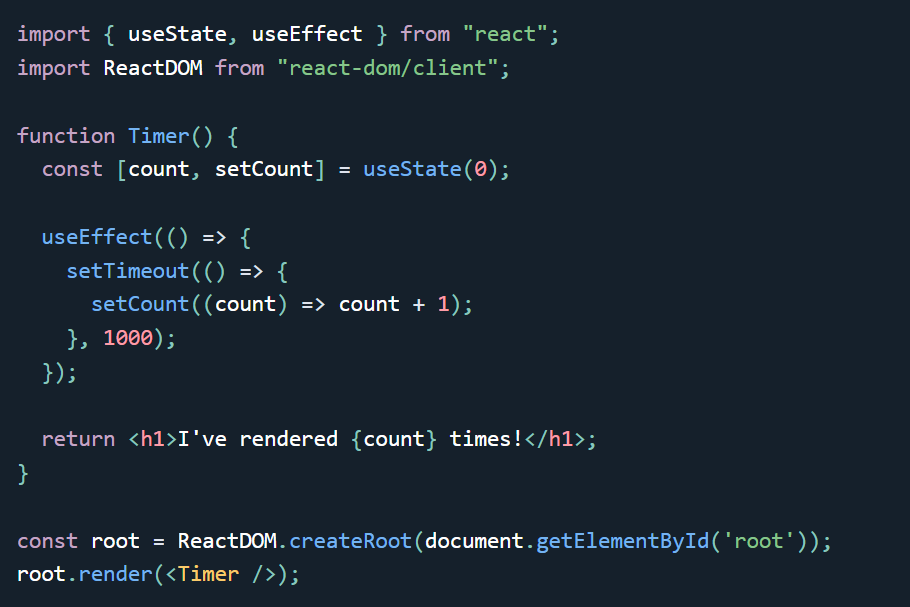
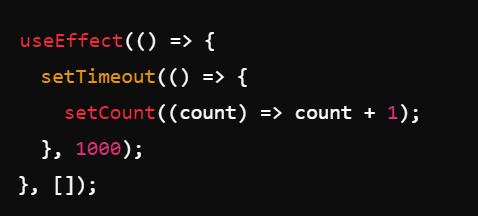
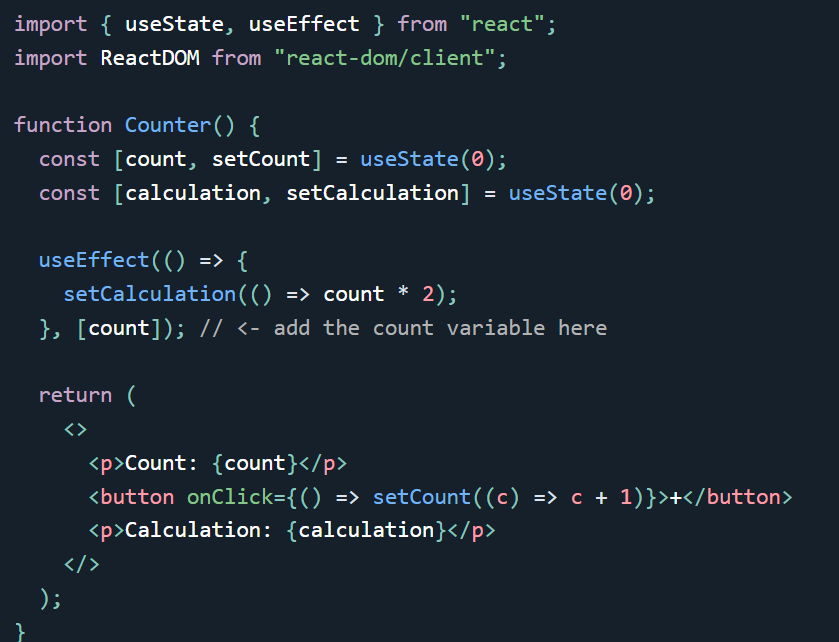
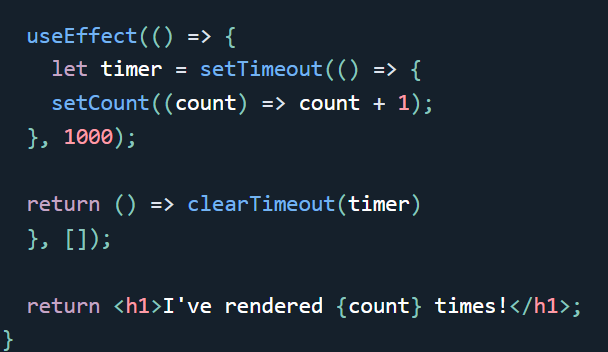
# Notes

## Hooks

1. UseState: The React useState Hook allows us to track state in a function component.
2. UseEffect: The useEffect Hook allows you to perform side effects in your components. useEffect runs on every render. That means that when the count changes, a render happens, which then triggers another effect. 

  
**Empty Dependency Array []:**

* Since the dependency array is empty, the effect will only run once—when the component is first rendered (mounted). It will not run again on subsequent renders or updates.
* 
* So when the value of count changes it triggers a side effect that is a function that on change of count multiplies 2 with calculation value.
* Some effects require cleanup to reduce memory leaks. Timeouts, subscriptions, event listeners, and other effects that are no longer needed should be disposed. We do this by including a return function at the end of the useEffect Hook.



1. 

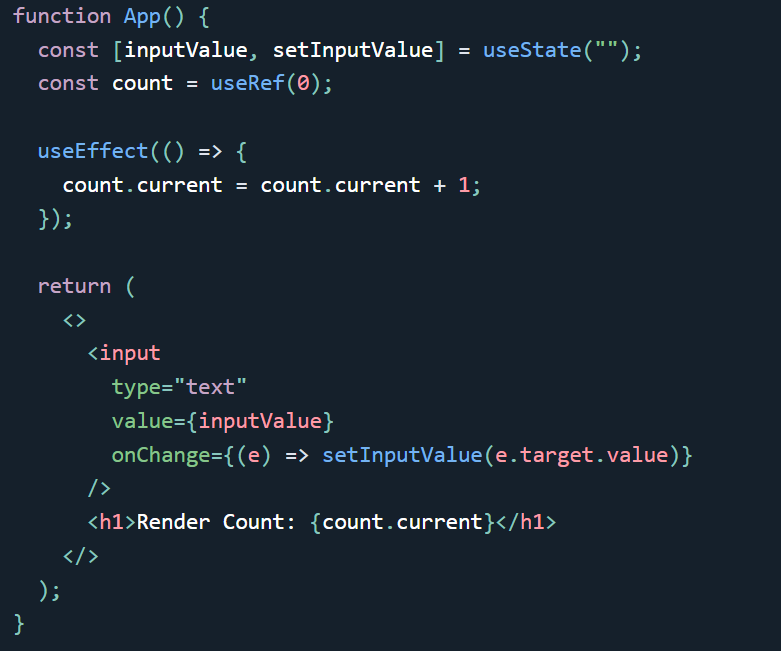
The useRef Hook allows you to persist values between renders.

It can be used to store a mutable value that does not cause a re-render when updated.

It can be used to access a DOM element directly.  
Does Not Cause Re-renders

If we tried to count how many times our application renders using the useState Hook, we would be caught in an infinite loop since this Hook itself causes a re-render.

To avoid this, we can use the useRef Hook.



Tracking State Changes

The useRef Hook can also be used to keep track of previous state values.

This is because we are able to persist useRef values between renders.

# Blender

* G to move the object selected and x y or z to move in that direction
* G Z 1 to move the object a step up on the floor
* G shift Z to move the object over the plane
* R X for rotation in X axis and R Y in Y axis and so on
* We can do scaling to make bigger or smaller using S
* Can move pivot using edit mode