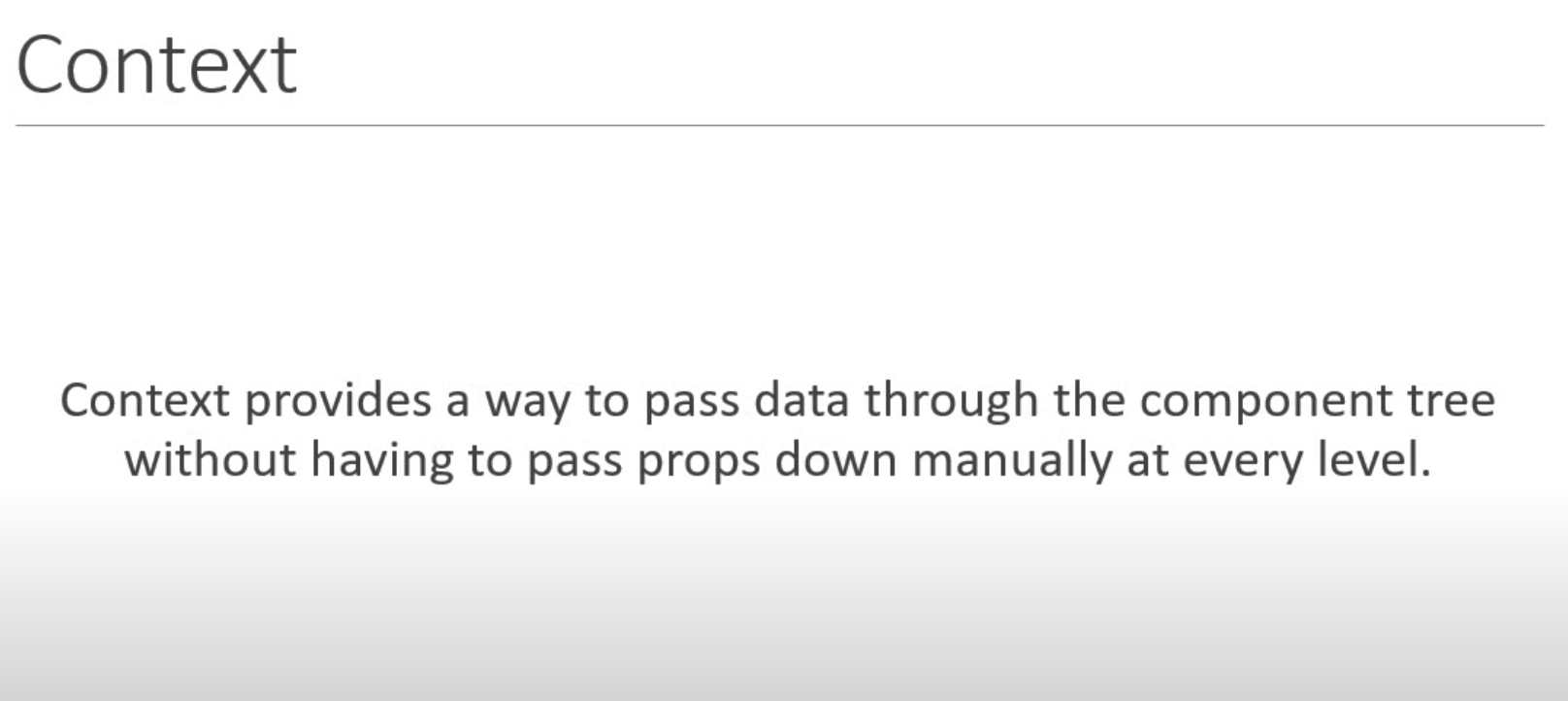
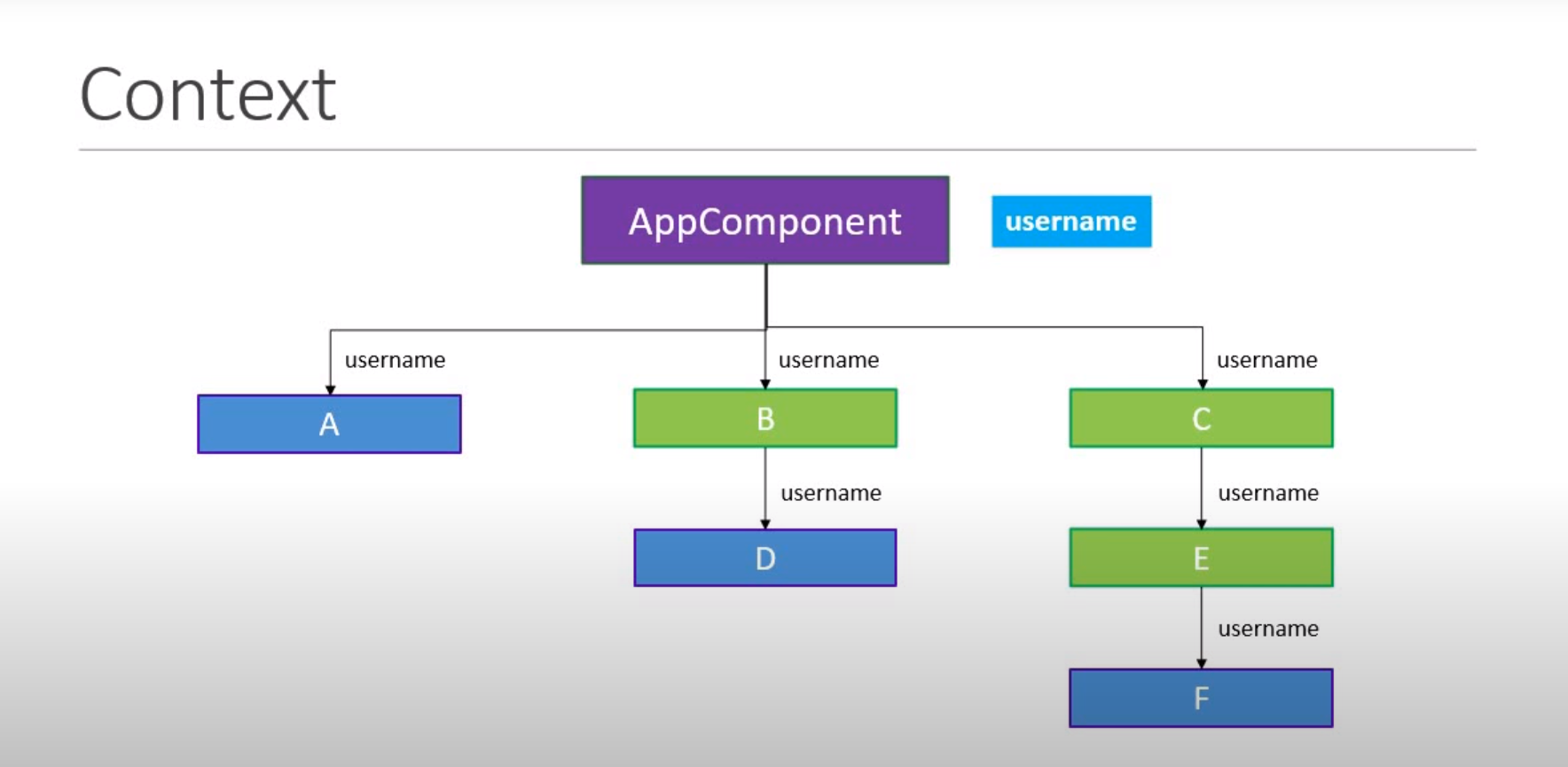
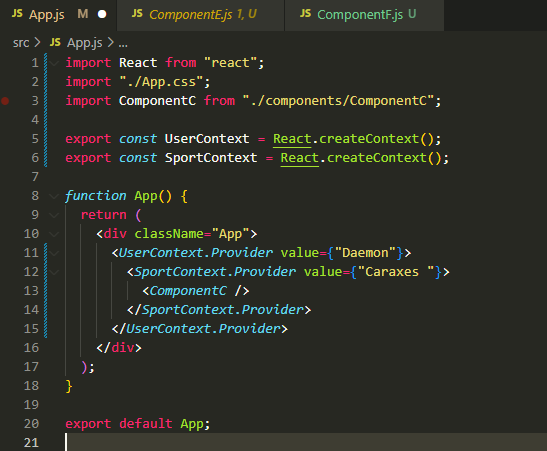
**useContext Hook**



In this example, we want to pass *username* from *AppComponent* to Component *F* without needing to pass it through Component *C* & Component *E.*



*App.js:*



1) We create a context using the *createcontext()* method.

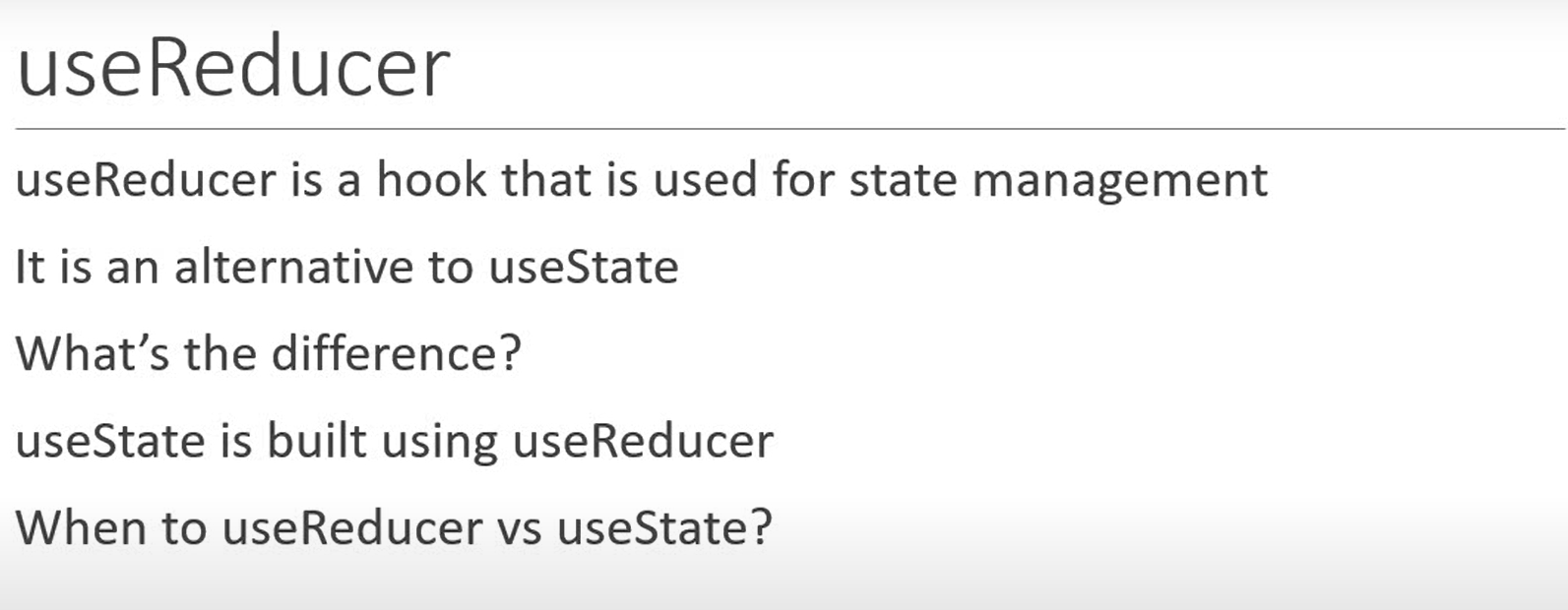
2) We provide the context value at a high level (*<ComponentC/>*) in the component tree.

3) We consume the context value using the render props pattern.

the *useContext()* hook only makes the consumption of the context value simpler.

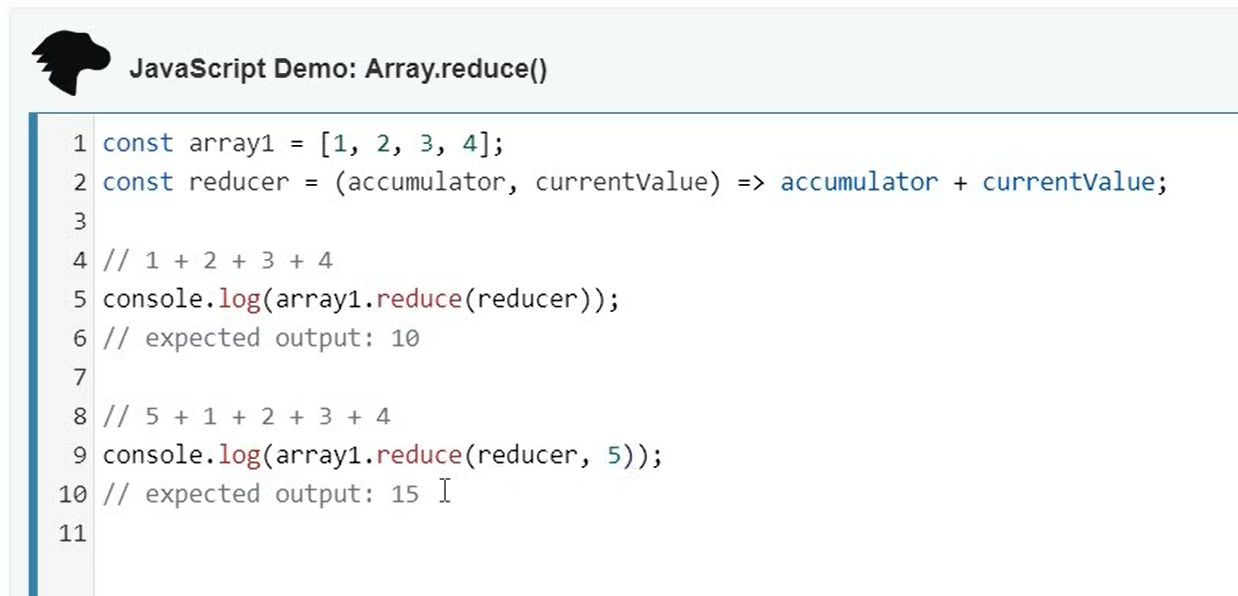


**useReducer Hook**

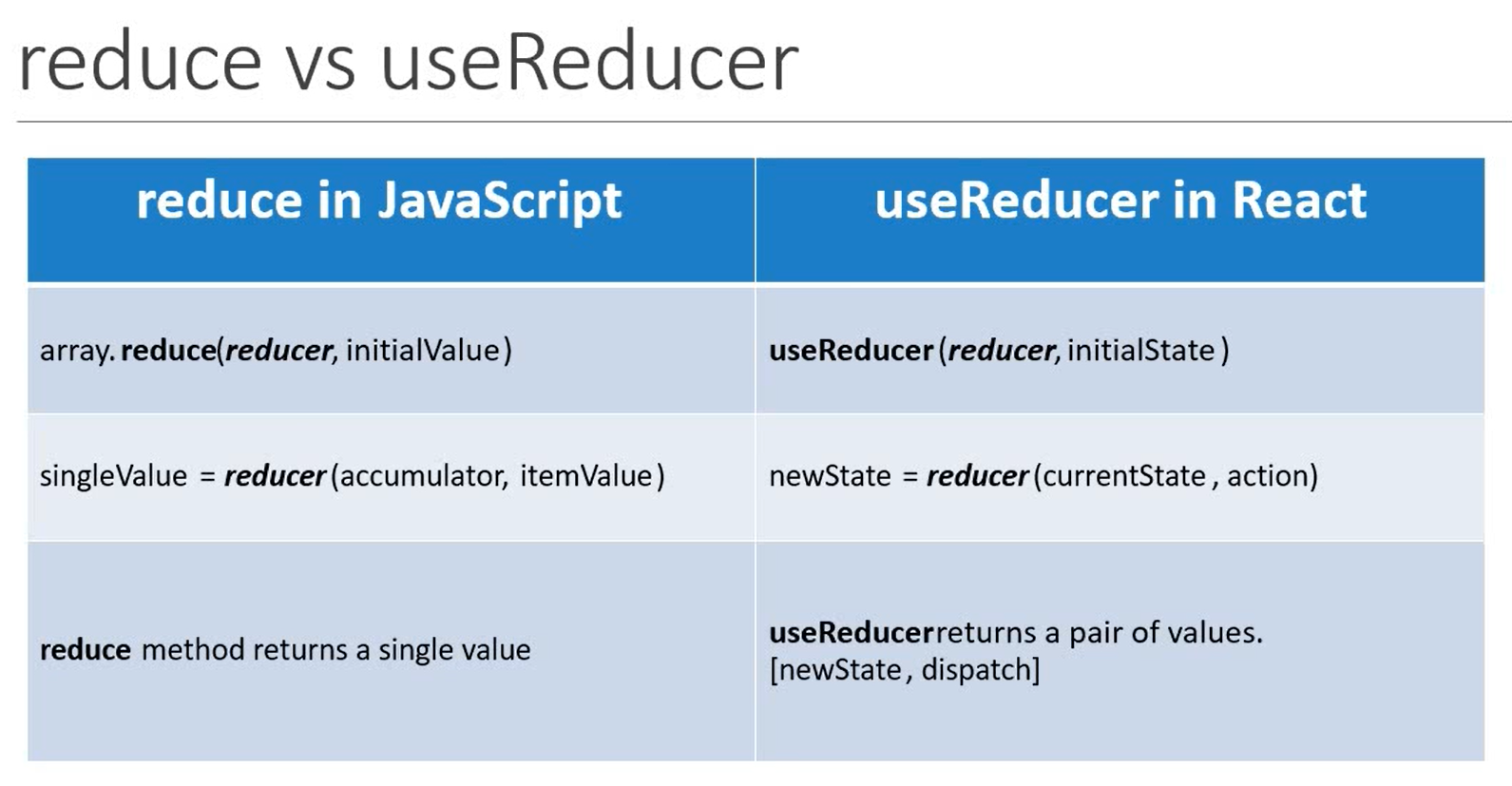


*useReducer* hook is similar to the *reduce()* method in JavaScript.

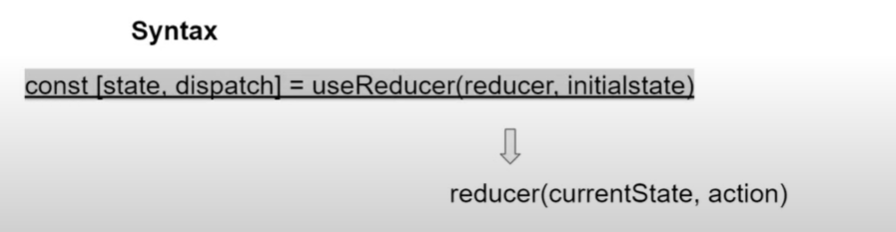
The *reduce()* method has two forms.



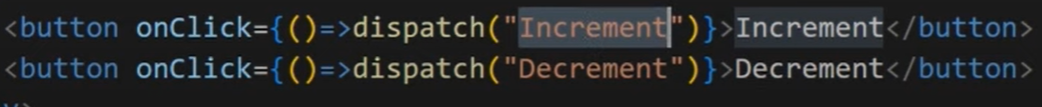
*reduce()* vs *useReducer* hook:



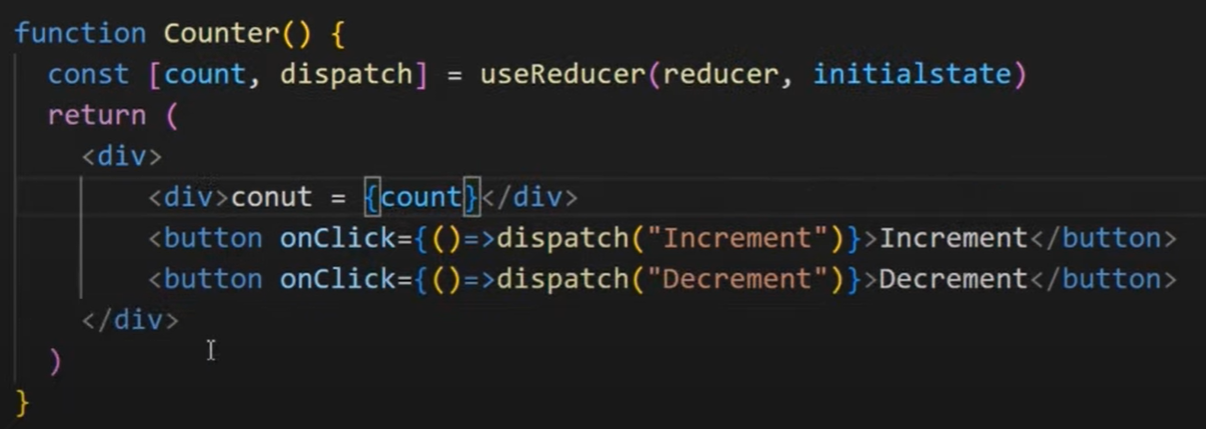
The syntax for *useReducer* is:

**

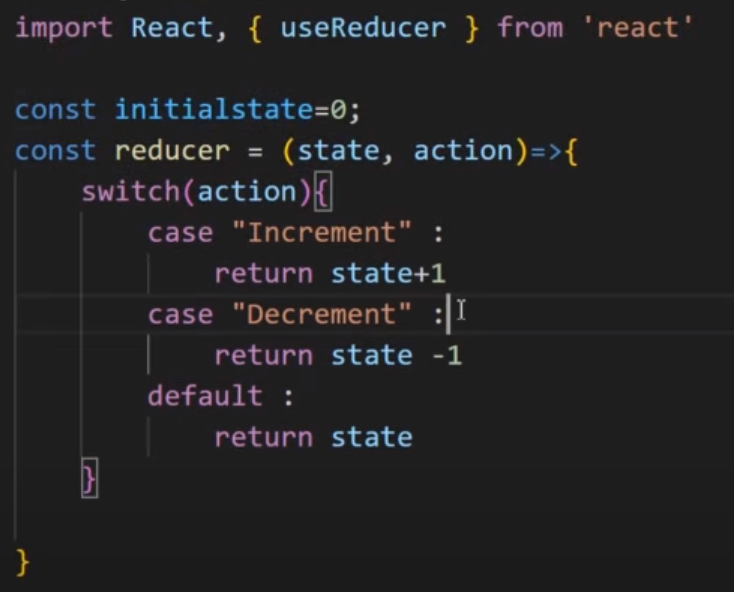
In *Counter.js* define two buttons:



The *Counter* function will look something like:

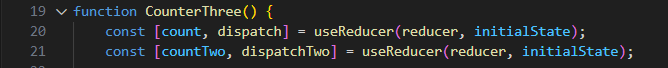


We need to declare the initial state variable (*initialstate*) & the *reducer* function.



We can use multiple *useReducer()* hooks to increase efficiency. In this example, we will make two counter buttons.

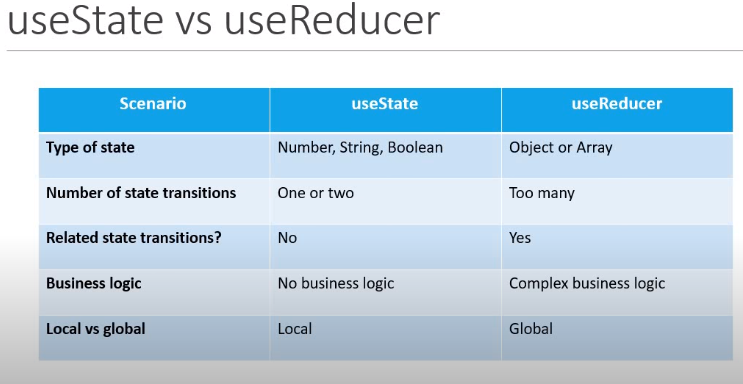
The *reducer* functioncan be reused again although to maintain their states separately, we need two separate *useReducer()* hooks.



In addition to the first button, we add the JSX for the second



**useState vs useReducer**



**useCallback hook**

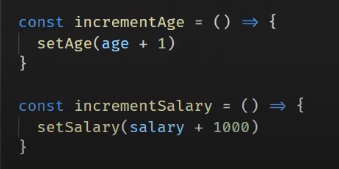
*useCallback* is a hook that will return a memorized version of the callback function that only changes if one of the dependencies has changed.

It is useful when passing callbacks to optimized child components that rely on reference equality to prevent unnecessary renders.

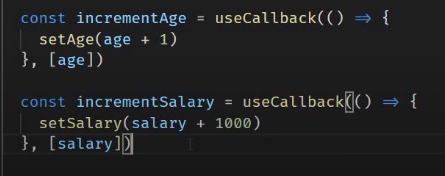
The syntax of *useCallback* hook is:

*const cachedFunc = useCallback (function, dependencies)*

without using *useCallback:*



using *useCallback:*

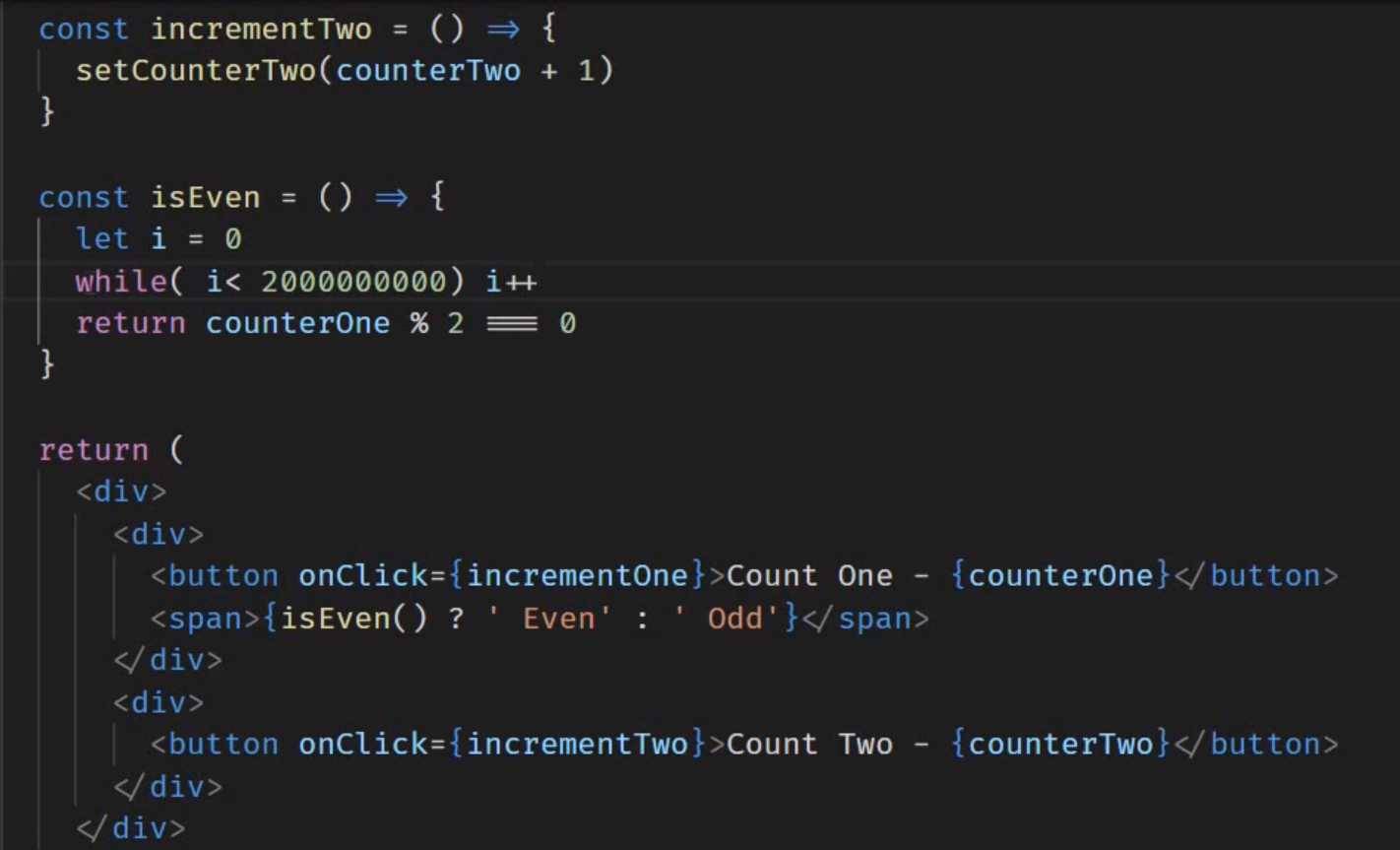


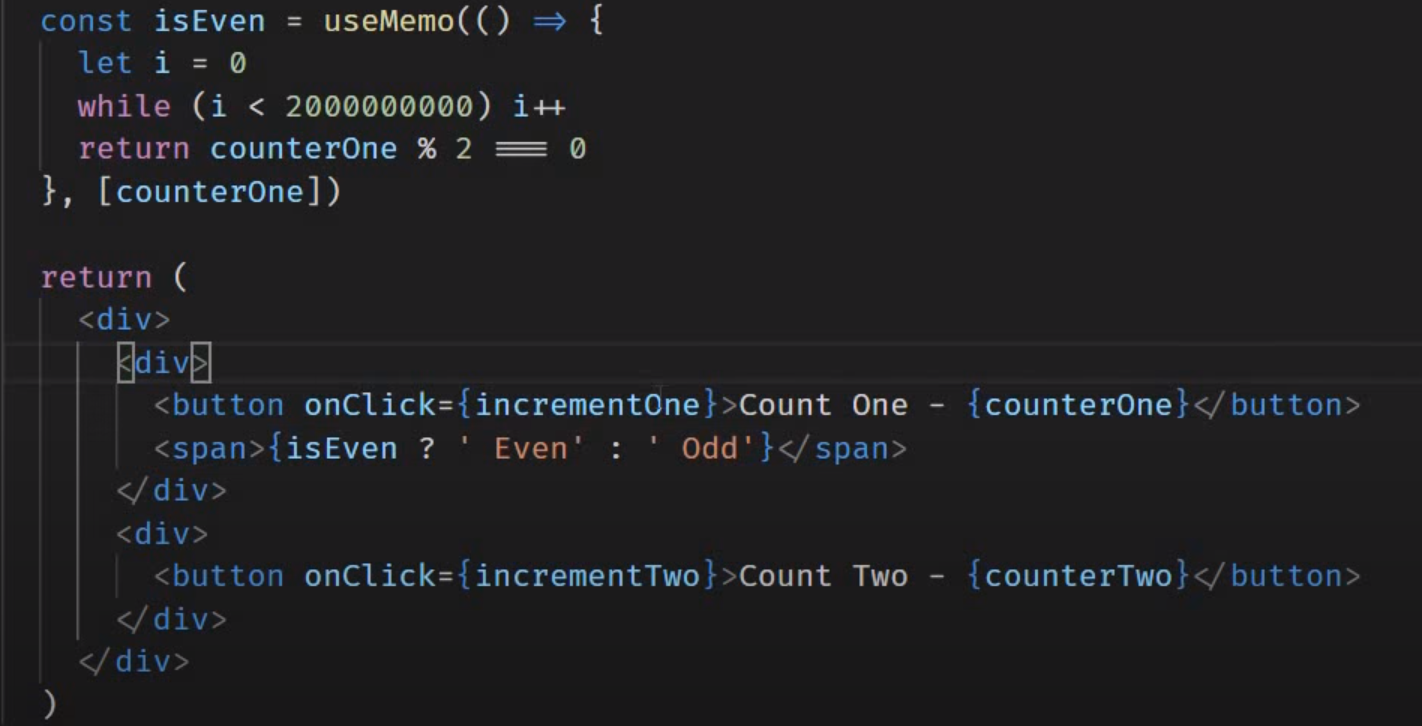
**useMemo hook**

The *useMemo* hook returns a memorized value. Think of memorization as caching a value so that it does not need to be recalculated.

The useMemo hook only runs when one of its dependencies update.

In this example,





The useMemo and useCallback Hooks are similar. The main difference is that useMemo returns a memoized value and useCallback returns a memoized function. You can learn more about useCallback in the useCallback chapter.