

HTTP Request, vuex

HTTP Request

- Back-end 서버에 데이터 저장/조회 등을 요청하기 위해 사용
- Vue application이 Http client가 된다.
- 문서가 로딩 될 때 (mounted) 또는 클릭 등의 이벤트가 발생했을 때 요청한다.
- Vue와 함께 연동할 서버는 json을 반환하는 RESTful server가 적당하다.

실습

- axios: jquery의 ajax에 해당하는 통신 라이브러리.
 - `npm install --save axios`
- <https://jsonplaceholder.typicode.com/comments> 의 mock업(mockup) 서버를 이용해 실습을 진행해본다.
 - 각 comment는 다음 구조를 가진다.
 - { "postId":1, "id":1, "name":"str", "email":"str", "body":"str" }

실습

- Review.vue를 다음과 같이 수정한다.

```
1  <template>
2    <div id="app">
3      <h1>Reviews</h1>
4      <ul>
5        <li v-for="c in comments" :key="c.id">
6          <p> {{c.body}} </p>
7        </li>
8      </ul>
9    </div>
10 </template>
11
12 <script>
13 import axios from 'axios'
14
15 export default{
16   name: 'Reviews',
17   data(){
18     return { comments: [] }
19   },
20   mounted(){
21     axios.get('https://jsonplaceholder.typicode.com/comments')
22       .then(response=>{
23         console.log(response)
24         this.comments=response.data
25       })
26     .catch(err=>{
27       console.log(err)
28     })
29   }
30 };
31 </script>
```

Proxy

- 요청할 때 마다 서버의 전체 주소를 입력할 필요 없도록 설정
- 프로젝트 루트에 vue.config.js 파일을 생성하고 다음 내용을 입력

```
1  module.exports={
2    devServer:{
3      proxy:{
4        '/api':{
5          target:'https://jsonplaceholder.typicode.com',
6          changeOrigin:true,
7          pathRewrite:{
8            '^/api':''
9          }
10        }
11      }
12    }
13  }
```

Proxy

- Reviews.vue에서 서버 주소를 다음과 같이 수정

```
20     mounted(){
21         axios.get('/api/comments')
22         .then(response=>{
23             console.log(response)
24             this.comments=response.data
25         })
26         .catch(err=>{
27             console.log(err)
28         })
29     }
```

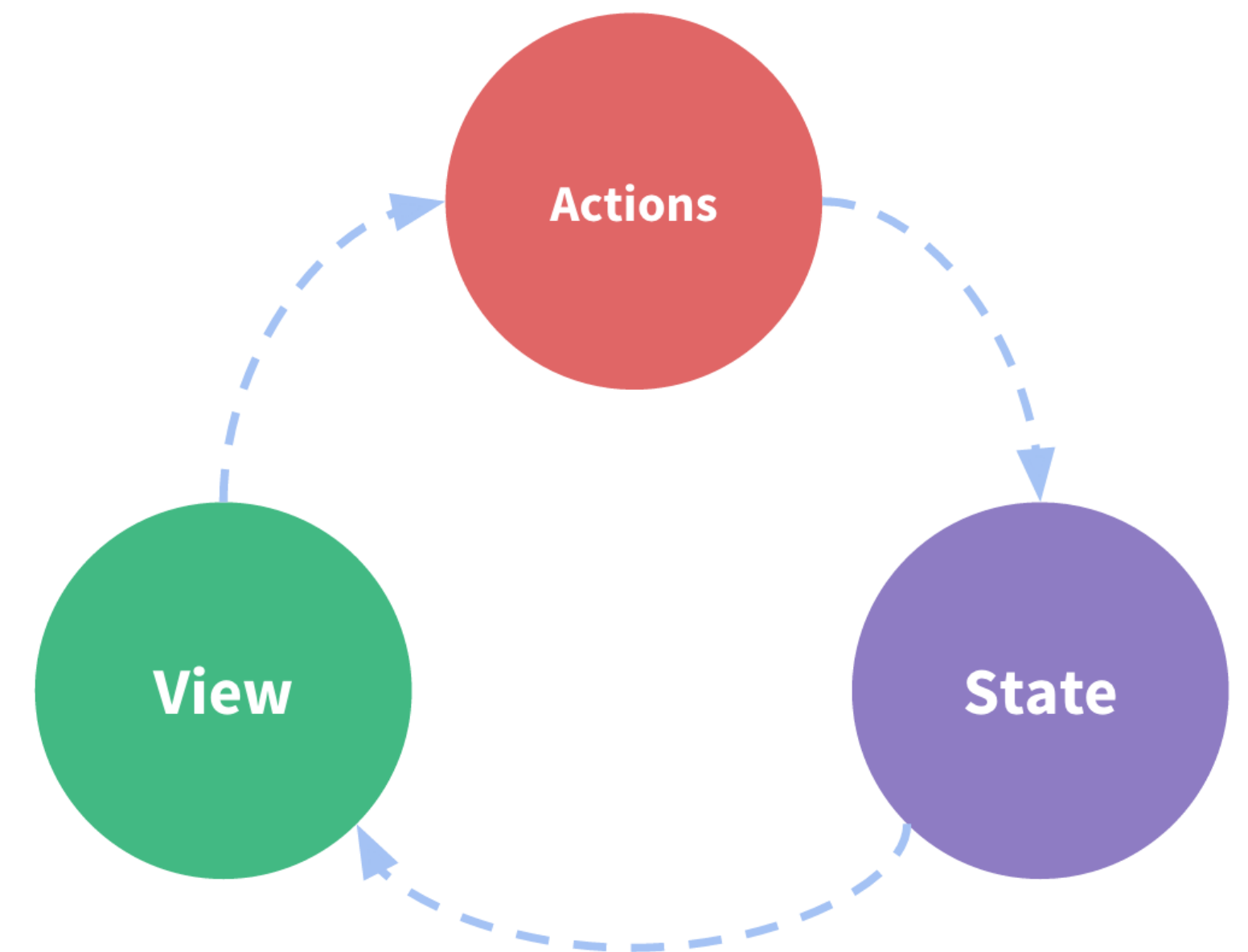
vuex: 저장소

준비

- `npm install --save axios vue-router veux es6-promise`

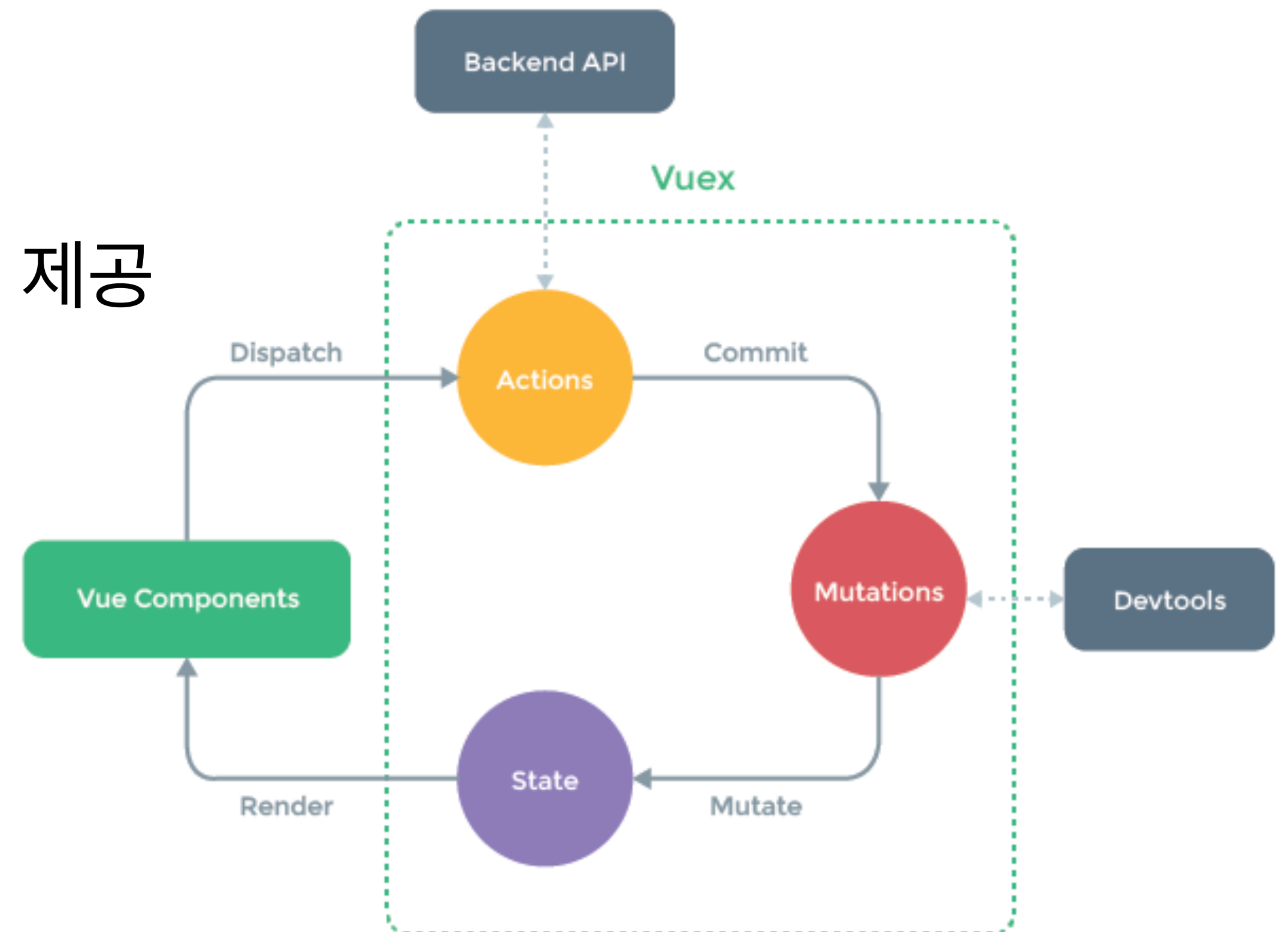
Vuex

- Vue.js 애플리케이션에 대한 상태 관리 패턴 + 라이브러리
- 일반적인 컴포넌트는 다음과 같이 동작
 - 뷰에서 사용자 입력이 발생
 - 액션이 상태를 바꿈
 - 상태가 뷰에 반영됨



Vuex

- 중첩된 컴포넌트를 통과해야 하는 props가 많을 때
- 이벤트를 이용해 동기화 하는 일이 많을 때
- 전체 컴포넌트들이 사용하는 싱글톤의 데이터를 제공
- IE 브라우저를 지원하려면 다음도 설치
 - `npm install --save es6-promise`



<https://vuex.vuejs.org/kr/>

vuex 설정

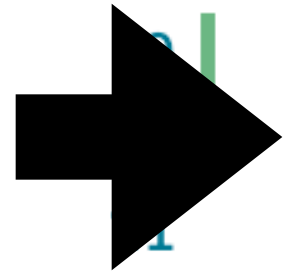
- src 폴더에 store 폴더를 추가하고 store.js 파일을 추가

```
1  import Vue from 'vue'
2  import Vuex from 'vuex'
3
4  Vue.use(Vuex);
5
6  export default new Vuex.Store({
7    state:{
8      |      userId: 'user'
9    }
10  });
```

vuex 설정

- main.js

```
1  import Vue from 'vue'
2  import App from './App.vue'
3  import router from './router/router'
4  import store from './store/store'
5
6  Vue.config.productionTip = false
7
8  new Vue({
9    router,
10    store:store,
11    render: h => h(App),
12  }).$mount('#app')
```



vuex 설정

- Home.vue에서 테스트

```
1  <template>
2  |    <h1>Welcome {{$store.state.userId}}</h1>
3  </template>
4
5  <script>
6  export default{
7  }
8  </script>
```

state

- 각 앱은 하나의 저장소를 가짐
- root Instance에 store를 전달함으로써 모든 하위 컴포넌트들이 사용할 수 있음
- 값 또는 함수를 반환할 수 있음
- 변화 상황을 추적하기 위해 직접 수정은 허용되지 않으며
- commit, dispatch 등을 통해서만 값을 수정할 수 있음.

mutations

- state를 변경하는 동작.
- mutations에 정의된 동작을 직접 호출할 수는 없으며 commit을 이용해 변경가능
- mutations의 함수는 무조건 동기적이어야 함

mutations

- store.js

```
1  import Vue from 'vue'
2  import Vuex from 'vuex'
3
4  Vue.use(Vuex);
5
6  export default new Vuex.Store({
7    state: {
8      userId: 'user'
9    },
10   mutations: {
11     updateUserId(state, newId) {
12       state.userId = newId;
13     }
14   }
15 });
```

Home.vue

```
1  <template>
2    <div id="app">
3      <h1>Welcome {{$store.state.userId}}</h1>
4      <input v-model="newId" type="text"><button @click="updateUserId">Save</button>
5    </div>
6  </template>
7
8  <script>
9  export default {
10    data() {
11      return { newId: '' }
12    },
13    methods: {
14      updateUserId() {
15        this.$store.commit('updateUserId', this.newId)
16      }
17    }
18  }
19  </script>
```


actions

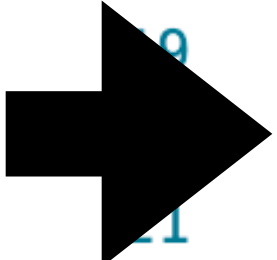
- state를 변이시키지만 비동기 작업이 포함될 수 있음.
 - state를 직접 변이하지 않고 작업이 완료되면 commit 함
 - 파라미터로 state가 아닌 저장소 instance의 method/속성들을 포함한 context를 받음
- store.dispatch 함수로 사용할 수 있음

actions

- store.js
 - axios 를 import한다
 - actions에 새 함수를 추가한다.



```
1  import Vue from 'vue'
2  import Vuex from 'vuex'
3  import axios from 'axios'
4
5  Vue.use(Vuex);
6
7  export default new Vuex.Store({
8    state: {
9      userId: 'user',
10     reviews: []
11   },
12   mutations: {
13     updateUserId(state, newId) {
14       state.userId = newId;
15     },
16     updateReviews(state, reviews) {
17       state.reviews = reviews
18     }
19   },
20   actions: {
21     getReviews({commit}) {
22       axios.get('/api/comments')
23         .then(response => {
24           commit('updateReviews', response.data)
25         })
26       .catch(err => {
27         console.log(err)
28       })
29     }
30   }
31 });
```



actions

- Home.vue
 - 리스트를 추가하고
 - dispatch 호출

```
1 <template>
2   <div id="app">
3     <h1>Welcome {{ $store.state.userId }}</h1>
4     <input v-model="newId" type="text"><button @click="updateUserId">Save</button><br>
5     <button @click="updateReviews">Update</button>
6     <ul>
7       <li v-for="r in $store.state.reviews" :key="r.id">
8         <p> {{r.body}}</p>
9       </li>
10    </ul>
11  </div>
12 </template>
13
14 <script>
15 export default{
16   data(){
17     return { newId:''}
18   },
19   methods:{
20     updateUserId(){
21       this.$store.commit('updateUserId', this.newId)
22     },
23     updateReviews(){
24       this.$store.dispatch('getReviews')
25     }
26   }
27 }
28 </script>
```

getters

- 저장소의 상태를 이용한 일종의 computed value.
- 계산된 값을 컴포넌트들이 쉽게 사용할 수 있게 함.
- `$store.getters.xxx` 로 사용 가능

getters

- store.js

```
7   export default new Vuex.Store({
8     state:{
9       userId: 'user',
10      reviews: []
11    },
12    mutations:{
13      updateUserId(state, newId){
14        state.userId = newId;
15      },
16      updateReviews(state, reviews){
17        state.reviews = reviews
18      }
19    },
20    actions:{
21      getReviews({commit}){
22        axios.get('/api/comments')
23          .then(response=>{
24            commit('updateReviews', response.data)
25          })
26          .catch(err=>{
27            console.log(err)
28          })
29      }
30    },
31     getters:{
32      reviewCount(state /* , getters */){
33        return state.reviews.length
34      }
35    }
36  });
```

getters

- Home.vue
 - script는 동일

```
1  <template>
2    <div id="app">
3      <h1>Welcome {{ $store.state.userId }}</h1>
4      <input v-model="newId" type="text"><button @click="updateUserId">Save</button><br>
5      <button @click="updateReviews">Update</button>
6      <h1>{{ $store.getters.reviewCount }}</h1>
7      <ul>
8        <li v-for="r in $store.state.reviews" :key="r.id">
9          <p> {{r.body}} </p>
10        </li>
11      </ul>
12    </div>
13 </template>
```



module

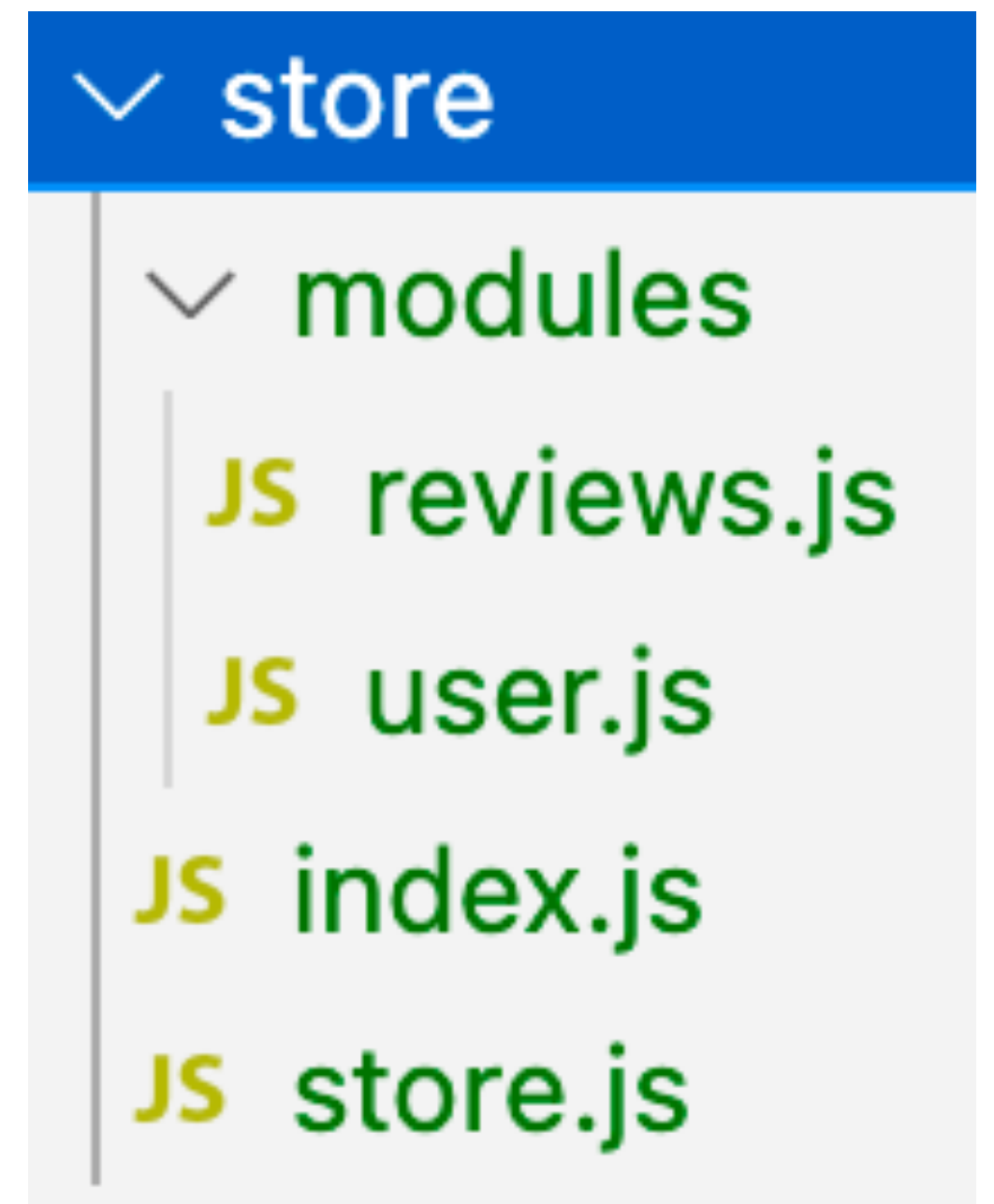
- 저장소가 방대해질 경우 모듈로 나눌 수 있음

```
├─ index.html
├─ main.js
├─ api
│   └─ ... # API 요청을 위한 추상화를 포함합니다.
├─ components
│   ├── App.vue
│   └─ ...
└─ store
    ├── index.js          # 모듈을 조합하고 저장소를 내보내는 곳 입니다.
    ├── actions.js       # 루트 액션
    ├── mutations.js     # 루트 변이
    └─ modules
        ├── cart.js      # cart 모듈
        └─ products.js   # products 모듈
```

<https://vuex.vuejs.org/kr/guide/structure.html>

module

- user 모듈과 review 모듈로 나누어본다.
- store 폴더에 modules 폴더를 추가하고
- user.js 와 reviews.js를 추가한다.
- store 폴더에는 index.js를 추가한다.



module

- user.js
 - state가 함수인 점을 주의한다.

```
1  export default {  
2      state: () => ({  
3          |      userId: 'user'  
4      |      } ),  
5      mutations: {  
6          |      updateUserId(state, newId) {  
7              |      state.userId = newId;  
8          |      }  
9      |      }  
10 };
```

module

- reviews.js

```
1  import axios from 'axios'
2
3  export default {
4    state: () => ({
5      reviews: []
6    }),
7    mutations: {
8      updateReviews(state, reviews) {
9        state.reviews = reviews
10      }
11    },
12    getters: {
13      reviewCount(state /* , getters, rootState */) {
14        return state.reviews.length
15      }
16    },
17    actions: {
18      getReviews({commit}) {
19        axios.get('/api/comments')
20          .then(response => {
21            commit('updateReviews', response.data)
22          })
23          .catch(err => {
24            console.log(err)
25          })
26      }
27    }
28  }
```

module

- index.js

```
1  import Vue from 'vue'
2  import Vuex from 'vuex'
3  import user from './modules/user'
4  import reviews from './modules/reviews'
5
6  Vue.use(Vuex)
7
8  export default new Vuex.Store({
9    modules:{
10      user,
11      reviews
12    }
13  })
```

- main.js (4번 라인)

```
1  import Vue from 'vue'
2  import App from './App.vue'
3  import router from './router/router'
4  import store from './store'
5
6  Vue.config.productionTip = false
7
8  new Vue({
9    router,
10   store:store,
11   render: h => h(App),
12 }).$mount('#app')
```

module

- Home.vue

```
1  <template>
2    <div id="app">
3      <h1>Welcome {{$store.state.user.userId}}</h1>
4      <input v-model="newId" type="text"><button @click="saveUserId">Save</button><br>
5      <button @click="updateReviews">Update</button>
6      <h1>{{$store.getters.reviewCount}}</h1>
7      <ul>
8        <li v-for="r in $store.state.reviews.reviews" :key="r.id">
9          <p> {{r.body}}</p>
10        </li>
11      </ul>
12    </div>
13  </template>
14
15  <script>
16    import {mapState, mapGetters, mapMutations, mapActions} from 'vuex'
17
18    export default{
19      data(){
20        return { newId: ''}
21      },
22      methods:{
23        saveUserId(){
24          this.$store.commit('updateUserId', this.newId)
25        },
26        updateReviews(){
27          this.$store.dispatch('getReviews')
28        }
29      }
30    }
31  }
32  </script>
```

module

- namespaces
 - 서로 다른 모듈이 같은 이름의 action이나 mutation을 가질 수도 있음
 - 이 경우 각각의 module 이름으로 구분해서 불러줌

module

- review.js에 namespace를 적용한 경우

Home.vue

```
1 import axios from 'axios'
2
3 export default {
4   namespace: true,
5   state: () => ({
6     reviews: []
7   })
```

```
<h1>{{ $store.getters['reviews/reviewCount'] }}</h1>

updateReviews() {
  this.$store.dispatch('reviews/getReviews')
}
```

Map...

- store에 선언된 값들을 간단하게 사용할 수 있도록 재정의 해 줌
 - mapState, mapMutations, mapActions, mapGetters

Map...

```
1  <template>
2    <div id="app">
3      <h1>Welcome {{userId}}</h1>
4      <input v-model="newId" type="text"><button @click="saveUserId">Save</button><br>
5      <button @click="updateReviews">Update</button>
6      <h1>{{reviewCount}}</h1>
7      <ul>
8        <li v-for="r in reviews" :key="r.id">
9          <p> {{r.body}}</p>
10        </li>
11      </ul>
12    </div>
13 </template>
```


Map...

```
15 <script>
16 import {mapState, mapGetters, mapMutations, mapActions} from 'vuex'
17
18 export default{
19   data(){
20     return { newId: '' }
21   },
22   computed:{
23     // namespaced 가 설정되지 않은 경우
24     ...mapState({
25       |   userId:state=>state.user.userId
26     }),
27     // namespaced 가 설정 된 경우 모듈명을 적는다.
28     ...mapState('reviews', {
29       |   reviews:state=>state.reviews
30     }),
31     ...mapGetters('reviews', ['reviewCount'])
32   },
33   methods:{
34     ...mapMutations(['updateUserId']),
35     ...mapActions('reviews', ['getReviews']),
36     saveUserId(){
37       |   this.updateUserId(this.newId)
38     },
39     updateReviews(){
40       |   this.getReviews()
41     }
42   }
43 }
44 }
45 </script>
```