todo 앱

2020-05-11

이승진

**학습목표**

대화상자 구현

firebase realtime database 이벤트 핸들러

**목차**

[1. 대화상자 2](#_Toc40097471)

[1) 대화상자 팝업 구현 2](#_Toc40097472)

[2) 대화상자 백드롭 구현 4](#_Toc40097473)

[2. todo 앱3 6](#_Toc40097474)

[1) todo3.css 6](#_Toc40097475)

[2) todo30.html 7](#_Toc40097476)

[3) todo31.html 10](#_Toc40097477)

[4) todo32.html 16](#_Toc40097478)

[5) todo33.html 19](#_Toc40097479)

[6) todo34.html 21](#_Toc40097480)

[3. firebase realtime database 24](#_Toc40097481)

[1) 원격 참조 객체 24](#_Toc40097482)

[2) 목록 전체 전송 24](#_Toc40097483)

[3) 항목 하나씩 전송 25](#_Toc40097484)

[4. todo 앱4 27](#_Toc40097485)

[1) todo40.html 27](#_Toc40097486)

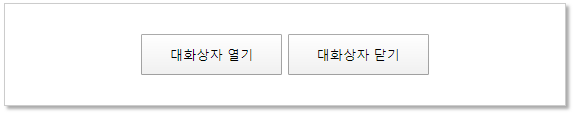
[2) todo41.html 31](#_Toc40097487)

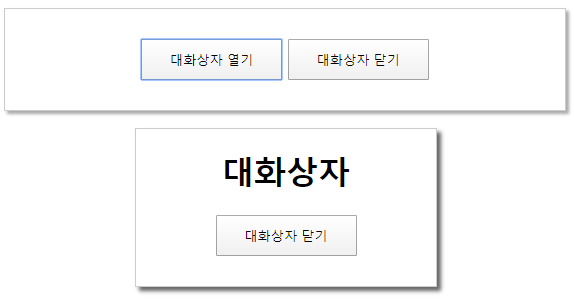
# 대화상자

## 대화상자 팝업 구현

### dialog1.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <style>      div#main { padding: 30px; margin: 30px auto; width: 500px; text-align: center;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa; }      button { padding: 0.7em 2em; }      div#dialog { position: absolute; top: 150px; left: 50%;        width: 300px; margin-left: -150px; padding-bottom: 30px;        background-color: white; text-align: center;        border: 1px solid #ccc; box-shadow: 4px 4px 4px #666;      }    </style>  </head>  <body>    <div id="app">      <div id="main">        <button type="button" v-on:click="showDialog = true">대화상자 열기</button>        <button type="button" v-on:click="showDialog = false">대화상자 닫기</button>      </div>      <div id="dialog" v-show="showDialog">        <h1>대화상자</h1>        <button type="button" v-on:click="showDialog = false">대화상자 닫기</button>      </div>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          showDialog: false        }      })     </script>  </body>  </html> |





### 대화상자 위치 서식

position: absolute;

이 태그는 다른 태그들 위에 떠서 위치함.

이 태그의 위치를 지정하는 x y 좌표는 각각 left, top 서식으로 지정함

top: 150px;

이 태그의 y 좌표는 화면 상단에서 150px 아래

left: 50%;

이 태그의 x 좌표는 부모 영역의 중앙이다.

position absolute인 태그의 부모는 화면(웹브라우저창)이다.

즉 이 태그의 x 좌표는 화면의 중앙이다.

width: 300px;

이 태그의 폭은 300px

margin-left: -150px;

이 태그의 왼쪽 여백은 -150px

왼쪽 여백이 양수이면, 그 값만큼 오른쪽으로 이동하고, 음수이면 왼쪽으로 이동함.

left 좌표가 화면의 중앙인데 (50%), 여기서 150px 왼쪽으로 이동함.

이 태그의 폭이 300px 이므로, 이 태그의 중앙이 화면의 중앙에 위치하게 됨.

즉 이 태그는 다른 태그들 위에 떠있으면서

세로 방향으로 화면 상단에서 150px 아래에

사로 방향으로 화면 중앙에 위치한다.

<button type="button" v-on:click="showDialog = true">대화상자 열기</button>

이 버튼을 클릭하면 showDialog 속성에 true 값이 대입된다.

그래서 아래 태그가 화면에 보이게 된다.

<div id="dialog" v-show="showDialog">

<button type="button" v-on:click="showDialog = false">대화상자 닫기</button>

이 버튼을 클릭하면 showDialog 속성에 false 값이 대입된다.

### 개선할 점

대화상자가 열려있으면, 대화 상자 아래를 클릭할 수 없도록 막자.

이런 방식의 대화상자를 modal dialog 라고 함.

## 대화상자 백드롭 구현

### dialog2.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <style>      div#main { padding: 30px; margin: 30px auto; width: 500px; text-align: center;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa; }      button { padding: 0.7em 2em; }      div#dialog { position: absolute; top: 150px; left: 50%;        width: 300px; margin-left: -150px; padding-bottom: 30px;        background-color: white; text-align: center;        border: 1px solid #ccc; box-shadow: 4px 4px 4px #666;      }      div#backdrop { position: absolute; left: 0; top: 0; width: 100%; height: 100%;        background-color: #aaa; opacity: 0.5; }    </style>  </head>  <body>    <div id="app">      <div id="main">        <button type="button" v-on:click="showDialog = true">대화상자 열기</button>        <button type="button" v-on:click="showDialog = false">대화상자 닫기</button>      </div>      <div id="backdrop" v-show="showDialog"></div>      <div id="dialog" v-show="showDialog">        <h1>대화상자</h1>        <button type="button" v-on:click="showDialog = false">대화상자 닫기</button>      </div>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          showDialog: false        }      })     </script>  </body>  </html> |

### <div id="backdrop"> 서식

position: absolute;

이 태그는 다른 태그들 위에 떠서 위치함.

그런데, <div id="dialog"> 태그도 다른 태그들 위에 떠서 위치한다.

이 두 태그들 중, 소스코드에서 더 아래에 있는 <div id="dialog"> 태그가, 더 높게 떠있다.

즉 다른 태그들 위에 <div id="backdrop"> 태그가 떠 있고,

그 위에 <div id="dialog"> 태그가 떠 있다.

left: 0; top: 0; width: 100%; height: 100%;

이 태그의 x, y 좌표는 부모의 왼쪽 위 구석이고,

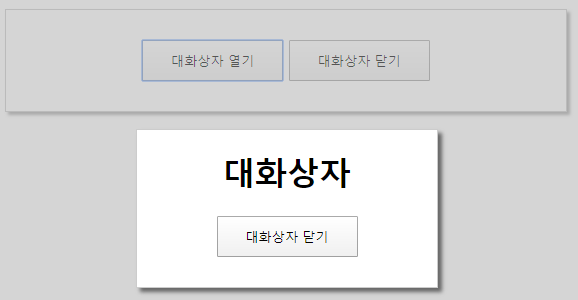
이 태그의 가로 세로 크기는 부모의 영역을 가득 채운다.

position absolute인 태그의 부모는 화면(웹브라우저창)이다.

즉 이 태그 크기는 웹브라우저창을 가득 채운다.

background-color: #aaa; opacity: 0.5;

이 태그의 배경색은 반투명한 회색이다.



반투명한 <div id="backdrop"> 태그가 대화상자 아래 화면을 덮고 있기 때문에,

대화상자 아래 부분을 클릭하면, <div id="backgroup"> 태그가 클릭된다.

따라서 아무 반응이 없다.

# todo 앱3

할일을 입력하기 위한 대화상자 구현

## todo3.css

|  |
| --- |
| /\* 공통 \*/  button { padding: 0.4em 1.5em; }  table { border-collapse: collapse; margin-bottom: 20px; width: 100%; }  tr:nth-child(1) { background-color: #ddd; }  tr.clickable:hover { background-color: #ffb; cursor: pointer; }  td { padding: 8px;  border: 1px solid #aaa; font-size: 10pt; }  td:nth-child(2) { width: 100px; }  /\* main \*/  div#main { padding: 30px; margin: 30px auto; max-width: 600px;    border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa;  }  div#main h1 { border-bottom: 1px solid gray; }  /\* dialog \*/  div#backdrop { position: absolute; left: 0; top: 0; width: 100%; height: 100%;    background-color: #aaa; opacity: 0.5;  }  div#dialog { position: absolute; top: 50px; left: 50%;    padding: 30px; margin-left: -250px; max-width: 500px;    background-color: white;    border: 1px solid #ccc; box-shadow: 4px 4px 4px #666;  }  div#dialog h1 { border-bottom: 1px solid gray; }  label, input[type=text], textarea { font-size: 10pt; }  input[type=text], input[type=date] { height: 2.4em; padding: 4px; }  input[type=text] { width: 100%; margin-bottom: 10px; }  textarea { padding: 4px; width: 100%; height: 250px; margin-bottom: 10px; } |

## todo30.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <script defer src="/\_\_/firebase/7.13.1/firebase-app.js"></script>    <script defer src="/\_\_/firebase/7.13.1/firebase-database.js"></script>    <script defer src="/\_\_/firebase/init.js"></script>    <link rel="stylesheet" type="text/css" href="todo3.css">  </head>  <body>    <div id="app">      <div id="main">        <h1>할 일</h1>        <table>          <tr>            <td>제목</td>            <td>날짜</td>          </tr>          <tr v-for="todo in todoList">            <td>{{ todo.title }}</td>            <td>{{ todo.due\_date }}</td>          </tr>          <tr v-if="todoList.length == 0">            <td colspan="2">등록된 자료가 없습니다.</td>          </tr>        </table>        <button type="button" v-on:click="showDialog = true">추가</button>      </div>      <div id="backdrop" v-show="showDialog"></div>      <div id="dialog" v-show="showDialog">        <h1>할일 등록</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" v-bind:disabled="!todo.title">저장</button>          <button type="button" v-on:click="showDialog = false">취소</button>        </div>      </div>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          todoList: [],          todo: {},          showDialog: false        },        methods: {          save: function() {            this.todoList.push(this.todo);            this.todo3Ref.set(this.todoList);            this.todo = {};            this.showDialog = false;          },        }      })      window.onload = function() {        app.todo3Ref = firebase.database().ref("todo3");        app.todo3Ref.on('value', (snapshot) => {          app.todoList  = snapshot.val() || [];        });      }     </script>  </body>  </html> |

<tr v-if="todoList.length == 0">

  <td colspan="2">등록된 자료가 없습니다.</td>

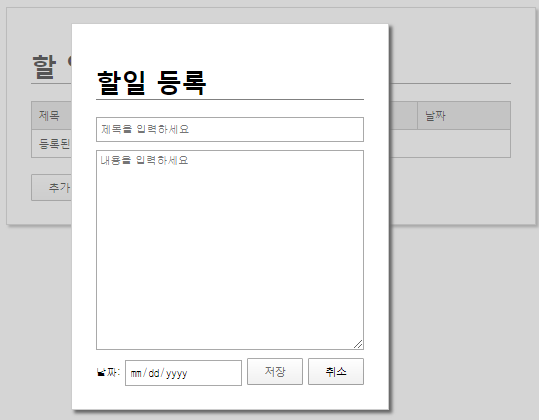
</tr>

todoList 목록이 비어있을 때 아래와 같이 출력된다.



<button type="button" v-on:click="showDialog = true">추가</button>

이 버튼을 클릭하면 대화상자가 화면에 팝업된다.



<input type="text" v-model.trim="todo.title" placeholder="제목을 입력하세요"/>

<textarea v-model.trim="todo.body" placeholder="내용을 입력하세요"></textarea>

<input type="date" v-model="todo.due\_date" />

대화상자의 input 태그에 입력된 값들은, todo 속성의 내부에 대입된다.

todo 속성의 값은 객체이다. 이 객체의 title, body, due\_date 속성에 대입된다.

## todo31.html

할일 수정 기능 구현

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <script defer src="/\_\_/firebase/7.13.1/firebase-app.js"></script>    <script defer src="/\_\_/firebase/7.13.1/firebase-database.js"></script>    <script defer src="/\_\_/firebase/init.js"></script>    <link rel="stylesheet" type="text/css" href="todo3.css">  </head>  <body>    <div id="app">      <div id="main">        <h1>할 일</h1>        <table>          <tr>            <td>제목</td>            <td>날짜</td>          </tr>          <tr v-for="(todo, index) in todoList" v-on:click="edit(index)" class="clickable">            <td>{{ todo.title }}</td>            <td>{{ todo.due\_date }}</td>          </tr>          <tr v-if="todoList.length== 0">            <td colspan="2">등록된 자료가 없습니다.</td>          </tr>        </table>        <button type="button" v-on:click="create">추가</button>      </div>      <div id="backdrop" v-show="showDialog"></div>      <div id="dialog" v-show="showDialog">        <h1>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">저장</button>          <button type="button" v-on:click="showDialog = false">취소</button>        </div>      </div>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          todoList: [],          todo: {},          showDialog: false        },        methods: {          create: function() {            this.showDialog = "등록";          ;            this.todo = {};          },          edit: function(index) {            this.showDialog = "수정";            this.todo = this.todoList[index];          },          save: function() {            if (this.showDialog == "등록")              this.todoList.push(this.todo);            this.todo3Ref.set(this.todoList);            this.showDialog = false;          }        }      })      window.onload = function() {        app.todo3Ref = firebase.database().ref("todo3");        app.todo3Ref.on('value', (snapshot) => {          app.todoList  = snapshot.val() || [];        });      }     </script>  </body>  </html> |

### 할일 추가

<button type="button" v-on:click="create">추가</button>

추가 버튼을 클릭하면 create 메소드가 호출된다.

create: function() {

  this.showDialog = "등록";

  this.todo = {};

}

this.showDialog = "등록";

showDialog 속성값이 "등록"이면,

<div id="dialog" v-show="showDialog"> 대화상자가 화면에 보이게 되고,

    <h1>할일 {{ showDialog }}</h1> 대화상자의 제목이 '할일 등록' 이 된다.

this.todo = {};

내용이 없는 새 객체를 생성해서, todo 속성에 대입한다.

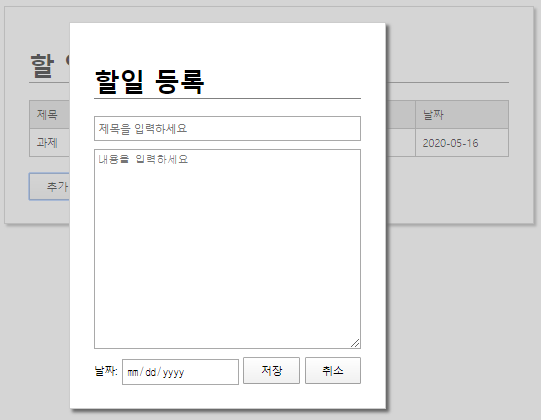
todo 속성은, 대화상자의 입력태그들에 연결되어 있다.

todo 속성값이 빈 객체이면, 대화상자의 입력태그들도 비어있게 된다.

<input type="text" v-model.trim="todo.title" placeholder="제목을 입력하세요"/>

<textarea v-model.trim="todo.body" placeholder="내용을 입력하세요"></textarea>

<input type="date" v-model="todo.due\_date" />



대화상자의 입력 태그들에 입력된 값은 todo 객체 내부의 속성에 대입된다.

<button type="button" v-on:click="save">저장</button>

저장 버튼을 클릭하면 save 메소드가 호출된다.

save: function() {

  if (this.showDialog == "등록")

    this.todoList.push(this.todo);

  this.todo3Ref.set(this.todoList);

  this.showDialog = false;

}

if 문이 true 이므로 this.todoList.push(this.todo); 코드가 실행된다.

대화상자에 입력된 내용이 todoList 배열의 끝에 추가된다.

this.todo3Ref.set(this.todoList);

todoList 배열을 firebase DB에 저장한다.

this.showDialog = false;

대화상자를 닫는다.

### 할일 수정

<tr v-for="(todo, index) in todoList" v-on:click="edit(index)" class="clickable">

할일 목록의 각 행에 clickable 서식이 적용된다.

tr.clickable:hover { background-color: #ffb; cursor: pointer; }

<tr class="clickable"> 태그에 마우스 포인터가 올라가면, 이 서식이 적용된다.

배경색은 밝은 노란색이되고, 마우스 포인터는 손가락 모양이된다.

<tr v-for="(todo, index) in todoList" v-on:click="edit(index)" class="clickable">

할일 목록에서 한 행을 클릭하면 edit 메소드가 호출된다.

이 메소드의 파라미터 값은, 클릭된 항목의 index 이다.

edit: function(index) {

  this.showDialog = "수정";

  this.todo = this.todoList[index];

},

this.showDialog = "수정";

showDialog 속성값이 "수정"이면,

<div id="dialog" v-show="showDialog"> 대화상자가 화면에 보이게 되고,

    <h1>할일 {{ showDialog }}</h1> 대화상자의 제목이 '할일 수정' 이 된다.

this.todo = this.todoList[index];

index 값은 클릭된 할일 항목의 index이다.

this.todoList[index] 값은 클릭된 할일 객체이다.

this.todoList[index] 값을 this.todo에 대입하면, 객체가 복사되어서 대입되는 것이 아니고,

참조값만 대입된다. 즉 this.todo 참조변수와 this.todoList[index] 참조변수가

동일한 객체를 참조하게 된다. 그 객체는 클릭된 할일 객체이다.

this.todo 참조변수가 참조하는 객체는 클릭된 할일 객체이다.

대화상자의 input 태그들은 this.todo 참조변수가 참조하는 객체의 속성들에 연결되어 있다.

예를 들어 두 번째 할일 항목이 클릭되었다면, 앱의 메모리 구조는 다음과 같다.



<input type="text" v-model.trim="todo.title" placeholder="제목을 입력하세요"/>

<textarea v-model.trim="todo.body" placeholder="내용을 입력하세요"></textarea>

<input type="date" v-model="todo.due\_date" />

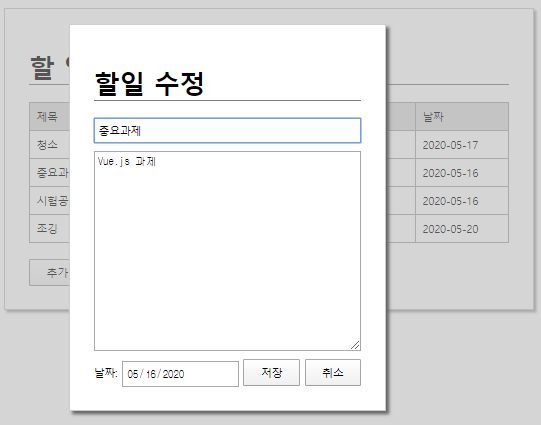
클릭된 할일 항목의 내용이 대화상자의 input 태그들에 채워진다.

input 태그의 값을 수정하면, 그 값은 todo 참조변수가 참조하는 객체 내부 속성에 즉시 대입된다.

즉 todoList 배열 내부의 클릭된 항목 객체 내부 속성에 즉시 대입된다.

예를 들어, 제목을 "중요과제" 라고 수정하면, 앱의 메모리 구조는 다음과 같다.





<button type="button" v-on:click="save">저장</button>

저장 버튼을 클릭하면 save 메소드가 호출된다.

save: function() {

  if (this.showDialog == "등록")

    this.todoList.push(this.todo);

  this.todo3Ref.set(this.todoList);

  this.showDialog = false;

}

if 문이 false 이므로 this.todoList.push(this.todo); 코드는 실행되지 않는다.

대화상자의 input 태그에 입력된 내용은 이미 todoList 배열 내부의 클릭된 항목 객체 내부에 채워져 있다.

this.todo3Ref.set(this.todoList);

todoList 배열을 firebase DB에 저장한다.

this.showDialog = false;

대화상자를 닫는다.

### 개선할 부분

input 태그의 값을 수정하면, 그 값이 todoList 배열 내부의 클릭된 항목 객체 내부 속성에 즉시 대입된다.

그래서 대화상자에서 취소 버튼을 클릭해도, 할일 목록의 내용은 이미 수정되어 있다.

수정된 내용이 DB에 저장되지는 않았다.

최소 버튼이 클릭되는 경우에는, 할일 목록의 내용이 수정되지 않도록 수정하자.

삭제 기능을 구현하자.

## todo32.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <script defer src="/\_\_/firebase/7.13.1/firebase-app.js"></script>    <script defer src="/\_\_/firebase/7.13.1/firebase-database.js"></script>    <script defer src="/\_\_/firebase/init.js"></script>    <link rel="stylesheet" type="text/css" href="todo3.css">  </head>  <body>    <div id="app">      <div id="main">        <h1>할 일</h1>        <table>          <tr>            <td>제목</td>            <td>날짜</td>          </tr>          <tr v-for="(todo, index) in todoList" v-on:click="edit(index)" class="clickable">            <td>{{ todo.title }}</td>            <td>{{ todo.due\_date }}</td>          </tr>          <tr v-if="todoList.length == 0">            <td colspan="2">등록된 자료가 없습니다.</td>          </tr>        </table>        <button type="button" v-on:click="create">추가</button>      </div>      <div id="backdrop" v-show="showDialog"></div>      <div id="dialog" v-show="showDialog">        <h1>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">저장</button>          <button type="button" v-on:click="this.showDialog = false">취소</button>        </div>      </div>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          todoList: [],          todo: {},          showDialog: false,          selectedIndex: -1        },        methods: {          create: function() {            this.showDialog = "등록";          ;            this.todo = {};          },          edit: function(index) {            this.showDialog = "수정";            this.todo = Object.assign({}, this.todoList[index]);            this.selectedIndex = index;          },          save: function() {            if (this.showDialog == "등록")              this.todoList.push(this.todo);            else              this.todoList[this.selectedIndex] = this.todo;            this.todo3Ref.set(this.todoList);            this.showDialog = false;          }        }      })      window.onload = function() {        app.todo3Ref = firebase.database().ref("todo3");        app.todo3Ref.on('value', (snapshot) => {          app.todoList  = snapshot.val() || [];        });      }     </script>  </body>  </html> |

앱의 selectedIndex 속성은, 클릭된 항목의 index 이다.

edit: function(index) {

  this.showDialog = "수정";

  this.todo = Object.assign({}, this.todoList[index]);

  this.selectedIndex = index;

}

할일 항목을 클릭하면 edit 메소드가 호출된다. 이 메소드의 파라미터는 클릭된 항목의 index 이다.

this.todo = Object.assign({}, this.todoList[index]);

클릭된 항목에 해당하는 복제하여 새 객체를 생성한다.

복제된 새 객체에 대한 참조를 this.todo 참조 변수에 대입한다.

this.selectedIndex = index;

클릭된 항목의 index 값을 selectedIndex 속성에 대입한다.

예를 들어 두 번째 할일 항목이 클릭되었다면, 앱의 메모리 구조는 다음과 같다.



<input type="text" v-model.trim="todo.title" placeholder="제목을 입력하세요"/>

<textarea v-model.trim="todo.body" placeholder="내용을 입력하세요"></textarea>

<input type="date" v-model="todo.due\_date" />

todo 속성이 참조하는 객체의 내용이 대화상자의 input 태그들에 채워진다.

input 태그의 값을 수정하면, 그 값은 todo 참조변수가 참조하는 객체 내부 속성에 즉시 대입된다.

이 객체는 복제된 객체이다.

예를 들어, 제목을 "중요과제" 라고 수정하면, 앱의 메모리 구조는 다음과 같다.



여기서 취소 버튼을 클릭하면, todoList 배열의 내용에는 변화가 없다.

저장 버튼을 클릭하면 save 메소드가 호출된다.

save: function() {

  if (this.showDialog == "등록")

    this.todoList.push(this.todo);

  else

    this.todoList[this.selectedIndex] = this.todo;

  this.todo3Ref.set(this.todoList);

  this.showDialog = false;

}

this.showDialog 속성값이 "수정" 이므로, if 문은 false 이고,

this.todoList[this.selectedIndex] = this.todo; 코드가 실행된다.

this.todo 참조변수가 참조하는 객체에 대한 참조값이, this.todoList[this.selectedIndex]에 대입된다.

이 코드가 실행된 후 앱의 메모리 구조는 다음과 같다.



즉 todoList 배열의 내용이 변경되었다.

this.todo3Ref.set(this.todoList);

todoList 배열의 내용을 firebase DB에 저장하고

this.showDialog = false;

대화상자를 닫는다.

## todo33.html

삭제 기능 구현

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <script defer src="/\_\_/firebase/7.13.1/firebase-app.js"></script>    <script defer src="/\_\_/firebase/7.13.1/firebase-database.js"></script>    <script defer src="/\_\_/firebase/init.js"></script>    <link rel="stylesheet" type="text/css" href="todo3.css">  </head>  <body>    <div id="app">      <div id="main">        <h1>할 일</h1>        <table>          <tr>            <td>제목</td>            <td>날짜</td>          </tr>          <tr v-for="(todo, index) in todoList" v-on:click="edit(index)" class="clickable">            <td>{{ todo.title }}</td>            <td>{{ todo.due\_date }}</td>          </tr>          <tr v-if="todoList.length == 0">            <td colspan="2">등록된 자료가 없습니다.</td>          </tr>        </table>        <button type="button" v-on:click="create">추가</button>      </div>      <div id="backdrop" v-show="showDialog"></div>      <div id="dialog" v-show="showDialog">        <h1>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">저장</button>          <button type="button" v-on:click="remove">삭제</button>          <button type="button" v-on:click="this.showDialog = false">취소</button>        </div>      </div>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          todoList: [],          todo: {},          showDialog: false,          selectedIndex: -1        },        methods: {          create: function() {            this.showDialog = "등록";          ;            this.todo = {};          },          edit: function(index) {            this.showDialog = "수정";            this.todo = Object.assign({}, this.todoList[index]);            this.selectedIndex = index;          },          save: function() {            if (this.showDialog == "등록")              this.todoList.push(this.todo);            else              this.todoList[this.selectedIndex] = this.todo;            this.todo3Ref.set(this.todoList);            this.showDialog = false;          },          remove: function() {            if (confirm("삭제하시겠습니까?") == false) return;            this.todoList.splice(this.selectedIndex, 1);            this.todo3Ref.set(this.todoList);            this.showDialog = false;          }        }      })      window.onload = function() {        app.todo3Ref = firebase.database().ref("todo3");        app.todo3Ref.on('value', (snapshot) => {          app.todoList  = snapshot.val() || [];        });      }     </script>  </body>  </html> |

노란색으로 칠한 코드만 변경되었다.

<button type="button" v-on:click="remove">삭제</button>

삭제 버튼을 클릭하면 remove 메소드가 호출된다.

remove: function() {

  if (confirm("삭제하시겠습니까?") == false) return;

this.todoList.splice(this.selectedIndex, 1);

  this.todo3Ref.set(this.todoList);

  this.showDialog = false;

}

confirm("삭제하시겠습니까?")

삭제하시겠습니까? 묻는 대화상자가 나타난다. 이 대화상자에서 cancel 버튼을 클릭하면,

confirm 함수는 false를 리턴하고, ok 버튼을 클릭하면 true를 리턴한다.

if (confirm("삭제하시겠습니까?") == false) return;

삭제하시겠습니까? 묻는 대화상자에서 cancel 버튼을 클릭하면, remove 메소드에서 리턴하고,

ok 버튼을 클릭하면, 다음 문장으로 실행을 계속한다.

this.todoList.splice(this.selectedIndex, 1);

todoList 배열에서 selectedIndex 위치의 항목을 제거한다.

배열의 크기는 1 줄어든다.

this.todo3Ref.set(this.todoList);

todoList 배열의 내용을 firebase DB에 저장하고

this.showDialog = false;

대화상자를 닫는다.

## todo34.html

fontawesome 아이콘 사용

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <script defer src="/\_\_/firebase/7.13.1/firebase-app.js"></script>    <script defer src="/\_\_/firebase/7.13.1/firebase-database.js"></script>    <script defer src="/\_\_/firebase/init.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, index) in todoList" v-on:click="edit(index)" class="clickable">            <td>{{ todo.title }}</td>            <td>{{ todo.due\_date }}</td>          </tr>          <tr v-if="todoList.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="cancel"><i class="fas fa-ban"></i> 취소</button>        </div>      </div>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          todoList: [],          todo: {},          showDialog: false,          selectedIndex: -1        },        methods: {          create: function() {            this.showDialog = "등록";          ;            this.todo = {};          },          edit: function(index) {            this.showDialog = "수정";            this.todo = Object.assign({}, this.todoList[index]);            this.selectedIndex = index;          },          save: function() {            if (this.showDialog == "등록")              this.todoList.push(this.todo);            else              this.todoList[this.selectedIndex] = this.todo;            this.todo3Ref.set(this.todoList);            this.showDialog = false;          },          remove: function() {            if (confirm("삭제하시겠습니까?") == false) return;            this.todoList.splice(this.selectedIndex, 1);            this.todo3Ref.set(this.todoList);            this.showDialog = false;          },          cancel: function() {            this.showDialog = false;          }        }      })      window.onload = function() {        app.todo3Ref = firebase.database().ref("todo3");        app.todo3Ref.on('value', (snapshot) => {          app.todoList  = snapshot.val() || [];        });      }     </script>  </body>  </html> |

### font awesome

아이콘을 출력 기능을 제공해 주는 서비스이다.

<script src="https://kit.fontawesome.com/68abb170e0.js" crossorigin="anonymous"></script>

이 태그가 head 태그에 포함되어야 한다.

<i class="fas fa-clipboard-check"></i>

<i class="fas fa-edit"></i>

<i class="fas fa-check"></i>

<i class="fas fa-trash-alt"></i>

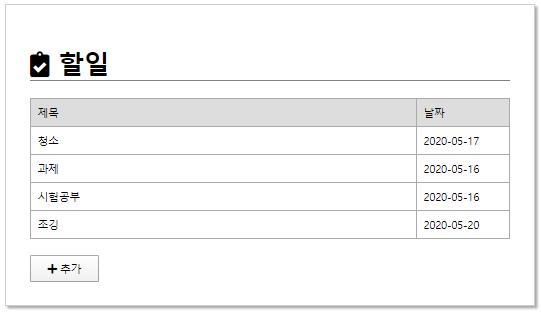
<i class="fas fa-ban"></i>

아이콘 출력 태그이다.

노란색으로 칠한 부분이 아이콘 이름이다.

<https://fontawesome.com/icons?d=gallery>

아이콘 목록을 볼 수 있다.





# firebase realtime database

## 원격 참조 객체

firebase DB 데이터 항목에 대한 원격 참조 객체 생성

|  |
| --- |
| todo3Ref = firebase.database().ref("todo3"); |

## 목록 전체 전송

### 목록 저장하기

|  |
| --- |
| todo3Ref.set(todoList); |

todo3Ref 원격 참조 객체가 참조하는 데이터 항목에, todoList 목록 전체를 한 번에 저장한다.

### value 이벤트 핸들러

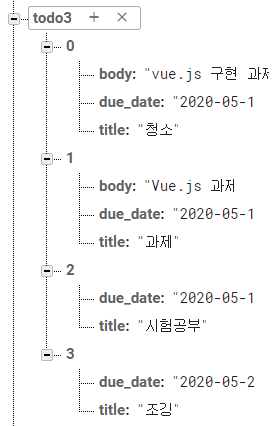
|  |
| --- |
| app.todo3Ref.on('value', (snapshot) => {    app.todoList  = snapshot.val();  }); |

todo3Ref 원격 참조 객체가 참조하는 데이터 항목의 내부 값이 변경되면, 콜백 함수가 즉시 호출된다.

데이터 항목의 값 전체가, 콜백함수의 파라미터로 전달된다.

### 배열 형태로 저장됨

목록 전체를 저장하면, 목록은 배열 형태로 저장된다.



## 항목 하나씩 전송

### 자식 항목 추가하기 (insert)

|  |
| --- |
| todo4Ref.push().set(todo); |

todo4Ref 원격 참조 객체가 참조하는 데이터 항목 아래에 새 자식 항목을 추가(insert)한다.

자식 항목을 추가(insert)하려면, 먼저 그 자식 항목에 대한 키(key)를 먼저 만들어야 한다.

위 코드의 push() 메소드에 의해서, 새 키(key)가 생성되고,

set 메소드에 의해서, 그 새 키(key)의 위치에 데이터가 저장된다.

### 자식 항목 수정하기 (update)

|  |
| --- |
| todo4Ref.child(key).set(todo); |

키(key) 위치의 데이터를 수정(update) 한다.

### 자식 항목 삭제하기 (delete)

|  |
| --- |
| todo4Ref.child(key).set(todo); |

키(key) 위치의 데이터를 삭제(delete) 한다.

### child\_added 이벤트 핸들러

|  |
| --- |
| todo4Ref.on('child\_added', (snapshot) => {    let key = snapshot.key;  let value = snapshot.val();  }); |

todo4Ref 원격 참조 객체가 참조하는 데이터 항목 아래에 새 자식 항목이 추가되면,

위 콜백 함수가 즉시 호출된다. 추가된 자식 항목의 값은 snapshot.val()이고, 키 값은 snapshot.key 이다.

### child\_updated 이벤트 핸들러

|  |
| --- |
| app.todo4Ref.on('child\_changed', (snapshot) => {    let key = snapshot.key;  let value = snapshot.val();  }); |

todo4Ref 원격 참조 객체가 참조하는 데이터 항목 아래의 자식 항목 중 하나가 수정되면,

위 콜백 함수가 즉시 호출된다. 수정된 자식 항목의 값은 snapshot.val()이고, 키 값은 snapshot.key 이다.

### child\_removed 이벤트 핸들러

|  |
| --- |
| app.todo4Ref.on('child\_removed', (snapshot) => {    let key = snapshot.key;  let value = snapshot.val();  }); |

todo4Ref 원격 참조 객체가 참조하는 데이터 항목 아래의 자식 항목 중 하나가 삭제되면,

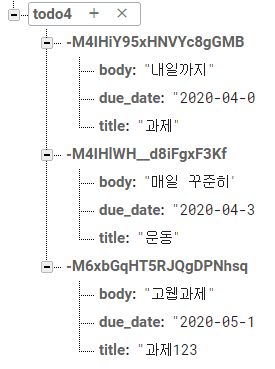
위 콜백 함수가 즉시 호출된다. 삭제된 자식 항목의 값은 snapshot.val()이고, 키 값은 snapshot.key 이다.

### 객체 형태로 저장

자식 항목을 하나씩 저장하면, 그 목록은 객체 형태로 저장된다.

자식 항목들 각각은 객체의 속성 형태로 저장된다.

키 값이 속성명이고, 자식 항목의 값이 속성값이 된다.



# todo 앱4

## todo40.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <script defer src="/\_\_/firebase/7.13.1/firebase-app.js"></script>    <script defer src="/\_\_/firebase/7.13.1/firebase-database.js"></script>    <script defer src="/\_\_/firebase/init.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 = {};          },          edit: function(key) {            this.showDialog = "수정";            this.todo = Object.assign({}, this.todoMap[key]);            this.selectedKey = key;          },          save: function() {            if (this.showDialog == "등록")              this.todo4Ref.push().set(this.todo);            else              this.todo4Ref.child(this.selectedKey).set(this.todo);            this.showDialog = false;          },          remove: function() {            if (confirm("삭제하시겠습니까?") == false) return;            this.todo4Ref.child(this.selectedKey).remove();            this.showDialog = false;          }        }      })      window.onload = function() {        app.todo4Ref = firebase.database().ref("todo4");        app.todo4Ref.on('child\_added', (snapshot) => {          app.todoMap[snapshot.key] = snapshot.val();          app.$forceUpdate();        });        app.todo4Ref.on('child\_changed', (snapshot) => {          app.todoMap[snapshot.key] = snapshot.val();          app.$forceUpdate();        });        app.todo4Ref.on('child\_removed', (snapshot) => {          delete app.todoMap[snapshot.key];          app.$forceUpdate();        });      }     </script>  </body>  </html> |

todoMap: {}

이제 할일 목록은 배열이 아니고 객체이다.

이 객체의 속성들 각각이 할일 항목이다.

할일 항목이 속성값이고, 그 항목의 키(key)가 속성 이름이다.

<tr v-for="(todo, key) in todoMap" v-on:click="edit(key)" v-bind:key="key" class="clickable">

todoMap 객체의 속성 각각에 대해서 tr 태그와 그 자식 태그들을 반복 생성한다.

속성의 값이 todo 변수에 대입되고, 속성 이름이 key 변수에 대입된다.

v-bind:key="key"

반복 생성되는 태그에 v-bind:key 값으로 유일한 값을 지정해 주면 화면 갱신 작업의 효율이 향상된다.

<tr v-if="Object.keys(todoMap).length == 0">

Object.keys(todoMap) todoMap 객체의 속성의 이름들로 구성된 배열을 리턴한다.

Object.keys(todoMap).length todoMap 객체의 속성의 이름들로 구성된 배열의 길이를 구한다.

즉 todoMap 객체의 속성의 수를 구한다.

이 값은 할일 목록의 크기이다.

selectedKey: null

선택된 항목을 구별하기 위해서, 그 항목의 키를 보관한다.

<tr v-for="(todo, key) in todoMap" v-on:click="edit(key)" v-bind:key="key" class="clickable">

할일 항목을 클릭하면 edit 메소드가 호출된다.

edit 메소드의 파라미터는 클릭된 항목의 키(key)이다.

edit: function(key) {

  this.showDialog = "수정";

  this.todo = Object.assign({}, this.todoMap[key]);

  this.selectedKey = key;

}

클릭된 항목을 복제한 새 객체를 생성하고, 그 객체에 대한 참조를 todo 참조 변수에 대입한다.

클릭된 항목의 키를 selectedKey 변수에 대입한다.

save: function() {

  if (this.showDialog == "등록")

    this.todo4Ref.push().set(this.todo);

  else

   this.todo4Ref.child(this.selectedKey).set(this.todo);

  this.showDialog = false;

},

this.todo4Ref.push().set(this.todo);

대화상자에 입력된 값을, firebase DB의 "todo4" 항목 아래에 추가한다. (insert)

this.todo4Ref.child(this.selectedKey).set(this.todo);

대화상자에 입력된 값을, firebase DB의 "todo4" 항목 아래의 selectedKey 키 항목에 저장한다. (update)

selectedKey 속성값은 수정하기 위해 클릭된 항목의 카(key)이다.

remove: function() {

  if (confirm("삭제하시겠습니까?") == false) return;

  this.todo4Ref.child(this.selectedKey).remove();

  this.showDialog = false;

}

this.todo4Ref.child(this.selectedKey).remove();

firebase DB의 "todo4" 항목 아래의 selectedKey 키 항목을 삭제한다. (delete)

app.todo4Ref.on('child\_added', (snapshot) => {

  app.todoMap[snapshot.key] = snapshot.val();

  app.$forceUpdate();

});

firebase DB의 "todo4" 항목 아래에 새 자식 항목이 추가되면 이 콜백함수가 호출된다.

추가된 자식 항목의 값은 snapshot.val()이고, 키 값은 snapshot.key 이다.

app.todoMap[snapshot.key] = snapshot.val();

todoMap 객체에 새 속성을 추가한다.

javascript에서 객체의 속성에 처음 값을 대입할 때 그 속성이 생성된다.

Vue 앱이 시작된 후에 생성된 속성은 그 값이 변경되어도 즉시 화면이 다시 그려지지 않는다.

그래서 화면이 다시 그려지도록 강자헤가 위해서 this.$forceUpdate() 메소드 호출이 필요하다.

app.todo4Ref.on('child\_changed', (snapshot) => {

  app.todoMap[snapshot.key] = snapshot.val();

  app.$forceUpdate();

});

firebase DB의 "todo4" 항목 아래의 자식 항목의 값이 수정되면 이 콜백함수가 호출된다.

수정된 자식 항목의 값은 snapshot.val()이고, 키 값은 snapshot.key 이다.

  app.todoMap[snapshot.key] = snapshot.val();

todoMap 객체의 snapshot.key 이름의 속성에 snapshot.val() 값을 대입한다.

app.todo4Ref.on('child\_removed', (snapshot) => {

  delete app.todoMap[snapshot.key];

  app.$forceUpdate();

});

firebase DB의 "todo4" 항목 아래에 자식 항목이 삭제되면 이 콜백함수가 호출된다.

  delete app.todoMap[snapshot.key];

todoMap 객체에서 snapshot.key 이름의 속성을 제거한다.

## todo41.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <script defer src="/\_\_/firebase/7.13.1/firebase-app.js"></script>    <script defer src="/\_\_/firebase/7.13.1/firebase-database.js"></script>    <script defer src="/\_\_/firebase/init.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 = {};          },          edit: function(key) {            this.showDialog = "수정";            this.todo = Object.assign({}, this.todoMap[key]);            this.selectedKey = key;          },          save: function() {            if (this.showDialog == "등록")              this.todo4Ref.push().set(this.todo);            else              this.todo4Ref.child(this.selectedKey).set(this.todo);            this.showDialog = false;          },          remove: function() {            if (confirm("삭제하시겠습니까?") == false) return;            this.todo4Ref.child(this.selectedKey).remove();            this.showDialog = false;          }        }      })      window.onload = function() {        app.todo4Ref = firebase.database().ref("todo4");        app.todo4Ref.on('child\_added', (snapshot) => {          Vue.set(app.todoMap, snapshot.key, snapshot.val());        });        app.todo4Ref.on('child\_changed', (snapshot) => {          app.todoMap[snapshot.key] = snapshot.val();        });        app.todo4Ref.on('child\_removed', (snapshot) => {          delete app.todoMap[snapshot.key];        });      }     </script>  </body>  </html> |

### Vue.set

Vue 앱이 시작된 후 나중에 생성된 속성은, 그 값이 변경되어도 즉시 화면이 다시 그려지지 않는다.

Vue 엔진이 그 변경을 감지할 수 없기 때문이다.

그런데, Vue.set(객체, 속성명, 값) 메소드 호출로 생성된 속성은, Vue 엔진이 그 값 변경을 감지할 수 있다.

그래서 더 이상 this.$forceUpdate() 메소드 호출이 필요 없다.