CMPS 356 – Fall 2022
Web Applications Design and Development

Lab 02
React Basics

Objective

1. Practice using basic features of React to build a web application
2. Use JSX to define user interface elements
3. Define and render components
4. Set and update component state and handle user-interface events
5. Monitor state changes and apply side effects using hooks

Prerequisites

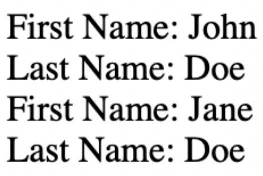
1. React app creation utility: `https://create-react-app.dev`
2. React tutorials: `https://reactjs.org/tutorial`
3. React Developer Tools browser extension: `https://blog.openreplay.com/an-introduction-to-react-dev-tools`
4. Visual Studio Code: `https://code.visualstudio.com`

1. Hello, React!

1. Under your private lab repository, create a directory for this lab and open it in Visual Studio Code.
2. Create a directory `01-hello-react` under the lab directory and use it for this exercise.
3. Create an `index.html` file and, inside it, a `Hello` component that returns "`Hello, React!`"
4. Render the `Hello` component inside the `root` division. Your source code will be similar to the following:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Hello, React!</title>
    <script crossorigin
      src="https://unpkg.com/react/umd/react.development.js"></script>
    <script crossorigin
      src="https://unpkg.com/react-dom/umd/react-dom.development.js"></script>
    <script crossorigin
      src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
    <script type="text/babel">
      const Hello = () => <h1>Hello, React!</h1>;
      const root = ReactDOM.createRoot(document.querySelector('#root'));
      root.render(<Hello />);
    </script>
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>
```

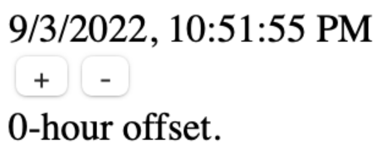
5. Extend the `Hello` component to accept a name parameter.
6. Redo the above example by creating a new React app using the `create-react-app` utility.
7. Explore the files and directories that are generated. What is part of this toolkit?
8. Create a `Hello` component that returns “`Hello, React!`”
9. Display the `Hello` component in `index.js`.
10. Experiment with rendering a couple of elements using JSX, e.g., an unordered list of a few elements.
11. Create a `User` component with first name and last name properties.
12. Render two `User` instances.



```
First Name: John
Last Name: Doe
First Name: Jane
Last Name: Doe
```

2. A Clock with a Difference

1. Create a directory `02-clock` under the lab directory and use it for this exercise.
2. Create a React application using the `create-react-app` utility.
3. Design a `Clock` component that displays the current time. Use a state variable to keep track of the time.
4. Update your component to keep the rendered time in sync with the current time
5. Augment your component with an hour offset property and an associated state variable. This difference is used to shift the time by that number of hours, similarly to a time-zone difference.
6. Add two buttons to increment and decrement, respectively, the hour offset by ½-hour steps.
7. Log a message to the console with the current value every time the date or hour offset are updated.



```
9/3/2022, 10:51:55 PM
0-hour offset.
```

3. Country Facts Application

1. Explore and run the Country Facts application provided under `base/country-facts` that was developed using plain JavaScript.
2. Recreate the application using React and use as many components as needed. Consult the Thinking in React tutorial: https://beta.reactjs.org/learn/thinking-in-react.

3. Populate the list of regions when the application is first loaded.
4. Fetch the list of countries whenever the region selection is updated.
5. Fetch and display the table of facts whenever the country is updated and update it whenever the country selection changes.