Kill Your Tech Interview

3877 Full-Stack, Coding & System Design Interview Questions

Answered To Get Your Next Six-Figure Job Offer

See All Questions

Data Science & ML QAs

Full-Stack, Web & Mobile | Coding & Data Structures | System I

Full-Stack | Coding | System Design

.NET Core 52 | ADO.NET 33 | ASP.NET 42 | ASP.NET MVC 36

ASP.NET Web API 33 | Agile & Scrum 47 | Android 113

Angular 120 | AngularJS 61 | C# 115 | CSS 50

Design Patterns 45 | DevOps 44 | Entity Framework 57

Flutter 68 | Git 36 | Golang 49 | GraphQL 25 | HTML5 55

Ionic 29 | Java 147 | JavaScript 142 | Kotlin 68 | LINQ 38

Laravel 41 | MongoDB 83 | MySQL 55 | Node.js 88

OOP 54 | Objective-C 43 | PHP 82 | PWA 22 | Python 91

React 155 | React Native 72 | Reactive Programming 12

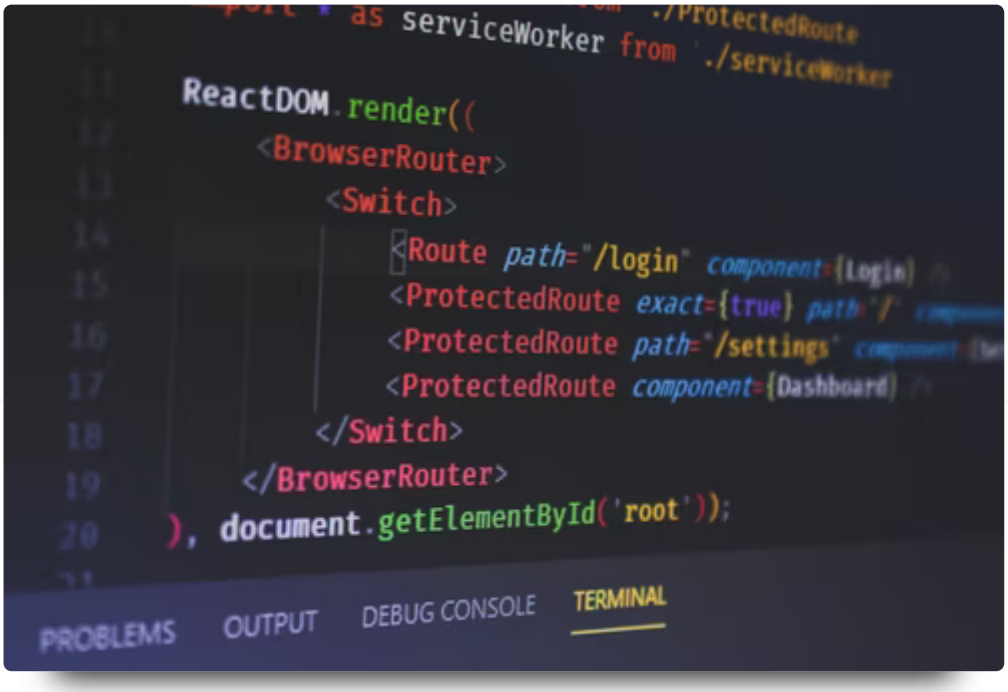
Redis 25 | Redux 30 | Ruby 84 | Ruby on Rails 72

SQL 42 | Software Testing 26 | Spring 87 | Swift 72

T-SQL 51 | TypeScript 39 | UX Design 78 | Vue.js 41

WCF 33 | WPF 46 | Web Security 58 | WebSockets 24

Xamarin 83 | iOS 36 | jQuery 51



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

MongoDB 83 Node.js 88 React 155

> MERN stands for MongoDB, Express, React, Node, after the four key technologies that make up the stack. MERN is one of several 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

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

Source: github.com/sudheerj

Q3: What is `props` in React?

Entry React 155

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.*

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

Q5: How is React different from AngularJS (1.x)?

Junior React 155

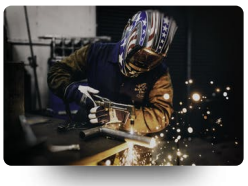
Answer

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

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

Source: codementor.io



27 Angular 9 Interview Questions (with Ivy UPDATE) for 2020

Angular 120

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* components?

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

Junior React 155

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

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

Node.js?

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, it will execute that operation in the background and continue with the processing of other requests. Thus it will not wait for the response from the previous requests.
- **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

27 Advanced DevOps Interview Questions (SOLVED) You Must Know

Availability & Reliability 14 CAP Theorem 13 DevOps 44

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.

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

Q11: What do you mean by Asynchronous API?

Junior Node.js 88

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

9 Basic webpack Interview Questions And Answers in 2019

Q14: What is the difference between returning a callback and just calling a callback?

Junior Node.js 88

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

- One-to-many relationships
- Performance reasons

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

JSON. In 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

20 CI/CD Interview Questions DevOps Engineers Should Know in 2020

DevOps 44 Docker 55 Microservices 34

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
    });
  }
```

```
    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

Q20: How can you achieve transaction and locking in

Mid MongoDB 83

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

40 Advanced OOP Interview Questions and Answers [2019 Update]

OOP 54

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

Mid Node.js 88

Answer

Node.js internally uses a single-threaded event loop to process queued events. But this approach may lead to blocking the entire process if there is a task running longer than expected. Node.js addresses this problem by incorporating callbacks also known as higher-order functions. So whenever a long-running 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.

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

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

Answer

Node provides a single thread to programmers so that code can be written easily and without bottleneck. 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.

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

37 Advanced iOS Developer Interview Questions (SOLVED and EXPLAINED)

Objective-C 43 Swift 72 iOS 36

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

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.

Hooks don't work inside classes (because they let you use React without classes). By using them, we can totally avoid using lifecycle methods, such as `componentDidMount`, `componentDidUpdate`, `componentWillUnmount`. Instead, we will use built-in hooks like `useEffect`.

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,

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

Source: tutorialspoint.com

50 Common Front End Developer Interview Questions [2019 Edition]

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

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

Streams are a collection of data that might not be available all at once and don't have to fit in memory. Streams provide chunks of data in a continuous manner. It is useful to read a large set of data and process it.

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

Know in 2021

Android 113 Design Patterns 45 Ionic 29

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.

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

Source: toptal.com

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:

```
      </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.

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

Source: tutorialspoint.com

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

32 Advanced Ruby on Rails Interview Questions (ANSWERED)

Ruby 84 Ruby on Rails 72

Q38: What is the difference between *ShadowDOM* and *VirtualDOM*?

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

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

Source: blog.risingstack.com

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

Mid React 155

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

Answer

Join **FullStack.Cafe** to open this Answer. It's Free!

Get Free Access To Answer

Join 92k+ Developer Who Trust FullStack.Cafe

Q43: How replication works in MongoDB?

Senior MongoDB 83

Answer

Join **FullStack.Cafe** to open this Answer. It's Free!

Get Free Access To Answer

Join 92k+ Developer Who Trust FullStack.Cafe

Answer

Join **FullStack.Cafe** to open this Answer. It's Free!

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

Senior Node.js 88

Answer

Join **FullStack.Cafe** to open this Answer. It's Free!

10 Big O Interview Questions Every Developer Must Answer

Big-O Notation 22

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

Senior Node.js 88

Answer

Join **FullStack.Cafe** to open this Answer. It's Free!

Q47: Update MongoDB field using value of another field

Senior MongoDB 83

Answer

Join **FullStack.Cafe** to open this Answer. It's Free!

Q48: What is the purpose of `super(props)` ?

Senior React 155

Answer

Join **FullStack.Cafe** to open this Answer. It's Free!

Q49: When to Redis or MongoDB?

Senior MongoDB 83

Answer

Join **FullStack.Cafe** to open this Answer. It's Free!

45 Advanced PHP Interview Questions That May Land You a Job

PHP 82

Q50: When to not use Node.js?

Senior Node.js 88

Answer

Join **FullStack.Cafe** to open this Answer. It's Free!

Q51: Why is a covered query important?

Senior MongoDB 83

Answer

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

Senior React 155

Answer

Q53: Explain the result of this code execution

Expert Node.js 88

Answer

Guest Voice: The Solution For The CV Problem In IT Recruitment

$ Career 1

Q54: How V8 compiles JavaScript code?

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**!

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!

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

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!

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

without `try/catch` block

Answer

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

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

Full-Stack, Web & Mobile | Coding & Data Structures | Sy

Full-Stack | Coding | System Design

Software Testing 26 | Entity Framework 57 | React Native 72

WebSockets 24 | C# 115 | Spring 87 | Swift 72

Node.js 88 | Java 147 | Git 36

15 ASP.NET Web API Interview Questions And Answers (2019 Update)

ASP.NET Web API 33

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!

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

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

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**!

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...

27 Objective-C Interview Questions (ANSWERED) Every iOS Developer Should Know

Objective-C 43

Objective-C is a nice middle ground that gets close to the performance of compiled languages (like C++) while still maintaining some dynamism that you find in interpreted languages (like JavaScript or Ruby). There is still some Objective-C developmen...

37 Advanced iOS Developer Interview Questions (SOLVED and EXPLAINED)

© Objective-C 43

The average iOS Developer salary in Australia is $120,000 per year or $61.54 per hour. Entry level positions start at $66,250 per year while most experienced workers make up to $162,900 per year. Follow along and check 37 most common iOS/Swif...

27 Advanced Swift Interview Questions (SOLVED) iOS Developers Must Know

iOS devices makeup over 50% of the Australian mobile operating system market, and capture around the same market share in the US. And Swift as a skill and iOS Developer roles are intrinsically intertwined and according to ZipRecruiter, the average iO...

Swift 72

and DevOps Developers

Kubernetes has become the go-to orchestration platform since it was launched in 2014. According Cloud Native Computing Foundation (CNCF) survey 84% of companies are using containers in production and the vast majority (78%) are using Kubernetes. Foll...

Kubernetes 27

29 Advanced MySQL Interview Questions Developers Must Know Before Tech Interview

MySQL has emerged as the most popular SQL-based RDBMS in 2019 with almost 40% of users opting for this platform. It is widely used as a stand-alone database solution as well as in combination with other solutions such as MongoDB and PostgreSQL. If yo...

MySQL 55

27 Binary Tree Interview Questions (SOLVED with CODE) Devs Must Know

Amongst different types of data structures are binary trees that come with more uses than most of the other types. The reason that binary trees are used more often than n-ary trees (for example in searching) is that n-ary trees are more complex, but ...

Binary Tree 26

32 Advanced Data Structures Interview Questions (ANSWERED) To Smash Your Next Programming Interview

Most interviews these days are whiteboard interviews demonstrating your knowledge of DSA. Unless you're interviewing for a top company, you can expect the questions to be fairly standard. The reason why questions about Data Structures are asked, has ...

Data Structures 61

40 Coding Challenges (SOLVED with CODE) To Kill Your Next Coding Interview

You may hate code challenges and coding interviews but reality is a lot of companies from Google to Amazon do care that you understand the difference between O(n log n) and O(n^2), that you do understand when different data structures are approp...

15 Hashing Interview Questions (EXPLAINED) To Check Before Next Coding Interview

Hashing is the practice of using an algorithm (or hash function) to map data of any size to a fixed length. This is called a hash value (or sometimes hash code or hash sums or even a hash digest if you're feeling fancy). In hashing, keys are converte...

Hash Tables 31

Load More...

FullStack.Cafe is a biggest hand-picked collection of top Full-Stack, Coding, Data Structures & System Design Interview Questions to land 6-figure job offer in no time.