

→ App.vue 컴포넌트나 index.html의 div#app 태그를  
명시적 권장으로 index.html을 통해 app.vue를 실행된다.

h: hyperlink의 약자  
HTML 구조를 사용하는 스크립트는 아니.

EXPLORER OPEN EDITORS

App.vue X

src > frontend > src > App.vue > template

1 <template> *→ 다른 컴포넌트로 변경*  
2 <v-content>  
3 <router-view/> *→ 라우터가 뷰를 선택*  
4 </v-content>  
5 </template>

vuexify.js router index.js

store views App.vue main.js .gitignore babel.config.js package-lock.json package.json README.md vue.config.js

main java/com/example/demo config DBConfig.java controller entity game nativeinterface repository

OUTLINE TIMELINE SONARLINT RULES JAVA PROJECTS SPRING BOOT DASHBOARD

index.js X

src > frontend > src > router > index.js ...

```
import Vue from 'vue'  
import VueRouter from 'vue-router'  
import Home from '../views/Home.vue'  
import About from '../views/About'  
import BoardListPage from "@views/BoardListPage";  
import BoardRegisterPage from "@views/BoardRegisterPage";  
Vue.use(VueRouter)
```

Component? 기본 HTML 예제들을 확장하여 새 사용자 환경을 제공하는  
설정이나 상태를 정의하는 Vue의 정의에 의해 동작이 처리된 사용자  
환경 예제입니다.

→ 사용한 Component import는?

router.js

Path: 라우터(Componet)에 할당할 URL 주소.  
name: 라우터 이름  
Component: 대상 시점 컴포넌트  
(Views 대상에 Vue 컴포넌트 사용)

```
const routes = [  
  {  
    path: '/',  
    name: 'Home',  
    component: Home  
  },  
  {  
    path: '/about',  
    name: 'About',  
    component: About  
  },  
  {  
    path: '/board',  
    name: 'BoardListPage',  
    components: {  
      default: BoardListPage  
    }  
  },  
  {  
    path: '/board/create',  
    name: 'BoardRegisterPage',  
    components: {  
      default: BoardRegisterPage  
    }  
  }  
]  
const router = new VueRouter({  
  mode: 'history',  
  base: process.env.BASE_URL,  
  routes  
})  
export default router
```

Ln 36, Col 1 Spaces: 2 UTF-8 LF JavaScript

## #전역등록

```
JS  
new Vue({  
  el: '#some-element',  
  // 옵션  
})
```

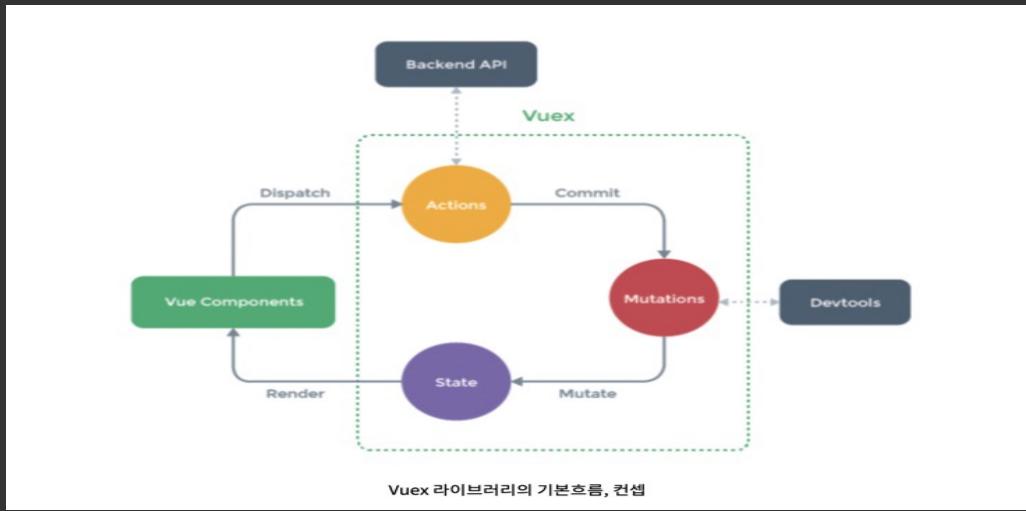
```
JS  
Vue.component('my-component', {  
  // 옵션  
})
```

# 컴포넌트는 인스턴스의 렝플릿에서 커스텀어的文字 사용 가능

```
HTML  
<div id="example">  
  <my-component></my-component>  
</div>
```

```
JS  
// 등록  
Vue.component('my-component', {  
  template: '<div>사용자 정의 컴포넌트 입니다!</div>'  
})  
  
// 루트 인스턴스 생성  
new Vue({  
  el: '#example'  
})
```

```
HTML  
<div id="example">  
  <div>사용자 정의 컴포넌트 입니다!</div>  
</div>
```



## 1). 상태 (State)

- 단일 상태트리 (단일 객체), 어떤 컴포넌트가 사용하는 하나의 원본 데이터.
- 모든 component는 데이터가 필요할 때, 이 원본 데이터를 가져서 사용.
- 디버깅이 쉽다는 장점.
- Vuex 스토어의 State에 접근해서 Vue 헐포워드의 Computed에 활용하면 된다.
- 상태 값은 변경되면 자동으로 Computed 속성이 자동으로 업데이트됨.

## 2).突변(Mutation)

- 상태 (State) 값을 변경하는 유파스 방법.

①突변 핸들러 생성

```
mutations: [ ]
```

②스토어의 Commit 메소드 활용.

```
this.board.commit('increment')
```

- Commit 메소드를 실행하면 Payload를 전달 가능. Payload는 상태 값을 업데이트할 때 사용.  
 대부분 Payload는 여러 파트를 포함할 수 있는 Object로 전달.

③Object Style Commit

```
this.board.commit({
  type: 'increment',
  amount: 10
})
```

### \* Reactivity Rule

Vuex 스토어의 상태 value는 Vue에 의해 변동

→ 상태 값 변경되면 Vue Components는 자동으로 업데이트됨.

→ 변경되었던 데이터 미리 초기화 / 새롭게 객체로 교체.

\* Mutation type은 상수 사용 지향!

### 3). 액션(Action)

- State / Mutation 의 모든 변경이 끝났을 때 execute Commit.  
⇒ 변경을 확정 하는 일.
- 이동기 처리를 따로 번역을 Commit 하고 Vuex 스토어의 데이터를 업데이트.

### 4). Getter

- Computed 와 같은 속성.
- 직접 수정하는 것의 아닌 상태 값을 활용해서 계산된 속성을 사용.
- State에 접근해서 computed 하거나 사용  
⇒ State 값 변화면 Getter 값도 변화.

## ❖ 모듈화

- Vuex 단일 스토어를 모듈 단위로 조작할 수 있다.
- 별도 Module은 파일로 관리한다.
- Store 모듈의 핵심: **namespaced**  
→ 모든 동작적 사용: true 사용하면됨.
- namespaced 모듈 내부에서 다른 모듈이나 전역 데이터 접근  
→ **rootState**, **rootGetter** .... 활용.

## \* Store Package

store

- actions.js
- index.js
- mutation-types.js
- mutations.js
- states.js

## \* States

states.js X

src > frontend > src > store > states.js > default

```

1 export default {
2   boards: [],
3   board: null
4 }
```

## \* mutations

mutation-types.js X

src > frontend > src > store > mutation-types.js > default

```

1
2 export const FETCH_BOARD_LIST = 'FETCH_BOARD_LIST'
3 export const FETCH_BOARD = 'FETCH_BOARD'
```

## \* mutations

mutations.js X

src > frontend > src > store > mutations.js > default

```

1 import {
2   FETCH_BOARD_LIST,
3   FETCH_BOARD
4 } from './mutation-types'
5
6 export default {
7   [FETCH_BOARD_LIST] (state, boards) {
8     state.boards = boards
9   },
10  [FETCH_BOA any] (state, board) {
11    state.board = board
12  }
13 }
```

## \* actions

actions.js X

src > frontend > src > store > actions.js > default

```

1
2 import {
3   FETCH_BOARD_LIST,
4   FETCH_BOARD
5 } from './mutation-types'
6
7 import axios from "axios";
8
9 export default {
10   fetchBoardList ({ commit }) {
11     return axios.get("http://localhost:7777/boards")
12       .then(res => {
13         commit(FETCH_BOARD_LIST, res.data)
14       })
15   },
16   fetchBoard ({ commit }, boardNo) {
17     console.log(`fetchBoard ${commit}, boardNo = ${boardNo}`)
18
19     return axios.get(`http://localhost:7777/boards/${boardNo}`)
20       .then(res => {
21         commit(FETCH_BOARD, res.data)
22       })
23   }
24 }
```

## \* total index.

index.js X

src > frontend > src > store > index.js > ...

```

1
2 import Vue from 'vue'
3 import Vuex from 'vuex'
4
5 import states from "@/store/states";
6 import actions from "@/store/actions";
7 import mutations from "@/store/mutations";
8
9 Vue.use(Vuex)
10
11 export default new Vuex.Store({
12   states,
13   mutations,
14   actions,
15   //getters
16 })
```

Vue와 MySQL 연동 시키는 파일.

```
1 package com.example.demo.config;
2
3 import org.springframework.context.annotation.Bean;
4 import org.springframework.context.annotation.Configuration;
5 import org.springframework.jdbc.datasource.DriverManagerDataSource;
6
7 import javax.sql.DataSource;
8
9 @Configuration
10 public class DBConfig {
11     @Bean
12     public DataSource dataSource() {
13         DriverManagerDataSource dataSource =
14             new DriverManagerDataSource();
15         dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver");
16         dataSource.setUrl(
17             "jdbc:mysql://localhost/vue_board?serverTimezone=UTC&useSSL=false"
18         );
19         dataSource.setUsername("root");
20         dataSource.setPassword("dkdlxl123");
21
22     }
23     return dataSource;
24 }
25 }
```

DB 연결.

src > frontend > src > views >  BoardListPage.vue > {} "BoardListPage.vue" >  template

```
1 <template>
2   <div id="board">
3     <h2>Board List</h2>
4     <router-link :to="{ name: 'BoardRegisterPage' }">
5       새로운 글쓰기
6     </router-link>
7     <board-list :boards="boards"/>
8   </div>
9 </template>
```

```
10
11 <script>
12 import BoardList from "@/components/BoardList";
13 import { mapState, mapActions } from 'vuex';
14 export default {
15   name: "BoardListPage",
16   components: {
17     BoardList
18   },
19   computed: {
20     ...mapState(['boards'])
21   },
22   mounted() {
23     this.fetchBoardList()
24   },
25   methods: {
26     ...mapActions(['fetchBoardList'])
27   }
28 }
29 </script>
```

→ Board list 3 Export

Vuex Store

```
30
31 <style scoped>
32 #board {
33 | color: deepskyblue;
34 }
35 </style>
```

ମହାନ୍ତିକୁ ଲାଗୁ to Propose.

→ Bond list 3 export

Vux Store.

DBConfig.java ● index.js ● BoardList.vue ● actions.js ● mutation-types.js ●

src > frontend > src > components > BoardList.vue > script > default > props > boards

과제 list.html 파일과 같은 동작.

```
1  <template>
2    <div>
3      <h3>게시판 보기</h3>
4      <table border="1">
5        <tr>
6          <th align="center" width="80">번호</th>
7          <th align="center" width="320">제목</th>
8          <th align="center" width="100">글쓴이</th>
9          <th align="center" width="180">등록일자</th>
10         </tr>
11
12        <tr v-if="!boards || (Array.isArray(boards) && boards.length === 0)">
13          <td colspan="4">
14            목록이 비었습니다.
15          </td>
16        </tr>
17
18        <tr v-else v-for="board in boards" :key="board.boardNo">
19          <td align="center">{{ board.boardNo }}</td>
20          <td align="left">
21            <router-link :to="{ name: 'BoardReadPage',
22                          params: { boardNo: board.boardNo.toString() } }">?
23              {{ board.title }}
24            </router-link>
25          </td>
26          <td align="right">{{ board.writer }}</td>
27          <td align="center">{{ board.regDate }}</td>
28        </tr>
29      </table>
30    </div>
31  </template>
32  <script>
33    export default {
34      name: "BoardList",
35      props: {
36        boards: {
37          type: Array
38        }
39      }
40    }
41  </script>
42  <style scoped>
43  </style>
```

Ln 39, Col 6 Spaces: 2 UTF-8 LF Vue ⚡ ⓘ Prettier ⚡

src &gt; frontend &gt; src &gt; views &gt; BoardRegisterPage.vue &gt; script &gt; default

```

1   <template>
2     <div align="center">
3       <h2>게시판 글쓰기</h2>
4       <board-register-form @submit="onsubmit"/>
5     </div>
6   </template>
7
8   <script>
9   import BoardRegisterForm from "@/components/BoardRegisterForm";
10  import axios from 'axios';
11  export default [
12    name: "BoardRegisterPage",
13    components: {
14      BoardRegisterForm
15    },
16    methods: {
17      onsubmit (payload) {
18        console.log('BoardRegisterPage onSubmit()')
19        const { title, content, writer } = payload
20        axios.post('http://localhost:7777/boards',
21          { title, content, writer})
22        .then(res => {
23          console.log(res)
24          alert('등록이 잘 되었습니다.')
25          this.$router.push({
26            name: 'BoardReadPage',
27            params: { boardNo: res.data.boardNo.toString() }
28          })
29        })
30      }
31    }
32  ]
33  </script>
34
35  <style scoped>
36  </style>

```

axios: HTTP 통신을 하는데 사용하는

JS 라이브러리.

Promise 기반으로 async/await 사용하여 XHR 방식을 쉽게 할 수 있다.

- axios.get/.post ⇒ method 불가능.

버튼 클릭

상태값 바꾸기.

이벤트로 인한 텍스트 내용은

axios의 post method처럼 promise 처리.

BoardRegisterForm.vue

src > frontend > src > components >  BoardRegisterForm.vue > {} "BoardRegisterForm.vue" >  script >  default >  methods >  onsubmit

2) register.html과 속성 같음

```
1 <template>
2   <form @submit.prevent="onsubmit">
3     <h3>게시판 글쓰기</h3>
4     <table>
5       <tr>
6         <td>Title</td>
7         <td>
8           <input type="text" v-model="title"/>
9         </td>
10      </tr>
11      <tr>
12        <td>Writer</td>
13        <td>
14          <input type="text" v-model="writer"/>
15        </td>
16      </tr>
17      <tr>
18        <td>Content</td>
19        <td>
20          <textarea v-model="content"></textarea>
21        </td>
22      </tr>
23    </table>
24
25    <div>
26      <button type="submit">등록하기</button>
27      <router-link :to="{ name: 'BoardListPage' }">
28        취소
29      </router-link>
30    </div>
31  </form>
32</template>
33
34 <script>
35   export default {
36     name: "BoardRegisterForm",
37   }
38 </script>
```

DBConfig.java index.js BoardRegisterForm.vue actions.js mutation-types.js

src > frontend > src > components > BoardRegisterForm.vue > {} "BoardRegisterForm.vue" > script > default > methods > onsubmit

```
34  <script>
35  export default {
36    name: "BoardRegisterForm",
37    data () {
38      return {
39        title: '',
40        writer: '',
41        content: ''
42      }
43    },
44    methods: {
45      onsubmit () {
46        const { title, writer, content } = this
47        this.$emit(['submit', { title, writer, content }])
48      } 이벤트 발생 코드. : 등록하기 클릭 → onsubmit 메소드 → board-register-form 실행 ?
49    }
50  }
51  </script>
52
53  <style scoped>
54  </style>
```

Ln 47, Col 55 Spaces: 2 UTF-8 LF Vue Prettier