ReactJS Interview Questions and Answers

Introduction

Knowledge of ReactJS is a necessity in the development of modern and highly efficient interfaces. With this list of ReactJS interview questions and answers, you will learn the fundamental and advanced concepts of ReactJS. This includes JSX, Virtual DOM, Hooks (useState, useEffect), and State Management with Redux and Context API. Knowledge of ReactJS will make the reader an invaluable asset in any modern web and mobile development team. Are you ready to become a certified front-end developer? Develop your technical skills by downloading our professional ReactJS Course Syllabus today.

List of ReactJS Interview Questions for Freshers

1. What is ReactJS?
2. What characteristics does React have?
3. What does React's render() function accomplish?
4. Explain props.
5. In React, what is a state, and how is it used?
6. In React, what are synthetic events?

7. Give an example of when using references.
8. Define Higher Order Components (HOC).
9. Define pure components in React.
10. Explain flux in React.

Enroll in our ReactJS training in Chennai to get started.

ReactJS Interview Questions and Answers for Freshers

1. What is ReactJS?

A JavaScript package called ReactJS is used to create reusable view-layer components in the MVC architecture. It renders components using a virtual DOM and is incredibly efficient. It is developed in JSX and operates on the client side.

2. What characteristics does React have?

The following is a list of React's key features:

- Rather than using the actual DOM, it uses the virtual DOM.
- Server-side rendering is used.
- It complies with data binding or unidirectional data flow.

3. What does React's render() function accomplish?

Render() is a required component for every React component. The representation of the native DOM component is returned in the form of a single React element. When rendering multiple HTML elements, they should be grouped inside an enclosing tag *(e.g., <form>, <group>, , etc.)*. This function needs to remain pure, meaning that every time it is called, it needs to return the same value.

4. Explain props.

Properties in React are shortened to props. They are components that are read-only and need to be kept pure or unchangeable. Throughout the program, they are always transferred from the parent to the child components.

A child component can't transmit a prop back to its parent component. These are typically employed to portray the dynamically generated data and aid in the maintenance of the unidirectional data flow.

5. In React, what is a state, and how is it used?

React components consist of states at their core. States are the objects that control how components are rendered and behave. They are interchangeable, which makes them more dynamic and engaging than the props. This is how you get to *this.state()*.

6. In React, what are synthetic events?

Synthetic events refer to the objects that function as a cross-browser wrapper for the browser's native event. They unify various browsers' behaviors into a unified API. To guarantee that the properties provided by the events are constant across different browsers, this is done.

7. Give an example of when using references.

The following situations call for the use of referees:

- If you need to focus, choose to play text or media.
- To initiate necessary animations
- Connect with external DOM libraries.

8. Define Higher Order Components (HOC).

A more sophisticated method of reusing component logic is using higher-order components. In essence, it's a pattern that comes from the compositional structure of React. Custom components known as HOC enclose another component.

They will not alter or duplicate any behavior from their input components, but they will accept any child component that is dynamically provided. HOC can be referred to as "pure" components.

9. Define pure components in React.

Pure components are the easiest and fastest components that can be written. Any component with merely a render() can be swapped out. These components enhance the functionality and readability of the application's code.

10. Explain flux in React.

The architectural pattern known as flux is what keeps the data flowing in one direction alone. Through the use of a central store with authority over all data, it manages derived data and facilitates communication between various components.

Any data updates across the application must happen here and nowhere else. Run-time errors are decreased, and application stability is increased via flux.

Explore our ReactJS tutorial for beginners and enhance your web development career.

List of ReactJS Interview Questions for Experienced

1. Describe Redux.
2. Describe the function of a reducer.
3. Describe React Router.
4. Why is a router necessary in React?
5. What distinguishes a React static component from a dynamic component?
6. In a React application, how is server-side rendering handled?
7. Describe Higher Order Components (HOCs) in React and their applications.
8. Describe the Redux Thunk idea.
9. How should a React application manage code splitting?
10. What distinguishes a React element from a React component?

Fine-tune your web development skills with our ReactJS project ideas.

ReactJS Technical Interview Questions and Answers for Experienced

1. Describe Redux.

Redux is one of the most widely used front-end programming libraries out there at the moment. It offers a consistent state container and is used for JavaScript applications' overall state management. Redux-developed applications are simple to test and exhibit consistent behavior across many settings.

2. Describe the function of a reducer.

Reducers are pure functions that define how an ACTION affects the state of the application. Reducers function by receiving the action and state from the previous state and returning to the new state.

Depending on the kind of action, it decides what kind of update is required and then returns the updated data. If no work is required, it restores the previous state to its original state.

3. Describe React Router.

React Router is a powerful routing library built on top of React that makes it easier to add new screens and processes to an application.

This keeps both the URL and the information displayed on the webpage current.

It is used to create single-page web applications and has a consistent structure and behavior. It's easy to use React Router's API.

4. Why is a router necessary in React?

A router is used to define several routes. When a user writes a certain URL, the router redirects them to the specified route if the path of the URL matches any of the routes that are defined inside the router.

To put it simply, we need to incorporate a router library into our program, which will enable us to create numerous routes, each of which will lead to a different view.

```
<switch>

  <route exact path='/' component={Home}/>

  <route path='/posts/:id' component={Newpost}/>
```

<route path='/posts' component={Post}/>

</switch>

5. What distinguishes a React static component from a dynamic component?

A component that is defined in React with a fixed set of characteristics or attributes and remains constant throughout its lifecycle is referred to as a static component.

A straightforward JavaScript method that yields a tree of items representing the component's user interface (UI) is used to define static components.

Conversely, a dynamic component can alter its state, behavior, or characteristics in response to user interactions or application-level events.

Usually, a class component or a functional component with the useState or useEffect Hooks is used to define a dynamic component.

Example:

function welcome(props) {

return <h1>Hello, {props.name}</h1>;

}

6. In a React application, how is server-side rendering handled?

- When using React, server-side rendering, or SSR, entails rendering your React components on the server and delivering the HTML result to the client.
- Numerous advantages result from this, such as enhanced performance and search engine optimization (SEO).
- You can use a library like Next.js or Razzle, which offer an intuitive framework for managing SSR, to implement SSR in a React application.
- As an alternative, you can manually render your components on the server by using the ReactDOMServer API.

Among the process's crucial steps are:

- Preparing your server to receive requests, process them, and provide the necessary resources.
- Rendering the components on the server using *ReactDOMServer.renderToString* or *ReactDOMServer.renderToStaticMarkup*.
- Including the generated HTML in the response and sending it to the client.
- Hydrating the client's parts to enable user interaction and control.

Recall that SSR has some added complexity and trade-offs, so you should carefully assess if this is the best option for your application.

7. Describe Higher Order Components (HOCs) in React and their applications.

In React, a function that accepts a component as an input and produces a new component with extra properties is called a Higher Order Component (HOC). Reusing logic across numerous components is the aim of an HOC. A HOC is a React pattern for reusing component code; it is not a "part" of React.

When necessary, use a HOC:

- Share common logic, such as authorization or data collection, among several components.
- Create a reusable HOC by abstracting state and behavior that you may use again throughout your application.
- Render one component inside another and give the wrapped component props.
- The connect HOC from react-redux and the withRouter HOC from react-router are two examples of HOCs.

8. Describe the Redux Thunk idea.

- In Redux, a function that returns another function rather than a simple action object is called a Thunk.
- It is employed to send out several actions and carry out asynchronous tasks.
- You can write action creators with thunks that return functions rather than actions.
- This can help send out numerous actions like one to signal that the API request has begun and another to signal that it has concluded, as well as for carrying out asynchronous tasks like API calls.
- The store's dispatch method, which can be used to dispatch operations at any time in the future, is passed as an argument to the inner function.
- Usually, a middleware, like the redux-thunk middleware, is used to implement thunks.

9. How should a React application manage code splitting?

The following methods can be used to handle code splitting in React:

Dynamic Imports: A component can be loaded asynchronously only when it's required, due to dynamic imports. This gives you the ability to divide code into smaller sections that may be loaded as needed and is accomplished using the import() syntax.

```
import React, { Suspense } from 'react';

const LazyComponent = React.lazy(() =>
import('./LazyComponent'));

function App() {

return (

<div>
```

```
<Suspense fallback={<div>Loading...</div>}>
<LazyComponent />
</Suspense>
</div>
);
}
import React, { lazy, Suspense } from 'react';
import { Route } from 'react-router-dom';
const Home = lazy(() => import('./Home'));
const About = lazy(() => import('./About'));
function App() {
return (
<div>
<Suspense fallback={<div>Loading...</div>}>
<Route exact path="/" component={Home} />
<Route path="/about" component={About} />
```

```
</Suspense>

</div>

);

}
```

The Webpack Bundle Analyzer is a tool that shows the code's size and structure visually. This program helps you find those big blocks of code that can be broken up into smaller pieces and loaded more slowly.

By decreasing the initial loading time and loading the necessary code only when needed, you can efficiently manage code splitting in a React application and enhance its efficiency.

10. What distinguishes a React element from a React component?

A JavaScript class or function that yields a React element is called a React component. It represents a section of the user interface and is a reusable element of UI.

Conversely, a React element is a simple JavaScript object that depicts a node in the DOM. It is an unchangeable depiction of a DOM node that can be made with JSX or React.createElement.

To put it briefly, an element is an instance of a component, and a component is a blueprint for constructing further elements.

Conclusion

Passing a ReactJS interview calls for a high-level understanding of Virtual DOM, Hooks, and State Management concepts. In a ReactJS interview, your potential as a ReactJS developer will be judged based on your understanding and expertise in moving beyond simple ReactJS component development and into Custom Hooks, Higher-Order Components, and Performance Optimization techniques such as memo and useCallback.

Ready to become a certified front-end expert? Master the art of modern web development at our software training institute in Chennai.