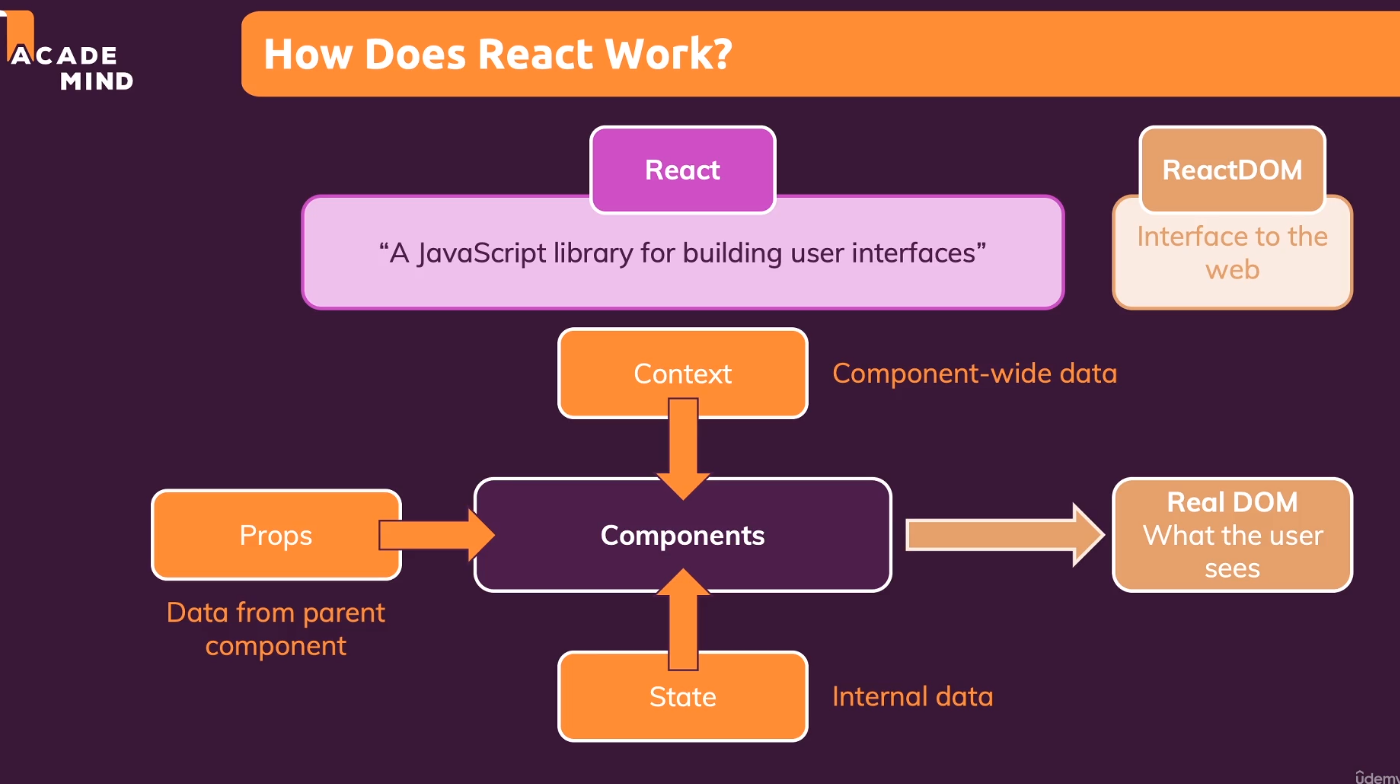
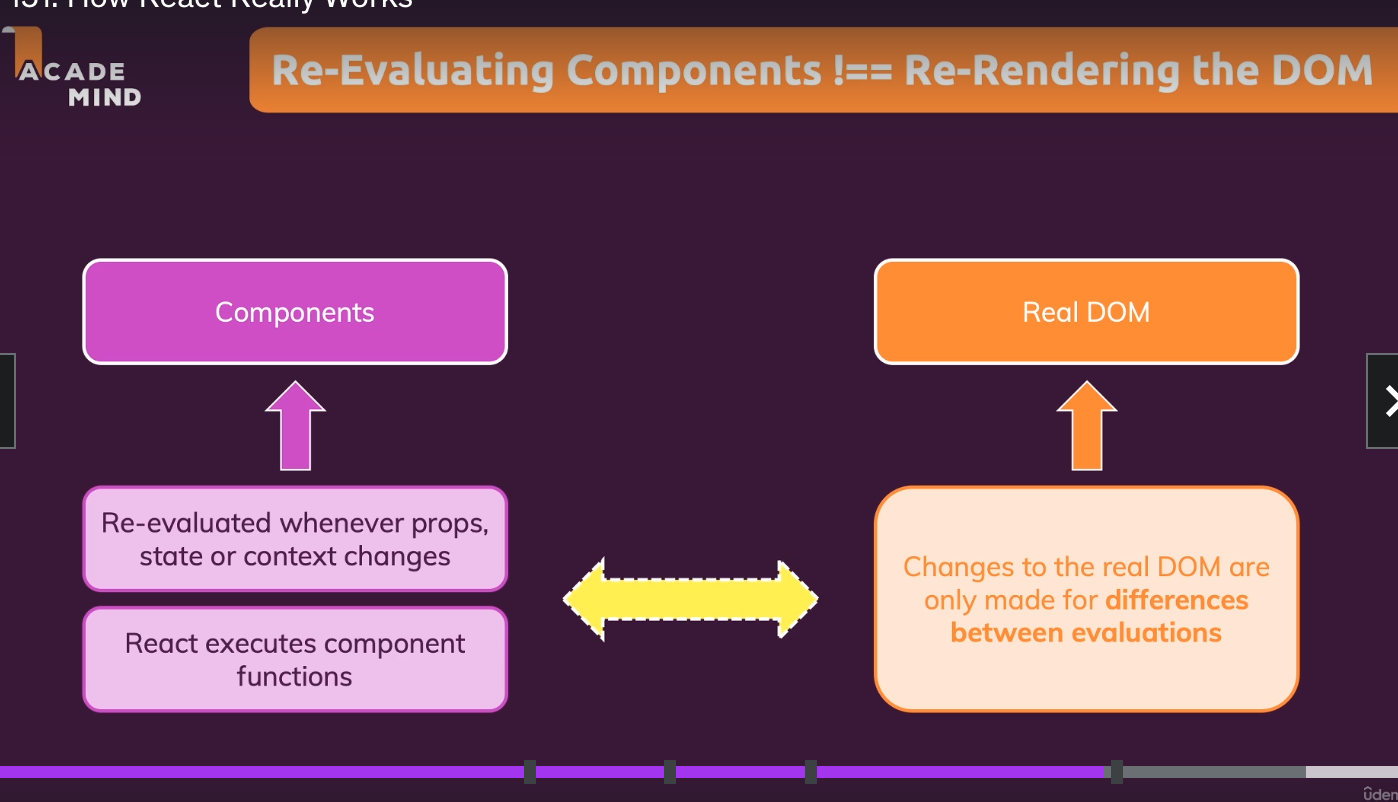
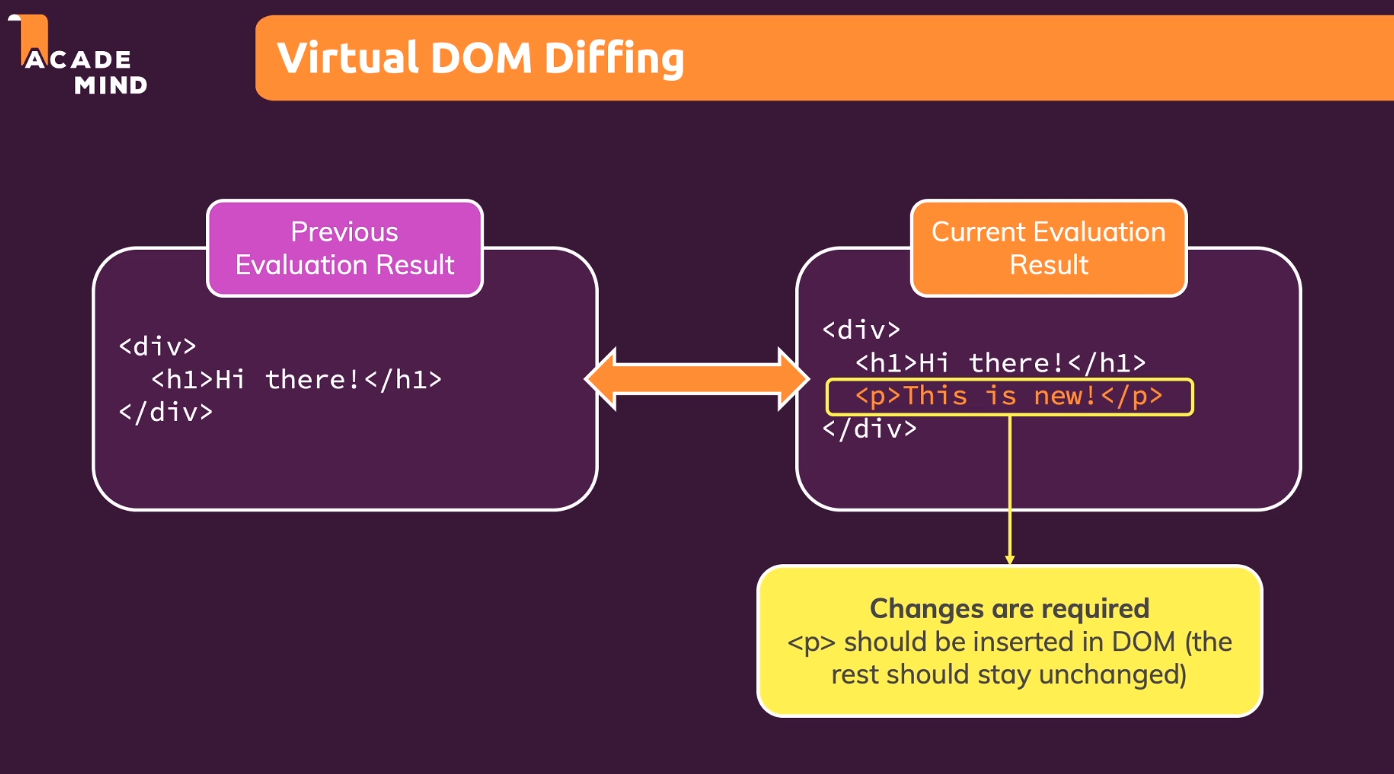
**9. React Optimization Techniques**

React is a javascript library for building user interfaces. How does react really work?



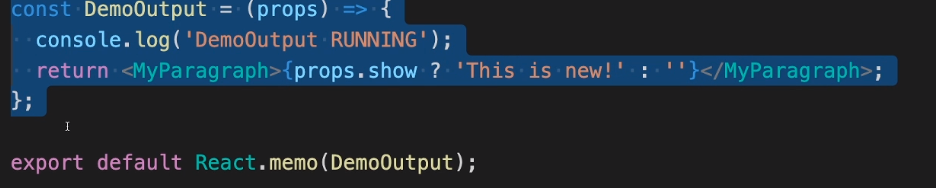
React does virtual DOM diffing. I.e it compares the real dom and virtual DOM and updates only that part which needs to be changed.





State changes, component changes result in rerendering of the paragraph. Only the difference between the virtual snapshots were considered for state updates. If there is a state change in the parent component, there can be unnecessary re-evaluation of the chain of the child components . This hampers the performance and unnecessary re-execution of the component chains and needs to be handled.

In order to avoid the re-render of the components, we can wrap those components in Reac.memo(DemoOutput). This ensures re-exucution if the value of the props change and otherwise re-execution would stop.



Why are we not using React.memo for all the components? Because React.memo has its own set of performance challenges because it involves comparison of props. This is recommended if there is heavy nested components which hamper the performance of the application over the props comparison.

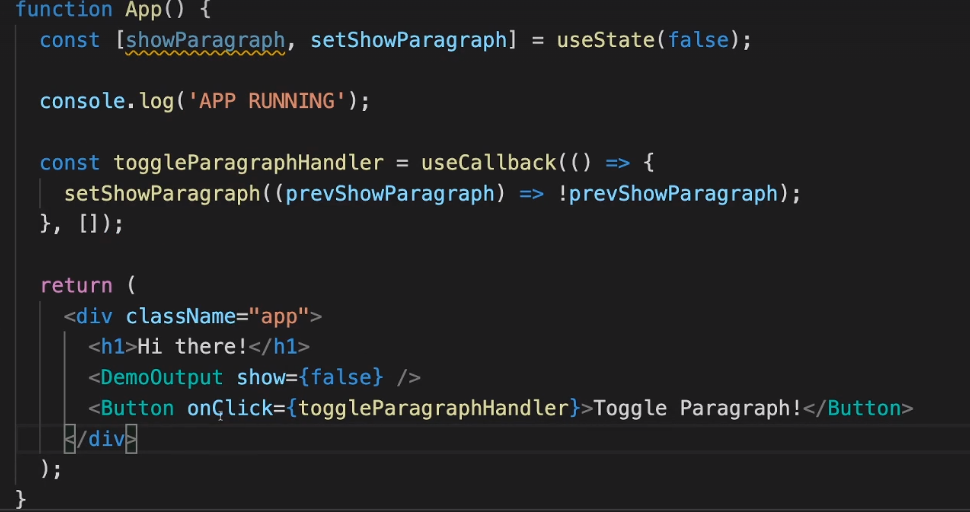
React.memo() may not work in scenarios where we would be comparing functions passed as properties.

For example <button onClick={handleButtonClick}>Click Me!</button>. Here this React.memo() would involve comparison of functions which would always return.

In order to solve the problem , react has provided us with useCallback() hook. useCallback() hook can be used to track function/object specific state changes and to avoid re-rendering of the unwanted components.

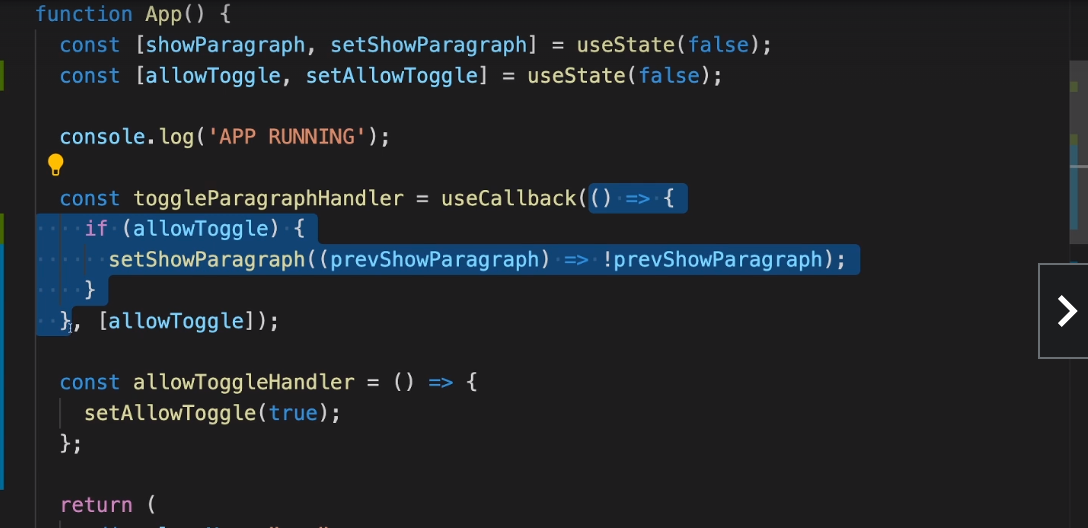
useCallBack also uses certain dependencies like useEffect(). It is followed by an array of dependencies which needs to be rendered only if the dependencies change.

So Apart from using react memo for that component like for example button in our case, we also need to enclose the property that is being passed to the button in useCallback().



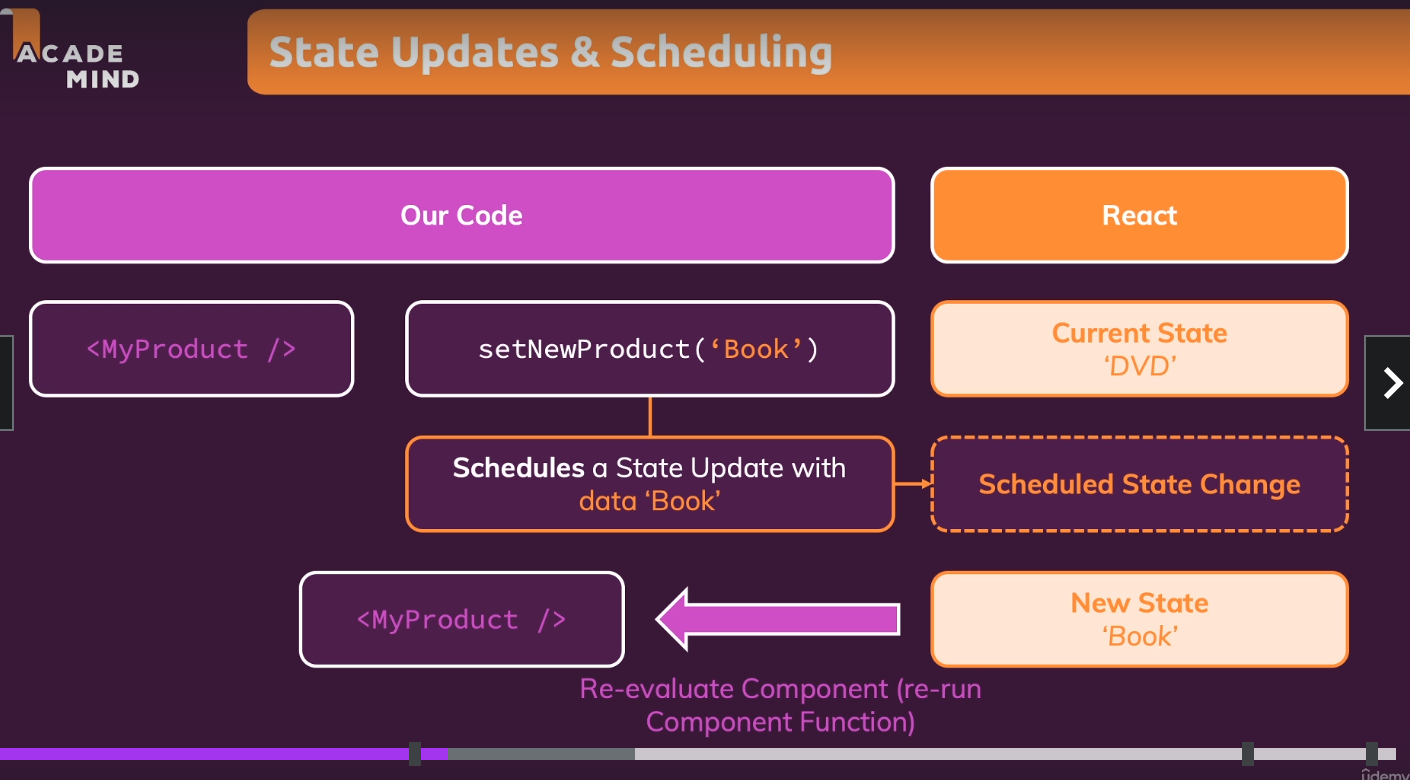
What is the dependency array in useCallback() react hook?

Consider an example where we have two states say allowToggle and showParagraph. I want to showParagraph only when allowToggle is set to true. useCallback() hook stores the reference of exactly the same function and the values associated to that function are also stored exactly the same and not the updated one. If we want to make sure that the updated values are rendered then we need to tell react to create a new copy of that function/ object whenever the dependency changes. That dependency is specified in that useCallback() react hook.

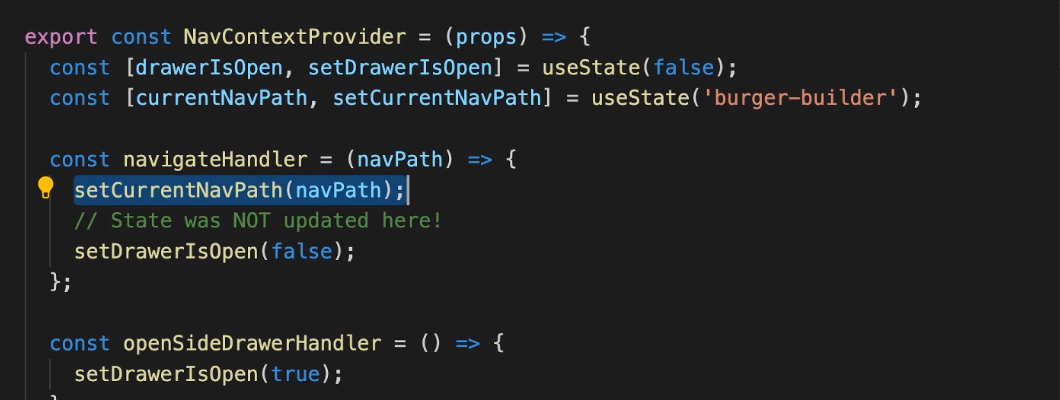


State Updates and Scheduling ➖

State updates are never instantaneous but they are scheduled .However in reality it feels instantaneously and very fast.



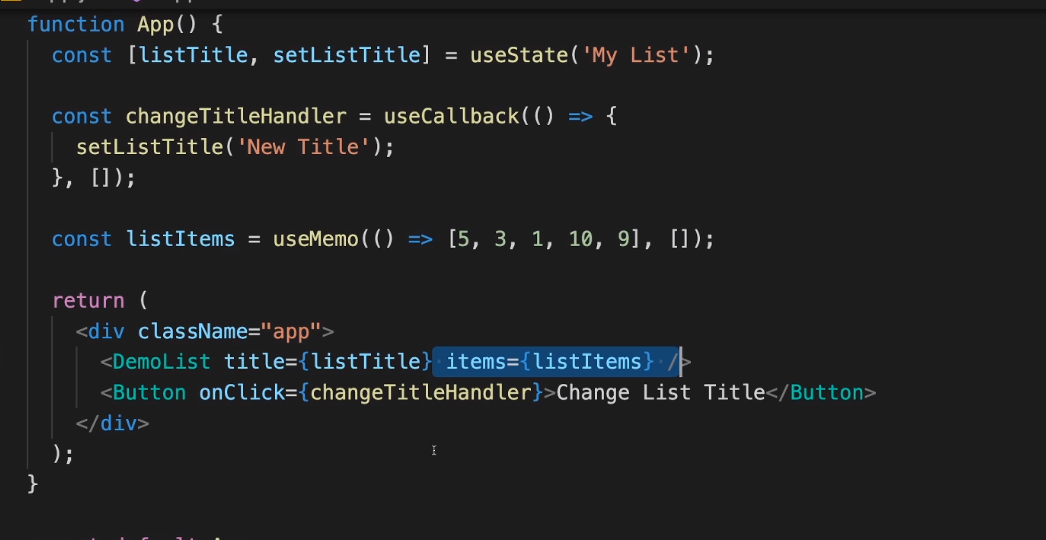
In one synchronous process, react will call the state simultaneously. For example ➖

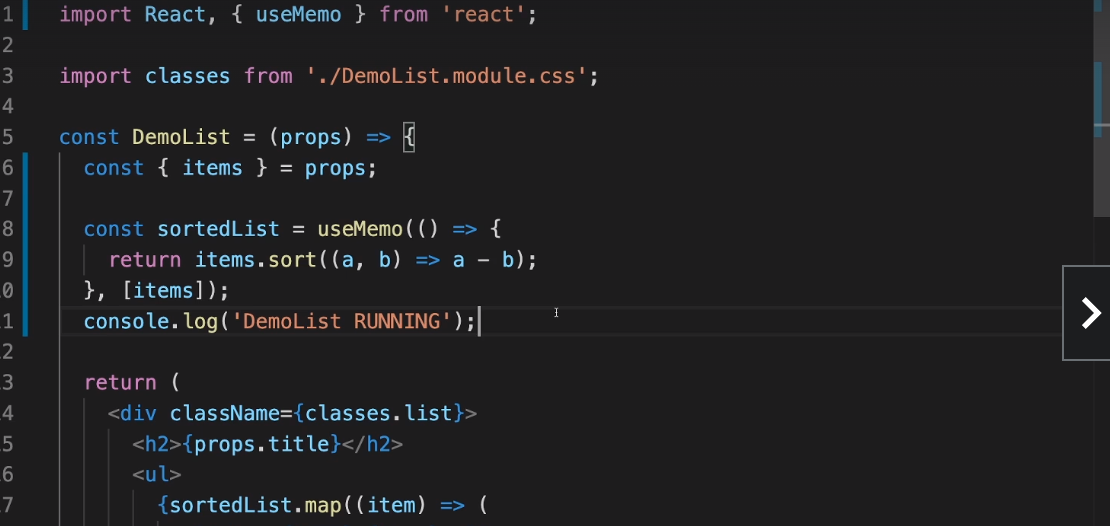


React state updates are done simultaneously as shown above. setCurrentNavPath(navPath) and setDrawerIsOpen(false) these state updates are done in batch simultaneously .

useMemo() optimization:-

There can be a scenario where certain props are passed to another component and that component does some performance intensive task. (for example sorting) So whenever a state is updated in a parent component, it re-renders and lets say if we are passing an array to the child component of that parent component, it re-creates a new array and sorting operation is performed in the child component everytime the parent component is re-rendered. This can be avoided using useMemo() as follows:-





Read this article it not clear ➖

https://www.google.com/search?q=react.memo,+useMemo,+useCallback&sxsrf=AJOqlzWj6I\_ipkHgirf2ENyJciF-XHtMVw:1674104410955&source=lnms&tbm=vid&sa=X&ved=2ahUKEwir6-z17NL8AhWLTGwGHSWAD1cQ\_AUoAXoECAEQAw&biw=1440&bih=698&dpr=2#fpstate=ive&vld=cid:5f487bbb,vid:uojLJFt9SzY