COMP 9783 Front-end Development

Lab-3-2 - React JSX Lab. (1% of the course mark)

Name:

Student Number:

The React JSX Lab is designed to provide hands-on experience with JSX, a syntax extension for JavaScript commonly used with React to describe the UI structure. JSX allows developers to write HTML-like code within JavaScript, making it easier to visualize the component structure and behavior. This lab will guide participants through the core concepts of JSX, including its syntax, embedding expressions, and using it to build interactive UI components.

Lab objectives:

1. Learn the basics of JSX syntax and how it integrates with JavaScript.
2. Explore the differences between JSX and traditional HTML.
3. Understand the rules and best practices for writing JSX.

Create a react app:

1. On **VSCode**, open the **terminal** and type **npm create vite@latest react-jsx-app --** **--template react** and **press enter**. This should create a basic react app.
2. **Build** and **start the app** by typing:
    a. **cd react-jsx-app**
    b. npm install
    c. npm run dev
3. On the **src folder**, **delete** the following files: **App.css** and **index.css**.
4. On src/**main.jsx** delete this: **import './index.css'** and **save the changes**.
5. **Modify ./public/index.html** and **update** the **title** html code and **save the changes**:

```
<title>react-jsx-app</title>
```

6. Inside the **src folder create** a **folder** named: **css** and **copy** the **3 css files (normalize.css, sakura.css and styles.css)** from the extracted lab.
7. Overwrite **App.jsx** with the following code and **save the changes**:

```
import "./css/normalize.css";
import "./css/sakura.css";
import "./css/styles.css";

function App() {
  return <h1>IT WORKS...</h1>;
}

export default App;
```

8. Open your browser and navigate to: http://localhost:5173/ and this should display **IT WORKS…** text. Make sure that your app is working before proceeding to the next step. Leave the app running.

9. **Try out** different **JSX scenarios** below, by **overwriting the body** of the **App() function**, use the **sample code as a guide** and create your **own code for each scenario**, also **capture a screenshot** of **each of the code you created**:

JSX Scenario	Sample code
Single line HTML	return Hello World ;
Single line HTML as a const variable	const htmlElement = Hello World ; return htmlElement;

Using expressions	return 1 + 1 = {1 + 1} ;
Multi line HTML, place inside parenthesis	return (JS Frontend Frameworks React Angular);
One top level HTML element	return (JS Frontend Frameworks JS Frontend Frameworks JS Frontend Frameworks);
One top level using fragment	return (<> JS Frontend Frameworks JS Frontend Frameworks JS Frontend Frameworks </>);
All Elements must be closed	return (<> Username: <input type="text" placeholder="Enter your username" /> </>);
camelCase attributes className	return <h1 className="redFont">Hello World</h1>;
if statement usage	const dayString = ["Monday", "Tuesday", "Wednesday",

	"Thursday",
	"Friday",
	"Saturday",
	"Sunday",
	];
	const currentDay = new Date().getDay();
	let weekEndOrDay = "Weekend";
	if (currentDay <= 5) {
	weekEndOrDay = "Weekday";
	}
	return (
	
	{dayString[currentDay - 1]} is a {weekEndOrDay}
	
	);

How to use ternary	const dayString = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday",]; const currentDay = new Date().getDay(); return ({dayString[currentDay - 1]} is a {currentDay <= 5 ? "Weekday" : "Weekend"});

Submission:

1. Use the html template: **index.html** and **write HTML codes for each screenshot**:
    a. Write a title and short description.
    b. Display the screenshot.

Note: Feel free to use any component from a CSS framework of your choice.

2. Create a new folder named **html** and copy all the **HTML**, **CSS** and **PNG** files used in the previous step.
3. Create a **zip file** of the **html folder**.
4. Submit the **zip file** to **GBC - D2L**.