Week 5 Day 1

React Intro

React: Single Page Applications

REVATURE

- Webapps that render on a single page
- Loads all HTML/CSS/JS at once
- Advantages
  - Fast/Responsive
  - Caching
  - Pleasant UX
- Disadvantages
  - Lacks in SEO
  - Less secure vs cross-site scripting
  - Longer initial load

React: Overview

REVATURE

- Lightweight JS Library
  - Introduced in 2011 by Facebook
- Library:
  - Less bulk than a framework
  - Use other modules to expand
  - Developer chooses what they want

React: Disadvantages

REVATURE

- Uses JSX/TSX over HTML templates
  - Higher JS Skill
  - Slightly different syntax and conventions
- Does not follow a MVC design pattern
- Frequent updates
- Too flexible at times
  - So many decisions to make

React: Generating Projects

REVATURE

- Generate projects with the create-react-app CLI
  - Install the CLI with `npm i –g create-create-app`
  - Use the CLI with `npx create-react-app name-of-app`
- By default, React generates the projects using Javascript, you must declare the Typescript template
  - `npx create-reacte-app name-of-app --template typescript`
  - Generating it with the template will include the special React typescript types

REVATURE

React: Generate First Project

React: Component Architecture

REVATURE

- React uses components to make up the application
  - The application will be broken up into multiple components communicate
- Single Responsibility Principle
  - A single component should have responsibility of a single functionality
  - Increases reusability and reliability

React: Components

REVATURE

- Class Based

```
export class Navbar extends React.Component {

    render(){
        return(
            <div className="navbar">
                <div className="nav-item">
                    <img className="nav-logo" src={logo} />
                </div>
                <div className="nav-item">
                    <p className="nav-link">Course Link</p>
                </div>
                <div className="nav-item">
                    <p className="nav-link">Student/Teacher Login</p>
                </div>
            </div>
        )
    }

}
```

- Function Based

```
export const Courses:React.FC = () => {

    const [courses, setCourses] = useState<Course[]>([]);

    useEffect(() => {
        setCourses([
            {
                courseId: 1,
                subject: "Reading",
                courseNumber: 100,
                courseName: "Intro to Reading",
                teacher: "NA"
            }
        ]);
        console.log(courses);
    }, [courses.length]);

    const addCourse = (course:Course) => {
        setCourses([...courses, course]);
    }

    return (
        <div className="courses">
            <NewCourse newCourse={addCourse} currentId={courses.length}></NewCourse>
        </div>
    )
}
```

React: JSX

REVATURE

- Our component templates are written in JSX/TSX
  - Javascript XML/Typescript XML
  - Component files are marked with .jsx/.tsx extensions
- React uses Babel to compile JSX into ES6
- JSX is case sensitive
- Always wrap JSX in a single grouping tag

React: Virtual DOM

REVATURE

- React uses a virtual DOM to map JSX to HTML on the page
- The virtual DOM is also more performant
  - Only modify the actual DOM when necessary
- When rendering, React injects all our components into a root div on an HTML page
  - That single HTML page will never reload

React: Class Components

REVATURE

```
export class GenericComponent extends React.Component<props:Prop,state:State> {

    state:State = {
        //some default state
    }

    //constructor for passing props, initializing state
    constructor(props:Prop){
        super(props);

        state = {
            //some initialized state
        }
    }

    render(
        return(
            <div>
                The content of the component
            </div>
        )
    )

}
```

React: Styling Components

REVATURE

- Components are styled in the same way as HTML
- Create CSS sheets, import them to the components you want to style

```
import React from 'react';
import logo from "../../img/Revature.png"
import './Navbar.css';
```

REVATURE

Create Class Based Components

React: Component Lifecycle

REVATURE

- Class based components have built in methods for different points in the component's life
  - Constructor used for initializing data
  - componentDidMount for fetching data on page load
  - componentWillUnmount for clean up on destroy
  - More in the React documentation
- React hooks (to be discussed) brought these method functionalities to function based

React: Component State

REVATURE

- State is component specific data
  - Should be completely encapsulated in the component
- State should not be modified directly
- To store data in a class-based component, create a field called `state`
- To modify the state use Reacts prebuilt `setState()` method
- React allows you to display the value of state on the page using {value}

React: Data Sharing

REVATURE

- State should not be accessible to other components unless explicitly shared through props
- Nested components can pass data from top down
- The data passed from parent to child are called `props`
- React has one-way data flow, cannot pass data from child to parent

REVATURE

Using State and Props