12. State 끌어올리기

여러 개의 컴포넌트들 사이에서 state를 공유하려면..?

공통된 부모 컴포넌트로 끌어올려서 공유하는 방식

12.1 Shared State

- 종종 동일한 데이터에 대해 여러 컴포넌트에서 필요할 경우 각 컴포넌트의 state에서 데이터를 각각 보관하는 것이 아니라 가장 가까운 공통된 조상 컴포넌트의 state를 공유해서 사용하는 것이 더 효율적
- 즉, 하위 컴포넌트가 공통된 상위 컴포넌트의 state를 공유하여 사용하는 것
- 어떻게 공유..?

12.2 State를 끌어올려 공유하기(실습)

- 하위 컴포넌트의 state를 공통된 부모 컴포넌트로 끌어올려서 공유하는 방식
- 그런 다음에 하위 컴포넌트에 props로 state를 넘겨줌
- my-app/src/chapter12 실습

13. 합성 vs 상속

상속 대신 합성

13.1 합성(Composition)(예제)

- 여러 개의 컴포넌트를 합쳐서 새로운 컴포넌트를 만드는 것
- 컴포넌트들을 어떻게 조합할 것인가?
  - 1) Containment(컴포넌트에서 다른 컴포넌트를 담기)
    - 하위 컴포넌트를 포함하는 형태
    - 어떤 자식 엘리먼트가 들어올 지 미리 예상할 수 없는 경우
    - props에 기본적으로 들어있는 children 속성을 사용
  - 2) Specialization(특수화)
    - 예를 들어, WelcomeDialog(구체적)는 Dialog(범용적)의 특수한 경우이다.
    - 범용적인 개념을 구별이 되게 구체화 하는 것
    - 범용적인 컴포넌트를 만들어 놓고 이를 구체화시켜서 사용
  - 3) 두 개를 같이 사용

13.2 ~~상속(Inheritance)~~

- Facebook에서는 수천 개의 React 컴포넌트를 사용하지만, 컴포넌트를 상속 계층 구조로 작성을 권장할만한 사례를 아직 찾지 못했습니다.
- 결론은 **복잡한 컴포넌트를 쪼개 재사용 가능한 여러 개의 컴포넌트로 만들고, 만든 컴포넌트들을 조합하여 새로운 컴포넌트를 만들자!**

13.3 Card 컴포넌트 만들기(실습)

- my-app/src/chapter13 실습