**Current Version: 18.2.0**

**1.  Differentiate between Real DOM and Virtual DOM.**

|  |  |
| --- | --- |
|  | |
| **Real DOM** | **Virtual  DOM** |
| 1. It updates slow. | 1. It updates faster. |
| 2. it Can directly update HTML. | 2. it Can’t directly update HTML. |
| 3. it Creates a new DOM if element updates. | 3. it Updates the JSX if element updates. |
| 4. DOM manipulation is very expensive. | 4. DOM manipulation is very easy. |
| 5. Too much of memory wastage. | 5. No memory wastage. |

**2. Differentiate between state and props.**

|  |  |
| --- | --- |
| **STATE** | **PROPS** |
| Parent component can not change value. | Parent component can change value. |
| We can change inside component. | We can not change inside component. |
| We can not change inside child components. | We can change inside child components. |

**3. How is React different from Angular?**

|  |  |
| --- | --- |
| **REACT** | **ANGULAR** |
| It is Only the View of MVC | It is Complete MVC |
| It is One-way data binding | It is Two-way data binding |
| It is Compile time debugging | It is Runtime debugging |
| It is developed by Facebook | It is developed by Google |
| It uses virtual DOM | It uses real DOM |
| It uses Server-side rendering | It uses Client-side rendering |

**4. How is React Router different from conventional routing?**

|  |  |  |
| --- | --- | --- |
| Conventional Routing vs React Routing | | |
|  | **React Routing** | **Conventional Routing** |
|  | Only single HTML page is involved. | Each view corresponds to a new file. |
|  | Only the History attribute is changed. | A HTTP request is sent to a server and corresponding HTML page is received. |
|  | User is duped thinking he is navigating across different pages. | User actually navigates across different pages for each view. |

**5. What do you know about controlled and uncontrolled components?**

|  |  |
| --- | --- |
| Controlled vs Uncontrolled Components | |
| **Controlled Components** | **Uncontrolled Components** |
| 1. They do not maintain their own state. | They maintain their own state. |
| 2. Data is controlled by parent component. | Data is controlled by DOM. |
| 3. They take current values through props and then notify the changes via callbacks. | Refs are used to get their current values. |

**6. How is Flux different from Redux?**

|  |  |
| --- | --- |
| Flux vs Redux | |
| **Flux** | **Redux** |
|  |  |
| 1. There are multiple stores. | There is single store. |
| 2. All stores are disconnected and flat. | Single store with hierarchical reducers. |
| 3. It has singleton dispatcher. | No concept of dispatcher. |
| 4. React components subscribe to the store. | Container components utilize connect. |
| 5. State is mutable. | State is immutable. |

**7. How different is React’s ES6 syntax when compared to ES5?**

The main difference is that we can use oops concept in ES6 while we can not use oops concept in ES5.

Syntax has changed from ES5 to ES6 in the following aspects:

1. require vs import

|  |  |
| --- | --- |
| 1  2  3  4  5 | // ES5  var React = require('react');    // ES6  **import** React from 'react'; |

1. export vs exports

|  |  |
| --- | --- |
| 1  2  3  4  5 | // ES5  module.exports = Component;    // ES6  export **default** Component; |

1. component and function

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | // ES5  var MyComponent = React.createClass({      render: function() {  **return**    <h3>Hello Edureka!</h3>  ;      }  });    // ES6  **class** MyComponent **extends** React.Component {      render() {  **return**    <h3>Hello Edureka!</h3>  ;      }  } |

1. props

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | // ES5  var App = React.createClass({      propTypes: { name: React.PropTypes.string },      render: function() {  **return**    <h3>Hello, {**this**.props.name}!</h3>  ;      }  });    // ES6  **class** App **extends** React.Component {      render() {  **return**    <h3>Hello, {**this**.props.name}!</h3>  ;      }  } |

1. state

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | // ES5  var App = React.createClass({      getInitialState: function() {  **return** { name: 'world' };      },      render: function() {  **return**    <h3>Hello, {**this**.state.name}!</h3>  ;      }  });    // ES6  **class** App **extends** React.Component {      constructor() {  **super**();  **this**.state = { name: 'world' };      }      render() {  **return**    <h3>Hello, {**this**.state.name}!</h3>  ;      }  } |

**8. How can you embed two or more components into one?**

We can embed components into one in the following way:

|  |  |
| --- | --- |
| 19  20  21  22  23  24  25 | **class** MyComponent **extends** React.Component{      render(){  **return**(    <div><h1>Hello</h1>                    <Header/>              </div>            );      }  }  **class** Header **extends** React.Component{      render(){  **return**    <h1>Header Component</h1>       };  }  ReactDOM.render(      <MyComponent/>, document.getElementById('content')  ); |

**9. What is React?**

* React is a front-end JavaScript library which is developed by Facebook in 2011.
* It follows component based approach which helps to build reusable UI components.
* It is used for developing complex and interactive web and mobile UI.

**10. What are the advantages of Redux?**

Advantages of Redux are listed below:

* Redux’s code becomes easier to maintain with a predictable outcome and strict structure.
* Redux’s code is mostly functions which are small, pure and isolated. This makes the code is testable and independent.

**11. List down the components of Redux.**

Redux is composed of the following components:

1. **Action** – It’s an object that describes what happened.
2. **Reducer**–  It is a place to determine how the state will change.
3. **Store** – State / Object tree of an entire application is saved in a single Store.
4. **View** – Simply displays the data provided by the Store.

**12. What are the features/Advantage of React?**

1. It uses **virtual DOM**.
2. It uses **server-side rendering**.
3. It follows data binding.
4. It increases the application’s performance.
5. React is easy to integrate with other frameworks.

**13. What are the limitations/Drawback of React?**

i. React is a library, not framework.

ii. Its library is very large and takes time to understand.

iii. It can be little difficult to understand for novice programmers.

**14. Why browsers can’t read JSX?**

Browser can not read jsx file, It can read only JavaScript objects because JSX is not a regular JavaScript object. To enable read JSX file for browser, we need to transform JSX file into a JavaScript object by using JSX transformers like Babel, than it pass to browser.

**15. What were the major problems with MVC framework?**

Following are some of the major problems with MVC framework:

* DOM manipulation is very expensive.
* Applications were slow and inefficient.
* Too much of memory wastage.

**16. Why** **switch keyword is used in React Router v4?**

The ‘switch’ keyword is used when we want to display only a single route to be rendered amongst the several defined routes.

**17. What is Props(Parmeter)?**

Props is a shorthand for Properties in React. These are read-only components which must be kept pure. They are always passed down from parent to child components throughout the application. child component can never send a prop back to the parent component.

**18. What do you understand by refs in React?**

Refs is a short hand for References in React. It is used to return references to a particular element or component returned by render() method. They come in handy when we need DOM measurements.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | **class** ReferenceDemo **extends** React.Component{       display() {  **const** name = **this**.inputDemo.value;           document.getElementById('disp').innerHTML = name;       }  render() {  **return**(    <div>              Name: <input type="text" ref={input => **this**.inputDemo = input} />              <button name="Click" onClick={**this**.display}>Click</button>    <h2>Hello <span id="disp"></span> !!!</h2>          </div>      );     }   } |

**19. What is JSX?**

JSX is a shorthand for JavaScript XML in react. This is a type of file used by React which utilizes the expressiveness of JavaScript along with HTML like template syntax. This makes HTML file really easy to understand. This file makes application robust and boosts its performance.

Below is an example of JSX:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | render(){  **return**(    <div><h1> Hello World from Edureka!!</h1></div>        );  } |

**20. What are synthetic events in React?**

Synthetic events are the objects which act as a cross-browser wrapper, around the browser’s native event. They combine the behavior of different browsers into one API.

**21. What is a state in React and how is it used?**

states are the objects which determines component rendering and behavior. They create dynamic and interactive components. They are accessed via this.state().

**22. What do you understand by Virtual DOM? Explain its works.**

virtual DOM is a JavaScript object which is originally copy of real DOM. It is a node tree that lists elements, their attributes and content as Objects. render method creates node tree out of the React components, then updates this tree in response to the mutations.

This Virtual DOM works in three simple steps**.[ignore]**

1. Whenever any underlying data changes, the entire UI is re-rendered in Virtual DOM representation.
2. Then the difference between the previous DOM representation and the new one is calculated.
3. Once the calculations are done, the real DOM will be updated with only the things that have actually changed.

**23. What is the purpose of render() method in React.**

Each React component must have a render() method. It returns single React element which is the representation of native DOM component. If more than one HTML element needs to be rendered, then they must be grouped together inside one enclosed tag such as <div>, <form>, <group> etc.

**24. What is the Higher Order Component(HOC)?**

Higher Order Component is an advanced way of reusing the component. Basically, it is a pattern that is derived from React’s compositional nature. Higher Order Components are custom components which is wrap around another component.

**25. What are Pure Components?**

Pure components are the simplest and fastest components which can be written. They can replace any component which only has a render() method.These components enhance the simplicity of the code and performance of application.

**26. Explain the role of Reducer.**

Reducers are pure functions which specify how the application’s state changes in response to an ACTION. Reducers work by taking in the previous state and action, then it returns a new state. It determines what sort of update needs to be done based on the type of the action, then returns new values. It returns the previous state as it is, if no work needs to be done.

**27. What is the significance of Store in Redux?**

A store is a JavaScript object which can hold the application’s state and provides helper methods to access the state, dispatch actions and register listeners. State/ Object tree of an entire application is saved in a single Store. We can pass middleware to the store to handle the processing of data, as well as to keep a log of various actions that can change the state of stores. All actions return a new state via reducers.

**28. What is React Router?**

React Router is a powerful routing library built on top of React, which helps to add new screens and flows to the application. This keeps the URL in sync with data that is being displayed on the web page. It maintains a standardized structure, behavior and is used for developing single page web applications. React Router has a simple API.

**29. What is FLUX ?**

Flux is an architectural pattern which enforces data binding. It controls derived data and enables communication between multiple components using a central Store which has authority of all data. Flux provides stability to the application and reduces run-time errors.

**30.** **What is Redux?**

Redux is one of the most trending libraries for front-end development in today’s marketplace. It is a predictable state container for JavaScript applications and used for the entire applications for state management.

**(31) What is Hooks?**

Hooks were added in React version 16.8. Hooks allow functional components to access the state and other React features. Because class components are generally no longer needed.

**(32) What is Material-UI?**

Material UI is an open-source React component library that implements Google's Material Design. It provides ready-made components. Developers can use these components with Material-UI styling library. The main advantages of Material UI is that, it is interoperable with different styling systems.

**diff b/w React and react dom ?**

**what is react dom ?**

**how events are different in React ?**

**how to write dynamically content(inner html) in Reacct?**

**what is react dom server?**

**what is lazy function in React ?**

**what is switch cases in React ?**

**What is diff b/w component and element in React?**

**What is component and it’s type ?**

**What is diff b/w clone component and create component in React?**

**what are the uncontrolled component ?**

**how to optimize in React?**

**31. Differentiate between stateful and stateless components. [ignore]**

|  |  |
| --- | --- |
| Stateful vs Stateless | |
| **Stateful Component** | **Stateless Component** |
| 1. Stores info about component’s state change in memory | 1. Calculates the internal state of the components |
| 2. Have authority to change state | 2. Do not have the authority to change state |
| 3. Contains the knowledge of past, current and possible future changes in state | 3. Contains no knowledge of past, current and possible future state changes |
| 4. Stateless components notify them about the requirement of the state change, then they send down the props to them. | 4. They receive the props from the Stateful components and treat them as callback functions. |

**32. What are the different phases of React component’s lifecycle? [ignore]**

There are three different phases of React component’s lifecycle:

1. *Initial Rendering Phase:* This is the phase when the component is about to start its life journey and make its way to the DOM.
2. *Updating Phase:*Once the component gets added to the DOM, it can potentially update and re-render only when a prop or state change occurs. That happens only in this phase.
3. *Unmounting Phase:*This is the final phase of a component’s life cycle in which the component is destroyed and removed from the DOM.

**33. What is an event in React? [ignore]**

In React, events are the triggered reactions to specific actions like mouse hover, mouse click, key press, etc. Handling these events are similar to handling events in DOM elements. But there are some syntactical differences like:

1. Events are named using camel case instead of just using the lowercase.
2. Events are passed as functions instead of strings.

The event argument contains a set of properties, which are specific to an event. Each event type contains its own properties and behavior which can be accessed via its event handler only.

**34. What are the three principles that Redux follows? [ignore]**

1. ***Single source of truth:***The state of the entire application is stored in an object/ state tree within a single store. The single state tree makes it easier to keep track of changes over time and debug or inspect the application.
2. ***State is read-only:***The only way to change the state is to trigger an action. An action is a plain JS object describing the change. Just like state is the minimal representation of data, the action is the minimal representation of the change to that data.
3. ***Changes are made with pure functions:*** In order to specify how the state tree is transformed by actions, you need pure functions. Pure functions are those whose return value depends solely on the values of their arguments.

**35. What do you understand by “Single source of truth”? [ignore]**

Redux uses ‘Store’ for storing the application’s entire state at one place. So all the component’s state are stored in the Store and they receive updates from the Store itself. The single state tree makes it easier to keep track of changes over time and debug or inspect the application.

**36. What is the significance of keys in React? [ignore]**

Keys are used for identifying unique Virtual DOM Elements with their corresponding data driving the UI. They help React to optimize the rendering by recycling all the existing elements in the DOM. These keys must be a unique number or string, using which React just reorders the elements instead of re-rendering them. This leads to increase in application’s performance.

**37. List some of the cases when you should use Refs. [ignore]**

Following are the cases when refs should be used:

* When you need to manage focus, select text or media playback
* To trigger imperative animations
* Integrate with third-party DOM libraries

**38. How do you modularize code in React? [ignore]**

We can modularize code by using the export and import properties. They help in writing the components separately in different files.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | //ChildComponent.jsx  export **default** **class** ChildComponent **extends** React.Component {      render() {  **return**(    <div>    <h1>This is a child component</h1>               </div>            );      }  }    //ParentComponent.jsx  **import** ChildComponent from './childcomponent.js';  **class** ParentComponent **extends** React.Component {      render() {  **return**(    <div>                  <App />              </div>            );      }  } |

**39. How do you create an event in React? [ignore]**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | **class** Display **extends** React.Component({      show(evt) {          // code      },      render() {          // Render the div with an onClick prop (value is a function)  **return** (    <div onClick={**this**.show}>Click Me!</div>            );      }  }); |