To use localStorage in useEffect React, you can use the JSON.parse() method to parse the variable and check the conditions. As localStorage stores values as strings only, the values of isLoggedIn are stored as “true” and “false,” which will both equate to true. Therefore, you need to parse the variable and then check the conditions.

Here’s an example of how to do this:

const Dashboard = () => {  
 useEffect(() => {  
 [...]  
 }, [localStorage.getItem('isLoggedIn')]);  
   
 const isLoggedIn = JSON.parse(localStorage.getItem("isLoggedIn"));  
   
 return (  
 <div style={{ alignContent: "center" }}>   
 {isLoggedIn && <div><HeaderLogout /> <Content /></div>}  
 {!isLoggedIn && <div><HeaderLogin /></div>}  
 </div>  
 );  
};  
export default Dashboard;

**React Hooks are JavaScript functions** that you can import from the React package to add capabilities to your components. Hooks allow React developers to use state and lifecycle methods within functional components. They also operate with existing code, making them easily adoptable into a codebase. To use localStorage with React Hooks, you will need two hooks:

* useEffect(): The Effect Hook is activated by default after the first render and each time the state is changed. As the names suggest, it is used to perform an effect each time the state changes. This hook is great for configuring listeners, retrieving data from the API, and deleting listeners before the component is removed from the DOM.
* useState(): The State Hook is used to add state variables to functional components.

**localStorage** provides access to a browser’s storage object, which includes five methods:

* setItem(): This method is used to add a key and a value to localStorage.
* getItem(): This method is used to get an item from localStorage using the key.
* removeItem(): This technique is used to delete an item from localStorage based on its key.
* clear(): This technique is used to delete all instances of localStorage.
* key(): When you supply a number, it aids in the retrieval of a localStorage key.

In this context, we will only consider the first two methods. To store objects in localStorage, you can use the setItem() method. This method is used to store values to a key. This value can be of any datatype, including text, integer, object, array, and so on. However, it is vital to remember that in order to store data in localStorage, you must first stringify it with the JSON.stringify() function. Here’s an example of how to use the setItem() method:

const [items, setItems] = useState([]);  
useEffect(() => {  
 localStorage.setItem('items', JSON.stringify(items));  
}, [items]);

To retrieve objects from localStorage, you can use the getItem() method. This method retrieves objects from localStorage. There are other methods to accomplish this with React, but we will use the useEffect() hook because it is the best one. The useEffect() hook helps us fetch all items on first render, which means that when the component mounts or re-renders, it obtains all of our data from localStorage. Note that this is why we passed in an empty second argument. Here’s an example of how to use the getItem() method:

const [items, setItems] = useState([]);  
useEffect(() => {  
 const items = JSON.parse(localStorage.getItem('items'));  
 if (items) {  
 setItems(items);  
 }  
}, []);

To listen for changes in localStorage, you should add an event listener for local storage. You can use the useEffect() hook to add an event listener for local storage. Here’s an example of how to do this:

useEffect(() => {  
 function checkUserData() {  
 const item = localStorage.getItem('userData')  
  
 if (item) {  
 setUserData(item)  
 }  
 }  
  
 window.addEventListener('storage', checkUserData)  
   
 return () => {  
 window.removeEventListener('storage', checkUserData)  
 }  
}, [])

To update state directly instead of a separate array, you should use the onChange method to update localStorage. Here’s an example of how to do this:

function InputTask() {  
 const [task, setTask] = useState();  
 const [tasks, setTasks] = useState(JSON.parse(localStorage.getItem("tasks") || "[]"));  
   
 function handleAddTask() {  
 setTasks([...tasks, task]);  
 setTask("");   
 }  
   
 useEffect(() => {  
 return () => localStorage.setItem("tasks", JSON.stringify(tasks));  
 }, []);  
}

In conclusion, to use localStorage in useEffect React, you can parse the variable and check the conditions using JSON.parse(). You can use React Hooks like useEffect() and useState() to add capabilities to your components. To store objects in localStorage, you should use the setItem() method, and to retrieve objects from localStorage, you should use the getItem() method. To listen for changes in localStorage,