Lesson 02 Demo 04

Create a React Application Using Conditional Rendering

Objective: To create an application so that counter value increases on each click, but the value remains an even number

Tools Required: Node terminal, React app, and Visual Studio Code

Prerequisites: Knowledge of creating a React app and understanding of the folder structure

Steps to be followed:

1. Create a new **React app**
2. Implement the **Number** component
3. Implement the **App** component
4. Render the **App** component
5. Run the app

Step 1: Create a new React app

1.1 Start by creating a new React app using the **create-react-app** command in your terminal: **npx create-react-app my-app2**

```
shreemayeebhatt@ip-172-31-22-250:~$ npx create-react-app my-app2
```

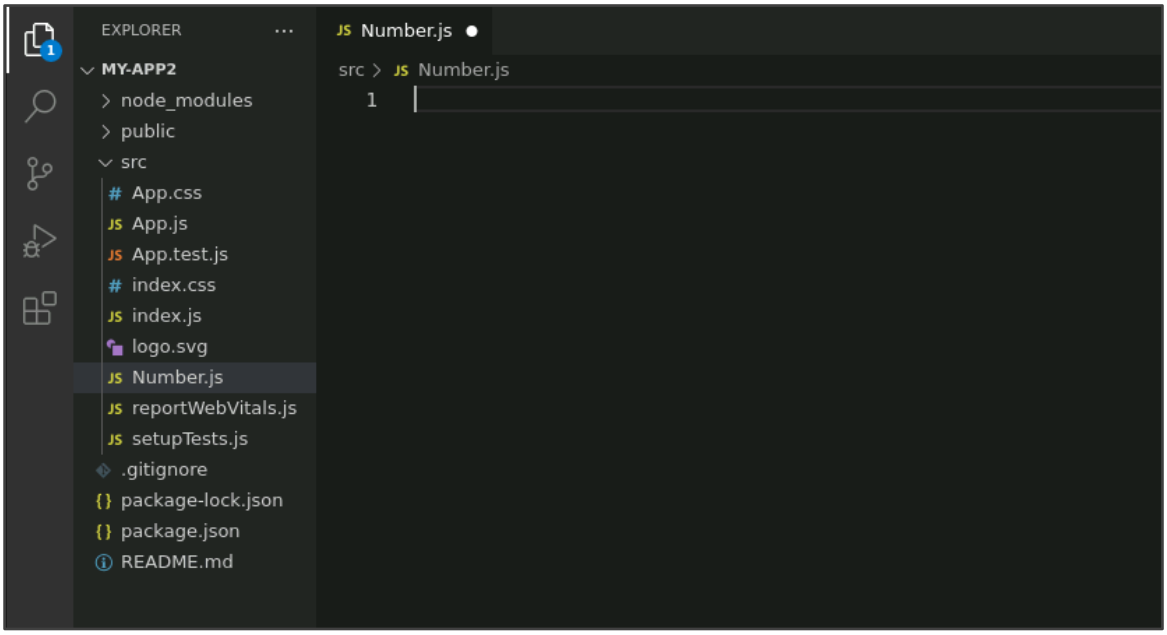
1.2 Move to the newly created directory by running the command **cd my-app2**

```
shreemayeebhatt@ip-172-31-22-250:~$ cd my-app2
```

1.3 Open **Visual Studio Code** and navigate to the project directory

Step 2: Implement the Number component

2.1 In the **src** directory, create a new file called **Number.js**



2.2 In the **Number.js** file, import **React** and define a class component called **Number**

2.3 Implement the **constructor** method inside the **Number** class, which accepts props and calls **super(props)**

```
import React from "react";

class Number extends React.Component {
    constructor(props) {
      super(props);
    }
```

2.4 Implement the **componentDidUpdate** lifecycle method and log a message to the console: **console.log(componentDidUpdate)**

```
componentDidUpdate() {
  console.log('componentDidUpdate');
}
```

2.5 Implement the **render** method and check if the **number** prop uses **(this.props.number % 2 === 0)**

2.6 If the number is even, return a **JSX** element with a heading tag displaying the number: **<h1>{this.props.number}</h1>**

2.7 If the number is odd, return **null**

```
    render() {
      if(this.props.number % 2 == 0) {
          return (
              <div>
                  <h1>{this.props.number}</h1>
              </div>
          );

  } else {
      return null;
    }
  }
}
```

src > JS Number.js > Number > render

```
1   import React from "react";
2
3   class Number extends React.Component {
4       constructor(props) {
5         super(props);
6       }
7
8       componentDidUpdate() {
9         console.log('componentDidUpdate');
10      }
11
12      render() {
13        if(this.props.number % 2 == 0) {
14            return (
15                <div>
16                    <h1>{this.props.number}</h1>
17                </div>
18            );
19
20    } else {
21        return null;
22      }
23    }
24
25    export default Number;
26
```

```
//Number.js
import React from "react";

class Number extends React.Component {
   constructor(props) {
    super(props);
   }

   componentDidUpdate() {
    console.log('componentDidUpdate');
   }

   render() {
    if(this.props.number % 2 == 0) {
       return (
          <div>
             <h1>{this.props.number}</h1>
          </div>
       );

  } else {
      return null;
    }
  }
}

 export default Number;
```

Step 3: Implement the App component

3.1 In the **src** directory, modify the existing file called **App.js**

3.2 Import **React** and modify the class component named **App**

3.3 Implement the constructor method inside the **App** class, accepting **props** and calling **super(props)**

3.4 Inside the constructor, initialize the **count** state property to 0: **this.state = { count: 0}**

```
class App extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 }
  }
```

3.5 Implement the **onClick** method, which updates the **count** state by incrementing its value:

```
onClick = () => {
        this.setState(prevState => ({
        count: prevState.count + 1
}));
};
```

```
  onClick(e) {
    this.setState(prevState => ({
      count: prevState.count + 1
    }));
  }
```

3.6 In the **render** method, return a **JSX** element containing:

- An instance of the **Number** component, with the number prop set to the **count** state value: **<Number number={this.state.count} />**
- A button with an **onClick** event handler that calls the **onClick** method when clicked: **<button onClick={this.onClick.bind(this)}>Count</button>**

```
render() {
    return (
      <div>
        <Number number={this.state.count} />
        <button onClick={this.onClick.bind(this)}>Count</button>
      </div>
    )
  }
}
```

```
//App.js
import logo from './logo.svg';
import './App.css';
import Number from './Number'
import React from 'react';

class App extends React.Component {
constructor(props) {
super(props);
this.state = { count: 0 }
}

onClick(e) {
this.setState(prevState => ({
count: prevState.count + 1
}));
}
render() {
   return (
    <div>
      <Number number={this.state.count} />

   <button onClick={this.onClick.bind(this)}>Count</button>
```

```
      </div>
    )
  }
}

export default App;
```

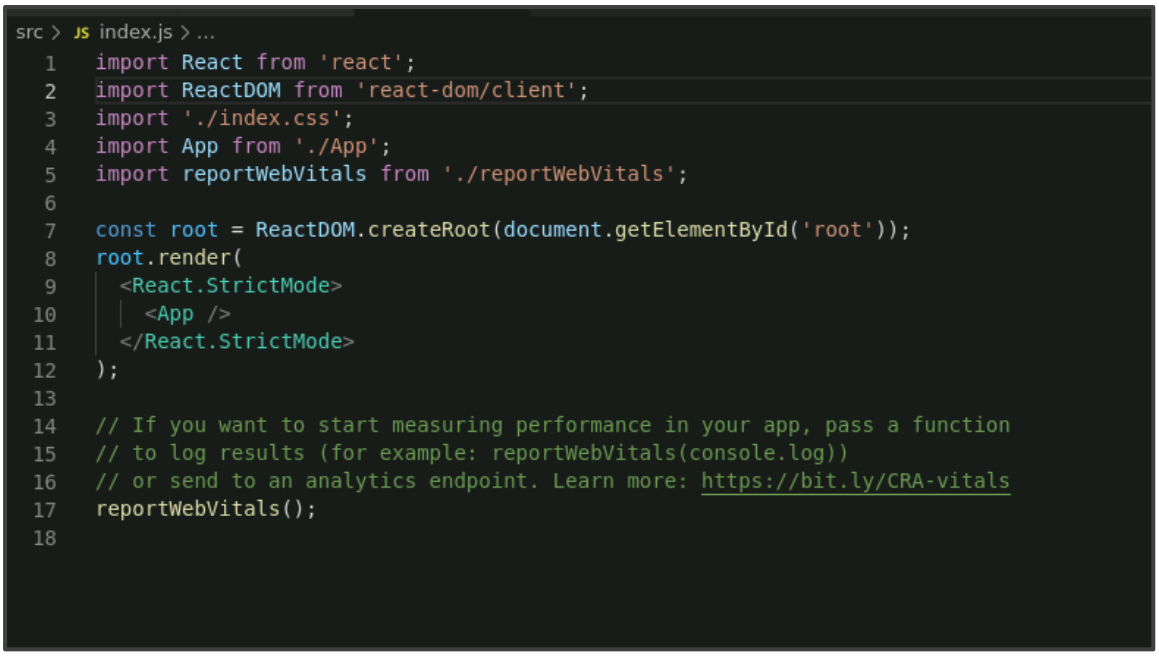
Step 4: Render the App component

4.1 Open the **index.js** file in the **src** directory

4.2 Import **React** and **ReactDOM**

4.3 Replace the **ReactDOM.render** line with the following code to render the **App** component:

```
ReactDOM.render
<React.StrictMode>
<App />
</React.StrictMode>,
 document.getElementById('root')
);
```



```
src > JS index.js > ...
1   import React from 'react';
2   import ReactDOM from 'react-dom/client';
3   import './index.css';
4   import App from './App';
5   import reportWebVitals from './reportWebVitals';
6
7   const root = ReactDOM.createRoot(document.getElementById('root'));
8   root.render(
9     <React.StrictMode>
10      <App />
11    </React.StrictMode>
12  );
13
14  // If you want to start measuring performance in your app, pass a function
15  // to log results (for example: reportWebVitals(console.log))
16  // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
17  reportWebVitals();
18
```

```
//index.js
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
 <React.StrictMode>
   <App />
 </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

Step 5: Run the app

5.1 In your terminal, navigate to the project's root directory

5.2 Run the command **npm start** to start the development server

5.3 Open your browser and navigate to **http://localhost:3000**

5.4 You should see the app with a number displayed and a button that increments the count when clicked