7/20 ~ 7/24 3주차 - 6

Vuex에서 store 활용 방법

Vuex의 주요 기능은 **개발하는 애플리케이션의 모든 컴포넌트에 대한 중앙 집중식 저장소 역활 및 관리이다.**

만약 이게 없다면 컴포넌트간 데이터를 주고 받기 위해서 부모는 자식에서 props의 방법으로, 자식은 부모에게 emit event의 방법으로 데이터를 주고 받아야 한다.

또한 형제 컴포넌트간 데이터를 주고 받으려면 너무 복잡해져서**EventBus를 활용해야** 그나마 사용을 할 수 있게 된다.

간단한 프로젝트인 경우에야 사용이 가능하겠지만 대형 프로젝트인 경우에는 이러한 방법으로는 도저히 감당이 되지 않고 개발 후 운영을 한다고 해도 힘든 부분이 있다

이러한 문제를 해결해 주는 것이 Vuex이다.

데이터를 store라는 파일을 통해 관리하고 프로젝트 전체에 걸쳐 활용할 수있게 해주는 방법이다.

State, Mutations, Actions, Getters의 특징

State

- state는 쉽게 말하면 프로젝트에서 공통으로 사용할 변수를 정의 한다.
- 프로젝트 내의 모든 곳에서 참조 및 사용이 가능하다.
- state를 통해 각 컴포넌트에서 동일한 값을 사용할 수 있다.

Mutations

- 주요 목적은 state를 변경 시키는 역할
- 비동기 처리가 아니라 동기 처리를 한다.
- 위의 함수가 실행되고 종료된 후 그 다음 아래의 함수가 실행됨
- commit('함수명', '전달인자')으로 실행 시킬 수 있다.
- mutations 내에 함수 형태로 작성된다.

Actions

- Mutations를 실행시키는 역할
- 비동기 처리를 한다.
- 순서에 상관없이 먼저 종료된 함수의 피드백을 받아 후속처리를 한다.
- dispatch('함수명', '전달인자')으로 실행 시킬 수 있다.
- actions내에 함수 형태로 작성한다.
- 비동기 처리이기 때문에 콜백함수로 주로 작성한다.

일반 형태로 실행

```
dispatch('setAccount', account );
export const actions = {
 setAccount({ commit, dispatch }, account) {
     commit('currentUser', account);
      dispatch('setIsAdmin', account.uid);
  }
```

```
}
```

Components에서 then()으로 콜백함수 실행

```
dispatch('setAccount', account ).then(() => { });
export const actions = {
  setAccount({ commit }, account) {
    return new Promise((resolve, reject) =>
    { setTimeout(() => {
        commit('currentUser', account);
        resolve() },
        1000) })
 }
}
```

Getters

- 각 Components의 계산된 속성(computed)의 공통 사용 정의이다.
- 여러 Components에서 동일한 computed가 사용 될 경우 Getters에 정의하여 공통으로 쉽게 사용할 수 ㅇ릿다.
- 하위 모듈의 getters를 불러오기 위해서는 특이하게 this.$store.getters["경로명/함수명"]을 사용해야 한다.