React Hooks

Hooks are the new feature introduced in the React 16.8 version. It allows you to use state and other React features without writing a class. Hooks are the functions which "hook into" React state and lifecycle features from function components. It does not work inside classes.

Hooks are backward-compatible, which means it does not contain any breaking changes. Also, it does not replace your knowledge of React concepts.

When to use a Hooks

If you write a function component, and then we want to add some state to it, previously you do this by converting it to a class. But, now you can do it by using a Hook inside the existing function component.

Rules of Hooks

Hooks are similar to JavaScript functions, but you need to follow these two rules when using them. Hooks rule ensures that all the stateful logic in a component is visible in its source code.

1. Only call Hooks at the top level

Do not call Hooks inside loops, conditions, or nested functions. Hooks should always be used at the top level of the React functions. This rule ensures that Hooks are called in the same order each time a component renders.

2. Only call Hooks from React functions

You cannot call Hooks from regular JavaScript functions. Instead, you can call Hooks from React function components. Hooks can also be called from custom Hooks.

useState()

Hook state is the new way of declaring a state in React app. Hook uses useState() functional component for setting and retrieving state.

```
1. import React, { useState } from 'react';

function CountApp() {
```

```
  // 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 me
      </button>
    </div>
  );
}
export default CountApp;
```

useState is the Hook which needs to call inside a function component to add some local state to it. The useState returns a pair where the first element is the current state value/initial value, and the second one is a function which allows us to update it. After that, we will call this function from an event handler or somewhere else. The useState is similar to this.setState in class.

useEffect()

The Effect Hook allows us to perform side effects (an action) in the function components. It does not use components lifecycle methods which are available in class components. In other words, Effects Hooks are equivalent to componentDidMount(), componentDidUpdate(), and componentWillUnmount() lifecycle methods.

Side effects have common features which the most web applications need to perform, such as

- Updating the DOM,
- Fetching and consuming data from a server API,
- Setting up a subscription, etc.

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

function CounterExample() {
```

```
  const [count, setCount] = useState(0);

  // Similar to componentDidMount and componentDidUpdate:
  useEffect(() => {
    // Update the document title using the browser API
    document.title = `You clicked ${count} times`;
  });

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
export default CounterExample;
```

useRef ()

The useRef is a hook that allows to directly create a reference to the DOM element in the functional component.The useRef hook is a new addition in React 16.8. To learn useRef the user should be aware about refs in React. Unlike useState if we change a value in useRef it will not re-render the webpage.

Reasons to use useRef hook

The main use of useRef hook is to access the DOM elements in a more efficient way as compared to simple refs. Since useRef hooks preserve value across various re-renders and do not cause re-renders whenever a value is changed they make the application faster and helps in caching and storing previous values

Structure of useRef hook

It accepts only one initial value

Syntax:

const refContainer = useRef(initialValue);

The useRef returns a mutable ref object. This object has a property called .current. The value is persisted in the refContainer.current property. These values are accessed from the current property of the returned object. The .current property could be initialised to the passed argument initialValue e.g. useRef(initialValue). The object can persist a value for a full lifetime of the component.

```
import React, {Fragment, useRef} from 'react';


function App() {


  // Creating a ref object using useRef hook

  const focusPoint = useRef(null);

  const onClickHandler = () => {

    focusPoint.current.value =

      "The quick brown fox jumps over the lazy dog";

      focusPoint.current.focus();

  };

  return (

    <Fragment>
```

```
      <div>
        <button onClick={onClickHandler}>
         ACTION
        </button>
      </div>
      <label>
       Click on the action button to
       focus and populate the text.
      </label><br/>
      <textarea ref={focusPoint} />
    </Fragment>
  );
};

export default App;
```

useReducer()

The `useReducer` Hook is similar to the `useState` Hook.

It allows for custom state logic.

If you find yourself keeping track of multiple pieces of state that rely on complex logic, `useReducer` may be useful.

Syntax:

```
const [state, dispatch] = useReducer(reducer, initialArgs, init);
```

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

// Defining the initial state and the reducer
const initialState = 0;
const reducer = (state, action) => {
  switch (action) {
    case "add":
      return state + 1;
    case "subtract":
      return state - 1;
    case "reset":
      return 0;
    default:
      throw new Error("Unexpected action");
  }
};

const App = () => {
    // Initialising useReducer hook
  const [count, dispatch] = useReducer(reducer, initialState);
```

```
  return (
    <div>
      <h2>{count}</h2>
      <button onClick={() => dispatch("add")}>
        add
      </button>
      <button onClick={() => dispatch("subtract")}>
        subtract
      </button>
      <button onClick={() => dispatch("reset")}>
        reset
      </button>
    </div>
  );
};


export default App;
```

useMemo()

The React `useMemo` Hook returns a memoized value.

Think of memoization as caching a value so that it does not need to be recalculated.

The `useMemo` Hook only runs when one of its dependencies update.

This can improve performance.

Note- follow class code for reference

React useContext( )

Context provides a way to pass data or state through the component tree without having to pass props down manually through each nested component. It is designed to share data that can be considered as global data for a tree of React components, such as the current authenticated user or theme(e.g. color, paddings, margins, font-sizes).

Context API uses Context. Provider and Context. Consumer Components pass down the data but it is very cumbersome to write the long functional code to use this Context API. So useContext hook helps to make the code more readable, less verbose and removes the need to introduce Consumer Component. The useContext hook is the new addition in React 16.8.

Syntax:

const authContext = useContext(initialValue);

The useContext accepts the value provided by React.createContext and then re-render the component whenever its value changes but you can still optimize its performance by using memoization.

Note- follow class code for reference

React Router

Routing is a process in which a user is directed to different pages based on their action or request. ReactJS Router is mainly used for developing Single Page Web Applications. React Router is used to define multiple routes in the application. 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.

React Router is a standard library system built on top of the React and used to create routing in the React application using React Router Package. 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.

Need of React Router

React Router plays an important role to display multiple views in a single page application. Without React Router, it is not possible to display multiple views in React applications. Most of the social media websites like Facebook, Instagram uses React Router for rendering multiple views.

React Router Installation

-npm install react-router-dom@6

Components in React Router

BrowserRouter: BrowserRouter is a router implementation that uses the HTML5 history API(pushState, replaceState, and the popstate event) to keep your UI in sync with the URL. It is the parent component that is used to store all of the other components.

Routes: It's a new component introduced in the v6 and an upgrade of the component. The main advantages of Routes over Switch are:

Relative s and s

Routes are chosen based on the best match instead of being traversed in order.

Route: Route is the conditionally shown component that renders some UI when its path matches the current URL.

Link: The link component is used to create links to different routes and implement navigation around the application. It works like an HTML anchor tag.

What is Route?

It is used to define and render component based on the specified path. It will accept components and render to define what should be rendered.

What is < Link> component?

This component is used to create links which allow to **navigate** on different **URLs** and render its content without reloading the webpage.

Now, we need to add some **styles** to the Link. So that when we click on any particular link, it can be easily **identified** which Link is **active**. To do this react router provides a new trick **NavLink** instead of **Link**. Now, in the index.js file, replace Link from Navlink and add properties **activeStyle**. The activeStyle properties mean when we click on the Link, it should have a specific style so that we can differentiate which one is currently active.