**React state management**

State management is simply a way to engender communication and sharing of data across components. It creates a concrete data structure to represent your app's State that you can read and write.

State is a JavaScript object that represents the part of a component that can change based on a resultant action of a user. You could also say states are simply the memory of a component.  
When a user performs an action in a typical React app, changes occur in the component's state. While this isn't bad, it quickly becomes a problem if the app begins to scale; hence, such an app's complexity makes it extremely difficult to keep track of all dependencies.

State management gets messy as the app grows bigger. That is why you need a state management tool like Redux, Recoil, making it easier to maintain these states.

**Functional components**

Functional components are some of the more common components that will come across while working in React. These are simply JavaScript functions. We can create a functional component to React by writing a JavaScript function. These functions may or may not receive data as parameters and, return a React element (JSX).

**Hook in react**

**Functional components** lack a significant amount of features as compared to **class-based components**. The gap is made up with the help of a special ReactJS concept called **“hooks”**. Hooks are special functions that allow ReactJS features to be used in **functional components**. If you write a function component, and then you 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.

Hooks are similar to JavaScript functions, but you need to follow these two rules when using them. These rules are:

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 components renders.

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.

## Hook uses useState() functional component for setting and retrieving state.

## 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, Effect Hooks are equivalent componentDidMount(), componentDid Update (), and componentWillUnmount() lifecycle methode.

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

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

In React component, there are two types of side effects:

1. Effects Without Cleanup
2. Effects With Cleanup

### **Effects without Cleanup**

It is used in useEffect which does not block the browser from updating the screen. It makes the app more responsive. The most common example of effects which don't require a cleanup are manual DOM mutations, Network requests, Logging, etc.

### **Effects with Cleanup**

Some effects require cleanup after DOM updating. For example, if we want to set up a subscription to some external data source, it is important to clean up memory so that we don't introduce a memory leak. React performs the cleanup of memory when the component unmounts. However, as we know that, effects run for every render method and not just once. Therefore, React also cleans up effects from the previous render before running the effects next time.

## Custom Hooks

A custom Hook is a JavaScript function. The name of custom Hook starts with "use" which can call other Hooks. A custom Hook is just like a regular function, and the word "use" in the beginning tells that this function follows the rules of Hooks. Building custom Hooks allows you to extract component logic into reusable functions.

**References:**

<https://www.javatpoint.com/react-hooks>

<https://www.geeksforgeeks.org/reactjs-usestate-hook/>

<https://www.freecodecamp.org/news/introduction-to-react-hooks/>

<https://www.geeksforgeeks.org/reactjs-functional-components/>

<https://dev.to/workshub/state-management-battle-in-react-2021-hooks-redux-and-recoil-2am0>