No.	Questions
	React Router
129	What is React Router?
130	How React Router is different from history library?
131	What are the <Router> components of React Router v4?
132	What is the purpose of push and replace methods of history?
133	How do you programmatically navigate using React router v4?
134	How to get query parameters in React Router v4
135	Why you get "Router may have only one child element" warning?
136	How to pass params to history.push method in React Router v4?
137	How to implement default or NotFound page?
138	How to get history on React Router v4?
139	How to perform automatic redirect after login?
	React Internationalization
140	What is React-Intl?
141	What are the main features of React Intl?
142	What are the two ways of formatting in React Intl?
143	How to use FormattedMessage as placeholder using React Intl?
144	How to access current locale with React Intl
145	How to format date using React Intl?
	React Testing
146	What is Shallow Renderer in React testing?
147	What is TestRenderer package in React?
148	What is the purpose of ReactTestUtils package?
149	What is Jest?
150	What are the advantages of Jest over Jasmine?
151	Give a simple example of Jest test case
	React Redux
152	What is Flux?
153	What is Redux?
154	What are the core principles of Redux?
155	What are the downsides of Redux compared to Flux?
156	What is the difference between mapStateToProps() and mapDispatchToProps()?
157	Can I dispatch an action in reducer?
158	How to access Redux store outside a component?
159	What are the drawbacks of MVW pattern
160	Are there any similarities between Redux and RxJS?
161	How to dispatch an action on load?

No.	Questions
232	In which scenarios error boundaries do not catch errors?
233	Why do you not need error boundaries for event handlers?
234	What is the difference between try catch block and error boundaries?
235	What is the behavior of uncaught errors in react 16?
236	What is the proper placement for error boundaries?
237	What is the benefit of component stack trace from error boundary?
238	What is the required method to be defined for a class component?
239	What are the possible return types of render method?
240	What is the main purpose of constructor?
241	Is it mandatory to define constructor for React component?
242	What are default props?
243	Why should not call setState in componentWillUnmount?
244	What is the purpose of getDerivedStateFromError?
245	What is the methods order when component re-rendered?
246	What are the methods invoked during error handling?
247	What is the purpose of displayName class property?
248	What is the browser support for react applications?
249	What is the purpose of unmountComponentAtNode method?
250	What is code-splitting?
251	What is the benefit of strict mode?
252	What are Keyed Fragments?
253	Does React support all HTML attributes?
254	What are the limitations with HOCs?
255	How to debug forwardRefs in DevTools?
256	When component props defaults to true?
257	What is NextJS and major features of it?
258	How do you pass an event handler to a component?
259	Is it good to use arrow functions in render methods?
260	How to prevent a function from being called multiple times?
261	How JSX prevents Injection Attacks?
262	How do you update rendered elements?
263	How do you say that props are read only?
264	How do you say that state updates are merged?
265	How do you pass arguments to an event handler?
266	How to prevent component from rendering?

No.	Questions
267	What are the conditions to safely use the index as a key?
268	Is it keys should be globally unique?
269	What is the popular choice for form handling?
270	What are the advantages of formik over redux form library?
271	Why do you not required to use inheritance?
272	Can I use web components in react application?
273	What is dynamic import?
274	What are loadable components?
275	What is suspense component?
276	What is route based code splitting?
277	Give an example on How to use context?
278	What is the purpose of default value in context?
279	How do you use contextType?
280	What is a consumer?
281	How do you solve performance corner cases while using context?
282	What is the purpose of forwardRef in HOCs?
283	Is it ref argument available for all functions or class components?
284	Why do you need additional care for component libraries while using forward refs?
285	How to create react class components without ES6?
286	Is it possible to use react without JSX?
287	What is diffing algorithm?
288	What are the rules covered by diffing algorithm?
289	When do you need to use refs?
290	Is it prop must be named as render for render props?
291	What are the problems of using render props with pure components?
292	How do you create HOC using render props?
293	What is windowing technique?
294	How do you print falsy values in JSX?
295	What is the typical use case of portals?
296	How do you set default value for uncontrolled component?
297	What is your favorite React stack?
298	What is the difference between Real DOM and Virtual DOM?
299	How to add Bootstrap to a react application?
300	Can you list down top websites or applications using react as front end framework?
301	Is it recommended to use CSS In JS technique in React?

“jsx harmony function Greeting({ message }) {   return <h1>{Hello, ${message}‘}

} “‘

2. **Class Components:** You can also use ES6 class to define a component. The above function component can be written as:

```
jsx harmony  class Greeting extends React.Component {
render() {      return <h1>{`Hello, ${this.props.message}`}</h1>
}  }
```

Back to Top

6.

When to use a Class Component over a Function Component?

If the component needs *state or lifecycle methods* then use class component otherwise use function component. *However, from React 16.8 with the addition of Hooks, you could use state , lifecycle methods and other features that were only available in class component right in your function component. So, it is always recommended to use Function components, unless you need a React function like a Error Function component equivalent is not present yet, like Error Boundaries*

Back to Top

7.

What are Pure Components?

React.PureComponent is exactly the same as *React.Component* except that it handles the `shouldComponentUpdate()` method for you. 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.

Back to Top

8.

What is state in React?

State of a component is an object that holds some information that may change over the lifetime of the component. We should always try to make our state as simple as possible and minimize the number of stateful components.

Let's create a user component with message state,

33.

What are the different phases of component lifecycle?

The component lifecycle has three distinct lifecycle phases:

1. **Mounting:** The component is ready to mount in the browser DOM. This phase covers initialization from `constructor()`, `getDerivedStateFromProps()`, `render()`, and `componentDidMount()` lifecycle methods.
2. **Updating:** In this phase, the component gets updated in two ways, sending the new props and updating the state either from `setState()` or `forceUpdate()`. This phase covers `getDerivedStateFromProps()`, `shouldComponentUpdate()`, `render()`, `getSnapshotBeforeUpdate()` and `componentDidUpdate()` lifecycle methods.
3. **Unmounting:** In this last phase, the component is not needed and gets unmounted from the browser DOM. This phase includes `componentWillUnmount()` lifecycle method.

It's worth mentioning that React internally has a concept of phases when applying changes to the DOM. They are separated as follows

1. **Render** The component will render without any side effects. This applies to Pure components and in this phase, React can pause, abort, or restart the render.
2. **Pre-commit** Before the component actually applies the changes to the DOM, there is a moment that allows React to read from the DOM through the `getSnapshotBeforeUpdate()`.
3. **Commit** React works with the DOM and executes the final lifecycles respectively `componentDidMount()` for mounting, `componentDidUpdate()` for updating, and `componentWillUnmount()` for unmounting.

React 16.3+ Phases (or an interactive version)

Before React 16.3

Back to Top

34.

What are the lifecycle methods of React?

Before React 16.3

- **componentWillMount:** Executed before rendering and is used for App level configuration in your root component.

Passing props:

```
class MyComponent extends React.Component {
  constructor(props) {
    super(props)

    console.log(this.props) // prints { name: 'John', age: 42 }
  }
}
```

Not passing props:

```
class MyComponent extends React.Component {
  constructor(props) {
    super()

    console.log(this.props) // prints undefined

    // but props parameter is still available
    console.log(props) // prints { name: 'John', age: 42 }
  }

  render() {
    // no difference outside constructor
    console.log(this.props) // prints { name: 'John', age: 42 }
  }
```


The above code snippets reveals that `this.props` is different only within the constructor. It would be the same outside the constructor.

Back to Top

41.

What is reconciliation?

When a component's props or state change, React decides whether an actual DOM update is necessary by comparing the newly returned element with the previously rendered one. When they are not equal, React will update the DOM. This process is called *reconciliation*.

Back to Top

42.

How to set state with a dynamic key name?

If you are using ES6 or the Babel transpiler to transform your JSX code then you can accomplish this with *computed property names*.

Why React uses `className` over `class` attribute?

`class` is a keyword in JavaScript, and JSX is an extension of JavaScript. That's the principal reason why React uses `className` instead of `class`. Pass a string as the `className` prop.

```
jsx harmony render() {   return <span className={'menu
navigation-menu'}>{'Menu'}</span> }
```

Back to Top

46.

What are fragments?

It's a common pattern in React which is used for a component to return multiple elements. *Fragments* let you group a list of children without adding extra nodes to the DOM.

```
jsx harmony render() {   return (     <React.Fragment>
<ChildA />       <ChildB />       <ChildC />     </React.Fragment>
) }
```

There is also a *shorter syntax*, but it's not supported in many tools:

```
jsx harmony render() {   return (     <>     <ChildA />
<ChildB />     <ChildC />     </> ) }
```

Back to Top

47.

Why fragments are better than container divs?

Below are the list of reasons,

1. Fragments are a bit faster and use less memory by not creating an extra DOM node. This only has a real benefit on very large and deep trees.
2. Some CSS mechanisms like *Flexbox* and *CSS Grid* have a special parent-child relationships, and adding divs in the middle makes it hard to keep the desired layout.
3. The DOM Inspector is less cluttered.

Back to Top

48.

What are portals in React?

Portal is a recommended way to render children into a DOM node that exists outside the DOM hierarchy of the parent component.

```
    return (
      // JSX
    )
  }
```

Back to Top

51.

How to apply validation on props in React?

When the application is running in *development mode*, React will automatically check all props that we set on components to make sure they have *correct type*. If the type is incorrect, React will generate warning messages in the console. It's disabled in *production mode* due to performance impact. The mandatory props are defined with `isRequired`.

The set of predefined prop types:

1. `PropTypes.number`
2. `PropTypes.string`
3. `PropTypes.array`
4. `PropTypes.object`
5. `PropTypes.func`
6. `PropTypes.node`
7. `PropTypes.element`
8. `PropTypes.bool`
9. `PropTypes.symbol`
10. `PropTypes.any`

We can define `propTypes` for `User` component as below:

"'jsx harmony import React from 'react' import PropTypes from 'prop-types'

class User extends React.Component { static propTypes = { name: PropTypes.string.isRequired, age: PropTypes.number.isRequired }

render() { return (<>

```
{Welcome, ${this.props.name}}

    <h2>{`Age, ${this.props.age}`}</h2>
  </>
)

} }

**Note:** In React v15.5 *PropTypes* were moved from `React.PropTypes` to `prop-types`

*The Equivalent Functional Component*
```

```jsx harmony
import React from 'react'
import PropTypes from 'prop-types'

function User({name, age}) {
  return (
    <>
      <h1>{`Welcome, ${name}`}</h1>
      <h2>{`Age, ${age}`}</h2>
    </>
  )
}

User.propTypes = {
    name: PropTypes.string.isRequired,
    age: PropTypes.number.isRequired
  }
```

Back to Top

52.

What are the advantages of React?

Below are the list of main advantages of React,

1. Increases the application's performance with *Virtual DOM*.
2. JSX makes code easy to read and write.
3. It renders both on client and server side (*SSR*).
4. Easy to integrate with frameworks (Angular, Backbone) since it is only a view library.
5. Easy to write unit and integration tests with tools such as Jest.

Back to Top

53.

What are the limitations of React?

Apart from the advantages, there are few limitations of React too,

1. React is just a view library, not a full framework.
2. There is a learning curve for beginners who are new to web development.
3. Integrating React into a traditional MVC framework requires some additional configuration.
4. The code complexity increases with inline templating and JSX.

server-side rendering (SSR). The following methods can be used in both the server and browser environments:

1. `renderToString()`
2. `renderToStaticMarkup()`

For example, you generally run a Node-based web server like Express, Hapi, or Koa, and you call `renderToString` to render your root component to a string, which you then send as response.

```
// using Express
import { renderToString } from 'react-dom/server'
import MyPage from './MyPage'

app.get('/', (req, res) => {
  res.write('<!DOCTYPE html><html><head><title>My Page</title></head><body>')
  res.write('<div id="content">')
  res.write(renderToString(<MyPage/>))
  res.write('</div></body></html>')
  res.end()
})
```

Back to Top

60.

How to use innerHTML in React?

The `dangerouslySetInnerHTML` attribute is React's replacement for using `innerHTML` in the browser DOM. Just like `innerHTML`, it is risky to use this attribute considering cross-site scripting (XSS) attacks. You just need to pass a `__html` object as key and HTML text as value.

In this example MyComponent uses `dangerouslySetInnerHTML` attribute for setting HTML markup:

“‘jsx harmony function createMarkup() { return { __html: ‘First · Second’ } }

function MyComponent() { return

} “‘

Back to Top

61.

How to use styles in React?

The `style` attribute accepts a JavaScript object with camelCased properties rather than a CSS string. This is consistent with the DOM style JavaScript property, is more efficient, and prevents XSS security holes.

“‘jsx harmony const divStyle = { color: ‘blue’, backgroundImage: ‘url(’ + imgUrl + ‘)’ };

function HelloWorldComponent() { return

Hello World!

} “‘

Style keys are camelCased in order to be consistent with accessing the properties on DOM nodes in JavaScript (e.g. `node.style.backgroundImage`).

Back to Top

62.

How events are different in React?

Handling events in React elements has some syntactic differences:

1. React event handlers are named using camelCase, rather than lowercase.
2. With JSX you pass a function as the event handler, rather than a string.

Back to Top

63.

What will happen if you use `setState()` in constructor?

When you use `setState()`, then apart from assigning to the object state React also re-renders the component and all its children. You would get error like this: *Can only update a mounted or mounting component.* So we need to use `this.state` to initialize variables inside constructor.

Back to Top

64.

What is the impact of indexes as keys?

Keys should be stable, predictable, and unique so that React can keep track of elements.

In the below code snippet each element’s key will be based on ordering, rather than tied to the data that is being represented. This limits the optimizations that React can do.

```
jsx harmony {todos.map((todo, index) =>   <Todo     {...todo}
key={index}   /> )}
```

What is the difference between `super()` and `super(props)` in React using ES6 classes?

When you want to access `this.props` in `constructor()` then you should pass props to `super()` method.

Using `super(props)`:

```
class MyComponent extends React.Component {
  constructor(props) {
    super(props)
    console.log(this.props) // { name: 'John', ... }
  }
}
```

Using `super()`:

```
class MyComponent extends React.Component {
  constructor(props) {
    super()
    console.log(this.props) // undefined
  }
}
```

Outside `constructor()` both will display same value for `this.props`.

Back to Top

92.

How to loop inside JSX?

You can simply use `Array.prototype.map` with ES6 *arrow function* syntax.

For example, the `items` array of objects is mapped into an array of components:

```
jsx harmony <tbody>   {items.map(item => <SomeComponent
key={item.id} name={item.name} />)} </tbody>
```

But you can't iterate using `for` loop:

```
jsx harmony <tbody>   for (let i = 0; i < items.length; i++)
{     <SomeComponent key={items[i].id} name={items[i].name}
/>   } </tbody>
```

This is because JSX tags are transpiled into *function calls*, and you can't use statements inside expressions. This may change thanks to `do` expressions which are *stage 1 proposal*.

Back to Top

104.

Is it possible to use React without rendering HTML?

It is possible with latest version (>=16.2). Below are the possible options:

```
jsx harmony render() {   return false }
```

```
jsx harmony render() {   return null }
```

```
jsx harmony render() {   return [] }
```

```
jsx harmony render() {   return <React.Fragment></React.Fragment>
}
```

```
jsx harmony render() {   return <></> }
```

Returning `undefined` won't work.

Back to Top

105.

How to pretty print JSON with React?

We can use `<pre>` tag so that the formatting of the `JSON.stringify()` is retained:

“‘jsx harmony const data = { name: ‘John’, age: 42 }

class User extends React.Component { render() { return (

)

} }

React.render(, document.getElementById(‘container’)) “‘

Back to Top

106.

Why you can't update props in React?

The React philosophy is that props should be *immutable* and *top-down*. This means that a parent can send any prop values to a child, but the child can't modify received props.

Back to Top

107.

3. **There is no nice Flow integration yet:** Flux currently lets you do very impressive static type checks which Redux doesn't support yet.

Back to Top

156.

What is the difference between `mapStateToProps()` and `mapDispatchToProps()`?

`mapStateToProps()` is a utility which helps your component get updated state (which is updated by some other components):

```
const mapStateToProps = (state) => {
  return {
    todos: getVisibleTodos(state.todos, state.visibilityFilter)
  }
}
```

`mapDispatchToProps()` is a utility which will help your component to fire an action event (dispatching action which may cause change of application state):

```
const mapDispatchToProps = (dispatch) => {
  return {
    onTodoClick: (id) => {
      dispatch(toggleTodo(id))
    }
  }
}
```


It is recommended to always use the "object shorthand" form for the `mapDispatchToProps`.

Redux wraps it in another function that looks like (...args) => dispatch(onTodoClick(...args)), and pass that wrapper function as a prop to your component.

```
const mapDispatchToProps = ({
  onTodoClick
})
```

Back to Top

157.

Can I dispatch an action in reducer?

Dispatching an action within a reducer is an **anti-pattern**. Your reducer should be *without side effects*, simply digesting the action payload and

returning a new state object. Adding listeners and dispatching actions within the reducer can lead to chained actions and other side effects.

Back to Top

158.

How to access Redux store outside a component?

You just need to export the store from the module where it created with `createStore()`. Also, it shouldn't pollute the global window object.

```
store = createStore(myReducer)

export default store
```

Back to Top

159.

What are the drawbacks of MVW pattern?

1. DOM manipulation is very expensive which causes applications to behave slow and inefficient.
2. Due to circular dependencies, a complicated model was created around models and views.
3. Lot of data changes happens for collaborative applications(like Google Docs).
4. No way to do undo (travel back in time) easily without adding so much extra code.

Back to Top

160.

Are there any similarities between Redux and RxJS?

These libraries are very different for very different purposes, but there are some vague similarities.

Redux is a tool for managing state throughout the application. It is usually used as an architecture for UIs. Think of it as an alternative to (half of) Angular. RxJS is a reactive programming library. It is usually used as a tool to accomplish asynchronous tasks in JavaScript. Think of it as an alternative to Promises. Redux uses the Reactive paradigm because the Store is reactive. The Store observes actions from a distance, and changes itself. RxJS also uses the Reactive paradigm, but instead of being an architecture, it gives you basic building blocks, Observables, to accomplish this pattern.

Back to Top

For example, you can add `redux-thunk` and `logger` passing them as arguments to `applyMiddleware()`:

```
import { createStore, applyMiddleware } from 'redux'
const createStoreWithMiddleware = applyMiddleware(ReduxThunk, logger)(createStore)
```

Back to Top

186.

How to set initial state in Redux?

You need to pass initial state as second argument to createStore:

```
const rootReducer = combineReducers({
  todos: todos,
  visibilityFilter: visibilityFilter
})

const initialState = {
  todos: [{ id: 123, name: 'example', completed: false }]
}

const store = createStore(
  rootReducer,
  initialState
)
```

Back to Top

187.

How Relay is different from Redux?

Relay is similar to Redux in that they both use a single store. The main difference is that relay only manages state originated from the server, and all access to the state is used via *GraphQL* queries (for reading data) and mutations (for changing data). Relay caches the data for you and optimizes data fetching for you, by fetching only changed data and nothing more.

188.

What is an action in Redux?

Actions are plain JavaScript objects or payloads of information that send data from your application to your store. They are the only source of information for the store. Actions must have a type property that indicates the type of action being performed.

For example, let's take an action which represents adding a new todo item:

PropTypes is a *basic type checker* (runtime checker) which has been patched onto React. It can't check anything other than the types of the props being passed to a given component. If you want more flexible typechecking for your entire project Flow/TypeScript are appropriate choices.

Back to Top

195.

How to use Font Awesome icons in React?

The below steps followed to include Font Awesome in React:

1. Install `font-awesome`:

   ```
   $ npm install --save font-awesome
   ```

2. Import `font-awesome` in your `index.js` file:

   ```
   import 'font-awesome/css/font-awesome.min.css'
   ```

3. Add Font Awesome classes in `className`:

   ```
   render() {
     return <div><i className={'fa fa-spinner'} /></div>
   }
   ```

Back to Top

196.

What is React Dev Tools?

React Developer Tools let you inspect the component hierarchy, including component props and state. It exists both as a browser extension (for Chrome and Firefox), and as a standalone app (works with other environments including Safari, IE, and React Native).

The official extensions available for different browsers or environments.

1. **Chrome extension**
2. **Firefox extension**
3. **Standalone app** (Safari, React Native, etc)

Back to Top

197.

Why is DevTools not loading in Chrome for local files?

If you opened a local HTML file in your browser (`file://...`) then you must first open *Chrome Extensions* and check `Allow access to file URLs`.

```
npx create-react-app my-app --typescript

# or

yarn create react-app my-app --typescript
```

But for lower versions of react scripts, just supply `--scripts-version` option as `react-scripts-ts` while you create a new project. `react-scripts-ts` is a set of adjustments to take the standard `create-react-app` project pipeline and bring TypeScript into the mix.

Now the project layout should look like the following:

```
my-app/
  .gitignore
  images.d.ts
  node_modules/
  public/
  src/
     ...
  package.json
  tsconfig.json
  tsconfig.prod.json
  tsconfig.test.json
  tslint.json
```


Miscellaneous

Back to Top

206.

What are the main features of Reselect library?

Let's see the main features of Reselect library,

1. Selectors can compute derived data, allowing Redux to store the minimal possible state.
2. Selectors are efficient. A selector is not recomputed unless one of its arguments changes.
3. Selectors are composable. They can be used as input to other selectors.

207.

Give an example of Reselect usage? Let's take calculations and different amounts of a shipment order with the simplified usage of Reselect:

```
 }
}
```

Note: In React v16.3,

Back to Top

216.

What is render hijacking in react?

The concept of render hijacking is the ability to control what a component will output from another component. It means that you decorate your component by wrapping it into a Higher-Order component. By wrapping, you can inject additional props or make other changes, which can cause changing logic of rendering. It does not actually enable hijacking, but by using HOC you make your component behave differently.

Back to Top

217.

What are HOC factory implementations?

There are two main ways of implementing HOCs in React.

1. Props Proxy (PP) and
2. Inheritance Inversion (II)

But they follow different approaches for manipulating the *WrappedComponent*.

Props Proxy

In this approach, the render method of the HOC returns a React Element of the type of the WrappedComponent. We also pass through the props that the HOC receives, hence the name **Props Proxy**.

```
function ppHOC(WrappedComponent) {
 return class PP extends React.Component {
   render() {
     return <WrappedComponent {...this.props}/>
   }
 }
}
```

Inheritance Inversion

In this approach, the returned HOC class (Enhancer) extends the WrappedComponent. It is called Inheritance Inversion because instead of the WrappedComponent extending some Enhancer class, it is passively

220.

What is the purpose of registerServiceWorker in React?

React creates a service worker for you without any configuration by default. The service worker is a web API that helps you cache your assets and other files so that when the user is offline or on a slow network, he/she can still see results on the screen, as such, it helps you build a better user experience, that's what you should know about service worker for now. It's all about adding offline capabilities to your site.

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import registerServiceWorker from './registerServiceWorker';

ReactDOM.render(<App />, document.getElementById('root'));
registerServiceWorker();
```

Back to Top

221.

What is React memo function?

Class components can be restricted from re-rendering when their input props are the same using **PureComponent or shouldComponentUpdate**. Now you can do the same with function components by wrapping them in **React.memo**.

```
const MyComponent = React.memo(function MyComponent(props) {
 /* only rerenders if props change */
});
```

Back to Top

222.

What is React lazy function?

The `React.lazy` function lets you render a dynamic import as a regular component. It will automatically load the bundle containing the `OtherComponent` when the component gets rendered. This must return a Promise which resolves to a module with a default export containing a React component.

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

function MyComponent() {
 return (
```

```
    <div>
      <OtherComponent />
    </div>
  );
}
```

Note: `React.lazy` and `Suspense` is not yet available for server-side rendering. If you want to do code-splitting in a server rendered app, we still recommend React Loadable.

Back to Top

223.

How to prevent unnecessary updates using setState?

You can compare the current value of the state with an existing state value and decide whether to rerender the page or not. If the values are the same then you need to return **null** to stop re-rendering otherwise return the latest state value.

For example, the user profile information is conditionally rendered as follows,

```
getUserProfile = user => {
  const latestAddress = user.address;
  this.setState(state => {
    if (state.address === latestAddress) {
      return null;
    } else {
      return { title: latestAddress };
    }
  });
};
```


Back to Top

224.

How do you render Array, Strings and Numbers in React 16 Version?

Arrays: Unlike older releases, you don't need to make sure **render** method return a single element in React16. You are able to return multiple sibling elements without a wrapping element by returning an array.

For example, let us take the below list of developers,

```
const ReactJSDevs = () => {
  return [
    <li key="1">John</li>,
```

2. Redux-Form calls your entire top-level Redux reducer multiple times ON EVERY SINGLE KEYSTROKE. This way it increases input latency for large apps.
3. Redux-Form is 22.5 kB minified gzipped whereas Formik is 12.7 kB

Back to Top

271.

Why are you not required to use inheritance?

In React, it is recommended to use composition over inheritance to reuse code between components. Both Props and composition give you all the flexibility you need to customize a component's look and behavior explicitly and safely. Whereas, If you want to reuse non-UI functionality between components, it is suggested to extract it into a separate JavaScript module. Later components import it and use that function, object, or class, without extending it.

Back to Top

272.

Can I use web components in react application?

Yes, you can use web components in a react application. Even though many developers won't use this combination, it may require especially if you are using third-party UI components that are written using Web Components.

For example, let us use `Vaadin` date picker web component as below,

```
import React, { Component } from 'react';
import './App.css';
import '@vaadin/vaadin-date-picker';
class App extends Component {
  render() {
    return (
      <div className="App">
        <vaadin-date-picker label="When were you born?"></vaadin-date-picker>
      </div>
    );
  }
}
export default App;
```

Back to Top

273.

refs get assigned to, and what types are exported. These changes can break apps and other libraries that depend on the old behavior.

Back to Top

285.

How to create react class components without ES6?

If you don't use ES6 then you may need to use the create-react-class module instead. For default props, you need to define getDefaultProps() as a function on the passed object. Whereas for initial state, you have to provide a separate getInitialState method that returns the initial state.

```
var Greeting = createReactClass({
  getDefaultProps: function() {
      return {
        name: 'Jhohn'
      };
    },
  getInitialState: function() {
      return {message: this.props.message};
    },
  handleClick: function() {
     console.log(this.state.message);
  },
  render: function() {
    return <h1>Hello {this.props.name}</h1>;
  }
});
```


Note: If you use createReactClass then auto binding is available for all methods. i.e, You don't need to use `.bind(this)` with in constructor for event handlers.

Back to Top

286.

Is it possible to use react without JSX?

Yes, JSX is not mandatory for using React. Actually it is convenient when you don't want to set up compilation in your build environment. Each JSX element is just syntactic sugar for calling `React.createElement(component, props, ...children)`.

For example, let us take a greeting example with JSX,

```
class Greeting extends React.Component {
  render() {
```

Back to Top

291.

What are the problems of using render props with pure components?

If you create a function inside a render method, it negates the purpose of pure component. Because the shallow prop comparison will always return false for new props, and each render in this case will generate a new value for the render prop. You can solve this issue by defining the render function as instance method.

Back to Top

292.

How do you create HOC using render props?

You can implement most higher-order components (HOC) using a regular component with a render prop. For example, if you would prefer to have a withMouse HOC instead of a component, you could easily create one using a regular with a render prop.

```
function withMouse(Component) {
  return class extends React.Component {
    render() {
      return (
        <Mouse render={mouse => (
          <Component {...this.props} mouse={mouse} />
        )}/>
      );
    }
  }
}
```

This way render props gives the flexibility of using either pattern.

Back to Top

293.

What is windowing technique?

Windowing is a technique that only renders a small subset of your rows at any given time, and can dramatically reduce the time it takes to re-render the components as well as the number of DOM nodes created. If your application renders long lists of data then this technique is recommended. Both react-window and react-virtualized are popular windowing libraries

311.

Do browsers understand JSX code?

No, browsers can't understand JSX code. You need a transpiler to convert your JSX to regular Javascript that browsers can understand. The most widely used transpiler right now is Babel.

Back to Top

312.

Describe about data flow in react?

React implements one-way reactive data flow using props which reduce boilerplate and is easier to understand than traditional two-way data binding.

Back to Top

313.

What is react scripts?

The `react-scripts` package is a set of scripts from the create-react-app starter pack which helps you kick off projects without configuring. The `react-scripts start` command serves up the development environment and starts a server, as well as hot module reloading.

Back to Top

314.

What are the features of create react app?

Below are the list of some of the features provided by create react app.

1. React, JSX, ES6, Typescript and Flow syntax support.
2. Autoprefixed CSS
3. CSS Reset/Normalize
4. A live development server
5. A fast interactive unit test runner with built-in support for coverage reporting
6. A build script to bundle JS, CSS, and images for production, with hashes and sourcemaps
7. An offline-first service worker and a web app manifest, meeting all the Progressive Web App criteria.

Back to Top

What is the difference between async mode and concurrent mode?

Both refers the same thing. Previously concurrent Mode being referred to as “Async Mode” by React team. The name has been changed to highlight React’s ability to perform work on different priority levels. So it avoids the confusion from other approaches to Async Rendering.

Back to Top

321.

Can I use javascript urls in react16.9?

Yes, you can use javascript: URLs but it will log a warning in the console. Because URLs starting with javascript: are dangerous by including unsanitized output in a tag like `<a href>` and create a security hole.

```
const companyProfile = {
  website: "javascript: alert('Your website is hacked')",
};
// It will log a warning
<a href={companyProfile.website}>More details</a>
```

Remember that the future version will throw an error for javascript URLs.

Back to Top

322.

What is the purpose of eslint plugin for hooks?

The ESLint plugin enforces rules of Hooks to avoid bugs. It assumes that any function starting with ”use” and a capital letter right after it is a Hook. In particular, the rule enforces that,

1. Calls to Hooks are either inside a PascalCase function (assumed to be a component) or another useSomething function (assumed to be a custom Hook).
2. Hooks are called in the same order on every render.

Back to Top

323.

What is the difference between Imperative and Declarative in React?

Imagine a simple UI component, such as a “Like” button. When you tap it, it turns blue if it was previously grey, and grey if it was previously blue.

The imperative way of doing this would be: