KB금융그룹 | 국민의 평생 금융파트너

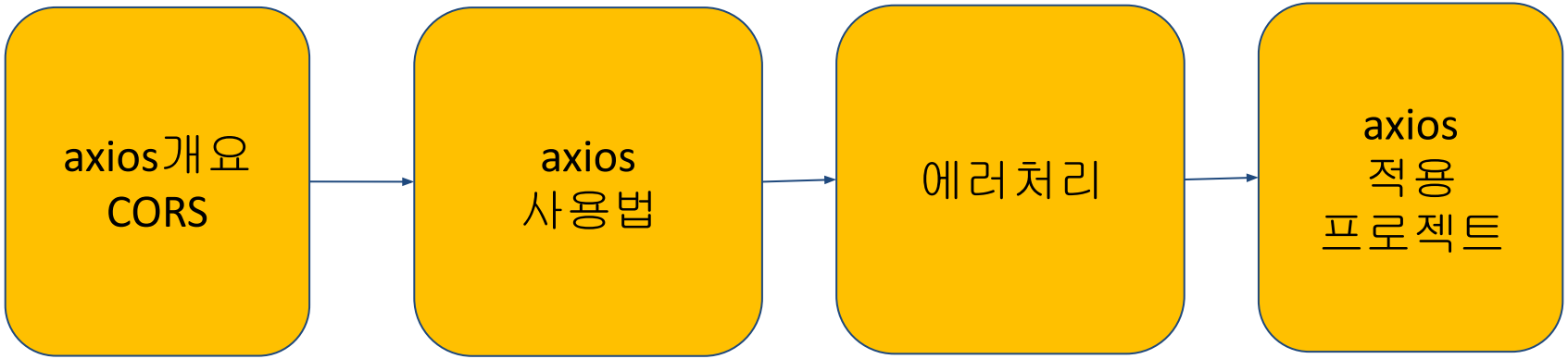
2024년 상반기 K-디지털 트레이닝

axios & router

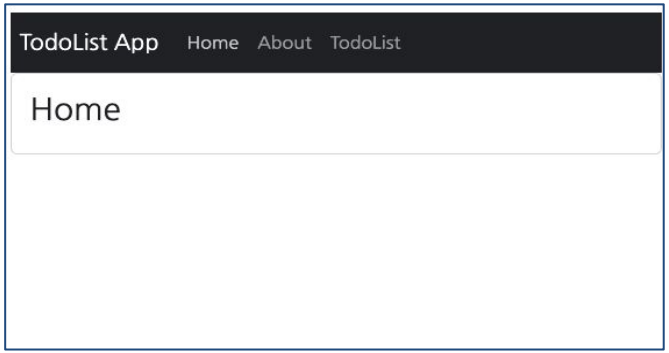
[KB] IT's Your Life

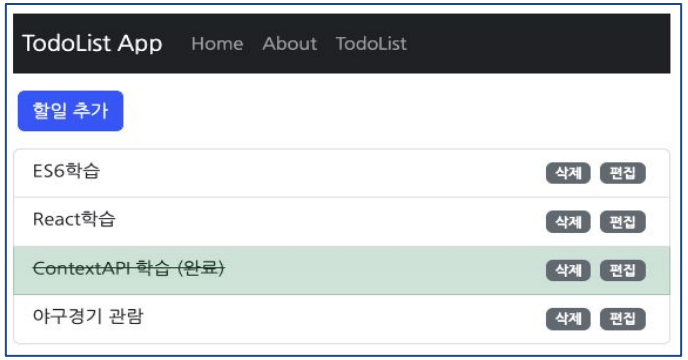
KB 국민은행

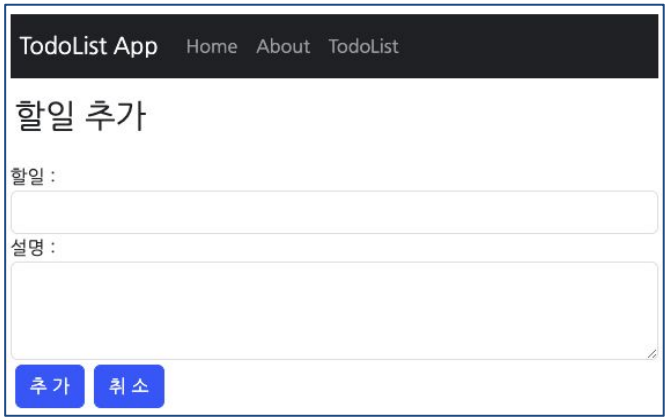
커리큘럼

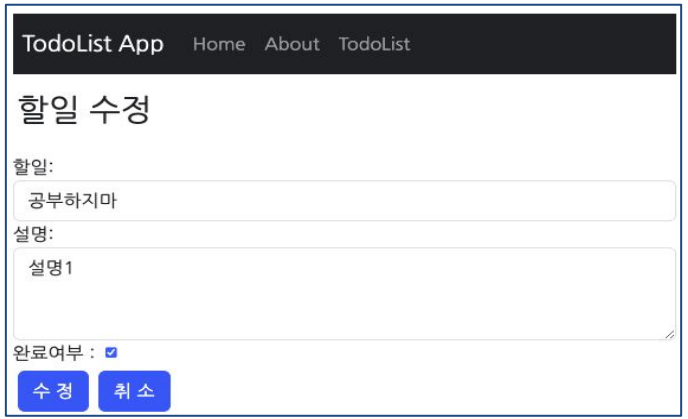


오늘의 목표


TodoList App
Home
About
TodoList
Home


TodoList App
Home
About
TodoList
할일 추가
ES6학습
삭제
편집
React학습
삭제
편집
ContextAPI 학습 (완료)
삭제
편집
야구경기 관람
삭제
편집


TodoList App
Home
About
TodoList
할일 추가
할일 :
설명 :
추 가
취 소


TodoList App
Home
About
TodoList
할일 수정
할일:
공부하지마
설명:
설명1
완료여부 :
수 정
취 소

axios

axios

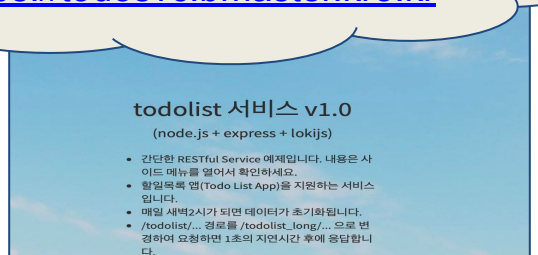
- **http통신을 지원하는 최근 가장 많이 사용되는 라이브러리**
  - fetch()와 동일한 역할, 최근에는 axios 더 많이 사용

구분	axios	fetch
모듈	설치 npm install –save axios	설치 불필요(JS에 내장)
Promise API	사용	사용
브라우저 호환성	뛰어남	IE지원X
timeout 기능	지원(timeout시간 후 통신 중단 가능)	지원X
JSON 자동 변환	지원(json을 바로 추출 가능)	지원X

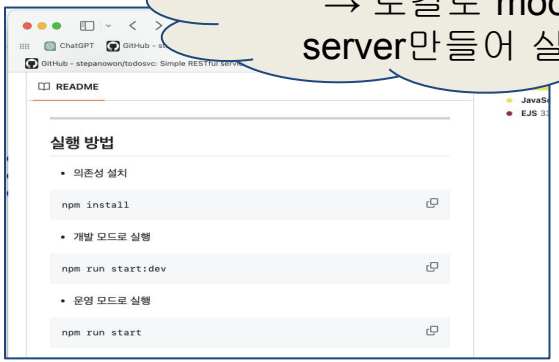
axios

- **http통신 서버 필요**
  - 우리는 추후에 웹서버로 Tomcat서버 사용 예정
  - 외부 JSON으로 응답하는 서버 사용 예정
  - 테스트를 위해 JSON을 응답받을 수 있는 임의의 서버 사용
    - json-server
    - mock-server
    - mocky.io
  - 우리는 json-server를 사용
    - 간단히 json파일로 서버 역할 가능

외부 mock server
https://todosvc.bmaster.kro.kr

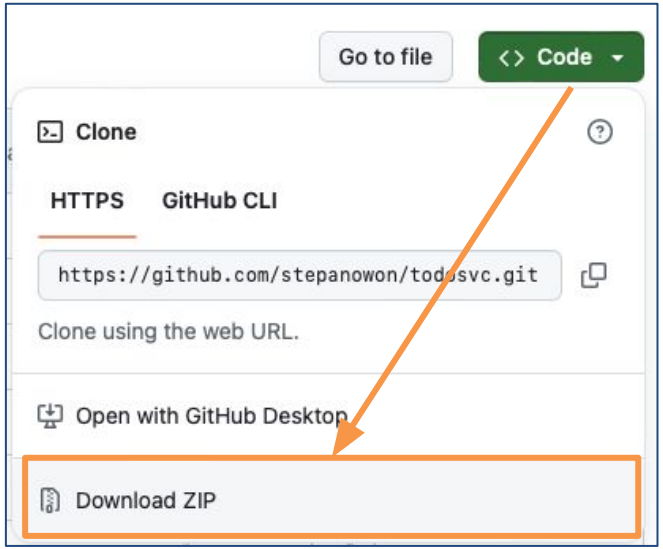


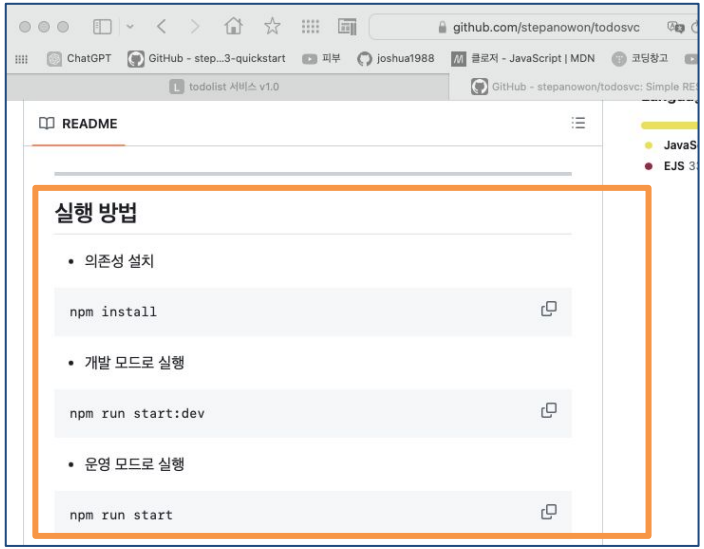
git → download
→ 로컬로 mock
server만들어 실행

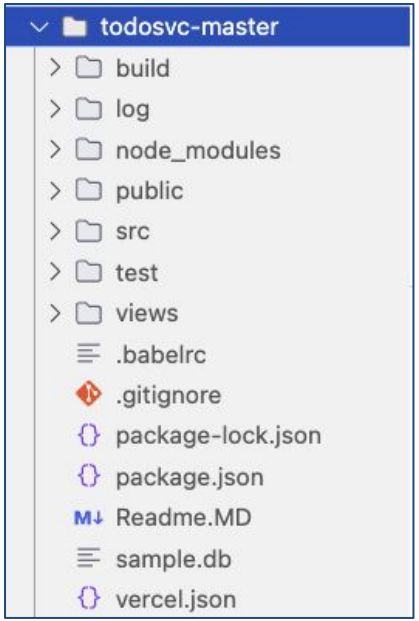


axios

- mock server만들기(github에서 다운로드)

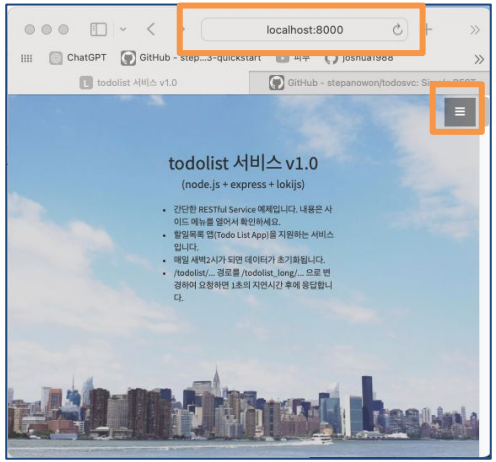


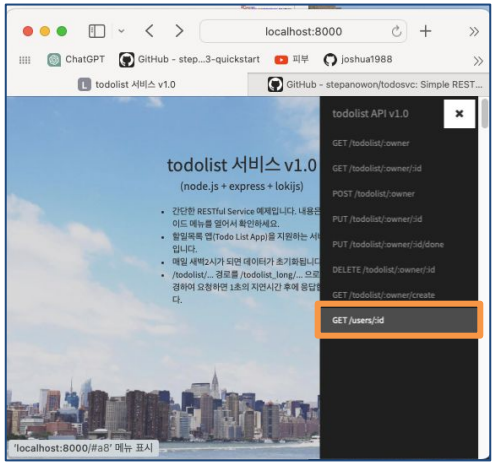


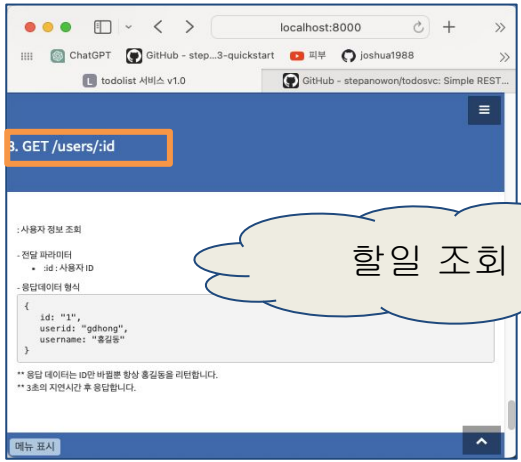



```
administrator@MacBook-Pro todosvc-master % npm install

administrator@MacBook-Pro todosvc-master % npm run start
```

axios


localhost:8000
todolist 서비스 v1.0
(node.js + express + lokijs)

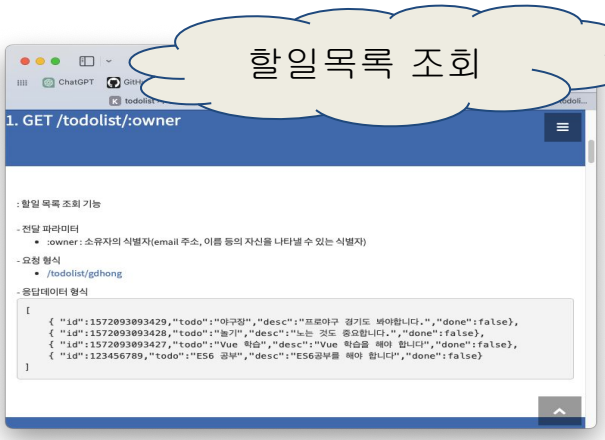

localhost:8000
todolist 서비스 v1.0
(node.js + express + lokijs)
todolist API v1.0
GET /todolist/:owner
GET /todolist/:owner/:id
POST /todolist/:owner
PUT /todolist/:owner/:id
PUT /todolist/:owner/:id/done
DELETE /todolist/:owner/:id
GET /todolist/:owner/create
GET /users/:id


localhost:8000
GET /users/:id
할일 조회


localhost:8000/users/1
ChatGPT
GitHub - step...3-quickstart
피부
joshua1988
클로저 - JavaScript | MDN
코딩창고
localhost:8000/users/1
GitHub - stepanowon/todosvc: Simple RESTful service for todol...
{"id":"1","userid":"gdhong","username":"홍길동"}

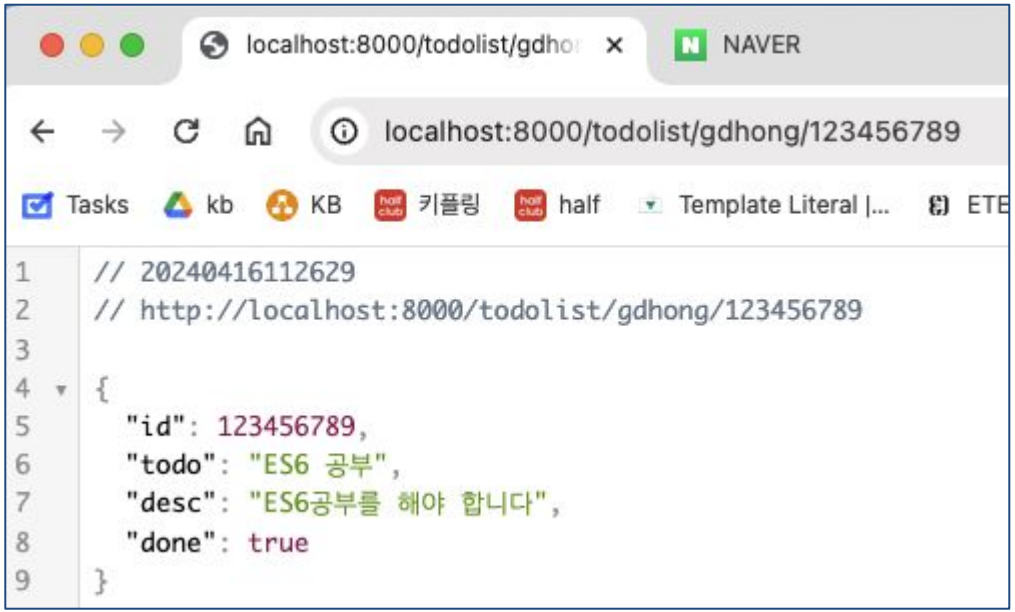
axios

- **http://localhost:8000의 주요 API**



axios

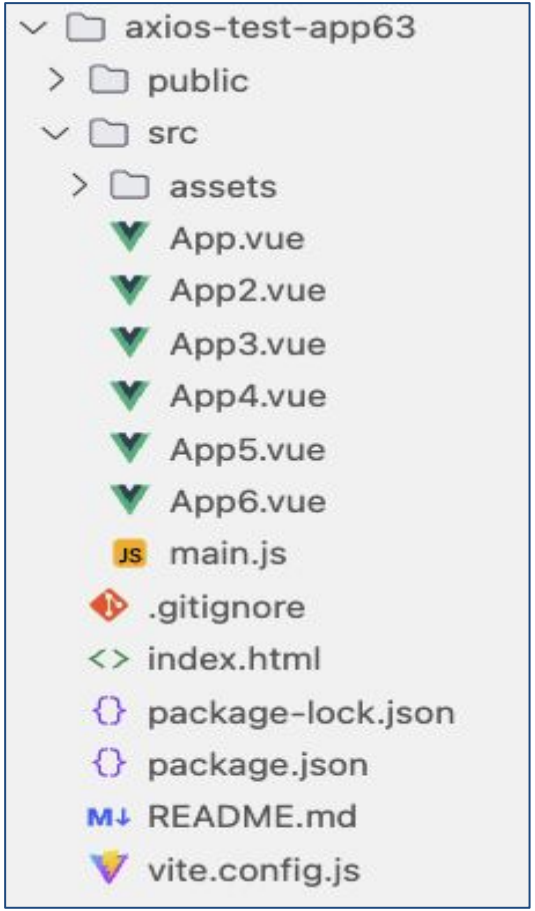
- 사용자 **gdhong**에 대한 데이터는 여러 개 들어있음.



axios

```
administrator@MacBook-Pro kb_ws_front % npm init vue axios-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/axios-app...

Done. Now run:

  cd axios-app
  npm install
  npm run dev
```

```
administrator@MacBook-Pro kb_ws_front % npm install axios

changed 2 packages, and audited 109 packages in 2s

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

found 0 vulnerabilities
```



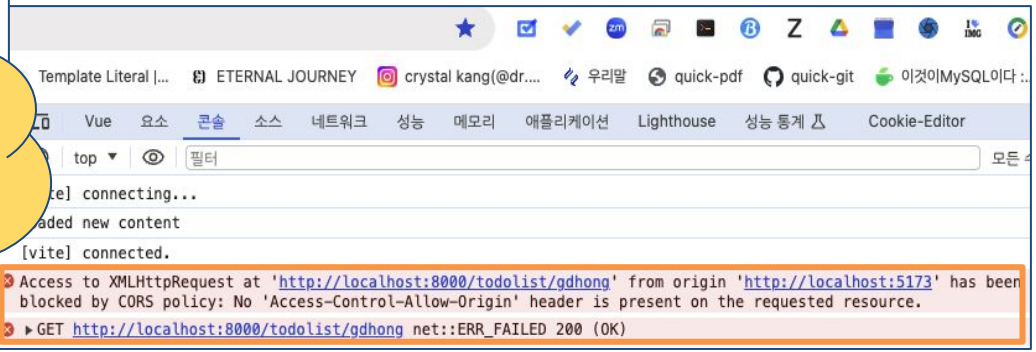
axios

App.vue

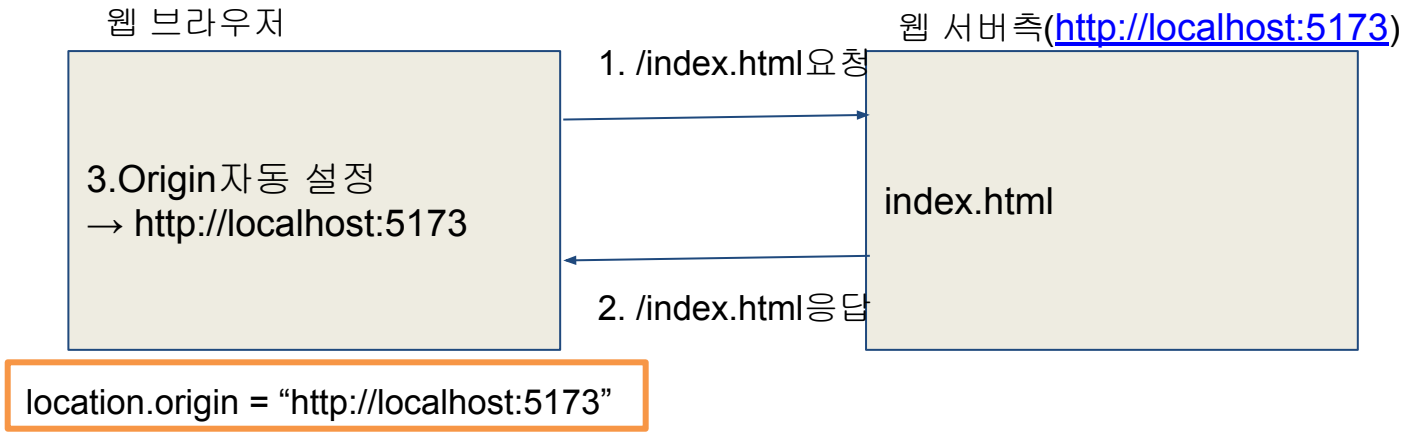
```vue
<template>
  <div>
    <h2>콘솔을 확인합니다.</h2>
  </div>
</template>

<script setup>
import axios from 'axios'

const requestAPI = () => {
    const url = "http://localhost:8000/todolist/gdhong";
    //const url = "/api/todolist/gdhong";
    axios.get(url).then((response) => {
        console.log("# 응답객체 : ", response);
    });
};

requestAPI();
</script>
```

mock server start후
실행해야함.
→ npm run dev

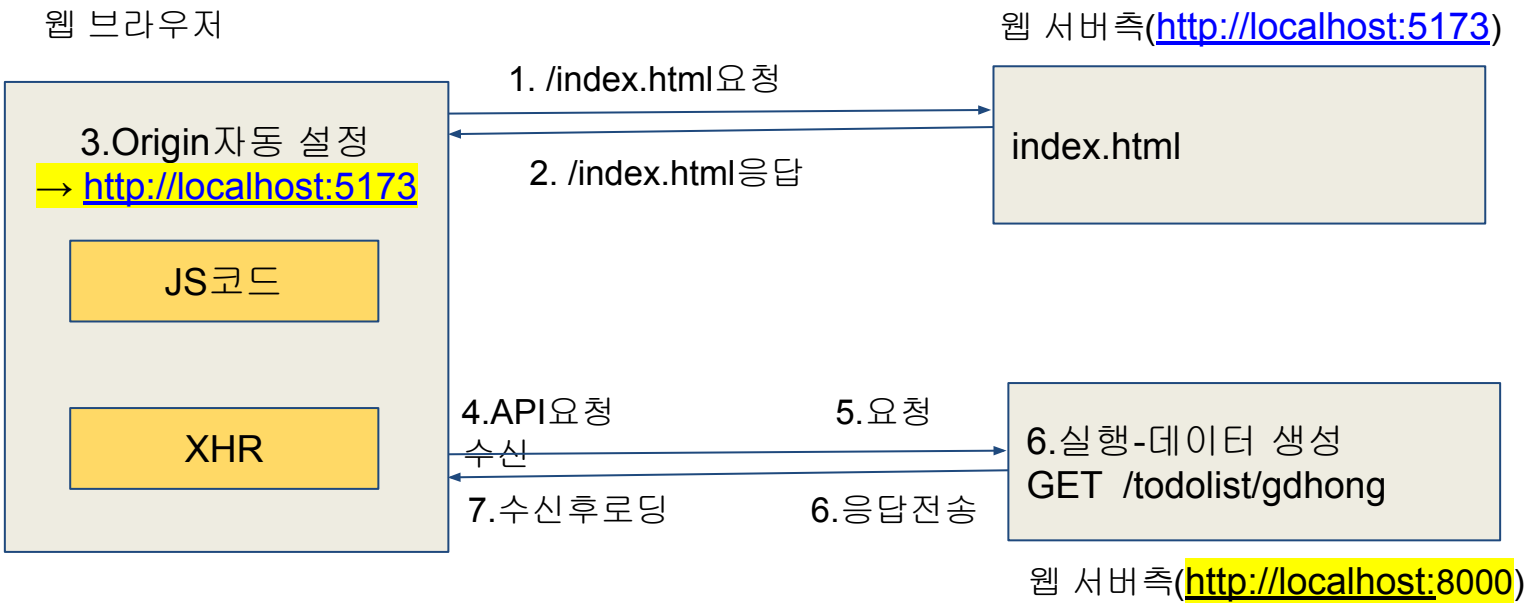
에러!
cross-origin
policy정책에 의해
block당함.



- **Cross Origin 문제 발생함**
  - 브라우저는 자신의 원래 주소와 다른 주소의 서버로 통신할 때 block시킬 수 있다.
  - 외부의 안전하지 않는 요청에 대한 관리 목적
  - Same Origin Policy(SOP)
- **브라우저가 인식하는 Same Origin**



- **SOP(Same Origin Policy)**
  - 브라우저의 보안 정책
  - 브라우저의 오리진 사이트와 동일한 오리진 값을 가진 서버일 때만 통신을 가능하게 한다.
  - 크로스 오리진에 대한 정책을 설정해야함.

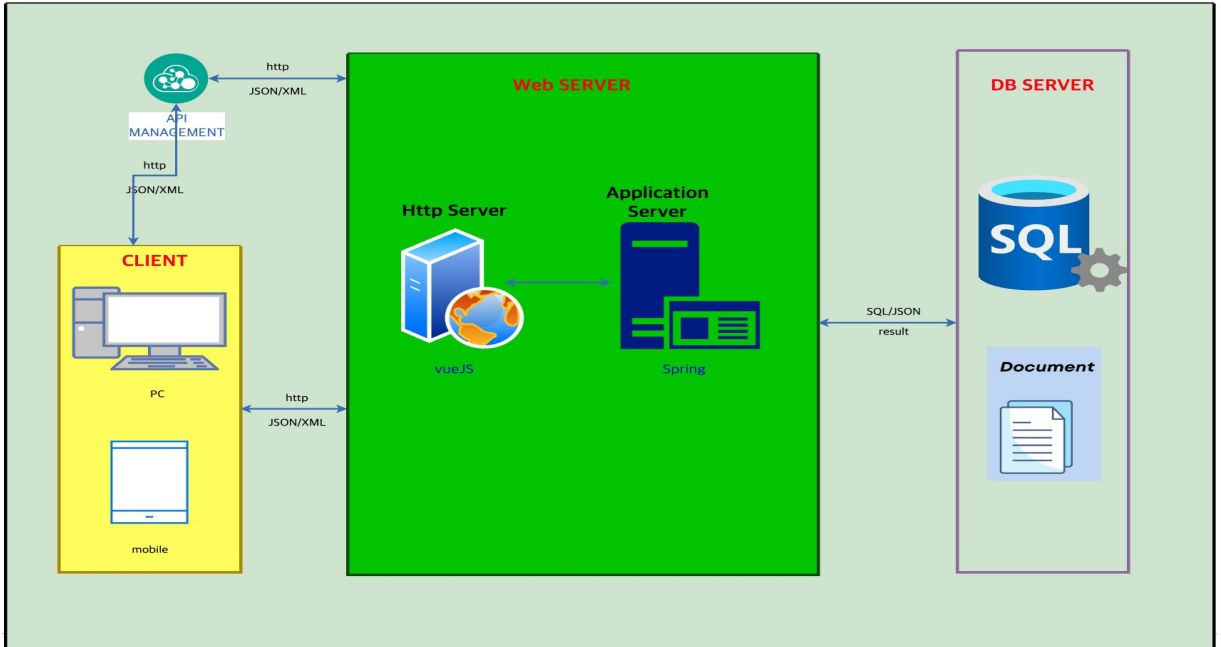


axios

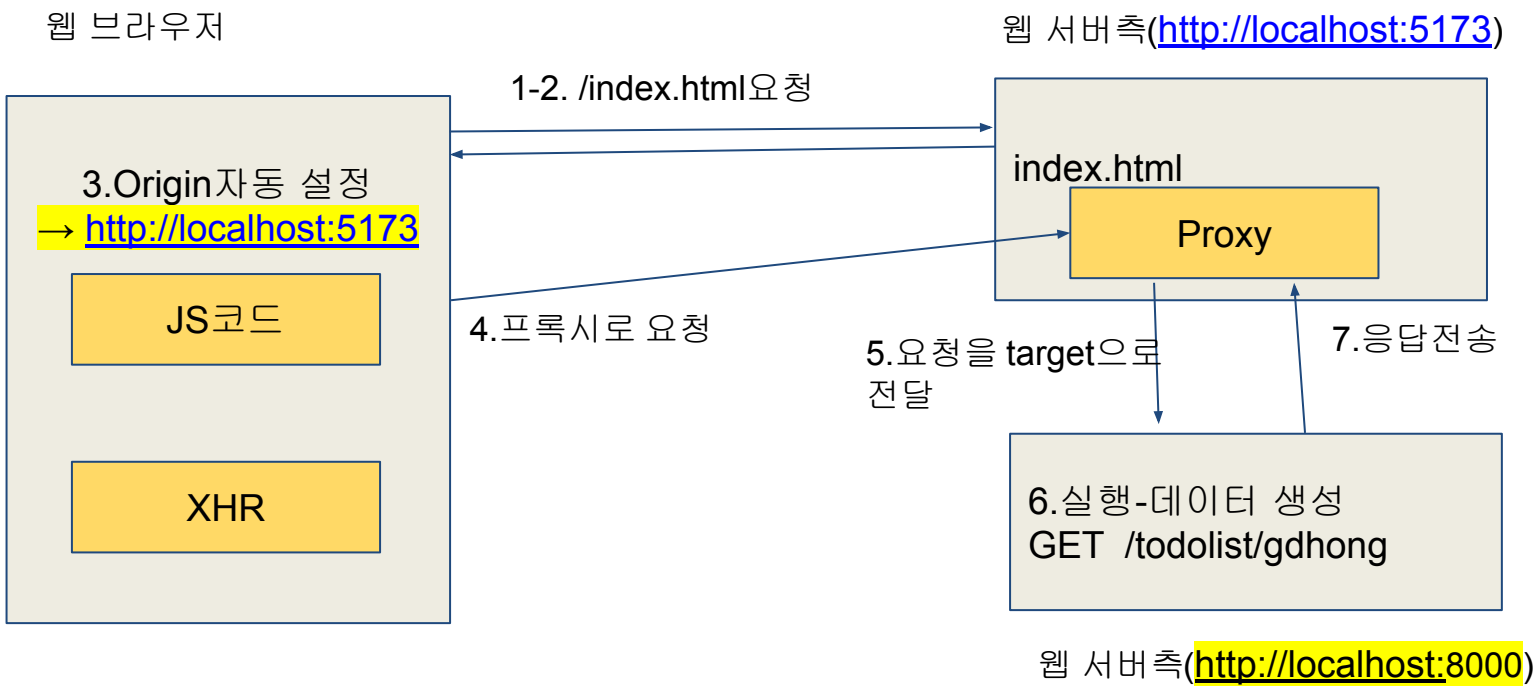
- **CORS요청 전체 흐름**
  - 브라우저는 서버에서 보낸 HTML을 받아와 자신의 오리진 속성을 설정
  - 브라우저가 JS코드로 서버로 요청시 자신의 오리진 속성값을 Origin Http헤더에 추가
  - 서버에 전송된 Origin Http헤더를 읽어내어 등록된 리스트에 일치하는지 확인
  - 서버는 Access-Control-Allow-Origin 응답 헤더를 추가하고 브라우저의 오리진을 값으로 지정하여 응답
  - 브라우저는 자신의 오리진과 서버로 부터 전송된 Access-Control-Allow-Origin헤더가 일치하면 허가된 것으로 간주하고 데이터를 로딩

axios

- 프록시(대리인) 서버를 통해 우회하는방법
- 브라우저가 프런트 서버의 프록시를 서버와 통신
  - 브라우저에서 동일 오리진으로 인식하게 함.
- 실제 배호푸 운영환경에는 프론트 엔드 서버와 백엔드 서버가 분리된 경우가 많음.



- 프록시를 사용한 경우


웹 브라우저
웹 서버측(http://localhost:5173)
1-2. /index.html요청
3.Origin자동 설정
→ http://localhost:5173
JS코드
XHR
index.html
Proxy
4.프록시로 요청
5.요청을 target으로
전달
7.응답전송
6.실행-데이터 생성
GET /todolist/gdhong
웹 서버측(http://localhost:8000)

axios

```
import { fileURLToPath, URL } from 'node:url'

import { defineConfig } from 'vite'
import vue from '@vitejs/plugin-vue'

// https://vitejs.dev/config/
export default defineConfig({
  plugins: [vue()],
  resolve: {
    alias: {
      '@': fileURLToPath(new URL('./src', import.meta.url))
    }
  },
  server: {
    proxy: {
      "/api": {
        target: "http://localhost:8000",
        changeOrigin: true,
        rewrite: (path) => path.replace(/^\/api/, ""),
      },
    },
  },
})
```

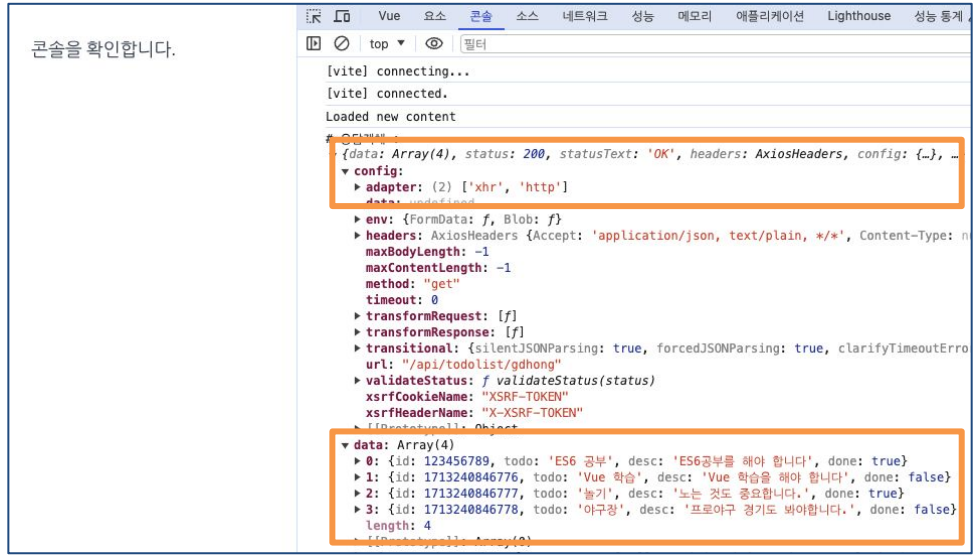
vite.config.js
프록시
설정하기

- 최초 요청 경로: /api/todolist/gdhong
- 타깃: http://localhost:8000
- 최종 전달 경로:
  http://localhost:8000/todolist/gdhong

axios

```vue
<template>
  <div>
    <h2>콘솔을 확인합니다.</h2>
  </div>
</template>

<script setup>
import axios from 'axios'

const requestAPI = () => {
    //const url = "http://localhost:8000/todolist/gdhong";
    const url = "/api/todolist/gdhong";
    axios.get(url).then((response) => {
        console.log("# 응답객체 : ", response);
    });
};

requestAPI();
</script>
```

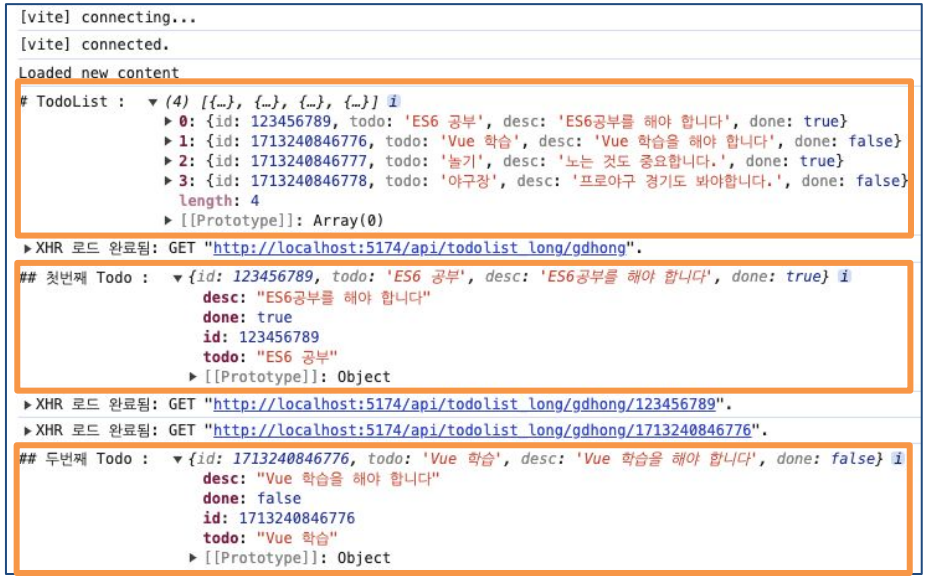
콘솔을 확인합니다.



axios 사용법

Promise, async~await

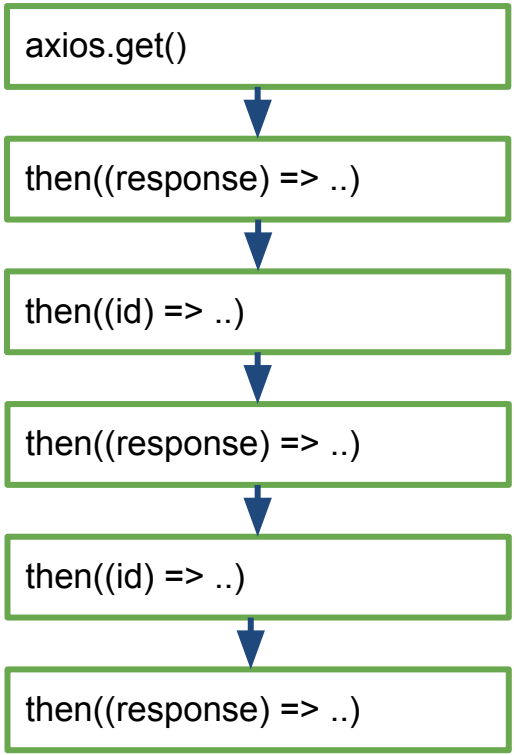
- **Promise**
  - 비동기를 순차적으로 실행되도록 하는 객체
  - then()을 함께 써서 비동기를 순차적으로 실행되도록 명시




```
<script setup>
const todoUrlPrefix = "/api/todolist_long/gdhong/";
//4건의 목록을 조회한 후 첫번째, 두번째 할일을 순차적으로 조회합니다.
const requestAPI = () => {
  let todoList = [];
  axios
    .get(listUrl)
    .then((response) => {
      todoList = response.data;
      console.log("# TodoList : ", todoList);
      return todoList[0].id;
    })
    .then((id) => {
      return axios.get(todoUrlPrefix + id);
    })
    .then((response) => {
      console.log("## 첫번째 Todo : ", response.data);
      return todoList[1].id;
    })
    .then((id) => {
      axios.get(todoUrlPrefix + id).then((response) => {
        console.log("## 두번째 Todo : ", response.data);
      });
    });
};

requestAPI();
</script>
```

Promise, async~await

- **then()에 명시된 순서대로 결과 처리 → 복잡**



TodoList요청

받아온 TodoList출력 후 첫번째 Todo의 id리턴

직전 리턴된 id를 인자로 받아 axios.get()요청 후 Promise리턴

응답된 첫번째 Todo출력하고 두번째 Todo의 id리턴

직전 리턴된 id를 인자로 받아 axios.get()요청 후 Promise리턴

응답된 두번째 Todo출력하고 종료

Promise, async~await

- **async와 await를 쓰면 좀더 간단하게 비동기 통신을 할 수 있음.**
  - 비동기로 호출하는 함수 앞부분에 async 추가
  - 함수 내부에서 Promise리턴하는 함수 호출 부분 앞에 await부여
  - await는 응답할 때까지 기다리고 있다가 처리된 다음에 리턴

```
//전체 목록을 조회한 후 한 건씩 순차적으로 순회하며 조회하기
const requestAPI = async () => {
  let todoList;

  let response = await axios.get(listUrl);
  todoList = response.data;
  console.log("# TodoList : ", todoList);
  for (let i = 0; i < todoList.length; i++) {
    response = await axios.get(todoUrlPrefix + todoList[i].id);
    console.log(`# ${i + 1}번째 Todo : `, response.data);
  }
};

requestAPI();
```

Promise, async~await

- 전체데이터를 순회하면서 순서대로 비동기 처리할 때

```
//전체 목록을 조회한 후 한 건씩 순차적으로 순회하며 조회하기 + 예외처리
const requestAPI = async () => {
  let todoList;

  try {
    let response = await axios.get(listUrl);
    todoList = response.data;
    console.log("# TodoList : ", todoList);
    for (let i = 0; i < todoList.length; i++) {
      response = await axios.get(todoUrlPrefix + todoList[i].id);
      console.log(`# ${i + 1}번째 Todo : `, response.data);
    }
  } catch(e) {
    if (e instanceof Error)  console.log(e.message);
    else console.log(e);
  }

};

requestAPI();
```

```
# TodoList :  (4) [{…}, {…}, {…}, {…}]
  0: {id: 123456789, todo: 'ES6 공부', desc: 'ES6공부를 해야 합니다', done: true}
  1: {id: 1713240846776, todo: 'Vue 학습', desc: 'Vue 학습을 해야 합니다', done: false}
  2: {id: 1713240846777, todo: '놀기', desc: '노는 것도 중요합니다.', done: true}
  3: {id: 1713240846778, todo: '야구장', desc: '프로야구 경기도 봐야합니다.', done: false}
  length: 4
  [[Prototype]]: Array(0)
5 XHR 로드 완료됨: GET "<URL>".
# 1번째 Todo :  {id: 123456789, todo: 'ES6 공부', desc: 'ES6공부를 해야 합니다', done: true}
  desc: "ES6공부를 해야 합니다"
  done: true
  id: 123456789
  todo: "ES6 공부"
  [[Prototype]]: Object
# 2번째 Todo :  {id: 1713240846776, todo: 'Vue 학습', desc: 'Vue 학습을 해야 합니다', done: false}
  desc: "Vue 학습을 해야 합니다"
  done: false
  id: 1713240846776
  todo: "Vue 학습"
  [[Prototype]]: Object
# 3번째 Todo :  {id: 1713240846777, todo: '놀기', desc: '노는 것도 중요합니다.', done: true}
  desc: "노는 것도 중요합니다."
  done: true
  id: 1713240846777
  todo: "놀기"
  [[Prototype]]: Object
# 4번째 Todo :  {id: 1713240846778, todo: '야구장', desc: '프로야구 경기도 봐야합니다.', done: false}
  desc: "프로야구 경기도 봐야합니다."
  done: false
  id: 1713240846778
  todo: "야구장"
  [[Prototype]]: Object
```

- Promise객체 - catch()
- async~await - try~catch()

Promise, async~await

- **axios.get()**
  - http GET요청 함수

```
<template>
    <div>
        <h2>콘솔을 확인합니다.</h2>
    </div>
</template>

<script setup>
import axios from "axios";

const requestAPI = async () => {
  const url = "/api/todolist/gdhong";
  const response = await axios.get(url);
  console.log("# 응답객체 : ", response);
};

requestAPI();
</script>
```

```
# 응답객체 :
▼ Object i
  ▶ config: {transitional: {…}, adapter: Array(2), transformRequest: Array(1), transformResp
  ▼ data: Array(6)
    ▶ 0: {id: 123456789, todo: 'ES6 공부', desc: 'ES6공부를 해야 합니다', done: true}
    ▶ 1: {id: 1713240846776, todo: 'Vue 학습', desc: 'Vue 학습을 해야 합니다', done: false}
    ▶ 2: {id: 1713240846777, todo: '놀기', desc: '노는 것도 중요합니다.', done: true}
    ▶ 3: {id: 1713240846778, todo: '야구장', desc: '프로야구 경기도 봐야합니다.', done: false}
    ▶ 4: {id: 1713245457312, todo: '윗몸일으키기 3세트', desc: '너무 빠르지 않게...', done: false}
    ▶ 5: {id: 1713245458593, todo: '윗몸일으키기 3세트', desc: '너무 빠르지 않게...', done: false}
      length: 6
    ▶ [[Prototype]]: Array(0)
  ▶ headers: AxiosHeaders {access-control-allow-origin: '*', cache-control: 'private, no-cac
  ▶ request: XMLHttpRequest {onreadystatechange: null, readyState: 4, timeout: 0, withCreden
    status: 200
    statusText: "OK"
  ▶ [[Prototype]]: Object
```

- data - 응답 데이터
- config - 요청시 사용된 데이터
- headers - 응답 HTTP header
- request - XMLHttpRequest객체 정보
- status - 응답 상태 코드
- statusText - 응답 상태 코드에 대한 문자열 정보

Promise, async~await

- **axios.post()**
  - http POST요청 함수
  - 많은 양의 데이터를 전달할 목적으로 보통 많이 사용
  - 보안에 민감한 데이터
  - 텍스트 이외의 데이터
- **axios.put(), axios.delete() - http PUT, http DELETE 요청 함수**

```
<script setup>
import axios from "axios";

const requestAPI = async () => {
  const url = "/api/todolist_long/gdhong";
  let data = { todo: "윗몸일으키기 3세트", desc: "너무 빠르지 않게..." };
  const resp1 = await axios.post(url, data);
  console.log(resp1.data);
};
requestAPI();
</script>
```

```
▼ {status: 'success', message: '추가 성공', item: {…}} i
  ▼ item:
      desc: "너무 빠르지 않게..."
      id: 1713245841685
      todo: "윗몸일으키기 3세트"
    ▶ [[Prototype]]: Object
    message: "추가 성공"
    status: "success"
  ▶ [[Prototype]]: Object
▶ XHR 로드 완료됨: POST "http://localhost:5174/api/todolist_long/gdhong".
```

axios 에러처리

Promise, async~await

- **axios기본 값 설정**
  - axios.defaults.baseURL = '/api/todolist_long'; → 1초의 의도적 지연시간을 가지는 엔드포인트
  - axios.defaults.headers.common['Authorization'] = JWT; → 인증토큰

```
<script setup>
import axios from "axios";

const requestAPI = async () => {
  const url = "/api/todolist_long/gdhong";
  try {
    const response = await axios.get(url, { timeout: 900 });
    console.log("# 응답객체 : ", response);
  } catch (e) {
    console.log("## 다음 오류가 발생했습니다.");
    if (e instanceof Error) console.log(e.message);
    else console.log(e);
  }
};

requestAPI();
</script>
```

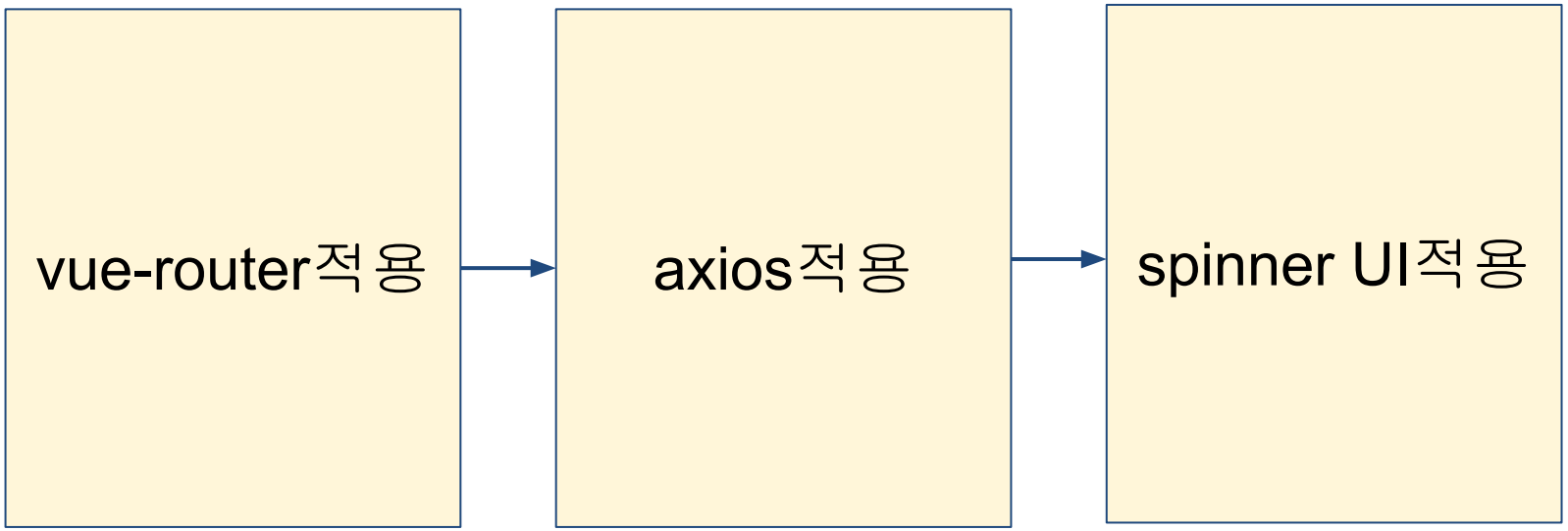
```
## 다음 오류가 발생했습니다.
timeout of 900ms exceeded
```

```
<script setup>
import axios from "axios";

const requestAPI = async () => {
  const url = "/api/todolist_long/gdhong";
  axios
    .get(url, { timeout: 900 })
    .then((response) => {
      console.log("# 응답객체 : ", response);
    })
};

requestAPI();
</script>
```

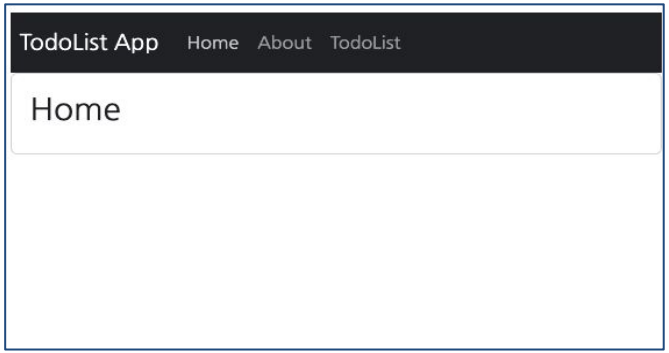
```
Uncaught (in promise)
▾ AxiosError i
    code: "ECONNABORTED"
  ▸ config: {transitional: {…}, adapter: Array(
    message: "timeout of 900ms exceeded"
    name: "AxiosError"
```

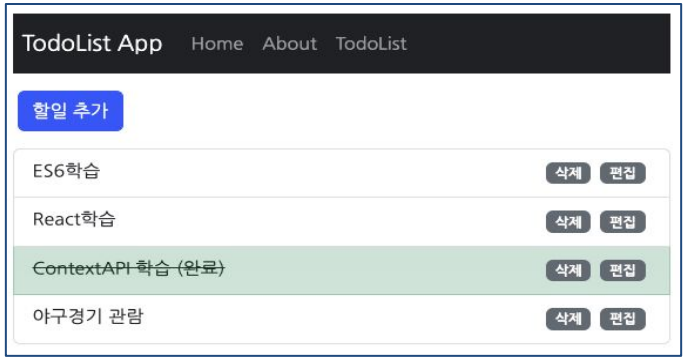
router-axios 적용 프로젝트

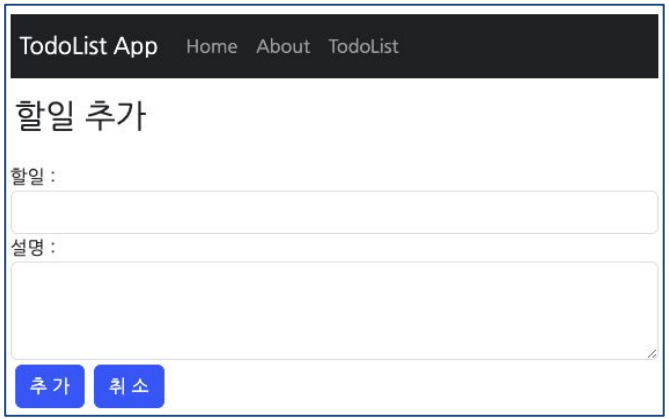
vue-router와 axios적용 프로젝트

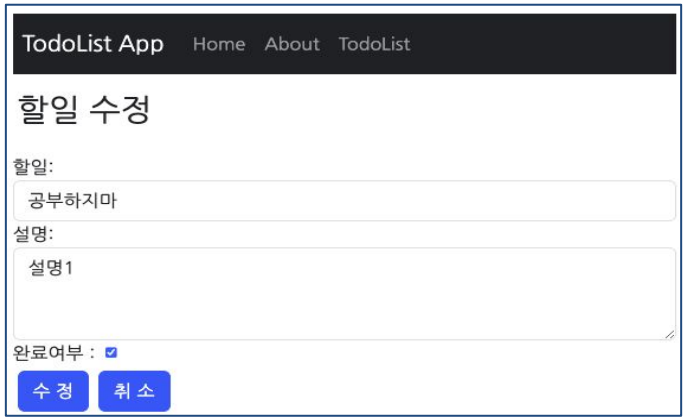


vue-router와 axios적용 프로젝트



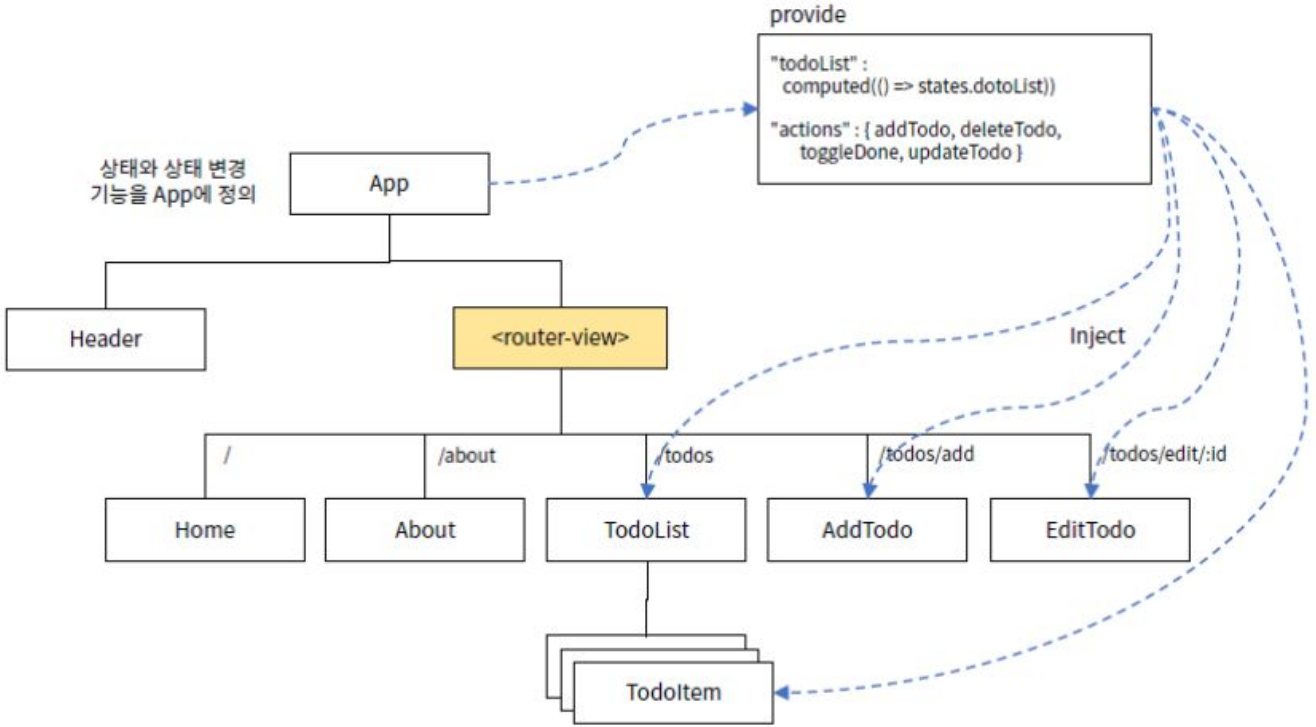




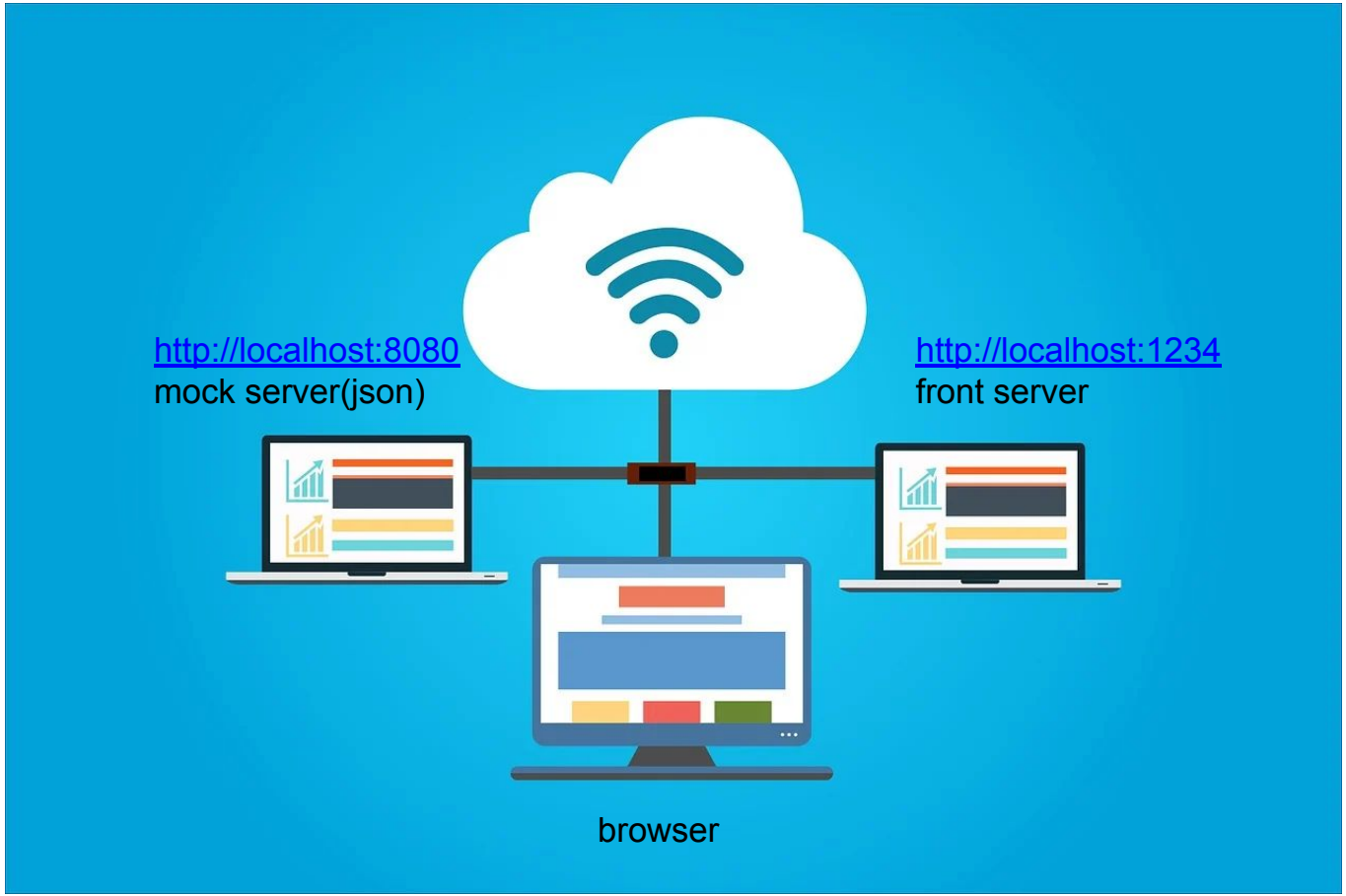


vue-router와 axios적용 프로젝트

- 컴포넌트 계층적 구조



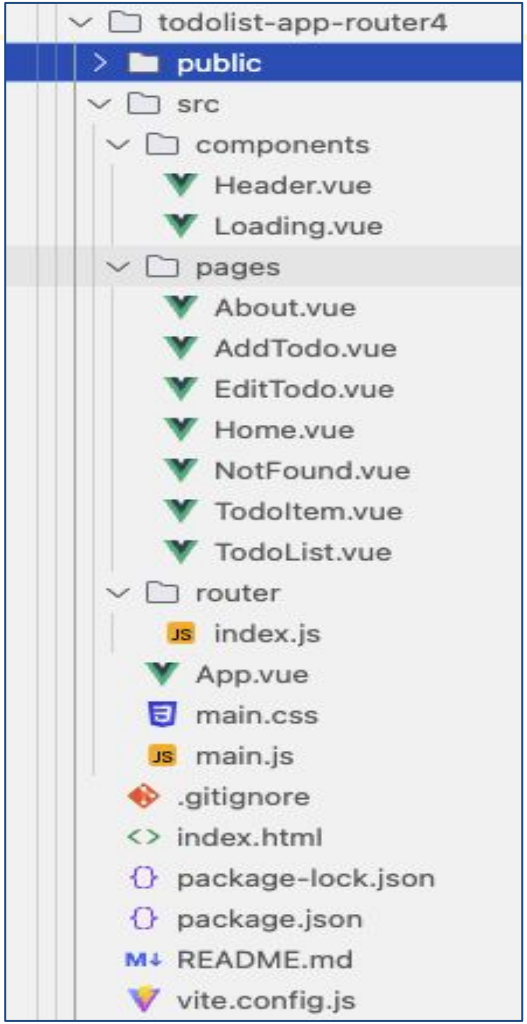
vue-router와 axios적용 프로젝트



vue-router와 axios적용 프로젝트

```
administrator@MacBook-Pro kb_ws_front % npm init vue todolist-router

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/todolist-router...

Done. Now run:

  cd todolist-router
  npm install
  npm run dev
```

```
administrator@MacBook-Pro kb_ws_front % npm install vue-router@4 bootstrap@5

added 2 packages, and audited 111 packages in 2s

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

found 0 vulnerabilities
```



vue-router와 axios적용 프로젝트-라우터


```
import { createRouter, createWebHistory } from 'vue-router'
import Home from '@/pages/Home.vue';
import About from '@/pages/About.vue';
import TodoList from '@/pages/TodoList.vue';
import AddTodo from '@/pages/AddTodo.vue';
import EditTodo from '@/pages/EditTodo.vue';
import NotFound from '@/pages/NotFound.vue';

const router = createRouter({
    history: createWebHistory(),
    routes: [
        { path: '/', component: Home },
        { path: '/about', component: About },
        { path: '/todos', component: TodoList },
        { path: '/todos/add', component: AddTodo },
        { path: '/todos/edit/:id', component: EditTodo },
        { path: '/:paths(.*)*', component: NotFound },
    ]
})

export default router;
```

router/index.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';

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

main.js

```
body {  margin: 0; padding: 0; font-family: sans-serif;  }
.title {  text-align: center; font-weight: bold; font-size: 20pt;  }
.todo-done {  text-decoration: line-through;  }
.container {  padding: 10px 10px 10px 10px;  }
.panel-borderless {  border: 0; box-shadow: none;  }
.pointer {  cursor: pointer;  }
```

main.css

vue-router와 axios적용 프로젝트-라우터

App.vue

```vue
<template>
    <div class="container">
        <Header />
        <router-view />
    </div>
</template>

<script setup>
import { reactive, computed, provide } from 'vue'
import Header from '@/components/Header.vue'

const states = reactive({
  todoList : [
    { id: 1, todo: "ES6학습", desc: "설명1", done: false },
    { id: 2, todo: "React학습", desc: "설명2", done: false },
    { id: 3, todo: "ContextAPI 학습", desc: "설명3", done: true },
    { id: 4, todo: "야구경기 관람", desc: "설명4", done: false },
  ]
})

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

const updateTodo = ({ id, todo, desc, done }) => {
  let index = states.todoList.findIndex((todo) => todo.id == id);
  states.todoList[index] = { ...states.todoList[index], todo, desc, done };
};

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

const toggleDone = (id) => {
  let index = states.todoList.fin
  states.todoList[index].done = !s
}

provide('todoList', computed(()=>states.todoList))
provide('actions', { addTodo, deleteTodo, toggleDone, updateTodo })
</script>
```

라우팅 처리된 결과 넣을 위치

reactive(객체) 반응성

provide 하위 컴포넌트에서 주입해서 언제든 사용 가능

vue-router와 axios적용 프로젝트-라우터

Header.vue

```vue
<template>
  <nav class="navbar navbar-expand-sm bg-dark navbar-dark">
    <span class="navbar-brand ps-2">TodoList App</span>
    <button class="navbar-toggler" type="button" @click="isNavShow = !isNavShow">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div :class="isNavShow ? 'collapse navbar-collapse show' : 'collapse navbar-collapse'">
      <ul class="navbar-nav">
        <li class="nav-item">
          <router-link class="nav-link" to="/">
            Home
          </router-link>
        </li>
        <li class="nav-item">
          <router-link class="nav-link" to="/about">
            About
          </router-link>
        </li>
        <li class="nav-item">
          <router-link class="nav-link" to="/todos">
            TodoList
          </router-link>
        </li>
      </ul>
    </div>
  </nav>
</template>
```

menu link <a>

isNavShow
반응성 적용

```vue
<script setup>
import { ref } from 'vue';

const isNavShow = ref(false);
</script>
```

About.vue

```vue
<template>
    <div class="card card-body">
        <h2>About</h2>
    </div>
</template>
```

NotFound.vue

```vue
<template>
    <div class="m-3">
        <h3>존재하지 않는 경로</h3>
        <p>요청 경로 : {{currentRoute.path}}</p>
    </div>
</template>

<script setup>
import { useRoute } from 'vue-router'
const currentRoute = useRoute();
</script>
```

vue-router와 axios적용 프로젝트-라우터

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>
        </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>
    </div>
</template>

<script setup>
import {inject} from 'vue';
import TodoItem from '@/pages/Todo

const todoList = inject('todoList');
</script>
```

할일 추가 페이지로 link

for문을 이용하여 할일 리스트 프린트

provide해둔 대상을 주입할 때 inject

App.vue

```
provide('todoList', computed(()⇒states.todo
provide('actions', { addTodo, deleteTodo, toggleDone, updateTodo })
```

vue-router와 axios적용 프로젝트-라우터

TodoItem.vue

```vue
<template>
    <li :class="todoItem.done ? 'list-group-item list-group                group-item'">
      <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 { inject } from 'vue';

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

const router = useRouter();
const { deleteTodo, toggleDone } = inject('actions');
</script>
```

클릭하면 라우팅 호출
router.push('호출주소/파라메터');

App.vue
provide → inject

```
provide('todoList', computed(()⇒states.todoList))
provide('actions', { addTodo, deleteTodo, toggleDone, updateTodo })
```

vue-router와 axios적용 프로젝트-라우터

EditTodo.vue

```
<template>
    <div class="row">
        <div class="col p-3">
            <h2>할일 수정</h2>
        </div>
    </div>
    <div class="row">
        <div class="col">
            <div class="form-group">
                <label htmlFor="todo">할일:</label>
                <input type="text" class="form-control" id="todo" v-model="todoItem.todo" />
            </div>
            <div class="form-group">
                <label htmlFor="desc">설명:</label>
                <textarea class="form-control" rows="3" id="desc" v-model="todoItem.desc"></textarea>
            </div>
            <div class="form-group">
                <label htmlFor="done">완료여부 : </label> 
                <input type="checkbox" v-model="todoItem.done" />
            </div>
            <div class="form-group">
                <button type="button" class="btn btn-primary m-1" @click="updateTodoHandler">
                  수 정
                </button>
                <button type="button" class="btn btn-primary m-1" @click="router.push('/todos')">
                  취 소
                </button>
            </div>
        </div>
    </div>
</template>
```

```
reactive } from 'vue';
{ useRouter, useRoute } from 'vue-router';

const todoList = inject('todoList');
const { updateTodo } = inject('actions');
const router = useRouter();
const currentRoute = useRoute();

const matchedTodoItem = todoList.value.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>
```

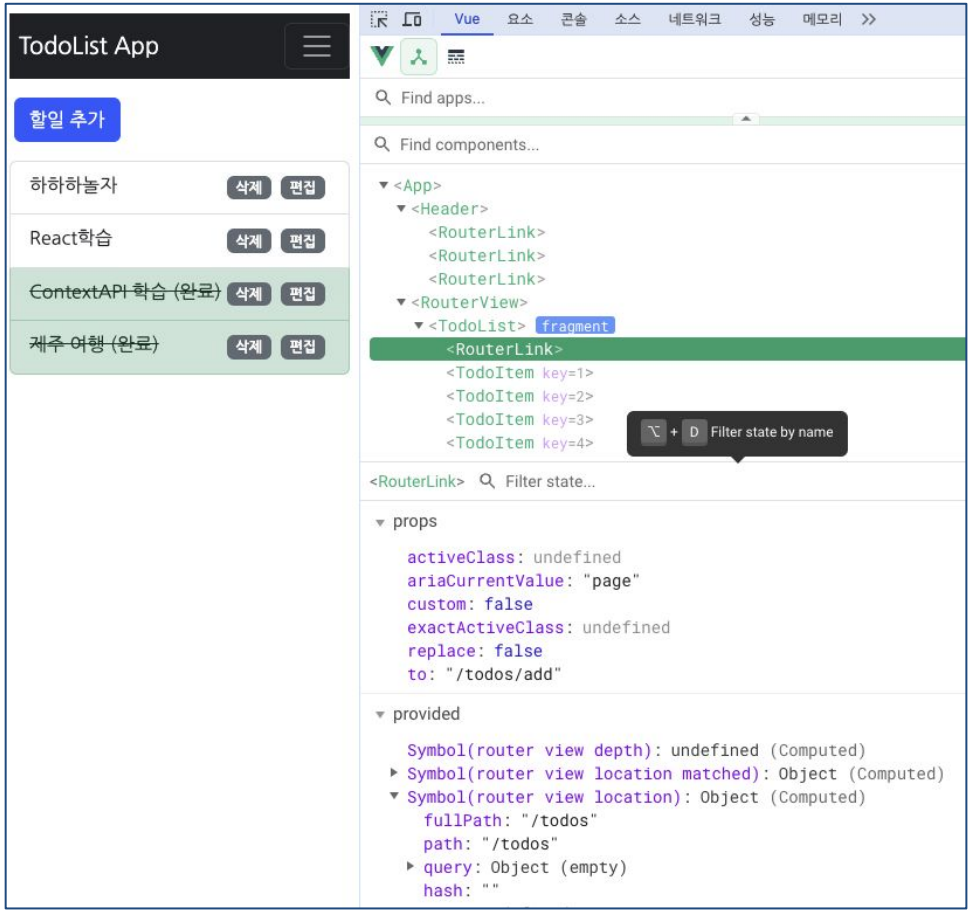
vue-router와 axios적용 프로젝트-라우터

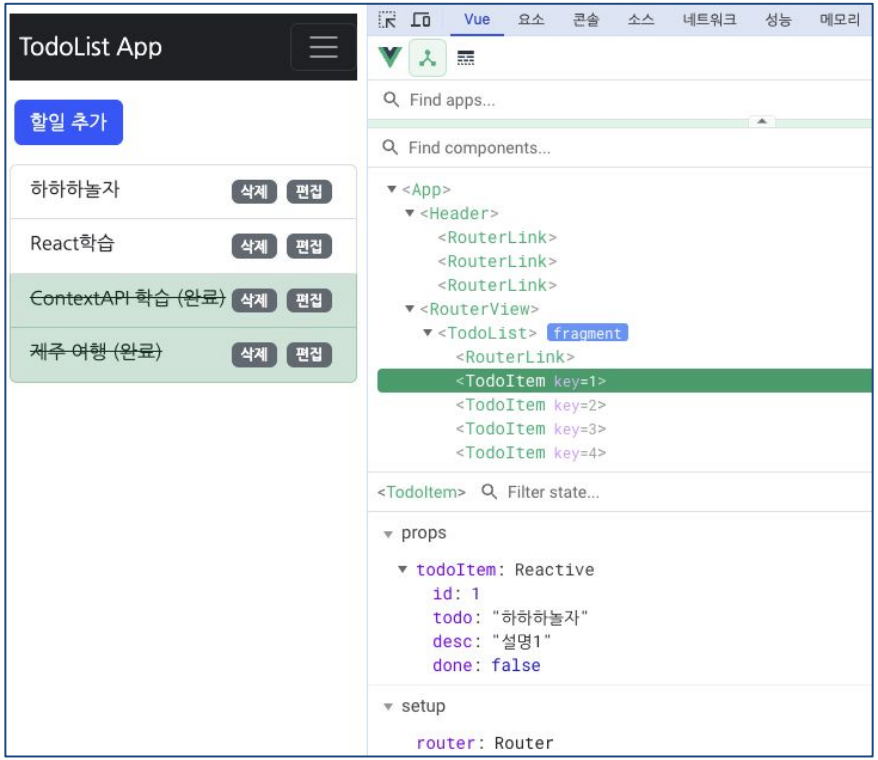
AddTodo.vue

```vue
<template>
    <div class="row">
        <div class="col p-3">
            <h2>할일 추가</h2>
        </div>
    </div>
    <div class="row">
        <div class="col">
            <div class="form-group">
                <label htmlFor="todo">할일 :</label>
                <input type="text" class="form-control" id="todo" v-model="todoItem.todo" />
            </div>
            <div class="form-group">
                <label htmlFor="desc">설명 :</label>
                <textarea class="form-control" rows="3" id="desc" v-model="todoItem.desc"></textarea>
            </div>
            <div class="form-group">
                <button type="button" class="btn btn-primary m-1" @click="addTodoHandler">
                추 가
                </button>
                <button type="button" class="btn btn-primary m-1" @click="router.push('/todos')">
                취 소
                </button>
            </div>
        </div>
    </div>
</template>
```

```vue
<script setup>
import { inject, reactive } from 'vue';
import { useRouter } from 'vue-router';

const router = useRouter();
const { addTodo } = inject('actions');
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-router와 axios적용 프로젝트-라우터





vue-router와 axios적용 프로젝트-axios적

- **axios 라이브러리 설치**

```
administrator@MacBook-Pro todolist-app-router3 % npm install --save axios
```

- **mock server start**

```
administrator@MacBook-Pro todosvc-master % npm run start

> todosvc@1.0.0 start
> node build/index.js

할일 목록 서비스가 8000번 포트에서 시작되었습니다!
```

- **vite.config.js proxy설정 변경**

```
server: {
  proxy: {
    "/api": {
      target: "http://localhost:8000",
      changeOrigin: true,
      rewrite: (path) => path.replace(/^\/api/, ""),
    },
  },
```

vue-router와 axios적용 프로젝트-axios적

App.vue

```vue
<template>
  <div class="container">
    <Header />
    <router-view />
  </div>
</template>

<script setup>
import { reactive, provide, computed } from 'vue'
import Header from '@/components/Header.vue'
import axios from 'axios';

const owner = "gdhong";
//의도적 지연 시간을 발생시키는 /todolist_long 이용
const BASEURI = "/todolist_long";
const states = reactive({ todoList:[] })

const getTodoList = async () => {
  try {
    const response = await axios.get(BASEURI + `/${owner}`);
    if (response.status === 200) {
      states.todoList = response.data;
    } else {
      alert('데이터 조회 실패');
    }
  } catch(error) {
    alert('에러발생 :' + error);
  }
}
```

axios call

axios call

```vue
<script setup>
//TodoItem을 변경합니다.
const updateTodo = async ({ id, todo, desc, done }, successCallback) => {
  try {
    const payload = { todo, desc, done };
    const response = await axios.put(BASEURI + `/${owner}/${id}`, payload);
    if (response.data.status === "success") {
      let index = states.todoList.findIndex((todo) => todo.id === id);
      states.todoList[index] = { id, todo, desc, done };
      successCallback();
    } else {
      alert('Todo 변경 실패 : ' + response.data.message);
    }
  } catch(error) {
    alert('에러발생 :' + error);
  }
}
//기존 TodoItem을 삭제합니다.
const deleteTodo = async (id) => {
  try {
    const response = await axios.delete(BASEURI + `/${owner}/${id}`)
    if (response.data.status === "success") {
      let index = states.todoList.findIndex((todo) => todo.id === id);
      states.todoList.splice(index, 1);
    } else {
      alert('Todo 삭제 실패 : ' + response.data.message);
    }
  } catch(error) {
    alert('에러발생 :' + error);
  }
}
```

axios call

vue-router와 axios적용 프로젝트-axios적

```
//기존 TodoItem의 완료여부(done) 값을 토글합니다.
const toggleDone = async (id) => {
  try {
    const response = await axios.put(BASEURI + `/${owner}/${id}/done`)
    if (response.data.status === "success") {
        let index = states.todoList.findIndex((todo) => todo.id === id);
        states.todoList[index].done = !states.todoList[index].done;
    } else {
        alert('Todo 완료 변경 실패 : ' + response.data.message);
    }
  } catch(error) {
    alert('에러발생 :' + error);
  }
}


provide('todoList', computed(()=>states.todoList));
provide('actions', { addTodo, deleteTodo, toggleDone, updateTodo, fetchTodoList })

fetchTodoList();
</script>
```

axios call
call pattern
/hong/1/done
/kim/2/done

provide → inject

vue-router와 axios적용 프로젝트-axios적

AddTodo.vue

```
<script setup>
import { inject, reactive } from 'vue';
import { useRouter } from 'vue-router';

const router = useRouter();
const { addTodo } = inject('actions');
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>
```

addTodo()후
/todos 호출

EditTodo.vue

```
<script setup>
import { inject, reactive } from 'vue';
import { useRouter, useRoute } from 'vue-router';

const todoList = inject('todoList');
const { updateTodo } = inject('actions');
const router = useRouter();
const currentRoute = useRoute();

const matchedTodoItem = todoList.value.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>
```

updateTodo()후
/todos 호출

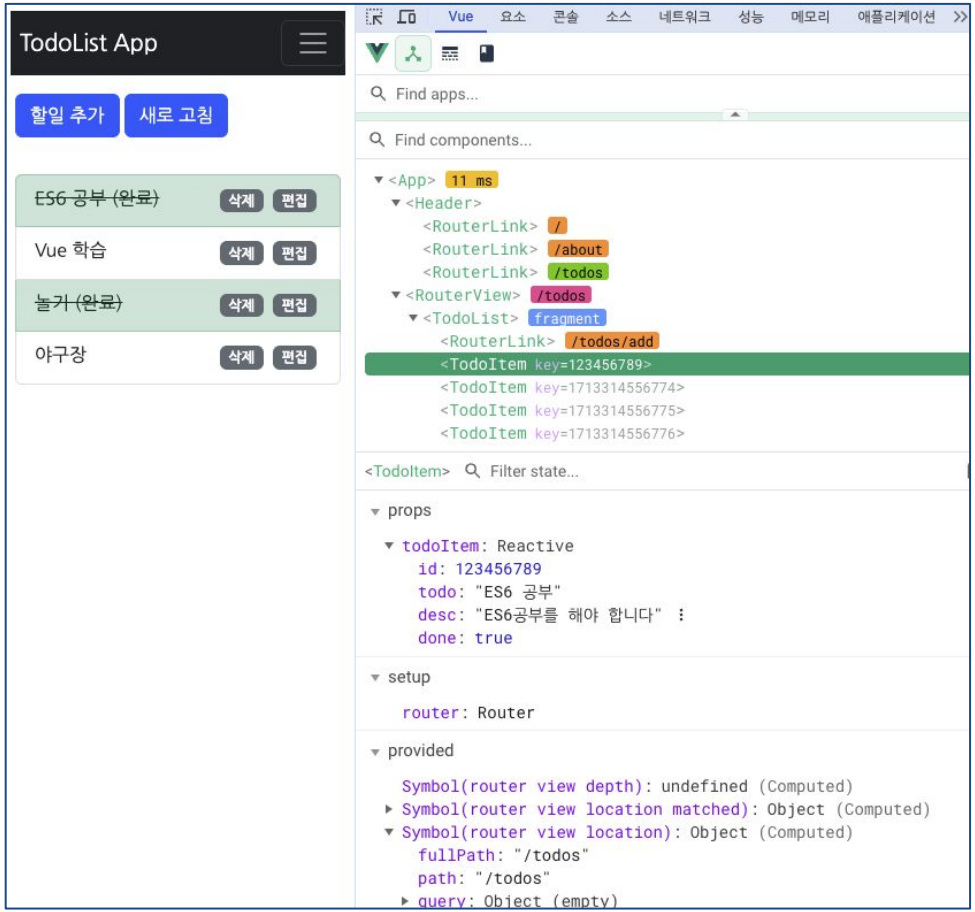
vue-router와 axios적용 프로젝트-axios적

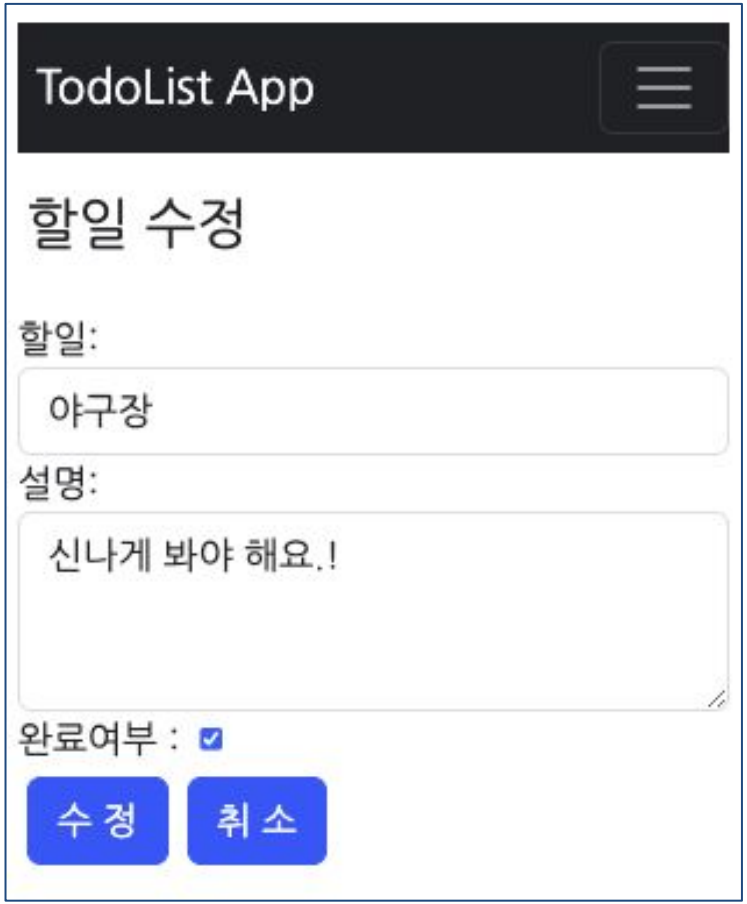
TodoList.vue

```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 p-3">
            <ul class="list-group">
                <TodoItem v-for="todoItem in todoList" :key="todoItem.id" :todoItem="todoItem" />
            </ul>
        </div>
    </div>
</template>

<script setup>
import {inject} from 'vue';
import TodoItem from '@/pages/TodoItem.vue'

const todoList = inject('todoList');
const { fetchTodoList } = inject('actions');
</script>
```

provide → inject

vue-router와 axios적용 프로젝트-axios적





- 지연시간 동안 기다려야함.
- 비동기 처리는 일반적으로 연결/로딩 중임을 알리는 **spinner UI**를 사용 권장

```
administrator@MacBook-Pro todolist-app-router4 % npm install --save vue-csspin
```

Loading.vue

```
<VueCsspin message="Loading" spin-style="cp-flip" />
</template>

<script setup>
import { VueCsspin } from 'vue-csspin'
import 'vue-csspin/dist/vue-csspin.css'
</script>
```

vue-router와 axios적용 프로젝트-spinnerUI적용

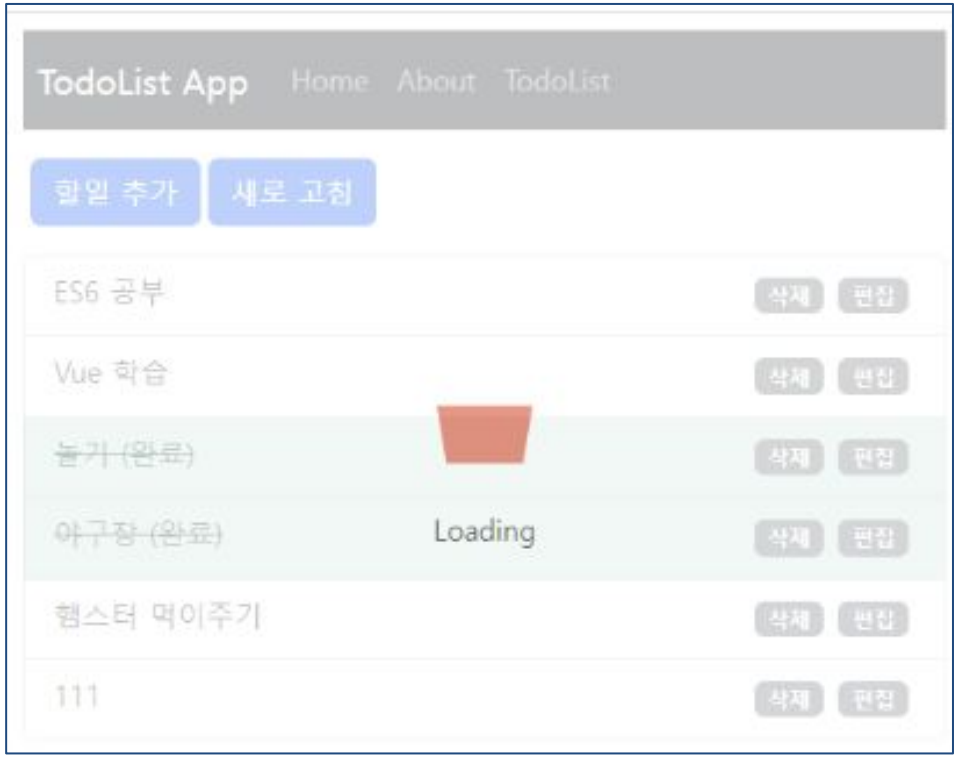
App.vue

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

<script setup>
import { reactive, provide, computed } from 'vue'
import Header from '@/components/Header.vue'
import Loading from '@/components/Loading.vue'
import axios from 'axios';

const owner = "gdhong";
const BASEURI = "/api/todolist_long";

const states = reactive({ todoList:[], isLoading:false })
```

spinner를 넣어주고 싶은 곳마다 설정

```vue
//TodoList 목록을 조회합니다.
const fetchTodoList = async () => {
  states.isLoading = true;
  try {
    const response = await axios.get(BASEURI + `/${owner}`);
    if (response.status === 200) {
        states.todoList = response.data;
    } else {
        alert('데이터 조회 실패');
    }
  } catch(error) {
    alert('에러발생 :' + error);
  }
  states.isLoading = false;
}
```

vue-router와 axios적용 프로젝트-spinnerUI적용



핵심정리

- **mock server구축**
  - http://localhost:8000
  - http://get요청
- **크로스 오리진 문제**
- **CORS**
- **프록시 우회**
  - 프록시 설정
  - 프록시 우회 CORS문제 해결
- **axios 기본 사용법**
- **Promise와 async~await 비교**
- **에러처리**
  - 에러에 대한 대응 가능