KB금융그룹 | 국민의 평생 금융파트너

2024년 상반기 K-디지털 트레이닝

pinia

[KB] IT's Your Life

KB 국민은행

커리큘럼

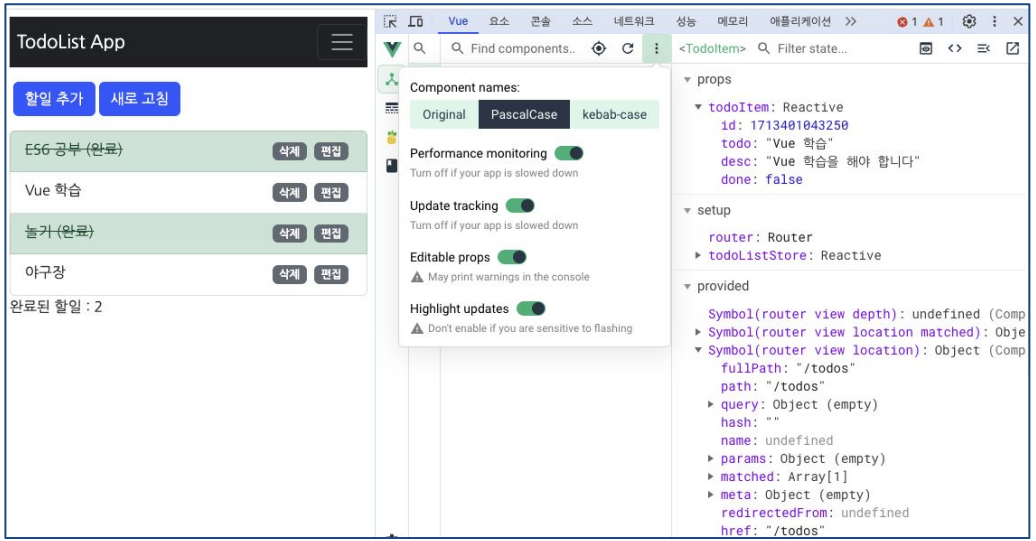
오늘의 목표

TodoList 테스트(Option API)

할일 추가 : 추가

- ES6학습 삭제
- React학습 삭제
- ContextAPI 학습 (완료) 삭제
- 야구경기 관람 삭제
- 할일없음 삭제
- 잘되네! 삭제

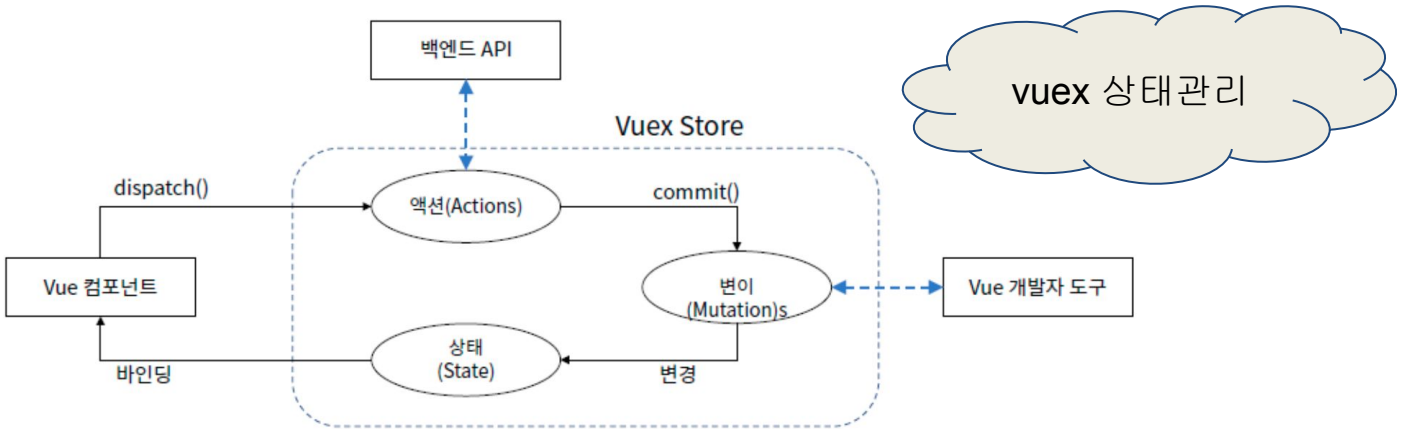
완료된 할일 수 : 1



pinia

pinia

- 상태관리를 중앙에서 가능하도록 하는 라이브러리
- **Composition API방식, Option API방식 모두 지원(Vuex는 Option API만 지원)**
- 기본 **Vuex**의 새로운 버전, 권장 표준
- **Vuex**보다 더 간단한 구조
- 라이브러리 크기가 **vuex**보다 경량
- **store**라는 중앙 집중 장소를 만들어 액션, 상태를 관리



pinia

- 상태 전달 방법
- 부모 <-> 자식
  - props/emit
  - provide/inject
  - pinia

props/emit

component
transfer

provide/inject

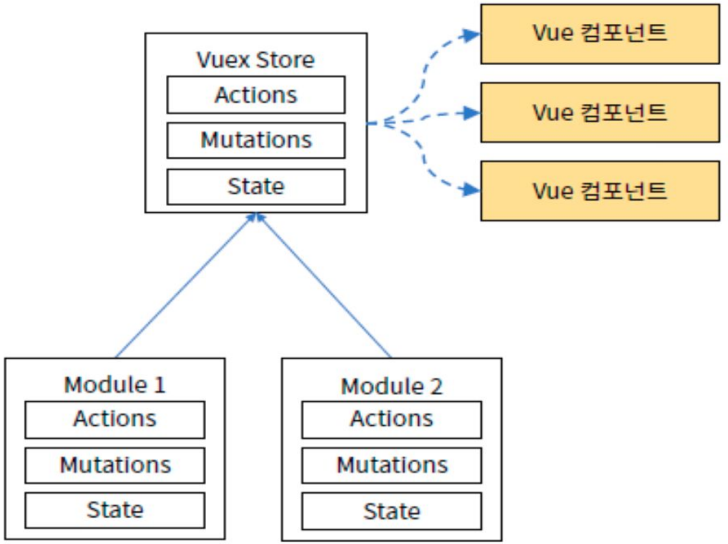
register anywhere

pinia

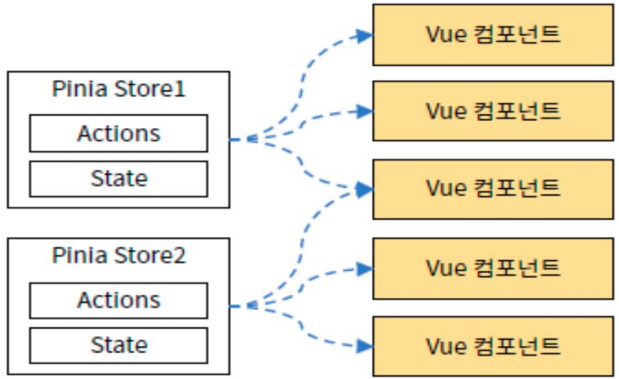
central
management

pinia

- **vuex**는 하나의 스토어를 사용했기 때문에 구조가 복잡
- **pinia**는 다중 스토어 사용 가능



Vuex store

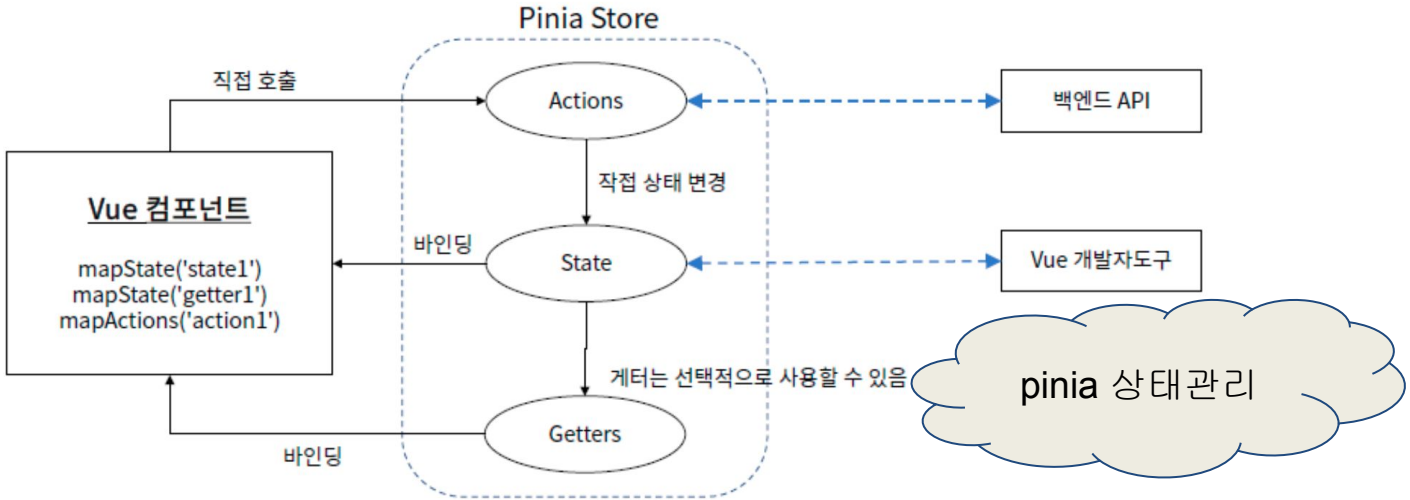


Pinia store

pinia 아키텍처

pinia 아키텍처

- **pinia 아키텍처**
  - store, actions, state, getters 등을 포함.
  - getters는 생략 가능
  - Option API - ...mapState(), ...mapActions() 사용
  - Composition API - use prefix붙는 사용자 정의함수 사용



pinia 아키텍쳐

- 중앙 집중식 **store정의**
  - defineStore()

```
import { defineStore } from "pinia";
import { reactive, computed } from 'vue';

// 옵션 API 방식의 todoList1 스토어
export const useCount1Store = defineStore('count1', {
    state: () => ({
        count: 0
    }),
    actions: {
        increment({num}){
            this.count += num;
        }
    }
})
```

```
// 컴포지션 API 방식의 todoList2 스토어
export const useCount2Store = defineStore("count2", ()=> {
    const state = reactive({ count : 0 })
    const increment = ({num}) => {
        state.count += num;
    }

    const count = computed(() => state.count)

    return { count, increment };
})
```

store/counter.js

pinia 아키텍쳐

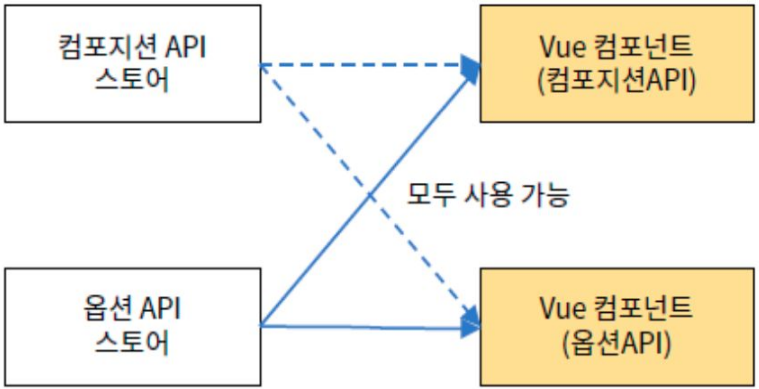
- **main.js import**

main.js

```
import { createApp } from 'vue'
import { createPinia } from 'pinia'
import App from './App.vue'


const pinia = createPinia()

const app = createApp(App)

app.use(pinia)
app.mount('#app')
```

pinia 아키텍쳐

- **Option API pinia defineStore 사용**
- **Option API + Composition API 서로 간 모두 사용 가능**



Option API

```
<script>
import { useCount1Store } from '@/stores/counter.js'
import { mapState, mapActions } from 'pinia';

export default {
  name:"App",
  computed : {
    ...mapState(useCount1Store, ['count'])
  },
  methods : {
    ...mapActions(useCount1Store, ['increment'])
  }
}
</script>
```

computed → store state

methods → store action

pinia 아키텍쳐

```
<script>
import { useCount1Store } from '@/stores/counter.j
import { ref, computed } from 'vue';

export default {
  setup() {
    const store = useCount1Store();
    const count = computed(() => store.count);
    const increment = store.increment;

    return { count, increment };
  }
}
</script>
```

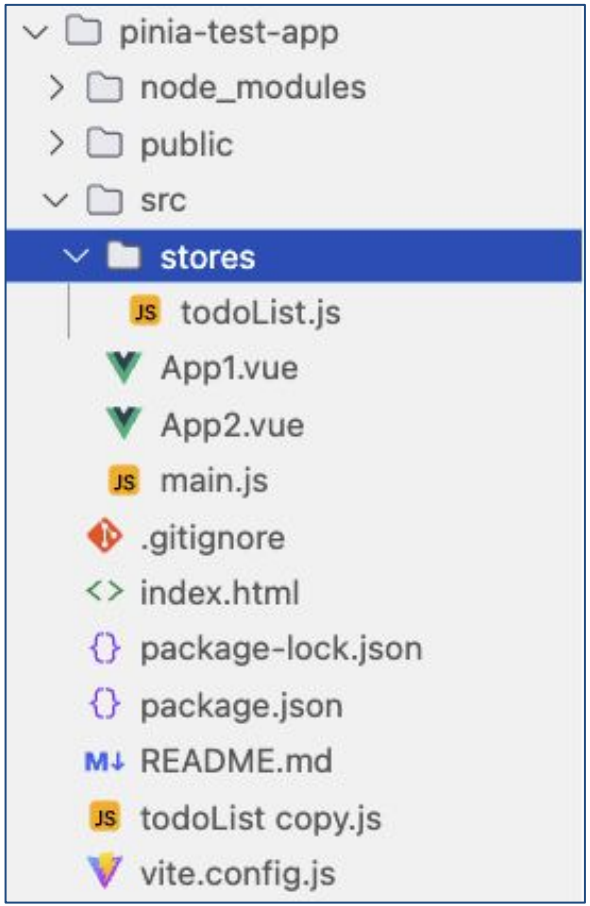
Option API

computed → store state

access operator

pinia 아키텍처


```
administrator@MacBook-Pro kb_ws_front % npm init vue pinia-app

Vue.js - The Progressive JavaScript Framework

✔ Add TypeScript? … No / Yes
✔ Add JSX Support? … No / Yes
✔ Add Vue Router for Single Page Application development? … No / Yes
✔ Add Pinia for state management? … No / Yes
✔ Add Vitest for Unit Testing? … No / Yes
✔ Add an End-to-End Testing Solution? › No
✔ Add ESLint for code quality? … No / Yes
✔ Add Vue DevTools 7 extension for debugging? (experimental) … No / Yes

Scaffolding project in /Users/administrator/Documents/kb_ws_front/pinia-app...

Done. Now run:

  cd pinia-app
  npm install
  npm run dev

administrator@MacBook-Pro kb_ws_front % cd pinia-app
administrator@MacBook-Pro pinia-app % npm install

added 27 packages, and audited 28 packages in 11s

4 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
administrator@MacBook-Pro pinia-app % npm i pinia
```



pinia 아키텍쳐

```
// 옵션 API 방식의 todoList1 스토어
export const useTodoList1Store = defineStore("todoList1", {
    state : () => ({
        todoList : [
            { id: 1, todo: "ES6학습",  done: false },
            { id: 2, todo: "React학습",  done: false },
            { id: 3, todo: "ContextAPI 학습",  done: true },
            { id: 4, todo: "야구경기 관람", done: false },
        ]
    }),
    getters : {
        doneCount : (state)=> {
            return state.todoList.filter((todoItem)=>todoItem.done === true).length;
        }
    },
    actions : {
        addTodo(todo) {
            this.todoList.push({ id: new Date().getTime(), todo, done: false })
        },
        deleteTodo(id) {
            let index = this.todoList.findIndex((todo) => todo.id === id);
            this.todoList.splice(index, 1);
        },
        toggleDone(id) {
            let index = this.todoList.findIndex((todo) => todo.id === id);
            this.todoList[index].done = !this.todoList[index].done;
        }
    }
})
```

todoList.js

```
// 컴포지션 API 방식의 todoList2 스토어
export const useTodoList2Store = defineStore("todoList2", ()=> {
    const state = reactive({
        todoList : [
            { id: 1, todo: "ES6학습",  done: false },
            { id: 2, todo: "React학습",  done: false },
            { id: 3, todo: "ContextAPI 학습",  done: true },
            { id: 4, todo: "야구경기 관람", done: false },
        ]
    })

    const addTodo = (todo) => {
        state.todoList.push({ id: new Date().getTime(), todo, done: false })
    }

    const deleteTodo = (id) => {
        let index = state.todoList.findIndex((todo) => todo.id === id);
        state.todoList.splice(index, 1);
    }

    const toggleDone =  (id) => {
        let index = state.todoList.findIndex((todo) => todo.id === id);
        state.todoList[index].done = !state.todoList[index].done;
    }

    const doneCount = computed(()=> {
        return state.todoList.filter((todoItem)=>todoItem.done === true).length;
    })

    const todoList = computed(()=> state.todoList);

    return { todoList, doneCount, addTodo, deleteTodo, toggleDone };
})
```

pinia 아키텍쳐

```vue
<template>
  <div>
    <h2>TodoList 테스트(Option API)</h2>
    <hr />
    할일 추가 :
    <input type="text" v-model="todo" />
    <button @click="addTodoHandler">추가</button>
    <hr />
    <ul>
      <li v-for="todoItem in todoList">
        <span style="cursor:pointer" @click="toggleDone(todoItem.id)">
        {{ todoItem.todo}} {{ todoItem.done ? "(완료)" : "" }}
        </span>
           
        <button @click="deleteTodo(todoItem.id)">삭제</button>
      </li>
    </ul>
    <div>완료된 할일 수 : {{doneCount}}</div>
  </div>
</template>
```

App1.vue

```vue
<script>
import { useTodoList1Store } from '@/stores/todoList.js'
import { mapState, mapActions } from 'pinia';

export default {
  name:"App1",
  data : ()⇒({ todo:"" }),
  computed : {
    ...mapState(useTodoList1Store, ['todoList', 'doneCount' ])
  },
  methods : {
    ...mapActions(useTodoList1Store,  ['addTodo', 'deleteTodo', 'toggleDone']),
    addTodoHandler() {
        this.addTodo(this.todo);
        this.todo = "";
    }
  }
}
</script>
```

pinia 아키텍쳐

```vue
<template>
  <div>
    <h2>TodoList 테스트(Composition API)</h2>
    <hr />
    할일 추가 :
    <input type="text" v-model="todo" />
    <button @click="addTodoHandler">추가</button>
    <hr />
    <ul>
      <li v-for="todoItem in todoList">
        <span style="cursor:pointer" @click="toggleDone(todoItem.id)">
        {{ todoItem.todo}} {{ todoItem.done ? "(완료)" : "" }}
        </span>
           
        <button @click="deleteTodo(todoItem.id)">삭제</button>
      </li>
    </ul>
    <div>완료된 할일 수 : {{doneCount}}</div>
  </div>
</template>
```

App2.vue

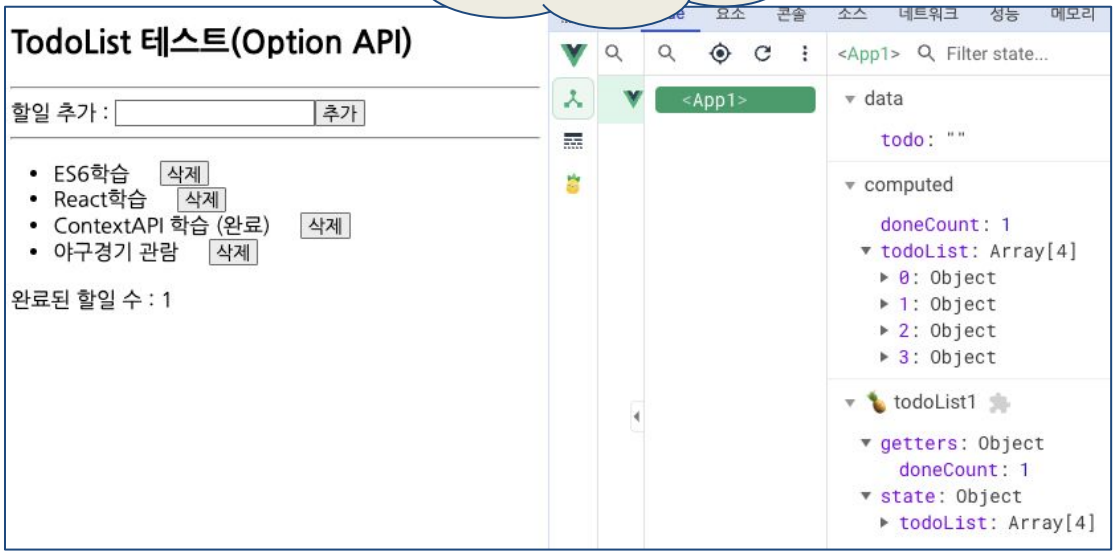
```vue
<script>
import { useTodoList2Store } from '@/stores/todoList.js'
import { ref, computed } from 'vue';

export default {
  name:"App2",
  setup() {
    const todo = ref("");
    const todoListStore = useTodoList2Store();
    const { todoList, addTodo, deleteTodo, toggleDone } = todoListStore;
    const doneCount = computed(()=>todoListStore.doneCount);

    const addTodoHandler = () => {
      addTodo(todo.value);
      todo.value = "";
    }
    return { todo, todoList, doneCount, addTodoHandler, deleteTodo, toggleDone }
  }
}
</script>
```

pinia 아키텍쳐

main.js

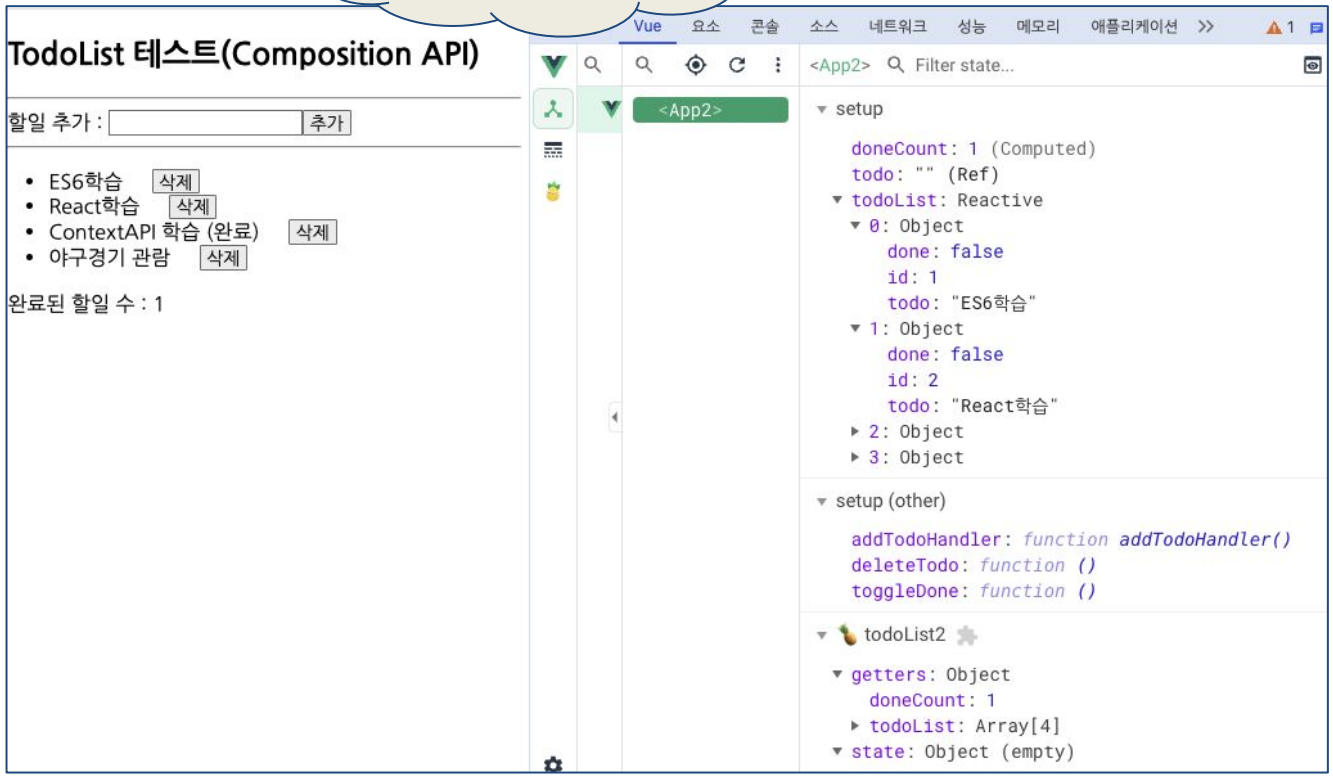
```
import { createApp } fr
import { createPinia } from 'pinia'
import App from './App1.vue'
//import App from './App2.vue'

const pinia = createPinia()

const app = createApp(App)

app.use(pinia)
app.mount('#app')
```

App1.vue import



pinia 아키텍쳐

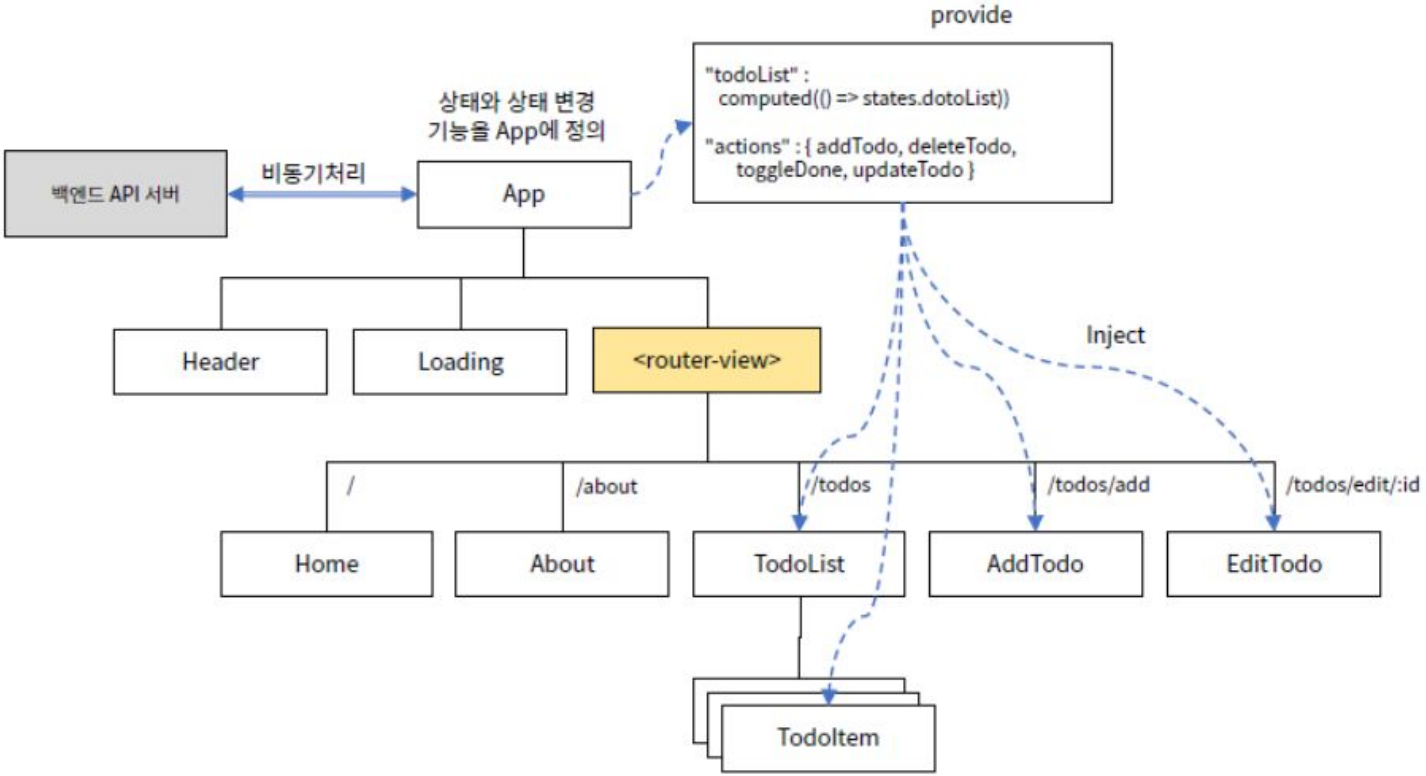
App2.vue import



pinia 적용 예제

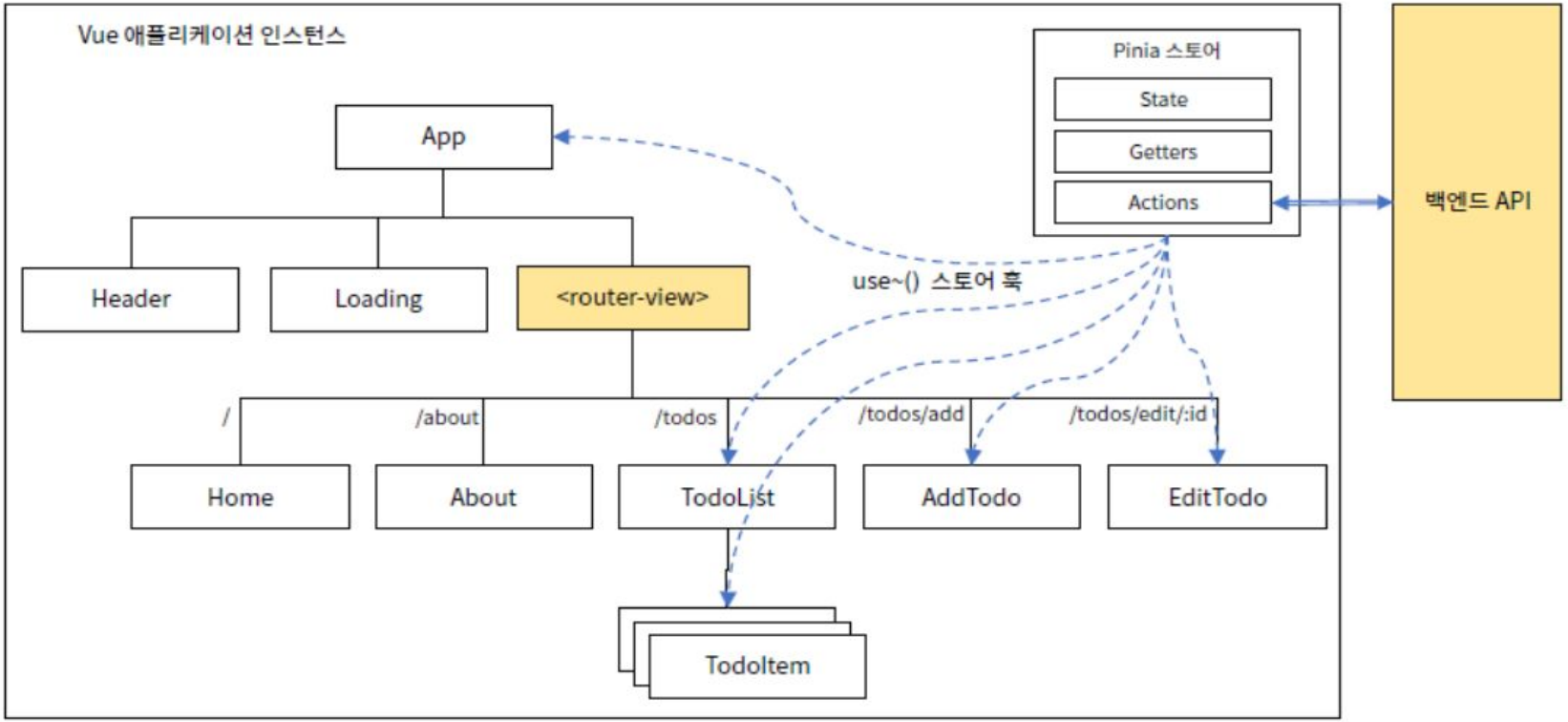
pinia 적용 예제

- 기존 예제 사용



pinia 적용 예제

- **pinia 적용 예제**
- **npm install –save pinia**



pinia 적용 예제

main.js

```
import { createApp } from 'vue'
import App from './App.vue'
import 'bootstrap/dist/css/bootstrap.css'
import router from './router/index.js'
import './main.css';
import { createPinia } from 'pinia';

const pinia = createPinia();
const app = createApp(App);
app.use(pinia);
app.use(router);
app.mount('#app')
```

pinia 적용 예제

```
import { defineStore } from "pinia";
import axios from 'axios';

//1. BASEURI는 vite.config.js의 프록시 설정에 맞추어 지정합니다.
//2. owner는 샘플 백엔드 API 서버(https://sample.bmaster.kro.kr)의
//   문서를 참조하여 지정합니다. 기본값 gdhong 데이터는 존재합니다.
const owner = "gdhong";
//의도적 1초의 지연시간을 발생시키는 API 사용
const BASEURI = "/api/todolist_long";

//todoList1 스토어 정의
export const useTodoListStore = defineStore("todoList1", {
    //상태 정의(todoList, isLoading)
    //isLoading : Loading 컴포넌트를 보여줄지 여부 결정을 위한 상태
    state : () => {
        return {
            todoList : [],
            isLoading: false
        }
    },
    //읽기 전용의 게터
    //doneCount : 완료된 할일의 건수를 읽기 전용으로 제공
    getters : {
        doneCount : (state)=> {
            const filtered = state.todoList.filter((todoItem)=>todoItem.done ===
            return filtered.length;
        }
    },
```

todoList.js

```
//액션
//비동기처리 시작에서 isLoading=true, 비동기 처리 완료 후
actions : {
    async fetchTodoList() {
        this.isLoading = true;
        try {
            const response = await axios.get(BASEURI + `/${owner}`);
            if (response.status === 200) {
                this.todoList = response.data;
            } else {
                alert('데이터 조회 실패');
            }
            this.isLoading = false;
        } catch(error) {
            alert('에러발생 :' + error);
            this.isLoading = false;
        }
    },
    async addTodo({ todo, desc }, successCallback) {
        if (!todo || todo.trim()==="") {
            alert('할일은 반드시 입력해야 합니다');
            return;
        }
        this.isLoading = true;
        try {
            const payload = { todo, desc };
            const response = await axios.post(BASEURI + `/${owner}`, payload)
            if (response.data.status === "success") {
                this.todoList.push({ id: response.data.item.id, todo, desc, done: false })
                successCallback();
              } else {
                alert('Todo 추가 실패 : ' + response.data.message);
              }
            this.isLoading = false;
```

pinia 적용 예제

```
async updateTodo({ id, todo, desc, done }, successCallback) {
        alert('할일은 반드시 입력해야 합니다');
        return;
    }
    this.isLoading = true;
    try {
        const payload = { todo, desc, done };
        const response = await axios.put(BASEURI + `/${owner}/${id}`, payload)
        if (response.data.status === "success") {
            let index = this.todoList.findIndex((todo) => todo.id === id);
            this.todoList[index] = { id, todo, desc, done };
            successCallback();
          } else {
            alert('Todo 변경 실패 : ' + response.data.message);
          }
        this.isLoading = false;
    } catch(error) {
        alert('에러발생 :' + error);
        this.isLoading = false;
    }
},
async deleteTodo(id) {
    this.isLoading = true;
    try {
        const response = await axios.delete(BASEURI + `/${owner}/${id}`)
        if (response.data.status === "success") {
            let index = this.todoList.findIndex((todo) => todo.id === id);
            this.todoList.splice(index, 1);
          } else {
            alert('Todo 삭제 실패 : ' + response.data.message);
          }
        this.isLoading = false;
    } catch(error) {
        alert('에러발생 :' + error);
```

todoList.js

App.vue

```
<template>
  <div class="container">
      <Header />
      <router-view />
      <Loading v-if="isLoading" />
  </div>
</template>

<script setup>
import { computed } from 'vue';
import Header from '@/components/Header.vue'
import { useTodoListStore } from '@/stores/todoList.js'
import Loading from '@/components/Loading.vue'

const todoListStore = useTodoListStore();
const isLoading = computed(()=>todoListStore.isLoading);
const fetchTodoList = todoListStore.fetchTodoList;
fetchTodoList();
</script>
```

pinia 적용 예제

todoList.vue

```vue
<template>
    <div class="row">
        <div class="col p-3">
            <router-link class="btn btn-primary" to="/todos/add">
            할일 추가
            </router-link>
            <button class="btn btn-primary ms-1" @click="fetchTodoList">새로 고침</button>
        </div>
    </div>
    <div class="row">
        <div class="col">
            <ul class="list-group">
                <TodoItem v-for="todoItem in todoList" :key="todoItem.id" :todoItem="todoItem" />
            </ul>
        </div>
        <span>완료된 할일 : {{doneCount}}</span>
    </div>
</template>

<script setup>
import { computed } from 'vue';
import { useTodoListStore } from '@/stores/todoList.js'
import TodoItem from '@/pages/TodoItem.vue'

const todoListStore = useTodoListStore();
const { fetchTodoList } = todoListStore;
const doneCount = computed(()=>todoListStore.doneCount);
const todoList = computed(()=>todoListStore.todoList);
</script>
```

pinia 적용 예제

todoItem.vue

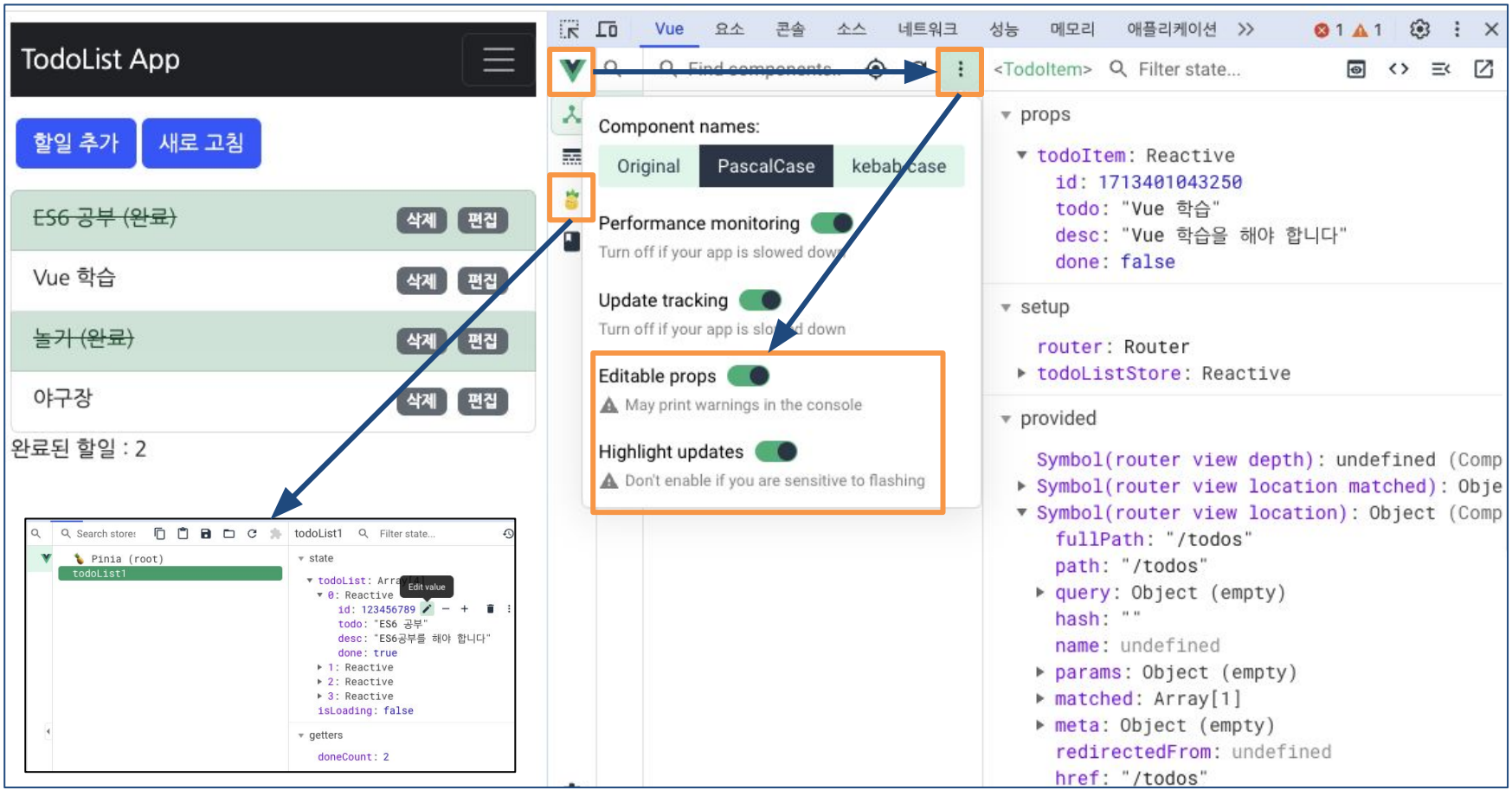
```vue
<template>
  <li :class="todoItem.done ? 'list-group-item list-group-item-success' : '
    <span :class="todoItem.done ? 'todo-done pointer' : 'pointer'"
      @click="toggleDone(todoItem.id)">
      {{todoItem.todo}}
      {{todoItem.done ? '(완료)' : '' }}
    </span>
    <span class="float-end badge bg-secondary pointer m-1"
      @click="router.push(`/todos/edit/${todoItem.id}`)">
      편집</span>
    <span class="float-end badge bg-secondary pointer m-1"
      @click="deleteTodo(todoItem.id)">
      삭제</span>
  </li>
</template>

<script setup>
import { useRouter } from 'vue-router';
import { useTodoListStore } from '@/stores/todoList.js

defineProps({
  todoItem: { Type: Object, required:true }
})

const router = useRouter();
const todoListStore = useTodoListStore();
const { deleteTodo, toggleDone } = todoListStore;
</script>
```

pinia 적용 예제


```vue
<script setup>
import { reactive } from 'vue';
import { useRouter } from 'vue-router';
import { useTodoListStore } from '@/stores/todoList.js'

const router = useRouter();
const { addTodo } = useTodoListStore();
const todoItem =  reactive({ todo:"", desc:"" })

const addTodoHandler = () => {
    let { todo } = todoItem;
    if (!todo || todo.trim()==="") {
        alert('할일은 반드시 입력해야 합니다');
        return;
    }
    addTodo({ ...todoItem }, ()=>{
        router.push('/todos')
    });
}
</script>
```

```vue
<script setup>
import { reactive } from 'vue';
import { useRouter, useRoute } from 'vue-router';
import { useTodoListStore } from '@/stores/todoList.js'

const router = useRouter();
const currentRoute = useRoute();

const { todoList, updateTodo, } = useTodoListStore();

const matchedTodoItem = todoList.find((item)=> item.id === parseInt(currentRoute.params.id))
if (!matchedTodoItem) {
    router.push('/todos');
}
const todoItem =  reactive({ ...matchedTodoItem })

const updateTodoHandler = () => {
    let { todo } = todoItem;
    if (!todo || todo.trim()==="") {
        alert('할일은 반드시 입력해야 합니다');
        return;
    }
    updateTodo({ ...todoItem }, ()=>{
        router.push('/todos');
    });
}
</script>
```

pinia 적용 예제


TodoList App
할일 추가
새로 고침
ES6 공부 (완료)
삭제
편집
Vue 학습
삭제
편집
놀기 (완료)
삭제
편집
야구장
삭제
편집
완료된 할일 : 2
Vue
요소
콘솔
소스
네트워크
성능
메모리
애플리케이션
Component names:
Original
PascalCase
kebab-case
Performance monitoring
Turn off if your app is slowed down
Update tracking
Editable props
May print warnings in the console
Highlight updates
Don't enable if you are sensitive to flashing
<TodoItem>
Filter state...
props
todoItem: Reactive
id: 1713401043250
todo: "Vue 학습"
desc: "Vue 학습을 해야 합니다"
done: false
setup
router: Router
todoListStore: Reactive
provided
Symbol(router view depth): undefined (Comp
Symbol(router view location matched): Obje
Symbol(router view location): Object (Comp
fullPath: "/todos"
path: "/todos"
query: Object (empty)
hash: ""
name: undefined
params: Object (empty)
matched: Array[1]
meta: Object (empty)
redirectedFrom: undefined
href: "/todos"
Search store
Pinia (root)
todoList1
todoList1
Filter state...
state
Edit value
id: 123456789
todo: "ES6 공부"
desc: "ES6공부를 해야 합니다"
done: true
isLoading: false
getters
doneCount: 2

핵심 정리

- **pinia 상태관리**
  - store에 저장된 액션, 상태를 중앙에서 관리 가능
- **pinia vs. vuex**
  - vuex보다 간단
  - Option API, Composition API 모두 사용 가능
- **pinia 아키텍처**
  - Actions, State, Getters
- **pinia 구성요소**
  - defineStore()
  - createPinia() 객체 생성
  - Option API - mapState, mapActions
  - Composition API - use prefix사용