Lesson 02 Demo 03

Creating a React Application Using Event Handler

Objective: To develop a React component that binds its event handler context

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 MyList component
3. Render the MyList component
4. Run the app

Step 1: Create a new React app

1.1 Create a new React app using the **create-react-app** command in your terminal:
npx create-react-app my-app1

```
shreemayeebhatt@ip-172-31-22-250:~$ npx create-react-app my-app1
```

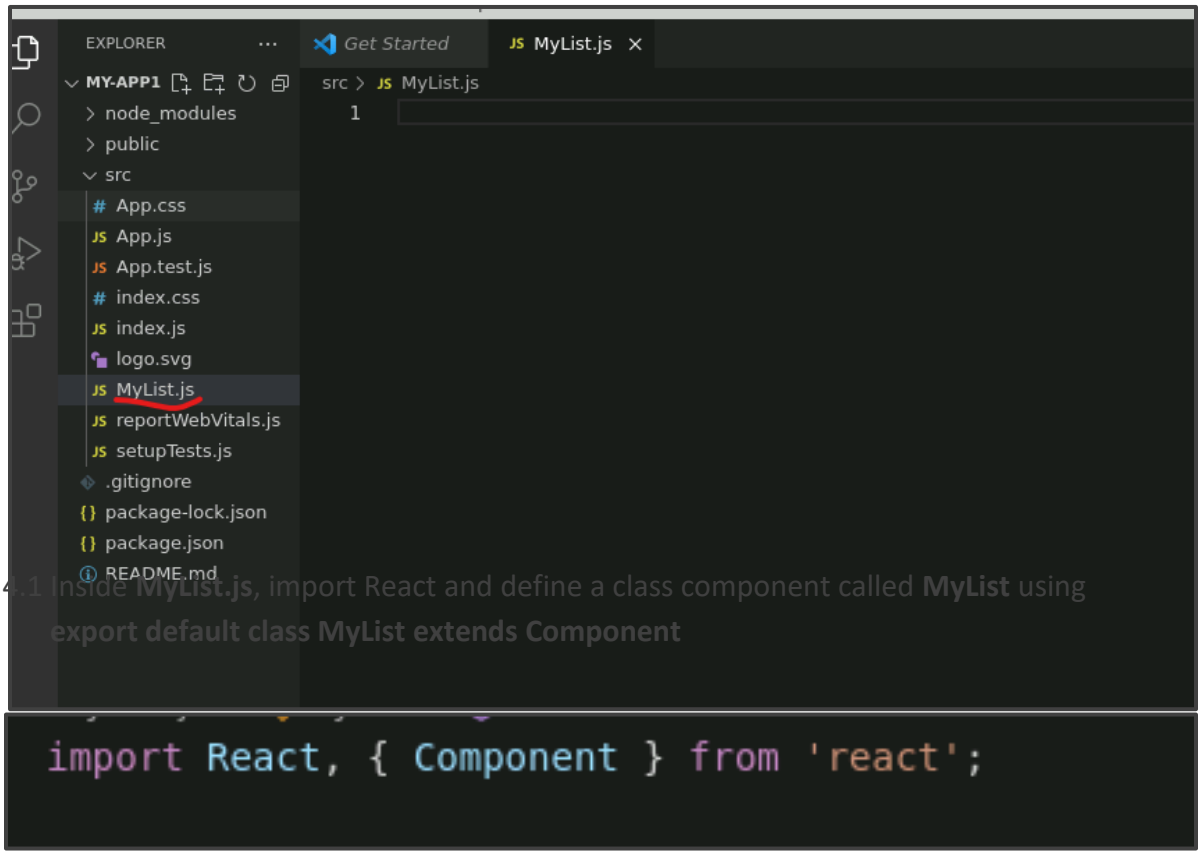
1.2 Move to the newly created directory by running the **cd my-app1** command in the terminal:

```
shreemayeebhatt@ip-172-31-22-250:~$ cd my-app1
```

Step 2: Implement the MyList component

2.1 Open the preferred code editor and navigate to the project directory

2.2 In the **src** directory, create a new file called **MyList.js**



2.3 Implement the constructor method inside the **MyList** class using the **constructor()**

2.4 Call **super()** to invoke the parent class constructor

2.5 In the constructor, bind the **onClick** method to the component's context using **this.onClick = this.onClick.bind(this);**

```
export default class MyList extends Component {
  constructor() {
    super();
    this.onClick = this.onClick.bind(this);
  }
```

2.6 Implement the **onClick method**, which takes an **id** argument and logs the name of the clicked item based on the **id** using the **console.log('clicked', `"${name}"`)**

```
onClick(id) {
    const { name } = this.props.items.find(i => i.id === id);
    console.log('clicked', `"${name}"`);
}
```

2.7 Implement the **render** method, which returns a **JSX** element representing the list using the **ul** and **li** tags

2.8 In the **render** method, map over the **items** prop using **this.props.items.map()** and create a **li** element for each item

2.9 Assign a unique **key** to each **li** element using **key={id}**

2.10 Attach an **onClick** event handler to each **li** element using **onClick={this.onClick.bind(null, id)}**

2.10 Display the **name** of each item inside the **li** element

```
render() {
  return (
    <ul>
      {/* Creates a new handler function with
          the bound "id" argument. Notice that
          the context is left as null since that
          has already been bound in the
          constructor. */}
      {this.props.items.map(({ id, name }) => (
        <li key={id} onClick={this.onClick.bind(null, id)}>
          {name}
        </li>
      ))}
    </ul>
  );
}
}
```

Step 3: Render the MyList component

3.1 Open the **index.js** file in the **src** directory
3.2 Import **React** and **{ render }** from **react-dom**
3.3 Import the **MyList** component from **./MyList**
3.4 Create an array of items to pass to the **MyList** component: **const items = [/* ... */];**
3.5 Use the **render** function to render the **MyList** component with the **items** prop: **render(<MyList items={items} />, document.getElementById('root'));**

```
JS index.js > ...
1  import React from 'react';
2  import {render} from 'react-dom';
3  import './index.css';
4  import MyList from './MyList';
5
6  const items = [
7    { id: 0, name: 'First' },
8    { id: 1, name: 'Second' },
9    { id: 2, name: 'Third' }
0  ];
1
2  render(<MyList items={items} />, document.getElementById('root'));
3
```

Step 4: Run the app

4.1 In the terminal, navigate to the project's root directory
4.2 Run the **npm start** command to start the development server

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

shreemayeebhatt@ip-172-31-22-250:~/my-app1$ npm start
```

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

You should see the app with a list of items rendered.

By following these steps, you have successfully developed a React application utilizing an event handler within the MyList component, demonstrating the binding of the event handler context for efficient handling of item clicks.