

## COMP 3133 – Full Stack Development – Lab 9

- Component State & Events
- Using Axios with React

### Developer Note:

- Work can be done in the same create-react-app application. Remember to not include node\_modules in the GitHub submission.

### References:

<http://jsonplaceholder.typicode.com/>

<https://github.com/axios/axios>

## Exercise 1 – Working with Component Data & Axios

### GET Request

1. Install Axios using npm install command.

```
npm install axios --save
```

2. Create a file StudentList.js and use the React snippet command cc + tab to create the following Class component.

```
class UserList extends React.Component {  
  state = { }  
  
  render() {  
    return ( );  
  }  
}  
  
export default UserList;
```

3. Import the the axios library in the **UserList** class component.

```
import axios from "axios";
```

4. Update the components internal state to store an array of Users.

```
state = {  
  users: []  
};
```

5. Add an internal Component LifeCycle method to make a **HTTP GET** request to jsonplaceholder and get a list of User to display after the component has been rendered.

- Add **componentDidMount()** method and make the following call.
- In the response of the GET request update the users in State using **this.setState**

```
axios.get(`https://jsonplaceholder.typicode.com/users`).then(res => {  
  const users = res.data;  
});
```

6. In the render method of the component, iterate over the list of users in state and output the user names.
7. In the **App.js**, remove the starter code from **react-create-app**. Import the **StudentList** component and render it with the following expected output.

- Leanne Graham
- Ervin Howell
- Clementine Bauch
- Patricia Lebsack
- Chelsey Dietrich
- Mrs. Dennis Schulist
- Kurtis Weissnat
- Nicholas Runolfsdottir V
- Glenna Reichert
- Clementina DuBuque

## POST Request

1. Create a Component class named **AddStudent** and import the Axios library
2. Add a State object that contains the user name we wish to Add

```
state = {
  name: ""
};
```

3. Add two events **handleChange** and **handleSubmit**.

```
handleChange = event => { };

handleSubmit = event => { };
```

4. Add the following form markup to the Control's render method.

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

5. Update the **handleChange** method to update the user name in the Component state object.
6. Update the **handleSubmit** method to send a **POST** request to the url below with the user name to add. The response should be outputted to the console.

```
https://jsonplaceholder.typicode.com/users
```

Note: in the **handleSubmit** we use the **event.preventDefault** as the first line in the method, to override the default behavior of the click event.

```
event.preventDefault();
```

7. Import the **AddStudent** component in the **App.js** and render it so that the following is visible in the browser.

Person Name:

- Leanne Graham
- Ervin Howell
- Clementine Bauch
- Patricia Lebsack
- Chelsey Dietrich
- Mrs. Dennis Schulist
- Kurtis Weissnat
- Nicholas Runolfsdottir V
- Glenna Reichert
- Clementina DuBuque

8. The following response should be logged to the console after a **POST** call to add the person name.

```
                                AddStudent.js:25
▼ {user: {...}, id: 11} ⓘ
  id: 11
  ▶ user: {name: "Jake The Snake"}
  ▶ __proto__: Object
```

## **DELETE Request**

1. Create a Component class named **DeleteStudent**. Copy the code from the **AddStudent.js** and modify the **POST** request to be a **DELETE** request to the following url. The internal state object will track the user Id to delete, not the name. Place this in the App.js render method for testing.

```
https://jsonplaceholder.typicode.com/users/${this.state.id}
```

Person ID:

### **Output**

```
                                DeleteStudent.js:19
▶ {data: {...}, status: 200, statusText: "", header
  s: {...}, config: {...}, ...}
▶ {}                                DeleteStudent.js:20
```

2. Once the **DeleteStudent** component is working, add it as a child component in the **StudentList** component. It will be passed the user id as it's props and handle the Delete action by calling the **DELETE** Request with the given id.

Person Name:

- Leanne Graham
- Ervin Howell
- Clementine Bauch
- Patricia Lebsack
- Chelsey Dietrich
- Mrs. Dennis Schulist
- Kurtis Weissnat
- Nicholas Runolfsdottir V
- Glenna Reichert
- Clementina DuBuque

### **Homework**

Add the **AddUser** component as a nested child component in the **StudentList** component. Once a new User has been added trigger a refresh GET Request in the parent **StudentList**.

Hint: This can be done by passing a event handler reference in the props to the child AddUser component. There will be some extra binding to be done to the function handler to make this work.

### **Challenge**

Implement the Delete user to update the list of Students after the student has been deleted from the list.