KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

axios를 이용한 HTTP 통신

백엔드랑 연결하여 사용

[KB] IT's Your Life

KB 국민은행

REST(Representational State Transfer)

- 정보를 자원으로 간주 식별이 가능해야한다.
- 각 자원에게 URI를 배정 표기방법 식별방법
- 자원에 대한 조작은 HTTP 메서드로 표현

→ URI와 HTTP 메소드를 이용해 객체화된 서비스에 접근하는 것

- 예 : 게시판

get 정보 읽기
post 정보 생성
웹브라우저에서는 위 2개의 메소드만 사용할 수 있다.
Form태그에 메소드 속성에 2개만 입력가능

나머지는 js를 통해 ajax통신을 할 때 사용 가능하다

URI	METHOD	의미
/board	GET	전체 목록 추출
/board/{boardId} 식별정보	GET	boardId의 게시글 한 개 추출
/board	POST	새로운 게시글 생성 내용은 BODY에 JSON으로 지정
/board/{boardId}	PUT 수정	boardId의 게시글 수정 내용은 BODY에 JSON으로 지정
/board/{boardId}	DELETE 삭제	boardId의 게시글 삭제

경로상에 대상이 있음

라우터에서는 {...} 은 동적인자로
/~/:id 이런식으로

1 REST API란

REST(Representational State Transfer)

어떤 요청(메소드)이냐에따라 {id}가 붙을 수도 있따.

todos

URI	METHOD	의미
/todos	GET	전체 TODO 목록 추출
/todos/{id}	GET	id의 TODO 한 개 추출
/todos	POST	새로운 TODO 생성 내용은 BODY에 JSON으로 지정
/todos/{id}	PUT	id의 TODO 수정 bin은 json말고 다른걸로 내용은 BODY에 JSON으로 지정
/todos/{id}	DELETE	id의 TODO 삭제

post&put에는 body파트가 있다.

get delete에는 body파트가 없다. 고로 추가적인 정보를 표시하고 싶다면 querystring 사용해라.

1 REST API란

REST 구성

- 자원(RESOURCE) – URI
- 행위(Verb) – HTTP METHOD
- 표현(Representations) – 일반적으로 JSON 사용 정보 자체에 대한 표현. JSON, bin은 다른방식

- REST로 처리되는 URI/METHOD 조합 → REST API
- REST API 서비스
  - 클라이언트의 종류에 제한을 두지 않음
  - 웹 브라우저, 일반 애플리케이션 등과 통신 가능

초창기에는 html말고 xml을 많이 사용했다.
좀더 구조화되어 있어서. 하지만 오버헤드가 너무 커.

AJAX asynchronous javascript and xml

JSON : 자바스크립트 객체 표기법을 써서 정보를 구조화하자. 유연하고 간단함.

ajax에서 xml은 잘 안쓰고 xml대신 json을 많이 사용.

rest를 보완한 graphQL

프로젝트 생성

npm init vue axios-test-app
cd axios-test-app
npm install

라우터 사용 안함 이번엔

서버 구현을 안해서
1. 페이크 서버를 이용. 쉬움. post put delete연습 잘 못함
2. json-server 모듈 이용하여 나만에 서버 만들기

npmjs에서 json-server 검색하여 알아보자
모듈로 사용할 수도 있고
따로 실행파일이 있어서 독자적으로
실행가능함

json-server

전역 패키지로 설치

-g없으면 현재 모듈에만 지역적으로 설치
매번 사용할 것 같은데 전역으로 설치하여 한번만 설치하자. -g옵션 줘
지역이 아니므로 해당 package.json에 안표기됨.

npm install -g json-server

운영할 데이터베이스에 해당하는 db.json 파일 준비

- 프로젝트 루트 디렉토리에 db.json 작성

```
{
  "todos": [
    { "id": "1", "todo": "야구장", "desc": "프로야구 경기도 봐야합니다.", "done": false },
    { "id": "2", "todo": "놀기", "desc": "노는 것도 중요합니다.", "done": false },
    { "id": "3", "todo": "Vue 학습", "desc": "Vue 학습을 해야 합니다", "done": false },
    { "id": "4", "todo": "ES6 공부", "desc": "ES6공부를 해야 합니다", "done": false }
  ]
}
```

(todo) 속성은 엔트포인트가 된다.

식별자는 문자열

서버의 데이터베이스에 해당되는 db json파일이 하나 필요함. db.json 파일 직접 만들고
JSON형식으로 데이터를 채우자

기동

운영할 파일 이름을 통해서
서버 실행.

이 방법은 전역설치 되었을 때 서버실행 하는 방법

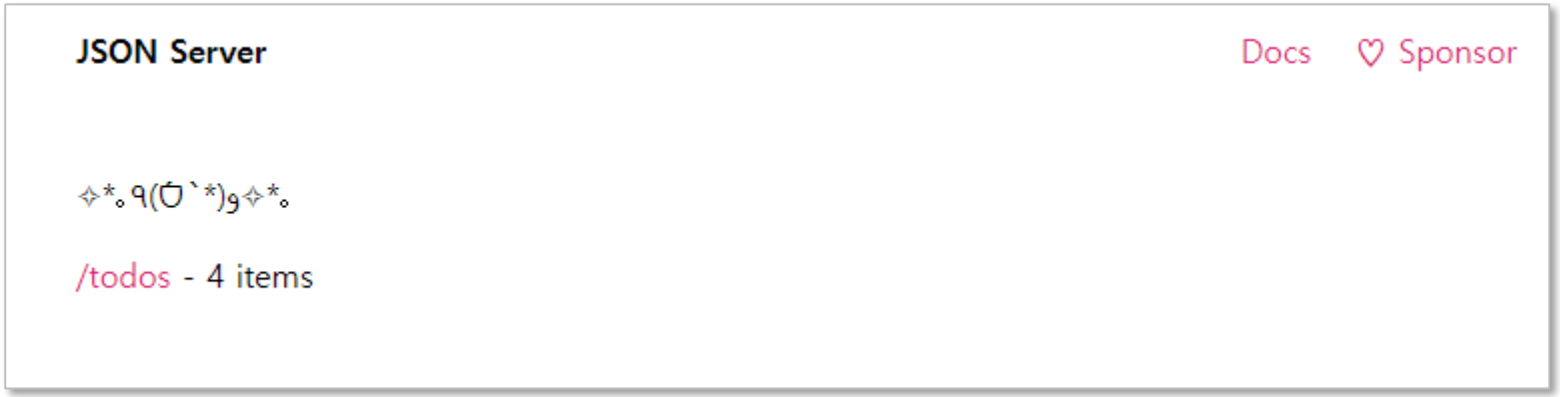
실행하면 엔드포인트들의 리스트가 뜬다

> json-server db.json 또는
> npx json-server db.json

또다른 실행 방법. npx(x는 execute)로 서버 실행하는 방법. npx방법은 해당 패키지 설치 없이도 실행 가능. 1회용 실행.

기동 확인

- http://localhost:3000/



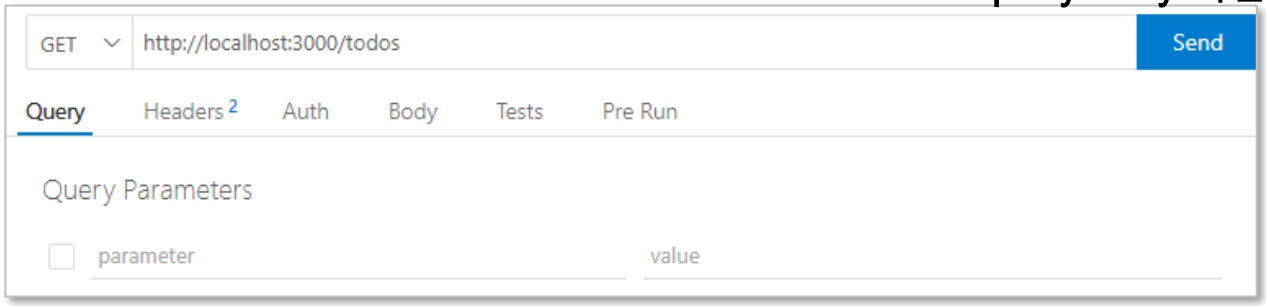
Thunder Client에서 확인

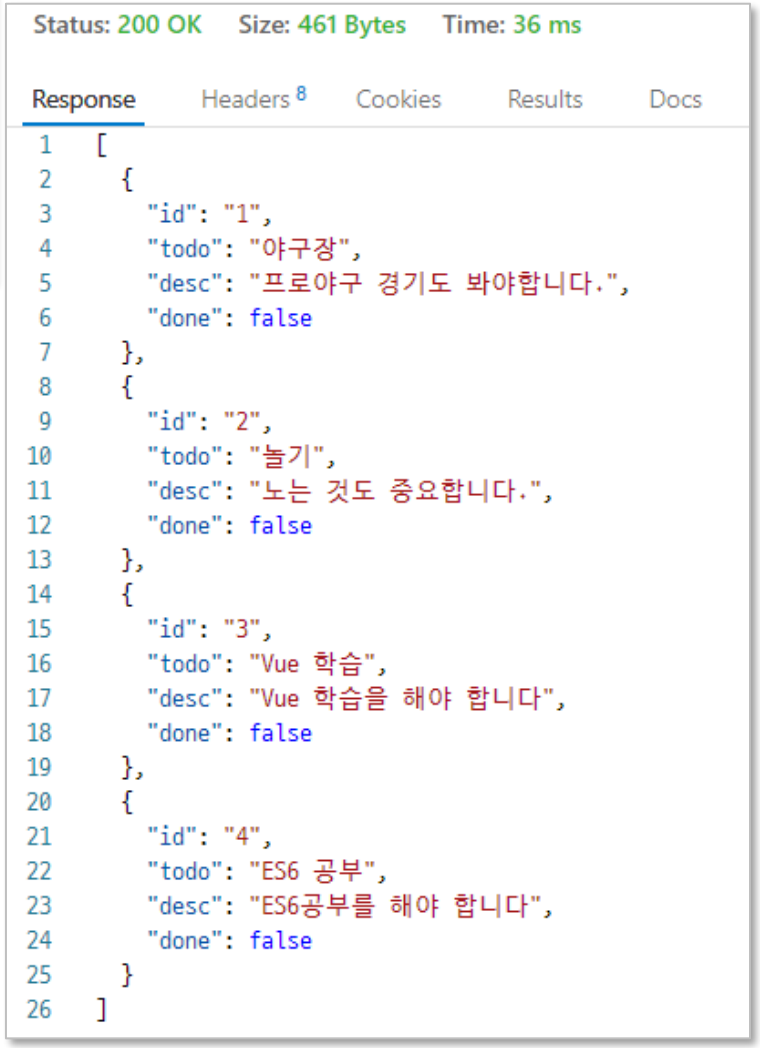
- GET http://localhost:3000/todos

확인을 위한 vscode확장팩 설치
메뉴바에서 thunder client > new request

설치하고 연습

query body 직접 구성 가능





http://localhost:3000/todos?_sort=-id

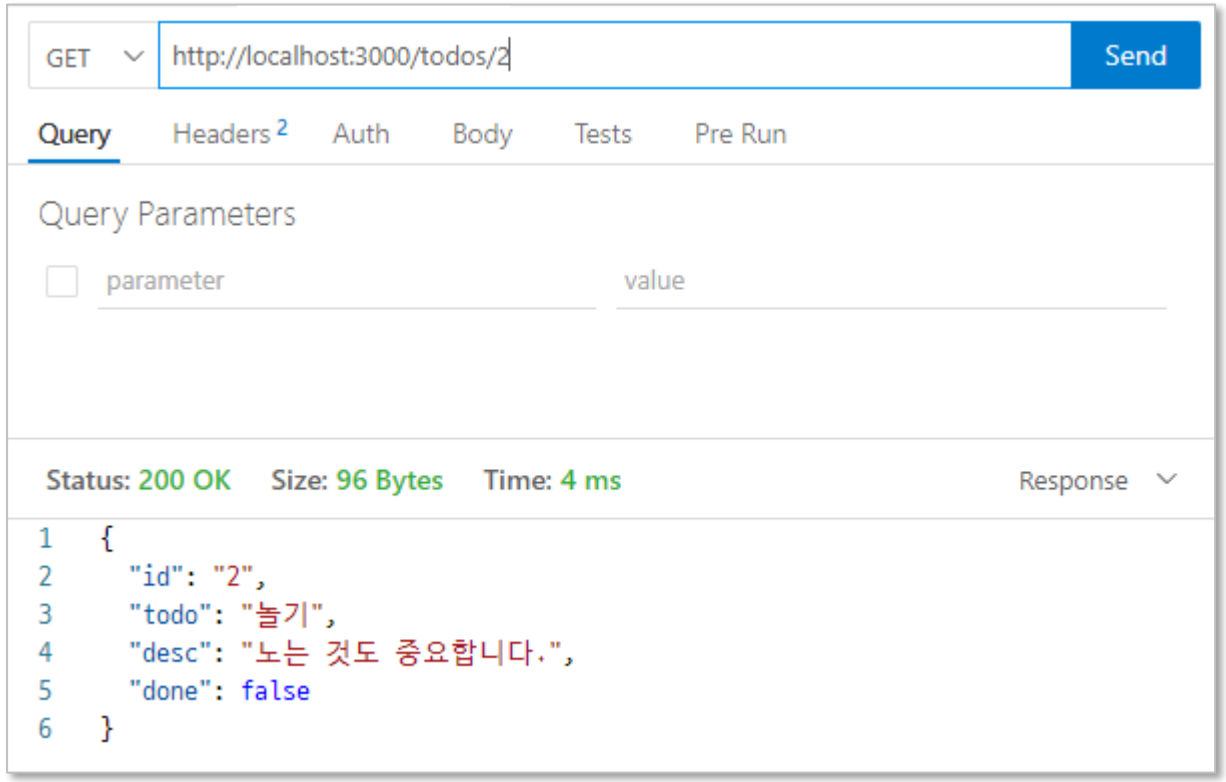
문법 활용하여 해당 데이터 달라고

http://localhost:3000/todos?_sort=-id&_page=1&_per_page=2
한페이지당 2개씩 표시해라

Thunder Client에서 확인

- GET http://localhost:3000/todos/2

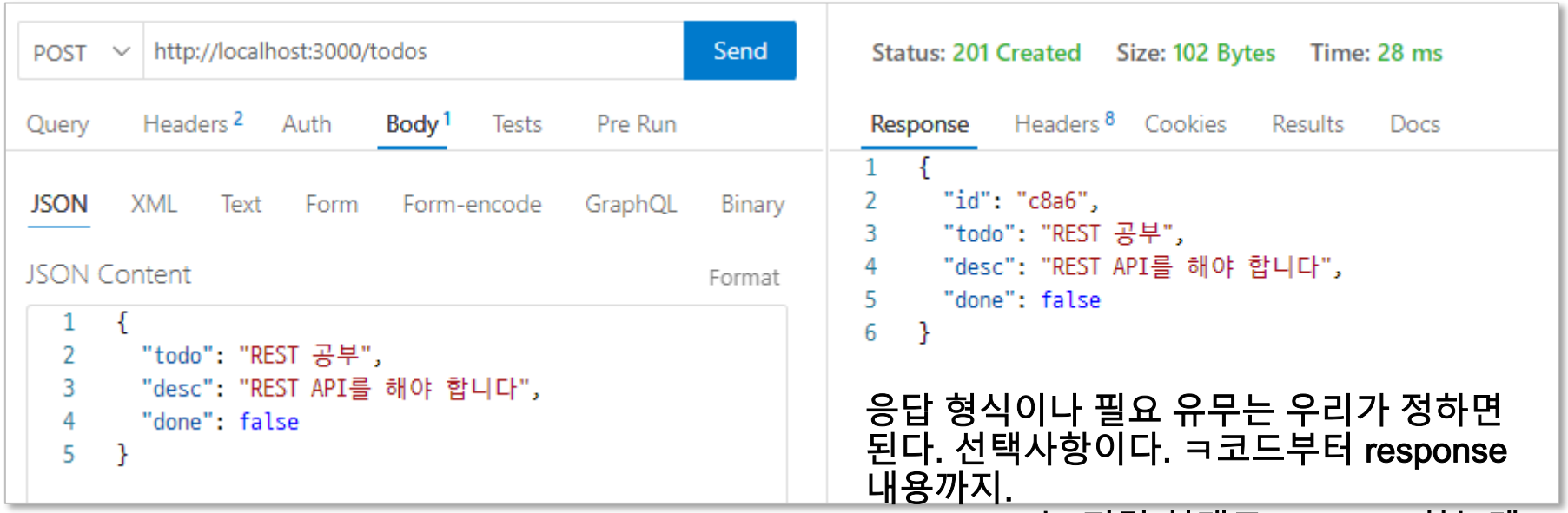
원하는 자원의 아이디를 뒤에 붙이면 돼



브라우저에서 원하는 것을 얻는 메소드는 get밖에 없음

Todo 추가하기

- POST http://localhost:3000/todos



이때는 추가할 내용을 body에 JSON으로 작성

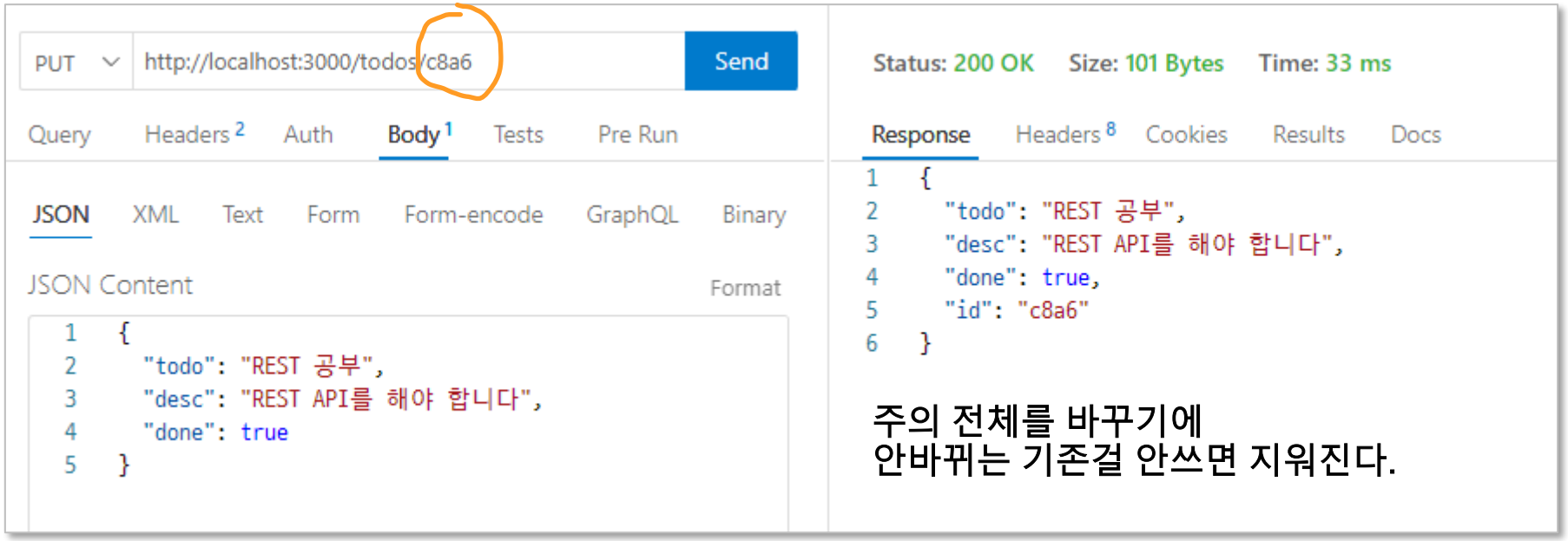
응답을 확인하자
상태코드200 201생성 성공
사이즈, 시간
response 내용 : 새로 저장된 정보가 리턴함 이번에 db에 들어갈때 자동으로 배정된 id도 추가되어 돌아옴

응답 형식이나 필요 유무는 우리가 정하면 된다. 선택사항이다. ㅋ코드부터 response 내용까지.
json-server는 저런 형태로 response하는게 디폴트.

Todo 수정하기

- PUT http://localhost:3000/todos/c8a6

200 상태 코드

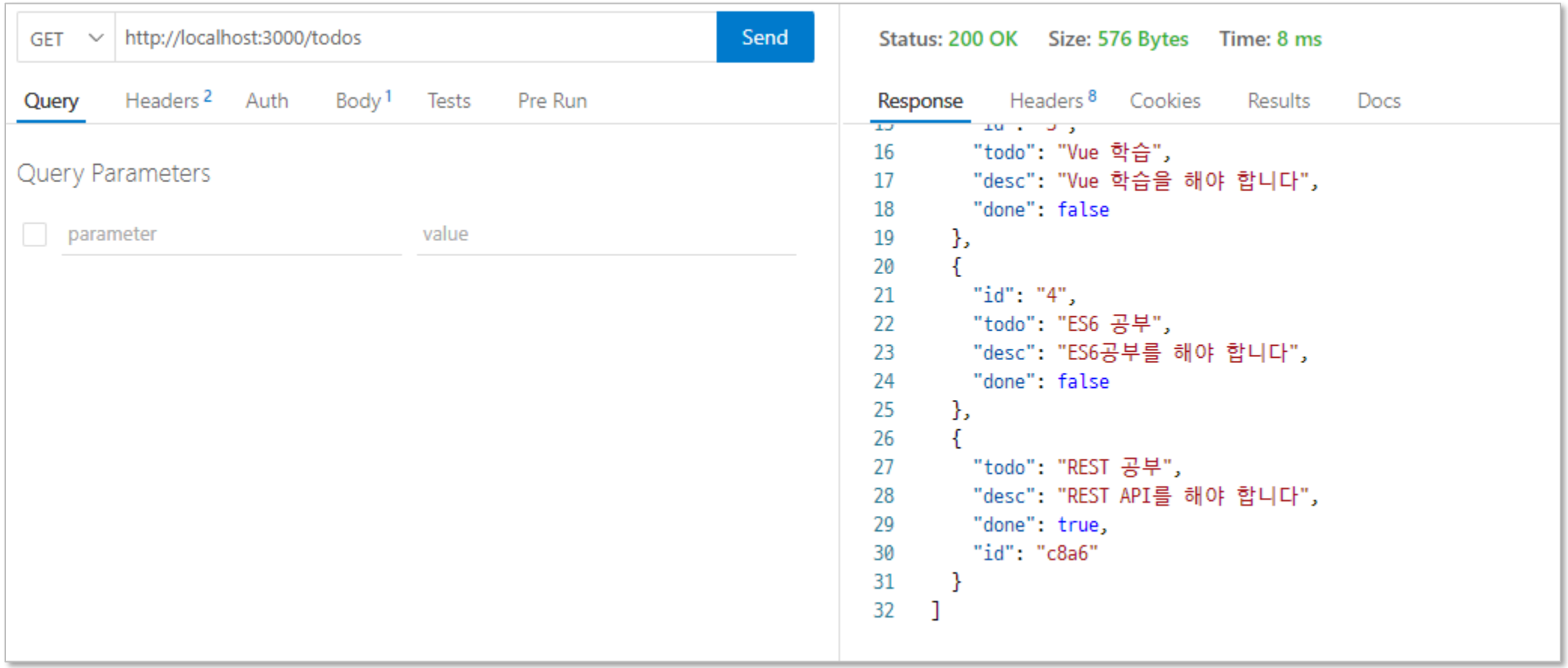


주의 전체를 바꾸기에
안바뀌는 기존걸 안쓰면 지워진다.

찾을 수 없다면 404
json형태가 잘못되면 500

변경사항 확인하기

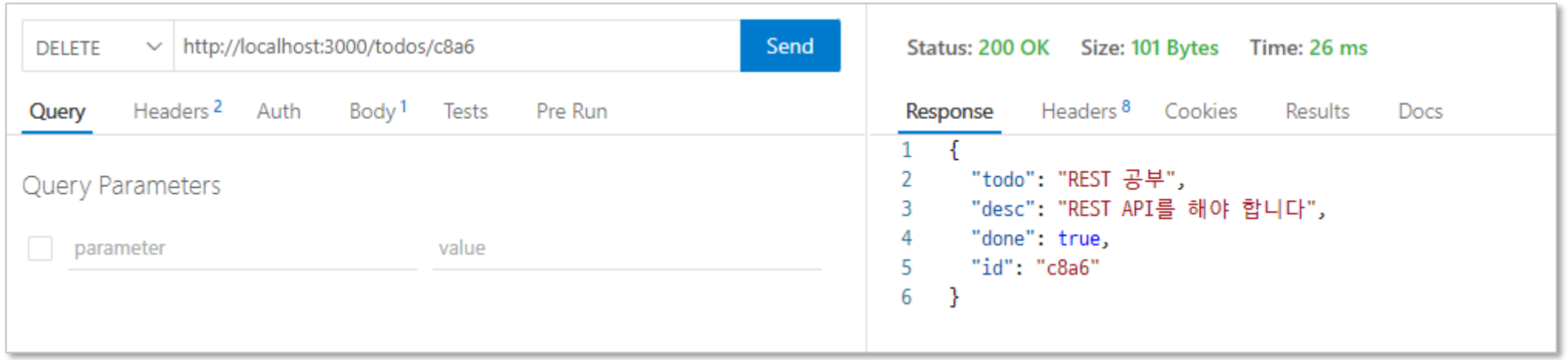
- GET http://localhost:3000/todos



Todo 삭제하기

- DELETE http://localhost:3000/todos/c8a6

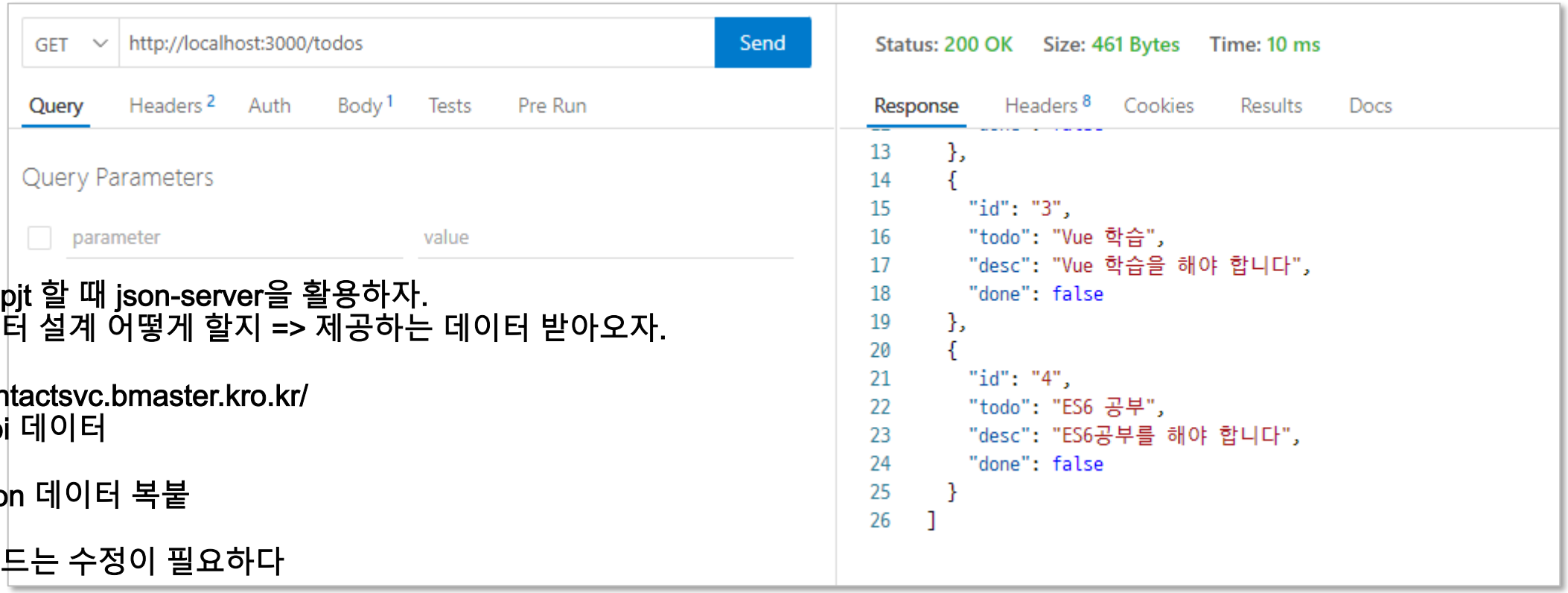
ㅇ delete는 바디 필요 없죠?



json-server는
지워진 내용 응답으로 돌아오네

변경사항 확인하기

- GET http://localhost:3000/todos



스켈레톤 pjt 할 때 json-server을 활용하자.
초기 데이터 설계 어떻게 할지 => 제공하는 데이터 받아오자.

https://contactsvc.bmaster.kro.kr/
테스트 api 데이터

필요한 json 데이터 복붙

식별자 필드는 수정이 필요하다

https://www.npmjs.com/package/json-server

간단한 사용법과 문법.

fetch는 웹브에서 기본으로 제공해주는 함수.
리액트에서는 fetch
vue에서는 액시오스 많이 쓴다

axios

- HTTP 기반 통신을 지원하는 자바스크립트 라이브러리
- axios와 fetch와 비교

둘다 비동기

구분	axios	fetch
모듈 설치	설치해야 함 (npm install —save axios)	설치할 필요 없음 (브라우저 내장 API)
Promise API	사용	사용
브라우저 호환성	뛰어남	IE 지원하지 않음 (IE에서 사용하려면 Polyfill 라이브러리를 사용해야 함)
timeout 기능	지원 (timeout 시간 내에 응답이 오지 않으면 중단시킬 수 있음)	지원하지 않음
JSON 자동 변환	지원 (Content-type 정보를 이용해 자동으로 객체로 변환함)	지원하지 않음 (수신한 JSON 데이터를 객체로 변환하는 Promise 체인을 추가해야 함)

리턴값 둘다
promise객체

jwt연동도
axios가 더 쉽다

axios 설치

npm install axios

src/App.vue

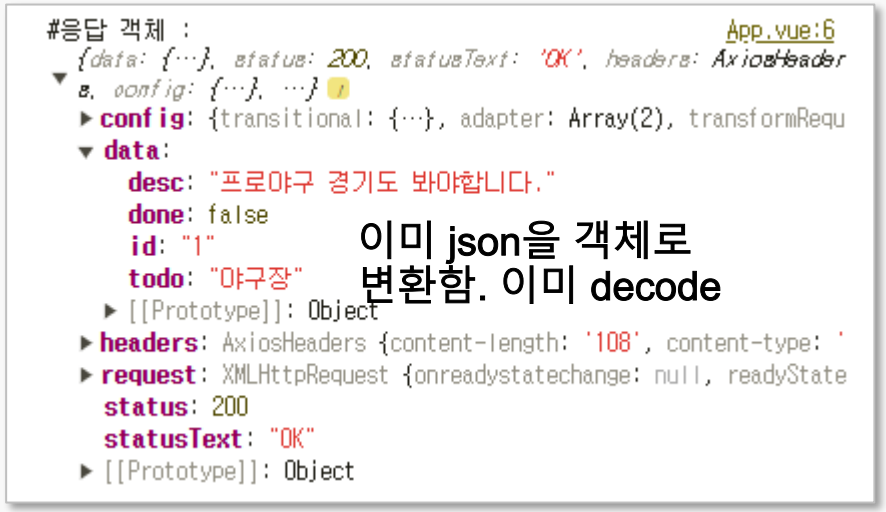
```
<template>
  <div>
    <h2>콘솔을 확인합니다.</h2>
  </div>
</template>

<script setup>
import axios from 'axios';

const requestAPI = () => {
    const url = 'http://localhost:3000/todos/1';
    axios.get(url).then((response) => {
        console.log('# 응답객체 : ', response);
    });
};
requestAPI();
</script>
```

모듈로써 바로 임폽
전역객체다.
모든 모듈에서 하나의
axios객체를 공유하여
사용하는 것이다.

axios.메서드(첫인자는URL, 두번째
부터는 메서드 종류마다 달라)
=>promise 반환

결과로
response객체

비동기함수는 탑레벨에서
호출하면 안되죠??
()=>{ 안에서 }



이미 json을 객체로
변환함. 이미 decode

남에 서버를 쓸때 생기는 문제점 cross origin

crossorigin pdf

vite.config.js 변경

```
import { fileURLToPath, URL } from 'node:url'
import { defineConfig } from 'vite'
import vue from '@vitejs/plugin-vue'

export default defineConfig({
  plugins: [vue()],
  resolve: {
    alias: {
      '@': fileURLToPath(new URL('./src', import.meta.url))
    }
  },
  server: {
    proxy: {
      '/api': {
        target: 'http://localhost:3000',
        changeOrigin: true,
        rewrite: (path) => path.replace(/^\/api/, ''),
      },
    },
  },
})
```

api로 시작하는 요청이면~

JSON-server로~ 해라.

경로 변환

요런 형식
빈문자열로

- 최초 요청 경로: /api/todolist/1
- 타깃: http://localhost:3000
- 최종 전달 경로: http://localhost:3000/todolist/1

src/App.vue

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

<script setup>
import axios from 'axios'

const requestAPI = () => {
    // const url = "http://localhost:3000/todos/1";
    const url = "/api/todos/1";
    axios.get(url).then((response) => {
        console.log("# 응답객체 : ", response);
    });
};

requestAPI();
</script>
```

나의 임시 개발 서버 경로로
하지만 프록시가
json-server 경로로 바꾸조?

Promise와 async-await

axios의 비동기 처리

- 함수의 마지막 인자에 콜백 함수가 있는 경우
  - 작업 완료시 콜백 함수 호출
- 콜백 함수 인자가 없는 경우
  - promise 객체를 리턴

src/App.vue

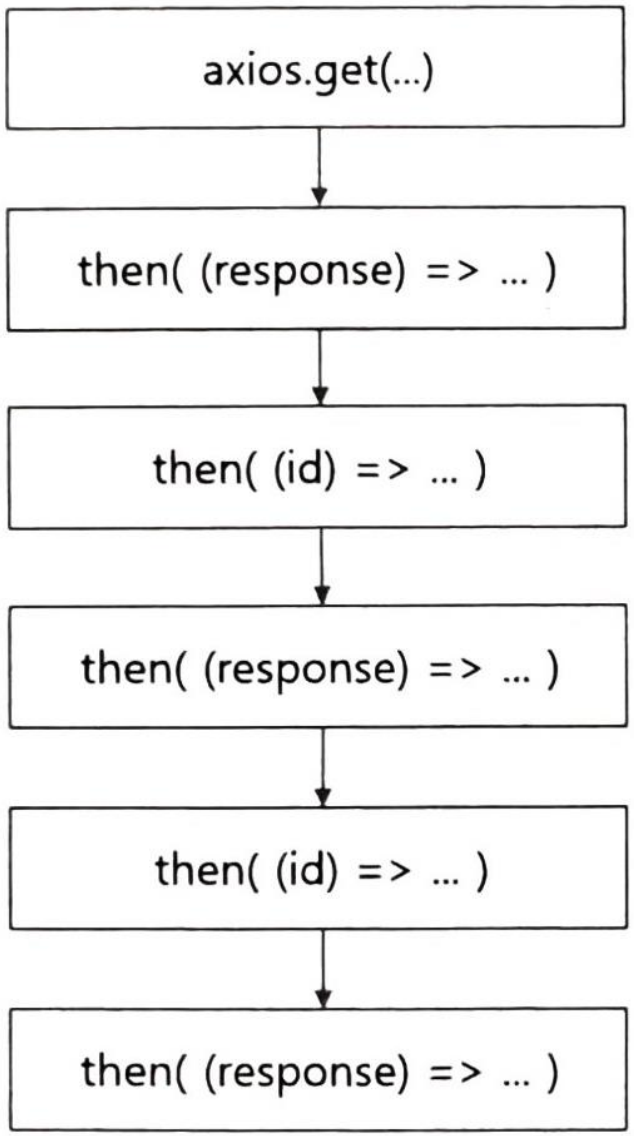
```
<template>
  <div>
    <h2>콘솔을 확인합니다.</h2>
  </div>
</template>

<script setup>
import axios from 'axios';

const listUrl = "/api/todos";
const todoUrlPrefix = "/api/todos/";

//4건의 목록을 조회한 후 첫번째, 두번째 할일을 순차적으로 조회합니다.
const requestAPI = () => {
  let todoList = [];
  axios
    .get(listUrl)
```

src/App.vue

```
    .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>
```

너무 복잡해

async await 써



TodoList 요청

받아온 TodoList 출력 후 첫번째 Todo의 id 리턴

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

응답된 첫번째 Todo 출력하고 두번째 Todo의 id 리턴

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

응답된 두번째 Todo 출력하고 종료

src/App2.vue 추가

```
…

//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();
</script>
```

간단하게 분해구조 할당 방식 이용
const { data } = await axios.get(~url);//이미json파싱됨

axios.get() 메서드 자동으로 url인코딩을 해준다

GET 요청 처리

[사용방법]
// url: 요청하는 백엔드 API의 URL을 지정합니다.
// config: 요청시에 지정할 설정값들 입니다.
// 요청 후에는 Promise를 리턴하며 처리가 완료된 후에는 response 객체를 응답받습니다.
axios.get(url, config)

```
config
{
  header : ~~,
  params : { 쿼리스트링내용 }
}
```

axios.get() 메서드

GET 요청 처리

[사용방법: Promise]

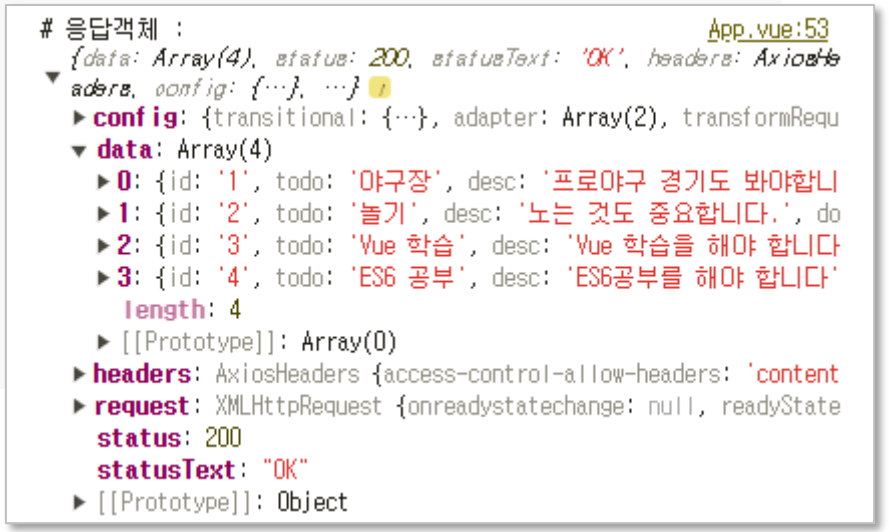
```
const requestAPI = async () => {
  const url = '/api/todos';
  axios.get(url)
    .then(response => {
      console.log('# 응답객체: ', response);
    });
};

requestAPI();
```

[사용방법: async/await]

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

src/App3.vue 추가

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

<script setup>
import axios from "axios";

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

requestAPI();
</script>
```



axios.response 객체

속성	설명
data	수신된 응답 데이터
config	요청시에 사용된 config 옵션
headers	백엔드 API 서버가 응답할 때 사용된 응답 HTTP 헤더
request	서버와의 통신에 사용된 XMLHttpRequest 객체의 정보
status	서버가 응답한 HTTP 상태 코드
statusText	서버의 HTTP 상태를 나타내는 문자열 정보

요청의 헤더 값 지정하기

timeout, Authorization 헤더 값 지정

```
axios.get(url, {
  timeout: 2000,
  headers: { Authorization: "Bearer xxxxxxx" }
});
```

jwt 연동시 jwt문자열을 넣어야해
authorization항목에

axios.post() 메서드

POST 요청 처리

- 데이터를 서버로 전송하여 서버에서 새로운 데이터 항목 생성(create)

```
// url, config는 axios.get()과 동일합니다.
// data는 POST 요청의 HTTP Content Body로 전송할 데이터입니다.
axios.post(url, data, config)
```

data
body에 들어갈 json 형식으로 써
물론 변환 안해도 돼 얘가 해줌

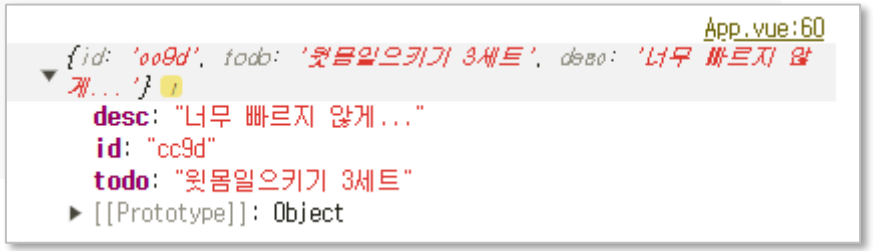
config 수정은 bin데이터를
보낼때 수정.

src/App4.vue 추가

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

<script setup>
import axios from "axios";

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

get은 에러 상황이 별로 없지만
나머지 메소드 종류는 에러처리를 해야한다 try catch

인증되지 않은 사람이 post하면
보안 에러나는 경우



기타 axios 함수

- axios.get(url, config)
- axios.post(url, data, config)
- axios.put(url, data, config)
- axios.delete(url, config)

axios 기본 설정 변경

- config 값을 전달하지 않으면 기본값이 사용됨

- 기본값의 변경 전역객체 axios 설정

```
axios.defaults.baseURL = '/api/todos';
axios.defaults.headers.common['Authorization'] = JWT;
axios.defaults.timeout = 2000;
```

하지만 특정한 곳에서만 특정 axios가 필요하다면?

axios.create 함수를 통해서
다른 axios객체를 생성해 활용할 수 있다.

에러 처리

src/App5.vue

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

<script setup>
import axios from "axios";

const requestAPI = async () => {
  const url = "/api/todos";
  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>
```

에러 코드

400번대에러
요청 실수 - 거의 개발자 실수

다 잡은 후 요청실수 없어야함.
로그인 안하고 요청할때 빼고
권한이 없거나

배포후

500번대에러 - 서버 버그문제
요청은 정상적.

개발자용이겠죠?

src/App6.vue

https://jsonplaceholder.typicode.com/
https://fakestoreapi.com/
https://contactsvc.bmaster.kro.kr/

```
…

<script setup>
import axios from "axios";

const requestAPI = async () => {
  const url = "/api/todos2";
  axios
    .get(url, { timeout: 900 })
    .then((response) => {
      console.log("# 응답객체 : ", response);
    })
    .catch((e) => {
      console.log('에러================');
      console.log(e);
      if (e instanceof Error) console.log(e.message);
      else console.log(e);
    });
};

requestAPI();
</script>
```

실제 배포에선 로그안찍고 다른 페이지로 이동하겠지 필요하면
라우터가 이동을 담당할 수 있음