프론트엔드 개발자를 위한 Vue.js 시작하기

Vue.js 시작하기

Vuex 활용

한국기술교육대학교
온라인평생교육원

학습내용

- Module, State
- Action, Getter

학습목표

- Module과 State 사용 방법에 대해 설명할 수 있다.
- Action과 Getter 사용 방법에 대해 설명할 수 있다.

Module, State

Module, State

1 Module

Module

- Store에는 actions, mutations, state가 등록됨
- 애플리케이션 내에 여러 개 Store 이용 가능

Module, State

1 Module

Module

- 애플리케이션 규모가 크면 상태별로 Store를 여러 개 이용하는 것이 효율적임
- Store를 모듈화해서 여러 개 등록하고 구분해서 사용하는 기법

Module, State

Module, State

2 모듈선언

- 모듈은 state, actions, mutations를 가지는 Object Literal임

Module, State

2 모듈선언

namespaced: true

- 모듈을 이름으로 구분해서 사용하겠다는 의미

```
const countStore = {
    namespaced: true,
    state() { },
    actions : { },
    mutations: { }
}
```

Module, State

Module, State

3 Module 등록

- Object Literal로 선언된 모듈을 modules 속성으로 등록해서 사용함

```
const store = createStore({
    modules: {
      countStore: countStore,
      todoStore: todoStore
    }
})
```

Module, State

4 Module의 이용

Module의 Action 이용

- Action 함수 앞에 모듈명을 추가해 이용함

```
this.$store.dispatch('countStore/increase')
```

Module, State

Module, State

4 Module의 이용

Module의 State 이용

- this.$store.state 뒤에 모듈명을 추가해 이용함

```
this.$store.state.countStore.count
```

Module, State

5 State

- 애플리케이션의 상태를 의미함
- Object Literal로 선언됨

```
state() {
        return {
            count: 0
        }
},
```

Module, State

Module, State

6 this.$store로 이용

```
export default {
  computed: {
    count() {  return this.$store.state.test.count;  }
  },
};
```

Module, State

7 mapState 활용

- state 값을 쉽게 이용하기 위한 일종의 Helper
- mapState에 함수를 등록하면 매개변수로 state를 전달해 줌

```
import { mapState } from 'vuex'
export default {
  computed: mapState({
    myCount: state => state.test.count,
    myCount2 (state) {
      return state.test.count
    }
  }),
```

Module, State

Module, State

7 mapState 활용

- state 값을 직접 mapState를 이용해 획득 가능

```
...mapState('test',['count']),
```

Action, Getter

Action, Getter

1 Action

- 애플리케이션의 State 값 변경을 위해 호출되는 함수

```
actions : {
        increase({commit }) {
            commit('increase')
        }
},
```

Action, Getter

2 Action 함수 호출

- this.$store의 dispatch 함수를 이용해 호출함

```
this.$store.dispatch('test/testAction', 100)
```

Action, Getter

Action, Getter

2 Action 함수 호출

- mapActions 함수 이용
- Action 함수를 쉽게 이용하기 위한 Helper

```
... mapActions("test", ["testAction"])
```

Action, Getter

3 Getter

- 필수 구성요소는 아님
- Getter는 state를 이용하기 위한 함수

Action, Getter

Action, Getter

4 Getter 선언

- 매개변수로 state가 전달되며 두번째 매개변수로 다른 Getter를 이용할 수 있는 getters가 전달됨

```
const getters = {
    doneTodos: state => {
        return state.todos.filter(todo => todo.done)
    },
    doneTodosCount: (state, getters) => {
        return getters.doneTodos.length
    }
}
```

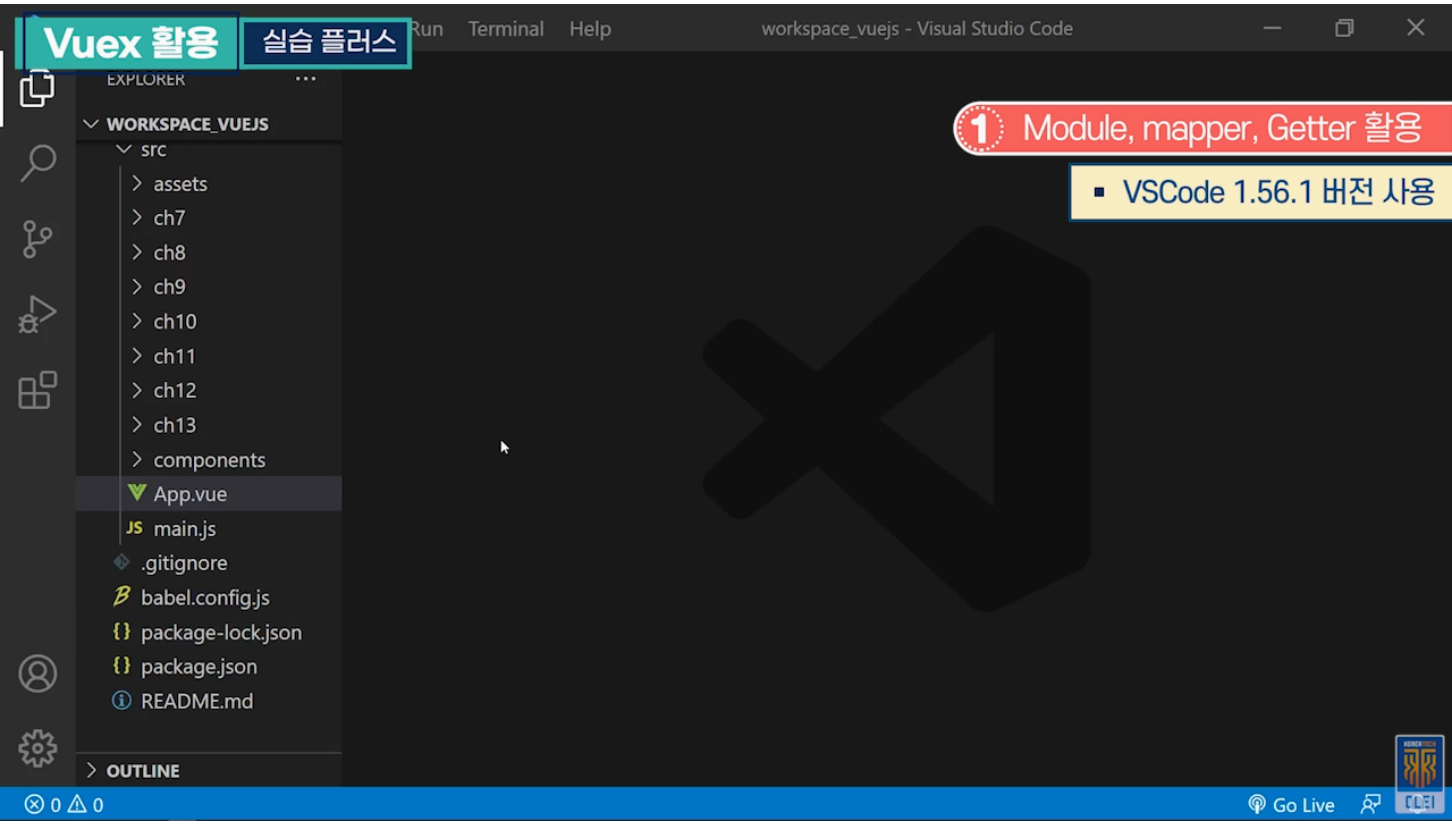
Action, Getter

5 Getter 이용

- mapGetters() 함수로 Getter를 이용함

```
import { mapState, mapGetters } from "vuex";
export default {
  computed: {
    ...mapGetters('test', {
      doneTodosCount: 'doneTodosCount'
    })
  },
```

실습 플러스



실습단계
src → ch14 폴더 생성, 서브 폴더 store 생성
Vuex 이용할 때 나오는 많은 구성요소들을 담기 위한 서브 폴더
store 폴더에 countStore.js 파일 생성
애플리케이션 전역에서 유지되어야 되는 데이터 중 count값과 관련되어 있는 데이터를 관리하기 위한 자바스트립트
애플리케이션 전역 데이터를 구분하기 위해 namespaced 선언
countStore에 의해서 관리되는 데이터를 이름 값으로 구분하여 이용하기 위한 설정
관리 되어야하는 데이터를 state에 등록
action 발생 시 호출될 함수를 actions에 등록, action 함수가 호출됐을 때 자체적인 업무 로직을 함수에 넣음
최종 데이터 변경을 위해 commit 함수 호출, mutation의 increase 함수 호출 → mutation 등록
mutation에 increase 함수가 호출됐을 때, 자체적으로 유지하고 있는 state 데이터 변경
Component에서 이용 편의성을 위해 getter라는 구성요소가 추가될 수 있음
store의 state 데이터를 component에서 이용, getter 없이도 이용 가능하지만 getter를 등록할 경우 편의성 있음
store → todoStore.js 파일 생성

실습 플러스

실습단계
state 데이터 변경을 위해 호출되는 함수가 action 함수
새로운 todo 글이 함수가 호출하면서 전달됨, 식별을 위한 id 값을 유지하기 위해서 기존 id 값에서 1을 더함
state 변경을 위한 commit 함수 호출
commit 함수에서 전달한 데이터가 두 번째 매개변수로 전달
외부에서 이용 가능하도록 export 시킴
Vuex 이용을 위한 Component 정의
Ch14 → AddComponent 파일 생성
API를 먼저 import 받음
두 개의 데이터는 애플리케이션 전역이 아니라 Component 내부에서만 유지하는 데이터
Vuex에 action을 발생시키기 위해 mapper 이용
화면 구성을 위해 template 정의
Two-Way Data Binding 기법으로 유저가 입력한 내용을 자동으로 데이터에 저장되게 처리
버튼 눌렀을 때 addItemAction 호출
Ch14에 ListComponent.vue 파일 생성
API를 import 받고 computed를 mapper 이용하여 정의
mapper를 이용하면, 함수 스타일로 등록하고 매개변수에 이용하고자 하는 state 값이 전달되어 편리성 있음
Ch14에 Vue 파일 Home.vue 제작
script 먼저 선언하여 두 개의 Component를 import받고, Vuex import 받음
데이터를 Vuex에 발생시키기 위한 action 함수를 mapper로 선언한 것
computed도 mapper 이용
component 등록
template으로 화면 구성
전체적으로 Home에 의해서 ListComponent와 AddComponent가 조합해서 나오는지 확인
이전에 생성한 ch13의 index.html, main.js, vue.config.js 파일 복사하여 이용
Home을 import 받고 store도 import시킴
두 개로 구분되는 것을 module로 등록 → 각각 구분되는 store라고 보면 됨
Vue cli 명령으로 실행
브라우저에 Component 출력