Software Studio

軟體設計與實驗

Server & Firebase

Hung-Kuo Chu

Department of Computer Science

National Tsing Hua University

CS2410

Codeblock Conventions

HTML5 Program
JavaScript Program
JSON or Rule

Basic Web Workflow

Client

Request: Website

Response: IP address

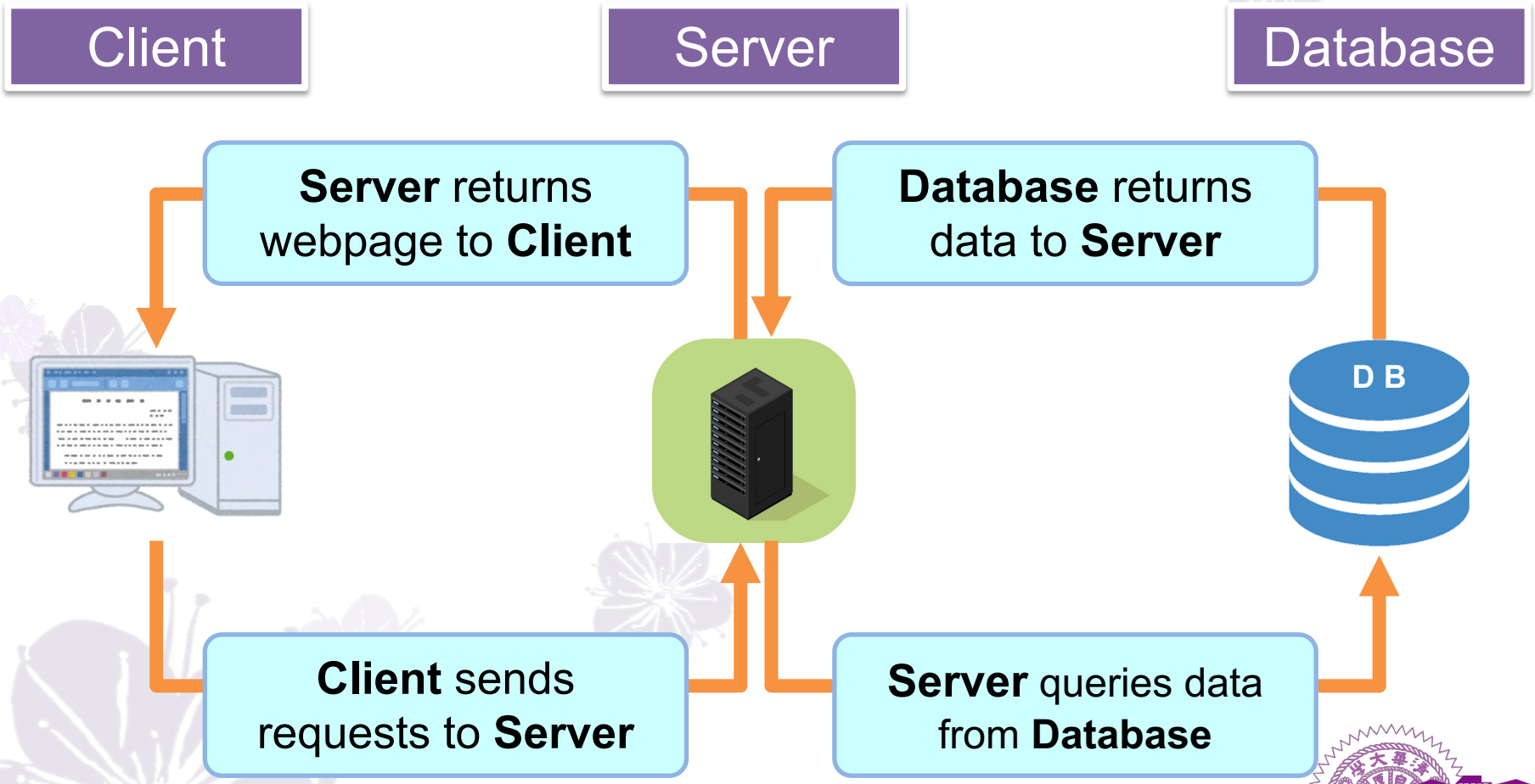
DNS Server

Request: IP address

Response: Data

Main Server

Web Architecture



Client

- **Client (Frontend)**
  - Send requests to Server.
  - Process/Display data received from Server via Web Browser.
  - HTML + CSS + Javascript can build a good interactive website
  - We cannot keep any data
  - We cannot collect data

Server

- **Server (backend)**
  - save data in server
  - do some private function
  - collect data and display to client
- **Database**
  - save some txt data
- **Storage**
  - save some multi media data

What is Firebase?

- Firebase is an application cloud development platform built by Google.
- Help developers **quickly set up** backend services in the cloud which effectively shorten the application development time.

Firebase

Why Use Firebase?

- A powerful **real-time database** which makes it an excellent candidate to drive **multi-player games.**
- Full document storage, analytics, hosting, etc.
- Using JavaScript API and provide user side high security.

Who are Using Firebase?

The New York Times

npr one

shazam

duolingo

Alibaba.com

lyft

venmo

The Economist

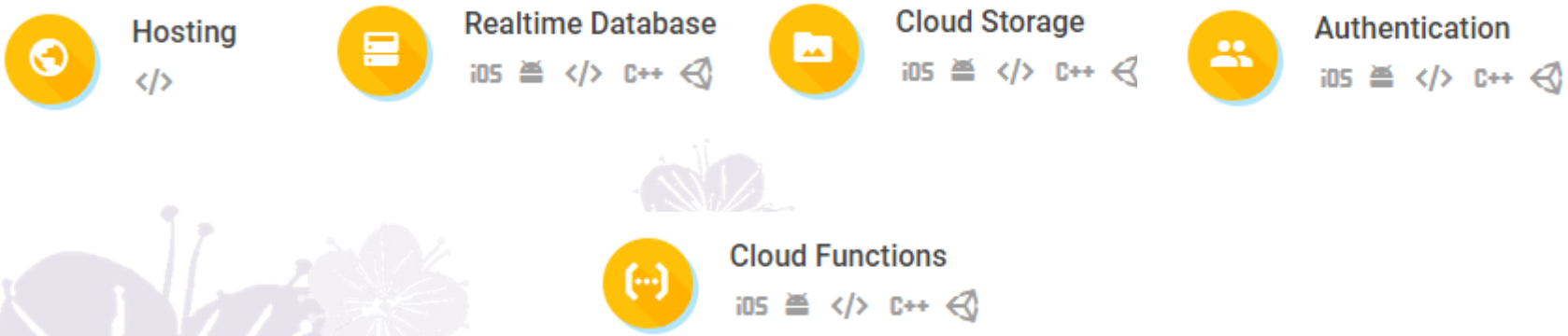
trivago

Ctrip

wattpad

What Will You Learn?

- Firebase has a lot of features and we will teach you



Step by step

1. Create a new Firebase Project
2. Create a new Web App on Firebase Console
3. Activate Firebase Features
   (Authentication, Database, Hosting…)
4. Install the Firebase CLI
5. Import Firebase SDK
6. Run Your Webpage

1. Create a new Firebase Project

- Go to Firebase Console (need a Google account) and add a project

This id will be used in database URL and webpage domain. Please record it!

1. Create a new Firebase Project

- Then finish remaining steps.

2. Create a new Web App on Firebase Console

Initialization Configuration

Pick a name for your application

Initialization Configuration

2 新增 Firebase SDK

使用 npm　使用 <script> 標記

如果你已使用 npm 和 webpack 或 Rollup 等模組整合工具，則可執行下列指令來安裝最新版 SDK：

```
$ npm install firebase
```

請初始化 Firebase，接著即可開始將 SDK 套用至要使用的產品。

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
const firebaseConfig = {
```

注意事項：這個選項會使用模組 JavaScript SDK，因此 SDK 的大小得以縮減。

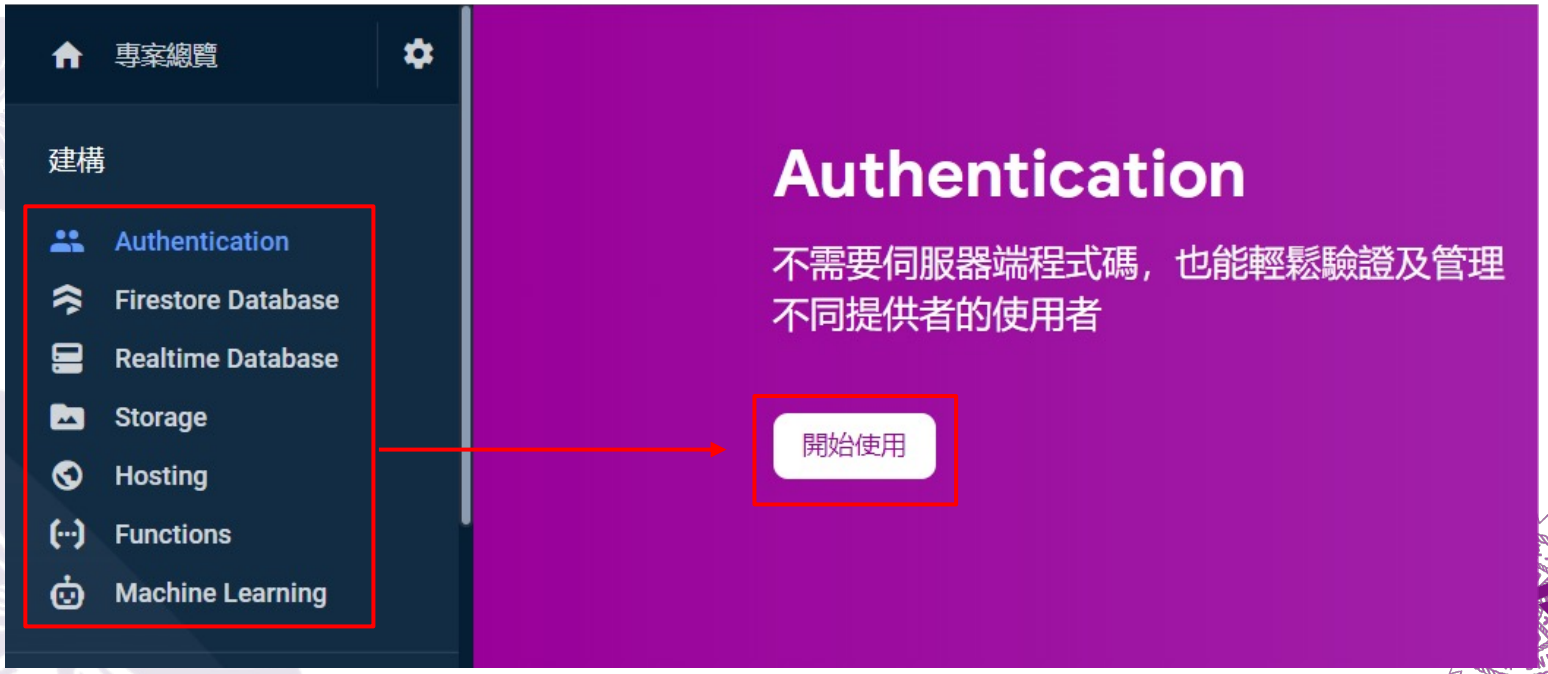
如要進一步瞭解適用於網頁應用程式的 Firebase，請查看下列資源：開始使用、Web SDK API 參考資料、使用範例

前往控制台

We will import Firebase SDK later

3. Activate Firebase Features

- Activate the firebase features you want to use one by one. (Authentication, Database, Hosting…)



3. Activate Firebase Features

3. Activate Firebase Features

Hosting

利用安全的全球內容傳遞聯播網，在彈指之間部署網路版和行動版網頁應用程式

開始使用

1 安裝 Firebase CLI

如要透過 Firebase 託管功能來代管您的網站，則必須使用 Firebase CLI 這項指令列工具。

執行下列 npm 指令，藉此安裝 CLI 或更新至最新版 CLI。

```
$ npm install -g firebase-tools
```

無法順利執行操作嗎？您不妨查看 Firebase CLI 參考資源或變更您的 npm 權限

☐ 一併顯示將 Firebase JavaScript SDK 新增至網頁應用程式的步驟
這項 SDK 含有 Cloud Firestore、Authentication、Performance Monitoring 等功能。您可以立即新增這項 SDK 或之後再新增。

下一步

4. Install the Firebase CLI

- Download node.js, 12.16.2 LTS
- Run the command at your project directory and complete all settings
  1. npm install -g firebase-tools
  2. firebase login
  3. firebase init

Firebase Initialization

- Press <space> to select Firebase features.

```
? Are you ready to proceed? Yes
? Which Firebase features do you want to set up for this directory? Press Space to s
elect features, then Enter to confirm your choices. (Press <space> to select, <a> to
 toggle all, <i> to invert selection, and <enter> to proceed)
 (*) Realtime Database: Configure a security rules file for Realtime Database and (o
ptionally) provision default instance
 ( ) Firestore: Configure security rules and indexes files for Firestore
 ( ) Functions: Configure a Cloud Functions directory and its files
>(*) Hosting: Configure files for Firebase Hosting and (optionally) set up GitHub Ac
tion deploys
 ( ) Hosting: Set up GitHub Action deploys
 ( ) Storage: Configure a security rules file for Cloud Storage
 ( ) Emulators: Set up local emulators for Firebase products
(Move up and down to reveal more choices)
```

Firebase Initialization

```
? Please select an option:
> Use an existing project
  Create a new project
  Add Firebase to an existing Google Cloud Platform project
  Don't set up a default project
```

- On default Firebase project selection, select your project
  - If you could not see the project you just create on firebase's console, you can directly add after initialization, so just select [don't setup a default project]
  - Run "**firebase projects:list**" and select one ID
  - Use "**firebase use [ID]**" to link your project

Firebase Initialization

- Use default setting in other selections
- Notice that if your application wants to use any firebase features, it need to be activated first. Otherwise, you can’t set it in ***firebase init***.

Firebase Initialization

```
? Which Firebase CLI features do you want to setup for this folder? Press Space to select features, then Enter to confirm your choices. Database: Deploy Firebase Realtime Database Rules, Functions: Configure and deploy Cloud Functions,
Hosting: Configure and deploy Firebase Hosting sites, Storage: Deploy Cloud Storage security rules

=== Project Setup

First, let's associate this project directory with a Firebase project.
You can create multiple project aliases by running firebase use --add,
but for now we'll just set up a default project.

? Select a default Firebase project for this directory: [don't setup a default project]

=== Database Setup

Firebase Realtime Database Rules allow you to define how your data should be
structured and when your data can be read from and written to.

? What file should be used for Database Rules? database.rules.json
+  Database Rules for  have been downloaded to database.rules.json.
Future modifications to database.rules.json will update Database Rules when you run
firebase deploy.

=== Functions Setup

A functions directory will be created in your project with a Node.js
package pre-configured. Functions can be deployed with firebase deploy.

? What language would you like to use to write Cloud Functions? JavaScript
? Do you want to use ESLint to catch probable bugs and enforce style? No
+  Wrote functions/package.json
+  Wrote functions/index.js
+  Wrote functions/.gitignore
? Do you want to install dependencies with npm now? Yes

> grpc@1.19.0 install C:\Users\Eric\Desktop\Firebase Testing\functions\node_modules\grpc
> node-pre-gyp install --fallback-to-build --library=static_library

node-pre-gyp WARN Using needle for node-pre-gyp https download
[grpc] Success: "C:\Users\Eric\Desktop\Firebase Testing\functions\node_modules\grpc\src\node\extension_binary\node-v64-win32-x64-unknown\grpc_node.node" is installed via remote

> protobufjs@6.8.8 postinstall C:\Users\Eric\Desktop\Firebase Testing\functions\node_modules\protobufjs
> node scripts/postinstall


> firebase-functions@2.2.1 postinstall C:\Users\Eric\Desktop\Firebase Testing\functions\node_modules\firebase-functions
> node ./upgrade-warning


======= WARNING! =======

This upgrade of firebase-functions contains breaking changes if you are upgrading from a version below v1.0.0.

To see a complete list of these breaking changes, please go to:

https://firebase.google.com/docs/functions/beta-v1-diff

npm notice created a lockfile as package-lock.json. You should commit this file.
added 310 packages from 222 contributors and audited 1031 packages in 7.971s
found 0 vulnerabilities


=== Hosting Setup

Your public directory is the folder (relative to your project directory) that
will contain Hosting assets to be uploaded with firebase deploy. If you
have a build process for your assets, use your build's output directory.

? What do you want to use as your public directory? public
? Configure as a single-page app (rewrite all urls to /index.html)? No
+  Wrote public/404.html
+  Wrote public/index.html

=== Storage Setup

Firebase Storage Security Rules allow you to define how and when to allow
uploads and downloads. You can keep these rules in your project directory
and publish them with firebase deploy.

? What file should be used for Storage Rules? storage.rules

i  Writing configuration info to firebase.json...
i  Writing project information to .firebaserc...
i  Writing gitignore file to .gitignore...

+  Firebase initialization complete!
```

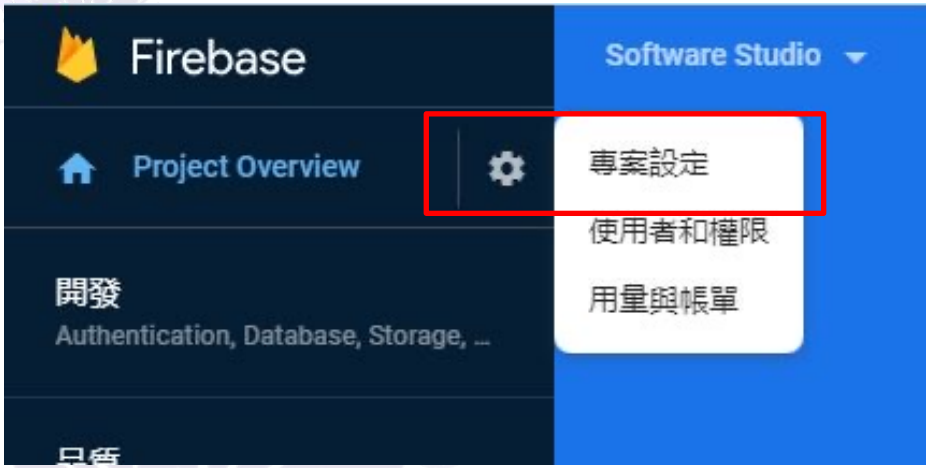
5. Import Firebase SDK

- Add Firebase SDK to your .html
  - firebase: Firebase core
  - firebase-app: The core Firebase client
  - firebase-auth: Firebase Authentication
  - firebase-database: Firebase Realtime Database
  - firebase-storage: Cloud Storage

```
<!-- Firebase App is always required and must be first -->
<script src="https://www.gstatic.com/firebasejs/7.8.2/firebase-app.js"></script>
<script src="https://www.gstatic.com/firebasejs/7.8.2/firebase-auth.js"></script>
<script src="https://www.gstatic.com/firebasejs/7.8.2/firebase-database.js"></script>
<script src="https://www.gstatic.com/firebasejs/7.8.2/firebase-firestore.js"></script>
<script src="https://www.gstatic.com/firebasejs/7.8.2/firebase-functions.js"></script>
```

5. Import Firebase SDK

- Go to Settings to get the `firebaseConfig`.



5. Import Firebase SDK

- Add Firebase Config to your config.js
- **Make sure there is databaseURL.**

```
<script src="config.js"></script>
```

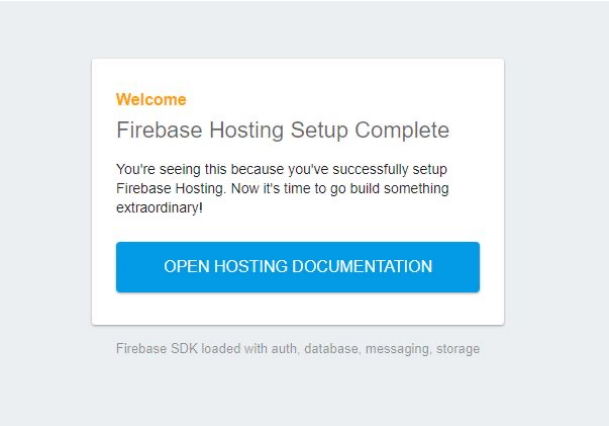
```
// Initialize Firebase
// TODO: Replace with your project's customized code snippet
const firebaseConfig = {
   apiKey: "<API_KEY>",
   authDomain: "<PROJECT_ID>.firebaseapp.com",
   databaseURL: "https://<DATABASE_NAME>.firebaseio.com",
   projectId: "<PROJECT_ID>",
   storageBucket: "< PROJECT_ID >.appspot.com",
   messagingSenderId: "<SENDER_ID>",
   appId: "<APPLICATION_ID>",
   measurementId: “<AnalyticsSDK>"
};
const app = initializeApp(firebaseConfig);
```

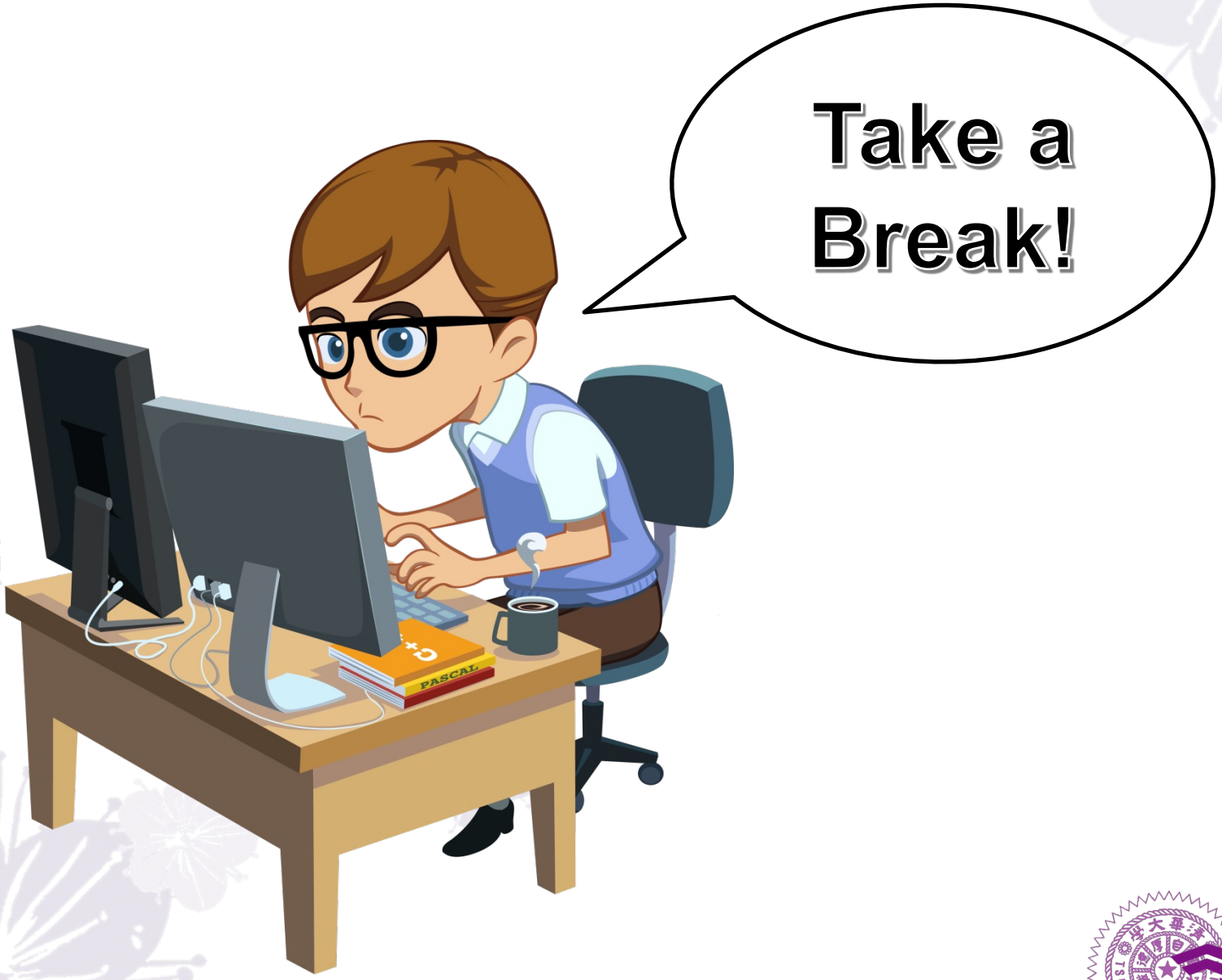
5. Import Firebase SDK

- If databaseURL is missing. You can find it in firebase console.

6. Run Your Webpage

- You can run a local server for testing
- Put all your web files(.html, .js, images…etc.) under the public folder
- Run the command at your project directory
  - firebase serve
- Your webpage will run on http://localhost:5000




Take a Break!
C++
PASCAL

Hosting

- Firebase Hosting has some key capabilities
  - Served over a secure connection
  - Fast content delivery
  - Rapid deployment
  - One-click rollbacks

Hosting

- Run the command at your project directory
  - firebase deploy
- Your app will be deployed to the domain <YOUR-FIREBASE-APP>.firebaseapp.com

Authentication

- You can use Firebase Authentication to allow users to sign in your app using one or more sign-in methods

Authentication

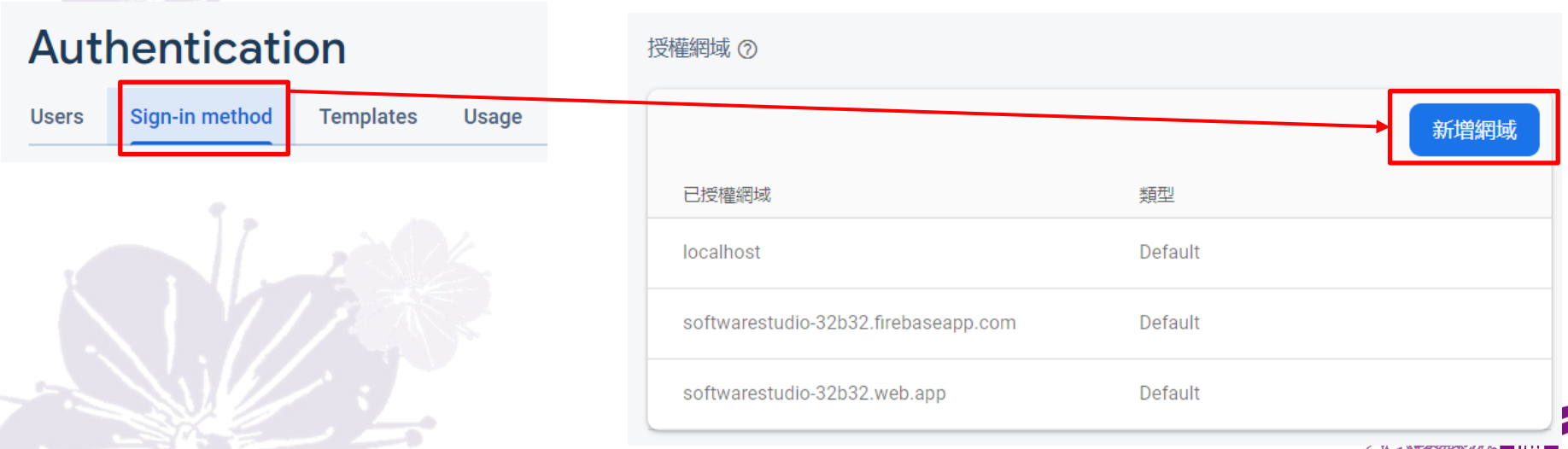
1. Add your web page domain to Authorized Domains. (You can skip this step if you are using firebase hosting.)

2. Enable the sign-in method

3. Implement authentication with firebase.auth()

Authorized Domains

- Before adding Firebase Authorized to your web page, you need to add your web page domain in firebase console.
- For example, your gitlab website url.



Email/Password Sign-In

- Enable the Email/password sign-in method

Email/Password Sign-In

- Sign up new users

```
firebase.auth().createUserWithEmailAndPassword(email, password)
  .then((userCredential) => {
    // Signed in
    var user = userCredential.user;
    // ...
  })
  .catch((error) => {
    var errorCode = error.code;
    var errorMessage = error.message;
    // …
  });
```

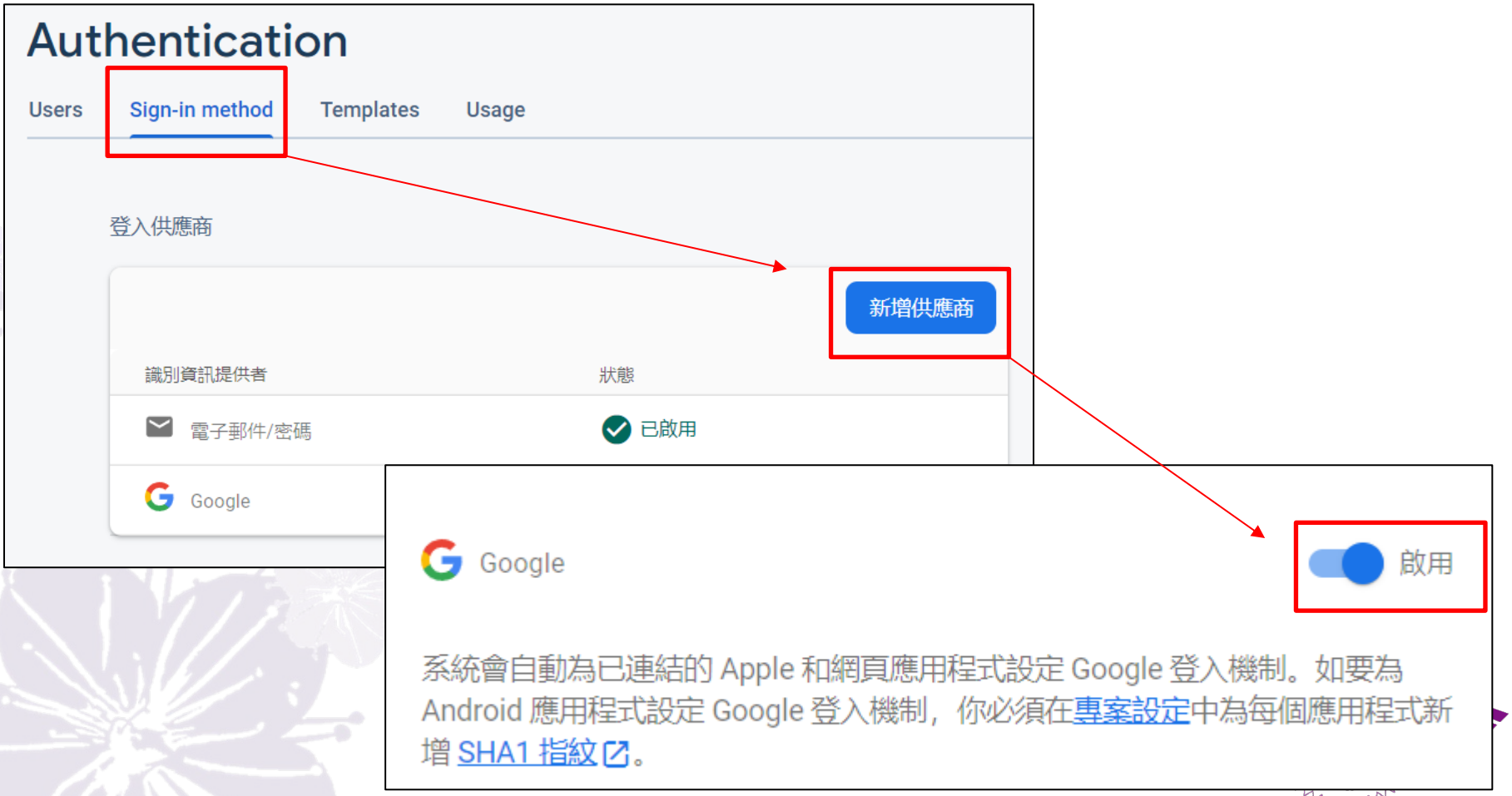
Email/Password Sign-In

- Sign in existing users

```
firebase.auth().signInWithEmailAndPassword(email, password)
  .then((userCredential) => {
    // Signed in
    var user = userCredential.user;
    // ...
  })
  .catch((error) => {
    var errorCode = error.code;
    var errorMessage = error.message;
    // ...
  });
```

Google Sign-In

- Enable the Google sign-in method



Google Sign-In

- Create an instance of the Google provider object

```
var provider = new firebase.auth.GoogleAuthProvider();
```

Google Sign-In

- Sign in with a pop-up window

```
firebase.auth().signInWithPopup(provider).then(function(result) {
  // This gives you a Google Access Token. You can use it to access the Google API.
  var token = result.credential.accessToken;
  // The signed-in user info.
  var user = result.user;
  // ...
}).catch(function(error) {
  // Handle Errors here.
  var errorCode = error.code;
  var errorMessage = error.message;
  // The email of the user's account used.
  var email = error.email;
  // The firebase.auth.AuthCredential type that was used.
  var credential = error.credential;
});
```

Google Sign-In

- Sign in by redirecting to the sign-in page

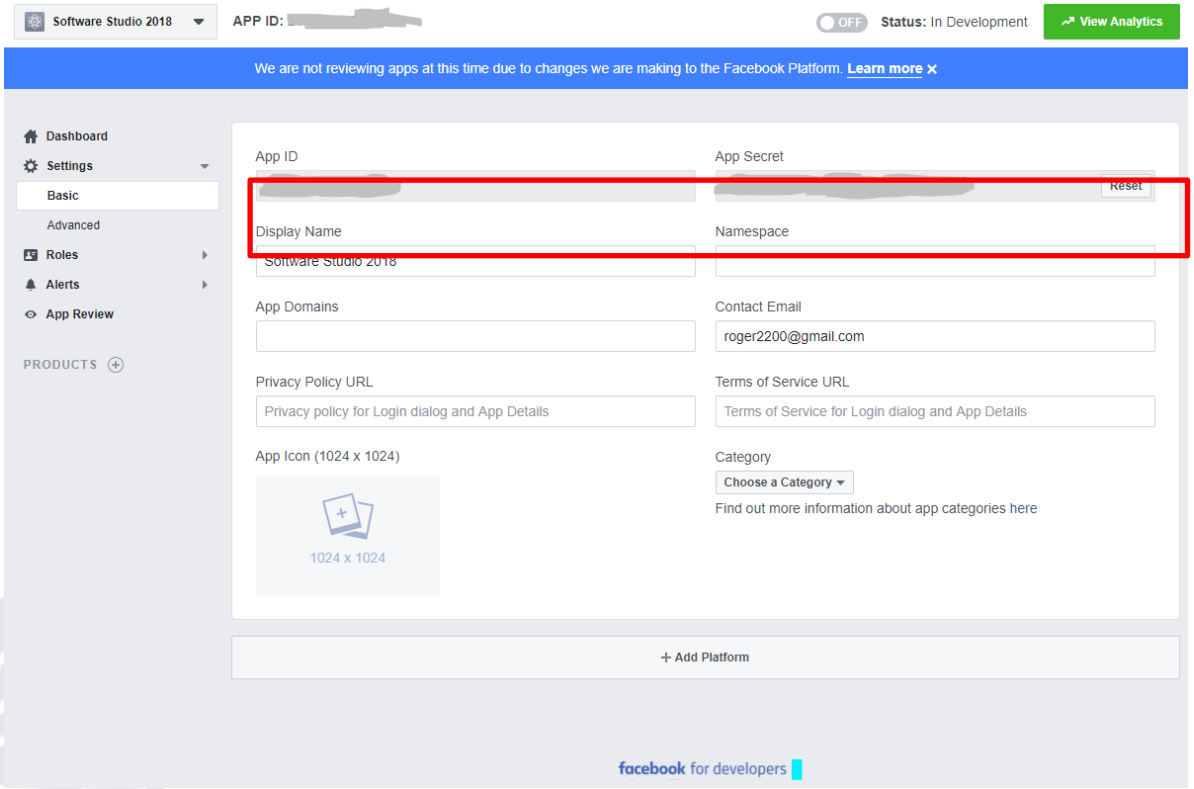
```
firebase.auth().signInWithRedirect(provider);
firebase.auth().getRedirectResult().then(function(result) {
  if (result.credential) {
    // This gives you a Google Access Token. You can use it to access the Google API.
    var token = result.credential.accessToken;
    // ...
  }
  // The signed-in user info.
  var user = result.user;
}).catch(function(error) {
  // Handle Errors here.
  var errorCode = error.code;
  var errorMessage = error.message;
  // The email of the user's account used.
  var email = error.email;
  // The firebase.auth.AuthCredential type that was used.
  var credential = error.credential;
});
```

Facebook Sign-In

- On the Facebook for Developers site, create a new App

Facebook Sign-In

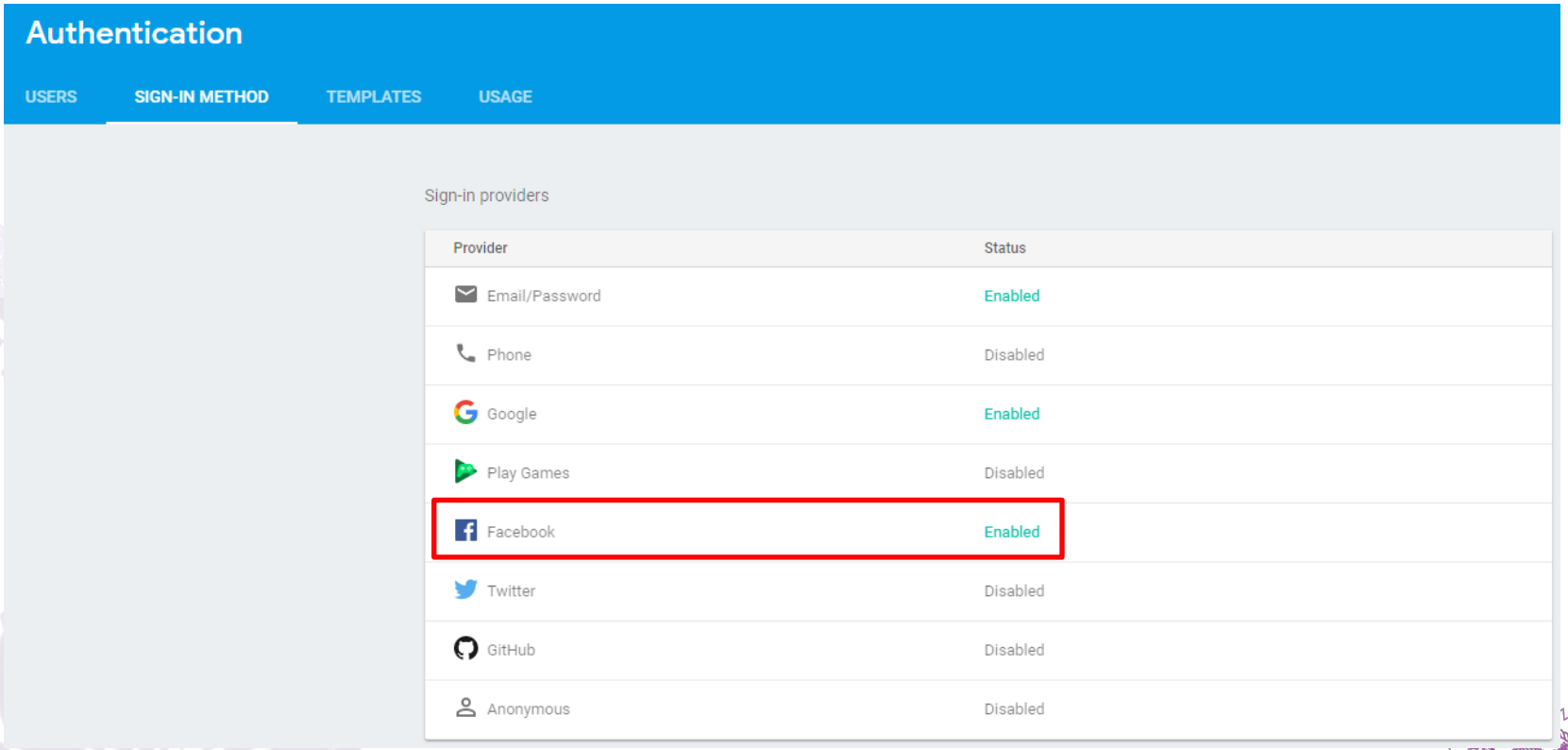
- Get the **App ID** and an **App Secret** for your app



(in Facebook for Developer)

Facebook Sign-In

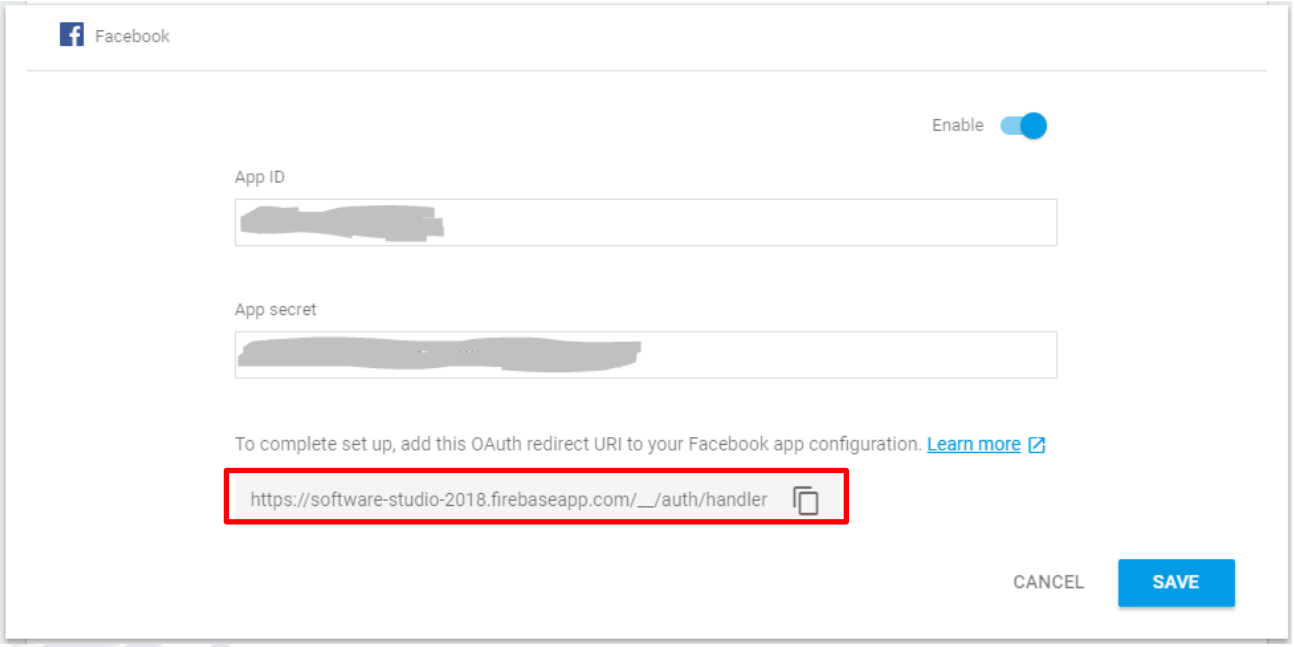
- Enable the Facebook sign-in method



(in firebase page)

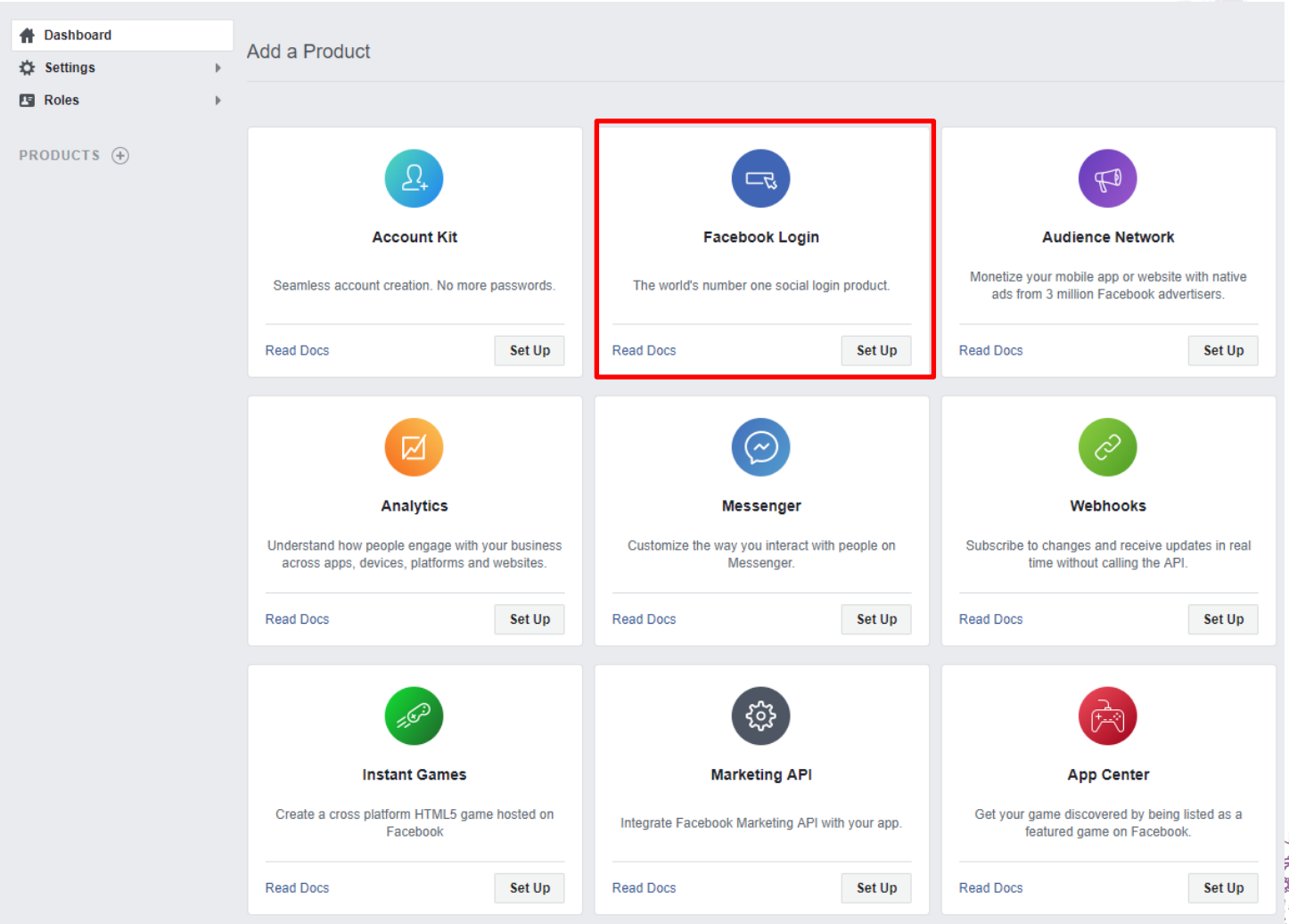
Facebook Sign-In

- Add OAuth redirect URL (find in firebase console) in **Product Settings > Facebook Login** on the [Facebook for Developers]



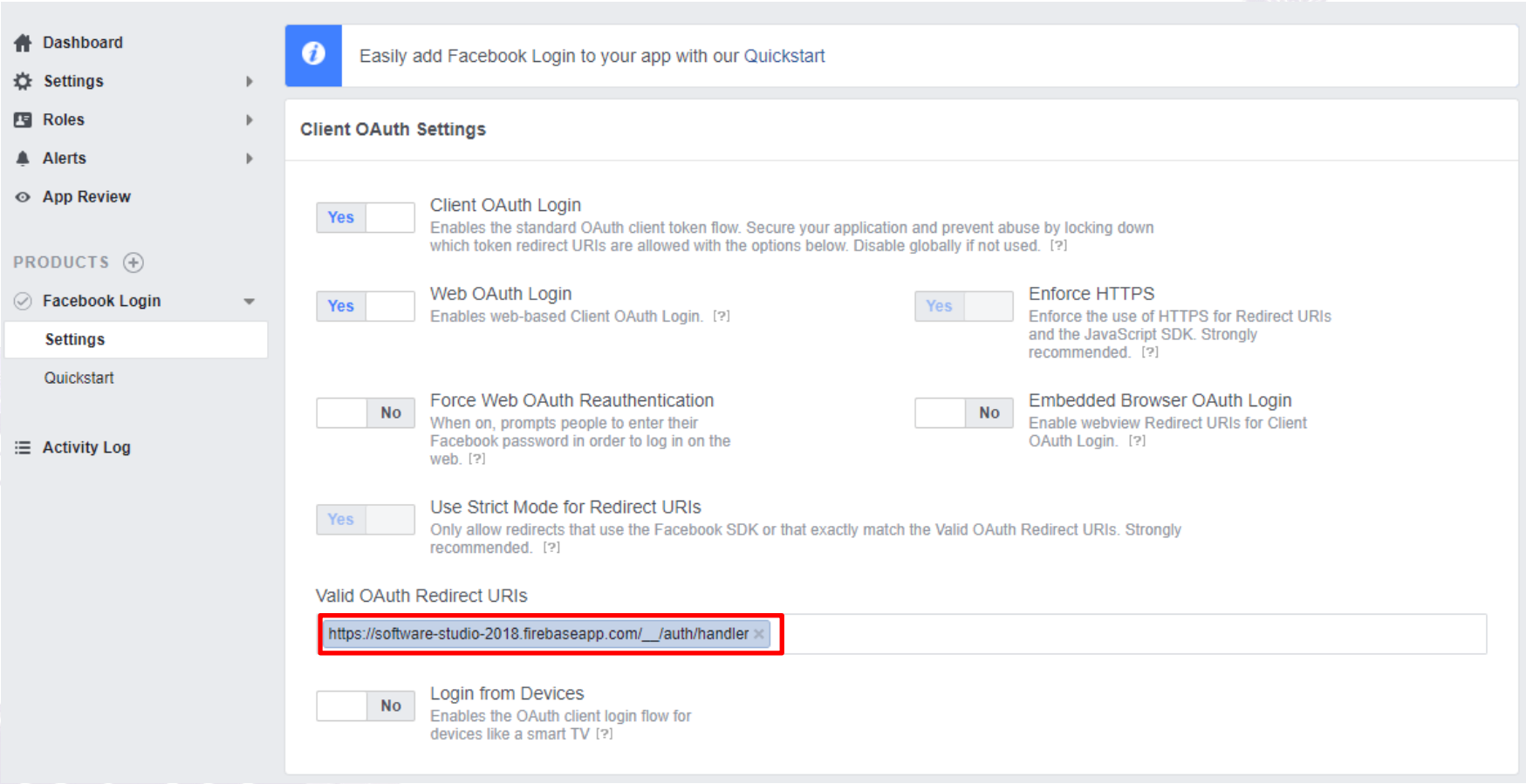
(in firebase page)

Facebook Sign-In



(in Facebook for Developer)

Facebook Sign-In



(in Facebook for Developer)

Facebook Sign-In

- Create an instance of the Facebook provider object

```
var provider = new firebase.auth.FacebookAuthProvider();
```

Facebook Sign-In

- Sign in with popup

```
firebase.auth().signInWithPopup(provider).then(function(result) {
    // This gives you a Facebook Access Token. You can use it to access the Facebook API.
    var token = result.credential.accessToken;
    // The signed-in user info.
    var user = result.user;
}).catch(function(error) {
    // Handle Errors here.
    var errorCode = error.code;
    var errorMessage = error.message;
    // The email of the user's account used.
    var email = error.email;
    // The firebase.auth.AuthCredential type that was used.
    var credential = error.credential;
});
```

Facebook Sign-In

- Sign in with redirect

```
firebase.auth().signInWithRedirect(provider);
```

- The **getRedirectResult()** function is shared with google

Manage Users

- You can get the current user is by setting an observer on the Auth object

```
firebase.auth().onAuthStateChanged(function(user) {
    if (user) {
        // User is signed in.
        var uid = user.uid;
    } else {
        // No user is signed in.
    }
});
```

Manage Users

- You can also get the currently signed-in user by using the currentUser property

```
var user = firebase.auth().currentUser;

if (user) {
    // User is signed in.
} else {
    // No user is signed in.
}
```

Sign-Out

- To sign-out a user, call signOut()

```
firebase.auth().signOut().then(() => {
  // Sign-out successful.
}).catch((error) => {
  // An error happened.
});
```

Reference

- https://firebase.google.com/docs/auth/web/start
- https://firebase.google.com/docs/reference/js/v8/firebase.auth.Auth

Realtime Database

- All Firebase Realtime Database data is stored as JSON objects
- Unlike a SQL database, there are no tables or records

database

```
{
   "users": {
      "alovelace": {
         "name": "Ada Lovelace",
         "contacts": { "ghopper": true },
   },
   "ghopper": { ... },
   "eclarke": { ... }
   }
}
```

Read and Write Data

- Get a database reference

```
// Get a reference to the database service
var database = firebase.database();
```

Write Data

- There are 4 ways to write data to the database
  - **set()**: Write or overwrite specify reference data
  - **push()**: Adding list data, every call to push() will generate a unique ID
  - **update()**: Give a key, and update the data of this key will not cover the entire data
  - **transaction()**: Updating complex data while updating will cause errors

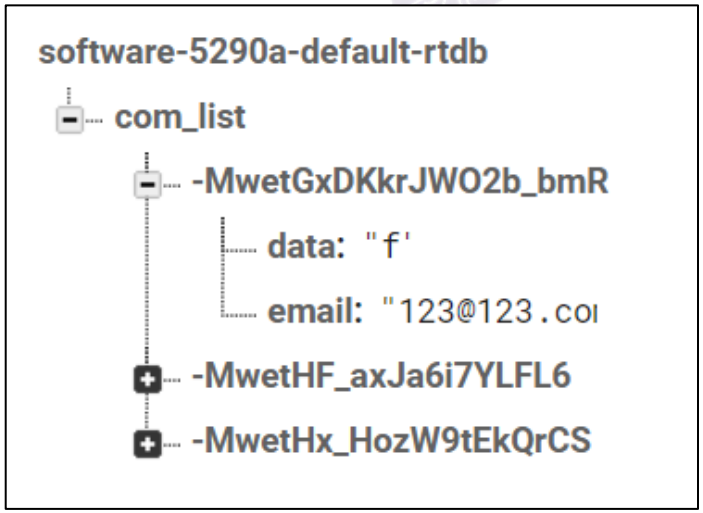
Write Data

- Use **ref()** to reference the specific path.
- Use **set()** to write data and replace any existing data at that path.

```
function writeUserData(userId, name, email, imageUrl) {
    firebase.database().ref('users/' + userId).set({
        username: name,
        email: email,
        profile_picture : imageUrl
    });
}
```

Write Data

- Use **push()** to push data.
- Every **push()** generate a unique key.
- Use **.key** to access it.




```
function writeUserData(userId, name, email, imageUrl) {
    var key = firebase.database().ref('com_list')
                    .push({data: data, email: email})
                    .key;
}
```

Read Data

- To read data at a path and listen for changes, use the **on**() or **once**() methods of reference to observe events or use query
  - On: listen for change
  - Once: for retrive one time, not listen

Read Data

- The first parameter of **on()** and **once()** is EventType. There are 5 EventTypes.
- Use '**value**' to read the value of that path.
- Use '**child_added**' to read every added child.

```
var starCountRef = firebase.database().ref('posts/' + postId + '/starCount');
starCountRef.on('value', function(snapshot) {
  updateStarCount(postElement, snapshot.val());
});
```

```
var userId = firebase.auth().currentUser.uid;
firebase.database().ref('/users/' + userId).once('value').then(function(snapshot){
    var username = (snapshot.val() && snapshot.val().username) || 'Anonymous';
});
```

Read Data

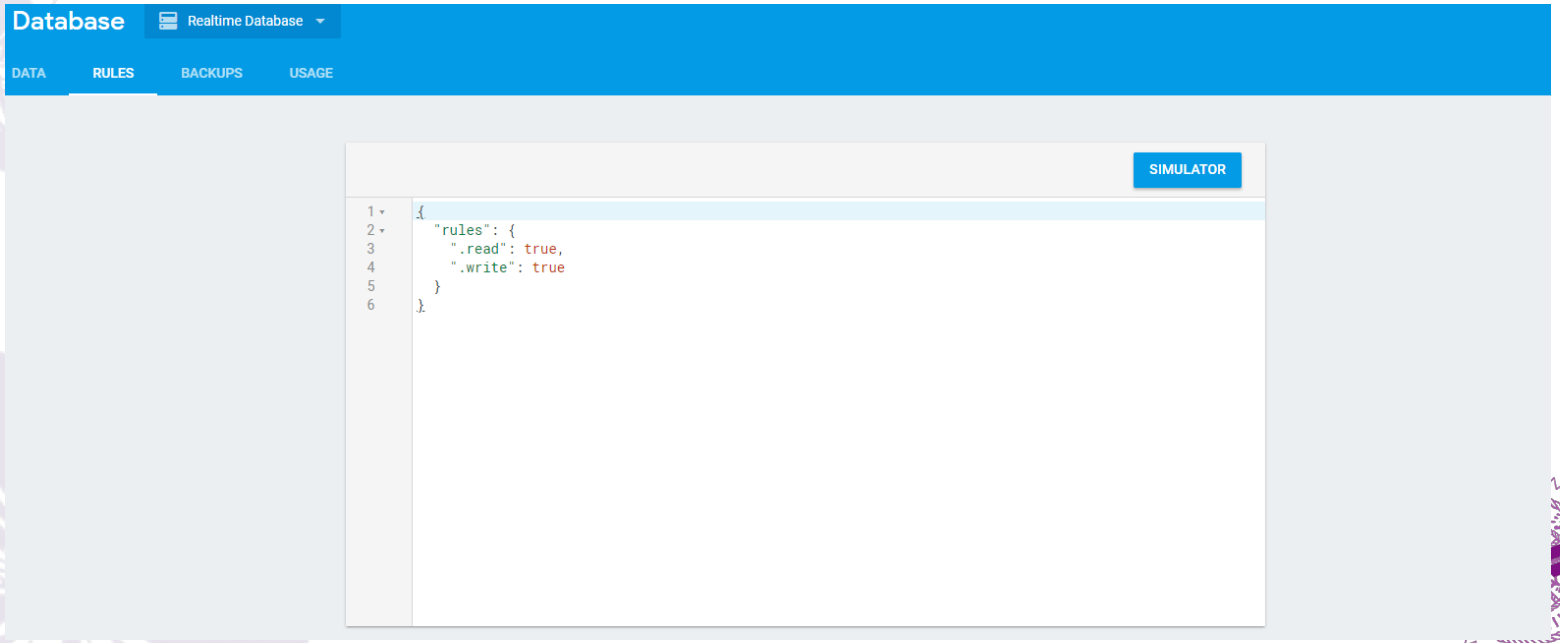
- **on()** and **once()** receive data as a DataSnapshot object.
- Use **val()** to get the content of the snapshot after you received data.

```
var starCountRef = firebase.database().ref('posts/' + postId + '/starCount');
starCountRef.on('value', function(snapshot) {
  updateStarCount(postElement, snapshot.val());
});
```

```
var userId = firebase.auth().currentUser.uid;
firebase.database().ref('/users/' + userId).once('value').then(function(snapshot){
    var username = (snapshot.val() && snapshot.val().username) || 'Anonymous';
});
```

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



Database Rules

- Rule Types
  - **.read**
  - **.write**
  - **.validate**: Defines what a correctly formatted value will look like, whether it has child attributes, and the data type
  - **.indexOn**: Specifies a child to index to support ordering and querying

Database Rules

- For example, here's a set of security rules that allows anyone to read the path /foo/, but no one to write to it

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

Database Rules

- The Firebase Database Rules include [built-in variables](#) and functions

rule

```
{
   "rules": {
      "users": {
         "$uid": {
            ".write": "$uid === auth.uid"
         }
      }
   }
}
```

Database data

```
{
  "users": {
     “James": {      // id 1
        “name”:””,
        “”
     },
    “John”:{         // id 2
        “name”:””,
        “”
     }
   }
}
```

Database Rules

- The rules language includes a **.validate** rule which allows you to apply validation logic using the same expressions

```
{
   "rules": {
      "foo": {
         ".validate": "newData.isString() && newData.val().length < 100"
      }
   }
}
```

Database Rules

- When you changed database.rules.json locally, you must use firebase deploy to update the rule. Otherwise, your changes won't apply.
- Or you can change the rules directly on firebase console.

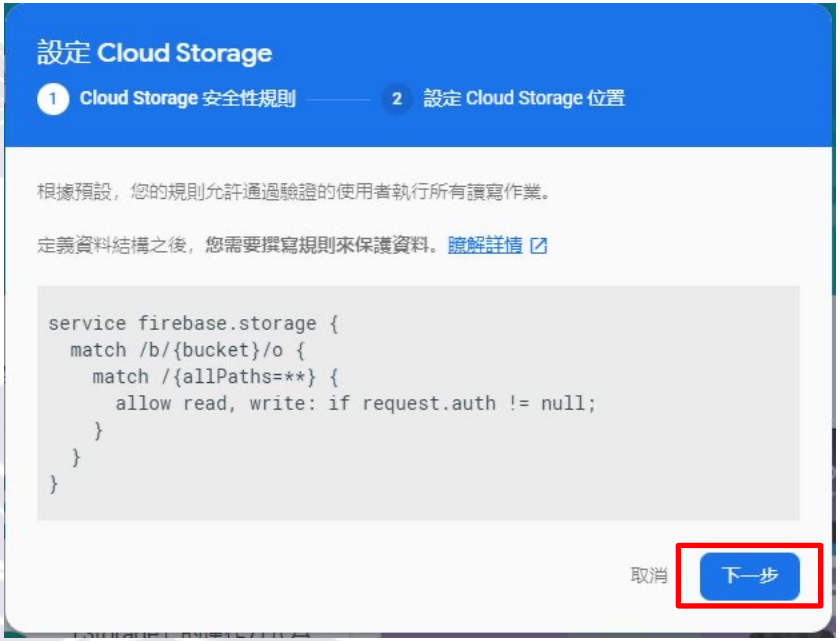
Database Rules

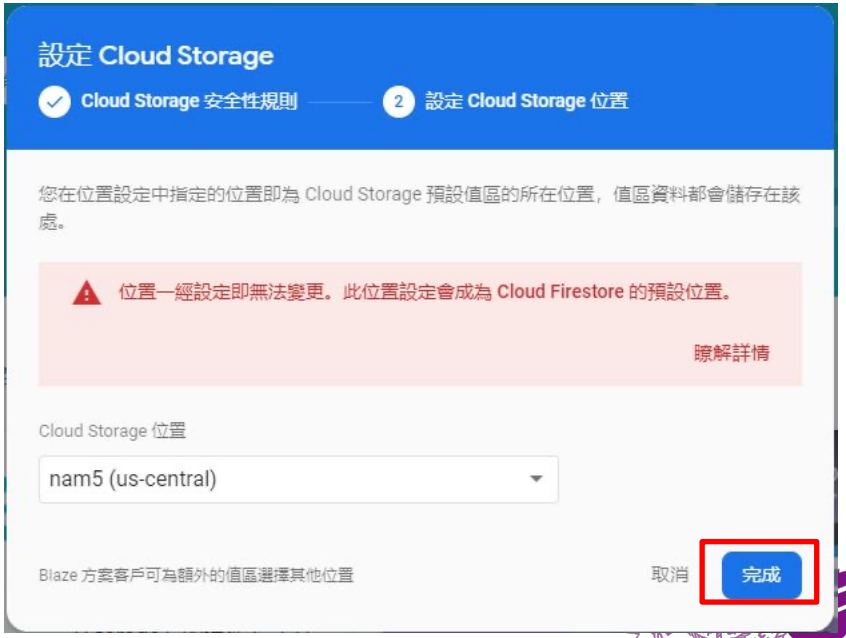
- More information and usage about database:
- https://firebase.google.com/docs/reference/security/database
- https://firebase.google.com/docs/database/web/start#web-version-8
- https://firebase.google.com/docs/database/web/read-and-write

Cloud Storage

- Cloud Storage for Firebase lets you upload and share user generated content, such as images and video, which allows you to build rich media content into your apps

Initialization





Storage Reference

- Similar to Database, Cloud Storage need a reference to upload/download files

```
// Get a reference to the storage service, which is used to create references in your storage bucket
var storage = firebase.storage();

// Create a storage reference from our storage service
var storageRef = storage.ref();

// Create a child reference, imagesRef now points to 'images'
var imagesRef = storageRef.child('images');

// Child references can also take paths delimited by '/'
// spaceRef now points to "images/space.jpg"
var spaceRef = storageRef.child('images/space.jpg');
```

Upload File

- Once you've created an appropriate reference, you then up call the put() method to up;

```
// Create a root reference
var storageRef = firebase.storage().ref();

// Create a reference to 'mountains.jpg'
var mountainsRef = storageRef.child('mountains.jpg');

var file = ... // use the Blob or File API to get the file
mountainsRef.put(file).then(function(snapshot) {
    console.log('Uploaded a blob or file!');
});
//now the file's reference in database is '/mountain.jpg'
```

Upload File (Cont'd)

```
// Create a root reference
var storageRef = firebase.storage().ref();

// Create a reference to 'images/mountains.jpg'
var mountainImagesRef = storageRef.child('images/mountains.jpg');

var file = ... // use the Blob or File API to get the file
mountainImagesRef.put(file).then(function(snapshot) {
    console.log('Uploaded a blob or file!');
});
//now the file's reference is '/images/mountains.jpg'
```

Download File

- Use **getDownloadURL()** to get file URL, your can download file directly by this URL or inserted into a HTML element

Download File

- Directly download the file

```
storageRef.child('images/stars.jpg').getDownloadURL().then(function(url) {
    // `url` is the download URL for 'images/stars.jpg'

    // Download the file directly
    var xhr = new XMLHttpRequest();
    xhr.responseType = 'blob';
    xhr.onload = function(event) {
        var blob = xhr.response;
    };
    xhr.open('GET', url);
    xhr.send();

}).catch(function(error) {
    // Handle any errors
});
```

Download File (Cont'd)

- Insert file into html element

```
storageRef.child('images/stars.jpg').getDownloadURL().then(function(url) {
    // `url` is the download URL for 'images/stars.jpg'

    // inserted into an <img> element:
    var img = document.getElementById('myimg');
    img.src = url;
}).catch(function(error) {
    // Handle any errors
});
```

Cloud Storage Rules

- Storage Security Rules manage the complexity for you by allowing you to specify path-based permissions

Storage Rules

```
// Rules can optionally specify a condition
allow write: if <condition>;
```

Storage Rules

```
service firebase.storage {
  // The {bucket} wildcard indicates we match files in all Cloud Storage buckets
  match /b/{bucket}/o {
    // Match filename
    match /filename {
      allow read: if <condition>;
      allow write: if <condition>;
    }
  }
}
```

Cloud Storage Rules

- You can also specify the file size

Storage Rules

```
// Rules can specify conditions that consider the request
service firebase.storage {
  match /b/{bucket}/o {
    match /images/{imageId} {
      // Only allow uploads of any image file that's less than 5MB
      allow write: if request.resource.size < 5 * 1024 * 1024
                   && request.resource.contentType.matches('image/.*');
    }
  }
}
```

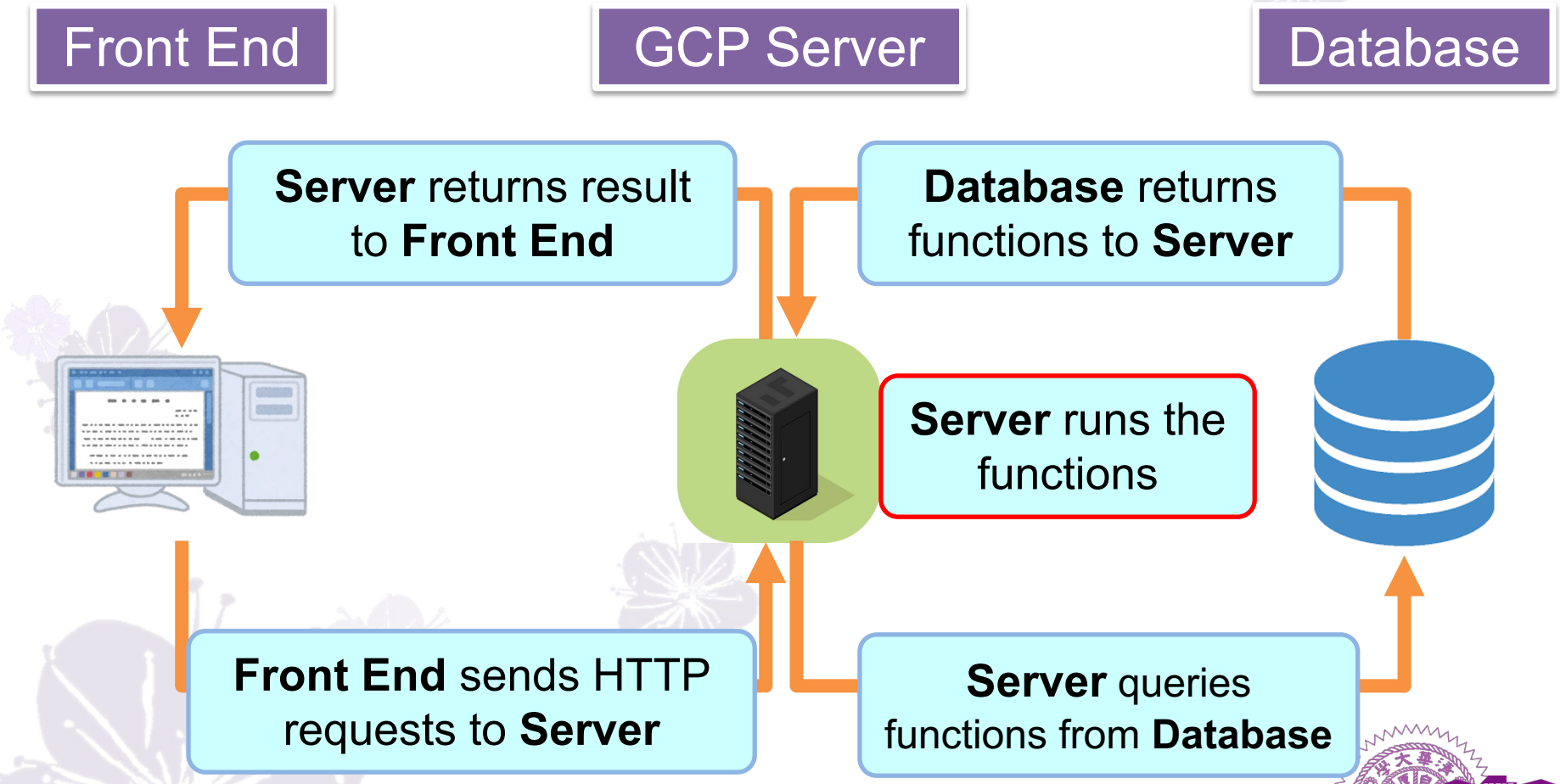
Cloud Storage Rules

- More information and usage about cloud storage rules:
- https://firebase.google.com/docs/storage/security

Cloud Function

- Cloud Functions let you automatically run backend code in response to events triggered by Firebase features and HTTPS requests.
- Your code is stored in Google's cloud and runs in a managed environment. There's no need to manage and scale your own servers.

Cloud Function



Cloud Function

- First, enable cloud functions to your project.

  Command : firebase init functions

```
? Are you ready to proceed? Yes

=== Project Setup

First, let's associate this project directory with a Firebase project.
You can create multiple project aliases by running firebase use --add,
but for now we'll just set up a default project.

? Please select an option: Use an existing project
? Select a default Firebase project for this directory: software-studio-6698b (Software Studio)
i  Using project software-studio-6698b (Software Studio)

=== Functions Setup

A functions directory will be created in your project with a Node.js
package pre-configured. Functions can be deployed with firebase deploy.

? What language would you like to use to write Cloud Functions? JavaScript
? Do you want to use ESLint to catch probable bugs and enforce style? No
+  Wrote functions/package.json
+  Wrote functions/index.js
+  Wrote functions/.gitignore
? Do you want to install dependencies with npm now? Yes

> protobufjs@6.8.9 postinstall D:\大三嗨起來\助教人生\軟體實驗\Lecture Program\Firebase\answer\functions\node_modules\protobufjs
> node scripts/postinstall

npm notice created a lockfile as package-lock.json. You should commit this file.
added 258 packages from 201 contributors and audited 930 packages in 73.817s

30 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities


i  Writing configuration info to firebase.json...
i  Writing project information to .firebaserc...
i  Writing gitignore file to .gitignore...

+  Firebase initialization complete!
```

Cloud Function

- Then you will get the following folder in your project.
- Now, we are going to edit **index.js** to create our cloud functions.

Get Firebase admin

- Just like what we did **firebase.initializeApp(config)** in our .js files. We initialize firebase again in index.js.
- After that we can use the firebase functions, such as database or storage.

```
const admin = require('firebase-admin');
admin.initializeApp();
```

index.js

Create a New Function

- This function is an HTTP endpoint. Any request to the endpoint results in ExpressJS-style Request and Response objects passed to the onRequest() callback. https://expressjs.com/zh-tw/

Serve code

```
exports.addMessage = functions.https.onRequest(async(req, res) => {
  // Grab the text parameter.
  const original = req.query.text;
  // Push the new message into the Realtime Database using the Firebase Admin SDK.
  const snapshot = await admin.database().ref('/messages').push({ original: original });
  // Redirect with 303 SEE OTHER to the URL of the pushed object in the Firebase console.
  res.redirect(303, snapshot.ref.toString());
});
```

Client code

```
// This client-end function is equivalent to the previous cloud function
function addMessageLocal(text){
   const snapchat = await admin.database().ref('/messages').push(text);
}
```

Setup Local Testing Server

- Once you finish editing your cloud functions, you can start up a local server to test the correctness of your functions.

  Command : firebase emulators:start --only functions

```
$ firebase emulators:start --only functions
i  emulators: Starting emulators: functions
!  Your requested "node" version "8" doesn't match your global version "12"
+  functions: Emulator started at http://localhost:5001
i  functions: Watching "C:\Users\penguin\Desktop\firebase\functions" for Cloud Functions...
+  functions[addMessage]: http function initialized (http://localhost:5001/software-studio-6698b/us-central1/addMessage).
+  All emulators started, it is now safe to connect.
```

Use Cloud Functions (local)

- After initialize firebase in your js code, you can get the functions you created.
- Remember that if you are using local server to call cloud functions, you have to use **firebase.functions().useFunctionsEmulator()** to redirect to localhost server.

Client code

```
// connect to localhost server
firebase.functions().useFunctionsEmulator('http://localhost:5001');
//comment this line if you use formal cloud functions

// call function : addMessage
var addMessage = firebase.functions().httpsCallable('addMessage');
addMessage({text:uppercasemetoo}).then(()=>{});
```

Deploy Cloud Functions

- Deploy cloud function to server require upgrading firebase service to **Blaze**

8 將功能部署到生產環境

一旦您的功能在仿真器中可以正常工作，您就可以繼續在生產環境中部署，測試和運行它們。請記住，要部署到建議的Node.js 12運行時環境，您的項目必須處於Blaze即用即付計費計劃中。請參閱"雲功能定價"。

Deploy Cloud Functions

- Deploy function to server make sure we don't need to run server by ourselves

Command : firebase deploy --only functions

```
$ firebase deploy --only functions

=== Deploying to 'software-studio-6698b'...

i  deploying functions
i  functions: ensuring necessary APIs are enabled...
+  functions: all necessary APIs are enabled
i  functions: preparing functions directory for uploading...
i  functions: packaged functions (26.79 KB) for uploading
+  functions: functions folder uploaded successfully
i  functions: creating Node.js 8 function addMessage(us-central1)...
+  functions[addMessage(us-central1)]: Successful create operation.
Function URL (addMessage): https://us-central1-software-studio-6698b.cloudfunctions.net/addMessage

+  Deploy complete!

Project Console: https://console.firebase.google.com/project/software-studio-6698b/overview
```

Use Cloud Functions

- After deploy firebase cloud functions, you can use the functions you created.

Client code

```
// call function : addMessage
var addMessage = firebase.functions().httpsCallable('addMessage');
addMessage({text:uppercasemetoo}).then(()=>{});
```

Demo

- Enter the following url in your browser.
- *https://us-central1-[MY_PROJECT].cloudfunctions.net/addMessage?text=uppercasemetoo*
- Then go to firebase console and check your real-time database. If cloud functions setup successfully, you will have following structure.

Reference

- Firebase Documentation
- Firebase Get Started – Web
- Firebase on the Web - Tutorials


thank you!
Question?