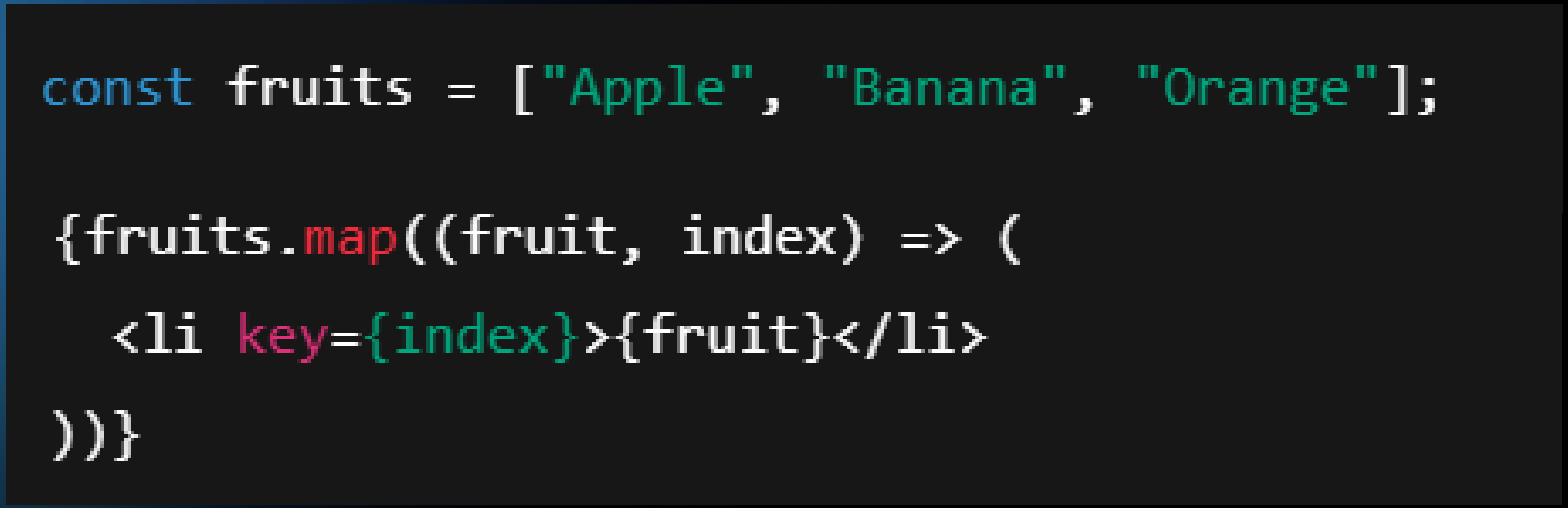
React JS-2

From Tech Jashwanth

Lists & Keys

Lists & Keys

1. A **key** is a unique identifier React uses to track each item in a list.
2. Helps React efficiently update only the changed items
3. Without key, React might re-render incorrectly



```
const fruits = ["Apple", "Banana", "Orange"];

{fruits.map((fruit, index) => (
  <li key={index}>{fruit}</li>
))}
```

Hooks in React

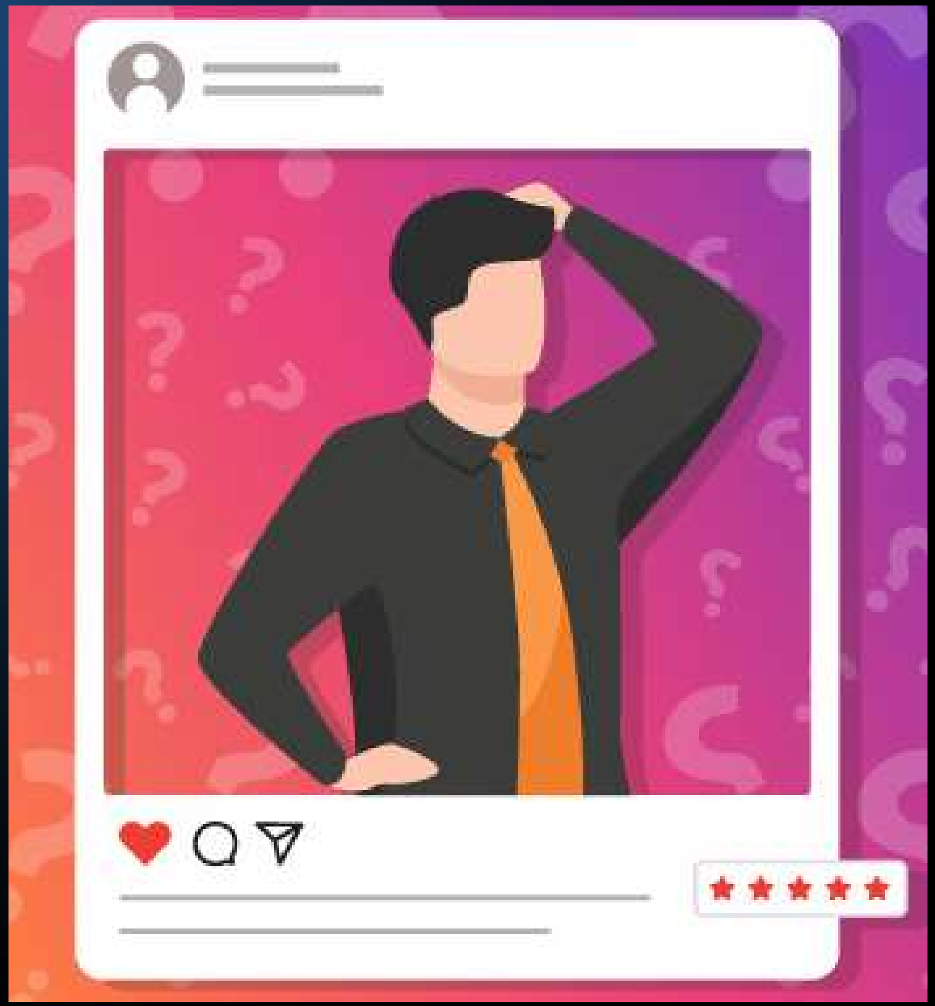
What Are Hooks in React?

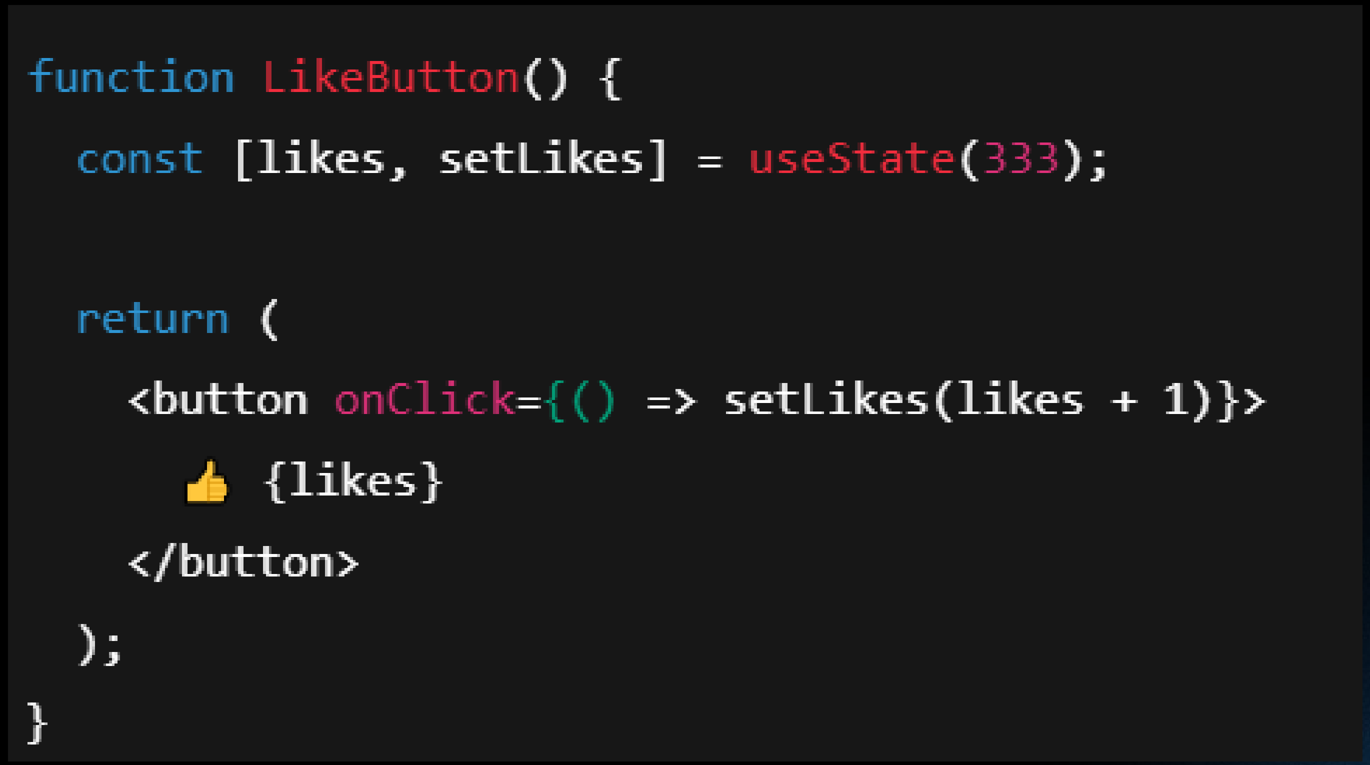
1. Notable React features
   a. Storing Values that can change over time. - state feature
   b. Running Code after component renders - side effects feature
   c. Accessing DOM values - Refs feature
2. **Hooks** are special functions in React that let you use ***state and other React features*** in functional components.
3. Most used **hooks** in React JS
   a. **useState()** - state feature
   b. **useEffect()** - side effects feature
   c. **useRef()** - refs feature
4. **Hooks** should always be called **at the top of the component**, not inside loops or conditions.

useState in React

What is State in React?

1. State is like a variable that lives inside a component and controls how the UI behaves.
2. React components automatically re-render when state changes.
3. Use state when your data can change over time.
4. Example: Instagram post’s like button. The likes count of a particular post changes over time



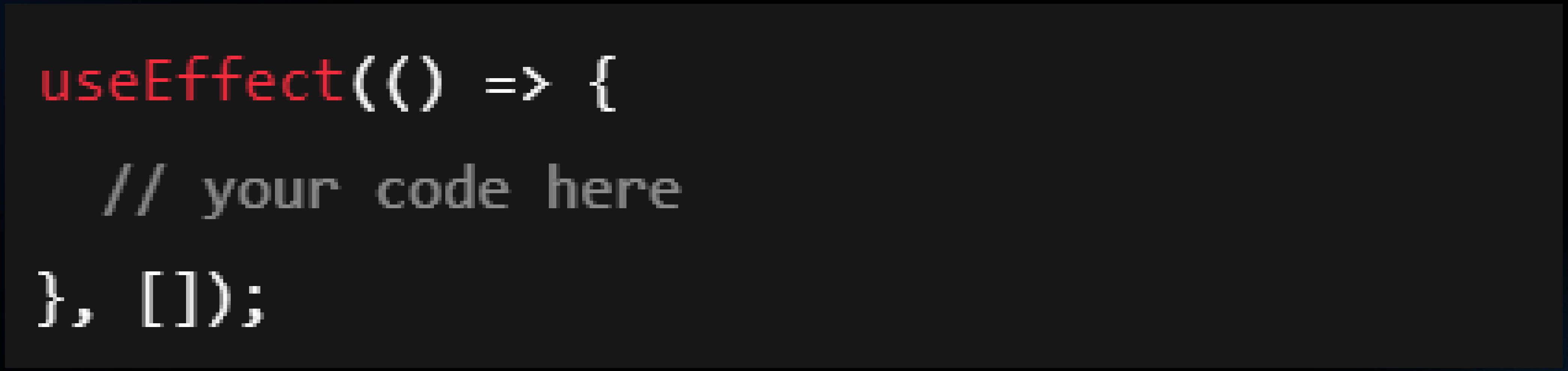


```
function LikeButton() {
  const [likes, setLikes] = useState(333);

  return (
    <button onClick={() => setLikes(likes + 1)}>
      👍 {likes}
    </button>
  );
}
```

useEffect in React

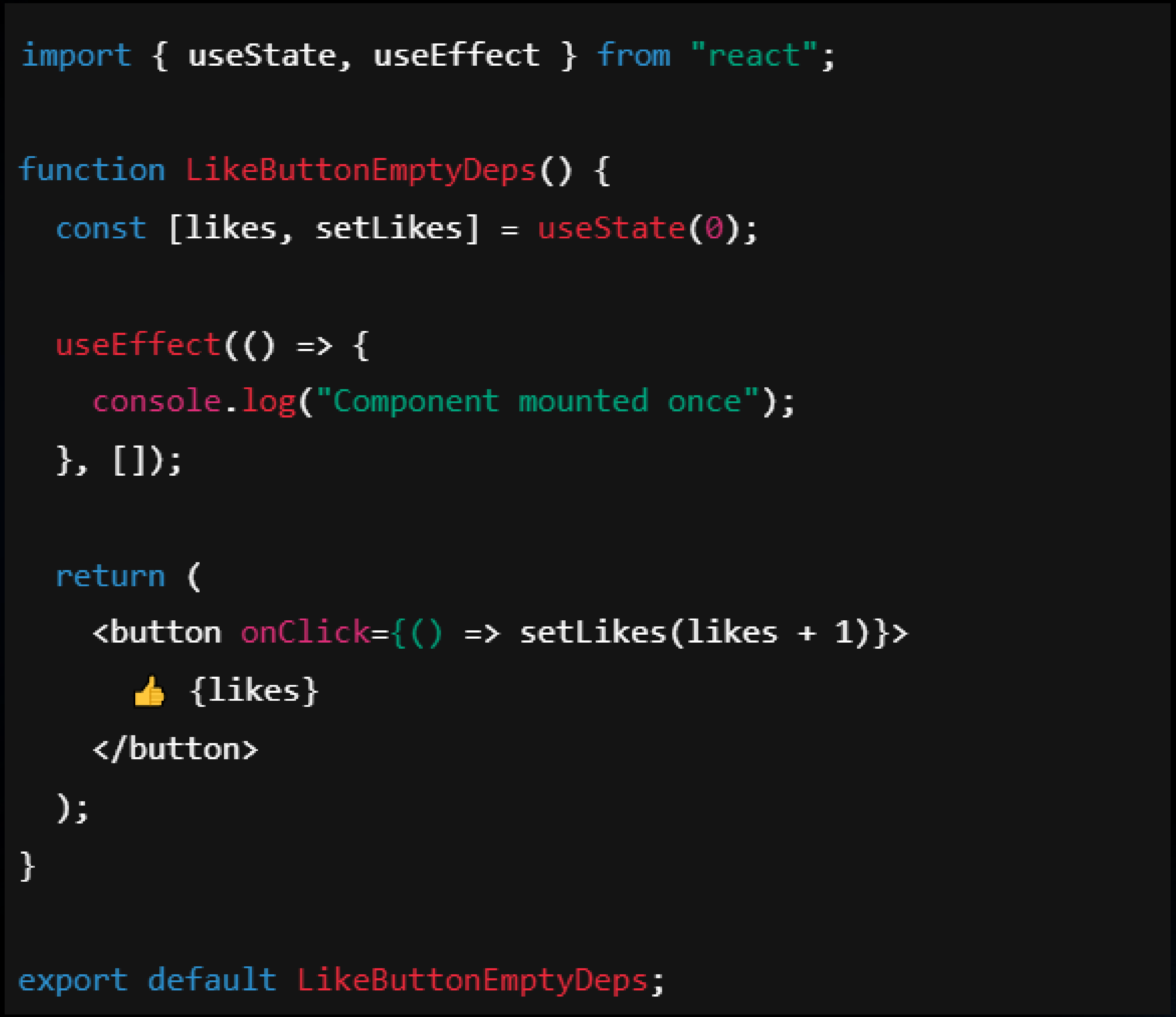
What Is useEffect in React?

1. useEffect() lets you run some code after the component renders.
2. It’s used for:
    a. Fetching data from APIs
    b. Updating the DOM manually
3. The first argument is a function.
4. The second argument (called the dependency array) controls when the effect runs.



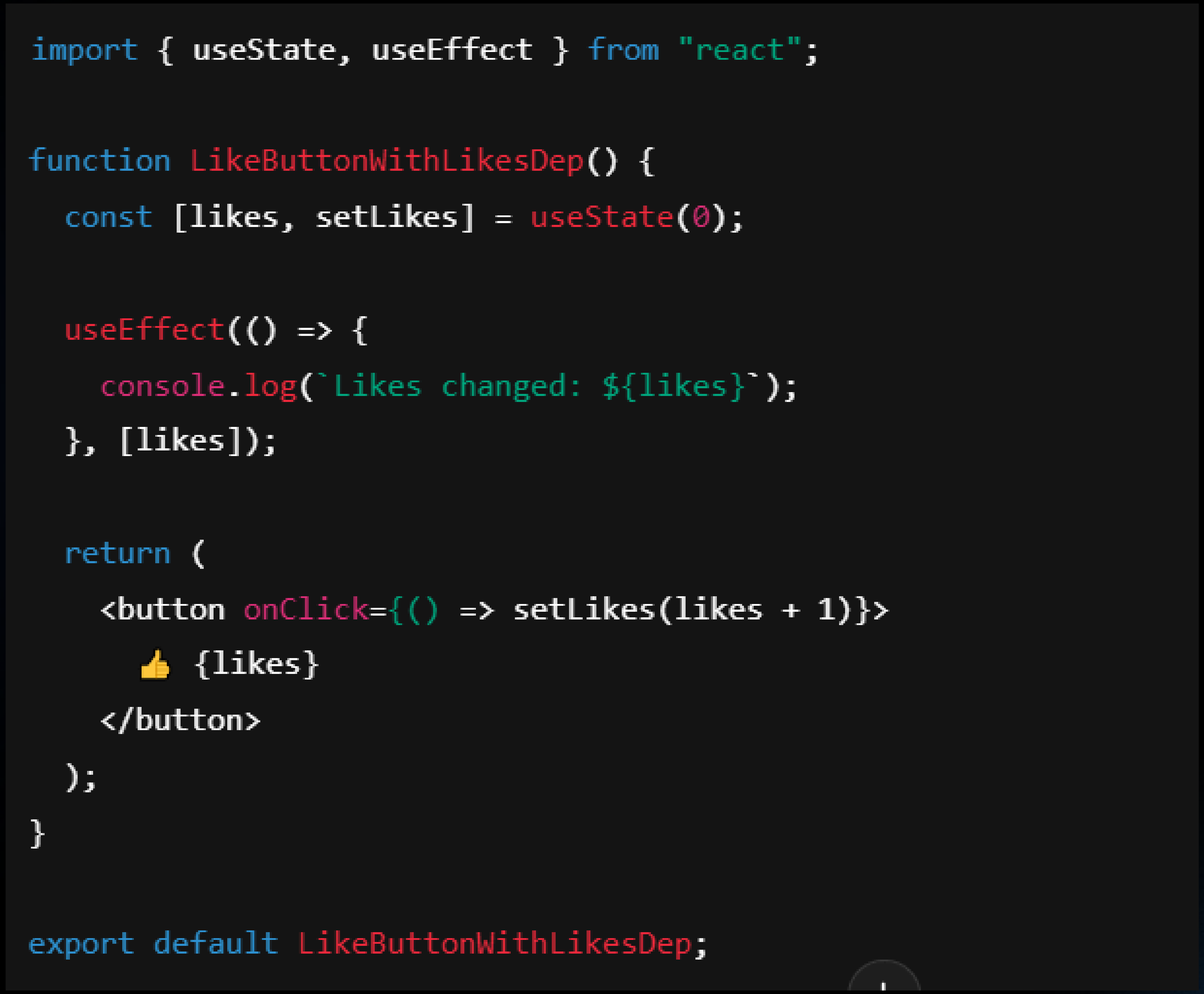
```
useEffect(() => {
  // your code here
}, []);
```

Case 1:



```jsx
import { useState, useEffect } from "react";

function LikeButtonEmptyDeps() {
  const [likes, setLikes] = useState(0);

  useEffect(() => {
    console.log("Component mounted once");
  }, []);

  return (
    <button onClick={() => setLikes(likes + 1)}>
      👍 {likes}
    </button>
  );
}

export default LikeButtonEmptyDeps;
```

Case 2:

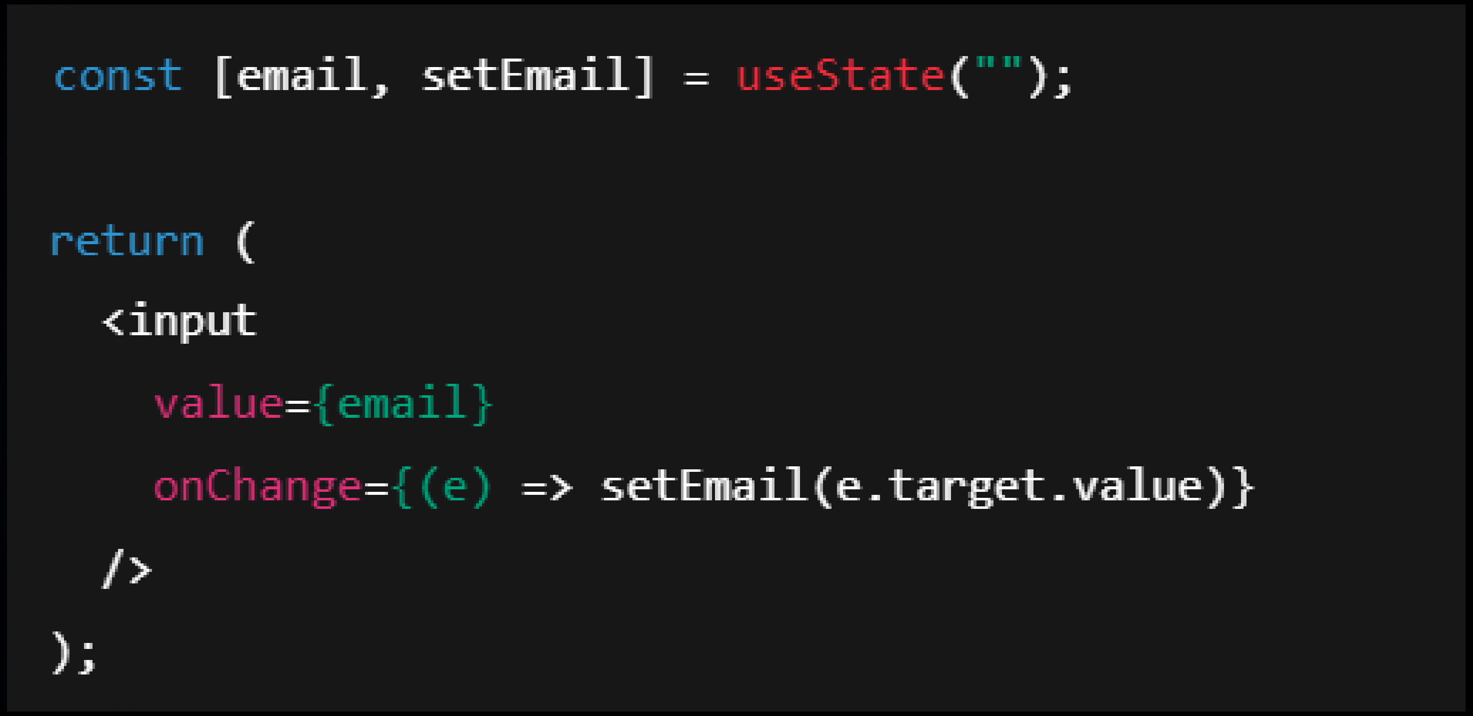


```
import { useState, useEffect } from "react";

function LikeButtonWithLikesDep() {
  const [likes, setLikes] = useState(0);

  useEffect(() => {
    console.log(`Likes changed: ${likes}`);
  }, [likes]);

  return (
    <button onClick={() => setLikes(likes + 1)}>
      👍 {likes}
    </button>
  );
}

export default LikeButtonWithLikesDep;
```

Two-way Binding

What Is two-way binding?

1. Two-way binding means the input field and the state are always in sync.



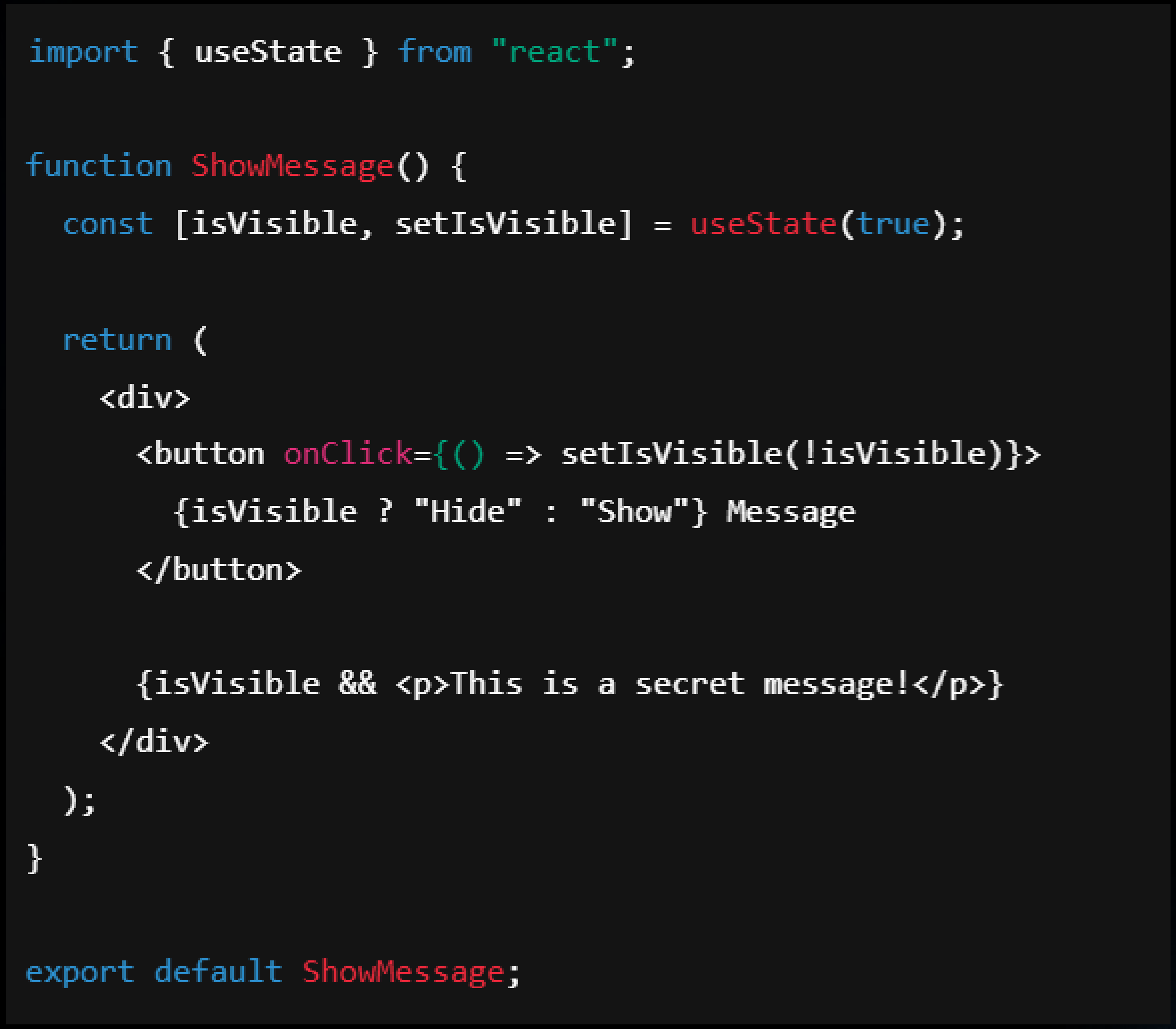
```
const [email, setEmail] = useState("");

return (
  <input
    value={email}
    onChange={(e) => setEmail(e.target.value)}
  />
);
```

1. value={email}: Input shows what's in the state
2. onChange: When user types, we update the state

Conditional Rendering

What is conditional Rendering?

1. It means showing different UI based on a condition (if/else).
2. In React, you can use JavaScript if/else, ternary (? :) or logical AND (&&) to decide what to render.

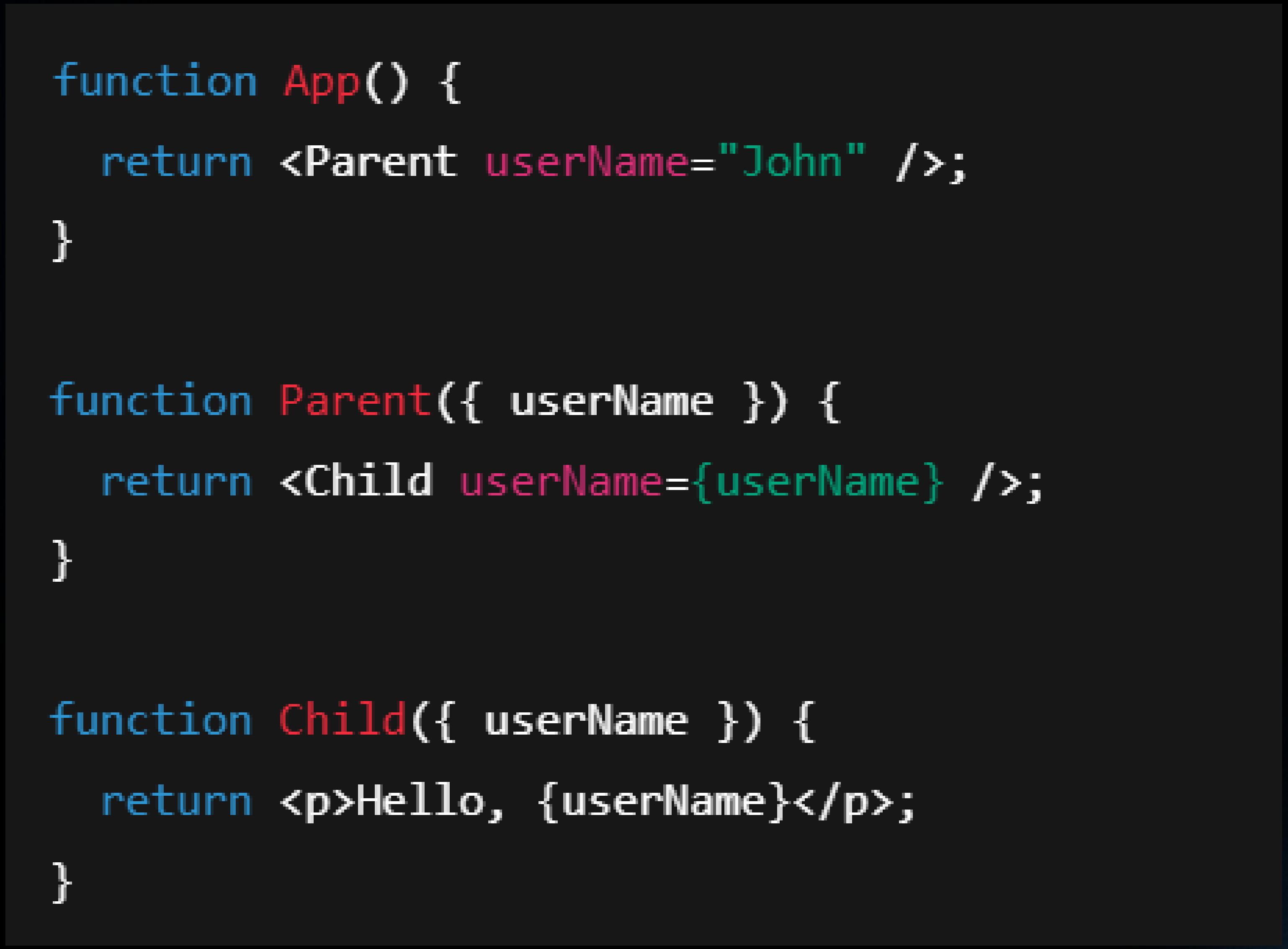


```jsx
import { useState } from "react";

function ShowMessage() {
  const [isVisible, setIsVisible] = useState(true);

  return (
    <div>
      <button onClick={() => setIsVisible(!isVisible)}>
        {isVisible ? "Hide" : "Show"} Message
      </button>

      {isVisible && <p>This is a secret message!</p>}
    </div>
  );
}

export default ShowMessage;
```

Prop Drilling

What is Prop Drilling?

1. Props are used to send data from one component to another (usually from parent to child).
2. Passing props through many layers of components, even when some of them don’t need that prop directly is called prop drilling.



```
function App() {
  return <Parent userName="John" />;
}

function Parent({ userName }) {
  return <Child userName={userName} />;
}

function Child({ userName }) {
  return <p>Hello, {userName}</p>;
}
```

SUBSCRIBE

to Tech Jashwanth