KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

axios를 이용한 HTTP 통신

[KB] IT's Your Life

KB 국민은행

1 REST API란

REST(Representational State Transfer)

- 정보를 자원으로 간주
- 각 자원에게 URI를 배정
- 자원에 대한 조작은 HTTP 메서드로 표현

→ URI와 HTTP 메소드를 이용해 객체화된 서비스에 접근하는 것

- 예 : 게시판

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

1 REST API란

REST(Representational State Transfer)

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 내용은 BODY에 JSON으로 지정
/todos/{id}	DELETE	id의 TODO 삭제

REST 구성

- 자원(RESOURCE) – URI
- 행위(Verb) – HTTP METHOD
- 표현(Representations) – 일반적으로 JSON 사용

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

프로젝트 생성

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

json-server

전역 패키지로 설치

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 }
  ]
}
```

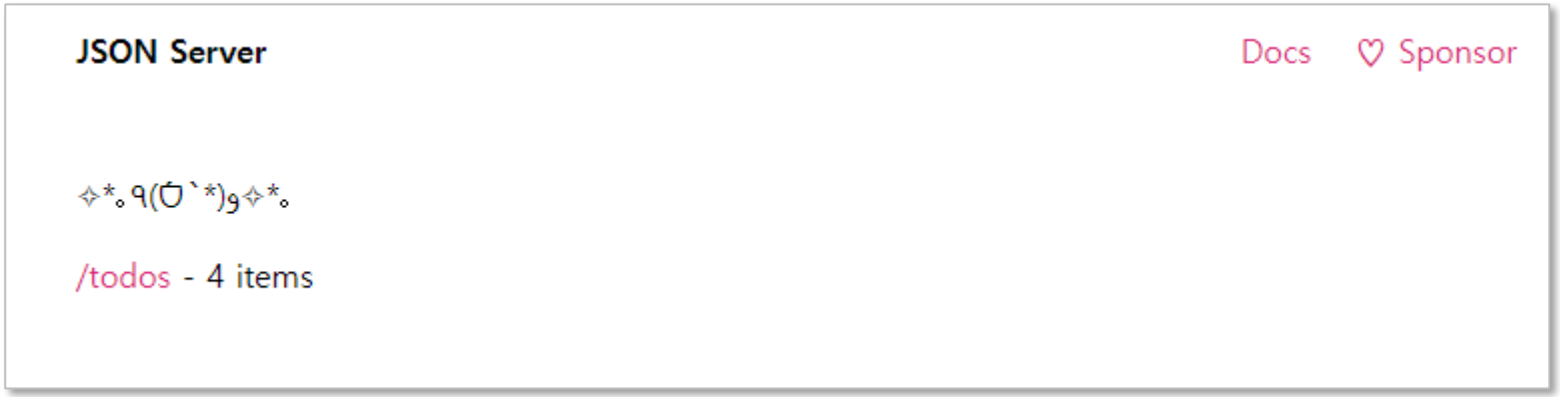
기동

> json-server db.json 또는

> npx json-server db.json

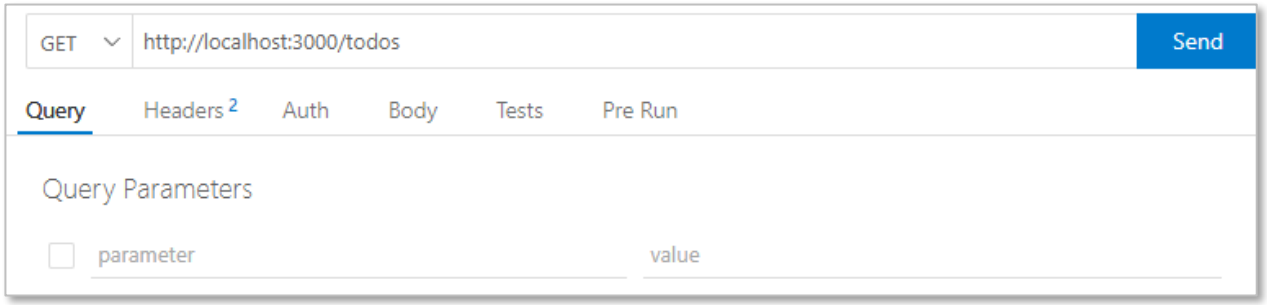
기동 확인

- http://localhost:3000/



Thunder Client에서 확인

- GET http://localhost:3000/todos



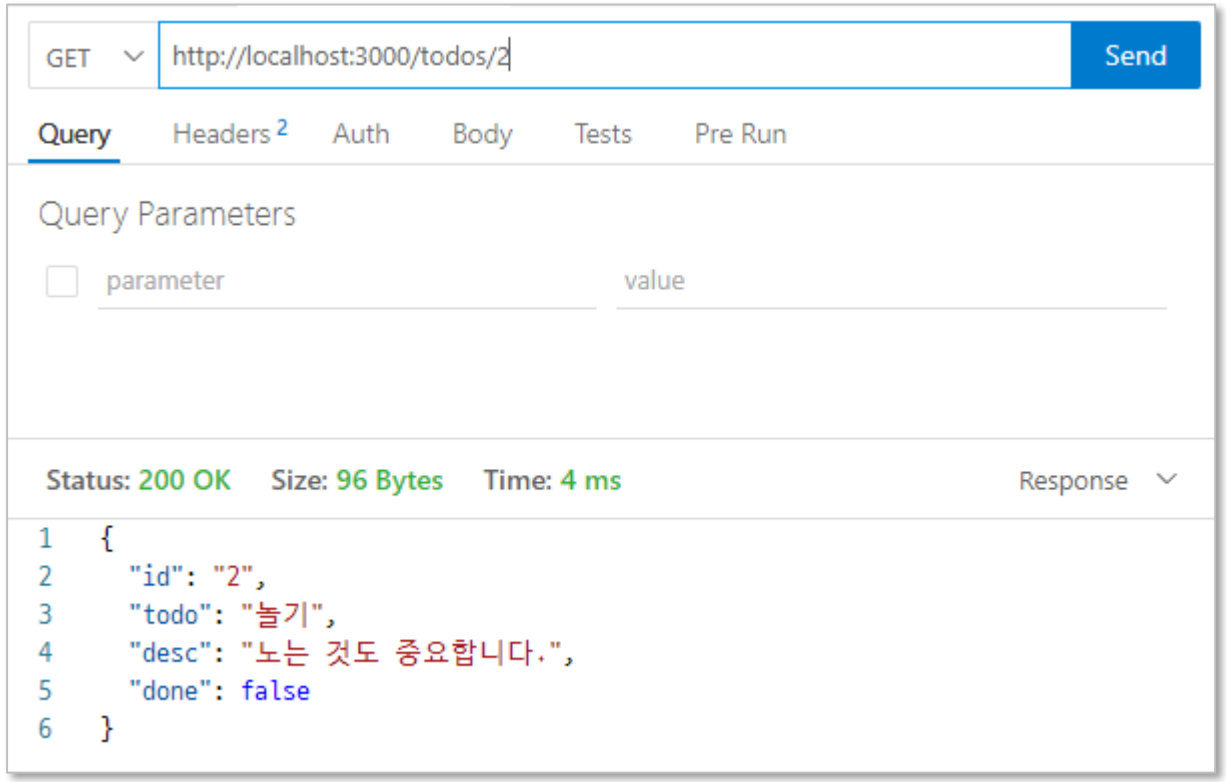
Status: 200 OK Size: 461 Bytes Time: 36 ms

Response Headers 8 Cookies Results Docs


```
[
  {
    "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
  }
]
```

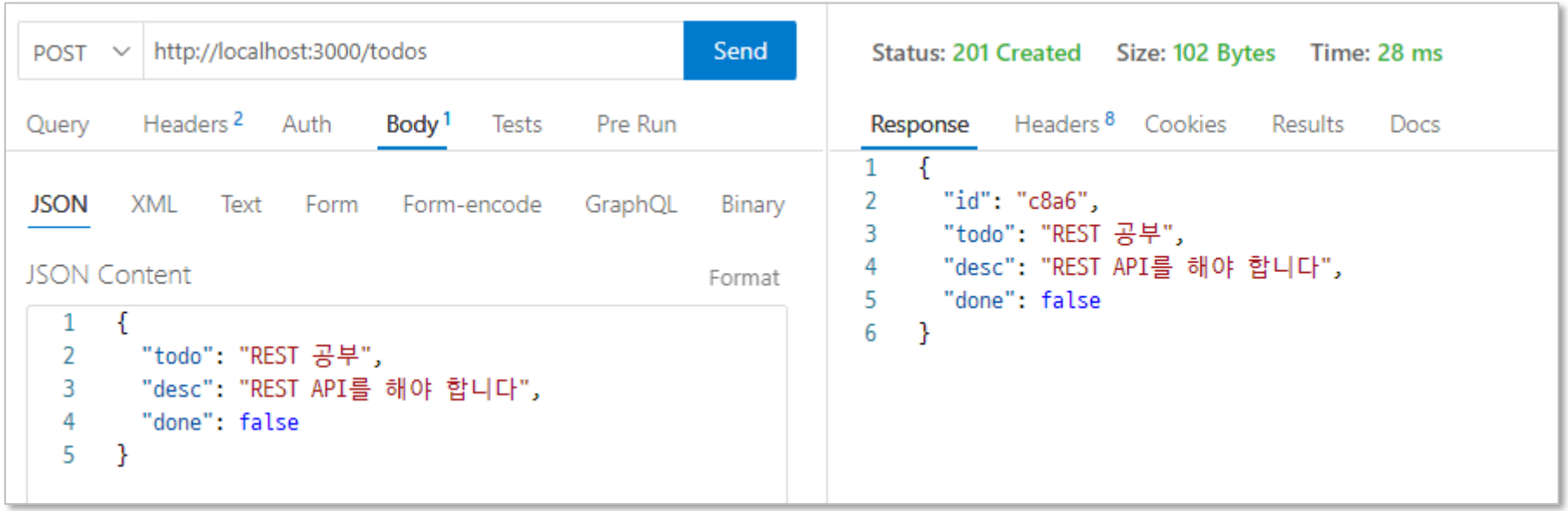
Thunder Client에서 확인

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



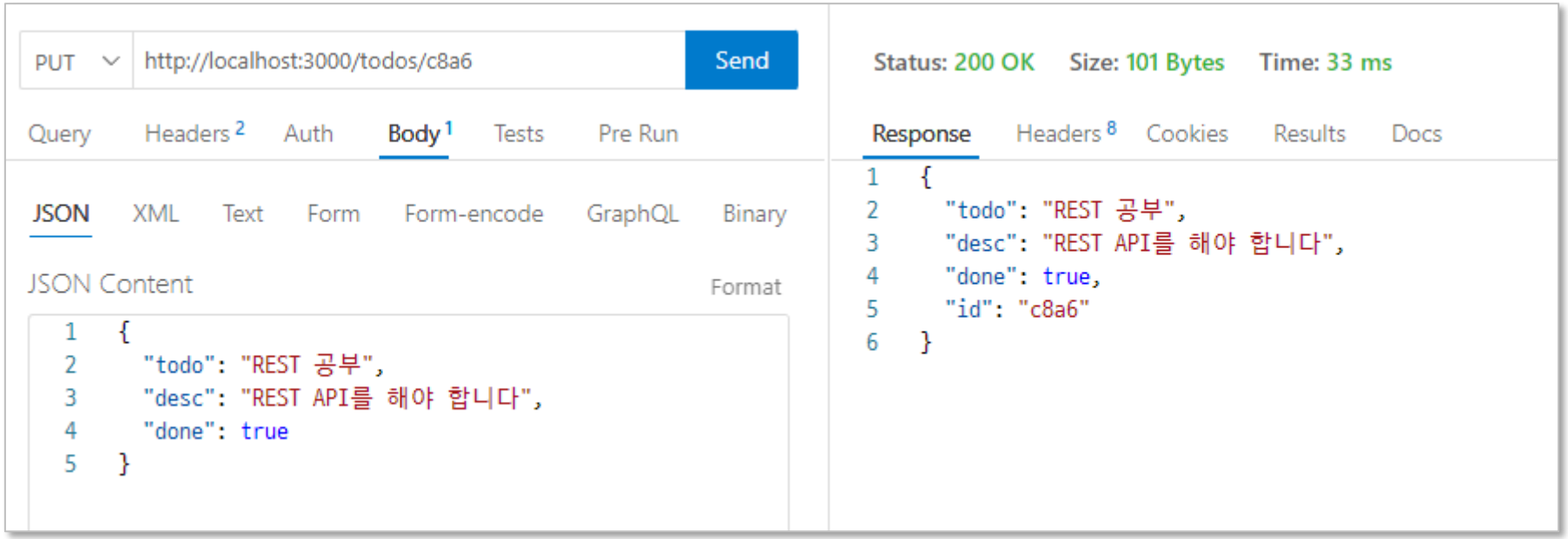
Todo 추가하기

- POST http://localhost:3000/todos



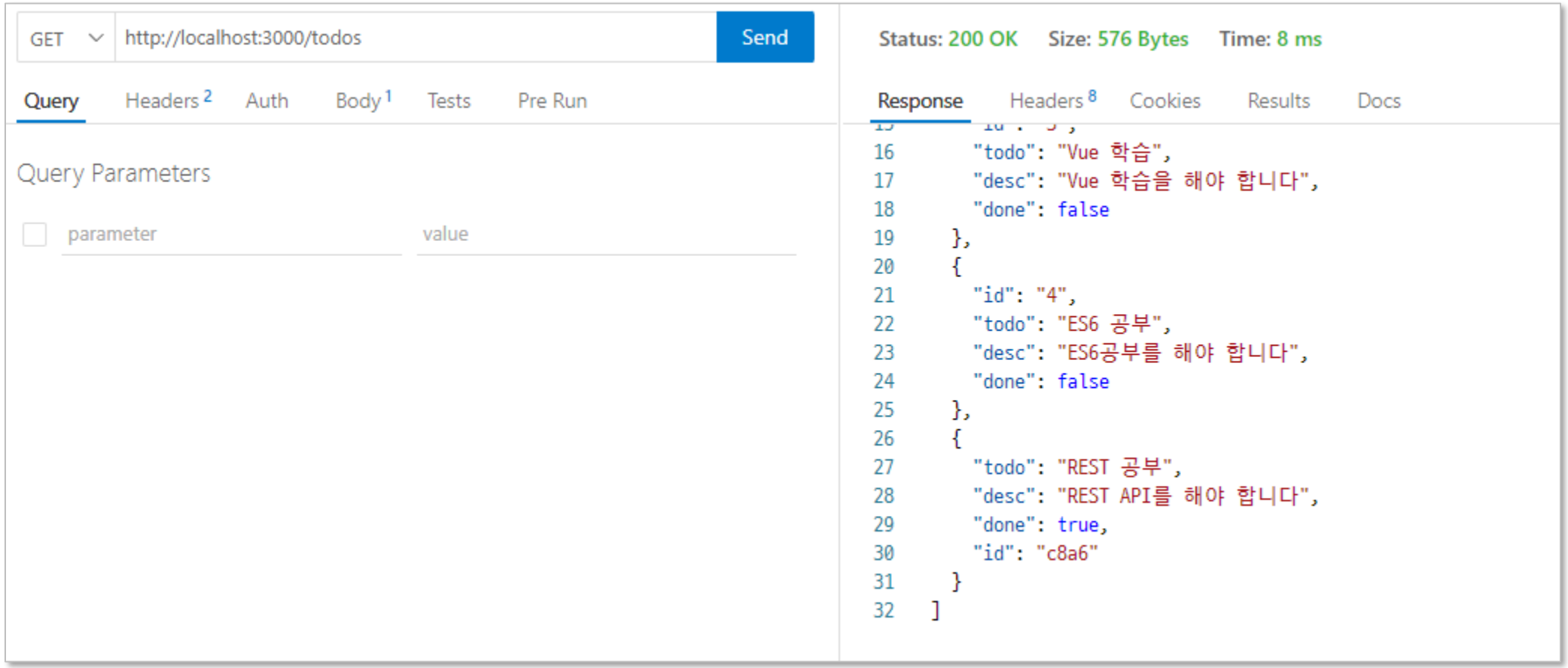
Todo 수정하기

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



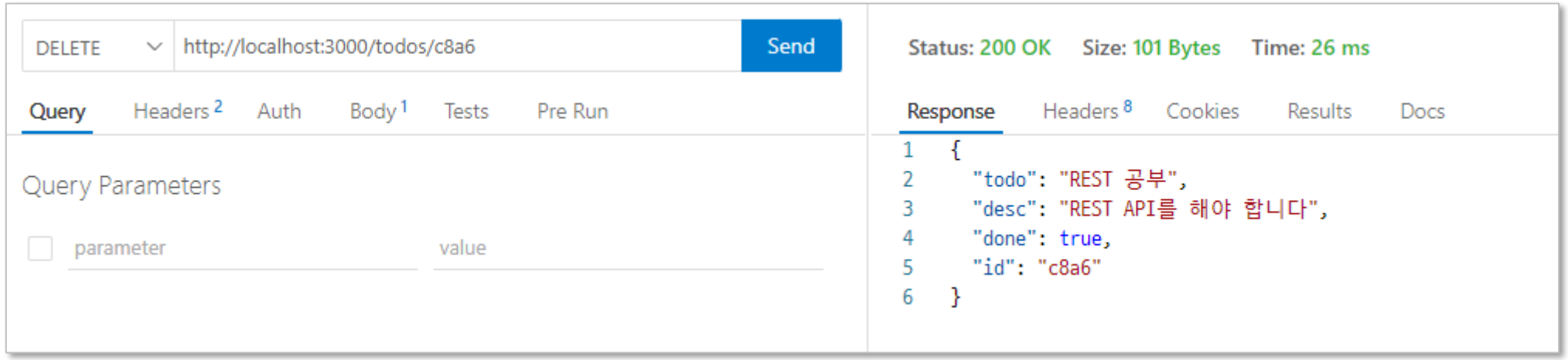
변경사항 확인하기

- GET http://localhost:3000/todos



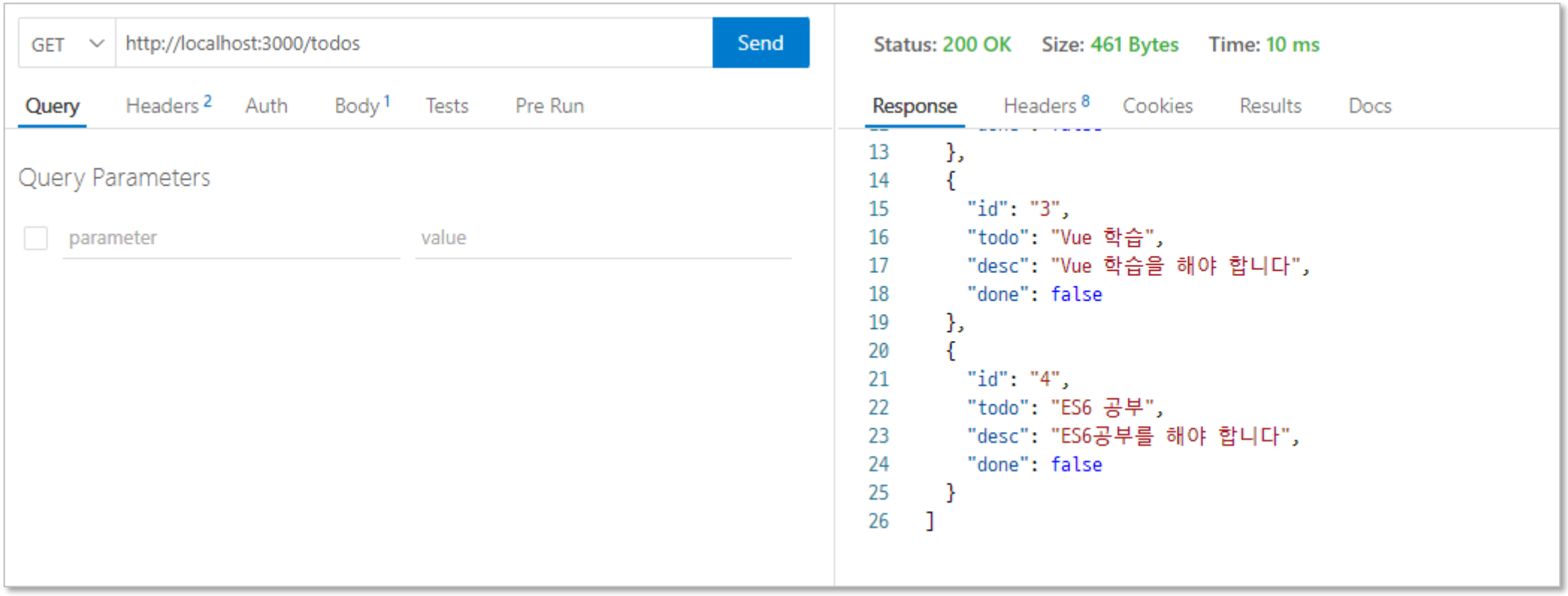
Todo 삭제하기

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



변경사항 확인하기

- GET http://localhost:3000/todos



axios

- HTTP 기반 통신을 지원하는 자바스크립트 라이브러리
- axios와 fetch와 비교

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)

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>
```

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/, ''),
      },
    },
  },
})
```

- 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>
```

Promise와 async-await

axios의 비동기 처리

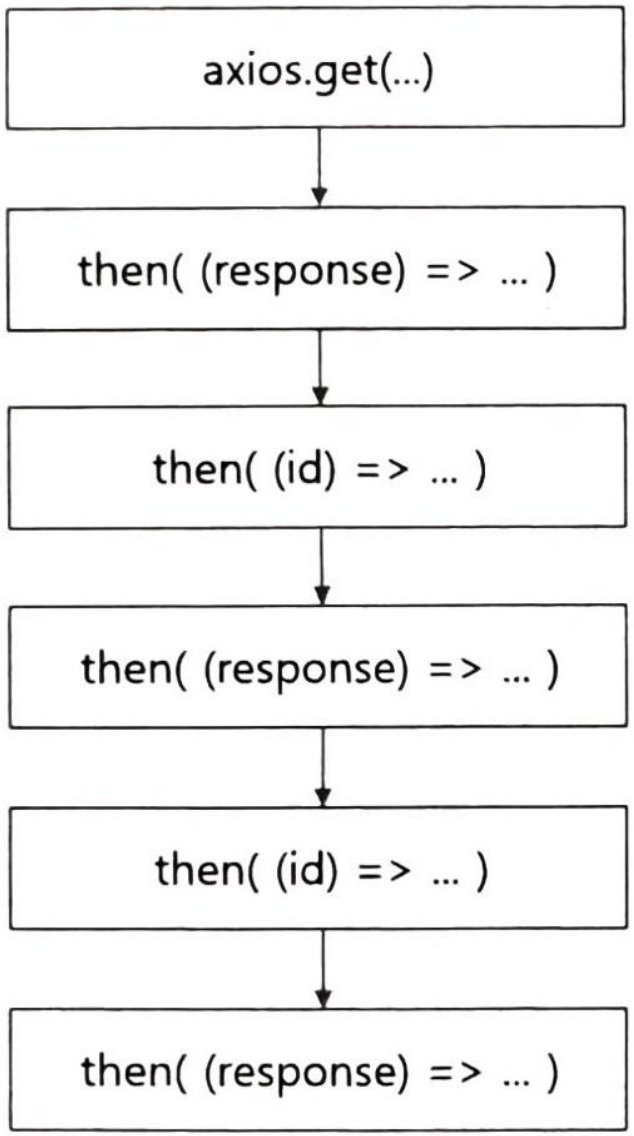
- 함수의 마지막 인자에 콜백 함수가 있는 경우
  - 작업 완료시 콜백 함수 호출
- 콜백 함수 인자가 없는 경우
  - promise 객체를 리턴

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>
```



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>
```

axios.get() 메서드

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)

axios.get() 메서드

GET 요청 처리

[사용방법: Promise]

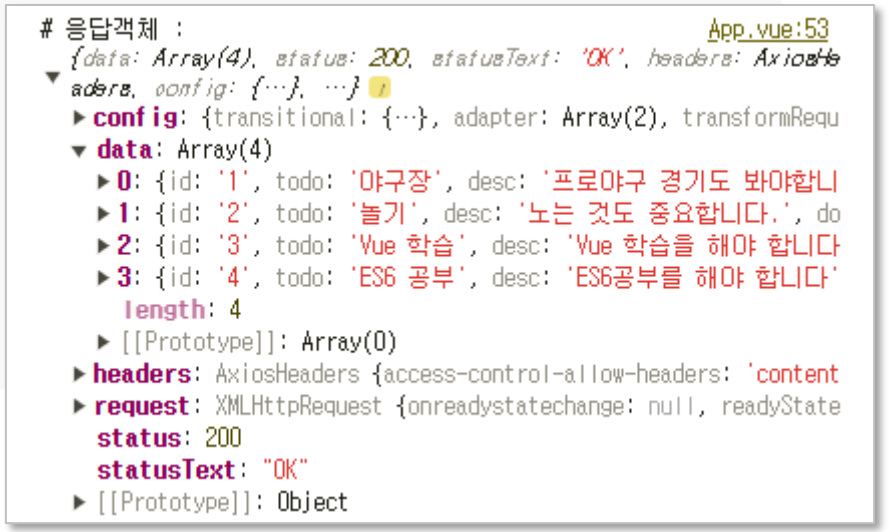
```
const requestAPI = async () => {
  const url = '/api/todos';
  axios.get(url)
    .then(response => {
      console.log('# 응답객체: ', response);
    });
};

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" }
});
```

axios.post() 메서드

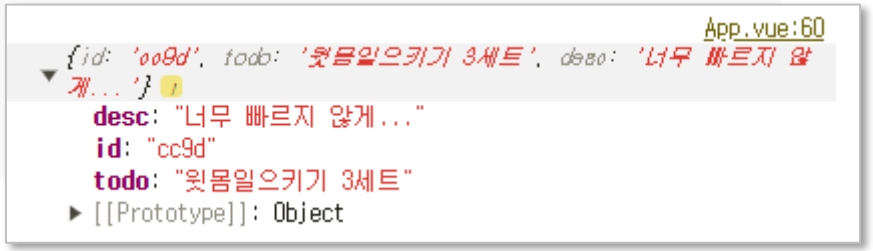
POST 요청 처리

- 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)
```

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>
```



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

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

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>
```

src/App6.vue

```
…

<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>
```