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

|  |  |
| --- | --- |
| **Real DOM** | **Virtual DOM** |
| 1. It updates slow. | 1. It updates faster. |
| 2. Can directly update HTML. | 2. Can’t directly update HTML. |
| 3. Creates a new DOM if element updates. | 3. 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. What is React?

* React is a front-end JavaScript library developed by Facebook in 2011.
* It follows the component based approach which helps in building reusable UI components.
* It is used for developing complex and interactive web and mobile UI.
* Even though it was open-sourced only in 2015, it has one of the largest communities supporting it.

### 3. What are the features of React?

Major features of React are listed below:

1. It uses the virtual DOM instead of the real DOM.
2. It uses server-side rendering.
3. It follows uni-directional data flow or data binding.

### 4. List some of the major advantages of React.

Some of the major advantages of React are:

1. It increases the application’s performance
2. It can be conveniently used on the client as well as server side
3. Because of JSX, code’s readability increases
4. React is easy to integrate with other frameworks like Meteor, Angular, etc
5. Using React, writing UI test cases become extremely easy

### 5. What are the limitations of React?

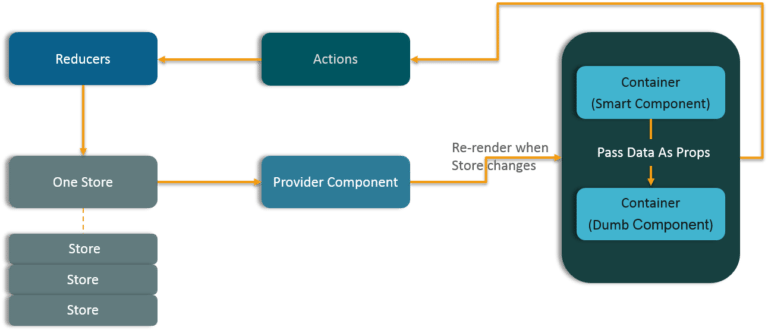
Limitations of React are listed below:

1. React is just a library, not a full-blown framework
2. Its library is very large and takes time to understand
3. It can be little difficult for the novice programmers to understand
4. Coding gets complex as it uses inline templating and JSX

### 6. What is JSX?

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | render(){  return(  <div>  <h1> Hello World from Edureka!!</h1>  </div>  );  } 7. What do you understand by Virtual DOM? Explain its working. A virtual DOM is a lightweight JavaScript object which originally is just the copy of the real DOM. It is a node tree that lists the elements, their attributes and content as Objects and their properties. React’s render function creates a node tree out of the React components. It then updates this tree in response to the mutations in the data model which is caused by various actions done by the user or by the system.  This Virtual DOM works in three simple steps.   1. Whenever any underlying data changes, the entire UI is re-rendered in Virtual DOM representation.Virtual DOM 1 - What Is ReactJS? - Edureka 2. Then the difference between the previous DOM representation and the new one is calculated.Virtual DOM 2 - React Interview Questions - Edureka 3. Once the calculations are done, the real DOM will be updated with only the things that have actually changed. Virtual DOM 3 - React Interview Questions - Edureka  8. Why can’t browsers read JSX? Browsers can only read JavaScript objects but JSX in not a regular JavaScript object. Thus to enable a browser to read JSX, first, we need to transform JSX file into a JavaScript object using JSX transformers like Babel and then pass it to the browser. 9. How different is React’s ES6 syntax when compared to ES5? Syntax has changed from ES5 to ES6 in following aspects:  i. require vs import  // ES5  var React = require('react');  // ES6  import React from 'react';  ii. export vs exports  // ES5  module.exports = Component;  // ES6  export default Component;  iii. component and function  // ES5  var MyComponent = React.createClass({  render: function() {  return  <h3>Hello Edureka!</h3>;  }  });  // ES6  class MyComponent extends React.Component {  render() {  return  <h3>Hello Edureka!</h3>;  }  }  iv. Props  // 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>;  }  }  v. state  // 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>;  }  } 10. How is React different from Angular?  |  |  |  | | --- | --- | --- | | TOPIC | REACT | ANGULAR | | *1. ARCHITECTURE* | Only the View of MVC | Complete MVC | | *2. RENDERING* | Server-side rendering | Client-side rendering | | *3. DOM* | Uses virtual DOM | Uses real DOM | | *4. DATA BINDING* | One-way data binding | Two-way data binding | | *5. DEBUGGING* | Compile time debugging | Runtime debugging | | *6. AUTHOR* | Facebook | Google |  React Components – React Interview Questions11. What do you understand from “In React, everything is a component.” Components are the building blocks of a React application’s UI. These components split up the entire UI into small independent and reusable pieces. Then it renders each of these components independent of each other without affecting the rest of the UI. 12. Explain the purpose of render() in React. Each React component must have a render() mandatorily. It returns a single React element which is the representation of the native DOM component. If more than one HTML element needs to be rendered, then they must be grouped together inside one enclosing tag such as <form>, <group>,<div> etc. This function must be kept pure i.e., it must return the same result each time it is invoked. 13. How can you embed two or more components into one? We can embed components into one in the following way:  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')  ); 14. What is Props? Props is the shorthand for Properties in React. They are read-only components which must be kept pure i.e. immutable. They are always passed down from the parent to the child components throughout the application. A child component can never send a prop back to the parent component. This help in maintaining the unidirectional data flow and are generally used to render the dynamically generated data. 15. What is a state in React and how is it used? States are the heart of React components. States are the source of data and must be kept as simple as possible. Basically, states are the objects which determine components rendering and behavior. They are mutable unlike the props and create dynamic and interactive components. They are accessed via this.state(). 16. Differentiate between states and props.  |  |  |  | | --- | --- | --- | | Conditions | State | Props | | 1. Receive initial value from parent component | Yes | Yes | | 2. Parent component can change value | No | Yes | | 3. Set default values inside component | Yes | Yes | | 4. Changes inside component | Yes | No | | 5. Set initial value for child components | Yes | Yes | | 6. Changes inside child components | No | Yes |  17. How can you update the state of a component? State of a component can be updated using this.setState().  class MyComponent extends React.Component {  constructor() {  super();  this.state = {  name: 'Maxx',  id: '101'  }  }  render()  { setTimeout(()=>{this.setState({name:'Jaeha', id:'222'})},2000)  return (<div>  <h1>Hello {this.state.name}</h1>  <h2>Your Id is {this.state.id}</h2>  </div> )  }  }  ReactDOM.render(  <MyComponent/>, document.getElementById('content')  ); 18. What is arrow function in React? How is it used? Arrow functions are more of brief syntax for writing the function expression. They are also called *‘fat arrow*‘ (=>) the functions. These functions allow to bind the context of the components properly since in ES6 auto binding is not available by default. Arrow functions are mostly useful while working with the higher order functions.  //General way  render() {  return(  <MyInput onChange={this.handleChange.bind(this) } />  );  }  //With Arrow Function  render() {  return(  <MyInput onChange={ (e) => this.handleOnChange(e) } />  );  } 19. Differentiate between stateful and stateless components.  |  |  | | --- | --- | | 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. |  20. What are the different phases of React component’s lifecycle? 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.  21. Explain the lifecycle methods of React components in detail. Some of the most important lifecycle methods are:   1. *componentWillMount()* – Executed just before rendering takes place both on the client as well as server-side. 2. *componentDidMount()* – Executed on the client side only after the first render. 3. *componentWillReceiveProps()* – Invoked as soon as the props are received from the parent class and before another render is called. 4. *shouldComponentUpdate()* – Returns true or false value based on certain conditions. If you want your component to update, return true else return false. By default, it returns false. 5. *componentWillUpdate()* – Called just before rendering takes place in the DOM. 6. *componentDidUpdate()* – Called immediately after rendering takes place. 7. *componentWillUnmount()* – Called after the component is unmounted from the DOM. It is used to clear up the memory spaces.  22. What is an event in React? 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. 23. How do you create an event in React? 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>  );  }  }); 24. 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. This is done to make sure that the events show consistent properties across different browsers. 25. What do you understand by refs in React? Refs is the short hand for References in React. It is an attribute which helps to store a reference to a particular React element or component, which will be returned by the components render configuration function. It is used to return references to a particular element or component returned by render(). They come in handy when we need DOM measurements or to add methods to the components.  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>  );  }  } 26. List some of the cases when you should use Refs. 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  27. How do you modularize code in React? We can modularize code by using the export and import properties. They help in writing the components separately in different files.  //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>  );  }  } 28. How are forms created in React? React forms are similar to HTML forms. But in React, the state is contained in the state property of the component and is only updated via setState(). Thus the elements can’t directly update their state and their submission is handled by a JavaScript function. This function has full access to the data that is entered by the user into a form.  handleSubmit(event) {  alert('A name was submitted: ' + this.state.value);  event.preventDefault();  }    render() {  return (  <form onSubmit={this.handleSubmit}>  <label>  Name:  <input type="text" value={this.state.value} onChange={this.handleSubmit} />  </label>  <input type="submit" value="Submit" />  </form>);  } 29. What do you know about controlled and uncontrolled components?  |  |  | | --- | --- | | Controlled Components | Uncontrolled Components | | 1. They do not maintain their own state | 1. They maintain their own state | | 2. Data is controlled by the parent component | 2. Data is controlled by the DOM | | 3. They take in the current values through props and then notify the changes via callbacks | 3. Refs are used to get their current values |  30. What are Higher Order Components(HOC)? Higher Order Component is an advanced way of reusing the component logic. Basically, it’s a pattern that is derived from React’s compositional nature. HOC are custom components which wrap another component within it. They can accept any dynamically provided child component but they won’t modify or copy any behavior from their input components. You can say that HOC are ‘pure’ components. 31. What can you do with HOC? HOC can be used for many tasks like:   * Code reuse, logic and bootstrap abstraction * Render High jacking * State abstraction and manipulation * Props manipulation  32. 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(). These components enhance the simplicity of the code and performance of the application. 33. What is the significance of keys in React? 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. React Redux – React Interview Questions34. What were the major problems with MVC framework? Following are some of the major problems with MVC framework:   * DOM manipulation was very expensive * Applications were slow and inefficient * There was huge memory wastage * Because of circular dependencies, a complicated model was created around models and views  35. Explain Flux. Flux is an architectural pattern which enforces the uni-directional data flow. It controls derived data and enables communication between multiple components using a central Store which has authority for all data. Any update in data throughout the application must occur here only. Flux provides stability to the application and reduces run-time errors.   36. What is Redux? Redux is one of the hottest libraries for front-end development in today’s marketplace. It is a predictable state container for JavaScript applications and is used for the entire applications state management. Applications developed with Redux are easy to test and can run in different environments showing consistent behavior. 37. What are the three principles that Redux follows?  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.    38. What do you understand by “Single source of truth”? 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. 39. List down the components of Redux.  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 the entire application is saved in the Store. 4. View – Simply displays the data provided by the Store.  40. Show how the data flows through Redux? |



### 41. How are Actions defined in Redux?

Actions in React must have a type property that indicates the type of ACTION being performed. They must be defined as a String constant and you can add more properties to it as well. In Redux, actions are created using the functions called Action Creators. Below is an example of Action and Action Creator:

function addTodo(text) {

return {

type: ADD\_TODO,

text

}

}

### 42. 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, and then it returns a new state. It determines what sort of update needs to be done based on the type of the action, and then returns new values. It returns the previous state as it is, if no work needs to be done.

### 43. What is the significance of Store in Redux?

A store is a JavaScript object which can hold the application’s state and provide a few helper methods to access the state, dispatch actions and register listeners. The entire state/ object tree of an application is saved in a single store. As a result of this, Redux is very simple and predictable. We can pass middleware to the store to handle the processing of data as well as to keep a log of various actions that change the state of stores. All the actions return a new state via reducers.

### 44. How is Redux different from Flux?

|  |  |
| --- | --- |
| Flux | Redux |
| 1. The Store contains state and change logic | 1. Store and change logic are separate |
| 2. There are multiple stores | 2. There is only one store |
| 3. All the stores are disconnected and flat | 3. Single store with hierarchical reducers |
| 4. Has singleton dispatcher | 4. No concept of dispatcher |
| 5. React components subscribe to the store | 5. Container components utilize connect |
| 6. State is mutable | 6. State is immutable |

### 45. What are the advantages of Redux?

Advantages of Redux are listed below:

* Predictability of outcome – Since there is always one source of truth, i.e. the store, there is no confusion about how to sync the current state with actions and other parts of the application.
* Maintainability – The code becomes easier to maintain with a predictable outcome and strict structure.
* Server-side rendering – You just need to pass the store created on the server, to the client side. This is very useful for initial render and provides a better user experience as it optimizes the application performance.
* Developer tools – From actions to state changes, developers can track everything going on in the application in real time.
* Community and ecosystem – Redux has a huge community behind it which makes it even more captivating to use. A large community of talented individuals contribute to the betterment of the library and develop various applications with it.
* Ease of testing – Redux’s code is mostly functions which are small, pure and isolated. This makes the code testable and independent.
* Organization – Redux is precise about how code should be organized, this makes the code more consistent and easier when a team works with it.

## React Router – React Interview Questions

### 46. What is React Router?

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

### 47. Why is switch keyword used in React Router v4?

Although a <div> is used to encapsulate multiple routes inside the Router. The ‘switch’ keyword is used when you want to display only a single route to be rendered amongst the several defined routes. The <switch> tag when in use matches the typed URL with the defined routes in sequential order. When the first match is found, it renders the specified route. Thereby bypassing the remaining routes.

### 48. Why do we need a Router in React?

A Router is used to define multiple routes and when a user types a specific URL, if this URL matches the path of any ‘route’ defined inside the router, then the user is redirected to that particular route. So basically, we need to add a Router library to our app that allows creating multiple routes with each leading to us a unique view.

<switch>

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

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

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

</switch>

### 49. List down the advantages of React Router.

Few advantages are:

1. Just like how React is based on components, in React Router v4, the API is *‘All About Components’*. A Router can be visualized as a single root component (<BrowserRouter>) in which we enclose the specific child routes (<route>).
2. No need to manually set History value: In React Router v4, all we need to do is wrap our routes within the <BrowserRouter> component.
3. The packages are split: Three packages one each for Web, Native and Core. This supports the compact size of our application. It is easy to switch over based on a similar coding style.

### 50. How is React Router different from conventional routing?

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

### **51.*What are React Lifecycles? How are they used? When does each lifecycle fire?***

Mounting

* [constructor()](https://facebook.github.io/react/docs/react-component.html#constructor)
* [componentWillMount()](https://facebook.github.io/react/docs/react-component.html#componentwillmount)
* [render()](https://facebook.github.io/react/docs/react-component.html#render)
* [componentDidMount()](https://facebook.github.io/react/docs/react-component.html#componentdidmount)

Updating

* [componentWillReceiveProps()](https://facebook.github.io/react/docs/react-component.html#componentwillreceiveprops)
* [shouldComponentUpdate()](https://facebook.github.io/react/docs/react-component.html#shouldcomponentupdate)
* [componentWillUpdate()](https://facebook.github.io/react/docs/react-component.html#componentwillupdate)
* [render()](https://facebook.github.io/react/docs/react-component.html#render)
* [componentDidUpdate()](https://facebook.github.io/react/docs/react-component.html#componentdidupdate)

Unmounting

* [componentWillUnmount()](https://facebook.github.io/react/docs/react-component.html#componentwillunmount)

### **52. What is JSX?**

JSX is simple JavaScript which allows HTML syntax and other HTML tags in the code. HTML syntax is processed into javascript calls of React framework.

### 53. What is FLUX in ReactJs?

Flux is an application architecture in React view Library which designed by Facebook for creating data layer in an application based on JavaScript.

### **54. What do you mean by Props and State in React?**

This is the basic ReactJs Interview Questions asked in an interview. Props mean the properties of the argument which is passed in javascript function. A state is used for creating a dynamic and interactive component.

#### 55. What are the refs in React?

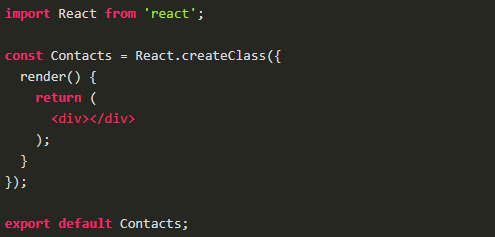
For focus management, trigger animation we use refs in React. It also contains the third party libraries.

#### 56. What do you mean a functional component in React?

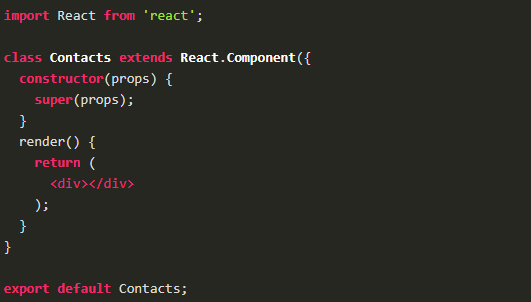
Functional component is those components which returns react elements as an element.

**57. What is React.createClass?**

**//ES5**



//ES6



**58.What is ReactDOM and what is the difference between ReactDOM andReact?**

Prior to v0.14, all ReactDOM functionality was part of React. But later, Reactand ReactDOM were split into two different libraries.

As the name implies, ReactDOM is the glue between React and the DOM. Often, we will only use it for one single thing: mounting with ReactDOM. Another useful feature of ReactDOM is ReactDOM.findDOMNode() which we can use to gain direct access to a DOM element.

For everything else, there’s React. We use React to define and create our elements, for lifecycle hooks, etc. i.e. the guts of a React application.

**59. What are the differences between a class component and functional component?**

Class components allows us to use additional features such as local state and lifecycle hooks. Also, to enable our component to have direct access to our store and thus holds state.

When our component just receives props and renders them to the page, this is a ‘stateless component’, for which a pure function can be used. These are also called dumb components or presentational components.

From the previous question, we can say that our Booklist component is functional components and are stateless.

****

On the other hand, the BookListContainer component is a class component.

**60. What are controlled components?**

In HTML, form elements such as <input>, <textarea>, and <select>typically maintain their own state and update it based on user input. When a user submits a form the values from the aforementioned elements are sent with the form. With React it works differently. The component containing the form will keep track of the value of the input in it's state and will re-render the component each time the callback function e.g. onChange is fired as the state will be updated. A form element whose value is controlled by React in this way is called a "controlled component".

With a controlled component, every state mutation will have an associated handler function. This makes it straightforward to modify or validate user input.

**61. What is a higher order component?**

A higher-order component (HOC) is an advanced technique in React for reusing component logic. HOCs are not part of the React API. They are a pattern that emerges from React’s compositional nature.

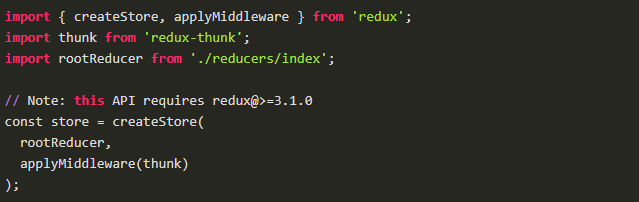
A higher-order component is a function that takes a component and returns a new component.

HOC’s allow you to reuse code, logic and bootstrap abstraction. HOCs are common in third-party React libraries. The most common is probably Redux’s connect function. Beyond simply sharing utility libraries and simple composition, HOCs are the best way to share behavior between React Components. If you find yourself writing a lot of code in different places that does the same thing, you may be able to refactor that code into a reusable HOC.

**62. What is Redux Thunk used for?**

Redux thunk is middleware that allows us to write action creators that return a function instead of an action. The thunk can then be used to delay the dispatch of an action if a certain condition is met. This allows us to handle the asyncronous dispatching of actions. The inner function receives the store methods dispatch and getState as parameters.

To enable Redux Thunk, we need to use applyMiddleware() as below

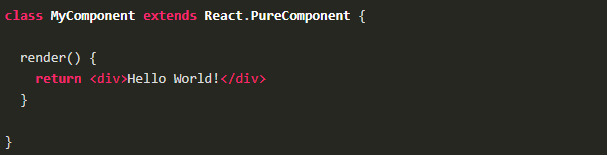


**63. What is PureComponent? When to use PureComponent over Component?**

PureComponent is exactly the same as Component except that it handles the shouldComponentUpdate method for us. When props or state changes, PureComponent will do a shallow comparison on both props and state. Component on the other hand won't compare current props and state to next out of the box. Thus, the component will re-render by default whenever shouldComponentUpdate is called.

When comparing previous props and state to next, a shallow comparison will check that primitives have the same value (eg, 1 equals 1 or that true equals true) and that the references are the same between more complex javascript values like objects and arrays.

It is good to prefer PureComponent over Component whenever we never mutate our objects.



**64. What are controlled and uncontrolled components in React?**

This relates to stateful DOM components (form elements) and the difference:

* A Controlled Component is one that takes its current value through props and notifies changes through callbacks like onChange. A parent component “controls” it by handling the callback and managing its own state and passing the new values as props to the controlled component. You could also call this a “dumb component”.
* A Uncontrolled Component is one that stores its own state internally, and you query the DOM using a ref to find its current value when you need it. This is a bit more like traditional HTML.

In most (or all) cases we should use controlled components.

**65. What is React.cloneElement? And the difference withthis.props.children?**

React.cloneElement clone and return a new React element using using the passed element as the starting point. The resulting element will have the original element's props with the new props merged in shallowly. New children will replace existing children. key and ref from the original element will be preserved.

React.cloneElement only works if our child is a single React element. For almost everything {this.props.children} is the better solution. Cloning is useful in some more advanced scenarios, where a parent send in an element and the child component needs to change some props on that element or add things like ref for accessing the actual DOM element.

**66. What is the second argument that can optionally be passed to setState and what is its purpose?**

A callback function which will be invoked when setState has finished and the component is re-rendered.

Since the setState is asynchronous, which is why it takes in a second callback function. With this function, we can do what we want immediately after state has been updated.

**67.What is redux?**

**Redux is an open-source JavaScript library for managing application state. It is most commonly used with libraries such as React or Angular for building user interfaces. Similar to Facebook's Flux architecture, it was created by Dan Abramov and Andrew Clark**

### **68. When should I learn Redux?**

What to learn can be an overwhelming question for a JavaScript developer. It helps to narrow the range of options by learning one thing at a time and focusing on problems you find in your work. Redux is a pattern for managing application state. If you do not have problems with state management, you might find the benefits of Redux harder to understand. Some UI libraries (like React) have their own state management system.

69.**Can Redux only be used with React?**

Redux can be used as a data store for any UI layer. The most common usage is with React and React Native, but there are bindings available for Angular, Angular 2, Vue, Mithril, and more.

70.**Do I need to have a particular build tool to use Redux?**

Redux is originally written in ES6 and transpiled for production into ES5 with Webpack and Babel.

71.**How do I share state between two reducers? Do I have to use combineReducers?**

Many users later want to try to share data between two reducers, but find that combineReducers does not allow them to do so. There are several approaches that can be used:

72.**Do I have to use the switch statement to handle actions?**

No. You are welcome to use any approach you'd like to respond to an action in a reducer. The switch statement is the most common approach, but it's fine to use if statements, a lookup table of functions, or to create a function that abstracts this away. In fact, while Redux does require that action objects contain a type field, your reducer logic doesn't even have to rely on that to handle the action. That said, the standard approach is definitely using a switch statement or a lookup table based on type.

**73.Can I put functions, promises, or other non-serializable items in my store state?**

It is highly recommended that you only put plain serializable objects, arrays, and primitives into your store. It's *technically* possible to insert non-serializable items into the store, but doing so can break the ability to persist and rehydrate the contents of a store, as well as interfere with time-travel debugging.

If you are okay with things like persistence and time-travel debugging potentially not working as intended, then you are totally welcome to put non-serializable items into your Redux store. Ultimately, it's *your* application, and how you implement it is up to you. As with many other things about Redux, just be sure you understand what tradeoffs are involved.

**74. What are the redux basic principles?**

Redux can be described in three fundamental principles

1. Single source of truth
2. State is read-only
3. Changes are made with pure functions

1. Single source of truth:

The state of your whole application is stored in an object tree within a single store.

2. State is read-only:

The only way to change the state is to emit an action, an object describing what happened.

OR

Your state should be read-only and can only be modified through actions

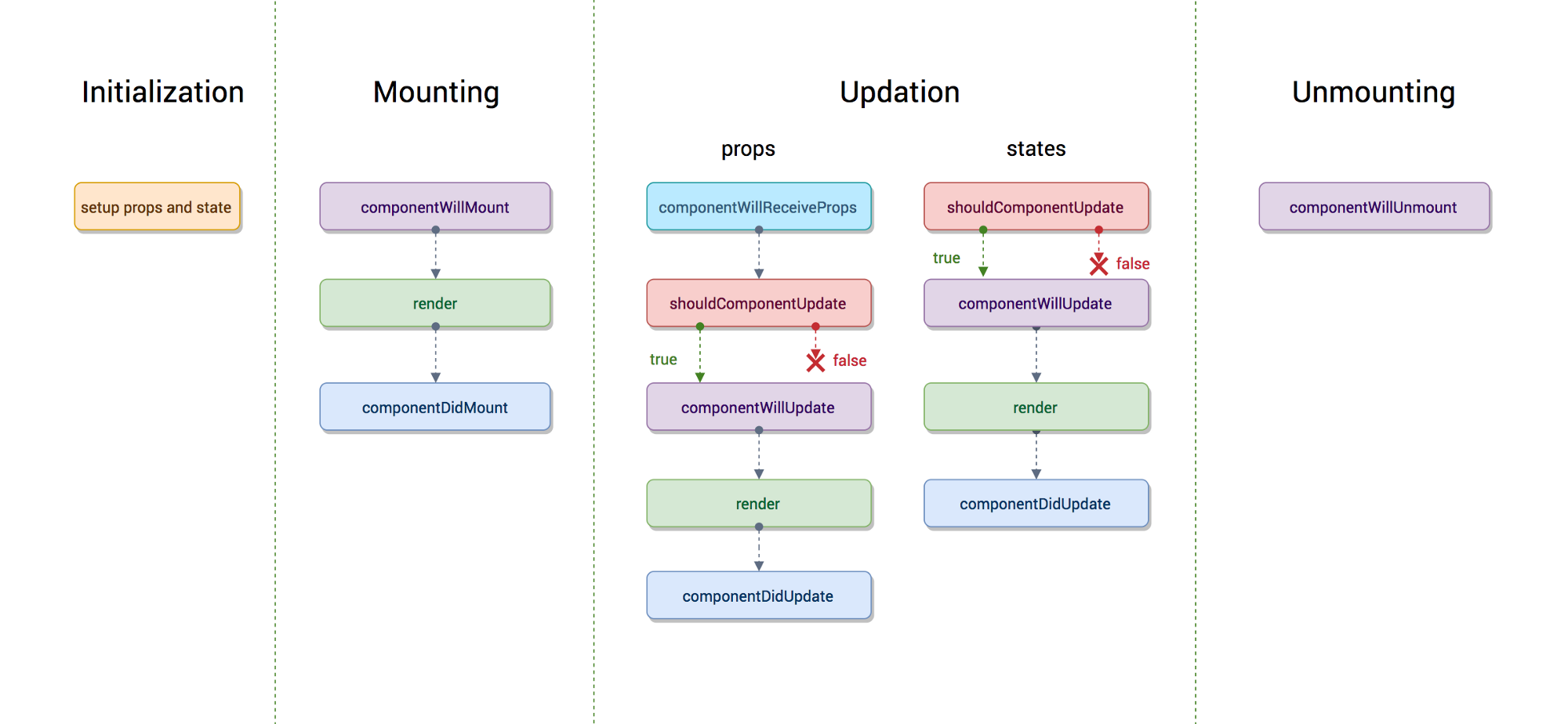
**75.What is the difference between Flux and Redux?**

|  |  |
| --- | --- |
| **REDUX** | **FLUX** |
| **Single Store** | **Multiple Store** |
| **Redux is a container for JavaScript apps.** | **Flux is a container for application state and logic that are registered to callbacks.** |
| **It offers interesting features such as writing applications, testing in different environments such as a server, client, etc.** | **It is an observer pattern that is modified to fit React.** |
| **In redux, actions can be functions or promises.** | **It is a fancy name given to observer pattern and Facebook has developed tools to aid the implementation of these patterns.** |
| **Redux is the first choice for web developers because it offers live code editing.** | **Flux supports actions that are simple JavaScript objects.** |

**76.What are the different phases of React component lifecycle?**

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

1. Initialization: In this phase react component prepares setting up the initial state and default props.
2. Mounting: The react component is ready to mount in the browser DOM. This phase covers componentWillMount and componentDidMount lifecycle methods.
3. Updating: In this phase, the component gets updated in two ways, sending the new props and updating the state. This phase covers shouldComponentUpdate, componentWillUpdate and componentDidUpdate lifecycle methods.
4. Unmounting: In this last phase, the component is not needed and get unmounted from the browser DOM. This phase include componentWillUnmount lifecycle method.



77.**What is children?**

In JSX expressions that contain both an opening tag and a closing tag, the content between those tags is passed to components automatically as a special prop: props.children.

There are some methods available in the React API to work with this prop. These include React.Children.map, React.Children.forEach, React.Children.count, React.Children.only, React.Children.toArray.

78.**What is "store" in Redux?**

Store is the object that holds the application state and provides a few helper methods to access the state, dispatch actions and register listeners. The entire state is represented by a single store. Any action returns a new state via reducers. That makes Redux very simple and predictable.

79.**What is connect function of Redux?**

**The connect() function connects a React component to a Redux store.**

**function connect(mapStateToProps?, mapDispatchToProps?, mergeProps?, options?)**

**80.What are the parameter of connect function?**

connect accepts four different parameters, all optional. By convention, they are called:

1. **mapStateToProps?: Function**
2. **mapDispatchToProps?: Function | Object**
3. **mergeProps?: Function**
4. **options?: Object**

**81.What is Provider in Redux?**

**The <Provider /> makes the Redux store available to any nested components that have been wrapped in the connect() function.**

import React from 'react'

import ReactDOM from 'react-dom'

import { Provider } from 'react-redux'

import { Router, Route } from 'react-router-dom'

import { App } from './App'

import { Foo } from './Foo'

import { Bar } from './Bar'

import createStore from './createReduxStore'

const store = createStore()

ReactDOM.render(

<Provider store={store}>

<Router history={history}>

<Route exact path="/" component={App} />

<Route path="/foo" component={Foo} />

<Route path="/bar" component={Bar} />

</Router>

</Provider>,

document.getElementById('root')

)

82.**How do I pass form value(s) to my own submit handler?**

redux-form covers handleSubmit functionality by [providing a respective handler](https://github.com/zacacollier/redux-form/blob/master/src/handleSubmit.js) that you pass as a prop - that is, a handleSubmit method for you to pass to onSubmit.

With that in mind, you can think of the redux-form handleSubmit as a middle layer for your form's submit handler. Simply write your components as you normally would, passing handleSubmit where appropriate:

import React from 'react';

import { reduxForm, Field } from 'redux-form';

const SearchBar = ({ handleChange, handleSubmit, value }) => (

<form onSubmit={handleSubmit}>

<div>

<Field

component="input"

name="search"

onChange={handleChange}

type="text"

value={value}

/>

</div>

</form>

)

export default reduxForm({ form: 'SearchBar' })(SearchBar)

You can access your form's input values via the values parameter provided to the onSubmit prop:

import React from 'react';

import { connect } from 'react-redux';

import SearchBar from './SearchBar';

let SearchContainer = ({ handleSearchSubmit }) =>

<SearchBar

onSubmit={values => handleSearchSubmit(values.search)}

/>

const mapDispatchToProps = (dispatch) => ({

handleSearchSubmit: value => dispatch({ type: 'SEARCH\_CONTAINER\_SUBMIT', payload: value }),

});

export default connect(null, mapDispatchToProps)(SearchContainer);

That's it! No need to specify event.preventDefault(). All that's left to do is handle the dispatched form data in your reducer.

83.How can I clear my form after my submission succeeds?

Excellent question. You have several options:

### **A) You can use** [**the plugin() API**](https://redux-form.com/8.1.0/docs/api/ReducerPlugin.md) **to teach the redux-form reducer to respond to the action dispatched when your submission succeeds.**

This is the proper Redux way to do it and has the benefit of not dispatching another action.

import {createStore, combineReducers} from 'redux';

import {reducer as formReducer} from 'redux-form';

import {ACCOUNT\_SAVE\_SUCCESS} from '../actions/actionTypes';

const reducers = {

// ... your other reducers here ...

form: formReducer.plugin({

account: (state, action) => { // <------ 'account' is name of form given to reduxForm()

switch(action.type) {

case ACCOUNT\_SAVE\_SUCCESS:

return undefined; // <--- blow away form data

default:

return state;

}

}

})

}

const reducer = combineReducers(reducers);

const store = createStore(reducer);

### **B) Simply unmount your form component**

For many use cases, you will want to either hide your form component after submission succeeds or navigate away to another page, which will cause redux-form's default behavior of destroying the form data in the reducer incomponentWillUnmount.

### **C) You can call this.props.reset() from inside your form after your submission succeeds.**

submitMyForm(data) {

const { createRecord, reset } = this.props;

return createRecord(data).then(() => {

reset();

// do other success stuff

});

}

render() {

const { handleSubmit } = this.props;

return (

<form onSubmit={handleSubmit(this.submitMyForm.bind(this))}>

// inputs

</form>

);

}

### **D) You can dispatch reset() from any connected component**

Extremely flexible, but you have to know your form name and have dispatch available.

import {reset} from 'redux-form';

...

dispatch(reset('myForm')); // requires form name

84.**How can I submit my form when the user presses Enter?**

**The default browser behavior for text and password inputs when ↵ is pressed is to activate the first <button> that does not have type="button" specified. The recommended way to structure your form is as follows:**

**render() {**

**const {fields: {firstName, lastName}, handleSubmit} = this.props.fields;**

**return (**

**<form onSubmit={handleSubmit}>**

**<button type="button">Load Data</button>**

**<button type="button">Delete Record</button>**

**<label htmlFor="firstName">First Name</label>**

**<input type="text" {...firstName}/>**

**<label htmlFor="lastName">Last Name</label>**

**<input type="text" {...lastName}/>**

**<button type="submit">Submit</button>**

**<button type="button">Do Something Else</button>**

**</form>**

**);**

**}**

**Things to notice:**

1. **handleSubmit is on the <form> element, not on the <button>. It does not hurt to put it on the onClick for the button, but it doesn't accomplish anything.**
2. **The submit button is explicitly marked type="submit".**
3. **You may place any other buttons inside the form as long as they are labeled type="button".**

**85.**