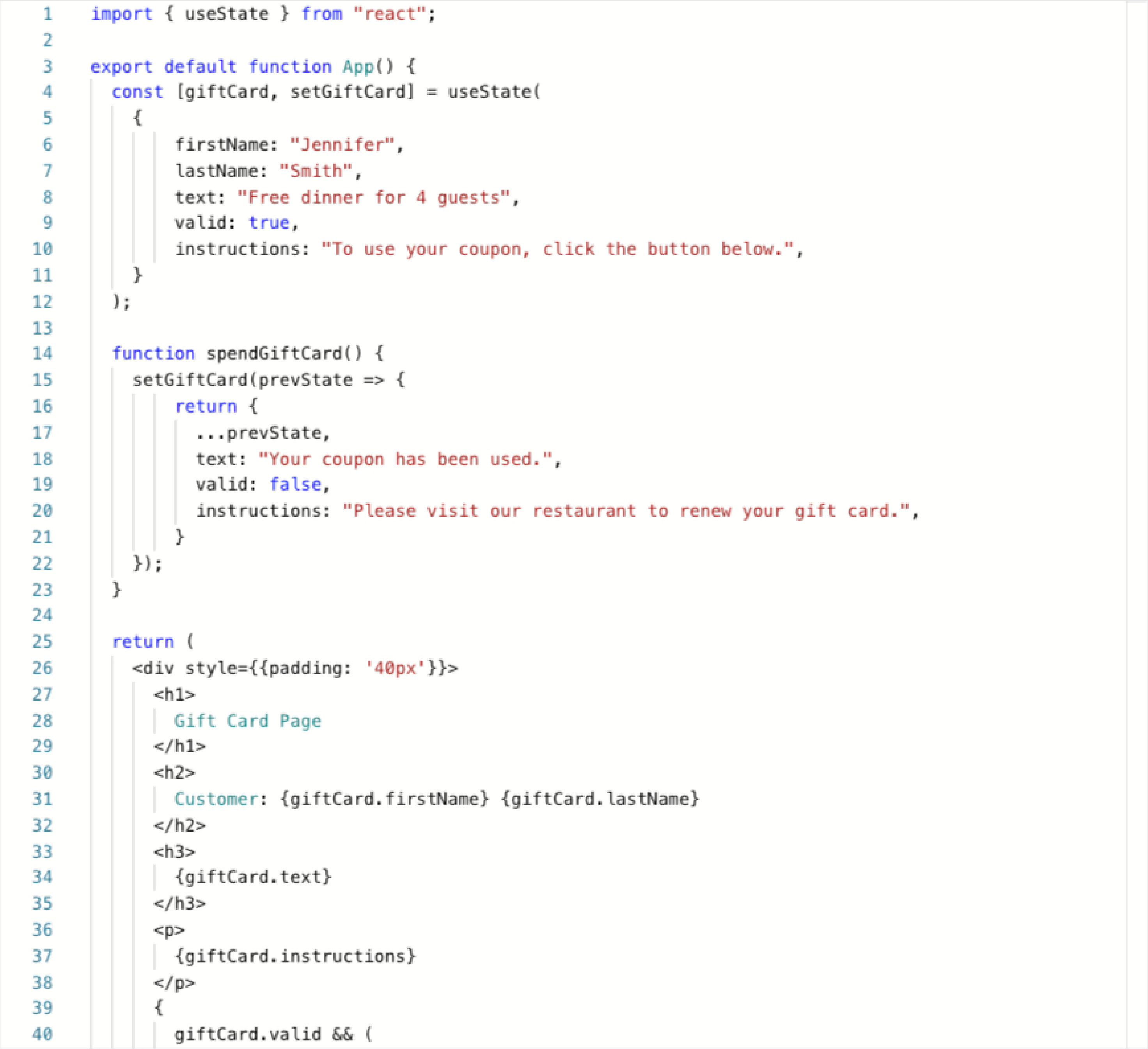
Solution: Managing state within a component

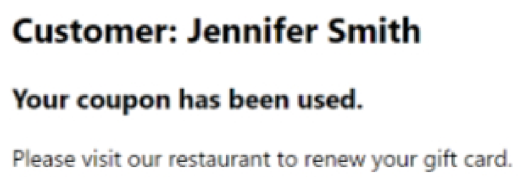
Here is the completed solution code for the **`App.js`** file:



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

export default function App() {
  const [giftCard, setGiftCard] = useState(
    {
      firstName: "Jennifer",
      lastName: "Smith",
      text: "Free dinner for 4 guests",
      valid: true,
      instructions: "To use your coupon, click the button below.",
    }
  );

  function spendGiftCard() {
    setGiftCard(prevState => {
      return {
        ...prevState,
        text: "Your coupon has been used.",
        valid: false,
        instructions: "Please visit our restaurant to renew your gift card.",
      }
    });
  }

  return (
    <div style={{padding: '40px'}}>
      <h1>
        Gift Card Page
      </h1>
      <h2>
        Customer: {giftCard.firstName} {giftCard.lastName}
      </h2>
      <h3>
        {giftCard.text}
      </h3>
      <p>
        {giftCard.instructions}
      </p>
      {
        giftCard.valid && (
```

Here is the output from the solution code for the **`App.js`** file.

Gift Card Page



Customer: Jennifer Smith

Your coupon has been used.

Please visit our restaurant to renew your gift card.

Step-by-step solution

Step 1

You opened the **`App.js`** file and located the **`spendGiftCard()`** function.

Inside the **`spendGiftCard()`** function, you invoked the **`setGiftCard()`** state-updating function, without passing it any parameters or doing anything else with it.

```
function spendGiftCard() {
setGiftCard()
}
```

Step 2

Inside the **`setGiftCard()`** function invocation's parentheses, you passed in an arrow function.

This arrow function has a single parameter, named **`prevState`**. After the arrow, you added a block of code.

```
function spendGiftCard() {
setGiftCard(prevState => {

  })
}
```

Step 3

Next, you returned a copy of the **`prevState`** objectusing the rest operator.

```
function spendGiftCard() {
setGiftCard(prevState => {
return ...prevState
  })
}
```

Step 4

Next, you combined this copy of the **`prevState`** object with those properties that you wanted updated, by updating some of the key-value pairs that already exist on the state object that were initially passed to the **`useState()`** function call.

```
function spendGiftCard() {
setGiftCard(prevState => {
return {
      ...prevState,
text: "Your coupon has been used.",
    }
  })
}
```

Step 5

Finally, you updated the remaining properties on the state object.

You updated the valid key's value to `false`.

Then, updated the instructions key's value to Please visit our restaurant to renew your gift card.

```
function spendGiftCard() {
setGiftCard(prevState => {
return {
        ...prevState,
text: "Your coupon has been used.",
valid: false,
instructions: "Please visit our restaurant to renew your gift card.",
      }
  });
}
```