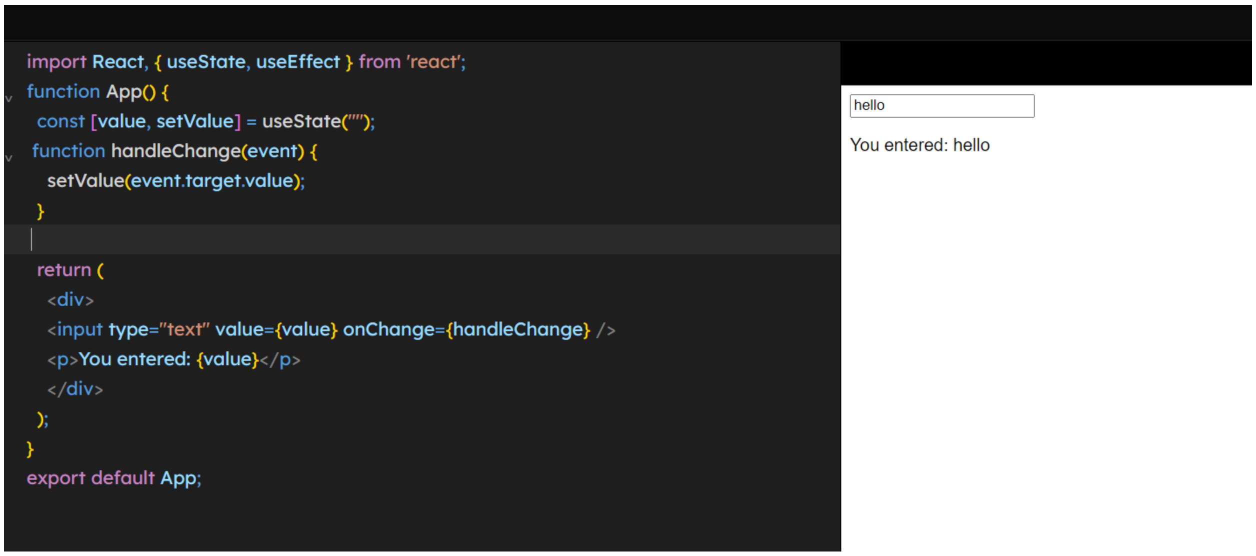
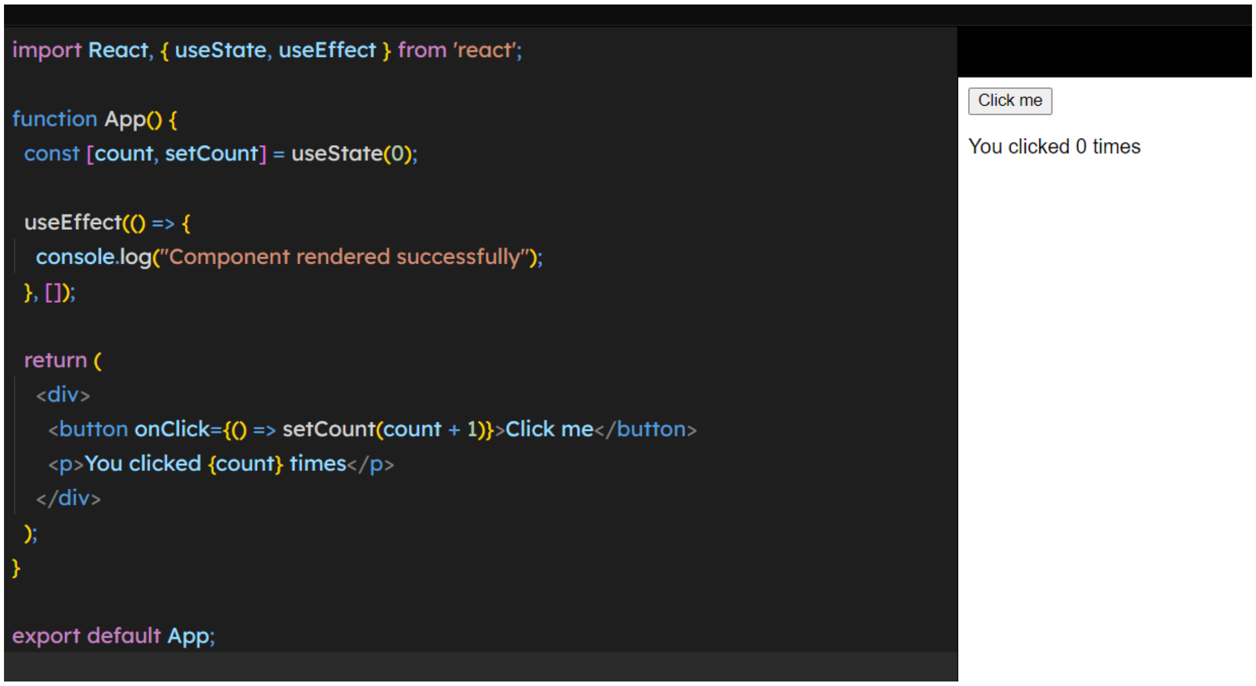


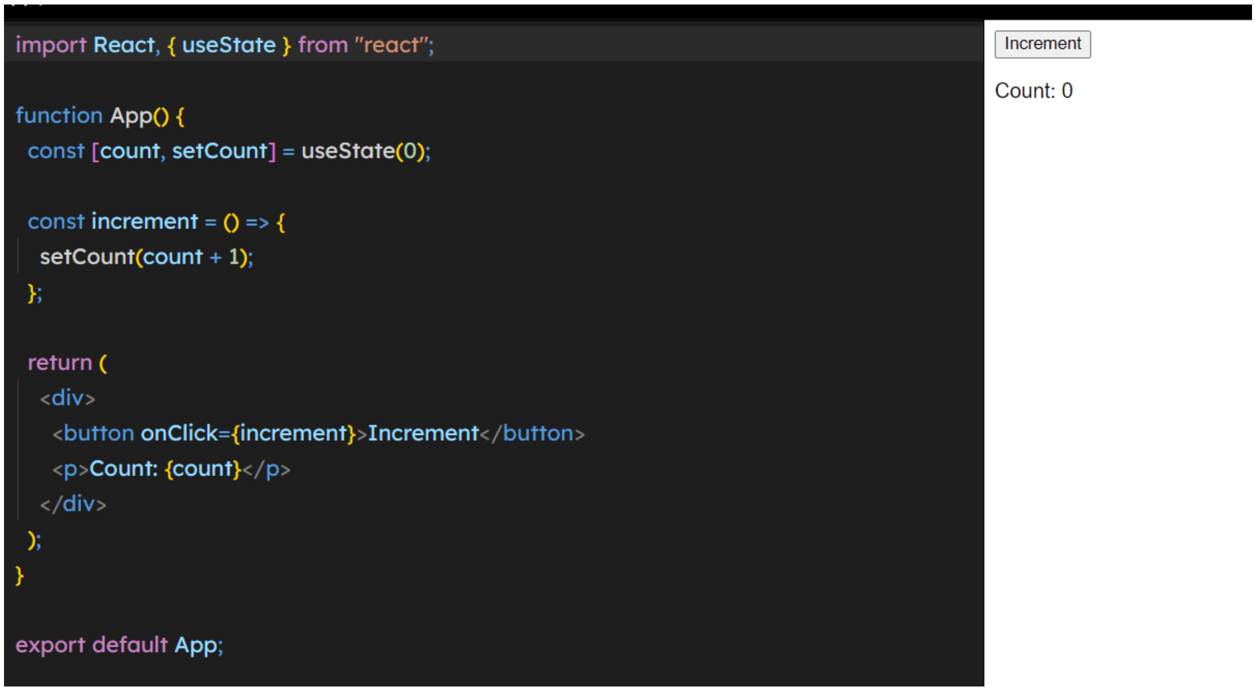
```
import React, { useState, useEffect } from 'react';
function App() {
  const [value, setValue] = useState("");
  function handleChange(event) {
    setValue(event.target.value);
  }

  return (
    <div>
    <input type="text" value={value} onChange={handleChange} />
    <p>You entered: {value}</p>
    </div>
  );
}
export default App;
```

```
hello

You entered: hello
```



```
import React, { useState, useEffect } from 'react';

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

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

  return (
    <div>
      <button onClick={() => setCount(count + 1)}>Click me</button>
      <p>You clicked {count} times</p>
    </div>
  );
}

export default App;
```

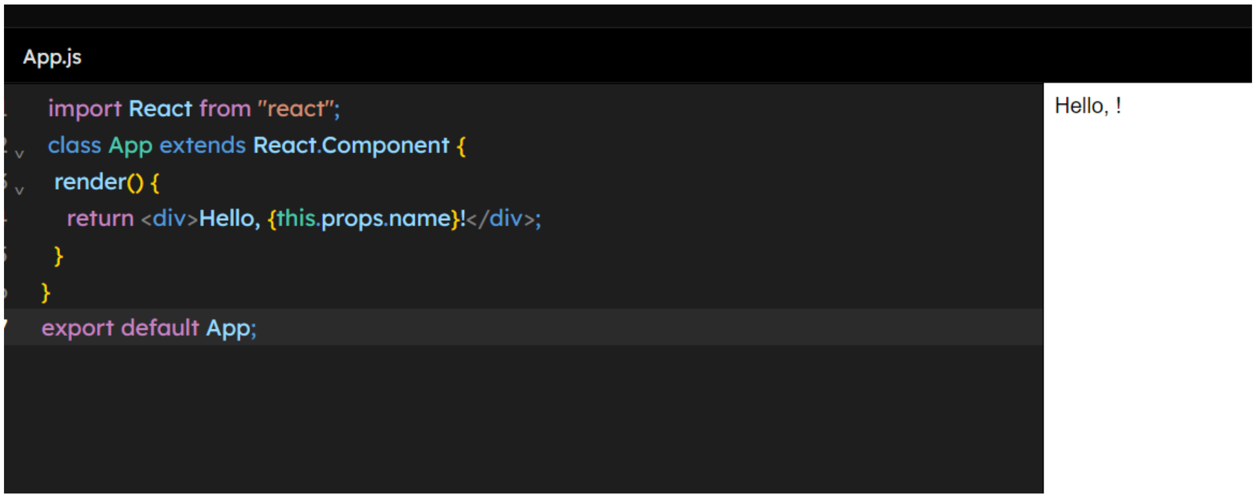
```
Click me

You clicked 0 times
```



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

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

  const increment = () => {
    setCount(count + 1);
  };

  return (
    <div>
      <button onClick={increment}>Increment</button>
      <p>Count: {count}</p>
    </div>
  );
}

export default App;
```

```
Increment

Count: 0
```

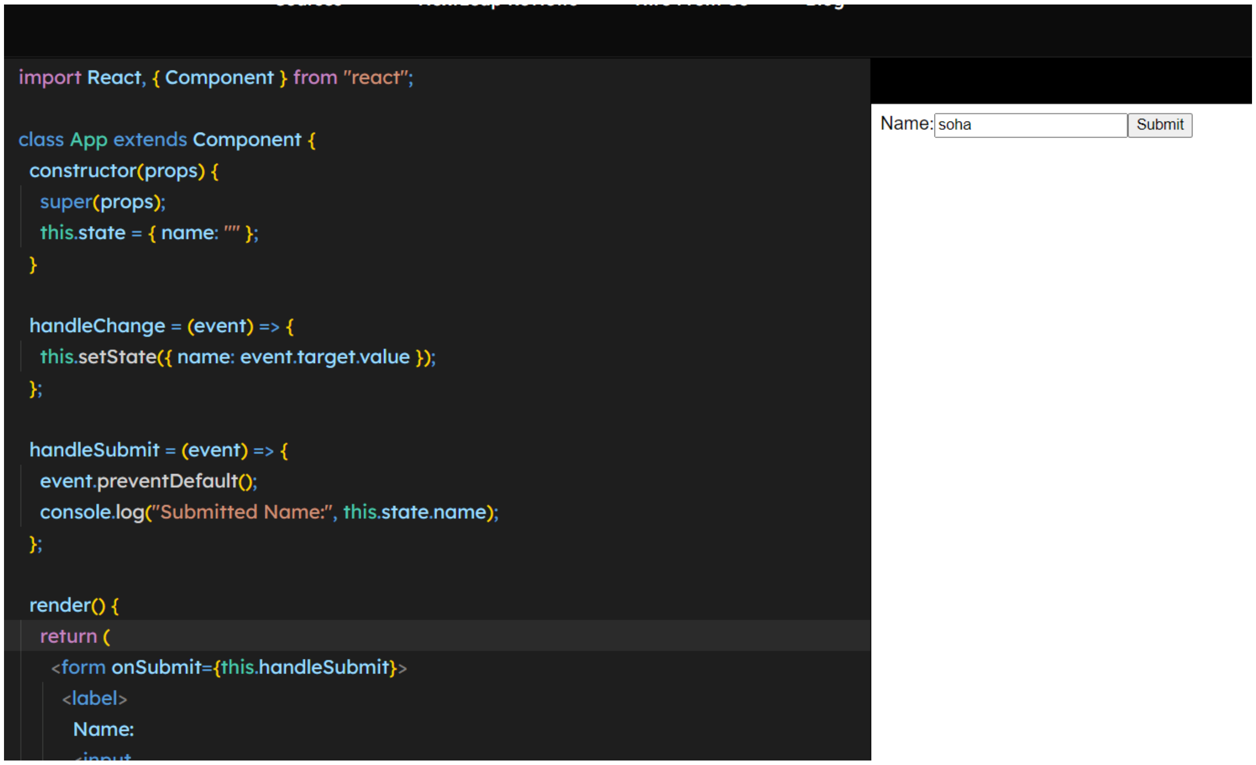
Analyze the below code snippet and advise what will be shown on the screen when the App component is rendered with *<App name=”Claire” />?*



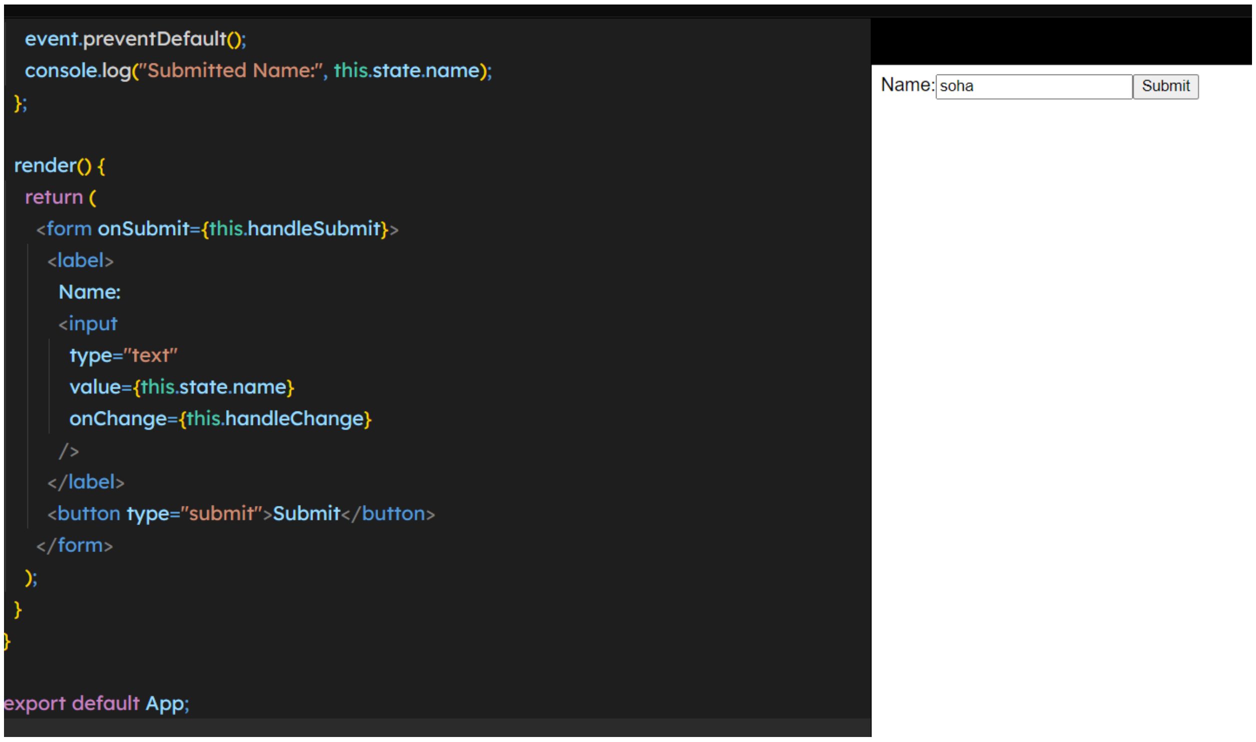
App.js

```
 import React from "react";
 class App extends React.Component {
  render() {
    return <div>Hello, {this.props.name}!</div>;
  }
 }
export default App;
```

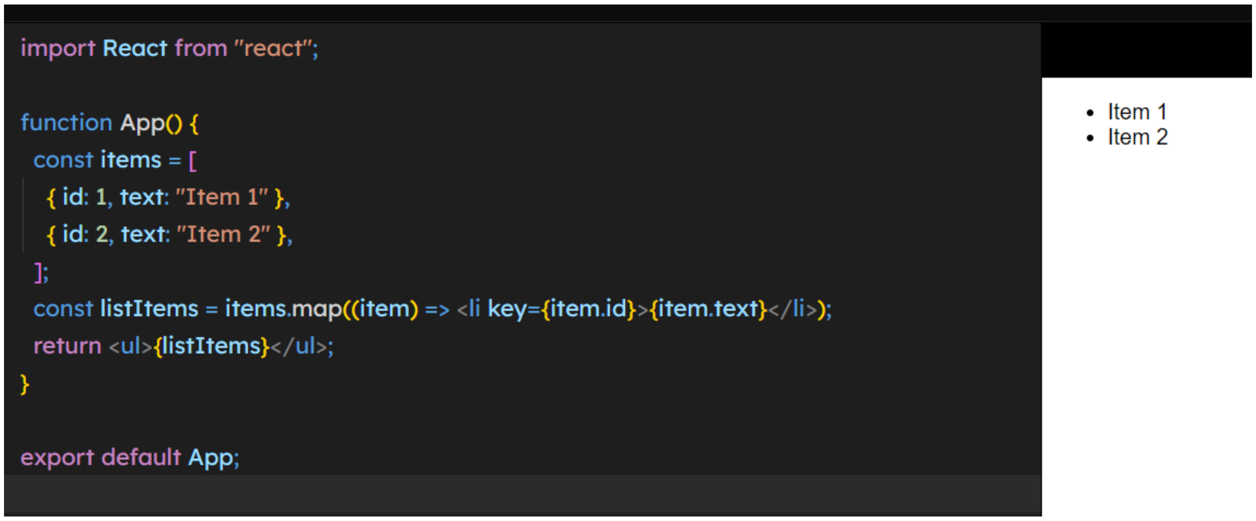
```
Hello, !
```



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

class App extends Component {
  constructor(props) {
    super(props);
    this.state = { name: "" };
  }

  handleChange = (event) => {
    this.setState({ name: event.target.value });
  };

  handleSubmit = (event) => {
    event.preventDefault();
    console.log("Submitted Name:", this.state.name);
  };

  render() {
    return (
      <form onSubmit={this.handleSubmit}>
        <label>
          Name:
          <input
```



```
  event.preventDefault();
  console.log("Submitted Name:", this.state.name);
};

render() {
  return (
    <form onSubmit={this.handleSubmit}>
      <label>
        Name:
        <input
          type="text"
          value={this.state.name}
          onChange={this.handleChange}
        />
      </label>
      <button type="submit">Submit</button>
    </form>
  );
}
}

export default App;
```



```
import React from "react";

function App() {
  const items = [
    { id: 1, text: "Item 1" },
    { id: 2, text: "Item 2" },
  ];
  const listItems = items.map((item) => <li key={item.id}>{item.text}</li>);
  return <ul>{listItems}</ul>;
}

export default App;
```

- Item 1
- Item 2

Create a stopwatch application through which users can start, pause and reset the timer. Use React state, event handlers and the *setTimeout* or *setInterval* functions to manage the timer's state and actions.

```
import React, { useState, useEffect, useRef } from 'react';


function Stopwatch() {
  const [time, setTime] = useState(0); // Time in seconds
  const [isRunning, setIsRunning] = useState(false); // Timer status
  const intervalRef = useRef(null); // Reference to the interval


  // Function to start the timer
  const startTimer = () => {
    if (!isRunning) {
      setIsRunning(true);
      intervalRef.current = setInterval(() => {
        setTime((prevTime) => prevTime + 1);
      }, 1000);
    }
```

```
};

// Function to pause the timer
const pauseTimer = () => {
  if (isRunning) {
    setIsRunning(false);
    clearInterval(intervalRef.current);
  }
};

// Function to reset the timer
const resetTimer = () => {
  setIsRunning(false);
  clearInterval(intervalRef.current);
  setTime(0);
};

// Clean up interval on component unmount
useEffect(() => {
  return () => clearInterval(intervalRef.current);
}, []);

// Convert time in seconds to MM:SS format
const formatTime = (time) => {
  const minutes = Math.floor(time / 60);
```

```
    const seconds = time % 60;
    return `${String(minutes).padStart(2, '0')}:${String(seconds).padStart(2, '0')}`;
  };

  return (
    <div>
      <h1>Stopwatch</h1>
      <div>
        <h2>{formatTime(time)}</h2>
      </div>
      <div>
        <button onClick={startTimer} disabled={isRunning}>Start</button>
        <button onClick={pauseTimer} disabled={!isRunning}>Pause</button>
        <button onClick={resetTimer}>Reset</button>
      </div>
    </div>
  );
}

export default Stopwatch;
```