



Ton Piromplad

[Follow](#)

Learn from the mistakes of others. You can't live long enough to make them all yourself.

Dec 15 · 4 min read

Vue 2 + Firebase Realtime Database ฉบับ 101

Vue 2 + Firebase สำหรับมือใหม่โดยเฉพาะ

อันดับแรกก่อนสร้าง Project ต้องติดตั้งอะไรบ้าง

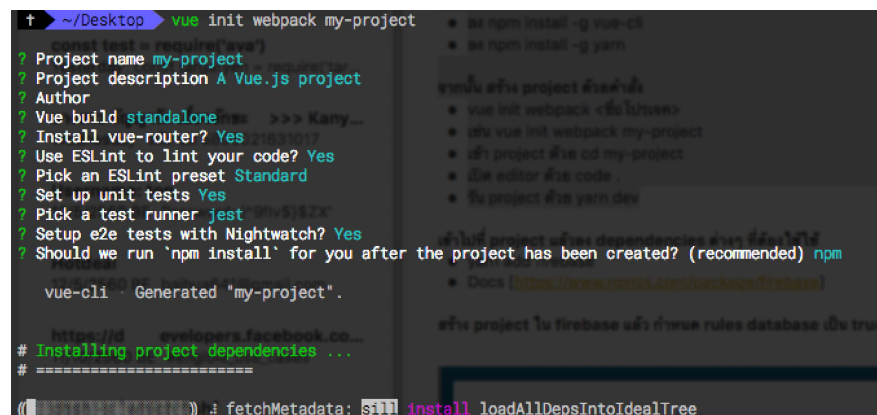
- [Node.js](#) เอาไว้รัน js
- [Visual Studio Code](#) เอาไว้เขียนโค้ด

จากนั้นก็ติดตั้ง สิ่งสำคัญที่ใช้ขึ้น Project Vue

```
npm install -g yarn
npm install -g vue-cli
```

จากนั้น สร้าง project ด้วยคำสั่ง

```
vue init webpack <ชื่อโปรเจก> // ตัวอย่างเช่น vue init webpack my-project
```

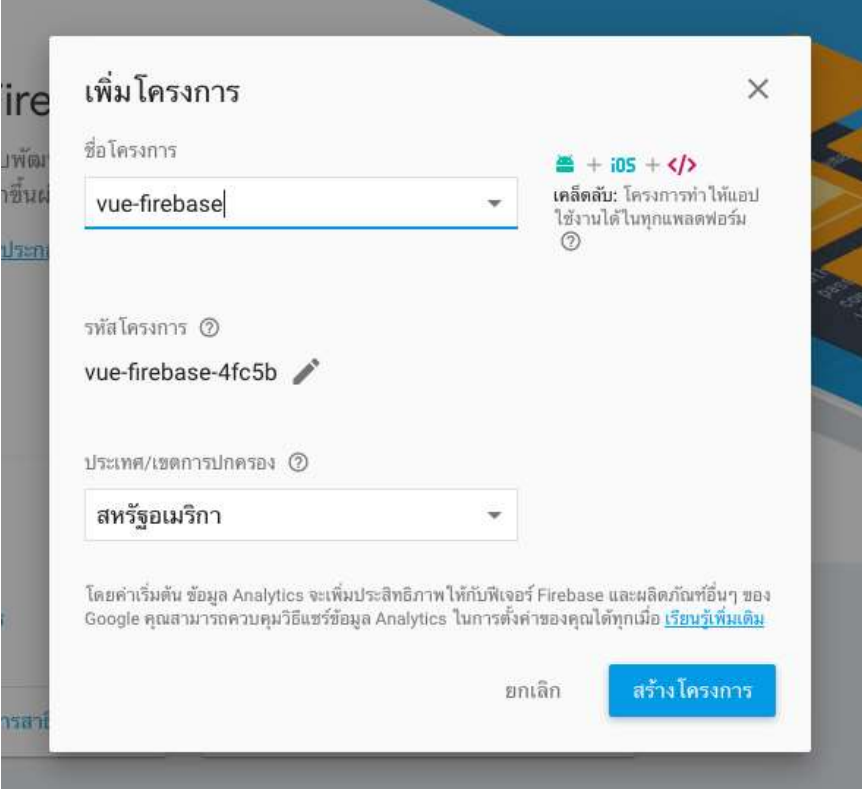


จะมีขึ้นให้เราใส่ข้อมูลต่างๆ แต่ถ้าเราไม่ใส่อะไรเลยก็ Enter รัวๆ ไปเลย

```
cd my-project          // เข้าไปที่โปรเจกต์
code .                 // เปิด editor ขึ้นมา
yarn add firebase      // เพิ่ม firebase เข้ามาใน project
yarn dev                // รัน project
```

จากนั้นไปสร้างโปรเจกต์ firebase ไว้รอเลย

ไปที่เว็บของ [firebase](#) แล้วสร้างโปรเจกต์ขึ้นมา



เพิ่มโครงการ

ชื่อโครงการ

vue-firebase

+ iOS + </>

เคล็ดลับ: โครงการทำให้แอปใช้งานได้ในทุกแพลตฟอร์ม

รหัสโครงการ

vue-firebase-4fc5b

ประเทศ/เขตการปกครอง

สหรัฐอเมริกา

โดยคำเริ่มต้น ข้อมูล Analytics จะเพิ่มประสิทธิภาพให้กับฟีเจอร์ Firebase และผลิตภัณฑ์อื่นๆ ของ Google คุณสามารถควบคุมวิธีแชร์ข้อมูล Analytics ในการตั้งค่าของคุณได้ทุกเมื่อ [เรียนรู้เพิ่มเติม](#)

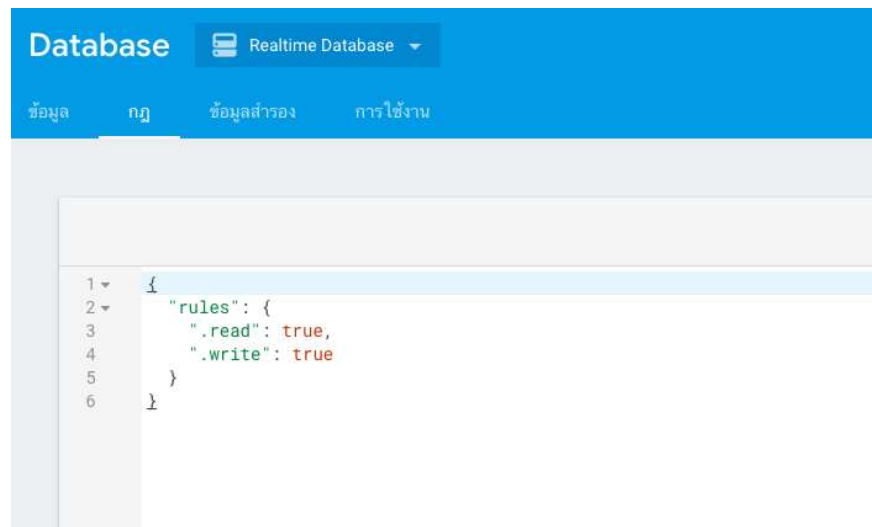
ยกเลิก สร้างโครงการ

เมื่อสร้างเสร็จแล้วไปเปิดใช้งานของ Realtime Database กันเลย



เพื่อให้เราสามารถใช้งาน database ได้โดยไม่ติด Authorize ใดๆ เราต้องไปกำหนด Rules หรือ กฎของการใช้งานก่อน โดยแก้ไขเป็นดังต่อไปนี้

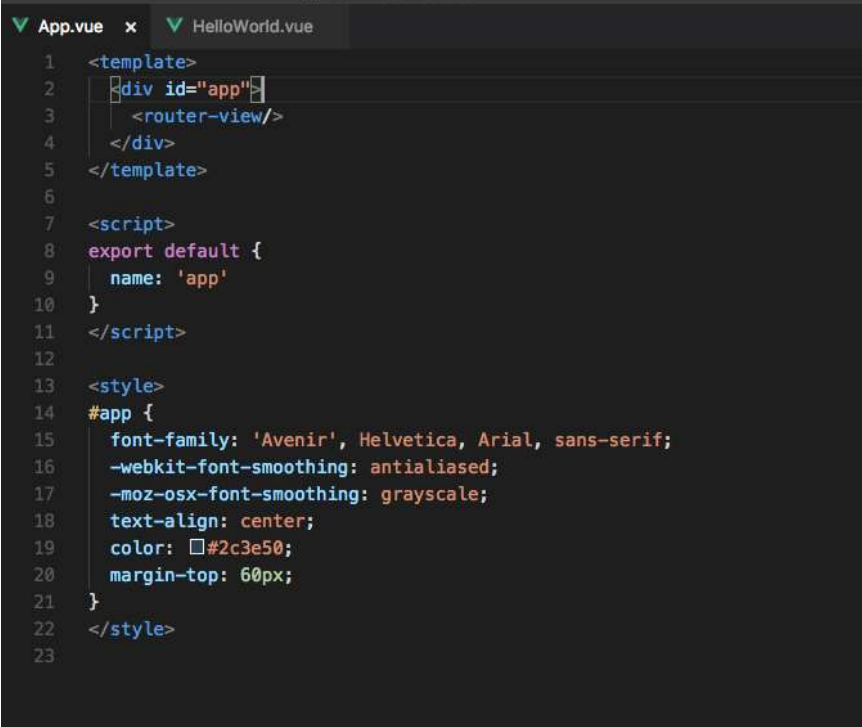
```
{
  "rules": {
    ".read": true,
    ".write": true
  }
}
```



เสร็จแล้ว พร้อมไป Coding กันเลย

ลุยๆ Code

ไปที่ ไฟล์ src/App.vue แล้วลบบางส่วนออกไปให้เหลือดังนี้



```
1 <template>
2   <div id="app">
3     <router-view/>
4   </div>
5 </template>
6
7 <script>
8 export default {
9   name: 'app'
10 }
11 </script>
12
13 <style>
14 #app {
15   font-family: 'Avenir', Helvetica, Arial, sans-serif;
16   -webkit-font-smoothing: antialiased;
17   -moz-osx-font-smoothing: grayscale;
18   text-align: center;
19   color: #2c3e50;
20   margin-top: 60px;
21 }
22 </style>
23
```

งานใหญ่รอเราอยู่ที่ไฟล์ src/components/HelloWorld.vue โดยโจทย์ที่จะลองทำเป็นการทำ สมุดโทรศัพท์แบบง่ายกันนะ

```

1  <template>
2    <div class='hello'>
3      <div>
4        <input type="text" v-model="name" placeholder="NAME"
5        <input type="text" v-model="tel" placeholder="TEL">
6        <button @click="insertToContact(tel, name)">Add</but
7      </div>
8
9      <hr>
10
11     <ul :key="key" v-for="(contact, key) in contacts">
12       <li v-if="updateKey === key">
13         <input type="text" v-model="updateName" placeholde
14         <input type="text" v-model="updateTel" placeholder
15         <button @click="updateContact(updateTel, updateNam
16       </li>
17       <li v-else>
18         {{contact.name}} : {{contact.tel}}
19         <button @click="setUpdateContact(key, contact)">Up
20         <button @click="deleteContact(key)">Delete</button
21       </li>
22     </ul>
23   </div>
24 </template>
25
26 <script>
27
28 import * as firebase from 'firebase'
29
30 var config = {
31   apiKey: '',
32   authDomain: '',
33   databaseURL: '',
34   projectId: '',
35   storageBucket: '',
36   messagingSenderId: ''
37 }
38 firebase.initializeApp(config)
39
40 var database = firebase.database()
41 var contactRef = database.ref('/contacts')

```

```
42
43 export default {
44   name: 'HelloWorld',
45   data () {
46     return {
47       contacts: {},
48       tel: '',
49       name: '',
50       updateTel: '',
51       updateName: '',
52       updateKey: ''
53     }
54   },
55   methods: {
56     insertToContact (tel, name) {
57       let data = {
58         tel: tel,
59         name: name
60       }
61       contactRef.push(data)
62       this.tel = ''
63       this.name = ''
64     },
65     setUpdateContact (key, contact) {
66       this.updateKey = key
67       this.updateTel = contact.tel
68       this.updateName = contact.name
```

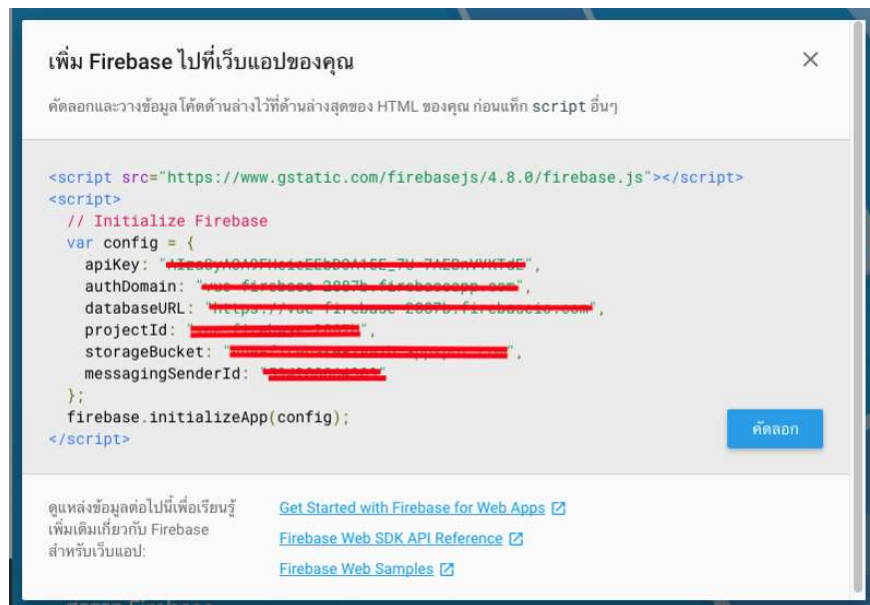
อันดับแรกต้อง import firebase เข้ามาใน project ก่อนเลย

```
import * as firebase from 'firebase'

var config = {
  apiKey: '',
  authDomain: '',
  databaseURL: '',
  projectId: '',
  storageBucket: '',
  messagingSenderId: ''
}

firebase.initializeApp(config)
```

แล้วเอา key จาก firebase มาใช้



```

9  import * as firebase from 'firebase'
10
11  var config = {
12    apiKey: 'AIzaSyA0A9FHzicEEbD0A15E_7U_7AED-WVKTdE',
13    authDomain: 'vue-firebase-2007b.firebaseio.com',
14    databaseURL: 'https://vue-firebase-2007b.firebaseio.com',
15    projectId: 'vue-firebase-2007b',
16    storageBucket: 'vue-firebase-2007b.appspot.com',
17    messagingSenderId: '594200064303'
18  }
19  firebase.initializeApp(config)
20
21  export default {
22    name: 'HelloWorld',
23    data () {
24      return {
25        msg: 'Welcome to Your Vue.js App'
26      }
27    }
28  }
29  </script>

```

จากนั้นเราก็ สร้าง Reference ให้ออนไลน์ต่อกับ firebase

```
var database = firebase.database()
var contactRef = database.ref('/contacts')
```

อันนี้แค่ สร้าง Ref ไว้ แต่ยังไม่ให้มันเป็น realtime

การทำ realtime ต้องมีตัวแปรมารองรับก่อนว่าเมื่อ data เปลี่ยนแล้วจะให้เอาข้อมูลจาก db ไปไว้ที่ไหน

```
data () {
  return {
    contacts: {},
  }
},
```

จากนั้นก็ทำให้ db นั้น realtime โดยเอาไว้ใน mounted ซึ่ง mounted จะทำงานก็ต่อเมื่อ DOM สร้างเสร็จสิ้น

```
mounted () {
  contactRef.on('value', (snapshot) => {
    this.contacts = snapshot.val()
  })
}
```

ตรง 'value' นี่คือการบอกว่าเมื่อ db ส่วนที่ ref ไว้ หรือ key contacts มีการ insert ,update หรือ delete จะให้มา trigger ที่ function นี้ และ เอาผลลัพธ์ ที่ได้ ไปเก็บไว้ที่ ตัวแปร contacts ใน data ที่สร้างมาก่อนหน้า (ต้อง เป็น snapshot.val() ด้วยนะ ตรงนี้สำคัญ)

เย้ realtime แล้ว มาทำ insert data กันเถอะ

เขียนส่วนการแสดงผล ไว้ใน template

```
<div>
  <input type="text" v-model="name" placeholder="NAME">
  <input type="text" v-model="tel" placeholder="TEL">
  <button @click="insertToContact(tel, name)">Add</button>
</div>
```

“v-model” คือ การเอาค่า จาก input ไปผูกไว้กับตัวแปร เช่น name และ tel ตามตัวอย่าง

“@click” คือ เป็นการรับ click event แล้วไปเรียก method `insertToContact` และส่งค่า `tel` กับ `name` ไปที่ method

ต้องเพิ่มตัวแปรใน data ตามที่ v-model ใช้อีก 2 ตัว คือ `name` และ `tel` ดังนี้

```
data () {  
  return {  
    contacts: {},  
    tel: '',  
    name: '',  
  }  
},
```

method นั้นเขียนดังนี้

```
insertToContact (tel, name) {  
  let data = {  
    tel: tel,  
    name: name  
  }  
  
  contactRef.push(data)  
  this.tel = ''  
  this.name = ''  
}
```

ส่วนของ `contactRef.push(data)` คือการ insert เข้า DB ส่วน `this.tel = ''` `this.name = ''` คือการ ล้างค่าไปหลังจาก insert เสร็จแล้ว

แล้วเราจะรู้ได้ไงว่าค่าเข้า db แล้ว ง่ายๆก็คือเข้าเว็บ firebase แล้วดู หรือ เขียน code ซะ

```
<ul :key="key" v-for="(contact, key) in contacts">  
  <li v-else>  
    {{contact.name}} : {{contact.tel}}  
  </li>  
</ul>
```

v-for คือการวน loop แสดงผล ซึ่งสามารถ loop ได้ทั้ง array และ object เช่น **v-for="(contact, key) in contacts"** ซึ่งประกอบด้วยกัน 2 ส่วน คือ (contact, key) และ contacts ที่ขึ้นด้วย in

ส่วนแรกคือตั้งชื่อตัวแปรที่จะเอามาใช้ในแต่ละ loop โดย contact คือชื่อใหม่ ตั้งว่าจะไรก็ได้ แต่ key คือ ถ้า data เป็น array ค่าที่ได้จะเป็น index ของ array นั้น แต่ถ้า data เป็น object ค่าที่ได้จะเป็น key ของ object นั้น

ส่วนที่ 2 คือ ตัวแปรที่เราจะเอามา วน loop นั้นเอง

เหนื่อยยัง? ถ้ายัง ต่อเลย

ทีนี้เราก็แสดงผลได้แล้ว แล้วถ้าเราอยากลบข้อมูลล่ะ ทำไงดี ไม่ยากๆ ตาม code ต่อไปนี้

```
<ul :key="key" v-for="(contact, key) in contacts">
  <li v-else>
    {{contact.name}} : {{contact.tel}}
    <button @click="deleteContact(key)">Delete</button>
  </li>
</ul>
```

เพิ่ม ปุ่ม Delete เข้าไป แล้วเมื่อ click ให้เรียก method **deleteContact** โดยส่ง key เข้าไปใน method ด้วย

```
deleteContact (key) {
  contactRef.child(key).remove()
}
```

contactRef.child(key) คือการเลือกที่จะกระทำใดๆ ที่ data ตัวนั้น เนื่องจาก data ที่ push เข้าไปใน db จะเป็นรูปแบบ key: value เมื่อเลือก child ที่จะกระทำแล้วก็ remove ได้เลย แค่นั้นเสร็จสิ้น

Insert ได้ Remove ได้ เหลือ แค่ แก้ไขแล้วนะ

อันนี้จะยากหน่อย เพราะจะต้องเอาค่าเก่ามาใช้ และ แก้ไข ไปที่ child เดิมจึงทำเพิ่มจากเดิมอีกเยอะเลยตาม code ข้างล่างนี้

```

<ul :key="key" v-for="(contact, key) in contacts">
  <li v-if="updateKey === key">
    <input type="text" v-model="updateName"
    placeholder="NAME">
    <input type="text" v-model="updateTel"
    placeholder="TEL">
    <button @click="updateContact(updateTel,
    updateName)">Save</button>
  </li>
  <li v-else>
    {{contact.name}} : {{contact.tel}}
    <button @click="setUpdateContact(key,
    contact)">Update</button>
    <button @click="deleteContact(key)">Delete</button>
  </li>
</ul>

```

ในนี้มีการสร้างการแสดงผล 2 ชุด โดยมีเงื่อนไข `v-if` เป็นสิ่งที่คอยเลือกจะทำงานอันไหน โดยเงื่อนไขว่า ถ้า `updateKey` เท่ากับ `key` จะให้แสดงผลที่ `li` แรก โดย `li` ที่มี `v-else` จะไม่ทำงาน โดย `updateKey` จะถูกเปลี่ยน เมื่อ `click` ปุ่ม `update` และส่ง `key` เข้าไป กำหนดค่า `updateKey` นั้นเอง

และมีตัวแปรต่างๆเพิ่มเข้ามาอีกเพียบเลยดังนี้

```

data () {
  return {
    contacts: {},
    tel: '',
    name: '',
    updateTel: '',
    updateName: '',
    updateKey: ''
  }
},

```

แถมยังเพิ่ม method มาอีก 2 อันตามต่อไปนี้

```

setUpdateContact (key, contact) {
  this.updateKey = key
  this.updateTel = contact.tel
  this.updateName = contact.name
},
updateContact (tel, name) {
  contactRef.child(this.updateKey).update({
    tel: tel,
    name: name
  })
}

```

```
  })  
  this.updateKey = ''  
  this.updateTel = ''  
  this.updateName = ''  
},
```

methods **setUpdateContact** เป็นการเตรียม data มาเพื่อที่จะแก้ไข

ส่วน **updateContact** คือการนำ data ที่ได้จากการ แก้ไขไป update ที่ database และการทำงานจะเหมือนกับการลบเลย เพียงแต่จะมีการ เพิ่ม data ที่จะแก้ไขเข้าไปด้วย และหาก data ที่มีอยู่มีมากกว่า key tel และ name เช่น

```
{  
  tel: 0805554444,  
  name: ttt,  
  address: BKK  
}
```

การแก้ไขที่ส่งไปตามตัวอย่างจะทำงาน update แค่ key ที่ส่งไปเท่านั้น จะไม่มีการยุ่งเกี่ยวกับ address เลย

แต่จะมีการแก้ไขข้อมูลอีกอย่างคือการ set() วิธีนี้จะเป็นการ แก้ไข ทุกอย่างในนี้ให้เป็นไปตามค่าใหม่ที่ส่ง

จบแล้ววว insert update remove กับ firebase realtime database

