ITMD 465/565

Project 2

Introduction
This project will have you create a simple React component to add to an existing web page.

Objective
Use the React JS framework to build a simple React component. This React component's behavior will be very similar to Lab 3 where you used AJAX to retrieve the employee list data from a JSON API endpoint.

Requirements
- Any HTML tags and page structure should be **HTML5** compliant
- Load React JS library in browser using the react and reactDOM cnd script tag links.
- Use Babel JS to translate JSX to ES5 JS – Use in browser prototyping method. Link to the babel library using the babel script tag
- Write components using JSX syntax
- Your JavaScript needs to be in a separate file and included in your html file at the bottom of the body section.
- Create a **PageWrapper** component to be the main wrapping element that all your other components will be included in. This PageWrapper component will be the one inserted into the page in the div with id="root". The page wrapper component will have the EmployeesTable as a child of it and any other components necessary.
- Create an **EmployeesTable** component that is the outer wrapper for our data table. Make sure you create a proper html table including column headers.
- Create components for **EmployeesTableRow** and use this component in to render rows in the employees table.
- Use AJAX to get the data that loads into the table from: http://libertyville.rice.iit.edu/scripts/4565_lab3.php
- Add CSS styling so the page and table look well styled. You may use bootstrap or any other CSS framework. If you use bootstrap or another framework, please load it via CDN so you don't have to include it in the submission.
- All JavaScript in main.js and your app should be React only code. Do not use jQuery or any other direct DOM manipulation to interact with the page. Put all your React classes and code in the main.js file.
- React components need to be created with ES6 class syntax. Do not try to use the reactCreateClass method. It is no longer supported.

Steps

1. Create a project folder named "**project2_myIITuserId**" using your myIIT username.
2. Inside that folder create your main HTML file named "**app.html**". Use the included template.html as a guide. The template file also includes the bootstrap CSS framework.
3. Inside that folder create your primary JS file name "**main.js**" and link to it with a script tag at the bottom of the body section. Make sure it is text/babel type.
4. Inside the main.js script file make sure you use an immediately-invoked function (IIFE) to wrap your code so it is not in the global namespace.
5. Create the top level React component that will wrap all other components first. Call this component "**PageComponent**". Start by just outputting some static HTML to the page to make sure it works and build from there.
6. Render the PageComponent to the app.html page in the div with id equal to root using the ReactDOM.render() method. This should be the only call to ReactDOM.render() in your app.
7. Create the top level employees table component and name it "**EmployeesTable**". This component will render out the table tag and any column header row/cells and then will have to evaluate the data from the api to generate the rows. To start I would suggest having this component just output the whole table and one data row. You can just manually put dummy data in the table for now. In future step you will replace this static info with React components.
8. You will need to load the employee data from the given api url. This url returns JSON as an array of employee objects. This can be done in either the PageComponent or EmployeesTable component. In React remember you usually pass data from parent to children as properties. If you need to pass data from a child to a parent you need to define a handler function in the parent, pass that handler to the child via props, and then call that function in the child using {this.props.???}. A common pattern is to store state in a common parent component and pass it with props to children. It may be easiest to do the AJAX in the page and pass the data array to the table component as a prop.
9. You will want to review the React Lifecycle methods. We mentioned in class. These methods can be defined in your components and get called at specific times in the lifecycle of the component. The componentDidMount method is the best choice to load data since you know the component rendered on page.
10. In the componentDidMount function use an XMLHttpRequest to make an AJAX request to the api endpoint and store the results in the components state. Remember to setup your state in the constructor. You may use the new Fetch API as demonstrated in class if you would prefer instead of regular XMLHttpRequest. Fetch is only supported in the newer chrome and firefox browsers.
11. Create a React component to represent a row of data in the table. This component should be called "**EmployeesTableRow**" and it should focus on rendering a single row. This includes the <tr> tags and the <td> tags. Again, start with some static data in the JSX to make sure it all is working and then worry about getting the data. This component should have an **employee** property that you pass a single employee object to it from the parent. Inside this component the employee object would be available as

{this.props.employee} and each data component of the object something like {this.props.employee.email}.

12. In the EmployeesTable component you will want to take the json data, which is an array, and create an array of EmployeeTableRow components from it and pass each individual employee object to the row as a property. This could easily be done with the javascript array map method. You need to have an array of EmployeesTableRow components. You can then just replace the rows in the table with an evaluation of that array.
13. Now you will need to go back to the EmployeesTable component and replace any data rows with EmployeesTableRow components. So if you name the array of EmployeesTableRow "rows", then in the render return statement you can replace the table html rows with {rows} and it will automatically loop through the array and output each component.
14. Above the table make sure you include two title lines. One that has the class number you are in and project 2. The second line should have your name and email. See the included image for an example.
15. Zip and unzip your project and test it multiple times. I will not try to troubleshoot and fix code issues, path issues, case sensitivity issues. I'm going to unzip the submission and open the app.html in chorme or firefox. It will run without errors or not.

Graduate Additional Requirements

If you are involved in **any section of 565** you need to complete the additional requirements listed here.

Implement a button to refresh the employee data from the api. You will need to add a button or link to the page and add an onClick attribute to it that runs a function to handle reloading the data. If you decide to manage state and do the ajax in the PageComponent component you could create a updateData method then pass that to the button component as a prop. That passed function becomes the onClick handler. When the function runs it would run the update method in the PageComponent which calls setState(). In this example you would also be passing the data from the PageComponent component to the EmployeesTable component as a prop.

README File

Not Required for this Project.

Due Date / Late Policy

This assignment is due **Saturday December 2, 2017 11:59 PM Chicago Time**. **No Extensions**. **Assignments turned in after this time will not be graded and you will receive a 0.**

Submission Guidelines

You must upload your submission, to the blackboard assignment by the due date. The submission must be in the following format and structure. If you do not submit your assignment exactly as specified, you will receive an immediate 5% deduction.

Submission Format Specification:

Zip your project folder and submit it to the blackboard assignment.