DECODING

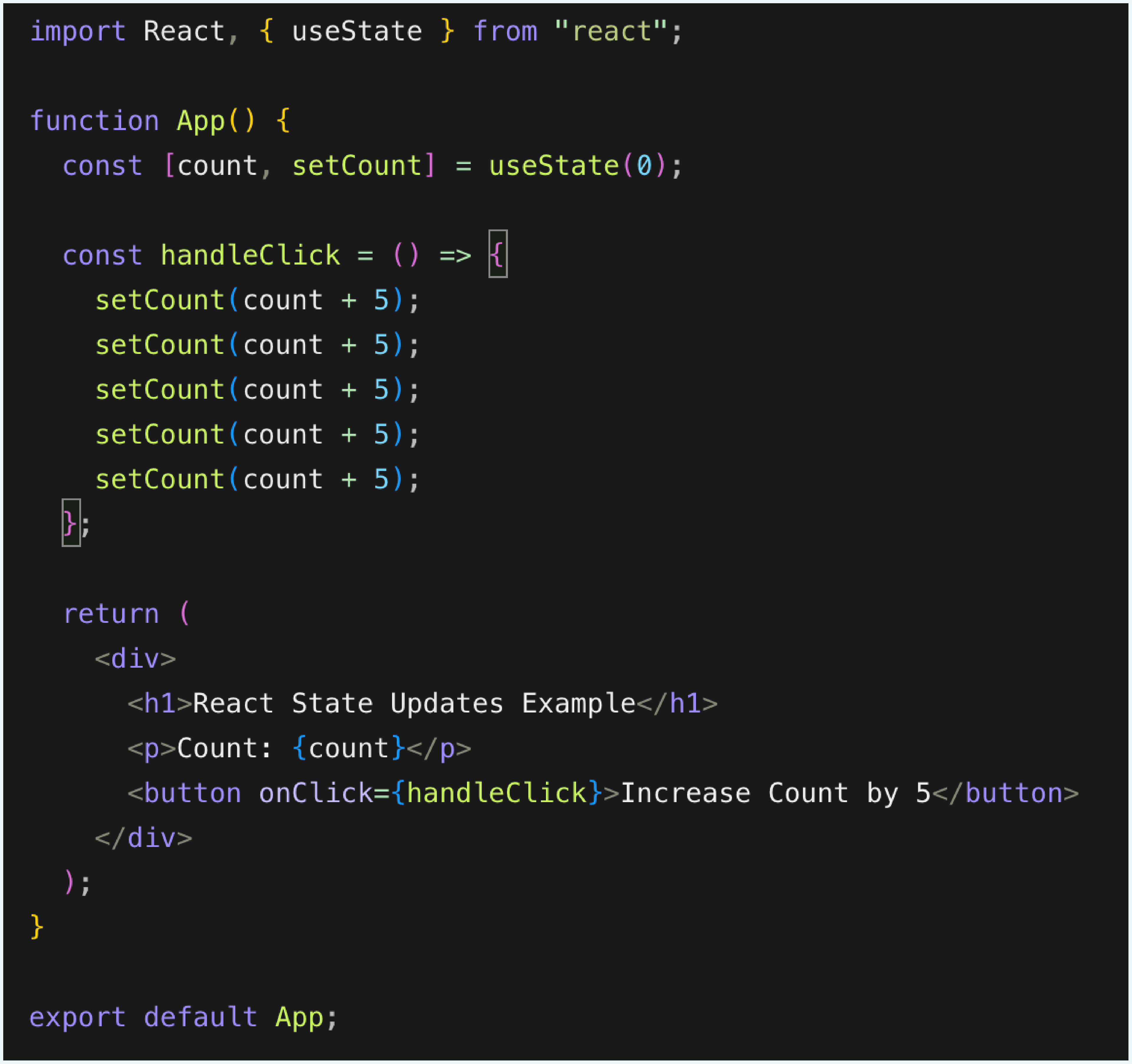
REACT STATE UPDATES

INTERVIEW QUESTIONS

Follow on

@DUVVURU KISHORE

INTERVIEWER: WHAT IS THE COUNT VALUE ON CLICK OF A BUTTON



```
import React, { useState } from "react";

function App() {
  const [count, setCount] = useState(0);

  const handleClick = () => {
    setCount(count + 5);
    setCount(count + 5);
    setCount(count + 5);
    setCount(count + 5);
    setCount(count + 5);
  };

  return (
    <div>
      <h1>React State Updates Example</h1>
      <p>Count: {count}</p>
      <button onClick={handleClick}>Increase Count by 5</button>
    </div>
  );
}

export default App;
```

Imagine you have a button that, when clicked, is supposed to increase a count by 5. Simple enough, right? But what happens behind the scenes can be a bit more complex.

In React, state updates are asynchronous. This means that when you call setCount(count + 5) multiple times in a single function, React doesn't immediately update the state each time.

```
SETCOUNT(COUNT + 5)
```

Here's how it works:

- You click the button, triggering the handleClick function.
- Inside handleClick, setCount(count + 5) is called five times in a row.
- However, since state updates are asynchronous, React doesn't update the state immediately. It queues up the updates and performs them all at once after the function finishes executing.

Asynchronous: In an asynchronous operation, tasks are executed independently of each other. Instead of waiting for each task to finish before starting the next one, the program can continue executing other tasks while waiting for certain operations to complete.

- **Each time setCount(count + 5) is called, it uses the current value of count (let's say it's 0) and adds 5 to it. But since the state hasn't been updated yet, it keeps using the original value of count for each call.**

SO FINALLY COUNT WILL BE 5