6. State와 Lifecycle

State(매우 중요) = 화면이 바뀌어야 할 때 state를 변경
Lifecycle(클래스 컴포넌트와 밀접, 개념만 슬쩍~)

6.1 State란?

- React 컴포넌트의 상태
  → React 컴포넌트의 "변경 가능한" 데이터를 의미
- **컴포넌트에서 보여줘야 하는 내용이 사용자 인터랙션에 따라 동적으로 바뀌어야 할 때 사용**
- 컴포넌트를 개발하는 개발자가 직접 정의해서 사용
- 렌더링과 관련된 값만 state에 포함시켜야 함
  - 이유? state가 변경될 경우 컴포넌트가 재렌더링 되기 때문
- 렌더링과 관련 없는 값을 포함하면 불필요한 재렌더링 발생으로 성능이 저하될 수 있음

6.2 State의 특징

- 자바스크립트 객체 형태로 존재
- 일반적인 객체의 값을 수정하듯이 직접적으로 변경하면 안됨
  - state는 컴포넌트의 렌더링과 관련 있기 때문에 마음대로 수정하게 되면 의도한 대로 작동하지 않을 수 있음(재렌더링 안됨)
- ~~클래스 컴포넌트~~
  - 생성자에서 모든 state를 한 번에 정의
  - state를 변경하고자 할 때에는 꼭 **setState() 함수**를 사용
- 함수 컴포넌트
  - useState() Hook을 사용하여 각각의 state를 정의
  - 각 state별로 주어지는 **set함수를 사용하여** state 값을 **변경**

6.3 Lifecycle

- React 컴포넌트의 생명주기
  - 예: 컴포넌트가 생성되는 시점과 사라지는 시점 등
- 컴포넌트는 계속 존재하는 것이 아니라 데이터의 흐름에 따라 생성되고 업데이트 되다가 사라지는 과정을 겪음
- 참고: https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/
- 생명주기는 이런 개념이 있다 정도로만 이해하고 넘어가기!

6.3 Lifecycle

- 마운트(출생)
  - 컴포넌트가 생성될 때
  - constructor() 호출
  - render() 호출
  - componentDidMount() 호출

6.3 Lifecycle

- 업데이트(인생)
  - 컴포넌트가 업데이트 될 때
    = 바뀐 부분에 대한 재렌더링이 일어날 때
  - **업데이트가 일어나는 조건?**
    - **New props: 컴포넌트의 props가 변경될 때**
    - **setState() 함수 호출에 의해 state가 변경될 때**
    - forceUpdate()라는 강제 업데이트 함수가 호출될 때
  - render() 호출
  - componentDidUpdate() 호출

6.3 Lifecycle

- 언마운트(사망)
  - 컴포넌트가 사라질 때
    - (상위 컴포넌트에서) 현재 컴포넌트를 더 이상 화면에 표시하지 않게 될 때
  - 언마운트 직전에 componentWillUnmount() 호출

- ~~이 외에도 생명주기 메서드가 더 존재하지만...~~

6.4 state 사용하기(실습)

- my-app/src/chapter6 실습
- 1) state 사용하기
- 2) React Developer Tools 설치 및 사용하기
  - React를 위해 별도로 개발된 React 개발자 도구
  - 구글 검색 후 크롬 웹 스토어에서 설치
  - 크롬 개발자 도구에 Components와 Profiler 탭이 생김
  - 컴포넌트를 누르면 props와 state 확인 가능
- 3) 생명주기 메서드 사용해보기