07-hook-event

#01. 프로젝트 생성 및 초기화

1) 프로젝트 생성

> 프로젝트 이름은 영어 소문자만 사용 가능함.

```
yarn create react-app 07-hook-event
```

2) 프로젝트 생성 후 기초작업

프로젝트 초기화

대상	작업 내용
src폴더	App.css, App.test.js, index.css, logo.svg, setupTests.js, reportWebVitals.js 삭제
App.js	App.css와 logo.svg에 대한 참조(import) 구문 제거
index.js	index.css와 reportWebVitals.js에 대한 참조(import) 구문 제거 맨 마지막 행에 있는 `reportWebVitals()` 부분 삭제

3) 추가 패키지 설치

패키지 이름	설명
react-router-dom	SPA앱을 만들 때 사용. URL에 따라 실행할 Javascript를 분기한다.
react-helmet-async	`<head>`태그 내의 SEO관련 태그를 설정한다.
sass	sass와 scss 컴파일 기능 제공
styled-components	styled component 지원
styled-components-breakpoints	styled component에서 media query를 쉽게 사용할 수 있게 한다.
dayjs	날짜 처리 기능 제공 (리액트와 직접적인 연관은 없음)

```
yarn add react-router-dom react-helmet-async sass styled-components
styled-components-breakpoints dayjs
```

4) Router 적용

index.js

import 구문 추가

```
import { BrowserRouter } from 'react-router-dom';
```

index.js 파일에서 `<App />`을 `<BrowserRouter><App /></BrowserRouter>`로 변경

App.js

```
import { NavLink, Routes, Route } from "react-router-dom";
```

혹은

```
import { Link, Routes, Route } from "react-router-dom";
```

5) 프로젝트 실행

프로젝트를 VSCode로 열고, `Ctrl` + `~`를 눌러 터미널 실행

```
yarn start
```

#02. 리액트 이벤트 시스템

1) 용어 정리

a) 이벤트

프로그램이 겪는 일련의 사건.

사용자가 프로그램에 대해 행하는 어떠한 행위를 포함한다.

b) 이벤트 리스너

이벤트가 발생할 때를 대기하고 있는 구현체.

웹에서는 HTML 태그에서 명시하는 이벤트 속성이 이에 해당한다.

```
<a onclick="...">
```

c) 이벤트 핸들러

이벤트가 발생한 순간 그에 반응하도록 구현된 코드나 함수 (또는 클래스)

3) 리액트에서 이벤트 구현시 주의점

1. 이벤트 리스너의 이름은 HTML속성이 아닌 JSX에 의한 자바스크립트 프로퍼티이므로 카멜 표기법으로 작성.
    - onclick (X)
    - onClick (O)
2. 이벤트 리스너에 전달할 이벤트 핸들러는 코드 형태가 아니라 반드시 함수 형태로 전달해야 한다.
3. DOM 요소(=HTML 태그)에만 이벤트 리스너가 존재한다.
    - 직접 구현한 컴포넌트에 대해서는 설정 불가

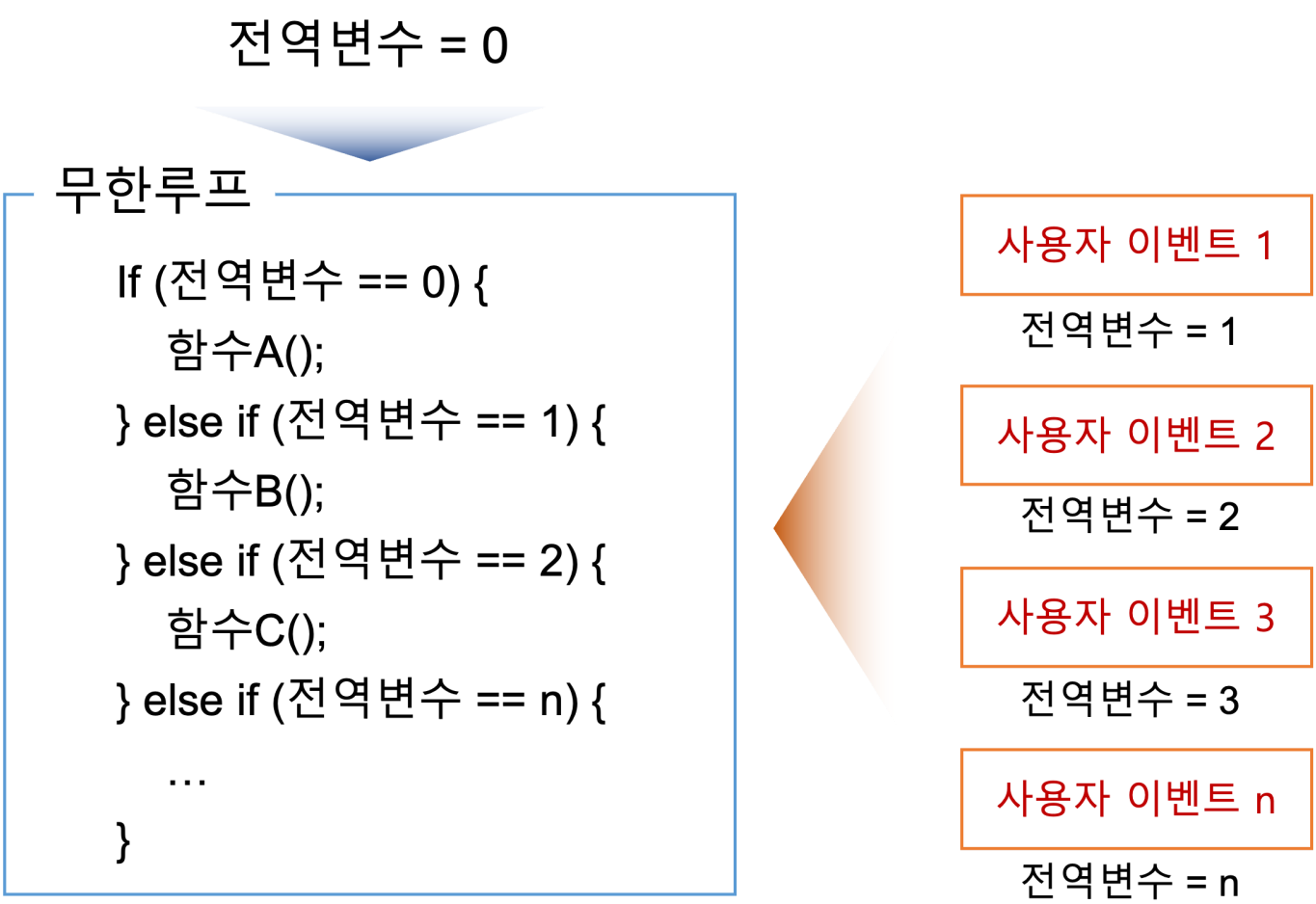
#03. Hooks

함수형 컴포넌트에서 상태값(state)를 관리하기 위한 기능으로 클래스형 컴포넌트의 LifeCycle에 대응된다.

> React v16.8부터 새로 추가되었음.

쉽게 이야기 하면 아래의 항목들은 특정 상황에서 자동으로 호출되는 함수들을 의미함.

1) 상태변수의 이해



2) 기본 Hook 함수들

a) useState

`useState()` 함수를 import하고 사용하는 경우.

```
import React, {useState} from 'react';
...
const [상태변수, 변수에대한setter함수] = useState(초기값);
```

`useState()` 함수를 import하지 않고 직접 사용하는 경우.

```
const [상태변수, 변수에대한setter함수] = React.useState(초기값);
```

- 가장 기본적인 Hook 함수
- 함수형 컴포넌트에서 state값을 생성한다.
- 하나의 useState 함수는 하나의 상태 값만 관리할 수 있다.
- 컴포넌트에서 관리해야 할 상태가 여러 개라면 useState를 여러번 사용하면 된다.

b) useEffect

useEffect는 기본적으로 렌더링 직후마다 실행되며,
두 번째 파라미터 배열에 무엇을 넣는지에 따라 실행되는 조건이 달라진다.

> 클래스형 컴포넌트의 **componentDidMount**와 **componentDidUpdate**를 합친 형태

렌더링 될 때마다 실행되는 함수 정의

최초 등장하거나 state값이 변경될 때 모두 실행된다.

```
useEffect(() => {
    ... 처리할 코드 ...
});
```

업데이트시에는 생략되는 함수 정의

컴포넌트가 마운트될 때 최초 1회만 실행된다. (state값이 변경될 때는 실행되지 않는다.)

```
useEffect(() => {
    ... 처리할 코드 ...
}, []);
```

특정 state, props값이 변경될 때만 호출되도록 설정하기

```
useEffect(() => {
    ... 처리할 코드 ...
```

```
}, [값이름]);
```

컴포넌트가 언마운트(화면에서 사라짐) 될 때만 실행되도록 설정하기

클로저(리턴되는 함수)를 명시한다.

```
useEffect(() => {
    return function() {
        ... 처리할 코드 ...
    };
}, [state값이름]);
```

```
useEffect(() => {
    return () => {
        ... 처리할 코드 ...
    };
}, [state값이름]);
```

c) useContext

거의 사용하지 않음.

3) 특정한 경우에만 사용되는 기능들.

> 다음의 Hook는 이전 섹션에서의 기본 Hook의 변경이거나 특정한 경우에만 필요한 것입니다. 익히는 것에 너무 압박 받지는 마세요.

a) useRef

함수형 컴포넌트에서 ref를 쉽게 사용할 수 있도록 처리해 준다.

Vanilla Script에서 `document.getElementById(...)`나 `document.querySelector(...)`로 DOM 객체를 취득하는 과정을 React 스타일로 표현한 것으로 이해할 수 있다.

b) useReducer

useState 보다 더 다양한 컴포넌트 상황에 따라 다양한 상태를 다른 값으로 업데이트 하고자 하는 경우 사용.

useState의 대체 함수로 이해할 수 있다.

state값이 다수의 하위값을 포함하거나 이를 활용하는 복잡한 로직을 만드는 경우에 useState보다 useReducer를 선호한다.

c) useMemo

함수형 컴포넌트 내에서의 연산 최적화.

숫자, 문자열, 객체처럼 일반 값을 재사용하고자 할 경우 사용한다.

> memoized 된 값을 반환한다. : 컴퓨터 프로그램이 동일한 계산을 반복해야 할 때, 이전에 계산한 값을 메모리에 저장함으로써 동일한 계산의 반복 수행을 제거하여 프로그램 실행 속도를 빠르게 하는 기술

d) useCallback

렌더링 성능 최적화에 사용됨.

이벤트 핸들러 함수를 필요한 경우에만 생성할 수 있다.

> memoized 된 콜백을 반환한다.

4) Hook 사용시 주의사항

1. 반복문, 조건문, 중첩된 함수 내에서 Hook을 실행할 수 없다.
2. React Component 내에서만 호출할 수 있다.