General React Interview Questions

1) What is React?

React is a declarative, efficient, flexible open source front-end JavaScript library developed by Facebook in 2011. It follows the component-based approach for building reusable UI components, especially for single page application. It is used for developing interactive view layer of web and mobile apps. It was created by Jordan Walke, a software engineer at Facebook. It was initially deployed on Facebook's News Feed section in 2011 and later used in its products like WhatsApp & Instagram.

For More Information, ***Click here***.

2) What are the features of React?

React framework gaining quick popularity as the best framework among web developers. The main features of React are:

- JSX
- Components
- One-way Data Binding
- Virtual DOM
- Simplicity
- Performance

For More Information, ***Click here***.

3) What are the most crucial advantages of using React?

Following is a list of the most crucial advantages of using React:

React is easy to learn and use

React comes with good availability of documentation, tutorials, and training resources. It is easy for any developer to switch from JavaScript background to React and easily understand and start creating web apps using React. Anyone with little knowledge of JavaScript can start building web applications using React.

React follows the MVC architecture.

React is the V (view part) in the MVC (Model-View-Controller) architecture model and is referred to as "one of the JavaScript frameworks." It is not fully featured but has many advantages of the open-source JavaScript User Interface (UI) library, which helps execute the task in a better manner.

React uses Virtual DOM to improve efficiency.

React uses virtual DOM to render the view. The virtual DOM is a virtual representation of the real DOM. Each time the data changes in a react app, a new virtual DOM gets created. Creating a virtual DOM is much faster than rendering the UI inside the browser. Therefore, with the use of virtual DOM, the efficiency of the app improves. That's why React provides great efficiency.

Creating dynamic web applications is easy.

In React, creating a dynamic web application is much easier. It requires less coding and gives more functionality. It uses JSX (JavaScript Extension), which is a particular syntax letting HTML quotes and HTML tag syntax to render particular subcomponents.

React is SEO-friendly.

React facilitates a developer to develop an engaging user interface that can be easily navigated in various search engines. It also allows server-side rendering, which is also helpful to boost the SEO of your app.

React allows reusable components.

React web applications are made up of multiple components where each component has its logic and controls. These components provide a small, reusable piece of HTML code as an output that can be reused wherever you need them. The code reusability helps developers to make their apps easier to develop and maintain. It also makes the nesting of the components easy and allows developers to build complex applications of simple building blocks. The reuse of components also increases the pace of development.

Support of handy tools

React provides a lot of handy tools that can make the task of the developers understandable and easier. Use these tools in Chrome and Firefox dev extension, allowing us to inspect the React component hierarchies in the virtual DOM. It also allows us to select the particular components and examine and edit their current props and state.

React has a rich set of libraries.

React has a huge ecosystem of libraries and provides you the freedom to choose the tools, libraries, and architecture for developing the best application based on your requirement.

Scope for testing the codes

React web applications are easy to test. These applications provide a scope where the developer can test and debug their codes with the help of native tools.

For More Information, ***Click here***.

4) What are the biggest limitations of React?

Following is the list of the biggest limitations of React:

- React is just a library. It is not a complete framework.
- It has a huge library which takes time to understand.
- It may be difficult for the new programmers to understand and code.
- React uses inline templating and JSX, which may be difficult and act as a barrier. It also makes the coding complex.

---

5) What is JSX?

JSX stands for JavaScript XML. It is a React extension which allows writing JavaScript code that looks similar to HTML. It makes HTML file easy to understand. The JSX file makes the React application robust and boosts its performance. JSX provides you to write XML-like syntax in the same file where you write JavaScript code, and then preprocessor (i.e., transpilers like Babel) transform these expressions into actual JavaScript code. Just like XML/HTML, JSX tags have a tag name, attributes, and children.

Example

```
class App extends React.Component {
  render() {
    return(
      <div>
        <h1>Hello JavaTpoint</h1>
      </div>
    )
  }
}
```

In the above example, text inside <h1> tag return as JavaScript function to the render function. After compilation, the JSX expression becomes a normal JavaScript function, as shown below.

```
React.createElement("h1", null, "Hello JavaTpoint");
```

For More Information, ***Click here***.

6) Why can't browsers read JSX?

Browsers cannot read JSX directly because they can only understand JavaScript objects, and JSX is not a regular JavaScript object. Thus, we need to transform the JSX file into a JavaScript object using transpilers like Babel and then pass it to the browser.

7) Why we use JSX?

- It is faster than regular JavaScript because it performs optimization while translating the code to JavaScript.
- Instead of separating technologies by putting markup and logic in separate files, React uses components that contain both.
- t is type-safe, and most of the errors can be found at compilation time.
- It makes easier to create templates.

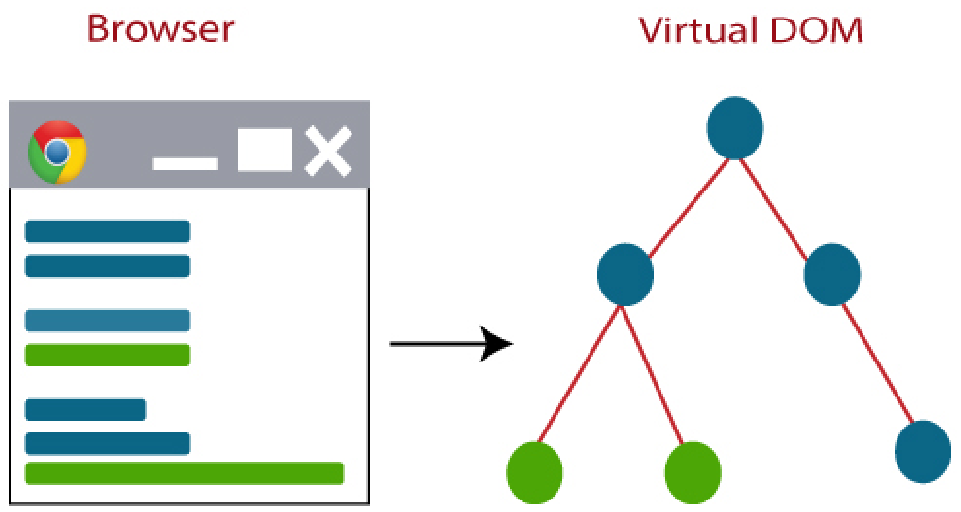
8) What do you understand by Virtual DOM?

A Virtual DOM is a lightweight JavaScript object which is an in-memory representation of real DOM. It is an intermediary step between the render function being called and the displaying of elements on the screen. It is similar to a node tree which lists the elements, their attributes, and content as objects and their properties. The render function creates a node tree of the React components and then updates this node tree in response to the mutations in the data model caused by various actions done by the user or by the system.

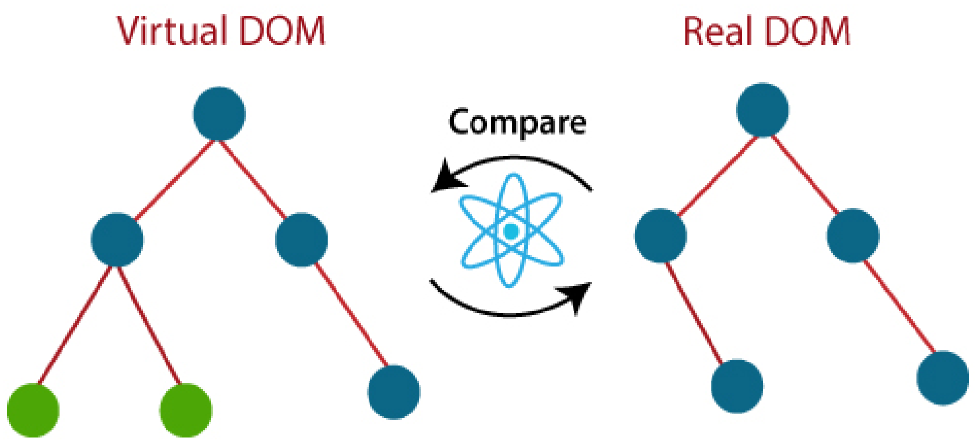
9) Explain the working of Virtual DOM.

Virtual DOM works in three steps:

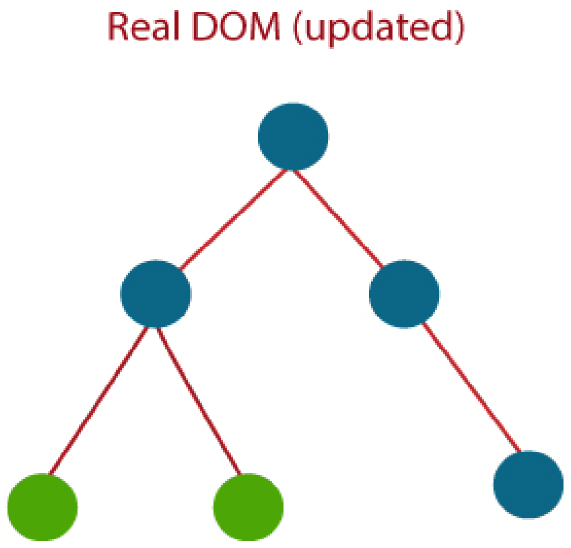
1. Whenever any data changes in the React App, the entire UI is re-rendered in Virtual DOM representation.



2. Now, the difference between the previous DOM representation and the new DOM is calculated.



3. Once the calculations are completed, the real DOM updated with only those things which are changed.



10) How is React different from Angular?

The React is different from Angular in the following ways.

	Angular	React
Author	Google	Facebook Community
Developer	Misko Hevery	Jordan Walke

Initial Release	October 2010	March 2013
Language	JavaScript, HTML	JSX
Type	Open Source MVC Framework	Open Source JS Framework
Rendering	Client-Side	Server-Side
Data-Binding	Bi-directional	Uni-directional
DOM	Regular DOM	Virtual DOM
Testing	Unit and Integration Testing	Unit Testing
App Architecture	MVC	Flux
Performance	Slow	Fast, due to virtual DOM.

For More Information, ***Click here***.

11) How React's ES6 syntax is different from ES5 syntax?

The React's ES6 syntax has changed from ES5 syntax in the following aspects.

require vs. Import

```
// ES5
var React = require('react');

// ES6
import React from 'react';
```

exports vs. export

```
// ES5
module.exports = Component;

// ES6
```

```
export default Component;
```

component and function

```
// ES5
var MyComponent = React.createClass({
   render: function() {
      return(
        <h3>Hello JavaTpoint</h3>
      );
   }
});

// ES6
class MyComponent extends React.Component {
   render() {
      return(
        <h3>Hello Javatpoint</h3>
      );
   }
}
```

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>
      );
   }
}
```

state

```
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>
      );
   }
}
```

12) What is the difference between ReactJS and React Native?

The main differences between ReactJS and React Native are given below.

SN	ReactJS	React Native
1.	Initial release in 2013.	Initial release in 2015.
2.	It is used for developing web applications.	It is used for developing mobile applications.
3.	It can be executed on all platforms.	It is not platform independent. It takes more effort to be executed on all platforms.
4.	It uses a JavaScript library and CSS for animations.	It comes with built-in animation libraries.
5.	It uses React-router for navigating web pages.	It has built-in Navigator library for navigating mobile applications.
6.	It uses HTML tags.	It does not use HTML tags.
7.	In this, the Virtual DOM renders the browser code.	In this, Native uses its API to render code for mobile applications.

For More Information, ***Click here***.

13) What is the difference between Real DOM and Virtual DOM?

The following table specifies the key differences between the Real DOM and Virtual DOM:

The real DOM creates a new DOM if the element updates.

Real DOM	Virtual DOM
The real DOM updates slower.	The virtual DOM updates faster.

The real DOM can directly update HTML.	The virtual DOM cannot directly update HTML.
The virtual DOM updates the JSX if the element updates.	
In real DOM, DOM manipulation is very expensive.	In virtual DOM, DOM manipulation is very easy.
There is a lot of memory wastage in The real DOM.	There is no memory wastage in the virtual DOM.

React Component Interview Questions

14) What do you understand from "In React, everything is a component."

In React, components are the building blocks of React applications. These components divide the entire React application's UI into small, independent, and reusable pieces of code. React renders each of these components independently without affecting the rest of the application UI. Hence, we can say that, in React, everything is a component.

15) Explain the purpose of render() in React.

It is mandatory for each React component to have a render() function. Render function is used to return the HTML which you want to display in a component. If you need to rendered more than one HTML element, you need to grouped together inside single enclosing tag (parent tag) such as <div>, <form>, <group> etc. This function returns the same result each time it is invoked.

Example: If you need to display a heading, you can do this as below.

```
import React from 'react'

class App extends React.Component {
   render (){
      return (
        <h1>Hello World</h1>
      )
    }
}
export default App
```

Points to Note:

- Each render() function contains a return statement.
- The return statement can have only one parent HTML tag.

16) How can you embed two or more components into one?

You can embed two or more components into the following way:

```
import React from 'react'

class App extends React.Component {
    render (){
      return (
        <h1>Hello World</h1>
      )
    }
}

class Example extends React.Component {
    render (){
      return (
        <h1>Hello JavaTpoint</h1>
      )
    }
}
export default App
```

17) What is Props?

Props stand for "Properties" in React. They are read-only inputs to components. Props are an object which stores the value of attributes of a tag and work similar to the HTML attributes. It gives a way to pass data from the parent to the child components throughout the application.

It is similar to function arguments and passed to the component in the same way as arguments passed in a function.

Props are immutable so we cannot modify the props from inside the component. Inside the components, we can add attributes called props. These attributes are available in the component as this.props and can be used to render dynamic data in our render method.

For More Information, ***Click here***.

---

18) What is a State in React?

The State is an updatable structure which holds the data and information about the component. It may be changed over the lifetime of the component in response to user action or system event. It is the heart of the react component which determines the behavior of the component and how it will render. It must be kept as simple as possible.

Let's create a "User" component with "message state."

```
import React from 'react'

class User extends React.Component {
  constructor(props) {
    super(props)

    this.state = {
      message: 'Welcome to JavaTpoint'
    }
  }

  render() {
    return (
      <div>
        <h1>{this.state.message}</h1>
      </div>
    )
  }
}
export default User
```

For More Information, ***Click here***.

19) Differentiate between States and Props.

The major differences between States and Props are given below.

SN	Props	State
1.	Props are read-only.	State changes can be asynchronous.
2.	Props are immutable.	State is mutable.
3.	Props allow you to pass data from one component to other components as an argument.	State holds information about the components.
4.	Props can be accessed by the child component.	State cannot be accessed by child components.

5.	Props are used to communicate between components.	States can be used for rendering dynamic changes with the component.
6.	The stateless component can have Props.	The stateless components cannot have State.
7.	Props make components reusable.	The State cannot make components reusable.
8.	Props are external and controlled by whatever renders the component.	The State is internal and controlled by the component itself.

For More Information, ***Click here***.

20) How can you update the State of a component?

We can update the State of a component using this.setState() method. This method does not always replace the State immediately. Instead, it only adds changes to the original State. It is a primary method which is used to update the user interface(UI) in response to event handlers and server responses.

Example

```
import React, { Component } from 'react';
import PropTypes from 'prop-types';

class App extends React.Component {
  constructor() {
    super();
    this.state = {
       msg: "Welcome to JavaTpoint"
    };
    this.updateSetState = this.updateSetState.bind(this);
  }
  updateSetState() {
     this.setState({
       msg:"Its a best ReactJS tutorial"
```

```
        });
    }
    render() {
        return (
            <div>
                <h1>{this.state.msg}</h1>
                <button onClick = {this.updateSetState}>SET STATE</button>
            </div>
        );
    }
}
export default App;
```

For More Information, ***Click here***.

21) Differentiate between stateless and stateful components.

The difference between stateless and stateful components are:

SN	Stateless Component	Stateful Component
1.	The stateless components do not hold or manage state.	The stateful components can hold or manage state.
2.	It does not contain the knowledge of past, current, and possible future state changes.	It can contain the knowledge of past, current, and possible future changes in state.
3.	It is also known as a functional component.	It is also known as a class component.
4.	It is simple and easy to understand.	It is complex as compared to the stateless component.
5.	It does not work with any lifecycle method of React.	It can work with all lifecycle method of React.
6.	The stateless components cannot be reused.	The stateful components can be reused.

22) What is arrow function in React? How is it used?

The Arrow function is the new feature of the ES6 standard. If you need to use arrow functions, it is not necessary to bind any event to 'this.' Here, the scope of 'this' is global and not limited to any calling function. So If you are using Arrow Function, there is no need to bind 'this' inside the constructor. It is also called 'fat arrow '(=>) functions.

```
//General way
render() {
   return(
       <MyInput onChange={this.handleChange.bind(this) } />
   );
}
//With Arrow Function
render() {
   return(
       <MyInput onChange={ (e) => this.handleOnChange(e) } />
   );
}
```

23) What is an event in React?

An event is an action which triggers as a result of the user action or system generated event like a mouse click, loading of a web page, pressing a key, window resizes, etc. In React, the event handling system is very similar to handling events in DOM elements. The React event handling system is known as Synthetic Event, which is a cross-browser wrapper of the browser's native event.

Handling events with React have some syntactical differences, which are:

- React events are named as camelCase instead of lowercase.
- With JSX, a function is passed as the event handler instead of a string.

For More Information, ***Click here***.

24) How do you create an event in React?

We can create an event as follows.

```
class Display extends React.Component({
    show(msgEvent) {
        // code
    },
    render() {
        // Here, we render the div with an onClick prop
        return (
          <div onClick={this.show}>Click Me</div>
        );
    }
});
```

Example

```
import React, { Component } from 'react';
class App extends React.Component {
    constructor(props) {
        super(props);
        this.state = {
            companyName: ''
        };
    }
    changeText(event) {
        this.setState({
            companyName: event.target.value
        });
    }
    render() {
        return (
            <div>
            <h2>Simple Event Example</h2>
            <label htmlFor="name">Enter company name: </label>
            <input type="text" id="companyName" onChange={this.changeText.bind(this)}/>
            <h4>You entered: { this.state.companyName }</h4>
          </div>
      );
   }
}
export default App;
```

For More Information, ***Click here***.

25) What are synthetic events in React?

A synthetic event is an object which acts as a cross-browser wrapper around the browser's native event. It combines the behavior of different browser's native event into one API, including stopPropagation() and preventDefault().

In the given example, e is a Synthetic event.

```
function ActionLink() {
   function handleClick(e) {
      e.preventDefault();
      console.log('You had clicked a Link.');
   }
   return (
      <a href="#" onClick={handleClick}>
            Click_Me
      </a>
   );
}
```

26) what is the difference between controlled and uncontrolled components?

The difference between controlled and uncontrolled components are:

SN	Controlled	Uncontrolled
1.	It does not maintain its internal state.	It maintains its internal states.
2.	Here, data is controlled by the parent component.	Here, data is controlled by the DOM itself.
3.	It accepts its current value as a prop.	It uses a ref for their current values.
4.	It allows validation control.	It does not allow validation control.
5.	It has better control over the form elements and data.	It has limited control over the form elements and data.

For More Information, ***Click here***.

27) Explain the Lists in React.

Lists are used to display data in an ordered format. In React, Lists can be created in a similar way as we create it in JavaScript. We can traverse the elements of the list using the map() function.

Example

```
import React from 'react';
import ReactDOM from 'react-dom';

function NameList(props) {
  const myLists = props.myLists;
  const listItems = myLists.map((myList) =>
    <li>{myList}</li>
  );
  return (
    <div>
       <h2>Rendering Lists inside component</h2>
```

```
            <ul>{listItems}</ul>
    </div>
  );
}
const myLists = ['Peter', 'Sachin', 'Kevin', 'Dhoni', 'Alisa'];
ReactDOM.render(
  <NameList myLists={myLists} />,
  document.getElementById('app')
);
export default App;
```

For More Information, ***Click here***.

28) What is the significance of keys in React?

A key is a unique identifier. In React, it is used to identify which items have changed, updated, or deleted from the Lists. It is useful when we dynamically created components or when the users alter the lists. It also helps to determine which components in a collection needs to be re-rendered instead of re-rendering the entire set of components every time. It increases application performance.

For More Information, ***Click here***.

29) How are forms created in React?

Forms allow the users to interact with the application as well as gather information from the users. Forms can perform many tasks such as user authentication, adding user, searching, filtering, etc. A form can contain text fields, buttons, checkbox, radio button, etc.

React offers a stateful, reactive approach to build a form. The forms in React are similar to HTML forms. But in React, the state property of the component is only updated via setState(), and a JavaScript function handles their submission. This function has full access to the data which is entered by the user into a form.

```
import React, { Component } from 'react';

class App extends React.Component {
  constructor(props) {
    super(props);
    this.state = {value: ''};
    this.handleChange = this.handleChange.bind(this);
    this.handleSubmit = this.handleSubmit.bind(this);
  }
  handleChange(event) {
    this.setState({value: event.target.value});
  }
  handleSubmit(event) {
    alert('You have submitted the input successfully: ' + this.state.value);
    event.preventDefault();
  }
  render() {
    return (
        <form onSubmit={this.handleSubmit}>
          <h1>Controlled Form Example</h1>
          <label>
              Name:
              <input type="text" value={this.state.value} onChange={this.handleChange} />
          </label>
          <input type="submit" value="Submit" />
       </form>
    );
  }
}
export default App;
```

For More Information, ***Click here***.

30) What are the different phases of React component's lifecycle?

The different phases of React component's lifecycle are:

Initial Phase: It is the birth phase of the React lifecycle when the component starts its journey on a way to the DOM. In this phase, a component contains the default Props and initial State. These default properties are done in the constructor of a component.

Mounting Phase: In this phase, the instance of a component is created and added into the DOM.

Updating Phase: It is the next phase of the React lifecycle. In this phase, we get new Props and change State. This phase can potentially update and re-render only when a prop or state change occurs. The main aim of this phase is to ensure that the component is displaying the latest version of itself. This phase repeats again and again.

Unmounting Phase: It is the final phase of the React lifecycle, where the component instance is destroyed and unmounted(removed) from the DOM.

For More Information, ***Click here***.

31) Explain the lifecycle methods of React components in detail.

The important React lifecycle methods are:

- **getInitialState():** It is used to specify the default value of this.state. It is executed before the creation of the component.
- **componentWillMount():** It is executed before a component gets rendered into the DOM.
- **componentDidMount():** It is executed when the component gets rendered and placed on the DOM. Now, you can do any DOM querying operations.
- **componentWillReceiveProps():** It is invoked when a component receives new props from the parent class and before another render is called. If you want to update the State in response to prop changes, you should compare this.props and nextProps to perform State transition by using this.setState() method.
- **shouldComponentUpdate():** It is invoked when a component decides any changes/updation to the DOM and returns true or false value based on certain conditions. If this method returns true, the component will update. Otherwise, the component will skip the updating.
- **componentWillUpdate():** It is invoked before rendering takes place in the DOM. Here, you can't change the component State by invoking this.setState() method. It will not be called, if

shouldComponentUpdate() returns false.

- **componentDidUpdate():** It is invoked immediately after rendering takes place. In this method, you can put any code inside this which you want to execute once the updating occurs.
- **componentWillUnmount():** It is invoked immediately before a component is destroyed and unmounted permanently. It is used to clear up the memory spaces such as invalidating timers, event listener, canceling network requests, or cleaning up DOM elements. If a component instance is unmounted, you cannot mount it again.

For More Information, ***Click here***.

32) What are Pure Components?

Pure components introduced in React 15.3 version. The React.Component and React.PureComponent differ in the shouldComponentUpdate() React lifecycle method. This method decides the re-rendering of the component by returning a boolean value (true or false). In React.Component, shouldComponentUpdate() method returns true by default. But in React.PureComponent, it compares the changes in state or props to re-render the component. The pure component enhances the simplicity of the code and performance of the application.

33) What are Higher Order Components(HOC)?

In React, Higher Order Component is an advanced technique for reusing component logic. It is a function that takes a component and returns a new component. In other words, it is a function which accepts another function as an argument. According to the official website, it is not the feature(part) in React API, but a pattern that emerges from React's compositional nature.

For More Information, ***Click here***.

34) What can you do with HOC?

You can do many tasks with HOC, some of them are given below:

- Code Reusability
- Props manipulation
- State manipulation

- Render highjacking

35) What is the difference between Element and Component?

The main differences between Elements and Components are:

SN	Element	Component
1.	An element is a plain JavaScript object which describes the component state and DOM node, and its desired properties.	A component is the core building block of React application. It is a class or function which accepts an input and returns a React element.
2.	It only holds information about the component type, its properties, and any child elements inside it.	It can contain state and props and has access to the React lifecycle methods.
3.	It is immutable.	It is mutable.
4.	We cannot apply any methods on elements.	We can apply methods on components.

5.	**Example:**
	const element = React.createElement(
	'div',
	{id: 'login-btn'},
	'Login'
	)
		Example:
		function Button ({ onLogin }) {
		return React.createElement(
		'div',
		{id: 'login-btn', onClick: onLogin},
		'Login'
		)
		}

36) How to write comments in React?

In React, we can write comments as we write comments in JavaScript. It can be in two ways:

1. Single Line Comments: We can write comments as /* Block Comments */ with curly braces:

```
{/* Single Line comment */}
```

2. Multiline Comments: If we want to comment more that one line, we can do this as

```
{ /*
   Multi
   line
   comment
*/ }
```

37) Why is it necessary to start component names with a capital letter?

In React, it is necessary to start component names with a capital letter. If we start the component name with lower case, it will throw an error as an unrecognized tag. It is because, in JSX, lower case tag names are considered as HTML tags.

38) What are fragments?

In was introduced in React 16.2 version. In React, Fragments are used for components to return multiple elements. It allows you to group a list of multiple children without adding an extra node to the DOM.

Example

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

There is also a shorthand syntax exists for declaring Fragments, but it's not supported in many tools:

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

For More Information, ***Click here***.

39) Why are fragments better than container divs?

- Fragments are faster and consume less memory because it did not create an extra DOM node.
- Some CSS styling like CSS Grid and Flexbox have a special parent-child relationship and add <div> tags in the middle, which makes it hard to keep the desired layout.
- The DOM Inspector is less cluttered.

40) How to apply validation on props in React?

Props validation is a tool which helps the developers to avoid future bugs and problems. It makes your code more readable. React components used special property PropTypes that help you to catch bugs by validating data types of values passed through props, although it is not necessary to define components with propTypes.

We can apply validation on props using App.propTypes in React component. When some of the props are passed with an invalid type, you will get the warnings on JavaScript console. After specifying the validation patterns, you need to set the App.defaultProps.

```
class App extends React.Component {
          render() {}
}
Component.propTypes = { /*Definition */};
```

For More Information, ***Click here***.

41) What is create-react-app?

Create React App is a tool introduced by Facebook to build React applications. It provides you to create single-page React applications. The create-react-app are preconfigured, which saves you from time-consuming setup and configuration like Webpack or Babel. You need to run a single command to start the React project, which is given below.

```
$ npx create-react-app my-app
```

This command includes everything which we need to build a React app. Some of them are given below:

- It includes React, JSX, ES6, and Flow syntax support.
- It includes Autoprefixed CSS, so you don't need -webkit- or other prefixes.
- It includes a fast, interactive unit test runner with built-in support for coverage reporting.
- It includes a live development server that warns about common mistakes.
- It includes a build script to bundle JS, CSS, and images for production, with hashes and source maps.

For More Information, ***Click here***.

42) How can you create a component in React?

There are two possible ways to create a component in React:

Function Components: This is the simplest way to create a component in React. These are the pure JavaScript functions that accept props object as the first parameter and return React elements:

```
function Greeting({ message }) {
  return <h1>{`Hello, ${message}`}</h1>
}
```

Class Components: The class components method facilitates you to use ES6 class to define a component. The above function component can be written as:

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

43) When do we prefer to use a class component over a function component?

If a component needs state or lifecycle methods, we should use the class component; otherwise, use the function component. However, after React 16.8, with the addition of Hooks, you could use state, lifecycle methods, and other features that were only available in the class component right in your function component.

44) Is it possible for a web browser to read JSX directly?

Web browsers can't read JSX directly. This is because the web browsers are built to read the regular JS objects only, and JSX is not a regular JavaScript object.

If you want a web browser to read a JSX file, you must transform the files into a regular JavaScript object. For this purpose, Babel is used.

45) What do you understand by the state in React?

In react, the state of a component is an object that holds some information that may change over the component's lifetime. It would be best to try to make your state as simple as possible and minimize the number of stateful components.

Let's see how to create a user component with message state:

```
class User extends React.Component {
  constructor(props) {
    super(props)
    this.state = {
      message: 'Welcome to React world'
    }
  }
  render() {
    return (
      <div>
        <h1>{this.state.message}</h1>
      </div>
    )
  }
}
```

The state is very similar to props, but it is private and fully controlled by the component. i.e., It is not accessible to any other component till the owner component decides to pass it.

46) What are the main changes that appear in React's ES6 syntax compared to ES5 syntax?/How different is React's ES6 syntax compared to ES5?

Following are the most visible syntax we can see while comparing ES6 and ES5:

require vs import

Syntax in ES5:

```
var React = require('react');
```

Syntax in ES6:

```
import React from 'react';
```

export vs exports

Syntax in ES5:

```
module.exports = Component;
```

Syntax in ES6:

```
export default Component;
```

component and function

Syntax in ES5:

```
var MyComponent = React.createClass({
    render: function() {
        return
<h3>Hello JavaTpoint!</h3>
;
    }
});
```

Syntax in ES6:

```
class MyComponent extends React.Component {
    render() {
        return
<h3>Hello JavaTpoint!</h3>
;
    }
}
```

props

Syntax in ES5:

```
var App = React.createClass({
    propTypes: { name: React.PropTypes.string },
    render: function() {
        return
<h3>Hello, {this.props.name}!</h3>
;
    }
});
```

Syntax in ES6:

```
class App extends React.Component {
    render() {
        return
<h3>Hello, {this.props.name}!</h3>
;
    }
}
```

state

Syntax in ES5:

```
var App = React.createClass({
    getInitialState: function() {
        return { name: 'world' };
    },
    render: function() {
        return
<h3>Hello, {this.state.name}!</h3>
;
    }
});
```

Syntax in ES6:

```
class App extends React.Component {
    constructor() {
        super();
        this.state = { name: 'world' };
    }
    render() {
        return
<h3>Hello, {this.state.name}!</h3>
;
    }
}
```

47) What do you understand by props in React?

In React, the props are inputs to components. They are single values or objects containing a set of values passed to components on creation using a naming convention similar to HTML-tag attributes. They are data passed down from a parent component to a child component.

The main purpose of props in React is to provide the following component functionality:

- Pass custom data to your component.
- Trigger state changes.
- Use via this.props.reactProp inside component's render() method.

For example, let us create an element with reactProp property:

```
<Element reactProp={'1'} />
```

This reactProp name becomes a property attached to React's native props object, which already exists on all React library components.

```
props.reactProp
```

React Refs Interview Questions

48) What do you understand by refs in React?

Refs is the shorthand used for references in React. It is an attribute which helps to store a reference to particular DOM nodes or React elements. It provides a way to access React DOM nodes or React elements and how to interact with it. It is used when we want to change the value of a child component, without making the use of props.

For More Information, ***Click here***.

49) How to create refs?

Refs can be created by using React.createRef() and attached to React elements via the ref attribute. It is commonly assigned to an instance property when a component is created, and then can be referenced throughout the component.

```
class MyComponent extends React.Component {
  constructor(props) {
    super(props);
    this.callRef = React.createRef();
  }
  render() {
    return <div ref={this.callRef} />;
  }
}
```

50) What are Forward Refs?

Ref forwarding is a feature which is used for passing a ref through a component to one of its child components. It can be performed by making use of the React.forwardRef() method. It is particularly useful with higher-order components and specially used in reusable component libraries.

Example

```
import React, { Component } from 'react';
import { render } from 'react-dom';

const TextInput = React.forwardRef((props, ref) => (
  <input type="text" placeholder="Hello World" ref={ref} />
));

const inputRef = React.createRef();

class CustomTextInput extends React.Component {
  handleSubmit = e => {
    e.preventDefault();
    console.log(inputRef.current.value);
  };
  render() {
    return (
      <div>
        <form onSubmit={e => this.handleSubmit(e)}>
          <TextInput ref={inputRef} />
          <button>Submit</button>
        </form>
      </div>
    );
  }
}
export default App;
```

For More Information, ***Click here***.

51) Which is the preferred option callback refs or findDOMNode()?

The preferred option is to use callback refs over findDOMNode() API. Because callback refs give better control when the refs are set and unset whereas findDOMNode() prevents certain improvements in React in the future.

```
class MyComponent extends Component {
  componentDidMount() {
    findDOMNode(this).scrollIntoView()
  }
  render() {
    return <div />
  }
}
```

The recommended approach is:

```
class MyComponent extends Component {
  componentDidMount() {
    this.node.scrollIntoView()
  }
  render() {
    return <div ref={node => this.node = node} />
  }
}
class MyComponent extends Component {
  componentDidMount() {
    this.node.scrollIntoView()
  }
  render() {
    return <div ref={node => this.node = node} />
  }
}
```

52) What is the use of Refs?

The Ref in React is used in the following cases:

- It is used to return a reference to the element.
- It is used when we need DOM measurements such as managing focus, text selection, or media playback.
- It is used in triggering imperative animations.
- It is used when integrating with third-party DOM libraries.
- It can also use as in callbacks.

For More Information, ***Click here***.

React Router Interview Questions

53) What is React Router?

React Router is a standard routing library system built on top of the React. It is used to create Routing in the React application using React Router Package. It helps you to define multiple routes in the app. It provides the synchronous URL on the browser with data that will be displayed on the web page. It maintains the standard structure and behavior of the application and mainly used for developing single page web applications.

For More Information, ***Click here***.

54) Why do we need a Router in React?

React Router plays an important role to display multiple views in a single page application. It is used to define multiple routes in the app. When a user types a specific URL into the browser, and if this URL path matches any 'route' inside the router file, the user will be redirected to that particular Route. So, we need to add a Router library to the React app, which allows creating multiple routes with each leading to us a unique view.

```
<switch>
    <h1>React Router Example</h1>
    <Route path="/" component={Home} />
    <Route path="/about" component={About} />
    <Route path="/contact" component={Contact} />
</switch>
```

55) List down the advantages of React Router.

The important advantages of React Router are given below:

- In this, it is not necessary to set the browser history manually.
- Link uses to navigate the internal links in the application. It is similar to the anchor tag.
- It uses Switch feature for rendering.
- The Router needs only a Single Child element.
- In this, every component is specified in <Route>.
- The packages are split into three packages, which are Web, Native, and Core. It supports the compact size of the React application.

56) How is React Router different from Conventional Routing?

The difference between React Routing and Conventional Routing are:

SN	Conventional Routing	React Routing
1.	In Conventional Routing, each view contains a new file.	In React Routing, there is only a single HTML page involved.
2.	The HTTP request is sent to a server to receive the corresponding HTML page.	Only the History attribute <BrowserRouter> is changed.
3.	In this, the user navigates across different pages for each view.	In this, the user is thinking he is navigating across different pages, but its an illusion only.

57) Why you get "Router may have only one child element" warning?

It is because you have not to wrap your Route's in a <Switch> block or <div> block which renders a route exclusively.

Example

```
render((
  <Router>
    <Route {/* ... */} />
    <Route {/* ... */} />
  </Router>
)
```

should be

```
render(
  <Router>
    <Switch>
      <Route {/* ... */} />
      <Route {/* ... */} />
    </Switch>
  </Router>
)
```

58) Why switch keyword used in React Router v4?

The 'switch' keyword is used to display only a single Route to rendered amongst the several defined Routes. The <Switch> component is used to render components only when the path will be matched. Otherwise, it returns to the not found component.

React Styling Interview Questions

59) How to use styles in React?

We can use style attribute for styling in React applications, which adds dynamically-computed styles at render time. It accepts a JavaScript object in camelCased properties rather than a CSS string. The style attribute is consistent with accessing the properties on DOM nodes in JavaScript.

Example

```
const divStyle = {
  color: 'blue',
  backgroundImage: 'url(' + imgUrl + ')'
};

function HelloWorldComponent() {
  return <div style={divStyle}>Hello World!</div>
}
```

60) How many ways can we style the React Component?

We can style React Component in mainly four ways, which are given below:

- Inline Styling
- CSS Stylesheet
- CSS Module
- Styled Components

For More Information, ***Click here***.

61) Explain CSS Module styling in React.

CSS Module is a CSS file where all class names and animation names are scoped locally by default. It is available only for the component which imports it, and without your permission, it cannot be applied to any other Components. You can create CSS Module file with the .module.css extension.

For More Information, ***Click here***.

62) What are Styled Components?

Styled-Components is a library for React. It is the successor of CSS Modules. It uses enhance CSS for styling React component systems in your application, which is written with a mixture of JavaScript and CSS. It is scoped to a single component and cannot leak to any other element in the page.

The styled-components provides:

- Automatic critical CSS
- No class name bugs
- Easier deletion of CSS
- Simple dynamic styling
- Painless maintenance

For More Information, ***Click here***.

63) What are hooks in React?

Hooks are the new feature introduced in React 16.8 version that facilitates us to use state and other React features without writing a class.

See the following example of useState hook:

```
import { useState } from 'react';
function Example() {
  // Declare a new state variable, which we'll call "count"
  const [count, setCount] = useState(0);
  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
       Click on this button
      </button>
    </div>
  );
}
```

64) What are the rules you should follow for the hooks in React?

We have to follow the following two rules to use hooks in React:

- You should call hooks only at the top level of your React functions and not inside the loops, conditions, or nested functions. This is used to ensure that hooks are called in the same order each time a component renders, and it also preserves the state of hooks between multiple useState and useEffect calls.
- You should call hooks from React functions only. Don't call hooks from regular JavaScript functions.

65) What are forms in React?

In React, forms are used to enable users to interact with web applications. Following is a list of the most common usage of forms in React:

- Forms facilitate users to interact with the application. By using forms, the users can communicate with the application and enter the required information whenever required.
- Forms contain certain elements, such as text fields, buttons, checkboxes, radio buttons, etc., that can make the application more interactive and beautiful.
- Forms are the best possible way to take inputs from the users.
- Forms are used for many different tasks such as user authentication, searching, filtering, indexing, etc.

66) What is an error boundary or error boundaries?

An error boundary is a concept introduced in version 16 of React. Error boundaries provide a way to find out the errors that occur in the render phase. Any component which uses one of the following lifecycle methods is considered an error boundary. Let's see the places where an error boundary can detect an error:

- Render phase
- Inside a lifecycle method
- Inside the constructor

Let's see an example to understand it better:

Without using error boundaries:

```
class CounterComponent extends React.Component{
 constructor(props){
  super(props);
  this.state = {
   counterValue: 0
  }
  this.incrementCounter = this.incrementCounter.bind(this);
 }
 incrementCounter(){
  this.setState(prevState => counterValue = prevState+1);
 }
 render(){
  if(this.state.counter === 2){
   throw new Error('Crashed');
  }
  return(
   <div>
    <button onClick={this.incrementCounter}>Increment Value</button>
    <p>Value of counter: {this.state.counterValue}</p>
   </div>
  )
 }
}
```

In the above code, you can see that when the counterValue equals 2, it throws an error inside the render method. We know that any error inside the render method leads to unmounting of the component so, to display an error that occurs inside the render method, we use error boundaries. When we are not using the error boundary, we see a blank page instead of seeing an error.

With error boundaries:

We have specified earlier that error boundary is a component using one or both of the following methods:

- static getDerivedStateFromError
- componentDidCatch

See the following code where we create an error boundary to handle errors in render phase:

```
class ErrorBoundary extends React.Component {
 constructor(props) {
  super(props);
  this.state = { hasError: false };
 }
 static getDerivedStateFromError(error) {
  return { hasError: true };
 }
  componentDidCatch(error, errorInfo) {
  logErrorToMyService(error, errorInfo);
 }
 render() {
  if (this.state.hasError) {
    return <h4>Something went wrong</h4>
  }
  return this.props.children;
 }
}
```

You can see in the above code the getDerivedStateFromError function renders the fallback UI interface when the render method has an error.

The componentDidCatch logs the error information to an error tracking service.

Now with error boundary, we can render the CounterComponent in the following way:

```
<ErrorBoundary>
  <CounterComponent/>
</ErrorBoundary>
```

67) In which cases do error boundaries not catch errors?

Following are some cases in which error boundaries don't catch errors:

- Error boundaries don't catch errors inside the event handlers.
- During the server-side rendering.
- In the case when errors are thrown in the error boundary code itself.
- Asynchronous code using setTimeout or requestAnimationFrame callbacks.

React Redux Interview Questions

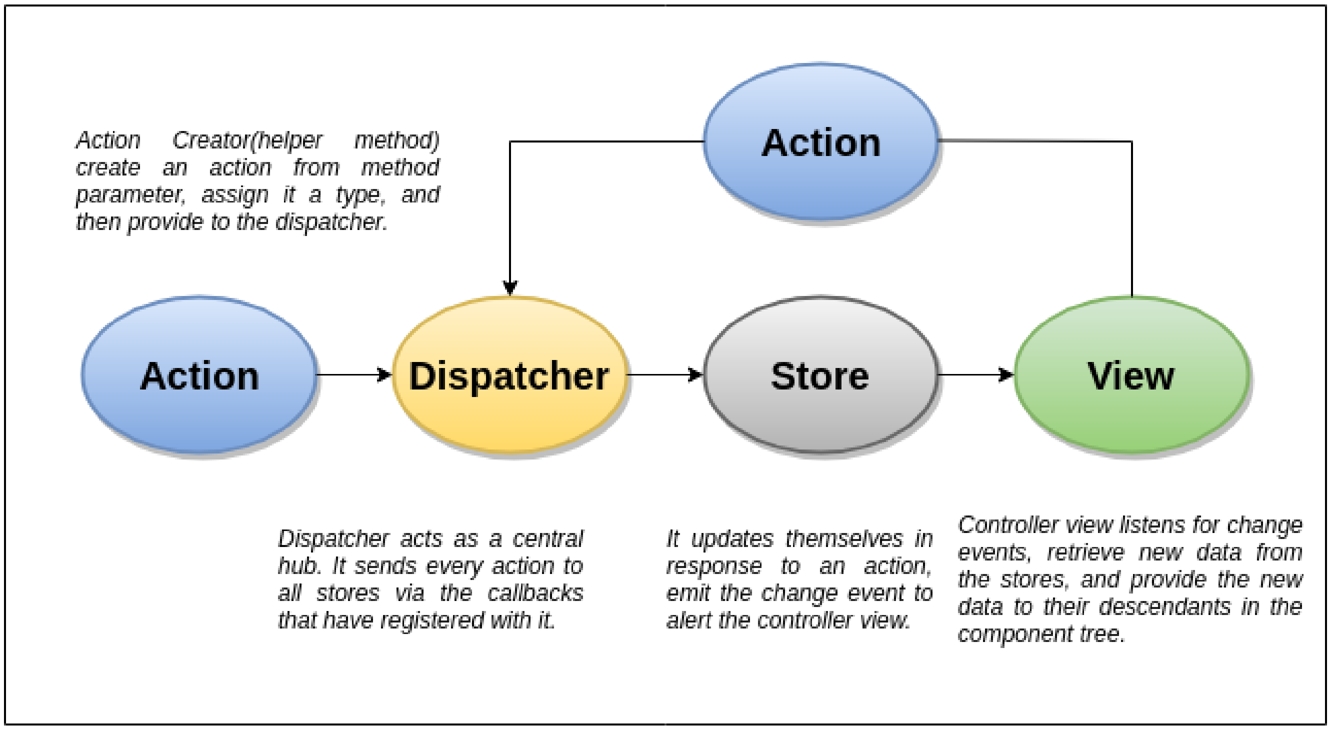
68) What were the major problems with MVC framework?

The major problems with the MVC framework are:

- DOM manipulation was very expensive.
- It makes the application slow and inefficient.
- There was a huge memory wastage.
- It makes the application debugging hard.

69) Explain the Flux concept.

Flux is an application architecture that Facebook uses internally for building the client-side web application with React. It is neither a library nor a framework. It is a kind of architecture that complements React as view and follows the concept of Unidirectional Data Flow model. It is useful when the project has dynamic data, and we need to keep the data updated in an effective manner.


Action Creator(helper method) create an action from method parameter, assign it a type, and then provide to the dispatcher.
Action
Action
Dispatcher
Store
View
Dispatcher acts as a central hub. It sends every action to all stores via the callbacks that have registered with it.
It updates themselves in response to an action, emit the change event to alert the controller view.
Controller view listens for change events, retrieve new data from the stores, and provide the new data to their descendants in the component tree.

For More Information, ***Click here***.

70) What is Redux?

Redux is an open-source JavaScript library used to manage application state. React uses Redux for building the user interface. The Redux application is easy to test and can run in different environments showing consistent behavior. It was first introduced by Dan Abramov and Andrew Clark in 2015.

React Redux is the official React binding for Redux. It allows React components to read data from a Redux Store, and dispatch Actions to the Store to update data. Redux helps apps to scale by providing a sensible way to manage state through a unidirectional data flow model. React Redux is conceptually simple. It subscribes to the Redux store, checks to see if the data which your component wants have changed, and re-renders your component.

For More Information, ***Click here***.

71) What are the three principles that Redux follows?

The three principles that redux follows are:

1. **Single source of truth:** The State of your entire application is stored in an object/state tree inside a single Store. The single State tree makes it easier to keep changes over time. It also makes it easier to debug or inspect the application.
2. **The State is read-only:** There is only one way to change the State is to emit an action, an object describing what happened. This principle ensures that neither the views nor the network callbacks can write directly to the State.
3. **Changes are made with pure functions:** To specify how actions transform the state tree, you need to write reducers (pure functions). Pure functions take the previous State and Action as a parameter and return a new State.

72) List down the components of Redux.

The components of Redux are given below.

- **STORE:** A Store is a place where the entire State of your application lists. It is like a brain responsible for all moving parts in Redux.
- **ACTION:** It is an object which describes what happened.
- **REDUCER:** It determines how the State will change.

For More Information, ***Click here***.

73) Explain the role of Reducer.

Reducers read the payloads from the actions and then updates the Store via the State accordingly. It is a pure function which returns a new state from the initial State. It returns the previous State as it is if no work needs to be done.

74) What is the significance of Store in Redux?

A Store is an object which holds the application's State and provides methods to access the State, dispatch Actions and register listeners via subscribe(listener). The entire State tree of an application is saved in a single Store which makes the Redux simple and predictable. We can pass middleware to the Store which handles the processing of data as well as keep a log of various actions that change the Store's State. All the Actions return a new state via reducers.

75) How is Redux different from Flux?

The Redux is different from Flux in the following manner.

SN	Redux	Flux
1.	Redux is an open-source JavaScript library used to manage application State.	Flux is neither a library nor a framework. It is a kind of architecture that complements React as view and follows the concept of Unidirectional Data Flow model.
2.	Store's State is immutable.	Store's State is mutable.
3.	In this, Store and change logic are separate.	In this, the Store contains State and change logic.
4.	It has only a single Store.	It can have multiple Store.
5.	Redux does not have Dispatcher concept.	It has single Dispatcher, and all actions pass through that Dispatcher.

76) What are the advantages of Redux?

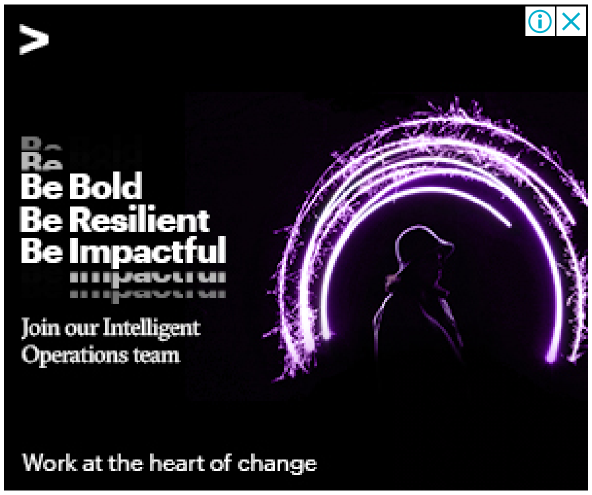
The main advantages of React Redux are:

- React Redux is the official UI bindings for react Application. It is kept up-to-date with any API changes to ensure that your React components behave as expected.
- It encourages good 'React' architecture.
- It implements many performance optimizations internally, which allows to components re-render only when it actually needs.
- It makes the code maintenance easy.
- Redux's code written as functions which are small, pure, and isolated, which makes the code testable and independent.

77) How to access the Redux store outside a component?

You need to export the Store from the module where it created with createStore() method. Also, you need to assure that it will not pollute the global window space.

```
store = createStore(myReducer)
export default store
```


Be Bold
Be Resilient
Be Impactful
Join our Intelligent Operations team
Work at the heart of change

Some Most Frequently Asked React MCQ

1) What is Babel in React?

a. Babel is a transpiler.

b. Babel is an interpreter.

c. Babel is a compiler.

d. Babel is both a compiler and a transpiler.

Show Answer **Workspace**

2) What do you understand by the Reconciliation process in React?

a. The Reconciliation process is a process through which React updates the DOM.

b. The Reconciliation process is a process through which React deletes the DOM.