React Interview Questions and Answers

Table of Contents

React is an indispensable part of the present-day JavaScript frontend web development industry. For those looking to work with JS on the front-end, having adequate knowledge of React is a necessity.

React stands among the leading JavaScript frameworks and probably will continue to do so going forward. Though regarded as a front-end framework, React is actually a front-end library. In fact, the whole idea of React as a web development framework is so good that it finished charting as one of the leading web development frameworks.

Here, we'll take a look at some common React interview questions and answers.

Top React Interview Questions and Answers

It's important to have both theoretical and practical knowledge of React, hence these interview questions. There is a good mix of React technical interview questions and React advanced interview questions.

1. What is React? What are some of its main features?

React is a front-end JavaScript library that was developed by Facebook in 2011. It enhances application performance while allowing for working on both client-side and server-side.

Writing UI test cases is simple with React, which is also easy to integrate with Angular, Meteor, and other popular JS frameworks. Here are some of the major features of React:

- Excellent for developing complex and interactive web and mobile UI.
- Follows the component-based approach and facilitates reusable UI components.
- Has excellent community support.
- Makes use of the virtual DOM instead of the real DOM.
- Relies on server-side rendering.
- Supports unidirectional data flow or data binding.

2. What is a state and how is it used?

States are the sources of data for React components. They are objects responsible for determining components' behavior and rendering. As such, they must be kept as simple as possible. You can access states with the syntax this.state()

State is mutable and creates dynamic and interactive components. The use of a state can be visualized with the following code snippet:

```
classAppextendsReact.Component {
```

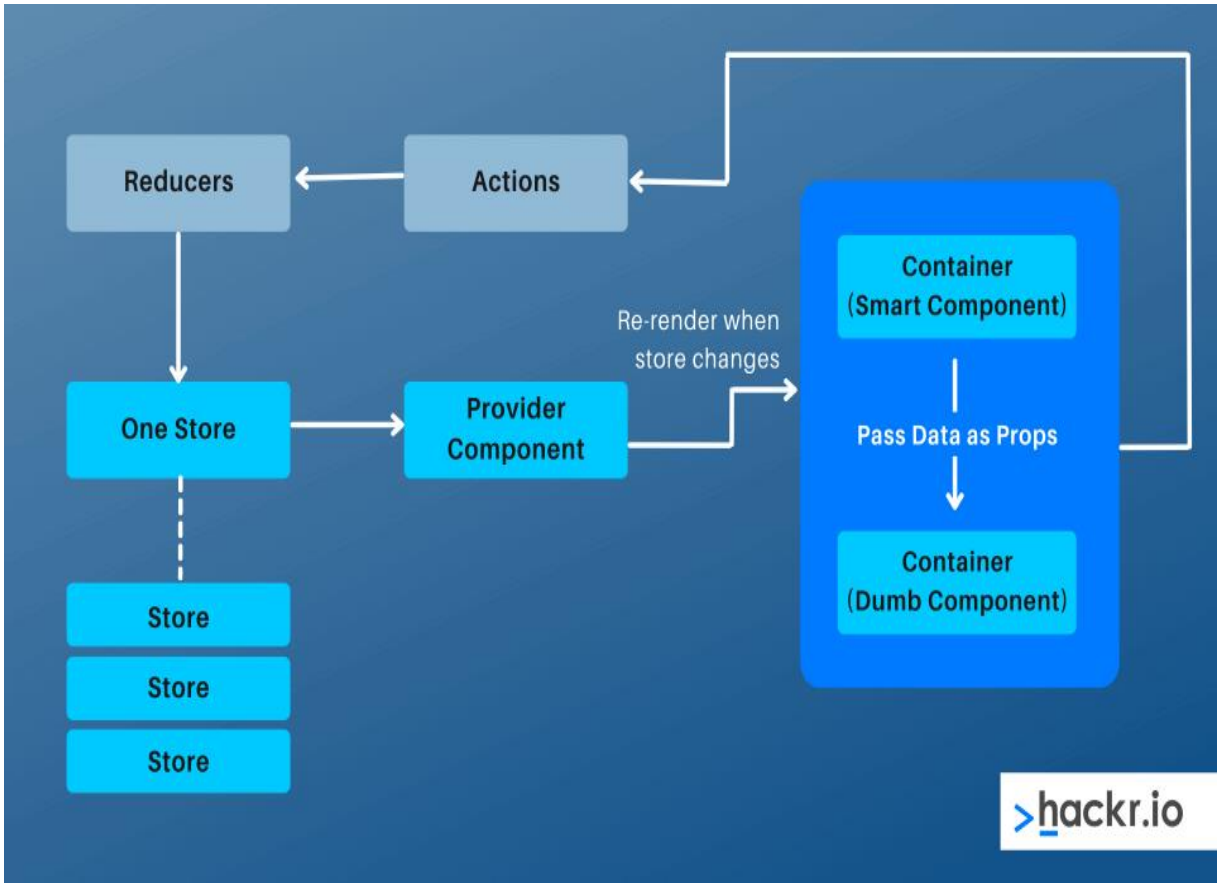
```
constructor() {

super();

this.state={

foo: 'bar'

}

}
```

3. Why do we use render() in React?

In React, each component must have a render() function. It returns a single React element, which is in fact the representation of the native DOM component.

When there is a need for more than one HTML element to be rendered, we group them in one enclosing tag. There is a need for the render() function to return the same result each time it is invoked, i.e. it needs to be kept pure.

4. Draw a diagram showing how data flows through Redux.



5. What are the major differences between the syntax in ES5 and ES6?

The syntax has witnessed a great change from ES5 to ES6. The important differences between the two releases of ECMAScript are:

- **Require vs. Import:** The require used in ES5 is now replaced with import.var React = require('react'); //is now replaced with
  import React from 'react'; //in ES6
- **Export vs. Exports:** Instead of exports, now export is used.export default Component; // replaces
  module.exports = Component; // in ES6
- **Component and Function:** The use of components and function has also changed from ES5 to ES6.

In ES5:

```
var MyComponent = React.createClass({



render: function() {
```

```
return

Hello World!

;

}

});
```

In ES6:

```
classMyComponentextendsReact.Component {

render() {

return

Hello World!

;

}

}
```

- **Props:** Rules for using props has also changed from ES5 to ES6

In ES5:

```
var App = React.createClass({

propTypes: { name: React.PropTypes.string },

render: function() {

return

Hello, !

;

}

});
```

In ES6:

```
classAppextendsReact.Component {

render() {

return

Hello, !

;
```

```
}

}
```

- **State:** Using state has also been tweaked for ES6.

In ES5:

```
var App = React.createClass({

getInitialState: function() {

return { name: 'world' };

},

render: function() {

return

Hello, !

;

}

});
```

In ES6:

```
classAppextendsReact.Component {


constructor() {


super();


this.state = { name: 'world' };


}


render() {


return


Hello, !


;


}


}
```

6. What is the Virtual DOM and how does it work?

A virtual DOM is a lightweight JS object. It is simply a copy of the real DOM. A virtual DOM is a node tree that lists various elements, their attributes, and content as objects and their properties.

The render() function in React is responsible for creating a node tree from the React components. This tree is then updated in response to the mutations resulting in the data model due to various actions made by the user or the system.

The virtual DOM operates in three simple steps:

- **Step 1:** The entire UI is re-rendered in Virtual DOM representation as soon as there are some underlying data changes.
- **Step 2:** The difference between the previous DOM representation and the new one (as a result of the underlying data changes) is calculated.
- **Step 3:** After the calculations are successfully carried out, the real DOM is updated in line with only the things that were changed.

7. How does the Real DOM differ from the Virtual DOM?

- **DOM Manipulation:** The real DOM is expensive when it comes to DOM manipulation. The virtual DOM, on the contrary, is inexpensive.
- **Element Update:** The real DOM creates a new DOM when an element updates. Virtual DOM does not and instead updates the JSX.
- **Memory Wastage:** The real DOM causes a lot of memory wastage while there is no memory wastage in the case of the virtual DOM.
- **Update Speed:** The real DOM updates slowly but the virtual DOM updates faster.
- **Updating HTML:** The real DOM can directly update HTML, while the virtual DOM can’t update HTML directly.

8. What are the lifecycle methods of React components?

- **componentDidMount():** Executes on the client-side after the first render.
- **componentDidUpdate():** Called immediately after rendering takes place in the DOM.
- **componentWillMount():** Executes immediately before rendering starts on both the client-side and the server-side.
- **componentWillReceiveProps():** Invoked when props are received from the parent class and before another render is called.
- **componentWillUnmount():** Used to clear up the memory space. Called right after the component is unmounted from the DOM.
- **componentWillUpdate():** Called immediately before rendering takes place in the DOM.
- **shouldComponentUpdate():** Returns either true or false. Though false by default, needs to be set to return true if the component needs to be updated.

9. Explain JSX with a code example. Why can’t browsers read it?

JSX is a contraction of JavaScript XML. It uses the expressiveness of JavaScript for making the HTML code easily understandable. JSX files make applications robust while boosting their performance. An example of JSX is:

```
render(){
```

```
return(

React learning made better by Hackr.io!!

);

}
```

JSX isn't a regular JS object. The inability of browsers to read JSX is due to the fact that browsers can only read regular JS objects.

In order to enable a web browser for reading the JSX file, it needs to be transformed into a regular JavaScript object. For this, JSX transformers, like Babel, are used.

10. Write code to demonstrate embedding two or more components into one.

```
classMyComponentextendsReact.Component{

render(){

return(

Hello

);

}
```

```
}


classHeaderextendsReact.Component{


render(){


return


Header Component


};


}


ReactDOM.render(


, document.getElementById('content')


);
```

11. With an example, explain how to modularize code in React.

In order to modularize code in React, export and import properties are used. They assist in writing the components distinctly in different files:

```
exportdefaultclassChildComponentextendsReact.Component {


render() {
```

```
return(

This is a child component

);

}

}

import ChildComponent from'./childcomponent.js';

classParentComponentextendsReact.Component {

render() {

return(

);

}

}
```

12. How does the React Router differ from conventional routing?

- **Changes in the URL:** A HTTP request is sent to a server for receiving a corresponding HTML page in conventional routing. React routing does so only for a change in the history attribute.
- **Navigation:** In conventional routing, the user actually navigates across different web pages for each individual view. In React routing, the users feel like they are navigating across distinct web pages while in actuality they aren't.
- **Pages:** In React routing only a single HTML page is involved, but in conventional routing, each view corresponds to a new file.

13. How does the state differ from props?

- Changes inside child components are possible with props but not with state.
- Changes inside the component aren't possible with props but they are with state.
- Props allow for a parent component to change the value, state doesn't.

14. How do you distinguish Redux from Flux?

- **Components:** React components subscribe to the store in flux whereas in redux, container components utilize connect
- **Dispatcher:** There is no dispatcher in redux. On the other hand, flux has a singleton dispatcher
- **Number of Stores:** While flux has several stores, there is only a single store for redux
- **State:** It is mutable for flux but immutable for redux
- **Store:** Influx, the store contains state as well as change logic. Contrary to this, the store in redux is separate from the change logic
- **Store Type:** All stores in flux are disconnected and flat. This is not the case with redux, where there is a single store with hierarchical reducers

15. How would you create a form in React?

React forms are identical to HTML forms. However, the state is contained in the state property of the component in React and is updateable only via the setState() method.

Therefore, the elements in a React form can't directly update their state. Their submission is handled by a JS function, which has full access to the data entered into the form by a user.

The following code demonstrates creating a form in React:

```
handleSubmit(event) {



alert('A name was submitted: ' + this.state.value);
```

```
event.preventDefault();


}


render() {


return (


Name:


);


}
```

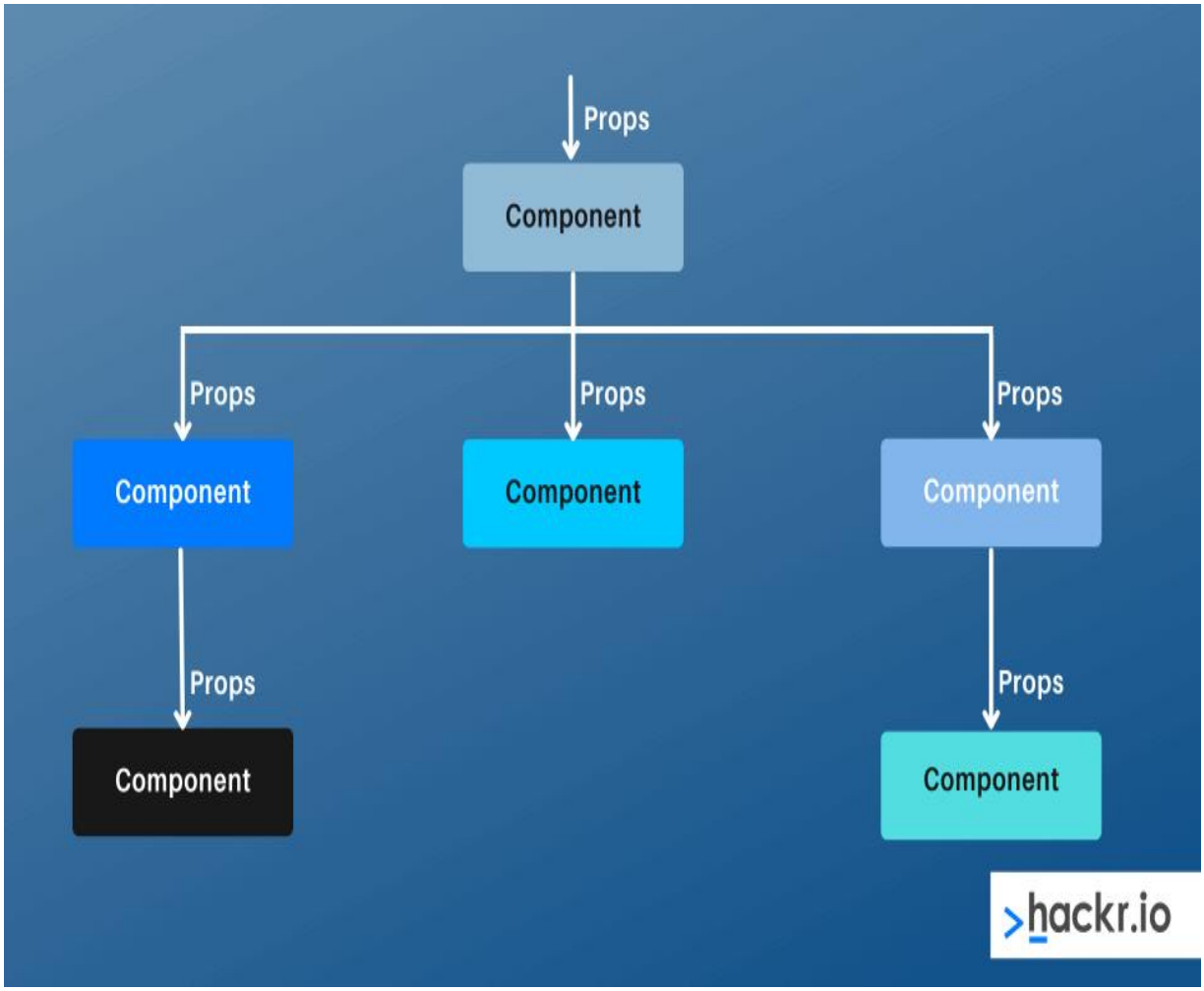
16. What are the advantages of using Redux?

- **Better Code Organization**: Redux is precise in terms of how the code needs to be organized. This results in a consistent code workable for any development team
- **Developer Tools**: The tools allow developers to track everything, ranging from actions to state changes, in real-time.
- **Easy Testing**: Redux code is mainly composed of functions that are isolated, pure, and small. This makes testing easier.
- **Large-scale Community**: Redux is backed by a mammoth community. It contributes to an ever-growing and refined library with ready-to-use applications.
- **Maintainability**: Thanks to a predictable outcome and strict structure, the code is easier to maintain.
- **Output Predictability**: There is no confusion about syncing the current state with actions as well as other parts of the application as there is only a single source of truth, which is the store.
- **Server-side Rendering**: You only need to pass the store created on the server-side to the client-side. In addition to this being useful for the initial

render, it also offers a better user experience because it optimizes application performance.

17. What is a prop?

Prop is short for Properties in React. These read-only components need to be kept immutable, i.e. pure. Throughout the application, props are passed down from the parent components to the child components.



In order to maintain the unidirectional data flow, a child component is restricted from sending a prop back to its parent component. This also helps in rendering the dynamically generated data.

18. Where would you put AJAX calls in your React code?

It is possible to use any AJAX library with React, such as Axios, jQuery AJAX, as well as the in-built browser window.fetch.

Data with AJAX calls need to be added to the componentDidMount() lifecycle method. By doing this, it is possible to use the setState() method for updating components as soon as the data is retrieved.

19. Write sample code to update the state of a component in React.

The state of a component in React is updated with this.setState() as demonstrated in the following code example:

```
classMyComponentextendsReact.Component {


constructor() {


super();


this.state = {


name: 'Akhil',


id: '101'


}


}


render()


{


setTimeout(()=>)},2000)


return (
```

```
Hello


Your Id is


);


}


}


ReactDOM.render(


, document.getElementById('content')


);
```

20. What is your take on the statement “In React, everything is a component”?

The building blocks of a React application’s UI are called components. Any app UI created using React is divisible into a number of small independent and reusable pieces, known as components.

React renders each of the components independent of each other. Hence, there is no effect of rendering a component on the rest of the app UI.

21. What are the most distinct features of React?

The most distinct features of React are:

- It uses the virtual DOM in place of the real DOM.
- Server-side rendering.
- Unidirectional flow of data.

- Data binding.

22. What are the advantages of React?

React has several advantages, including:

- Improved application performance.
- Can be used for client-side and server-side rendering.
- Increases the readability of JSX code.
- Easy to integrate with other frameworks, including Angular, Meteor, etc.
- The UI can be made more intuitive with React.

23. Are there any limitations to React?

There are several limitations to React, including:

- It acts as a library and not as a framework.
- The contents of the library are so large that it consumes a considerable amount of time to understand.
- It is difficult to understand for the novice.
- The coding process becomes more complicated when inline templating and JSX are applied.

24. Can browsers read JSX?

No, browsers cannot read JSX because it is not a regular JavaScript object.

25. What are the differences between React and Angular?

There are several differences between React and Angular, such as:

Feature	React	Angular
Architecture	Only supports the view of MVC.	Supports a complete MVC view.
Render	Both client and server-side rendering.	Client-side rendering.
DOM	Uses the virtual DOM.	Uses the real DOM.
Data binding	One-way data binding.	Two-way data binding.
Debug	Compile-time debugging.	Runtime debugging.
Developer	Developed by Facebook.	Developed by Google.

26. What is the relation between React and its components?

React and its components are closely related. The components of React act as the building blocks of a React application for the UI. The splitting up of this entire UI into small, independent, and reusable pieces helps in creating the React application's UI.

27. What are states in React?

States in React act as a source of data and are kept simple so that the objects which determine component rendering and behavior become mutable.

28. Can the parent component change value in States and Props?

The parent component can change the value in props but not in the state.

29. Can changes be made inside the component?

The changes can be made inside the state but not in props.

30. Can we make changes inside child components?

Yes, we can make changes inside the child component in props but not in the case of states.

31. What is a stateful component?

A stateful component stores a change in memory. It has the authority to change state and contains vital information about past, current, and future changes.

32. How is a stateless component different from a stateful component?

A stateless component calculates the internal state of the component but does not have the authority to change the state. There is no knowledge about the past, current, or future but receives props from the stateful component, which are treated as a callback function.

33. What are synthetic events in React?

Synthetic events in React are objects in React which act as a cross-browser wrapper around the browser's native event. The main purpose is to combine the different browsers on the API so that the event shows various properties.

34. What are refs in React?

Refs stands for References to React. It helps store a reference to a particular react element or component that can be returned by the component render configuration function.

35. When are refs mostly used?

Refs are mostly used in the following cases:

- When there is a need to manage focus, select the text, or apply media playback.
- To initiate imperative animations.
- To join third-party DOM libraries.

36. Can we modularize code in React? How?

Yes, we can modularize code in React. It can be done by using export and import properties.

37. What are the controlled components in React?

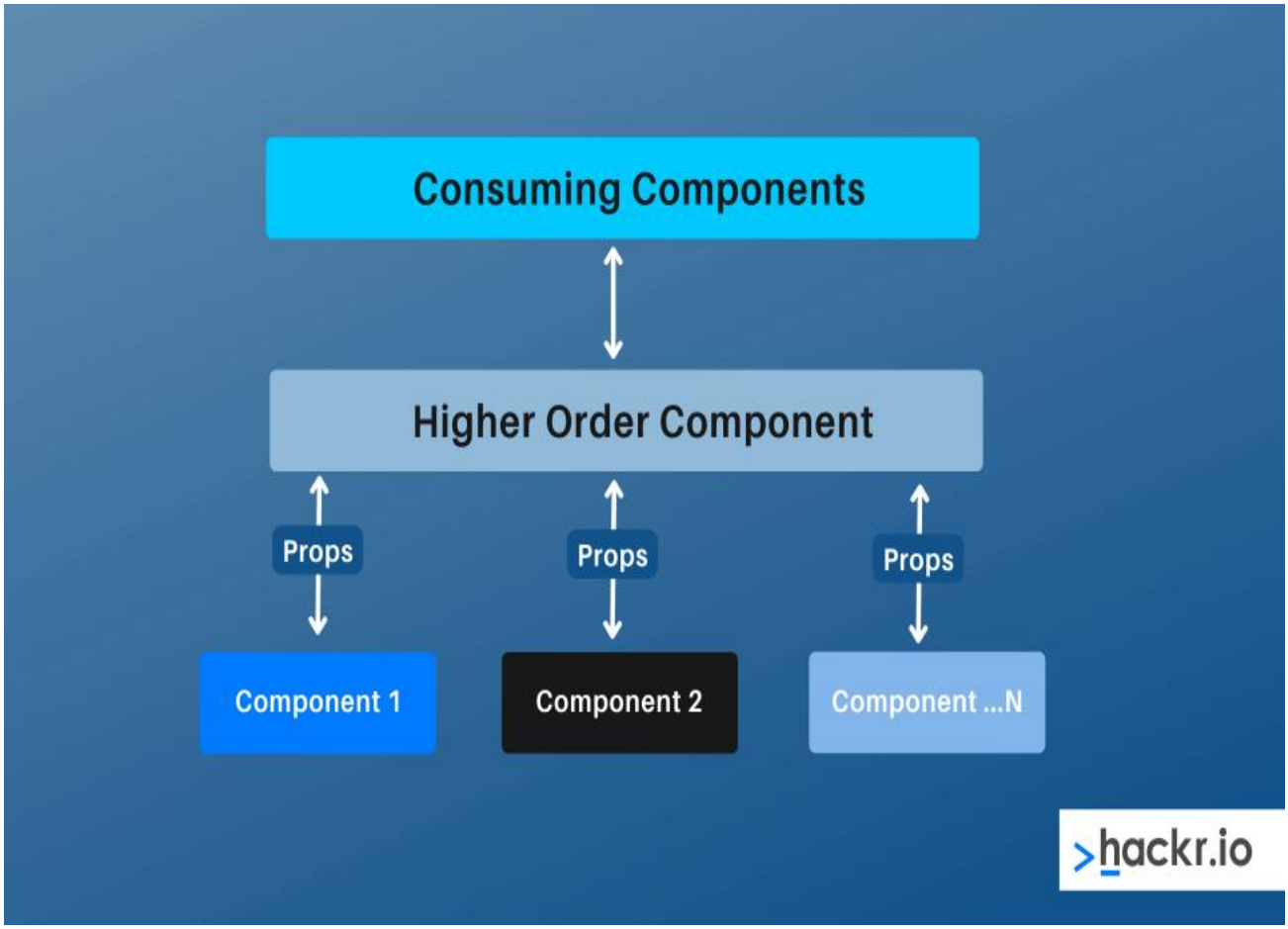
The controlled components in React are those components that can maintain their state. The data is controlled by their parent component, and they take into consideration the current values using props and thereafter, notify the changes using callbacks.

38. How are uncontrolled components different from controlled components?

The uncontrolled components maintain their state, and their data is controlled by DOM. The Refs are used in uncontrolled components to get their current values instead of using props in the case of controlled components.

39. What is HOC?

HOC stands for Higher-Order Component. It is an advanced way of reusing the component logic, which wraps another component along with it.



40. What are the benefits of HOC?

There are several benefits of HOC, including:

- Reusable code.
- Application of logic and bootstrap abstraction.
- Offers a high hacking facility.
- Supports state abstraction and manipulation.
- Prop manipulation.

41. What are pure components?

Pure components are components that are simple and easily written. They can easily replace any component as a render().

42. What are reducers?

Reducers are pure functions that clearly state how the application state changes when certain actions are made. This way, it takes into account the previous state and action to create a new state.

43. What is a store in Redux?

A store is a JavaScript object that can hold application states, access them and apply dispatch actions along with register listeners.

44. Why do we need a router?

We need a router in React so that we could define multiple routes whenever the user types a particular URL. This way, the application of a particular router can be made when the URL matches the path defined inside the router.

Frequently Asked Questions

1. What Can I Expect in a React Interview?

You can expect to be asked both theoretical and React coding questions in an interview. Make you have your basics down, and go over the questions here to get a sense of what is commonly asked.

2. What Should I Study for a React Interview?

You should study everything from simple concepts to code implementations. Focus on such things as components, states, Redux, props - basically what you see in the React JS interview questions list here.

3. How Do You Prepare for a React Interview?

Prepare for a React interview by making sure you have your fundamentals down and by practicing. Watching video tutorials and reading books are also helpful. Remember, React coding interview questions will also be there, so don’t skimp on those