React

DAY 2

Default Event

- `<a href="www.kakao.com">`
  - 클릭하면 해당 주소로 이동하는 event
- `<input type="submit">`
  - 입력하고 누르게되면 설정한 특정 URL로 request를 보내는 event

> **이처럼 위와 같은 method가 가지는 기본적인 event 속성이 있다.**
> **Default Event는 이런 기본적인 event를 막아주는것이다.**

```
handleSubmit = (e) => {
  // 현재 submit event가 발생해서 그 것을 처리하고 있다.
  // submit 이라는 default event가 진행되고 있기 때문에 이 처리를 하고 싶지 않다.
  // 그럼 event 객체를 처리하면 된다. -> 그럼 이제 입력을 누르더라도 다시 request하지 않는다.
  e.preventDefault();
};
```

이제는 Class Component를 Functional Component로 바꿔보자!!

```
import React from 'react'

  export default function App_back() {
    return (
      <div>App_back</div>
    )
  }
```

함수형 컴포넌트는 `render( )` method가 없다. 함수의 return값이 JSX이기 때문이다.

재사용성과 유지보수를 위해서 Component를 분리해보자!!

App.js

```
import React, { useState } from "react";
import "./App.css";
import Form from "./components/Form";
import Lists from "./components/Lists";

// render() method가 존재하지 않는다.
// 우리 함수의 return 값이 JSX이다.
export default function App() {
  const [todoData, setTodoData] = useState([
    {
      id: "1",
      title: "Training",
      completed: false,
      className: "font-mono",
    },
  ]);
  const [value, setValue] = useState("");

  return (
```

```
    <div className="flex items-center justify-center w-screen h-screen bg-blue-100 font-mono ">
      <div className="w-full p-6 m-4 bg-white rounded shadow md:w-3/4 md:max-w-lg lg:w-3/4 lg: max-w-lg font-mono ">
        <div className="flex justify-between mb-3 font-mono">
          <h2>What is your main focus for today?</h2>
        </div>

        <Lists todoData={todoData} setTodoData={setTodoData} />

        <Form
          todoData={todoData}
          setTodoData={setTodoData}
          value={value}
          setValue={setValue}
        />
      </div>
    </div>
  );
}
```

List.js

```
import React from "react";
import { DragDropContext, Draggable, Droppable } from "react-beautiful-dnd";

export default function Lists({ todoData, setTodoData }) {
  const deleteClick = (id) => {
    // 해당 ID에 대한 목록을 지워야한다.
    // DATA를 지워야한다. => 열심히 하면 배열 처리 가능하다.
    // 변경된 데이터를 지우고 화면을 다시 그려야한다.
    const newTodoData = todoData.filter((data) => data.id !== id);
    // 변경된 Array를 가지고 어떻게 화면에 출력할까??
    // React State
    // React에서 데이터가 변할 때 화면을 다시 rendering하기 위해서 사용한다.
    // State는 component안에서 관리된다!
    // 일반적으로 생성자 안에서 정의된다!
    // 이름이 state라는 property
    setTodoData(newTodoData);
  };

  const handleCompleteChange = (id) => {
    // id에 대한 todoData의 Completed값을 변경시켜야한다.
    let newTodoData = todoData.map((data) => {
      if (data.id === id) {
        data.completed = !data.completed;
      }
      return data;
    });
    setTodoData(newTodoData);
  };
  const handleDrop = (e) => {
    // e: event 객체, event에 대한 세부정보를 가지고 있다.
    // e.source : drag한 객체, e.destination : drop한 객체
    if (!e.destination) return;

    const newTodoData = todoData;

    // drag되는것을 삭제시키는 Code
    const [reorder] = newTodoData.splice(e.source.index, 1);

    // drop되는 위치에 삽입시키는 Code
    newTodoData.splice(e.destination.index, 0, reorder);

    setTodoData(newTodoData);
  };

  return (
    <div>
      <DragDropContext onDragEnd={handleDrop}>
        <Droppable droppableId="to-do">
          {(provided) => (
            <div {...provided.droppableProps} ref={provided.innerRef}>
              {todoData.map((data, index) => (
                <Draggable
                  key={data.id}
                  draggableId={data.id.toString()}
                  index={index}
                >
```

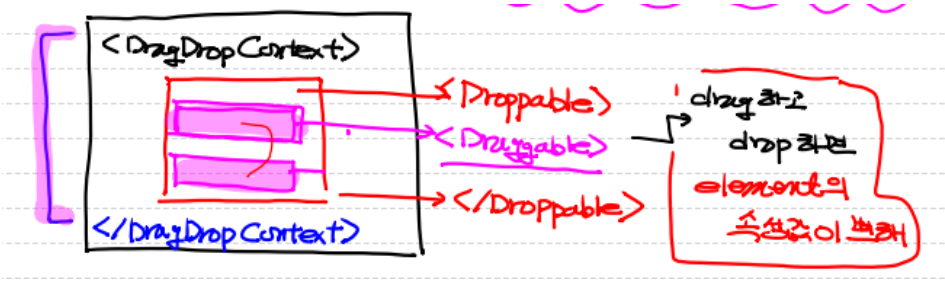
```
                  {(provided, snapshot) => (
                    <div
                      key={data.id}
                      {...provided.draggableProps}
                      ref={provided.innerRef}
                      {...provided.dragHandleProps}
                    >
                      <div
                        className={` ${
                          snapshot.isDragging ? "bg-violet-200" : "bg-pink-100"
                        } flex items-center justify-between w-full px-4 py-1 my-2 text-gray-600  border rounded font-mono`}
                      >
                        <div className="items-center">
                          <input
                            type="checkbox"
                            defaultChecked={false}
                            onChange={() => handleCompleteChange(data.id)}
                          />{" "}
                          <span
                            className={
                              data.completed ? "line-through" : undefined
                            }
                          >
                            {data.title}
                          </span>
                        </div>
                        <div className="items-center font-mono ">
                          <button onClick={() => deleteClick(data.id)}>
                            delete
                          </button>
                        </div>
                      </div>
                    </div>
                  )}
                </Draggable>
              ))}
              {provided.placeholder}
            </div>
          )}
        </Droppable>
      </DragDropContext>
    </div>
  );
}
```

Form.js

```
import React from "react";

export default function Form({ todoData, setTodoData, value, setValue }) {
  const handleChange = (e) => {
    setValue(e.target.value);
  };
  const handleSubmit = (e) => {
    // Submit event 발생해서 처리하고 있다.
    // 이런 default event처리를 안할래요!
    e.preventDefault();

    let newTodo = {
      id: Date.now(), // Unique한 값을 표현하기 위해서 현재 시간을 찍는다.
      title: value,
      completed: false,
    };

    setTodoData([...todoData, newTodo]);
    setValue("");
  };
  return (
    <form onSubmit={handleSubmit} className="flex pt-2 ">
      <input
        type="text"
        name="todoItem"
        className=" w-full px-3 py-2 mr-4 text-gray-500 border rounded shadow font-sans  "
        placeholder="Please enter a new to do"
        value={value}
        onChange={handleChange}
      />
```

```
      <input
        type="submit"
        className="p-2 text-blue-400 border-blue-400 rounded hover:text-white hover:bg-blue-200 font-sans"
        value="Input"
      />
    </form>
  );
}
```

Drag & Drop




```
return (
    <div>
      <DragDropContext onDragEnd={handleDrop}>
        <Droppable droppableId="to-do">
          {(provided) => (
            <div {...provided.droppableProps} ref={provided.innerRef}>
              {todoData.map((data, index) => (
                <Draggable
                  key={data.id}
                  draggableId={data.id.toString()}
                  index={index}
                >
                  {(provided, snapshot) => (
                    <div
                      key={data.id}
                      {...provided.draggableProps}
                      ref={provided.innerRef}
                      {...provided.dragHandleProps}
                    >
                      <div
                        className={` ${
                          snapshot.isDragging ? "bg-violet-200" : "bg-pink-100"
                        } flex items-center justify-between w-full px-4 py-1 my-2 text-gray-600  border rounded font-mono`}
                      >
                        <div className="items-center">
                          <input
                            type="checkbox"
                            defaultChecked={false}
                            onChange={() => handleCompleteChange(data.id)}
                          />{" "}
                          <span
                            className={
                              data.completed ? "line-through" : undefined
                            }
                          >
                            {data.title}
                          </span>
                        </div>
                        <div className="items-center font-mono ">
                          <button onClick={() => deleteClick(data.id)}>
                            delete
                          </button>
                        </div>
                      </div>
                    </div>
                  )}
```

```
              </Draggable>
            ))}
            {provided.placeholder}
          </div>
        )}
      </Droppable>
    </DragDropContext>
  </div>
);
```

코드를 보면서 복습해보자,,,