FULL STACK DEVELOPMENT

Unit-3

Imp Q & A

1) What is react is? its components and its features?

Ans) ReactJS is a JavaScript library designed for crafting dynamic and interactive applications, elevating UI/UX for web and mobile platforms. Operating as an open-source, component-based front-end library, React is dedicated to UI design and streamlines code debugging by employing a component-oriented approach.

We will discuss about the following featured of React:

- JSX (JavaScript Syntax Extension):
- Virtual DOM:
- One-way Data Binding:
- Performance:
- Extension:
- Conditional Statements:
- Components:
- Simplicity:

Let's understand each of them in detail:

1. JSX (JavaScript Syntax Extension):

JSX is a combination of HTML and JavaScript. You can embed JavaScript objects inside the HTML elements. JSX is not supported by the browsers, as a result, Babel compiler transcompile the code into JavaScript code. JSX makes codes easy and understandable. It is easy to learn if you know HTML and JavaScript.

```
const name="GeekforGeeks";
const ele = <h1>Welcome to {name}</h1>;
```

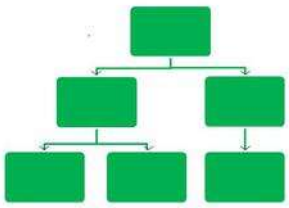
2. Virtual DOM:

DOM stands for Document Object Model. It is the most important part of the web as it divides into modules and executes the code. Usually, JavaScript Frameworks updates the whole DOM at once, which makes the web application slow. But react uses virtual DOM which is an exact copy of real DOM. Whenever there is a modification in the web application, the whole virtual DOM is updated first and finds the difference between real DOM and Virtual DOM.

In the above-shown figure, when the whole virtual DOM has updated there is a change in the child components. So, now DOM finds the difference and updates only the changed part.

3. One-way Data Binding:

One-way data binding, the name itself says that it is a one-direction flow. The data in react flows only in one direction i.e. the data is transferred from top to bottom i.e. from parent components to child components. The properties(props) in the child component cannot return the data to its parent component but it can have communication with the parent components to modify the states according to the provided inputs.



One-way Data Binding

As shown in the above diagram, data can flow only from top to bottom.

4. Performance:

As we discussed earlier, react uses virtual DOM and updates only the modified parts. So , this makes the DOM to run faster. DOM executes in memory so we can create separate components which makes the DOM run faster.

5. Extension:

React has many extensions that we can use to create full-fledged UI applications. It supports mobile app development and provides server-side rendering. React is extended with Flux, Redux, React Native, etc. which helps us to create good-looking UI.

6. Conditional Statements:

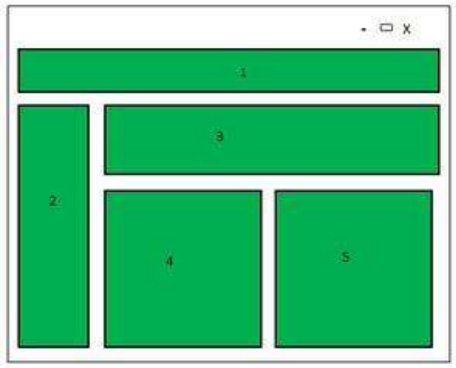
JSX allows us to write conditional statements. The data in the browser is displayed according to the conditions provided inside the JSX.

Syntax:

```
const age = 12;
if (age >= 10)
{
   <p> Greater than { age } </p>;
}
else
{
   <p> { age } </p>;
}
```

7. Components:

React.js divides the web page into multiple components as it is component-based. Each component is a part of the UI design which has its own logic and design as shown in the below image. So the component logic which is written in JavaScript makes it easy and run faster and can be reusable.



Multiple components

8. Simplicity:

React.js is a component-based which makes the code reusable and React.js uses JSX which is a combination of HTML and JavaScript. This makes code easy to understand and easy to debug and has less code.

Steps to Create React Application:

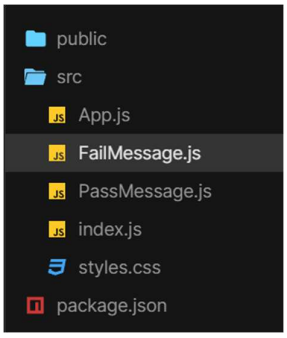
Step 1: Create a react application by using the following command:

npx create-react-app foldername

Step 2: Change your directory to the newly created folder.

cd foldername

Project Structure:



2) What is React Router how it's very useful while creating application?

Ans) React Router is a standard library for creating dynamic routes and navigation in React JS Applications. It allows you to manage navigation in your app by defining routes that connect the URL paths to specific components.

With React Router, you can implement different views for different parts of your application without the need for a full-page refresh. This is a key feature of single-page applications (SPAs), where only the necessary content is updated as the user navigates.

The current latest verstion is **React router dom v6.**

Features of React Router

- **Declarative Routing:** React Router v6 uses the Routes and Route components to define routes declaratively, making the routing configuration simple and easy to read.
- **Nested Routes:** It supports nested routes, allowing for complex and hierarchical routing structures, which helps in organizing the application better.
- **Programmatic Navigation:** The useNavigate hook enables programmatic navigation, allowing developers to navigate between routes based on certain conditions or user actions.
- **Route Parameters:** It provides dynamic routing with route parameters, enabling the creation of routes that can match multiple URL patterns.
- **Improved TypeScript Support:** Enhanced TypeScript support ensures that developers can build type-safe applications, improving development efficiency and reducing errors.

Components of React Router

React Router mainly comprises of the below components

1. BrowserRouter and HashRouter

- **BrowserRouter**: Uses the HTML5 history API to keep your UI in sync with the URL.
- **HashRouter**: Uses the hash portion of the URL (i.e., window.location.hash) to keep your UI in sync with the URL.

```
<BrowserRouter>
  (/* Your routes go here */}
</BrowserRouter>
```

2. Routes and Route

- **Routes**: A container for all your route definitions.
- **Route**: Defines a single route with a path and the component to render.

```
<Routes>
  <Route path="/" element={<Home />} />
  <Route path="/about" element={<About />} />
</Routes>
```

3. Link and NavLink

- **Link**: Creates navigational links in your application.
- **NavLink**: Similar to Link but provides additional styling attributes when the link is active.

```
<nav>
  <NavLink to="/" activeClassName="active">Home</NavLink>
  <Link to="/about">About</Link>
</nav>
```

Steps to Create Routes using React Router

Step 1: Initialize React Project

Create React application using the following command.

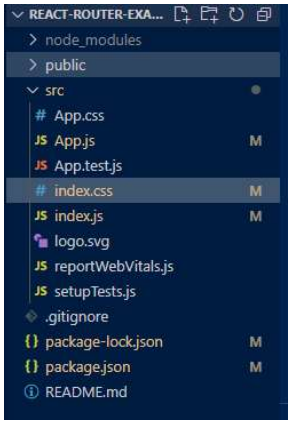
npx create-react-app react-router-example
cd react-router-example

Step 2: Install React Router

Install react-router in your application write the following command in your terminal

npm install react-router-dom@6

Project Structure



3) Write a React Form simple code?

Ans) Steps

1. **Create a Form Component (Form.js)**
2. **Handle Input Change (useState)**
3. **Display Submitted Data**

1. Simple Form Code (Form.js)

```
import React, { useState } from "react";


const Form = () => {
 const [data, setData] = useState({ name: "", email: "" });


 const handleChange = (e) => {
  setData({ ...data, [e.target.name]: e.target.value });
 };


 const handleSubmit = (e) => {
  e.preventDefault();
  alert(`Name: ${data.name}\nEmail: ${data.email}`);
 };
```

```
 return (
   <form onSubmit={handleSubmit}>
     <input type="text" name="name" placeholder="Name" onChange={handleChange} />
     <input type="email" name="email" placeholder="Email" onChange={handleChange} />
     <button type="submit">Submit</button>
   </form>
 );
};

export default Form;
```

2. Use in App.js

```
import React from "react";
import Form from "./Form";

const App = () => <Form />;

export default App;
```

3. Run the Application

```
npm start
```

4) What is redux? why it having more advantages?

Ans) What is Redux?

Redux is a **state management library** used with **React** to store and manage global state efficiently. It helps avoid **prop drilling** and makes data flow **predictable**.

Why Use Redux? (Advantages)

1 Centralized State → One place for managing app data.
2 Predictable Updates → Uses **actions & reducers** to update state.
3 Easier Debugging → Track state changes with **Redux DevTools**.
4 No Prop Drilling → Components can directly access the **Redux Store**.
5 Better Performance → Minimizes unnecessary re-renders.

How Redux Works?

Redux has 3 main parts:
✓ **Store** → Holds global state.

✓ **Actions** → Objects that describe *what happened*.
✓ **Reducers** → Functions that update the state based on actions.

Simple Redux Example (Counter App)

1 Install Redux

```
npm install redux react-redux
```

2 Create Store (store.js)

```
import { createStore } from "redux";


const reducer = (state = { count: 0 }, action) => {
  return action.type === "INCREMENT" ? { count: state.count + 1 } : state;
};


const store = createStore(reducer);
export default store;
```

3 Provide Store (index.js)

```
import { Provider } from "react-redux";
import store from "./store";
<Provider store={store}><App /></Provider>;
```

4 Use Redux in Component (Counter.js)

```
import { useSelector, useDispatch } from "react-redux";


const Counter = () => {
  const count = useSelector(state => state.count);
  const dispatch = useDispatch();


  return (
    <div>
      <h1>{count}</h1>
      <button onClick={() => dispatch({ type: "INCREMENT" })}>+</button>
    </div>
  );
};
```

```
export default Counter;
```

5) What is use Effect, use State explain along with class Level Component and function Level component?

Ans) useState & useEffect in React

1 useState → Manages Component State

- Used to create and update state in **functional components**.
- **Replaces this.state in class components.**

Class Component (Before Hooks)

```
class Counter extends React.Component {
  constructor() {
    super();
    this.state = { count: 0 };
  }
  render() {
    return (
      <div>
        <h1>{this.state.count}</h1>
        <button onClick={() => this.setState({ count: this.state.count + 1 })}>
          +
        </button>
      </div>
    );
  }
}
```

Functional Component (With useState)

```
import React, { useState } from "react";

const Counter = () => {
  const [count, setCount] = useState(0);

  return (
```

```
    <div>
      <h1>{count}</h1>
      <button onClick={() => setCount(count + 1)}>+</button>
    </div>
  );
};
```

2 useEffect → Handles Side Effects

- Used for **API calls, event listeners, timers, etc.**
- **Replaces componentDidMount, componentDidUpdate, componentWillUnmount.**

Class Component (Before Hooks)

```
class Timer extends React.Component {
  constructor() {
    super();
    this.state = { time: new Date().toLocaleTimeString() };
  }

  componentDidMount() {
    this.interval = setInterval(() => {
      this.setState({ time: new Date().toLocaleTimeString() });
    }, 1000);
  }

  componentWillUnmount() {
    clearInterval(this.interval);
  }

  render() {
    return <h1>{this.state.time}</h1>;
  }
}
```

Functional Component (With useEffect)

```
import React, { useState, useEffect } from "react";
```

```
const Timer = () => {
  const [time, setTime] = useState(new Date().toLocaleTimeString());

  useEffect(() => {
    const interval = setInterval(() => {
      setTime(new Date().toLocaleTimeString());
    }, 1000);

    return () => clearInterval(interval); // Cleanup
  }, []);

  return <h1>{time}</h1>;
};
```

Summary (Class vs. Functional Components)

Feature	Class Component	Function Component
State	this.state	useState()
Update State	this.setState()	setState()
Side Effects	componentDidMount()	useEffect()
Code Complexity	More Boilerplate	Simpler & Cleaner

Why Use Hooks?

✓ Less code & better readability
✓ No need for class components
✓ Easier to manage lifecycle events