REST API

2020-05-15

이승진

**학습목표**

REST API 이해

asios 사용법

**목차**

[1. REST API 2](#_Toc40671991)

[1) REST API는 무엇인가? 2](#_Toc40671992)

[2) 프론트엔드와 백엔드 2](#_Toc40671993)

[3) request method 3](#_Toc40671994)

[2. JSON 형식 4](#_Toc40671995)

[1) JSON 형시의 예 4](#_Toc40671996)

[2) serialization / unserialization 4](#_Toc40671997)

[3) serialization / unserialization 활용 5](#_Toc40671998)

[3. firebase realtime database REST API 6](#_Toc40671999)

[4. axios 라이브러리 7](#_Toc40672000)

[1) 사용법 7](#_Toc40672001)

[5. todo 앱5 9](#_Toc40672002)

[1) todo51.html 9](#_Toc40672003)

[2) todo52.html 10](#_Toc40672004)

[3) todo53.html 12](#_Toc40672005)

[4) todo54.html 14](#_Toc40672006)

[5) todo55.html 16](#_Toc40672007)

[6. 과제 18](#_Toc40672008)

[1) todo17.html 18](#_Toc40672009)

# REST API

## REST API는 무엇인가?

모바일 앱이나, javascript로 구현된 웹 앱에는 데이터를 저장할 서버가 필요하다.

인터넷을 통해서 앱과 통신할 수 있는 서버를 개발해야 한다.

이런 서버를 처음부터 새로 만드는 것은 쉽지 않은 일이다.

네트웍 프로그래밍이 쉽지 않은 기술이기 때문이다.

웹 서버 프로그래밍 기술을 이용하면, 네트웍 프로그래밍 없이,

아주 편하게 서버를 개발할 수 있다.

일반적인 웹 서버 프로그래은, 웹브라우저창에 그려질 HTML 태그들을 출력한다.

모바일 앱이나, javascvript 앱을 위한 웹 서버 프로그램은,

앱이 사용할 데이터를 출력해야 한다.

보통 JSON 형식이나 XML 형식으로 데이터를 출력한다.

위와 같은 목적으로 구현된 웹 서버 프로그램을 RESTful Service 이라고 한다.

즉 웹브라우저를 위한 웹 서버 프로그램이 아니고,

클라이언트 앱을 위한 웹 서버 프로그램이 RESTful Service 이다.

RESTful Service가 제공하는 URL 목록이 REST API 이다.

클라이언트 앱은 REST API URL을 서버에 요청해서, 서버와 데이터를 주고 받는다.

클라이언트 앱과 RESTful Service 서버 사이의 통신은 당연히 HTTP, HTTPS 프로토콜을 사용한다.

## 프론트엔드와 백엔드

javascript 언어로 개발된 클라이언트 앱이 프론트엔드이다.

클라이언트 앱에 데이터를 제공하기 위한 서버가 백엔드이다.

백엔드는 REST API를 클라이언트 앱에 제공해야 한다.

클라이언트 앱은 백엔드의 REST API URL을 요청해서, 서버와 데이터를 주고 받는다.

javascript 개발자는 node.js 기술로 백엔드를 개발한다.

java 개발자는 spring 기술로 백엔드를 개발한다.

클라이언트 앱과 서버 사이에 전달되는 데이터는 대부분 JSON 형식이다.

XML 형식을 사용하기도 한다.

## request method

일반적인 웹 서버 프로그래밍에서, 웹브라우저가 웹서버를 호출하는 요청 방식(request method)은

GET 방식과 POST 방식이 있다.

REST API 에서는 GET, POST, PUT, DELETE 요청 방식을 사용한다.

각각의 request method는 다음과 같은 경우에 사용한다.

|  |  |
| --- | --- |
| request method | 요청 |
| GET | 데이터 조회 요청 |
| POST | 신규 데이터 등록 요청 |
| PUT | 데이터 수정 요청 |
| DELETE | 데이터 삭제 요청 |

# JSON 형식

(Javascript Object Notation)

javascript 언어의 객체와 배열 문법으로 데이터를 표현한 것을 말한다.

## JSON 형시의 예

### 객체

{ "name": "홍길동", "age": 16 }

### 배열

[{ "name": "홍길동", "age": 16 }, { "name": "임꺽정", "age": 19 }];

javascript 문법에서 객체의 속성명을 따옴표로 묶어도 되고, 묶지 않아도 되지만,

JSON 형식에서는 반드시 속성명을 따옴표로 묶어야 한다.

## serialization / unserialization

객체나 배열을 JSON 형식의 문자열로 변환하는 것을 serialization 이라고 한다.

JSON 형식의 문자열로부터 객체나 배열을 생성하는 것을 unserialization 이라고 한다.

클라이언트 앱에서 서버로 데이터를 보낼 때, JSON 형식의 문자열로 serialize 해서 보내야 한다.

서버에서 받아온 JSON 형식의 문자열을 unserialize 해서 데이터 객체를 만들어야 한다.

### serialize

|  |
| --- |
| let person = { name: "홍길동", age: 16 };  let s = JSON.stringify(person);  console.log(s); |

JSON.stringify(객체)

객체를 JSON 형식의 문자열로 변환해서 리턴한다.

### unserialize

|  |
| --- |
| let s = '{"name":"홍길동","age":16}';  let person = JSON.parse(s);  console.log(person); |

JSON.parse(문자열)

JSON 형식의 문자열로부터 객체를 생성해서 리턴한다.

## serialization / unserialization 활용

### deep copy

serialize와 unserialize를 이용하여 deep copy를 구현할 수 있다.

|  |
| --- |
| let a1 = [{ "name": "홍길동", "age": 16 }, { "name": "임꺽정", "age": 19 }];  let s = JSON.stringify(a1);  let a2 = JSON.parse(s);  console.log(a2); |

a1 배열을 deep copy 하여 a2 배열을 생성하였다.

a1 배열과 그 내부 데이터들을 전부 JSON 형식의 문자열로 만들고,

그 JSON 형식의 문자열로부터 새로 a2 배열을 생성하였다. => deep copy

### deep compare

serialize를 이용하여 deep equals를 쉽게 구현할 수 있다.

|  |
| --- |
| let a1 = [{ "name": "홍길동", "age": 16 }, { "name": "임꺽정", "age": 19 }];  let s = JSON.stringify(a1);  let a2 = JSON.parse(s);  console.log(a2);  if (JSON.stringify(a1) == JSON.stringify(a2))    console.log("equals"); |

a1 배열과 a2 배열을 그 내부 데이터들을 전부 JSON 형식의 문자열로 만들고,

이 두 문자열을 비교한다. => deep compare

조심할 부분, 객체의 속성이 만들어진 순서가 다르면, 직렬화된 문자열의 내용도 다르다.

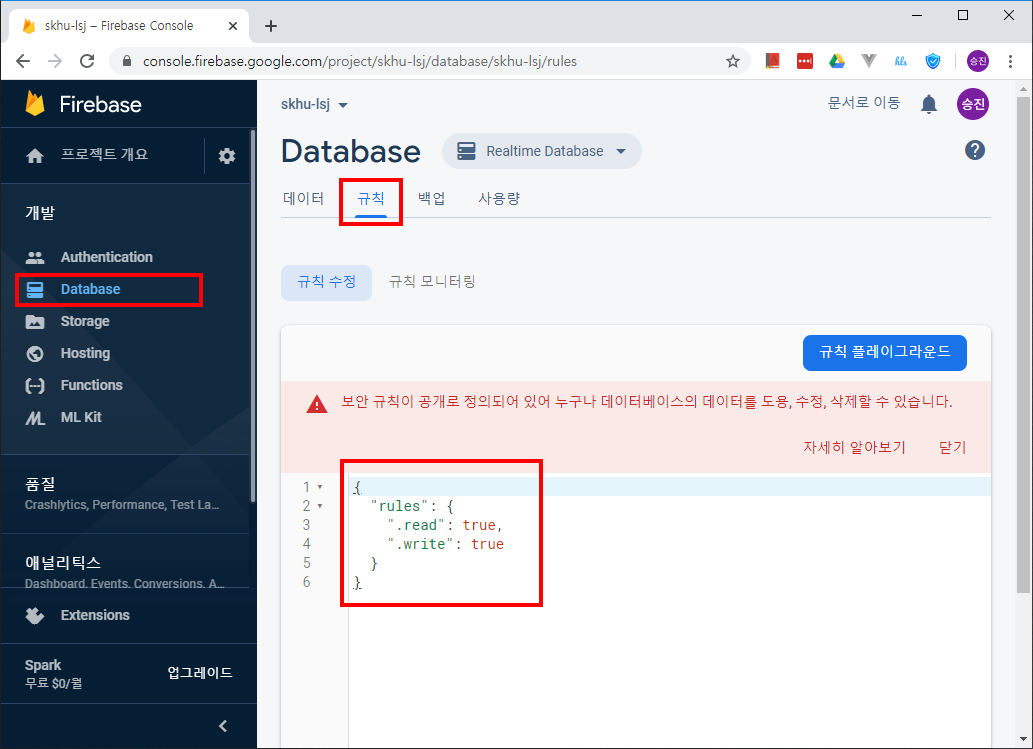
객체의 속성의 순서가 다를뿐 속성의 값이 같은 경우에, 위 방법으로 비교하면 false가 된다.

# firebase realtime database REST API

REST API 예제를 구현하려면, RESTful Service를 제공하는 백엔드 서버가 필요하다.

firebase realtime database는 REST API 기능을 제공한다.

이 기능을 이용하면, 우리는 백엔드 서버를 개발하지 않고도 REST API 예제를 구현해 볼 수 있다.



이 REST API 기능을 사용할 수 있으려면,

익명의(anonymous) 사용자도 DB를 읽고 쓸 수 있도록 위와 같이 설정해야 한다.

# axios 라이브러리

javascript 클라이언트 앱에서 백엔드 서버를 호출하기 위해 사용하는 라이브러리 중 하나가 axios 이다.

## 사용법

axios 라이브러리를 이용하여 firebase realtime database REST API를 사용하는 방법은 다음과 같다.

### html 문서에 axios 라이브러리 포함하기

|  |
| --- |
| <script src="https://unpkg.com/axios/dist/axios.min.js"></script> |

html 문서에서 axios 라이브러리를 사용하려면, 위 태그를 header 부분에 넣어야 한다.

위 태그에 의해서, axios.min.js 파일이 서버로부터 웹브라우저로 전송되어 실행된다.

### 데이터 조회 요청

|  |
| --- |
| axios.get('https://skhu-lsj.firebaseio.com/todo4.json')       .then(response => app.todoMap = response.data); |

get 메소드의 파라미터는 요청할 URL 이고

then 메소드의 파라미터는 화살표 함수이다.

'https://skhu-lsj.firebaseio.com/todo4.json' URL을 get 방식으로 요청한다.

skhu-lsj 부분은 firebase 서버 프로젝트 ID 이다.

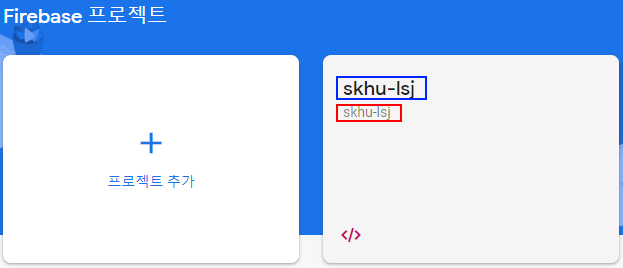
todo4 부분은 firebase realtime database 항목 이름이다.

firebase DB의 todo4 데이터 항목을 조회한다.

서버가 전송한 데이터가 도착하면, 화살표 함수가 호출된다.

서버가 전송한 데이터는 response.data 이다.

### firebase 서버 프로젝트 ID



firebase 서버 프로젝트 ID는 <https://console.firebase.google.com/> 에서 확인할 수 있다.

빨간색 사각형 부분이 firebase 프로젝트 ID 이다.

파란색 사각형 부분은 firebase 프로젝트 이름이다.

### firebase realtime database REST API URL

'https://프로젝트ID.firebaseio.com/데이터\_항목\_경로.json'

자신의 firebase 프로젝트ID를 확인해서 URL을 수정하자.

### 신규 데이터 등록 요청

|  |
| --- |
| axios.post('https://skhu-lsj.firebaseio.com/todo4.json', this.todo)       .then(response => this.reload()); |

post 메소드의 첫째 파라미터는 요청할 URL 이고, 둘째 파라미터는 등록할 데이터이다.

then 메소드의 파라미터는 화살표 함수이다.

'https://skhu-lsj.firebaseio.com/todo4.json' URL을 post 방식으로 요청한다.

이 때 this.todo 데이터가 request parameter 데이터가 되어 서버에 전송된다.

firebase DB의 todo4 데이터 항목 아래에 신규 데이터가 등록된다.

서버가 작업을 마치고, 작업 결과 데이터를 전송하면, 화살표 함수가 호출된다.

서버가 전송한 작업 결과 데이터는 response.data 이다.

### 데이터 수정 요청

|  |
| --- |
| axios.put('https://skhu-lsj.firebaseio.com/todo4/데이터\_키.json', this.todo)       .then(response => this.reload()); |

put 메소드의 첫째 파라미터는 요청할 URL 이고, 둘째 파라미터는 저장할 데이터이다.

then 메소드의 파라미터는 화살표 함수이다.

'https://skhu-lsj.firebaseio.com/todo4/데이터\_키.json' URL을 post 방식으로 요청한다.

이 때 this.todo 데이터가 request parameter 데이터가 되어 서버에 전송된다.

firebase DB의 todo4 데이터 항목 아래의 '데이터\_키' 키(key)에 해당하는 데이터 항목을 업데이트 한다.

서버가 작업을 마치고, 작업 결과 데이터를 전송하면, 화살표 함수가 호출된다.

서버가 전송한 작업 결과 데이터는 response.data 이다.

### 데이터 삭제 요청

|  |
| --- |
| axios.delete('https://skhu-lsj.firebaseio.com/todo4/데이터\_키.json')       .then(response => this.reload()); |

delete 메소드의 첫째 파라미터는 요청할 URL 이다.

then 메소드의 파라미터는 화살표 함수이다.

'https://skhu-lsj.firebaseio.com/todo4/데이터\_키.json' URL을 delete 방식으로 요청한다.

firebase DB의 todo4 데이터 항목 아래의 '데이터\_키' 키(key)에 해당하는 데이터 항목을 삭제한다.

서버가 삭제 작업을 마치고, 작업 결과 데이터를 전송하면, 화살표 함수가 호출된다.

서버가 전송한 작업 결과 데이터는 response.data 이다.

# todo 앱5

## todo51.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <script src="https://unpkg.com/axios/dist/axios.min.js"></script>    <script src="https://kit.fontawesome.com/68abb170e0.js" crossorigin="anonymous"></script>    <link rel="stylesheet" type="text/css" href="todo3.css">  </head>  <body>    <div id="app">      <div id="main">        <h1><i class="fas fa-clipboard-check"></i> 할일</h1>        <table>          <tr>            <td>제목</td>            <td>날짜</td>          </tr>          <tr v-for="(todo, key) in todoMap" v-bind:key="key">            <td>{{ todo.title }}</td>            <td>{{ todo.due\_date }}</td>          </tr>          <tr v-if="Object.keys(todoMap).length == 0">            <td colspan="2">등록된 자료가 없습니다.</td>          </tr>        </table>      </div>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          todoMap: {}        }      })      window.onload = function() {        axios.get('https://skhu-lsj.firebaseio.com/todo4.json')             .then(response => app.todoMap = response.data);      }     </script>  </body>  </html> |

firebase DB에 직업 접근하지 않기 때문에, firebase DB 관련 코드가 전부 제거되었다.

## todo52.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <script src="https://unpkg.com/axios/dist/axios.min.js"></script>    <script src="https://kit.fontawesome.com/68abb170e0.js" crossorigin="anonymous"></script>    <link rel="stylesheet" type="text/css" href="todo3.css">  </head>  <body>    <div id="app">      <div id="main">        <h1><i class="fas fa-clipboard-check"></i> 할일</h1>        <table>          <tr>            <td>제목</td>            <td>날짜</td>          </tr>          <tr v-for="(todo, key) in todoMap" v-bind:key="key">            <td>{{ todo.title }}</td>            <td>{{ todo.due\_date }}</td>          </tr>          <tr v-if="Object.keys(todoMap).length == 0">            <td colspan="2">등록된 자료가 없습니다.</td>          </tr>        </table>        <button type="button" v-on:click="create"><i class="fas fa-plus"></i> 추가</button>      </div>      <div id="backdrop" v-show="showDialog"></div>      <div id="dialog" v-show="showDialog">        <h1><i class="fas fa-edit"></i> 할일 {{ showDialog }}</h1>        <input type="text" v-model.trim="todo.title" placeholder="제목을 입력하세요"/>        <textarea v-model.trim="todo.body" placeholder="내용을 입력하세요"></textarea>        <div>          <label>날짜:</label>          <input type="date" v-model="todo.due\_date" />          <button type="button" v-on:click="save"><i class="fas fa-check"></i> 저장</button>          <button type="button" v-on:click="showDialog = false"><i class="fas fa-ban"></i> 취소</button>        </div>      </div>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          todoMap: {},          todo: {},          showDialog: false        },        methods: {          create: function() {            this.showDialog = "등록";          ;            this.todo = {};          },          save: function() {            axios.post('https://skhu-lsj.firebaseio.com/todo4.json', this.todo);            this.showDialog = false;          }        }      })      window.onload = function() {        axios.get('https://skhu-lsj.firebaseio.com/todo4.json')             .then(response => app.todoMap = response.data);      }     </script>  </body>  </html> |

새 할일을 입력하고 저장 버튼을 눌렀을 때, 할일 목록에 변화가 없다.

firebase DB에 새 할일이 저장되었지만, 변경된 목록을 다시 조회하지 않기 때문이다.

웹브라우저에서 새로 고침을 하면, 방금 등록한 할일이 보인다.

firebase realtime databas에 직접 연결하는 경우에는,

서버에 데이터 항목의 값이 변경되면, 그 변경된 값이 즉시 다시 전송되는 기능이 있었다.

그렇지만 REST API 에서는 그런 기능이 없다.

변경된 데이터를 화면에 출력하려면, 서버에서 다시 조회해야 한다.

## todo53.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <script src="https://unpkg.com/axios/dist/axios.min.js"></script>    <script src="https://kit.fontawesome.com/68abb170e0.js" crossorigin="anonymous"></script>    <link rel="stylesheet" type="text/css" href="todo3.css">  </head>  <body>    <div id="app">      <div id="main">        <h1><i class="fas fa-clipboard-check"></i> 할일</h1>        <table>          <tr>            <td>제목</td>            <td>날짜</td>          </tr>          <tr v-for="(todo, key) in todoMap" v-bind:key="key">            <td>{{ todo.title }}</td>            <td>{{ todo.due\_date }}</td>          </tr>          <tr v-if="Object.keys(todoMap).length == 0">            <td colspan="2">등록된 자료가 없습니다.</td>          </tr>        </table>        <button type="button" v-on:click="create"><i class="fas fa-plus"></i> 추가</button>      </div>      <div id="backdrop" v-show="showDialog"></div>      <div id="dialog" v-show="showDialog">        <h1><i class="fas fa-edit"></i> 할일 {{ showDialog }}</h1>        <input type="text" v-model.trim="todo.title" placeholder="제목을 입력하세요"/>        <textarea v-model.trim="todo.body" placeholder="내용을 입력하세요"></textarea>        <div>          <label>날짜:</label>          <input type="date" v-model="todo.due\_date" />          <button type="button" v-on:click="save"><i class="fas fa-check"></i> 저장</button>          <button type="button" v-on:click="showDialog = false"><i class="fas fa-ban"></i> 취소</button>        </div>      </div>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          todoMap: {},          todo: {},          showDialog: false        },        methods: {          create: function() {            this.showDialog = "등록";          ;            this.todo = {};          },          reload: function() {            axios.get('https://skhu-lsj.firebaseio.com/todo4.json')                 .then(response => this.todoMap = response.data);          },          save: function() {            axios.post('https://skhu-lsj.firebaseio.com/todo4.json', this.todo)                 .then(response => this.reload());            this.showDialog = false;          }        }      })      window.onload = function() {        app.reload();      }     </script>  </body>  </html> |

reload: function() {

  axios.get('https://skhu-lsj.firebaseio.com/todo4.json')

       .then(response => app.todoMap = response.data);

}

reload 메소드는 todo4 데이터 항목 조회를 서버에 요청하고,

그 요청 결과 데이터가 서버로부터 도착하면 그 데이터를 todoList에 채운다.

save: function() {

  axios.post('https://skhu-lsj.firebaseio.com/todo4.json', this.todo)

       .then(response => this.reload());

this.showDialog = false;

}

save 메소드는, this.todo 데이터를 todo4 데이터 항목 아래에 신규 등록하라고 서버에 요청하고,

그 요청 결과 데이터 서버로부터 도착하면, this.reload() 메소드를 호출한다.

reload() 메소드는 todo4 데이터 항목을 다시 조회한다.

window.onload = function() {

  app.reload();

}

웹브라우저 창의 내용이 전부 로드(load)되면 app.reload() 메소드를 호출한다.

reload() 메소드는 todo4 데이터 항목을 다시 조회한다.

새 할일을 등록한 후 저장 버튼을 누르면,

할일 목록이 다시 서버에서 조회되고, 방금 등록한 할일이 목록에 보인다.

## todo54.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <script src="https://unpkg.com/axios/dist/axios.min.js"></script>    <script src="https://kit.fontawesome.com/68abb170e0.js" crossorigin="anonymous"></script>    <link rel="stylesheet" type="text/css" href="todo3.css">  </head>  <body>    <div id="app">      <div id="main">        <h1><i class="fas fa-clipboard-check"></i> 할일</h1>        <table>          <tr>            <td>제목</td>            <td>날짜</td>          </tr>          <tr v-for="(todo, key) in todoMap" v-on:click="edit(key)" v-bind:key="key" class="clickable">            <td>{{ todo.title }}</td>            <td>{{ todo.due\_date }}</td>          </tr>          <tr v-if="Object.keys(todoMap).length == 0">            <td colspan="2">등록된 자료가 없습니다.</td>          </tr>        </table>        <button type="button" v-on:click="create"><i class="fas fa-plus"></i> 추가</button>      </div>      <div id="backdrop" v-show="showDialog"></div>      <div id="dialog" v-show="showDialog">        <h1><i class="fas fa-edit"></i> 할일 {{ showDialog }}</h1>        <input type="text" v-model.trim="todo.title" placeholder="제목을 입력하세요"/>        <textarea v-model.trim="todo.body" placeholder="내용을 입력하세요"></textarea>        <div>          <label>날짜:</label>          <input type="date" v-model="todo.due\_date" />          <button type="button" v-on:click="save"><i class="fas fa-check"></i> 저장</button>          <button type="button" v-on:click="showDialog = false"><i class="fas fa-ban"></i> 취소</button>        </div>      </div>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          todoMap: {},          todo: {},          showDialog: false,          selectedKey: null        },        methods: {          create: function() {            this.showDialog = "등록";          ;            this.todo = {};          },          reload: function() {            axios.get('https://skhu-lsj.firebaseio.com/todo4.json')                 .then(response => this.todoMap = response.data);          },          edit: function(key) {            this.showDialog = "수정";            this.todo = Object.assign({}, this.todoMap[key]);            this.selectedKey = key;          },          save: function() {            if (this.showDialog == "등록")              axios.post('https://skhu-lsj.firebaseio.com/todo4.json', this.todo)                   .then(response => this.reload());            else              axios.put('https://skhu-lsj.firebaseio.com/todo4/' + this.selectedKey + '.json', this.todo)                   .then(response => this.reload());            this.showDialog = false;          }        }      })      window.onload = function() {        app.reload();      }     </script>  </body>  </html> |

수정 기능 구현

## todo55.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <script src="https://unpkg.com/axios/dist/axios.min.js"></script>    <script src="https://kit.fontawesome.com/68abb170e0.js" crossorigin="anonymous"></script>    <link rel="stylesheet" type="text/css" href="todo3.css">  </head>  <body>    <div id="app">      <div id="main">        <h1><i class="fas fa-clipboard-check"></i> 할일</h1>        <table>          <tr>            <td>제목</td>            <td>날짜</td>          </tr>          <tr v-for="(todo, key) in todoMap" v-on:click="edit(key)" v-bind:key="key" class="clickable">            <td>{{ todo.title }}</td>            <td>{{ todo.due\_date }}</td>          </tr>          <tr v-if="Object.keys(todoMap).length == 0">            <td colspan="2">등록된 자료가 없습니다.</td>          </tr>        </table>        <button type="button" v-on:click="create"><i class="fas fa-plus"></i> 추가</button>      </div>      <div id="backdrop" v-show="showDialog"></div>      <div id="dialog" v-show="showDialog">        <h1><i class="fas fa-edit"></i> 할일 {{ showDialog }}</h1>        <input type="text" v-model.trim="todo.title" placeholder="제목을 입력하세요"/>        <textarea v-model.trim="todo.body" placeholder="내용을 입력하세요"></textarea>        <div>          <label>날짜:</label>          <input type="date" v-model="todo.due\_date" />          <button type="button" v-on:click="save"><i class="fas fa-check"></i> 저장</button>          <button type="button" v-on:click="remove"><i class="fas fa-trash-alt"></i> 삭제</button>          <button type="button" v-on:click="showDialog = false"><i class="fas fa-ban"></i> 취소</button>        </div>      </div>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          todoMap: {},          todo: {},          showDialog: false,          selectedKey: null        },        methods: {          create: function() {            this.showDialog = "등록";          ;            this.todo = {};          },          reload: function() {            axios.get('https://skhu-lsj.firebaseio.com/todo4.json')                 .then(response => this.todoMap = response.data);          },          edit: function(key) {            this.showDialog = "수정";            this.todo = Object.assign({}, this.todoMap[key]);            this.selectedKey = key;          },          save: function() {            if (this.showDialog == "등록")              axios.post('https://skhu-lsj.firebaseio.com/todo4.json', this.todo)                   .then(response => this.reload());            else              axios.put('https://skhu-lsj.firebaseio.com/todo4/' + this.selectedKey + '.json', this.todo)                   .then(response => this.reload());            this.showDialog = false;          },          remove: function() {            if (confirm("삭제하시겠습니까?") == false) return;            axios.delete('https://skhu-lsj.firebaseio.com/todo4/' + this.selectedKey + '.json')                   .then(response => this.reload());            this.showDialog = false;          }        }      })      window.onload = function() {        app.reload();      }     </script>  </body>  </html> |

# 과제

## todo17.html

todo16.html 예제에서 firebase realtime database 관련 코드를 제거하고

axios 라이브러리와 firebase REST API를 이용하여 데이터 조회, 저장, 삭제 기능을 구현하시오.