Share:

If you're looking for React JS Interview Questions and Answers? If YES, then this blog is for you!

There are a lot of opportunities from many reputed companies in the world. According to research, React JS has a **market share of about 7.3%**.

> *#Interested to learn more about React framework - Then check out our **Best React Course** to gain expertise to build React JS Applications.*

Mindmajix designed this blog with the latest 2020 updated **ReactJS Interview Questions** and Answers for freshers and experienced professionals. These React interview questions will help you to crack the React interview easily. Let's get into them.

Top React JS Interview Questions (Updated - 2020)

Here top ReactJS interview questions:

- ***What is React?***
- ***Why React is used?***
- ***How React works?***

- ***What are the features of ReactJS?***
- ***What are the Advantages of React JS?***
- ***How is React different from AngularJS?***
- ***How ReactJS framework is different as compared to others?***
- ***What are the life Cycles of ReactJS?***

ReactJS Interview Questions and Answers -For Beginners

Q1: What is React?

Ans. React is a front-end JavaScript library that mainly follows the component-based approach for building a user interface (UI) components for a single page application. It is also used for handling the view layer in both mobile and web apps. Moreover, react plays a crucial role in developing interactive mobile and web UIs. It was created and developed by Jordan Walke; it was deployed first on the Facebook newsfeed in 2011.

Q2: Why React is used?

Ans: The following reasons make one to use React for building User Interfaces (UI), and they are:

- Easy to learn nature
- Simplicity
- High scalability
- Increase performance

Also Read: **Full Stack Developer Interview Questions**

Q3: How React works?

Ans. Below is the sequence of steps which gives an idea about how does react work

- Firstly the react runs the diffing algorithm to identify the changes that are made in the virtual DOM.
- Next step is reconciliation, this is used to update the DOM as per the new features.
- Now, the virtual DOM, which is lightweight in nature and is detached from the specific implementation of the browser.

- Followingly the ReactElements which are present in virtual DOM are used to build basic nodes.
- Finally, if the ReactComponent changes the state; the diffing algorithm runs faster and identify the changes. After identification, it automatically updates the DOM with the change difference.

Q4: What are the features of ReactJS?

Ans: The features of React JS are as follows:

1. React improves SEO performance

React boost the performance of the SEO to higher levels as a search engine faces the problem while reading JavaScript of high loaded applications.

2. React acts as a standard for mobile app development

It provides a transition process as an ideal solution for both mobile and web applications for building rich user interfaces.

3. React makes the process of writing components easier

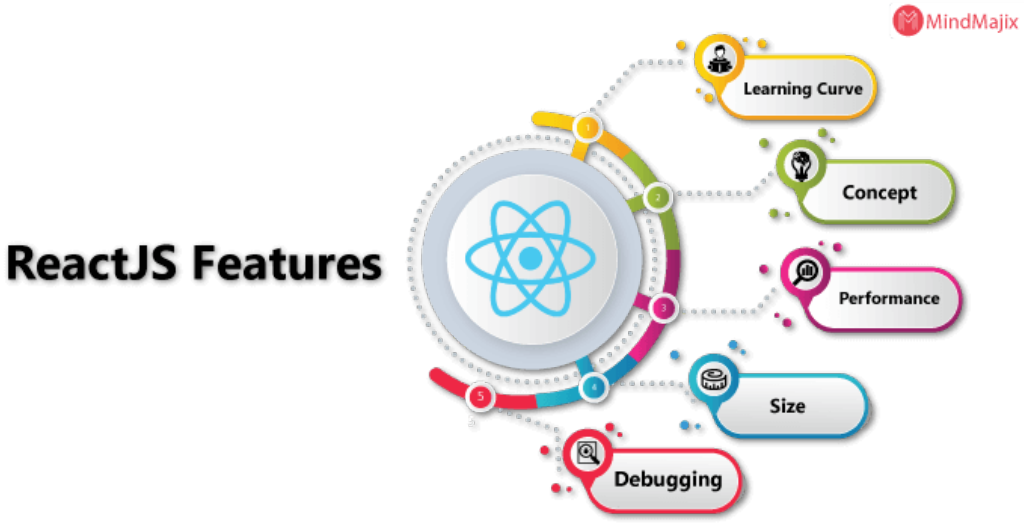
Using React along with JSX will make you write components and code efficiently and clearly.

4. React increases efficiency

As the React boost the efficiency of components by reusing them. This is the reason why it is considered as an ideal feature of React. It is considered as the most reusable system component.

5. React ensures stable code

It ensures the stability of the code of an application by making use of downward dataflow.



Q5: What are the Advantages of React JS?

Ans: The advantages of React are as follows

- Usage of JSX makes easier to read and write code
- Improves the performance of applications with the use of virtual DOM
- Provides an easier way to integrate with frameworks
- It can be shared and rendered both on server and client-side
- Writing integration and unit tests can be made smother by using tools

Q6. How is React different from AngularJS?

Ans: The following table shows the major difference between AngularJS and React - Learn more at React JS vs Angular JS

Factor	React JS	AngularJS
Usage of DOM	Uses virtual DOM	Uses real DOM
Language	Uses JavaScript with the extended XML syntax	Uses TypeScript which is the superset of JavaScript
App Structure	It is represented only using the view of MVC	Made of Complete MVC
Data Binding	One-way binding	Two-way binding

Q7: How ReactJS framework is different as compared to others?

Ans. Basically, ReactJS is a limited library that builds UI parts, it is essentially not quite the same as a considerable measure of other JavaScript structures. One common example is AngularJS approaches building an app simply by expanding

HTML markup and infusing different develop such as controller at runtime. Therefore, AngularJS is exceptionally obstinate about the more noteworthy engineering of your application. Learn more differences at React JS vs Angular JS

Also Read: React JS Tutorial for Beginners

Q8: Does ReactJS use HTML?

Ans. No, It uses JSX which is similar to HTML.

Q9: What is JSX?

Ans: It is basically a novel dialect of the popular JavaScript that simply integrates the HTML templates into the code of JavaScript. The browser is not capable to read the code simply and thus there is a need for this integration. Generally, WebPack or Babel tools are considered for this task. It has become a very popular approach in the present scenario among the developers.

Q10: What are the life Cycles of ReactJS?

Ans.

1. Initialization
2. State/Property Updates
3. Destruction

Q11: When ReactJS released?

Ans. March 2013

Q12: How is ReactJs different from AngularJS?

The first difference between both of them is their code dependency. ReactJS depends less on the code whereas AngularJS needs a lot of coding to be done. The packaging on React is quite strong as compared to the AngularJS. Another difference is React is equipped with Virtual Dom while the Angular has a Regular DOM. ReactJS is all about the components whereas AngularJS focus mainly on the Models, View as well as on Controllers. AngularJS was developed by Google while the ReactJS is the outcome of facebook. These are some of the common differences between the two.

Q13: What is Redux?

Ans: It is one of the most in-demand libraries for front-end development in today's growing world. It is defined as the predictable state container mainly designed for JavaScript apps and also it is used for managing the entire state of an application. Redux is very small in size and has no dependencies. It builds applications that are

easy to deploy in different environments and easy to test. Redux is very small in size and has no dependencies.

Q14: What is Use of Redux thunk?

Ans: Redux thunk acts as middleware which allows an individual to write action creators that return functions instead of actions. This is also used as a delay function in order to delay dispatch of an action if a certain condition is met. The two store methods *getState()* and *dispatch()* are provided as parameters to the inner function.

In order to activate Redux thunk, we must first use applyMiddleware() method as shown below:

```
import{ createStore, applyMiddleware } from 'redux';
import thunk from 'redux-thunk';
import rootReducer from './reducers/index';

//Note: this API requires redux@>=3.1.0

const store= createStore(
     rootReducer,
     applyMiddleware(thunk)
);
```

Q15: What do you know about Flux?

Ans. Basically, Flux is a basic illustration that is helpful in maintaining the unidirectional data stream. It is meant to control construed data unique fragments to make them interface with that data without creating issues. Flux configuration is insipid; it's not specific to React applications, nor is it required to collect a React application. Flux is basically a straightforward idea, however in you have to exhibit a profound comprehension of its usage.

Q16: What is the current stable version of ReactJS?

Ans. ******Version: 16.12.0

******Release on: Nov 14, 2019

Q17: What is the Repository URL of ReactJS?

Ans. https://github.com/facebook/react

Q18: What do you know about the component lifecycle in ReactJS?

Ans. Component lifecycle is an essential part of this platform. Basically, they have lifecycle events that fall in the three prime categories which are property updates, Initialization and third are Destruction. They are generally considered as a method of simply managing the state and properties of every reach component.

Q19: Do you think ReactJS has any limitations? If so, tell a few?

Ans. Yes, there are a few drawbacks which are associated with this platform. The leading drawback of the ReactJS is the size of its library. It is very complex and creates a lot of confusion among the developers. Also, there are lots of developers all over the world which really don't like the JSX and inline templating. In addition to this, there is another major limitation of ReactJS and i.e. only cover one layer of the app and i.e.View. Thus to manage the development, developers have to depend on several other technologies which consume time.

Q20: How the parent and child components exchange information?

Ans. This task is generally performed with the help of functions. Actually, there are several functions which are provided to both parent and child components. They simply make use of them through props. Their communication should be accurate and reliable. The need of same can be there anytime and therefore functions are considered for this task. They always make sure that information can be exchanged easily and in an efficient manner among the parent and child components.

Q21: What is a State in React and How is it used?

Ans. In React, State is an object that represents how the component renders and behaves. States are the sources of data and allow you to create dynamic and interactive components. They are accessed using *this.state()*. For changing a value in the state object, call it using *this.setState()* method.

Q22: What are the differences between the Class component and Functional component?

Ans.

Parameter	Class Component	Functional Component

Syntax	This component requests you to extend from React. Component to create render function that in turn returns a react element	It is just a plain JavaScript function that accepts props as their arguments and returns the react element.
Life cycle hooks	Lifecycle hooks are created from the React Component. This class component makes lifecycle hooks available in it.	We cannot use lifecycle hooks in a functional component.
Readability	They are very difficult to test and read	They are much easier to test and read

Class Component Coding

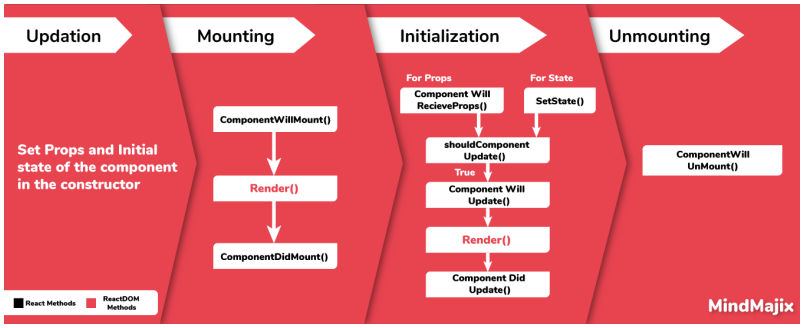
```
class App extends Component {
    render () {
        return (
            <Text>Hello World!</Text>
        )
    }
}
```

Functional Component Coding

```
const PageOne = () => {
    return (
        <h1>Page One</h1>
    );
}
```

Q23. What is props in React?

Ans. Props stand for properties in React and used for passing the information from one component to another. But the data with the Props are passed in a unidirectional flow, i.e., one way from parent to child. Further, they are read-only data, which means child components cannot change data coming from the parent.



Q24. What are the significant differences between state and props?

Ans. The difference between state and props are as follows:

State	Props
The state is completely managed within a component for internal communication.	Props are directly passed to its parents with child component.
State can be modified using setState() method.	A particular component should never modify its own props.
State changes can be asynchronous	Props are read-only

Q25. What is the higher-order component?

Ans. In ReactJS high order component can be defined as the function that is mainly used to collect the component and returns a new component. These components are the patterns that are extracted from the React's compositional nature. One important aspect of this component is that it is used as a reusable component logic in the React. It provides us with the best way to share behaviour between different React components.

Q26: How to embed two components in One component?

```
import React from 'react';
class App extends React.Component{
    render(){
        return(
            <div>
            <Header/>
            <Content/>
            </div>
        );
    }
}
class Header extends React.Component{
    render(){
        return(
            <div>
            <h1> Header</h1>
            </div>

        )
    }
}
class Content extends React.Component{
    render(){
        return(
            <h2>Content</h2>
            <p>The Content Text!!!</p>
            </div>
        )
    }
}
export default App;
```

Q27. What are Synthetic events in React?

Ans. React implements Synthetic events to improve the consistency and performance of applications and interfaces. The synthetic event is a cross-browser wrapper around the browser's native event. It combines the behaviour of multiple browsers into a single API to make sure events have the same properties across different browsers and platforms.

ReactJS Interview Questions for Experienced

Q28: Give one basic difference between pros and state?

Pros are immutable while the state is mutable. Both of them can update themselves easily.

Q29: How do you tell React to build in Production mode and what will that do?

Ordinarily, you'd utilize Webpack's **DefinePlugin** strategy to set **NODE_ENV** to production. This will strip out things like prototype approval and additional notices. Over that, it's likewise a smart thought to minify your code in light of the fact that React utilizes Uglify's dead-code end to strip out advancement just code and remarks, which will radically diminish the measure of your package.

Q30: What do you understand with the term polling?

The server needs to be monitored to for updates with respect to time. The primary aim in most of the cases is to check whether novel comments are there or not. This process is basically considered as pooling. It checks for updates approximately every 5 seconds. It is possible to change this time period easily. Pooling help keeping an eye on the users and always make sure that no negative information is present on the servers. Actually, it can create issues related to several things and thus pooling is considered.

Q31: When would you use a Class Component over a Functional Component?

If your component has state or a lifecycle method(s), use a Class component. or else, use a Functional component.

Q32: What do you mean by virtual DOM?

For all the available DOM objects in ReactJS, there is a parallel virtual DOM object. It is nothing but can be considered as the lighter version of the true copy and is powerful

in eliminating the complex code. It is also used as a Blue Print for performing several basic experiments. Many developers also use it while practicing this technology.

Q33: Compare MVC with Flux?

MVC approaches are presently considered as outdated. Although they are capable to handle data concerns, controllers as well as UI, many developers found that it doesn't properly work when applications size increases. However, they are capable to handle some of the key issues such as eliminating the lack of data integrity as well as managing the data flow which is not properly defined. On the other side, Flux works perfectly with all the sizes irrespective of their size.

Q34: What's the difference between an Element and a Component in React?

Basically, a React component describes what you need to see on the screen. Not all that basically, a React element is a protest portrayal of some UI.

A React component is a function or a class which alternatively acknowledges input and returns a React component (ordinarily by means of **JSX** which gets transpiled to a createElement invocation).

Q32: Tell us three reasons behind the success of ReactJS?

ReactJS is a technology that can be trusted for complex tasks. While performing any task through it, developers need not worry about the bugs. It always ensures error-free outcomes and the best part is it offers scalable apps. It is a very fast technology and can simply be trusted for quality outcomes.

Q33: In which lifecycle event do you make AJAX requests and why?

AJAX solicitations ought to go in the **componentDidMount** lifecycle event.
There are a couple of reasons behind this,

Fiber, the following usage of React's reconciliation algorithm, will be able to begin and quit rendering as required for execution benefits. One of the exchange offs of this is **componentWillMount**, the other lifecycle event where it may bode well to influence

an AJAX to ask for, will be "non-deterministic". This means React may begin calling **componentWillMount** at different circumstances at whenever point it senses that it needs to. This would clearly be a bad formula for AJAX requests.

You can't ensure the AJAX request won't resolve before the component mounts. In the event that it did, that would imply that you'd be attempting to setState on an unmounted component, which won't work, as well as React will holler at you for. Doing AJAX in componentDidMount will ensure that there's a component to update.

Q34: What is the difference between createElement and cloneElement?

createElement is the thing that JSX gets transpiled to and is the thing that React uses to make React Elements (protest representations of some UI). cloneElement is utilized as a part of request to clone a component and pass it new props. They nailed the naming on these two.

Q35: What is meant by event handling?

To capture the user's information and other similar data, the event handling system is considered. It is generally done through DOM elements which are present in the code. This task is simple to accomplish. Two-way communication is considered in this approach.

Q36: What is the second argument that can optionally be passed to setState and what is its purpose?

A callback work which will be conjured when **setState** has completed and the part is re-rendered.
Something that is not talked about a great deal is that **setState** is asynchronous, which is the reason it takes in a moment callback function. Ordinarily, it's best to utilize another lifecycle strategy instead of depending on this callback function, however, it's great to know it exists.

```
Class Training extends Course
{
this.state = {
sampleItem: 'learn',
}
handleChange = (event) => {
console.log(this.state.sampleItem)
this.setState({
sampleItem: event.target.value      //event.target.value = Welcome
}, () => console.log(this.state.sampleItem))
};
```

Output:

Learn

Welcome

Q37: How many outermost elements can be there in a JSX expression?

It must have one JSX element present so that the task can be accomplished easily. Having more than one expression is not an issue but probably it will slow down the process. There are also chances of confusion with more than one expression if you are new to this technology.

Q38: What are controlled and uncontrolled components?

There are components in the ReactJS that maintain their own internal state. They are basically considered as uncontrolled components. On the other side, the components which don’t maintain any internal state are considered as controlled components in ReactJS. Controlled components can easily be controlled by several methods. Most of the React components are controlled components.

Q39: Mention the key benefits of Flux?

Applications that are built on Flux have components that can simply be tested. By simply updating the store, developers are able to manage and test any react component. It cut down the overall risk of data affection. All the applications are highly scalable and suffer no compatibility issues.

Q40: What's wrong with the following code?

```
this.setState{(prevState, props)=>
{
return {
streak: prevState.streak+props.count
}
})
```

Nothing isn't right with it. It's once in a while utilized and not outstanding, but rather you can likewise pass a function to setState that gets the past state and props and returns another state, similarly as we're doing above. Furthermore, is nothing amiss with it, as well as effectively recommended in case you're setting state in light of the previous state.

Q41: Why browsers cannot read JSX?

Actually, JSX is not considered as a proper JavaScript. Browsers cannot read them simply. There is always a need to compile the files that contain JavaScript Code. This is usually done with the help of JSX compiler which performs its task prior to file

entering the browser. Also, compiling is not possible in every case. It depends on a lot of factors such as the source or nature of file or data.

Q42: What are pure functional Components?

Traditional React Components as we have seen so far are making a class with class Example extends React.Component or React.createClass(). These make stateful components on the off chance that we at any point set the state (i.e. this.setState(), getInitialState(), or this.state = {} inside a constructor()).

In the event that we have no expectation for a Component to require state, or to require lifecycle methods, we can really compose Components with a pure function, consequently the expression "pure function Component":

```
function Date(props)
{
let {msg="The date is:"} = props
let now = new Date()
return <div>
<span> {msg}</span>
<time> {now.toLocaleDateString()}</time>
</div>

}
```

This function that returns a React Element can be used wherever we see fit:

```
DOM.render(<div> <Date msg="Today is"/></div>)
```

You might notice that also takes a prop – we can still pass information into the Component.

Q43. What is the difference between Real DOM and virtual DOM?

Ans.

- DOM stands for Document Object Model. It allows scripts and programs to dynamically access and update the content, structure, and style of a document. DOM is an abstraction of a structured code called HTML, also described as HTML DOM.
- Virtual DOM
- is a lightweight Javascript object, which is the copy of the representation of a DOM object. It is an abstraction of HTML DOM. Virtual is quite faster compared to DOM, performs its tasks reliably.

Virtual DOM	Real DOM
Updates faster	Updates slower
No memory wastage	Excess memory wastage
Can't update HTML directly	Update HTML directly
DOM manipulation is easy	DOM manipulation costly

Q44: What happens during the lifecycle of a React component?

A standout amongst the most valuable parts of React is its segment lifecycle — so seeing precisely how segments components after some time is instrumental in building a viable application.

Q45: What exactly you can do if the expression contains more than one line?

In such a situation, enclosing the multi-line JSX expression is an option. If you are a first time user, it may seem awkward but later you can understand everything very easily. Many times it becomes necessary to avoid multi-lines to perform the task reliably and for getting the results as expected.

Q46: Is it possible to use the word “Class” in JSX. Why or why not?

No, it is not possible in the JSX. This is because the word “Class” is a reticent (occupied) word in the JavaScript. However, you can use you are free to use the word “ClassName”. If you use the word “Class” the JSX will be translated to JavaScript immediately.

a) High-Level Component Lifecycle:

At the highest level, React components have lifecycle events that fall into 3 general classifications:

1. Initialization
2. State/Property Updates
3. Destruction

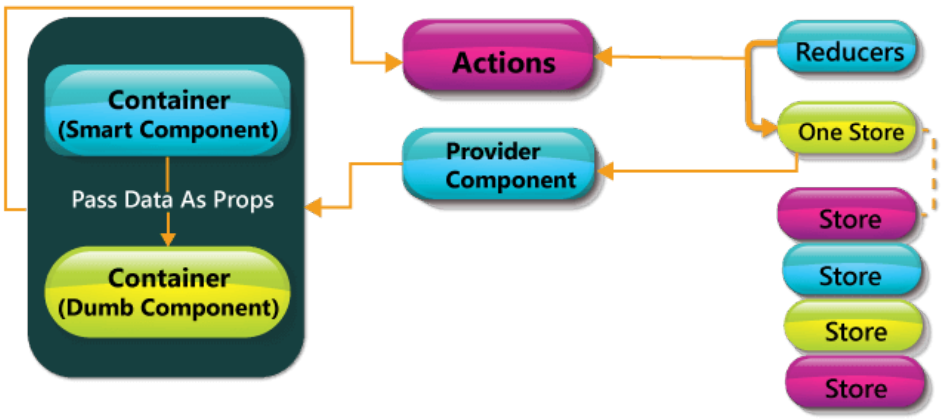
Each React component defines these events as a system for dealing with its properties, state, and rendered output. Some of these events just happen once, others happen more as often as possible; understanding these 3 general classes should help you clearly visualize when certain logic required to be applied.

For instance, a component may need to add an event audience to the DOM when it initially mounts. In any case, it ought to likely expel those event listeners when the component unmounts from the DOM with the goal that not relevant handling that doesn't occur.

```
class MyComponent extends React.Component{
   //when the component is added to the DOM...
componentDidMount(){
window.addEventListener('resize',this.onResizeHandler);
}
//when the component is removed from the DOM...
componentWillmount(){
window.addEventListener('resize',this.onResizeHandler);
}

onResizeHandler(){
console.log('The window has been resized!');
}
}
```

b) Low-Level Component Lifecycle:



Inside these 3 general buckets exist various particular lifecycle hooks — basically unique techniques - that can be used by any React component to all the more precisely manage updates. Seeing how and when these hooks fire is vital to building stable components and will empower you to control the rendering procedure (enhancing execution).

Observe the diagram above. The events under "Initialization" just happen when a component is first initialized or added to the DOM. Thus, the events under "Devastation" just happen once (when the component is expelled from the DOM). However, the events under "Update" happen each time the properties or state of the component change.

For instance, components will naturally re-render themselves whenever their properties or state change. However, at times a component should not update - so

keeping the component from re-rendering may enhance the execution of our application.

```
class MyComponent extends React.Component{
    //only re-render if the ID has changed!
    shouldComponentUpdate(nextProps, nextState;
    }
}
```

Q47: What do you know about React Router?

Rendering the components is an important task in ReactJS. React router is used to decide which components is to be rendered and which one should not. It also performs dictation during several activities.

React Interview Questions for Experienced

Q 48: Compare Flux vs MVC

Conventional MVC designs have functioned admirably to separate the worries of data (Model), UI (View) and logic (Controller) — however many web engineers have found impediments with that approach as applications develop in measure. In particular, MVC architectures as often as possible experience 2 primary issues:

Ineffectively defined data flow: The cascading updates which happen crosswise over perspectives frequently prompt a tangled web of events which is hard to debug.

Lack of data integrity: Model data can be changed from anyplace, yielding erratic results over the UI.

With the Flux pattern complex, UIs never again experience the ill effects of cascading updates; any given React component will have the capacity to recreate its state in light of the information given by the store. The flux pattern likewise upholds data integrity by limiting direct access to the shared data.

While a technical interview, it is awesome to talk about the contrasts between the Flux and MVC configuration designs inside the setting of a particular illustration:

For instance, imagine we have a "master/detail" UI in which the client can choose a record from a rundown (master view) and alter it utilizing an auto-populated form (detail view).

With a MVC architecture, the data contained inside the Model is shared between both the master and detail views. Each of these perspectives may have its own particular Controller assigning updates between the Model and the View. Anytime the information contained inside the Model may be updated — and it's hard to know where

precisely that change happened. Did it occur in one of the Views sharing that Model, or in one of the Controllers? Since the Model's information can be transformed by any performing artist in the application, the danger of information contamination in complex UIs is more prominent than we'd like.

With a Flux architecture, the Store data is correspondingly shared between different Views. However this data can't be straightforwardly changed — the greater part of the solicitations to update the data must go through the Action > Dispatcher chain first, eliminating the risk of arbitrary data pollution. At the point when refreshes are made to the data, it's presently significantly less demanding to find the code requesting for those progressions.

Q49: What are the stateless components?

On the off chance that React components are basically state machines that produce UI markup, at that point what are stateless segments?

Stateless components (a kind of "reusable" components) are simply pure functions that render DOM construct exclusively with respect to the properties gave to them.

As you can see, this component has no requirement for any internal state — not to mention a constructor or lifecycle handlers. The yield of the component is absolutely a function of the properties gave to it.

Q50: What is one of the core types in React?

ReactNode

Q52: What is redux?

A method os handling the state (or data) of an application.

Q53: Is it possible to display props on a parent component?

Yes, it is possible. The best way to perform this task is by using the spread operator. It can also be done with listing the properties but this is a complex process.

Q54: In ReactJS, why there is a need to capitalize on the components?

It is necessary because components are not the DOM element but they are constructors. If they are not capitalized, they can cause various issues and can confuse developers with several elements. At the same time, the problem of integration of some elements and commands can be there.

Q55. What are Synthetic events in React?

Ans. React implements Synthetic events to improve the consistency and performance of applications and interfaces. The synthetic event is a cross-browser wrapper around the browser's native event. It combines the behaviour of multiple browsers into a single API to make sure events have the same properties across different browsers and platforms.

Q56: Explain DOM diffing?

When the components are rendered twice, Virtual Dom begins checking the modifications elements have got. They represent the changed element on the page simply. There are several other elements that don't go through changes. To cut down the changes to the DOM as an outcome of user activities, DOM doffing is considered. It is generally done to boost the performance of the browser. This is the reason for its ability to perform all the tasks quickly.

Q57: Is it possible to nest JSX elements into other JSX elements?

It is possible. The process is quite similar to that of nesting the HTML elements. However, there are certain things that are different in this. You must be familiar with the source and destination elements to perform this task simply.

By Pranaya Pathpi 2020-05-16 99524

Share: