Part1 General React – React Interview Questions

1. Differentiate between Real DOM and Virtual DOM.

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

2. What is React?

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

3. What are the features of React?

Major features of React are listed below:

i. It uses the **virtual DOM** instead of the real DOM.
ii. It uses **server-side rendering**.
iii. It follows **uni-directional data flow** or data binding.

4. List some of the major advantages of React.

Some of the major advantages of React are:

i. It increases the application's performance
ii. It can be conveniently used on the client as well as server side
iii. Because of JSX, code's readability increases
iv. React is easy to integrate with other frameworks like Meteor, Angular, etc
v. Using React, writing UI test cases become extremely easy

5. What are the limitations of React?

Limitations of React are listed below:

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

6. What is JSX?

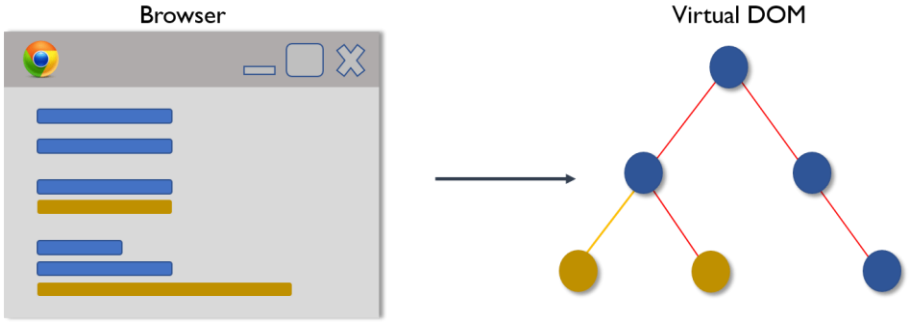
JSX is a shorthand for JavaScript XML. This is a type of file used by React which utilizes the expressiveness of JavaScript along with HTML like template syntax. This makes the HTML file really easy to understand. This file makes applications robust and boosts its performance. Below is an example of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

```
render(){
return(

  <div>

<h1> Hello World from Edureka!!</h1>

  </div>

  );
}
```

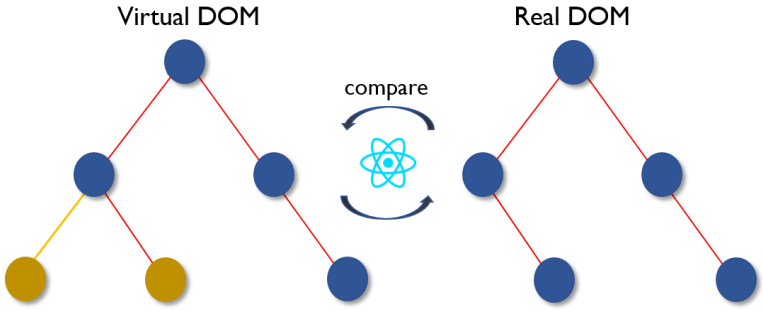
7. What do you understand by Virtual DOM? Explain its working.

A virtual DOM is a lightweight JavaScript object which originally is just the copy of the real DOM. It is a node tree that lists the elements, their attributes and content as Objects and their properties. React's render function creates a node tree out of the React components. It then updates this tree in response to the mutations in the data model which is caused by various actions done by the user or by the system. This Virtual DOM works in three simple steps.

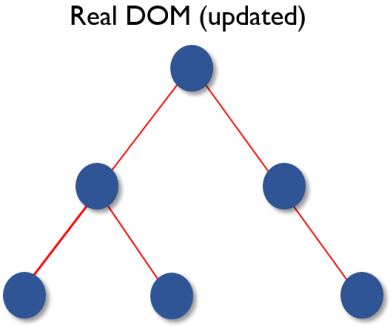
1. Whenever any underlying data changes, the entire UI is re-rendered in Virtual DOM representation.



2. Then the difference between the previous DOM representation and the new one is calculated.



3. Once the calculations are done, the real DOM will be updated with only the things that have actually changed.



8. Why can't browsers read JSX?

Browsers can only read JavaScript objects but JSX in not a regular JavaScript object. Thus to enable a browser to read JSX, first, we need to transform JSX file into a JavaScript object using JSX transformers like Babel and then pass it to the browser.

9. How different is React's ES6 syntax when compared to ES5?

Syntax has changed from ES5 to ES6 in following aspects:

i. require vs import

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

ii. export vs exports

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

iii. component and function

```
1  // ES5
2  var MyComponent = React.createClass({
3  render: function() {
4  return
5
6  <h3>Hello Edureka!</h3>
7  ;
8  }
9  });
10
```

```
// ES6
class MyComponent extends React.Component {
  render() {
    return

<h3>Hello Edureka!</h3>
;
  }
}
```

iv. props

```
// ES5
var App = React.createClass({
propTypes: { name: React.PropTypes.string },
  render: function() {
    return

<h3>Hello, {this.props.name}!</h3>
;
  }
});

// ES6
class App extends React.Component {
  render() {
    return

<h3>Hello, {this.props.name}!</h3>
;
  }
}
```

v. state

```
// ES5
var App = React.createClass({
  getInitialState: function() {
    return { name: 'world' };
  },
  render: function() {
    return

<h3>Hello, {this.state.name}!</h3>
;
  }
});

// ES6
class App extends React.Component {
```

```
16                                  constructor() {
17                                      super();
18                                  this.state = { name: 'world' };
19                                      }
20                                  render() {
21                                      return
22
23                                  <h3>Hello, {this.state.name}!</h3>
24                                      ;
25                                      }
26                                  }
```

10. How is React different from Angular?

React vs Angular		
TOPIC	**REACT**	**ANGULAR**
1. ARCHITECTURE	Only the View of MVC	Complete MVC
2. RENDERING	Server-side rendering	Client-side rendering
3. DOM	Uses virtual DOM	Uses real DOM
4. DATA BINDING	One-way data binding	Two-way data binding
5. DEBUGGING	Compile time debugging	Runtime debugging
6. AUTHOR	Facebook	Google

React Components – React Interview Questions

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

Components are the building blocks of a React application's UI. These components split up the entire UI into small independent and reusable pieces. Then it renders each of these components independent of each other without affecting the rest of the UI.

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

Each React component must have a **render()** mandatorily. It returns a single React element which is the representation of the native DOM component. If more than one HTML element needs to be rendered, then they must be grouped together inside one enclosing tag such as **<form>, <group>, ** etc. This function must be kept pure i.e., it must return the same result each time it is invoked.

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

We can embed components into one in the following way:

```
1                                   class MyComponent extends React.Component{
2                                       render(){
3                                       return(
4
5                                       <div>
6
7                                   <h1>Hello</h1>
8
```

```
 9                                                    <Header/>
10                                                    </div>
11
12                                                     );
13                                                    }
14                                                   }
15                                          class Header extends React.Component{
16                                                  render(){
17                                                    return
18
19                                                <h1>Header Component</h1>
20
21                                                    };
22                                                   }
23                                                ReactDOM.render(
24                                          <MyComponent/>, document.getElementById('content')
25                                                    );
```

14. What is Props?

Props is the shorthand for Properties in React. They are read-only components which must be kept pure i.e. immutable. They are always passed down from the parent to the child components throughout the application. A child component can never send a prop back to the parent component. This help in maintaining the unidirectional data flow and are generally used to render the dynamically generated data.

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

States are the heart of React components. States are the source of data and must be kept as simple as possible. Basically, states are the objects which determine components rendering and behavior. They are mutable unlike the props and create dynamic and interactive components. They are accessed via **this.state().**

16. Differentiate between states and props.

States vs Props

Conditions	State	Props
1. Receive initial value from parent component	Yes	Yes
2. Parent component can change value	No	Yes
3. Set default values inside component	Yes	Yes
4. Changes inside component	Yes	No
5. Set initial value for child components	Yes	Yes
6. Changes inside child components	No	Yes

17. How can you update the state of a component?

State of a component can be updated using this.setState().

```
1                                          class MyComponent extends React.Component {
2                                                  constructor() {
```

```
3                                                   super();
4                                                  this.state = {
5                                                  name: 'Maxx',
6                                                    id: '101'
7                                                     }
8                                                    }
9                                                  render()
10                                                   {
11                 setTimeout(()=>{this.setState({name:'Jaeha', id:'222'})},2000)
12                                                 return (
13
14                                                    <div>
15
16                                              <h1>Hello {this.state.name}</h1>
17
18                                              <h2>Your Id is {this.state.id}</h2>
19
20                                                    </div>
21
22                                                    );
23                                                   }
24                                                  }
25                                               ReactDOM.render(
26                 <MyComponent/>, document.getElementById('content')
27                                                  );
```

18. What is arrow function in React? How is it used?

Arrow functions are more of brief syntax for writing the function expression. They are also called *'fat arrow'* (**=>**) the functions. These functions allow to bind the context of the components properly since in ES6 auto binding is not available by default. Arrow functions are mostly useful while working with the higher order functions.

```
1                                                  //General way
2                                                  render() {
3                                                    return(
4                  <MyInput onChange={this.handleChange.bind(this) } />
5                                                    );
6                                                   }
7                                                //With Arrow Function
8                                                  render() {
9                                                    return(
10                 <MyInput onChange={ (e) => this.handleOnChange(e) } />
11                                                   );
12                                                  }
```

19. Differentiate between stateful and stateless components.

Stateful vs Stateless

Stateful Component	Stateless Component
1. Stores info about component's state change in memory	1. Calculates the internal state of the components
2. Have authority to change state	2. Do not have the authority to change state

3. Contains the knowledge of past, current and possible future changes in state	3. Contains no knowledge of past, current and possible future state changes
4. Stateless components notify them about the requirement of the state change, then they send down the props to them.	4. They receive the props from the Stateful components and treat them as callback functions.

20. What are the different phases of React component's lifecycle?

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

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

21. Explain the lifecycle methods of React components in detail.

Some of the most important lifecycle methods are:

i. ***componentWillMount()*** – Executed just before rendering takes place both on the client as well as server-side.
ii. ***componentDidMount()*** – Executed on the client side only after the first render.
iii. ***componentWillReceiveProps()*** – Invoked as soon as the props are received from the parent class and before another render is called.
iv. ***shouldComponentUpdate()*** – Returns true or false value based on certain conditions. If you want your component to update, return **true** else return **false**. By default, it returns true.
v. ***componentWillUpdate()*** – Called just before rendering takes place in the DOM.
vi. ***componentDidUpdate()*** – Called immediately after rendering takes place.
vii. ***componentWillUnmount()*** – Called after the component is unmounted from the DOM. It is used to clear up the memory spaces.

22. What is an event in React?

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

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

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

23. How do you create an event in React?

```
1  class Display extends React.Component({
2    show(evt) {
3      // code
```

```
},
render() {
// Render the div with an onClick prop (value is a function)
return (

<div onClick={this.show}>Click Me!</div>

);
}
});
```

24. What are synthetic events in React?

Synthetic events are the objects which act as a cross-browser wrapper around the browser's native event. They combine the behavior of different browsers into one API. This is done to make sure that the events show consistent properties across different browsers.

25. What do you understand by refs in React?

Refs is the short hand for References in React. It is an attribute which helps to store a reference to a particular React element or component, which will be returned by the components render configuration function. It is used to return references to a particular element or component returned by render(). They come in handy when we need DOM measurements or to add methods to the components.

```
class ReferenceDemo extends React.Component{
display() {
const name = this.inputDemo.value;
document.getElementById('disp').innerHTML = name;
}
render() {
return(

<div>
Name: <input type="text" ref={input => this.inputDemo = input} />
<button name="Click" onClick={this.display}>Click</button>

<h2>Hello <span id="disp"></span> !!!</h2>

</div>
);
}
}
```

26. List some of the cases when you should use Refs.

Following are the cases when refs should be used:

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

27. How do you modularize code in React?

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

```
//ChildComponent.jsx
export default class ChildComponent extends React.Component {
render() {
return(

<div>

<h1>This is a child component</h1>

</div>

);
}
}

//ParentComponent.jsx
import ChildComponent from './childcomponent.js';
class ParentComponent extends React.Component {
render() {
return(

<div>
<App />
</div>

);
}
}
```

28. How are forms created in React?

React forms are similar to HTML forms. But in React, the state is contained in the state property of the component and is only updated via setState(). Thus the elements can't directly update their state and their submission is handled by a JavaScript function. This function has full access to the data that is entered by the user into a form.

```
handleSubmit(event) {
alert('A name was submitted: ' + this.state.value);
event.preventDefault();
}

render() {
return (

<form onSubmit={this.handleSubmit}>
<label>
Name:
<input type="text" value={this.state.value} onChange={this.handleSubmit} />
</label>
<input type="submit" value="Submit" />
</form>
```

```
16
17                                                    );
18                                                  }
```

29. What do you know about controlled and uncontrolled components?

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

30. What are Higher Order Components(HOC)?

Higher Order Component is an advanced way of reusing the component logic. Basically, it's a pattern that is derived from React's compositional nature. HOC are custom components which wrap another component within it. They can accept any dynamically provided child component but they won't modify or copy any behavior from their input components. You can say that HOC are 'pure' components.

31. What can you do with HOC?

HOC can be used for many tasks like:

- Code reuse, logic and bootstrap abstraction
- Render High jacking
- State abstraction and manipulation
- Props manipulation

32. What are Pure Components?

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

33. What is the significance of keys in React?

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

React Redux – React Interview Questions

34. What were the major problems with MVC framework?

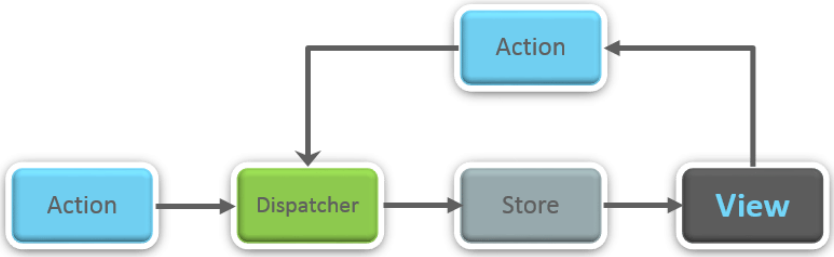
Following are some of the major problems with MVC framework:

- DOM manipulation was very expensive
- Applications were slow and inefficient
- There was huge memory wastage

- Because of circular dependencies, a complicated model was created around models and views

35. Explain Flux.

Flux is an architectural pattern which enforces the uni-directional data flow. It controls derived data and enables communication between multiple components using a central Store which has authority for all data. Any update in data throughout the application must occur here only. Flux provides stability to the application and reduces run-time errors.



36. What is Redux?

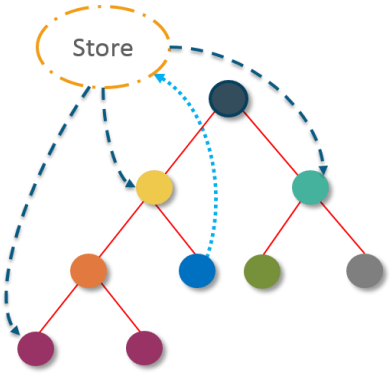
Redux is one of the hottest libraries for front-end development in today's marketplace. It is a predictable state container for JavaScript applications and is used for the entire applications state management. Applications developed with Redux are easy to test and can run in different environments showing consistent behavior.

37. What are the three principles that Redux follows?

i. ***Single source of truth:*** The state of the entire application is stored in an object/ state tree within a single store. The single state tree makes it easier to keep track of changes over time and debug or inspect the application.

ii. ***State is read-only:*** The only way to change the state is to trigger an action. An action is a plain JS object describing the change. Just like state is the minimal representation of data, the action is the minimal representation of the change to that data.

iii. ***Changes are made with pure functions:*** In order to specify how the state tree is transformed by actions, you need pure functions. Pure functions are those whose return value depends solely on the values of their arguments.



38. What do you understand by "Single source of truth"?

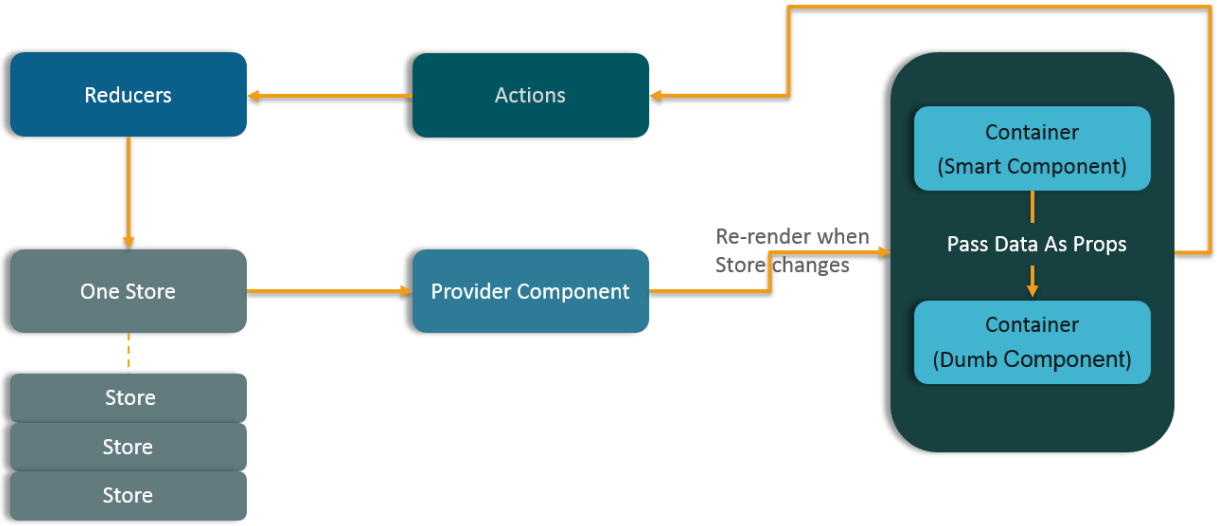
Redux uses 'Store' for storing the application's entire state at one place. So all the component's state are stored in the Store and they receive updates from the Store itself. The single state tree makes it easier to keep track of changes over time and debug or inspect the application.

39. List down the components of Redux.

Redux is composed of the following components:

i. **Action** – It's an object that describes what happened.
ii. **Reducer** – It is a place to determine how the state will change.
iii. **Store** – State/ Object tree of the entire application is saved in the Store.
iv. **View** – Simply displays the data provided by the Store.

40. Show how the data flows through Redux?



41. How are Actions defined in Redux?

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

```
1  function addTodo(text) {
2    return {
3      type: ADD_TODO,
```

```
4                                                    text
5                                                  }
6                                                }
```

42. Explain the role of Reducer.

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

43. What is the significance of Store in Redux?

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

44. How is Redux different from Flux?

Flux vs Redux

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

45. What are the advantages of Redux?

Advantages of Redux are listed below:

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

React Router – React Interview Questions

46. What is React Router?

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

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

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

48. Why do we need a Router in React?

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

```
1    <switch>
2    <route exact path='/' component={Home}/>
3    <route path='/posts/:id' component={Newpost}/>
4    <route path='/posts'   component={Post}/>
5    </switch>
```

49. List down the advantages of React Router.

Few advantages are:

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

50. How is React Router different from conventional routing?

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