Redux

💡 Redux를 사용하는 이유

리액트의 경우 대부분의 작업에서 부모 컴포넌트가 중간자 역할을 합니다. 즉, 부모 컴포넌트에서 모든 state와 공유되는 함수들을 관리하고 이를 자식에게 전달해주는 방식으로 작업하게 되는데 이는 매우 번거로운 과정입니다.

이를 해결하기 위해 많은 방법이 상태관리 라이브러리들이 탄생했고 그 중 현재 가장 많이 사용하고 있는 것은 Redux 입니다.

💡 Redux 구조

리덕스는 기본적으로 store에 여러 상태값을 저장하고, 컴포넌트에서는 store에서 상태 값을 불러와 사용합니다. 즉, 전역 상태 공간이 존재하게 하는 것이 Redux의 핵심입니다.

컴포넌트는 Store를 구독(Subscribe) 하고, 컴포넌트에서 Store의 상태에 변화를 주고 싶다면 dispatch라는 함수를 통해 액션을 Store에게 던져줍니다.

여기서 액션이란 Store의 상태에 변화를 일으킬 때, 참조할 객체입니다. 기본적으로 Action 안에는 type, payload가 있으며, type은 어떤 변화를 일으킬지, payload는 상태에 변화를 줄 값을 넣어줍니다.

예를 들어 `action: { type: 'INCREMENT', payload: 2 }` 라고 한다면 Store의 상태 값을 2만큼 올려달라는 뜻이 됩니다.

이렇게 액션을 dispatch를 통해 넘겨주면 Reducer에서 실제로 상태를 업데이트 해줍니다. 그리고 상태가 변화되면 구독하고 있던 컴포넌트에게 상태가 업데이트 되었음을 알려주는

형식입니다.

> 💡 Redux의 3가지 규칙

1. 하나의 애플리케이션 안에는 하나의 스토어가 있습니다.
2. 상태는 읽기 전용입니다.
    - React에서 모든 변수는 불변성을 지켜야 합니다. state를 업데이트 하고 싶을땐 setState를 사용하고, 배열을 업데이트 하고 싶다면 기존 배열을 복사 이후 이를 수정하는 방식으로 작업됩니다.
    - 리덕스 또한 마찬가지로 기존의 상태는 건들이지 않고 새로운 상태를 생성해 업데이트 해주는 방식으로 작업해야 합니다.
3. 변화를 일으키는 함수, 리듀서는 순수함수여야 합니다.
    - 순수함수라는 것은 즉, 파라미터로 받지 않은 변수는 사용하지 않고, 동일한 입력에 대해선 항상 동일한 출력을 내어놓아야 한다는 것입니다.