Zustand 공부

참고

https://www.heropy.dev/p/n74Tgc

기본 사용

생성

- `create` **함수로 스토어 생성**
    - `set` 과 `get` 을 매개변수로 가진다
- `create` 함수의 콜백이 반환하는 객체에서의 **속성은 상태(state)**이고, **메소드는 액션(Action)**
- `set` **함수 호출 시 콜백을 사용**하면, get 없이도 스토어 객체값을 얻을 수 있다

```
import { create } from 'zustand'
export const use이름Store = create(set => {
  return {
    상태: 초깃값,
    액션: () => {
      set(state => ({ 상태: state.상태 + 1 }))
    }
  }
})
```

사용

- 상태가 변경되면, 리렌더링이 일어난다

```
import { use이름Store } from '~/store/스토어'

export default function 컴포넌트() {
    //개별 import
  const 상태 = use이름Store(state => state.상태)
```

```
  const 액션 = use이름Store(state => state.액션)

  //필요한 것만 한 번에 import
  const {상태, 액션} = use이름Store((state) => ({
      상태 : state.상태,
      액션 : state.액션
  }));

  return (
    <>
      <h2>{상태}</h2>
      <button onClick={액션}>+</button>
    </>
  )
}
```

- 개별 상태나 액션이 아닌 **스토어 자체**를 불러올 수도 있지만, 이는 **사용하지 않는 상태가 변경되어도 컴포넌트가 다시 렌더링**되기 때문에 권장하지 않는다
  - `use이름Store` 는 전체 객체를 반환한다 !!

```
import { use이름Store } from '~/store/스토어'
export default function 컴포넌트() {
  const 스토어 = use이름Store() //권장하지 않음
  return (
    <>
      <h2>{스토어.상태}</h2>
      <button onClick={스토어.액션}>+</button>
    </>
  )
}
```

액션 분리

- 여러 컴포넌트에서 단일 스토어의 액션을 많이 사용한다면, 액션만 분리도 가능

```
import { create } from 'zustand'

export const useCountStore = create<{
  count: number
  actions: {
      inc: () => void,
      dec: () => void,
```

```
  }
}>(set => ({
  count: 1,
  actions: {
    inc: () => set(state => ({ count: state.count + 1 })),
    dec: () => set(state => ({ count: state.count - 1 }))
  }
}))
```

상태 초기화

- 상태를 초깃값으로 되돌릴 때, `resetState` 함수를 만들어서 간편하게 사용 가능
- (심화) 상태와 액션을 분리해서 타입과 초깃값을 작성하기

```
interface State {
  count: number
  double: number
  min: number
  max: number
}

interface Actions {
  actions: {
    increase: () => void
    decrease: () => void
    resetState: () => void
  }
}

const initialState: State = {
...
}
export const useCountStore = create<State & Actions>(set => ({
  ...initialState,
  actions: {
    increase: () => set(state => ({ count: state.count + 1 })),
    decrease: () => set(state => ({ count: state.count - 1 })),
    resetState: () => set(initialState)
  }
}))
```

```
import { useCountStore } from './store/count'

export default function resetState() {
  const { resetState } = useCountStore(state => state.actions)
  return (
    <>
      <button onClick={() => resetState()}>
        Reset All!
      </button>
    </>
  )
}
```