Full Stack IV – Lab 2

- Components & JSX
- Intro: Testing with Jest

Developer Note:

- Work can be done in the same react-app. Remember to not include node_modules in the GitHub submission.

References:

https://facebook.github.io/create-react-app/

https://jestjs.io/

https://airbnb.io/enzyme/

Exercise 1 – Hello JSX

1. JSX also allows you to easily include variables into your HTML, for example, let's assume that you have the following variable available:

```
let age = 73;
```

If you want to include the value of that variable into your HTML code dynamically, you can do it like this:

```
let output = <span> Sly Stallone is { age } years old </span>
```

Then, we can render the everything in the website content using ReactDOM.render like this:

```
ReactDOM.render(output, document.querySelector('#myDiv'));
```

The resulting website HTML document will look like this:

```
<div id="myDiv">
  <span>Sly Stallone is 73 years old</span>
</div>
```

Run **create-react-app** to create a new application. Given the following code file, create a file named **exercise.html** in the **public** folder of your react application.

https://drive.google.com/file/d/1KcGHrOysQYojxaF-a5IWXwCGEPB78QsN/view?usp=sharing

```
var ipsumText = 'Lorem ipsum dolor sit amet, consectetur adipiscing elit. ';

ReactDOM.render(
  <div>
    // Your work here
  </div>,
  document.getElementById('impl')
);
```

Try to match the markup of the box contents. I recommend referring to the HTML tab and/or inspecting the DOM.

Exercise 2 – Functional and Class Components

1. In the components folder create two new files **student.js** and **college.js**

2. In the **student.js** file create a functional component with the following code.

```
import React from "react";

const Student = () => {
  return <p> Student name with student number </p>;
};

export default Student;
```

3. In the **college.js** file use the installed React Snippet extension command **imr + tab** followed by **cc + tab**. The following code snippet will be generated.

```
import React from 'react';

class  extends Component {
    state = {  }
    render() {
        return (  );
    }
}

export default ;
```

4. Modify the component class file such that the code is as follows:

```
import React from "react";

class College extends React.Component {
  render() {
    return <p> College name with address </p>;
  }
}

export default College;
```

5. In the **App.js** file, import both the **Student** and **College** class components. Then remove the **Greeter** component in the **App render** and add the new components.

```
class App extends Component {
  render() {
    return
      (
        <Student />
        <Student />
        <Student />
        <College />
      )
  }
}
```

6. The following compile error will occur. This will occur because React render must have only one root element. Write the Component JSX in a parent element.

```
<div>
  <Student />
  <Student />
  <Student />
  <College />
</div>
```

Failed to compile

```
./src/App.js
  Line 14:  Parsing error: Adjacent JSX elements must be wrapped in an enclosing tag. Did you want a JSX fragment <>...</>?

  12 |       (
  13 |         <Student />
> 14 |         <Student />
     |         ^
  15 |         <Student />
  16 |         <Course />
  17 |       )
```

6. The output in the browser will be as follows:

Student name with student number

Student name with student number

Student name with student number

College name with address

7. **Challenge:**

- Update the render method to use React.Fragment (or the short form for React.Fragment) as the parent element.

Exercise 3 – Props & Nested Components

1. In **App.js**, modify the JSX, so we can pass values to the components via **props**. (React properties)

```
class App extends Component {
  render() {
    return (
      <div>
        <Student name="Rick Rude" number="11111" />
        <Student name="Shawn Michaels" number="22222" />
        <Student name="Bret Hart" number="33333" />
        <College name="George Brown" location="Casa Loma" />
      </div>
    );
  }
}
```

2. In **student.js** functional component add props as the parameter and use it to pass data to the function.

```
const Student = props => {
  return (
    <p>
      Student <b>{props.name}</b> with student number <b>{props.number}</b>
    </p>
  );
};
```

3. In **college.js** class component add props using the this keyword. The props are automatically passed to the class..

```
class College extends React.Component {
  render() {
    return (
      <p>
        College <b>{this.props.name}</b> with address<b>{this.props.location}</b>
      </p>
    );
  }
}
```

4. Add a new functional component name **courses.js**. Use the react snippet command **imr+tab** to import React and the **sfc+tab** for the functional component snippet. This component will have

props as a parameter to display the course number ie. 1, 2, 3 etc in a paragraph tag.

5. Import and nest the **Courses** component in the **Student** component and set the course number will be set to **props.enrolled** in JSX.

6. In the **App.js** component add another **props** property enrolled to the **Student** component. This value will be passed down to the nested **Course** components.

```
class App extends Component {
  render() {
    return (
      <div>
        <Student name="Rick Rude" number="11111" enrolled="2"/>
        <Student name="Shawn Michaels" number="22222" enrolled="1"/>
        <Student name="Bret Hart" number="33333" enrolled="3"/>
        <College name="George Brown" location="Casa Loma" />
      </div>
    );
  }
}
```

The browser will display the following result:

Student **Rick Rude** with student number **11111**

Student is enrolled in **Course 2**

Student **Shawn Michaels** with student number **22222**

Student is enrolled in **Course 1**

Student **Bret Hart** with student number **33333**

Student is enrolled in **Course 3**

College **George Brown** with address **Casa Loma**

7. ## Challenge:

   - Modify **Student** component, so that it will take the **enrolled props value** and do a for loop that will output the **Course** control. The browser should display the following:

Student **Rick Rude** with student number **11111**

Student is enrolled in

Course 0

Course 1

Student **Shawn Michaels** with student number **22222**

Student is enrolled in

Course 0

Student **Bret Hart** with student number **33333**

Student is enrolled in

Course 0

Course 1

Course 2

College **George Brown** with address **Casa Loma**

Homework – Intro to Testing with Jest

1. Use npm to install Jest testing framework.

```
npm install --save-dev jest
```

2. Update the package.json script section and add the following for jest

```
"scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "jest",
  "eject": "react-scripts eject"
},
```

3. Delete **the App.test.js** file, since we won't be needing it for this exercise.

4. Create a simple **hello.test.js** file for testing. We will be using Jest's matchers to test values. https://jestjs.io/docs/en/using-matchers

```
describe("First passing testing with Jest", () => {
  test("adds 1 + 2 to equal 3", () => {
    expect(1 + 2).toBe(3);
  });
});

describe("First failing testing with Jest", () => {
  test("adds 1 + 1 to not equal 3", () => {
    expect(1 + 1).toBe(3);
  });
});
```

5. Run the **npm test** on the command line will return the following test results.

```
npm test
```

```
Test Suites: 1 failed, 1 total
Tests:       1 failed, 1 passed, 2 total
Snapshots:   0 total
Time:        3.493s
Ran all test suites.
```

Challenge: Install React Testing Library & Examples
https://testing-library.com/docs/react-testing-library/intro

```
npm install --save-dev @testing-library/react
```

https://react-testing-examples.com/