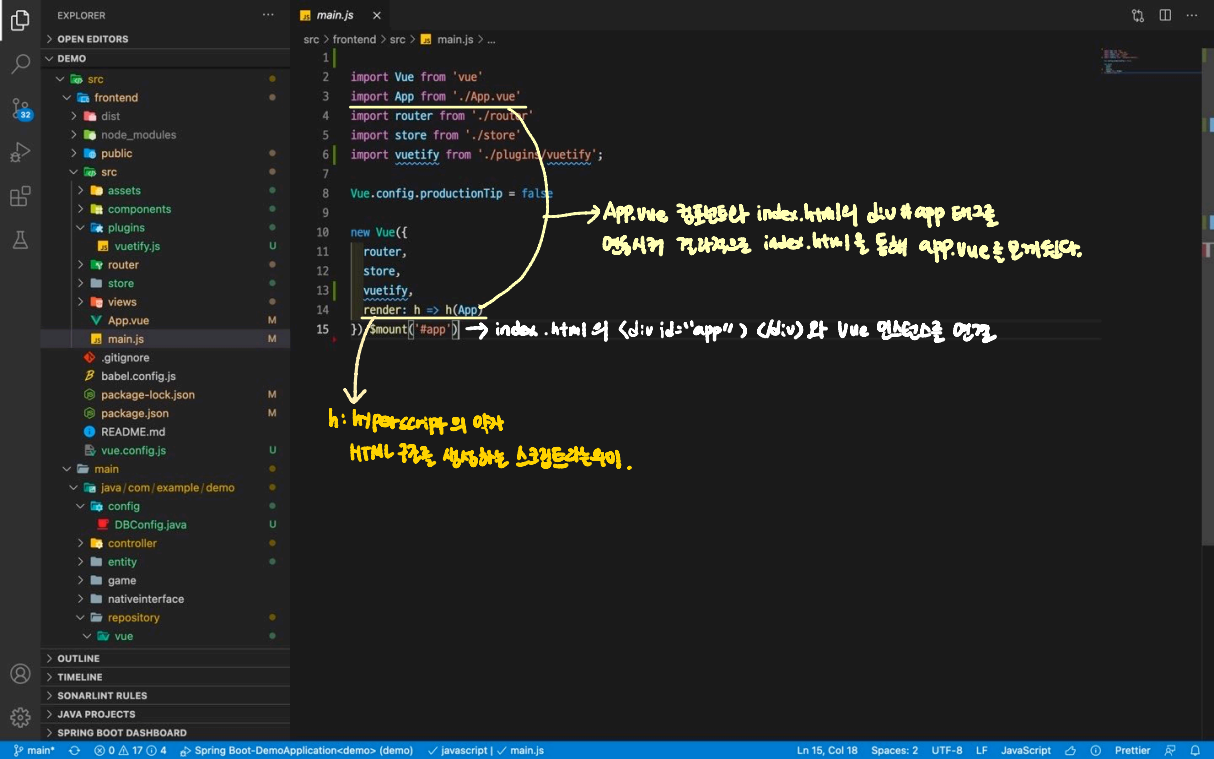
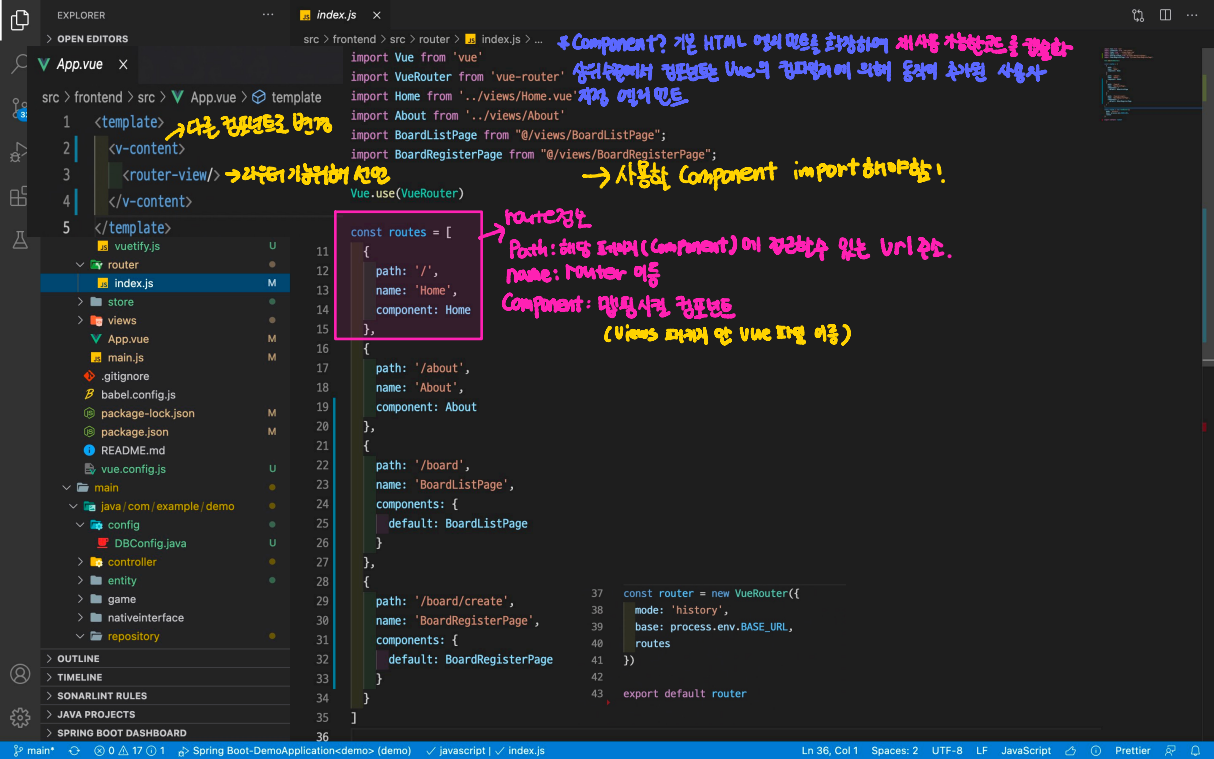

EXPLORER
OPEN EDITORS
DEMO
src
frontend
dist
node_modules
public
src
assets
components
plugins
vuetify.js
router
store
views
App.vue
main.js
.gitignore
babel.config.js
package-lock.json
package.json
README.md
vue.config.js
main
java / com / example / demo
config
DBConfig.java
controller
entity
game
nativeinterface
repository
vue
OUTLINE
TIMELINE
SONARLINT RULES
JAVA PROJECTS
SPRING BOOT DASHBOARD
main.js
src > frontend > src > main.js > ...
import Vue from 'vue'
import App from './App.vue'
import router from './router'
import store from './store'
import vuetify from './plugins/vuetify';
Vue.config.productionTip = false
new Vue({
router,
store,
vuetify,
render: h => h(App)
}).$mount('#app')
App.vue 컴포넌트와 index.html의 div #app 태그를
연결시켜 라우터로 index.html을 통해 app.vue를 보여준다.
index.html의  와 Vue 인스턴스를 연결
h: 뷰에서 render의 약자
HTML 구조를 생성하는 스크립트적이.
main*
Spring Boot-DemoApplication<demo> (demo)
javascript | main.js
Ln 15, Col 18
Spaces: 2
UTF-8
LF
JavaScript
Prettier


EXPLORER
OPEN EDITORS
App.vue
src > frontend > src > App.vue > template
1 <template>
2 <v-content>
3 <router-view/>
4 </v-content>
5 </template>
대표 컴포넌트로 지정
라우터 기능에서 사용
vuetify.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
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)
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
* Component? 개발 HTML 영역 태그를 효율적이고 재사용 가능하도록 캡슐화
상위수준에서 컴포넌트는 Vue의 컴파일러에 의해 동적 추가된 사용자
지정 엘리먼트
→ 사용할 Component import 해야함!
router정보
Path: 해당 페이지(Component)에 접근할 수 있는 Url 경로.
Name: router 이름
Component: 맵핑시킬 컴포넌트
(Views 패키지 안 Vue 파일 이름)
main*
0 17 1
Spring Boot-DemoApplication<demo> (demo)
javascript | index.js
Ln 36, Col 1
Spaces: 2
UTF-8
LF
JavaScript
Prettier

# 전역 등록

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

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

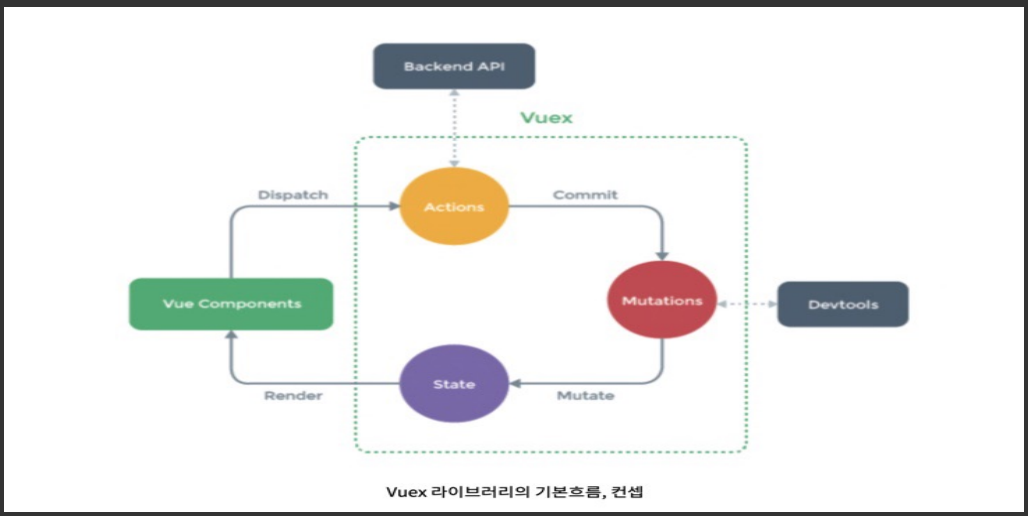
# 컴포넌트는 인스턴스의 템플릿에서 커스텀 엘리먼트 사용가능

```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>
```



Vuex 라이브러리의 기본흐름, 컨셉

## 1). 상태 (State)

- 단일 상태트리 (단일객체), 어플리케이션에 사용하는 하나의 원본데이터.
- 모든 component 는 데이터가 필요할때, 이원본 데이터를 꺼내서 사용.
- 디버깅이 쉽다는 장점.
- Vuex 스토어의 state에 정의해서 Vue 컴포넌트의 computed에 주입하면 된다.
- 상태 값이 변경되면, 자동으로 computed 역시 자동으로 변경, 업데이트 됨.

## 2). 뮤이 (Mutation)

- 상태 (State) 값을 변경하는 올바른 방법.

① 뮤이 핸들러 생성

```
mutations: {  }
```

② 스토어의 commit 메소드 호출.

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

- commit 메소드를 실행할때 Payload 값 전달가능. Payload는 상태 값을 변이 시킬때 사용.
  대부분 Payload는 여러 필드를 포함하는 Object로 전달.

③ Object Style Commit

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

### ※ Reactivity Rule

Vuex 스토어의 상태 value는 Vue에 의해 반응
→ 상태 값 변경되면 Vue Components 는 자동으로 업데이트.
⇒ 변경감지할 데이터 미리 초기화 / 새로운 객체로 교체.

※ Mutation type은 상수 사용 지향!

## 3). 액션 (Action)

- State/Mutation 의 모든 과정이 끝났을때 비로소 Commit.
  ⇒ 변경을 확정 짓는 일.
- 이들의 처리를 마치고 변이를 Commit 하고 Vuex 스토어의 데이터를 업데이트.

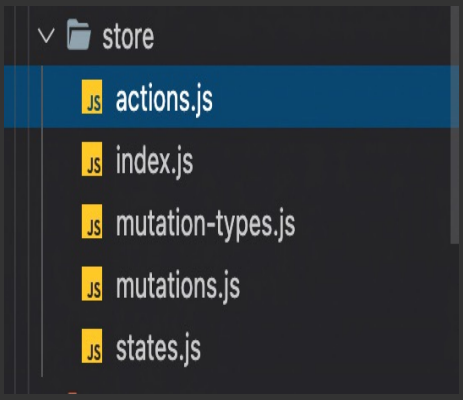
## 4). Getter

- Computed 와 같은 속성.
- 직접수정하는 값이 아닌 상태 값을 활용해서 계산된 속성으로 사용.
- State 에 접근해서 Computed 하듯 사용
  ⇒ State 값 변화면 Getter 값도 변한다.

## ✳ 모듈화.

- Vuex 단일 스토어를 모듈 단위로 쪼개수있다.
- 모든 Module은 파일로 관리한다.
- Store 모듈의 핵심 : namespaced
  → 모듈 독립적 사용 : true 사용하면된.
- namespaced 모듈 내부에서 다른모듈이나 전역 데이터 접근
  → rootState , rootGetter .... 활용.

## *Store Package




## *States

states.js

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

```
export default {
    boards: [],
    board: null
}
```

## *mutations

mutation-types.js

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

```

export const FETCH_BOARD_LIST = 'FETCH_BOARD_LIST'
export const FETCH_BOARD = 'FETCH_BOARD'
```

## *mutations

mutations.js

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

```
import {
    FETCH_BOARD_LIST,
    FETCH_BOARD
} from './mutation-types'

export default {
    [FETCH_BOARD_LIST] (state, boards) {
        state.boards = boards
    },
    [FETCH_BOA any state, board) {
        state.board = board
    }
}
```

## *actions

actions.js

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

```
import {
    FETCH_BOARD_LIST,
    FETCH_BOARD
} from './mutation-types'

import axios from "axios";

export default {
    fetchBoardList ({ commit }) {
        return axios.get('http://localhost:7777/boards')
            .then(res => {
                commit(FETCH_BOARD_LIST, res.data)
            })
    },
    fetchBoard ({ commit }, boardNo) {
        console.log('fetchBoard ' + commit + ', boardNo = ' + boardNo)

        return axios.get('http://localhost:7777/boards/${boardNo}')
            .then(res => {
                commit(FETCH_BOARD, res.data)
            })
    }
}
```

## *total index.

index.js

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

```
import Vue from 'vue'
import Vuex from 'vuex'

import states from "@/store/states";
import actions from "@/store/actions";
import mutations from "@/store/mutations";

Vue.use(Vuex)

export default new Vuex.Store({
    states,
    mutations,
    actions,
    //getters
})
```

```
computed: {
  //Vuex: 단일객체트리
  //이 단일 객체는 모든 애플리케이션 수준의 상태를 포함하며 "원본 소스" 역할을 합니다.
  //이는 각 애플리케이션마다 하나의 저장소만 갖게 된다는 것을 의미
  //단일 상태 트리는 모듈성과 충돌하지 않습니다.
  //그러면 Vue 컴포넌트에서 저장소 내부의 상태를 어떻게 표시하나요?
  //Vuex 저장소는 반응적이기 때문에 저장소에서 상태를 "검색"하는 가장 간단한 방법은
  //계산된 속성 (opens new window)내에서 일부 저장소 상태를 가져오는 것입니다.
  //그러나 이 패턴은 컴포넌트가 전역 저장소 단독 항목에 의존하게합니다.
  //모듈 시스템을 사용할 때는 저장소 상태를 사용하는 모든 컴포넌트에서 저장소를 가져와야하며 컴포넌트를 테스트 할 때는 가짜데이터가 필요합니다.
  //Vuex는 store 옵션(Vue.use(Vuex)에 의해 가능)으로 루트 컴포넌트의 모든 자식 컴포넌트에 저장소를 "주입"하는 메커니즘을 제공합니다.
  //컴포넌트가 여러 저장소 상태 속성이나 getter를 사용해야하는 경우 계산된 속성을 모두 선언하면 반복적이고 장황해집니다.
  //이를 처리하기 위해 우리는 계산된 getter 함수를 생성하는 mapState 헬퍼를 사용하여 키 입력을 줄일 수 있습니다.


  //States 안 보드
  ...mapState(['boards'])
},
```

DBConfig.java

src > main > java > com > example > demo > config > DBConfig.java > Language Support for Java(TM) by Red Hat > DBConfig > dataSource()

Vueer My sql 연동 시키는 파일.

```java
package com.example.demo.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.jdbc.datasource.DriverManagerDataSource;

import javax.sql.DataSource;

@Configuration
public class DBConfig {
    @Bean
    public DataSource dataSource() {
        DriverManagerDataSource dataSource =
                new DriverManagerDataSource();
        dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver");
        dataSource.setUrl(
                "jdbc:mysql://localhost/vue_board?serverTimezone=UTC&useSSL=false"
        );

        dataSource.setUsername("root");
        dataSource.setPassword("dkdlxl123");

        return dataSource;
    }
}
```

DB 이름.

main* 0 17 2 Spring Boot-DemoApplication<demo> (demo) java | DBConfig.java Ln 22, Col 1 Spaces: 4 UTF-8 LF Java JavaSE-11

BoardListPage.vue

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

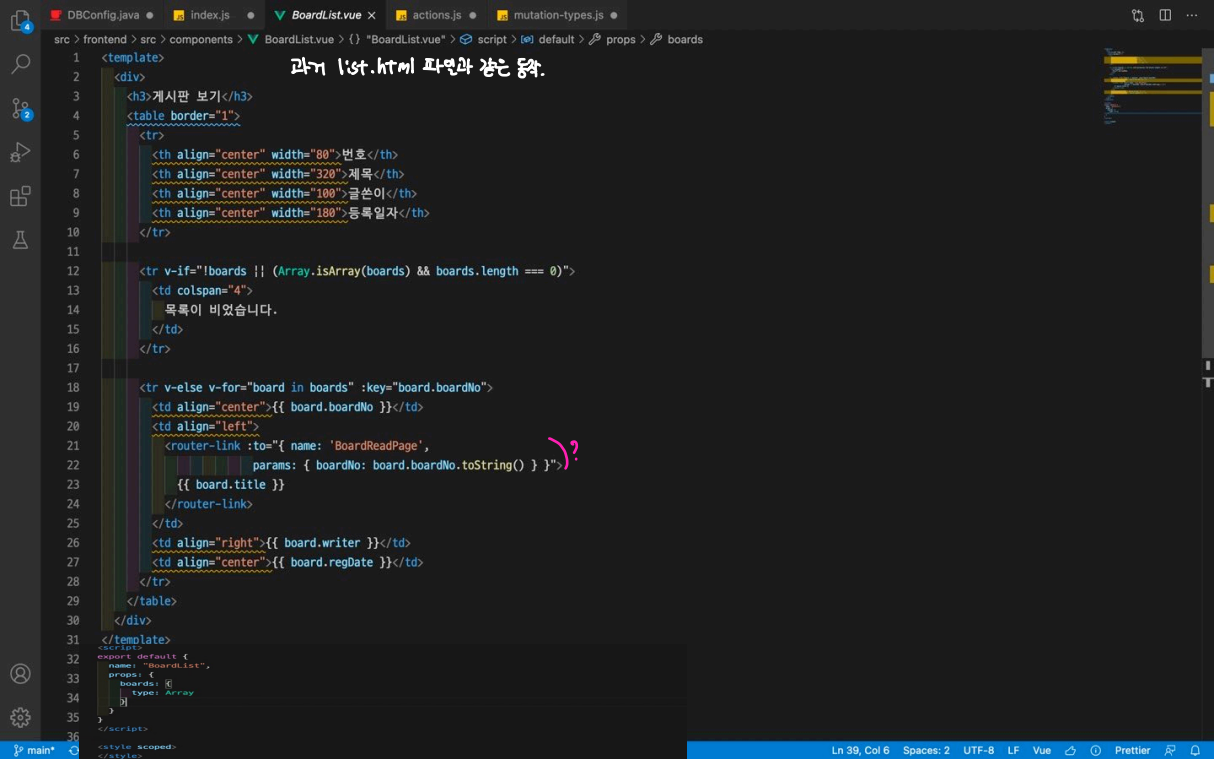
```vue
<template>
  <div id="board">
    <h2>Board List</h2>
    <router-link :to="{ name: 'BoardRegisterPage' }">
      새로운 글쓰기
    </router-link>
    <board-list :boards="boards"/>
  </div>
</template>

<script>
import BoardList from "@/components/BoardList";
import { mapState, mapActions } from 'vuex';
export default {
  name: "BoardListPage",
  components: {
    BoardList
  },
  computed: {
    ...mapState(['boards'])
  },
  mounted() {
    this.fetchBoardList()
  },
  methods: {
    ...mapActions(['fetchBoardList'])
  }
}
</script>

<style scoped>
#board {
  color: deepskyblue;
}
</style>
```



main* ⊗ 0 ⚠ 17 ⓘ 1 Spring Boot-DemoApplication<demo> (demo) vue | ✓ BoardListPage.vue Ln 9, Col 12 Spaces: 2 UTF-8 LF Vue Prettier


DBConfig.java
index.js
BoardList.vue
actions.js
mutation-types.js
src > frontend > src > components > BoardList.vue > {} "BoardList.vue" > script > default > props > boards
과거 list.ht에 파면과 같은 당착.
<template>

<h3>게시판 보기</h3>
<table border="1">
<tr>
<th align="center" width="80">번호</th>
<th align="center" width="320">제목</th>
<th align="center" width="100">글쓴이</th>
<th align="center" width="180">등록일자</th>
</tr>
<tr v-if="!boards || (Array.isArray(boards) && boards.length === 0)">
<td colspan="4">
목록이 비었습니다.
</td>
</tr>
<tr v-else v-for="board in boards" :key="board.boardNo">
<td align="center">{{ board.boardNo }}</td>
<td align="left">
<router-link :to="{ name: 'BoardReadPage',
params: { boardNo: board.boardNo.toString() } }">
{{ board.title }}
</router-link>
</td>
<td align="right">{{ board.writer }}</td>
<td align="center">{{ board.regDate }}</td>
</tr>
</table>

</template>
<script>
export default {
name: "BoardList",
props: {
boards: {
type: Array
}
}
}
</script>
<style scoped>
</style>
main*
Ln 39, Col 6
Spaces: 2
UTF-8
LF
Vue
Prettier

BoardRegisterPage.vue

src > frontend > src > views > BoardRegisterPage.vue > { } "BoardRegisterPage.vue" > script > default

```vue
<template>
  <div align="center">
    <h2>게시판 글쓰기</h2>
    <board-register-form @submit="onsubmit"/>
  </div>
</template>

<script>
import BoardRegisterForm from "@/components/BoardRegisterForm";
import axios from 'axios'
export default {
  name: "BoardRegisterPage",
  components: {
    BoardRegisterForm
  },
  methods: {
    onsubmit (payload) {
      console.log('BoardRegisterPage onSubmit()')
      const { title, content, writer } = payload
      axios.post('http://localhost:7777/boards',
          { title, content, writer})
      .then(res => {
        console.log(res)
        alert('등록이 잘 되었습니다.')
        this.$router.push({
          name: 'BoardReadPage',
          params: { boardNo: res.data.boardNo.toString() }
        })
      })
    }
  }
}
</script>

<style scoped>
</style>
```

axios: HTTP 통신을 하는데 사용하는
JS 라이브러리.
Promise 기반으로 asyac/await 사용하여 XHR 요청을 처리할 수 있다.

-axios.get/.post 로 method 불러와옴.

버튼클릭

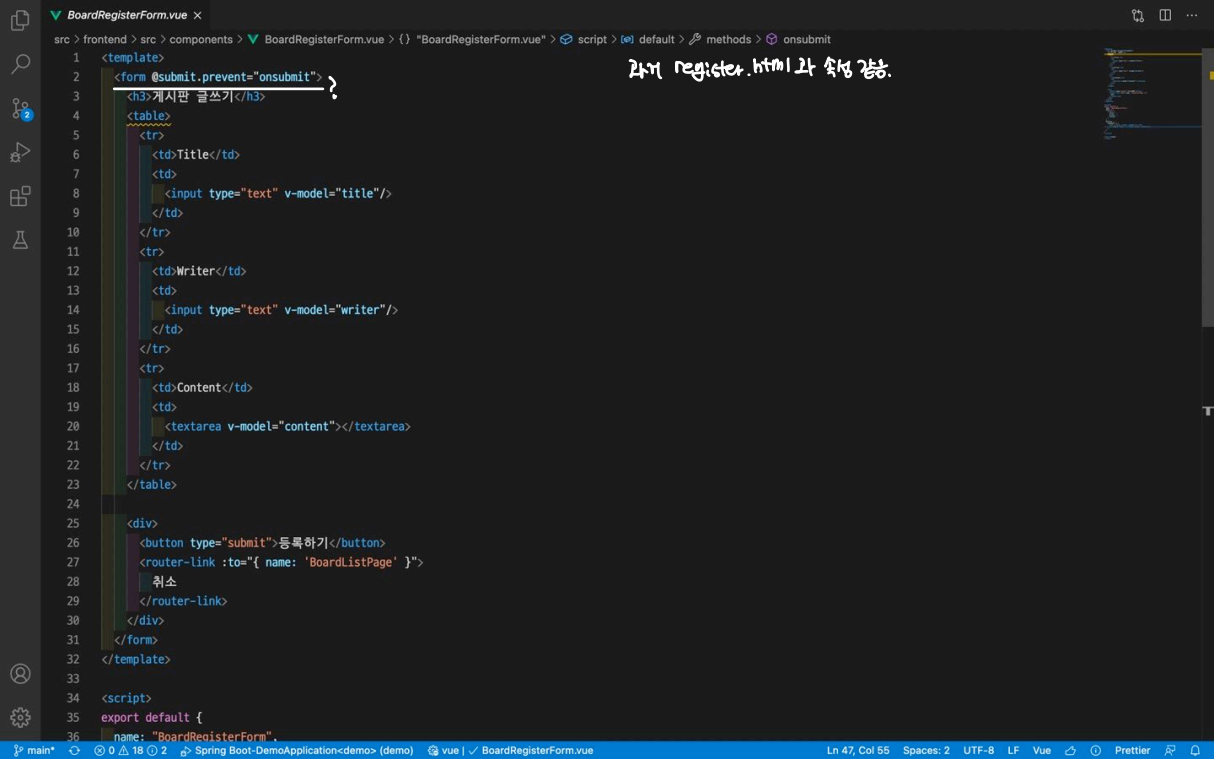
상태값 변화.

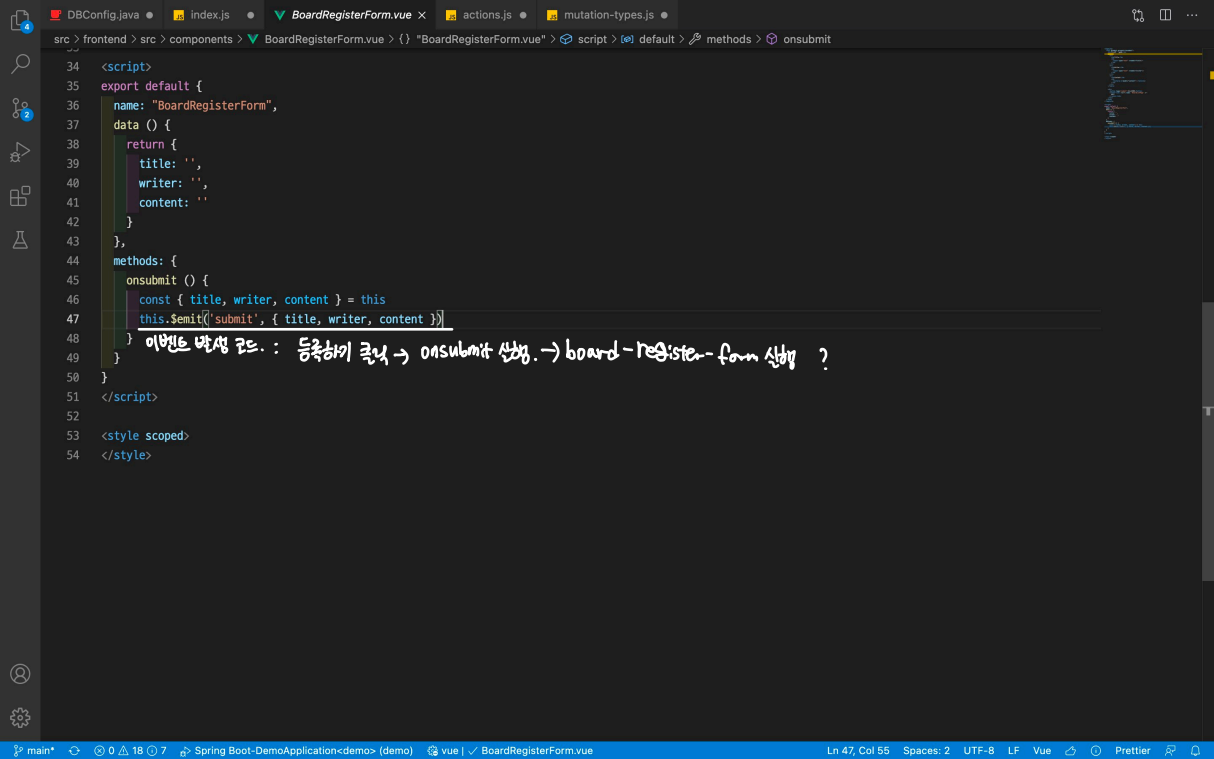
이벤트로 받은 데이터 내용을

axios의 post method 그리고 Promise 처리.

?

?


BoardRegisterForm.vue
src > frontend > src > components > BoardRegisterForm.vue > { } "BoardRegisterForm.vue" > script > default > methods > onsubmit
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 
26 <button type="submit">등록하기</button>
27 <router-link :to="{ name: 'BoardListPage' }">
28 취소
29 </router-link>
30 
31 </form>
32 </template>
33
34 <script>
35 export default {
36 name: "BoardRegisterForm",
과거 register.html과 유사함.
main* 0 18 2 Spring Boot-DemoApplication<demo> (demo) vue | BoardRegisterForm.vue
Ln 47, Col 55 Spaces: 2 UTF-8 LF Vue Prettier


DBConfig.java
index.js
BoardRegisterForm.vue
actions.js
mutation-types.js
src > frontend > src > components > BoardRegisterForm.vue > { } "BoardRegisterForm.vue" > script > default > methods > onsubmit
<script>
export default {
name: "BoardRegisterForm",
data () {
return {
title: '',
writer: '',
content: ''
}
},
methods: {
onsubmit () {
const { title, writer, content } = this
this.$emit('submit', { title, writer, content })
}
}
}
</script>
<style scoped>
</style>
이벤트 버블 링 : 등록하기 클릭 -> onsubmit 실행 -> board-register-form 실행 ?
main*
0 18 7
Spring Boot-DemoApplication<demo> (demo)
vue | BoardRegisterForm.vue
Ln 47, Col 55
Spaces: 2
UTF-8
LF
Vue
Prettier