



Firebase

@Chalach.mo



Develop & test your app



Realtime Database

iOS



Authentication

iOS



Test Lab



Crashlytics

iOS



Cloud Functions

iOS



Cloud Firestore

iOS



Cloud Storage

iOS



Performance Monitoring

iOS



Crash Reporting

iOS



Hosting



Grow & engage your audience



Analytics

iOS



Invites

iOS



Cloud Messaging

iOS



Predictions

iOS



AdMob

iOS



Dynamic Links

iOS



AdWords

iOS



Remote Config

iOS



App Indexing

iOS

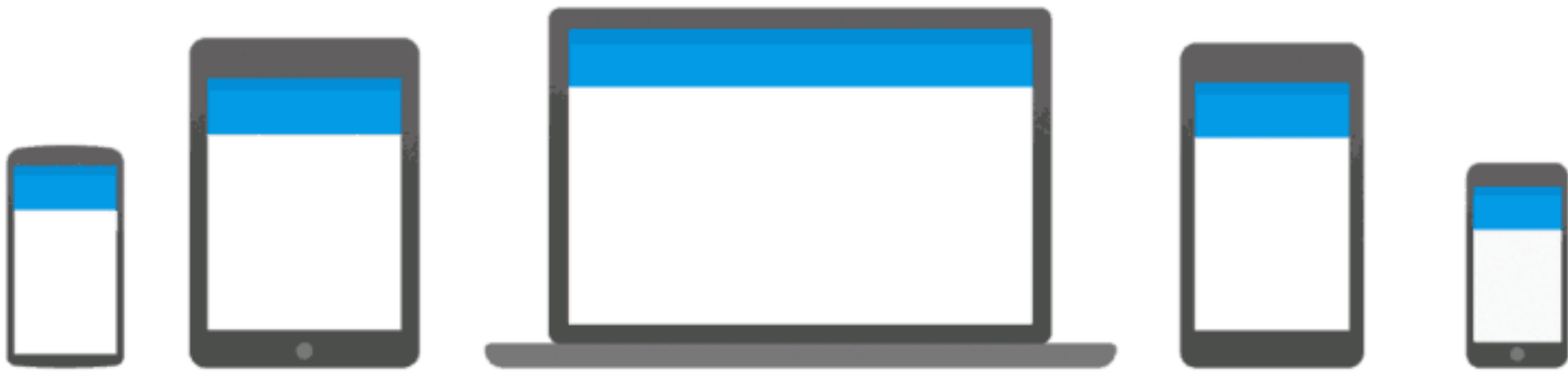


Firebase

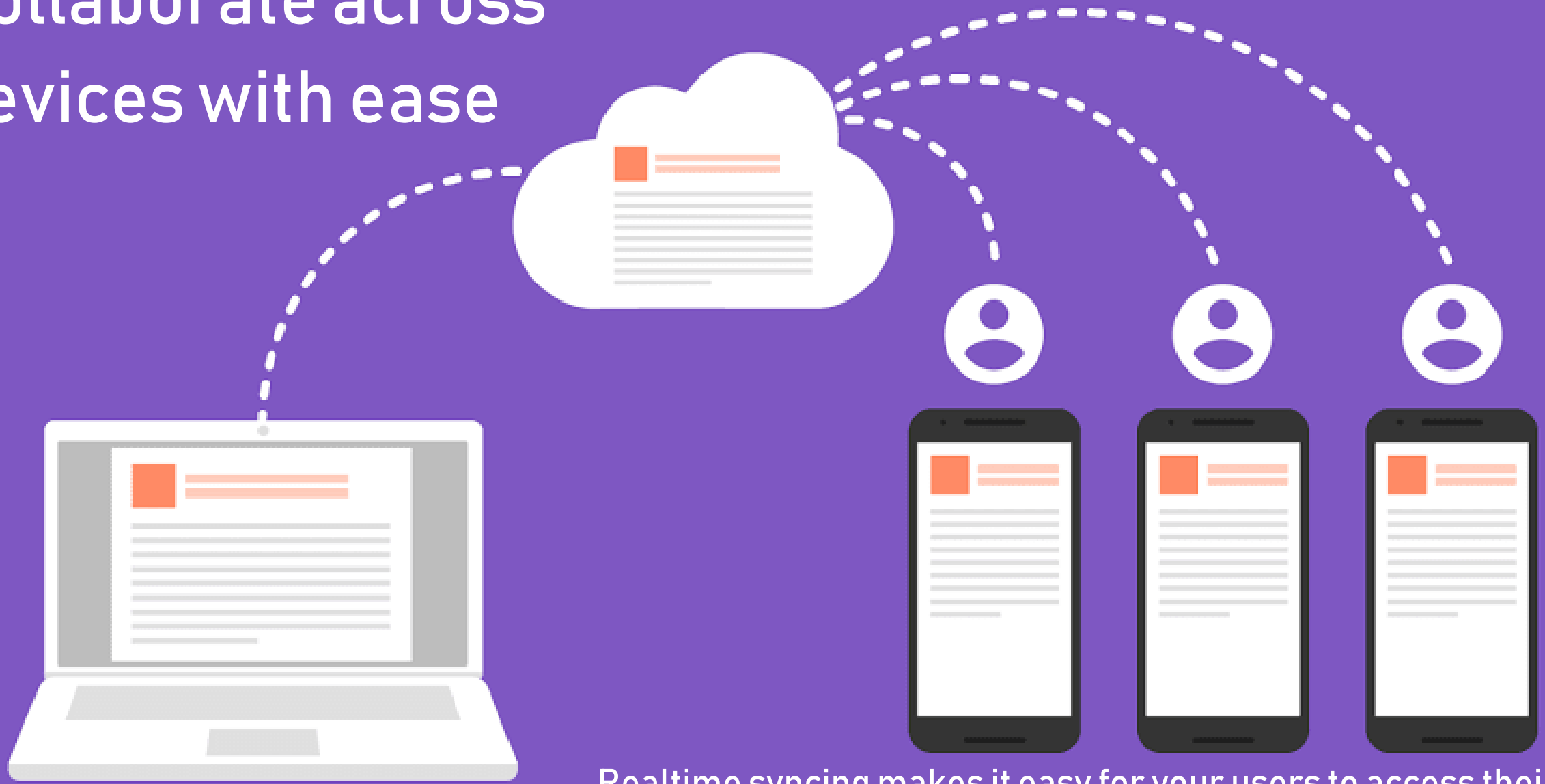
Realtime Database



- The Firebase Realtime Database is a cloud-hosted database.
- Data is stored as JSON and synchronized in realtime to every connected client.
- When you build cross-platform apps with our iOS, Android, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data.



Collaborate across devices with ease



Realtime syncing makes it easy for your users to access their data from any device: web or mobile, and it helps your users collaborate with one another.

Build serverless apps

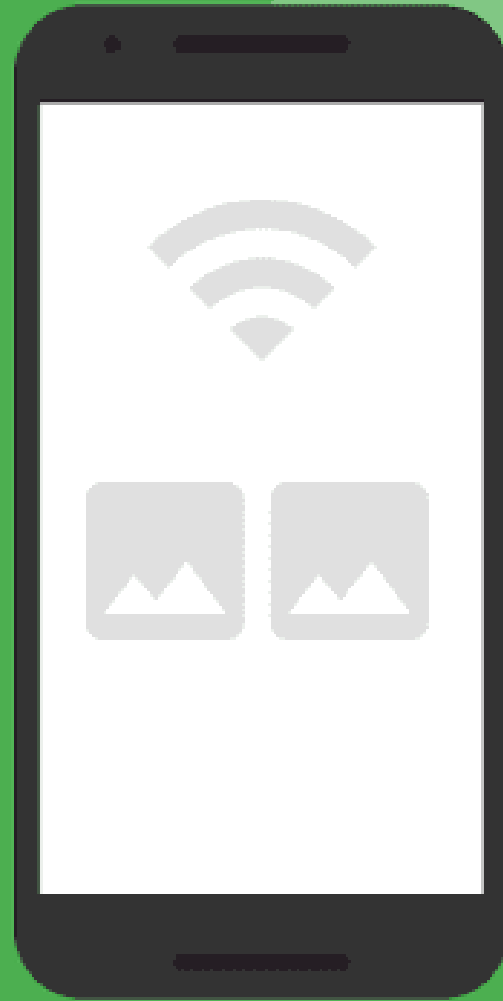
Realtime Database ships with mobile and web SDKs so you can build apps without the need of servers.



Optimized for offline use

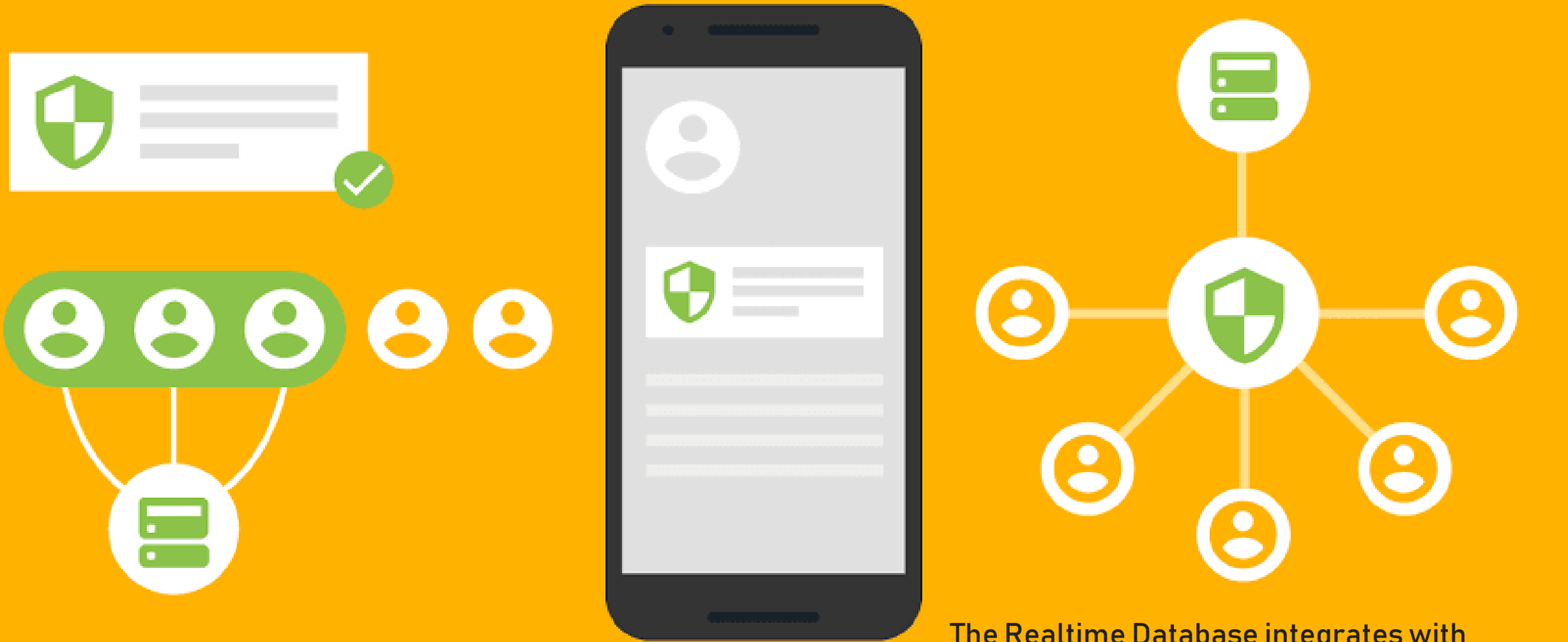


When your users go offline, the Realtime Database SDKs use local cache on the device to serve and store changes. When the device comes online, the local data is automatically synchronized.





Strong user-based security



The Realtime Database integrates with Firebase Authentication to provide simple and intuitive authentication for developers.

Configuration Firebase

The screenshot shows the Firebase console interface. On the left is a sidebar with navigation links: Project Overview, Authentication, Database, Storage, Hosting, Functions, STABILITY, ANALYTICS, GROW, and Spark. The main area displays the configuration for a web app named 'react-turbo'. A modal window is open with the title 'เพิ่ม Firebase ไปที่เว็บแอปของคุณ' (Add Firebase to your web app). The modal contains instructions in Thai and a code block for the Firebase initialization script. Below the code block are links for 'Get Started with Firebase for Web Apps', 'Firebase Web SDK API Reference', and 'Firebase Web Samples'. A 'คัดลอก' (Copy) button is also present. The background of the console shows a blue header with the Firebase logo and a user profile icon.

react-turbo

Firebase

Project Overview

DEVELOP

- Authentication
- Database
- Storage
- Hosting
- Functions

STABILITY

Crashlytics, Crash Reporting, Perf...

ANALYTICS

Dashboard, Events, Audiences, At...

GROW

Predictions, Notifications, Remot...

Spark

ฟรี \$0/เดือน

อัปเดต

เพิ่ม Firebase ไปที่เว็บแอปของคุณ

คัดลอกและวางข้อมูลโค้ดด้านล่างไว้ที่ด้านล่างสุดของ HTML ของคุณ ก่อนแท็ก script อื่นๆ

```
<script src="https://www.gstatic.com/firebasejs/4.10.1/firebase.js"></script>
<script>
  // Initialize Firebase
  var config = {
    apiKey: "AIzaSyAUzbCA07tw-U0rGXkvm0q5jyQ8PF6Z_u0",
    authDomain: "react-turbo-94b01.firebaseio.com",
    databaseURL: "https://react-turbo-94b01.firebaseio.com",
    projectId: "react-turbo-94b01",
    storageBucket: "react-turbo-94b01.appspot.com",
    messagingSenderId: "167029306316"
  };
  firebase.initializeApp(config);
</script>
```

คัดลอก

ดูแหล่งข้อมูลต่อไปนี้เป็นเพื่อเรียนรู้เพิ่มเติมเกี่ยวกับ Firebase สำหรับเว็บแอป:

- [Get Started with Firebase for Web Apps](#)
- [Firebase Web SDK API Reference](#)
- [Firebase Web Samples](#)

สมัครรับอีเมล



Firebase

Realtime Database

Read Data

Example Read with React

react-turbo-94b01

```
├── messages
│   ├── -L6Px39KEk2MaAxaTKR6
│   │   └── message: "1111111111111111"
│   ├── -L6Px3K-9GMxuM5Mxuln
│   │   └── message: "react t"
│   ├── -L6Px3Qc2-9WGIEjx8ia
│   │   └── message: "test dai loey"
│   ├── -L6Px3Z8QjgbG5hpVA8K
│   │   └── message: "4"
│   ├── -L6Px3gtRUYyRFLYR-J2
│   │   └── message: "5"
│   ├── -L6RDDP691xo4RPkmergi
│   │   └── message: "aaaaaaaaaaaaaaaaaaaaa"
│   └── -L6aHX-Q_PinkSYPNFzt
│       └── message: "asdsac"
```

```
let dbCon = firebase.database().ref('messages');
```

1 dbCon.on('value', snapshot => {
 console.log(snapshot.val());
});



Firebase

Realtime Database

Saving Data

Manual Saving(Add/Insert/Create) Data

firebase-demo ▾

ไปที่เอกสาร 🔔

Database

Realtime Database ▾

ข้อมูล กฎ ข้อมูลสำรอง การใช้งาน

🔗 https://firebest-demo.firebaseio.com/ + - ⋮

⚠️ กฎความปลอดภัยของคุณได้กำหนดเป็นสาธารณะ ทุกคนจะสามารถอ่านหรือเขียนไปที่ฐานข้อมูลของคุณได้

เรียนรู้เพิ่มเติม ปิด

firebest-demo: null ×

ชื่อ users + ×

ชื่อ id-00001 ค่า Chalach ×

ยกเลิก

เพิ่ม

Way To Saving(Add/Insert/Create) Data [Coding]

- This document covers the four methods for writing data to your Firebase Realtime Database: set, update, push, and transactions support.

4 Way To Saving

set	Write or replace data to a defined path , like <code>messages/users/<username></code>
update	Update some of the keys for a defined path without replacing all of the data
push	Add to a list of data in the database. Every time you push a new node onto a list, your database generates a unique key, like <code>messages/users/<unique-user-id>/<username></code>
transaction	Use transactions when working with complex data that could be corrupted by concurrent updates

Example Saving with React

react-turbo-94b01

```
└─ messages
  └─ -L6Px39KEk2MaAxaTKR6
      └─ message: "1111111111111111"
  └─ -L6Px3K-9GMxuM5Mxuln
      └─ message: "react t"
  └─ -L6Px3Qc2-9WGIEjx8ia
      └─ message: "test dai loey"
  └─ -L6Px3Z8QjgbG5hpVA8K
      └─ message: "4"
  └─ -L6Px3gtRUYyRFLYR-J2
      └─ message: "5"
  └─ -L6RDDP691xo4RPkmergi
      └─ message: "aaaaaaaaaaaaaaaaaaaaa"
  └─ -L6aHX-Q_PinkSYPNFzt
      └─ message: "asdsac"
```

```
let dbCon = firebase.database().ref('/messages' + messageId);
let dbConNormal = firebase.database().ref('/messages');
```

1

```
dbCon.set({
  message: xxx
});
```

2

```
dbConNormal.push({
  message: xxx
});
```

3

```
var obj = {message: xxx};
dbConNormal.child(messageId).update(obj);
```

4



Firestore

Realtime Database

Update Data

Example Update with React

react-turbo-94b01

```
└─ messages
  └─ -L6Px39KEk2MaAxaTKR6
      └─ message: "1111111111111111"
  └─ -L6Px3K-9GMxuM5Mxuln
      └─ message: "react t"
  └─ -L6Px3Qc2-9WGIEjx8ia
      └─ message: "test dai loey"
  └─ -L6Px3Z8QjgbG5hpVA8K
      └─ message: "4"
  └─ -L6Px3gtRUYyRFLYR-J2
      └─ message: "5"
  └─ -L6RDDP691xo4RPkmergi
      └─ message: "aaaaaaaaaaaaaaaaaaaaa"
  └─ -L6aHX-Q_PinkSYPNFzt
      └─ message: "asdsac"
```

```
let dbCon = firebase.database().ref('/messages');
```

1

```
var obj = {message: xxx};
dbCon.child(messageId).update(obj);
```



Firestore

Realtime Database

Delete Data

Example Delete with React

react-turbo-94b01

```
└─ messages
  └─ -L6Px39KEk2MaAxaTKR6
      └─ message: "1111111111111111"
  └─ -L6Px3K-9GMxuM5Mxuln
      └─ message: "react t"
  └─ -L6Px3Qc2-9WGIEjx8ia
      └─ message: "test dai loey"
  └─ -L6Px3Z8QjgbG5hpVA8K
      └─ message: "4"
  └─ -L6Px3gtRUYyRFLYR-J2
      └─ message: "5"
  └─ -L6RDDP691xo4RPkmergi
      └─ message: "aaaaaaaaaaaaaaaaaaaaa"
  └─ -L6aHX-Q_PinkSYPNFzt
      └─ message: "asdsac"
```

```
let dbCon = firebase.database().ref('/messages');
```

1

```
dbCon.child(messageId).remove();
```



Firestore

Realtime Database

Sorting Data

Sort Data

- You can use the Realtime Database **Query** class to retrieve data sorted by key, by value, or by value of a child. You can also filter the sorted result to a specific number of results or a range of keys or values.

Method	Usage
<code>orderByChild()</code>	Order results by the value of a specified child key or nested child path.
<code>orderByKey()</code>	Order results by child keys.
<code>orderByValue()</code>	Order results by child values.

```
let dbCon = firebase.database().ref('messages' + messageId).orderByChild('xxx');
```

```
let dbCon = firebase.database().ref('messages' + messageId).orderByKey();
```

```
let dbCon = firebase.database().ref('messages' + messageId).orderByValue();
```



Firestore

Realtime Database

Filtering Data

Filter Data

- To filter data, you can combine any of the limit or range methods with an order-by method when constructing a query.

Method	Usage
<code>limitToFirst()</code>	Sets the maximum number of items to return from the beginning of the ordered list of results.
<code>limitToLast()</code>	Sets the maximum number of items to return from the end of the ordered list of results.
<code>startAt()</code>	Return items greater than or equal to the specified key or value, depending on the order-by method chosen.
<code>endAt()</code>	Return items less than or equal to the specified key or value, depending on the order-by method chosen.
<code>equalTo()</code>	Return items equal to the specified key or value, depending on the order-by method chosen.

Different

Between SQL and



Firebase

Realtime Database

SQL Database (Relational Database)

ID	First_Name	Title	Team
10011	Debra	Programmer	Eng
10018	Yolanda	Programmer	Eng
10019	Glen	Product Designer	Mkt
10028	Casey	Account Exec	Sal
10049	Tang	Support Tech	Sup
10051	Serge	UX Designer	Eng
10059	Maria	Sales Director	Sal

Create Table

```
CREATE TABLE Customers (  
    Id INT(5) NOT NULL AUTO_INCREMENT,  
    FirstName VARCHAR(100) NOT NULL,  
    Birthday DATE,  
    Location VARCHAR(250),  
    PRIMARY KEY(Id)  
);
```

Insert Data

```
INSERT INTO Customers  
    (FirstName, Birthday, Location)  
VALUES  
    ("David", Now(), "SF");
```

Insert Data(2)

```
INSERT INTO Customers
  (FirstName, LastName, Birthday
  ,Location)
VALUES
  ("David", "East", NOW(), "SF");
```

ERROR: LastName does not exist!

Alter Table

ID	First_Name	Title	Team
10011	Debra	Programmer	Eng
10018	Yolanda	Programmer	Eng
10019	Glen	Product Designer	Mkt
10028	Casey	Account Exec	Sal
10049	Tang	Support Tech	Sup
10051	Serge	UX Designer	Eng
10059	Maria	Sales Director	Sal

```
ALTER TABLE Customers
ADD COLUMN LastName
VARCHAR(100) NOT NULL
```

Alter Table

```
ALTER TABLE Customers  
ADD COLUMN LastName  
VARCHAR(100) _____
```


Alter Table

```
ALTER TABLE Customers  
MODIFY LastName VARCHAR(100)  
NOT NULL
```

Firestore Database (NoSQL Database)

```
{  
  "customers": {  
    "customer_one": {  
      "firstName": "David",  
      "birthday": 1475189812156,  
      "location": "SF"  
    }  
  }  
}
```

```
customers.child("customer_one").child("lastname").setValue("New Lastname")
```

Understand Firebase Realtime Database Rules

- Firebase Realtime Database Rules determine who has read and write access to your database, how your data is structured, and what indexes exist. These rules live on the Firebase servers and are enforced automatically at all times. Every read and write request will only be completed if your rules allow it. By default, your rules are set to allow only authenticated users full read and write access to your database. This is to protect your database from abuse until you have time to customize your rules or set up authentication.
- Firebase Database Rules have a JavaScript-like syntax and come in four types:

Rule Types	
<code>.read</code>	Describes if and when data is allowed to be read by users.
<code>.write</code>	Describes if and when data is allowed to be written.
<code>.validate</code>	Defines what a correctly formatted value will look like, whether it has child attributes, and the data type.
<code>.indexOn</code>	Specifies a child to index to support ordering and querying.

Rules .read .write

firebase-demo

Database Realtime Database

ข้อมูล กฎ ข้อมูลสำรอง การใช้งาน

★ กฎความปลอดภัยเริ่มต้นกำหนดให้มีการตรวจสอบสิทธิ์ผู้ใช้ เรียนรู้เพิ่มเติม ปิด

```
1 {
2   "rules": {
3     ".read": "auth != null",
4     ".write": "auth != null"
5   }
6 }
```

เครื่องมือจำลอง

กฎความปลอดภัยของคุณได้กำหนดเป็นสาธารณะ ทุกคนจะสามารถอ่านหรือเขียนไปพื้นฐานข้อมูลของคุณได้ เรียนรู้เพิ่มเติม ปิด

```
1 {
2   "rules": {
3     ".read": true,
4     ".write": true
5   }
6 }
```

Rules .validate

```
{  
  "rules": {  
    "customers": {  
      "$uid": {  
        ".validate": "newData.child('firstName').isString() &&  
                      newData.child('birthday').isNumber() &&  
                      newData.child('location').isString()"  
      }  
    }  
  }  
}
```

SQL Database



Convert TO

Firebase

Realtime Database

SQL Structure Example

Users	
ID	int
Surname	varchar(4)
FirstName	varchar(100)
LastName	varchar(100)
Age	int
Location	varchar(100)



Attendees	
ID	int
UID	int
EventId	int



Events	
ID	int
Name	varchar(20)
Description	varchar(120)
Time	varchar(15)
Category	varchar(20)


NoSQL Structure Example

```
{
  "users": {
    "1": {
      "name": "David"
    },
    "9": {
      "name": "Alice"
    }
  },
  "events": {
    "fm": {
      "name": "Firebase Meetup",
      "date": 983275235320,
      "attendees": {
        "1": "David",
        "9": "Alice"
      }
    }
  }
}
```

```
{
  "users": {
    "1": {
      "name": "David"
    },
    "9": {
      "name": "Alice"
    }
  },
  "events": {
    "fm": {
      "name": "Firebase Meetup",
      "date": 983275235320,
    }
  },
  "eventAttendees": {
    "fm": {
      "1": "David",
      "9": "Alice"
    }
  }
}
```


SELECT

```
SELECT *  
FROM Events  
WHERE Name == "Firebase Meetup";
```

```
const db = firebase.database();  
const eventsRef = db.child('events');  
  
eventsRef.orderFunction().queryFunction();  
  
eventsRef.orderByKey().limitToFirst(10);
```

SELECT (2)

```
SELECT event.Name as EventName
      ,event.Date as EventDate
      ,user.Name as AttendeeName
FROM Events as event
INNER JOIN Attendees as a
      ON e.Id === a.EventId
INNER JOIN Users as user
      ON u.Uid = a.Uid
WHERE e.Id == 4;
```

```
const db = firebase.database();
const events = db.child('events/fm');
const attendees = db.child('eventAttendees/fm');

events.on('value', snap => {
  // render data to HTML
});

attendees.on('child_added', snap => {
  // append attendees to list
});
```

SELECT (3)

```
const db = firebase.database();
const events = db.child('events');
const query = events
    .orderByChild('name')
    .equalTo('Firebase Meetup')
    .limitToFirst(1);

query.on('value', snap => {
    // render data to HTML
});
```



Firebase

Realtime Database

Querying Data

SQL Database

uid	name	email	age	location
1	Jirawat	jirawatee@gmail.com	18	Surat Thani
2	FirebaseThailand	firebasethailand@gmail.com	32	Bangkok

Firestore Database

```
{
  "users" : {
    "1" : {
      "age" : 18,
      "email" : "jirawatee@gmail.com",
      "location" : "SuratThani",
      "name" : "Jirawat"
    },
    "2" : {
      "age" : 32,
      "email" : "firebasethailand@gmail.com",
      "location" : "Bangkok",
      "name" : "FirebaseThailand"
    }
  }
}
```

SQL Query

1. Select a user by UID

- `SELECT * FROM Users WHERE uid = 1`

2. Find a user by email address

- `SELECT * FROM users WHERE email = 'firebasethailand@gmail.com'`

3. Limit to 10 users

- `SELECT * FROM users LIMIT 10`

4. Get all users names that start with 'F'

- `SELECT * FROM users WHERE name LIKE 'F%'`

5. Get all users who are age less than 25

- `SELECT * FROM user WHERE age < 25`

6. Get all users who are age greater than 25

- `SELECT * FROM users WHERE age >= 25;`

7. Get all users who are age between 18 and 32

- `SELECT * FROM users WHERE age >= 18 && age <= 32;`

8. Get all users who are 32 and live in Bangkok

- `SELECT * FROM users WHERE age = 32 && Location = 'Bangkok';`

Firestore Query

```
const mRootRef = firebase.database().ref('...');
```

1. Select a user by UID

- `mRootRef.child("users").child("1");`

2. Find a user by email address

- `mRootRef.child("users").orderByChild('email').equalTo("firebasethailand@gmail.com");`

3. Limit to 10 users

- `mRootRef.child("users").orderByKey().limitToFirst(10);`

4. Get all users names that start with 'F'

- `mRootRef.child("users").orderByChild("name").startAt("F");`

5. Get all users who are age less than 25

- `mRootRef.child("users").orderByChild("age").endAt(25);`

6. Get all users who are age greater than 25

- `mRootRef.child("users").orderByChild("age").startAt(25);`

7. Get all users who are age between 18 and 32

- `mRootRef.child("users").orderByChild("age").startAt(18).endAt(32);`

8. Get all users who are 32 and live in Bangkok

- `mRootRef.child("users").orderByChild("age").equalTo(32)
 .orderByChild("location").equalTo("Bangkok");`



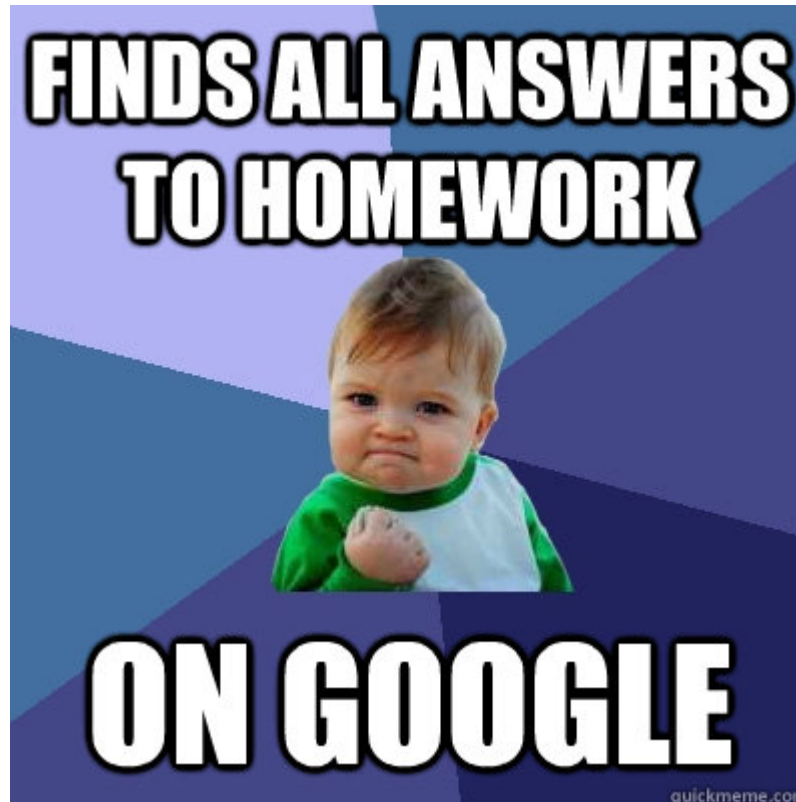
Firebase

Realtime Database

Comment Workshop

Homework

1. -



Reference

- <https://firebase.google.com/docs>
- <https://developers.ascendcorp.com/รู้จัก-firebase-realtime-database-ตั้งแต่-zero-จนเป็น-hero-5d09210e6fd6๗>
- <https://developers.ascendcorp.com/มาทำความเข้าใจกับ-sql-database-และ-firebase-database-กันเถอะ-4aed4a19e339>
- <https://appdividend.com/2017/07/22/react-firebase-tutorial/>
- React: Functional Web Development with React and Redux 1st Edition, Kindle Edition