COMP 9783 Front-end Development

Lab-3-5 - React Events Lab. (1% of the course mark)

Name:

Student Number:

The React Events Lab is designed to provide a comprehensive, hands-on experience with handling events in React applications. Events are fundamental to creating interactive user interfaces, and this lab will cover the basics of event handling for ensuring responsive and efficient event-driven applications. Participants will learn how to handle various types of events such as the use of synthetic events.

Lab objectives:

- 1. Overview of event handling in React.
- 2. Demonstrate how to handle click and keyboard events.
- 3. Using event objects to access event properties.

Create a react app:

- 1. On VSCode, open the terminal and type npm create vite@latest react-events-app --
 - **--template react** and **press enter**. This should create a basic react app.
- 2. **Build** and **start the app** by typing:
 - a. cd react-events-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-events-app</title>
```

- 6. This lab will use **bulma** as the **css frontend**. To install **bulma** type: **npm install bulma** and **press enter**.
- 7. Overwrite **App.jsx** with the following code and **save the changes**:

```
import "bulma/css/bulma.min.css";
import Content from "./components/Content";

function App() {
  return <Content></Content>;
}

export default App;
```

- 8. Inside the **src** folder create a folder named: **components**.
- 9. Inside the **components folder**, create a file named: **ChangeEvent.jsx** and add the following code below:

```
<label class="label">GBC Campus</label>
        <div class="select">
          <select id="dropDownList" onChange={changeHandler}>
            <option>Casaloma</option>
            <option>Downtown</option>
            <option>Waterfront</option>
         </select>
        </div>
      </div>
      <div class="field">
        <label class="label">Status</label>
        <div class="control">
         <label class="radio">
            <input</pre>
              id="fullTimeStatus"
             onChange={changeHandler}
             type="radio"
             name="status"
             Fulltime
         </label>
           
         <label class="radio">
            <input</pre>
             id="partTimeStatus"
             onChange={changeHandler}
              type="radio"
             name="status"
             Parttime
         </label>
       </div>
      </div>
 );
function changeHandler(event) {
  alert(`${event.target.id}: ${event.target.value}`);
```

```
}
export default ChangeEvent;
```

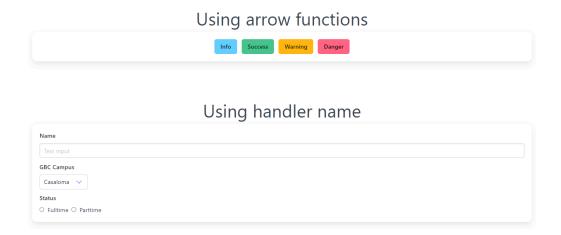
10. Inside the **components folder**, create a file named: **ClickEvent.jsx** and add the following code below:

```
function ClickEvent() {
  return (
    <div class="is-flex is-justify-content-center buttons">
      <button onClick={(event) => clickHandler(event)} class="button is-info">
        Info
      </button>
      <button</pre>
        onClick={(event) => clickHandler(event)}
        class="button is-success"
        Success
      </button>
      <button
        onClick={(event) => clickHandler(event)}
        class="button is-warning"
        Warning
      </button>
      <button onClick={(event) => clickHandler(event)} class="button is-danger">
        Danger
      </button>
    </div>
  );
function clickHandler(event) {
  alert(`Button: ${event.target.innerText} was clicked!`);
export default ClickEvent;
```

11. Inside the **components folder**, create a file named: **Content.jsx** and add the following code below:

```
import ClickEvent from "./ClickEvent";
import ChangeEvent from "./ChangeEvent";
function Content() {
 return (
     <section className="section">
       <div className="container">
         <h1 className="is-size-1 has-text-centered">Using arrow functions</h1>
         <div className="box">
            <ClickEvent></ClickEvent>
         </div>
       </div>
     </section>
     <section className="section">
       <div className="container">
         <h1 className="is-size-1 has-text-centered">Using handler name</h1>
         <div className="box">
            <ChangeEvent></ChangeEvent>
         </div>
       </div>
     </section>
 );
export default Content;
```

12. **Open** your **browser** and **navigate** to: http://localhost:5173/ this should display a web app (**see below**). Make sure that your app is working before proceeding to the next step.



- 13. Interact with the ui elements of the web app by clicking the buttons, radio buttons, drop down lists and entering text. The handler should display an alert box when an event is triggered.
- 14. Update all the instances of the onChange handler of ChangeEvent.jsx and save the changes.

From:

```
onChange={changeHandler}
```

To:

```
onChange={(event) => changeHandler(event)}
```

15. Update all the instances of the onClick handler of ClickEvent.jsx and save the changes.

From:

```
onClick={(event) => clickHandler(event)}
```

To:

```
onClick={clickHandler}
```

16. Navigate to http://localhost:5173/ The web app should still work since we just interchanged the 2 different ways of handling events.

Submission:

- 1. Create a new folder named react and copy ChangeEvent.jsx and ClickEvent.jsx.
- 2. Create a zip file of the react folder.
- 3. Submit the **zip file** to **GBC D2L**.