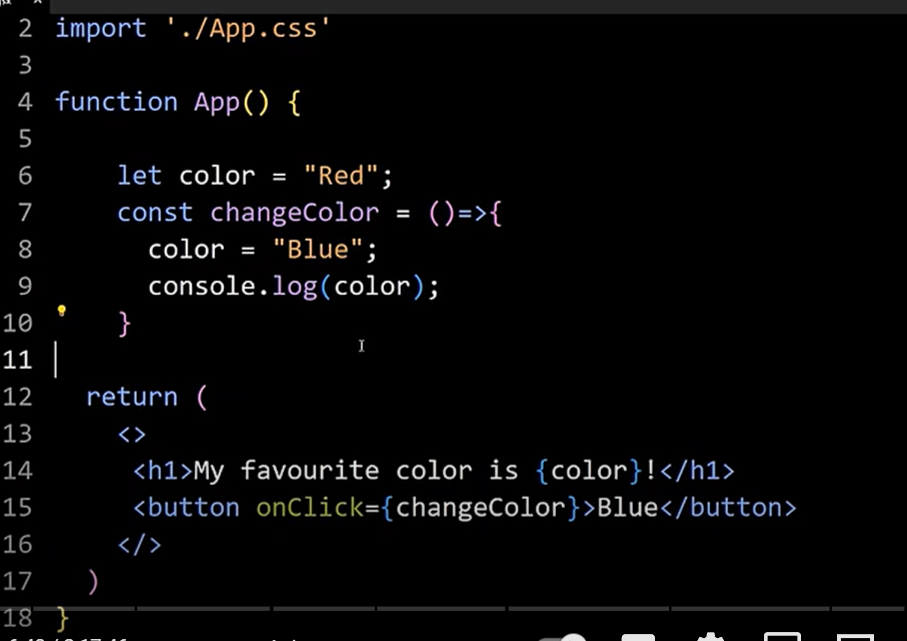


1)Problem without useState: we have used the variable {color} in <h1> and when we click the button it changes the value of the color variable ..but in the ui doing this much does not make any changes even though the value of variable color gets changed . Thus to resolve this we use useState hook.

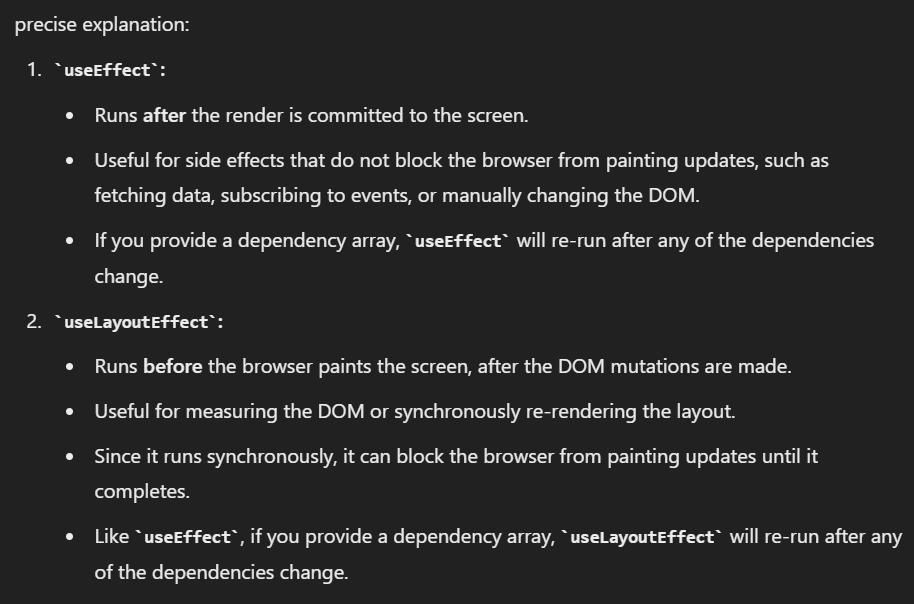


**useLayoutEffect:**

It is same as useEffect with the only difference that useEffect loads after what mentioned in the dependency array gets changed and gets rendered.

Note that, the useEffect hook comes into play after the changed dependency elements get rendered or in case of empty dependency array, the entire page gets rendered.

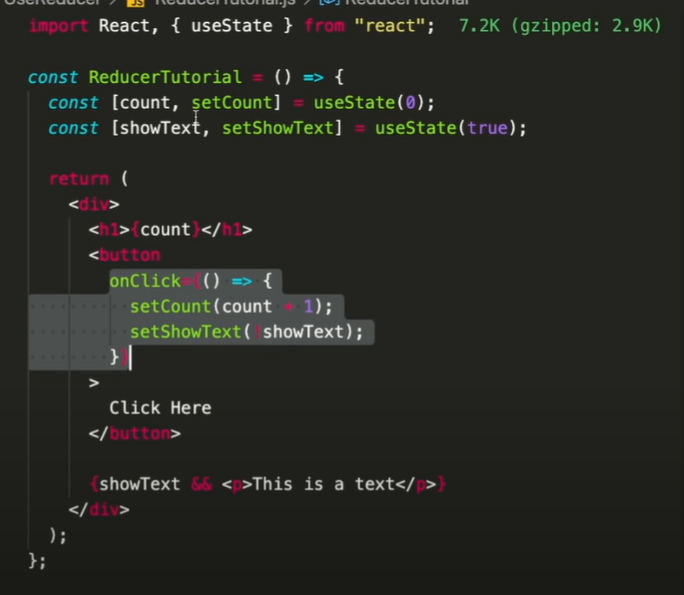
But in case of useLayoutEffect() hook, the function gets executed before rendering of layout or the page itself or before the changed elements of depenedency array.



UseReducer hook:

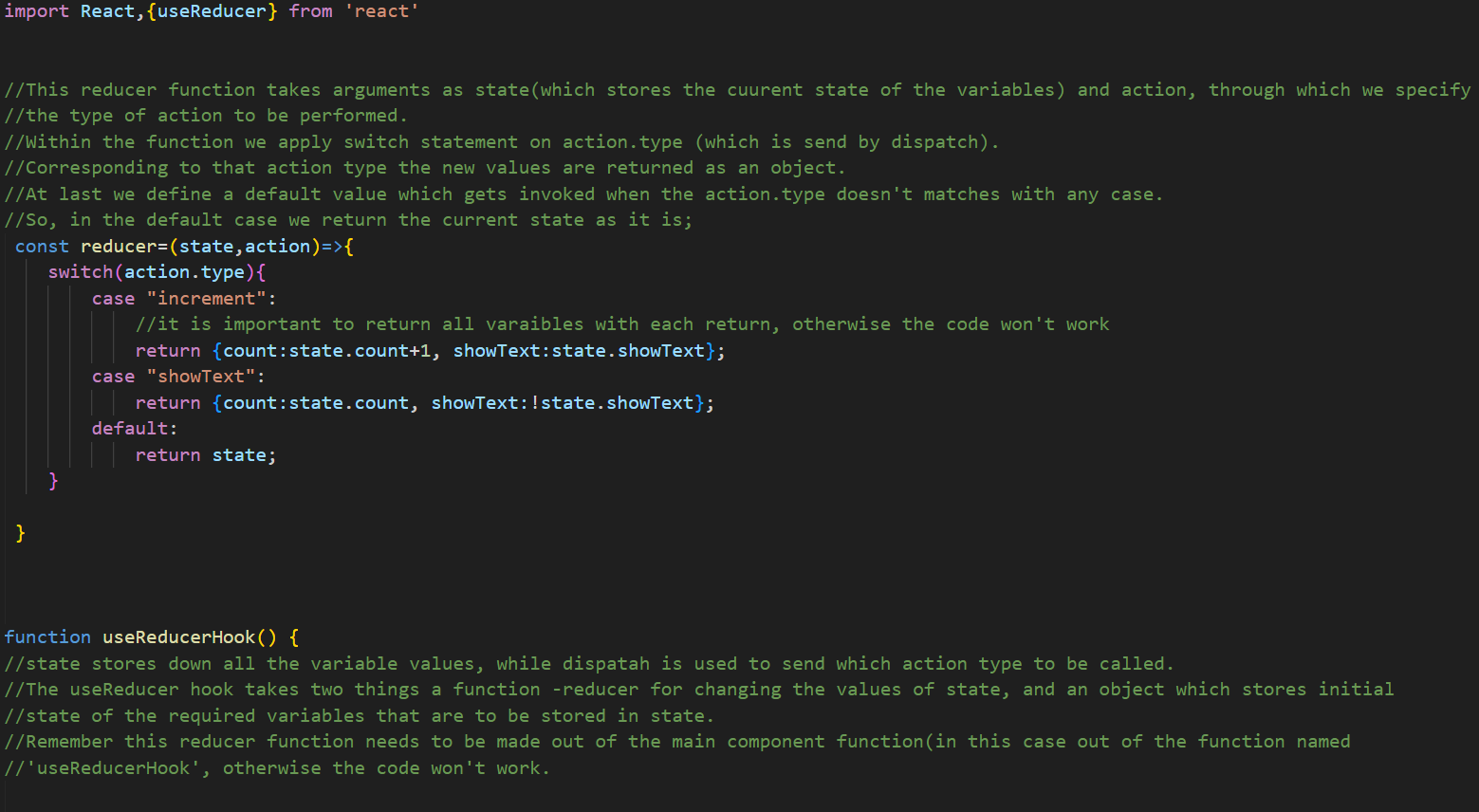
It can be seen as an alternative or a replacement to the useState hook.

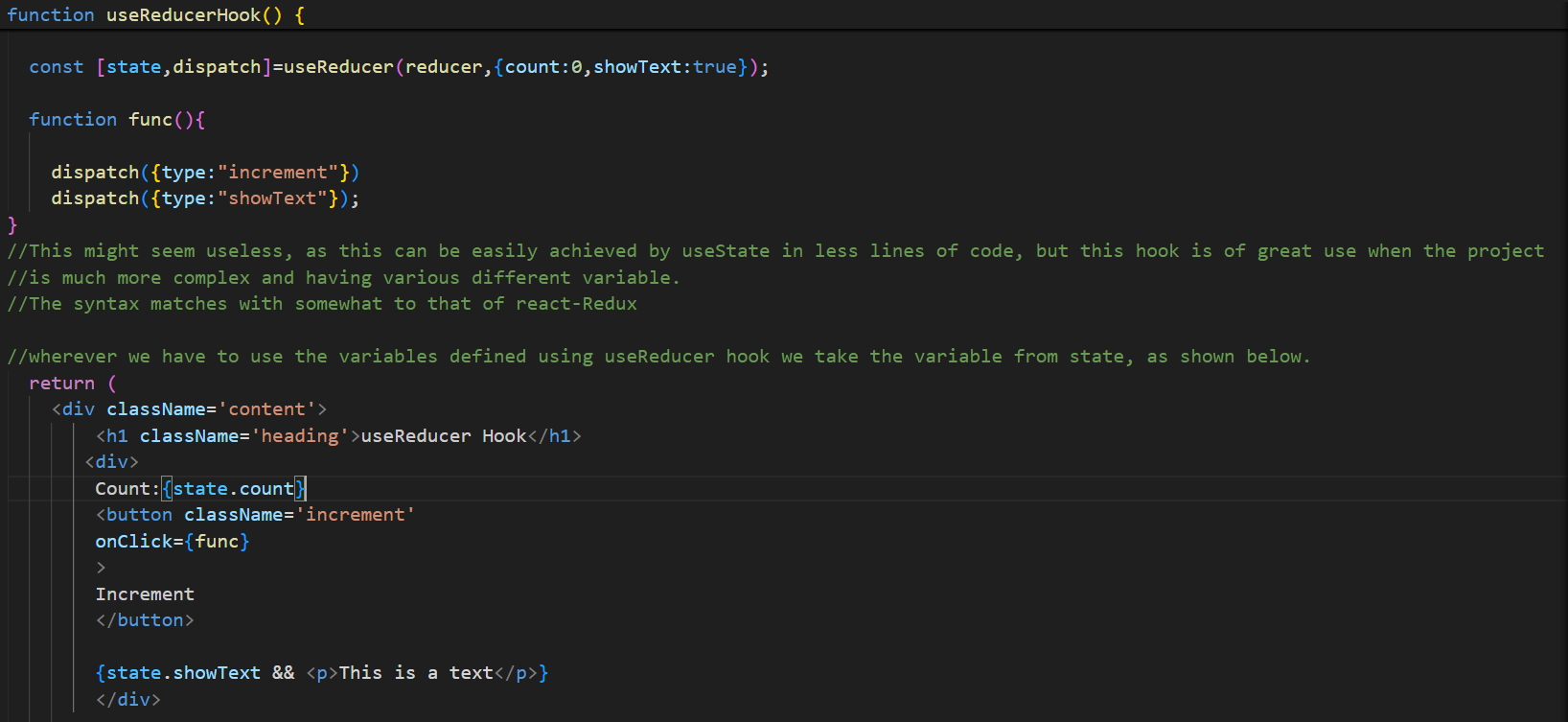
We can say this also is used to create variables and render their values whenever they get changed.



So, let’s see as whenever we want to change more than one variable at occurrence of some event, say onClick then we have to change all variables by using their respective functions declared using useState hook.

This is where useReducer hook is used.





UseEffect:

It is a hook or simply a function which gets called, whenever the page re-renders.

Apart from re-rendering the whole page we can also decide what to detect in order to call this function by using the dependency array.

But remember if we don’t put anything in the dependency array, useEffect will get invoked even if a very small thing causes the page to re-render (i.e. at each re-render).

useRef:

Easiest way to access and manipulate DOM elements.