongoDB 83

Answer

Consider SQL command:

```
UPDATE Person SET Name = FirstName + ' ' + LastName
```

In Mongo, is it possible to update the value of a field using the value from another field?

Answer

You cannot refer to the document itself in an update (yet). You'll need to iterate through the documents and update each document using a function like:

```
db.person.find().snapshot().forEach(
    function (elem) {
        db.person.update(
            {
                _id: elem._id
            },
            {
                $set: {
                    name: elem.firstname + ' ' + elem.lastname
                }
            }
        );
    }
);
```

Having Tech or Coding Interview? Check 👉 83 MongoDB Interview Questions

Source: stackoverflow.com

super(props)?

Answer

When you pass `props` to `super`, the props get assigned to `this`. Take a look at the following scenario:

```
constructor(props) {
    super();
    console.log(this.props) //undefined
}
```

How ever when you do :

```
constructor(props) {
    super(props);
    console.log(this.props) //props will get logged.
}
```

Note that passing or not passing `props` to `super` has **no effect** on later uses of `this.props` outside `constructor`. That is `render`, `shouldComponentUpdate`, or event handlers **always** have access to it.

Having Tech or Coding Interview? Check 👉 155 React Interview Questions

Source: stackoverflow.com

Q49: When to Redis or MongoDB?

Senior MongoDB 83

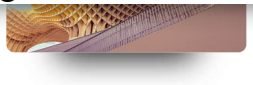
Answer

- **Use MongoDB if you don't know yet how you're going to query your data or what schema to stick with.** MongoDB is suited for Hackathons, startups or every time you don't know how you'll query the data you inserted. MongoDB does not make any assumptions on your underlying schema. While MongoDB is schemaless and non-relational, this does not mean that there is no schema at all. It simply means that your schema needs to be defined in your app (e.g. using Mongoose). Besides that, MongoDB is great for prototyping or trying things out. Its performance is not that great and can't be compared to Redis.
- **Use Redis in order to speed up your existing application.** It is very uncommon to use Redis as a standalone database system (some people prefer referring to it as a "key-value"-store).

Having Tech or Coding Interview? Check 👉 83 MongoDB Interview Questions

Source: stackoverflow.com

35 MongoDB Interview Q... Unlock 3877 (ANSWERED)



MongoDB 83

Q50: When to not use Node.js?

Senior Node.js 88

Answer

We can use Node.js for a variety of applications. But it is a single threaded framework, so we **should not use** it for cases

- **where the application requires long processing time (If the server is doing some calculation)** ,

it won't be able to process any other requests. Hence, Node.js is best when processing needs less dedicated CPU time.

Having Tech or Coding Interview? Check 88 Node.js Interview Questions

Source: techbeamers.com

Q51: Why is a covered query important?

Senior MongoDB 83

return the result fields using the same index without looking inside the documents. Since indexes are stored in RAM or sequentially located on disk, such access is a lot faster.

Having Tech or Coding Interview? Check 👉 83 MongoDB Interview Questions

Source: tutorialspoint.com

Q52: Why would you need to bind event handlers to `this`?

Senior React 155

Answer

Binding is not something that is specifc to React, but rather how `this` works in Javascript. When you define a component using an ES6 class, a common pattern is for an event handler to be a method on the class. In JavaScript, class methods are not bound by default. If you forget to `bind this.someEventHandler` and pass it to `onChange`, this will be undefined when the function is actually called.

Generally, if you refer to a method without `()` after it, such as `onChange={this.someEventHandler}`, you should bind that method.

Having Tech or Coding Interview? Check 👉 155 React Interview Questions

Source: stackoverflow.com

Q53: Explain the result of this code execution

Expert Node.js 88

Answer

Unlock **FullStack.Cafe** to open all answers and get your next figure job offer!

Share this blog post to **open Expert question**!

19+ Advanced Node.js Interview Questions For Senior Developers

Node.js 88

Answer

Unlock **FullStack.Cafe** to open all answers and get your next figure job offer!

Q55: How does libuv work under the hood?

Expert Node.js 88

Answer

Unlock **FullStack.Cafe** to open all answers and get your next figure job offer!

Q56: How to find document with array that contains a specific value?

Expert MongoDB 83

Answer

Unlock **FullStack.Cafe** to open all answers and get your next figure job offer!

Q57: Rewrite the code sample without `try/catch` block

Expert Node.js 88

Answer

Unlock **FullStack.Cafe** to open all answers and get your next figure job offer!

35 Top Angular 7 Interview Questions To Crack in 2019

Angular 120

Q58: Where does MongoDB stand in the CAP theorem?

Expert MongoDB 83

Answer

Unlock **FullStack.Cafe** to open all answers and get your next figure job offer!

Q59: Why should you separate Express *app* and *server*?

Expert Node.js 88

Share this blog post to **open Expert question**!

Unlock 3877 Answers

50+ Mobile Developer Interview Questions (ANSWERED) to Know in 2021

Android 113

Mobile app developers are responsible for developing the applications both on Android and iOS and using all sort of tech including PWA, React Native, Ionic, Xamarin and etc. iOS developers can expect to earn, on average, over $113,000, with some jobs...