Firebase: REST and Web API

INF 551

Wensheng Wu

Firebase

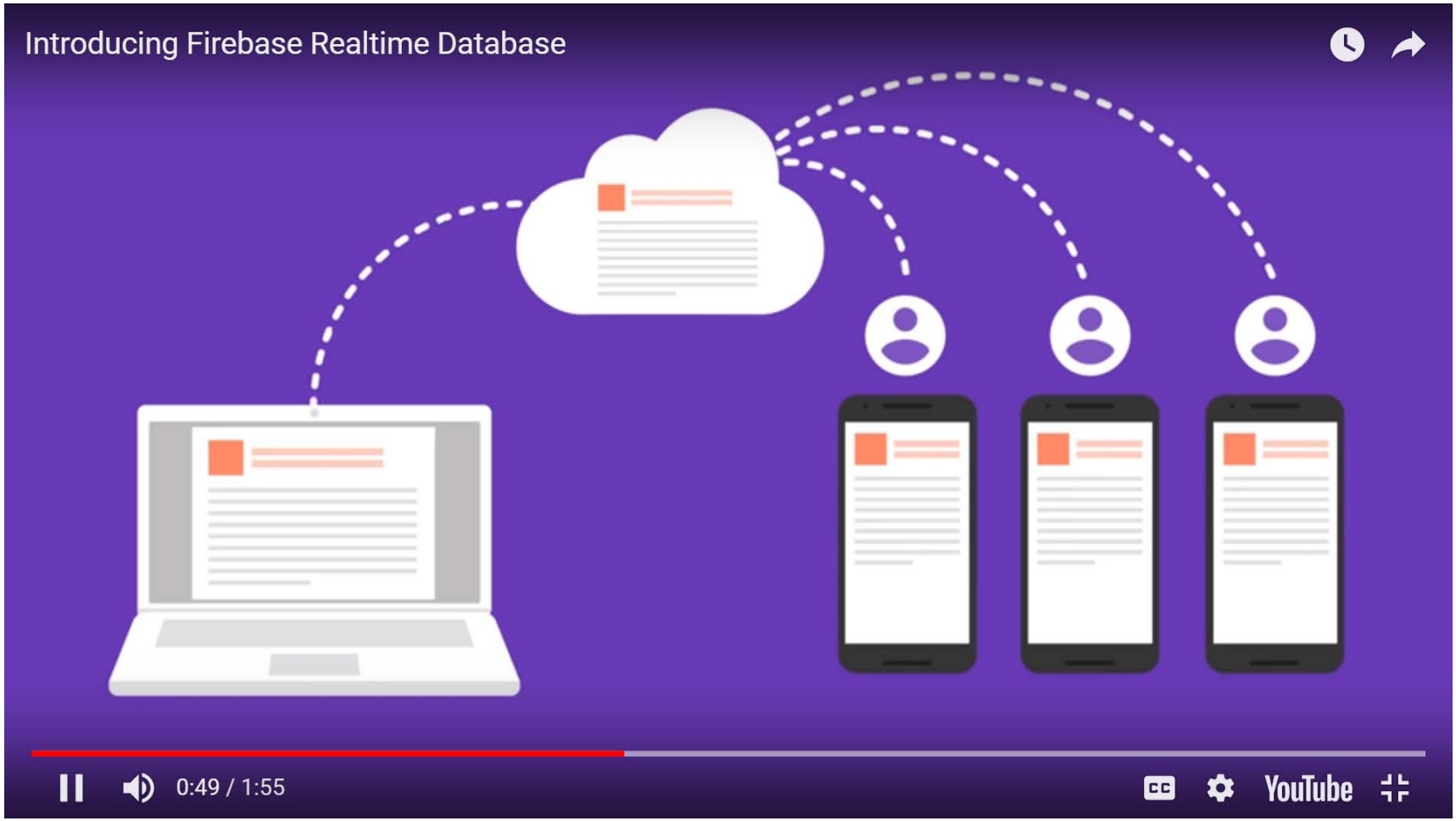
- A cloud-based platform to support web and mobile app development
- Used to be Envolve, a startup founded in 2011
  - For adding online chat functions into websites
- Later expanded into Firebase which was then acquired by Google in 2015

Products

Also has Cloud Firestore in Beta

- Firebase (realtime) database
  - Manage JSON documents
  - Real-time syncing data between users and devices
- Firebase (cloud) storage
  - Store images, photos, videos
- Firebase (user) authentication
  - Support signin using Google, Facebook

Firebase realtime database



Firebase

INF55x

Project Overview

Develop

Authentication

Database

Storage

Hosting

Functions

ML Kit

Storage

Files Rules Usage

gs://inf55x.appspot.com

Name

tropical-coast-10132.j

tropical-coast-1013...

Name
tropical-coast-10132.jpg

Size
554,404 bytes

Type
image/jpeg

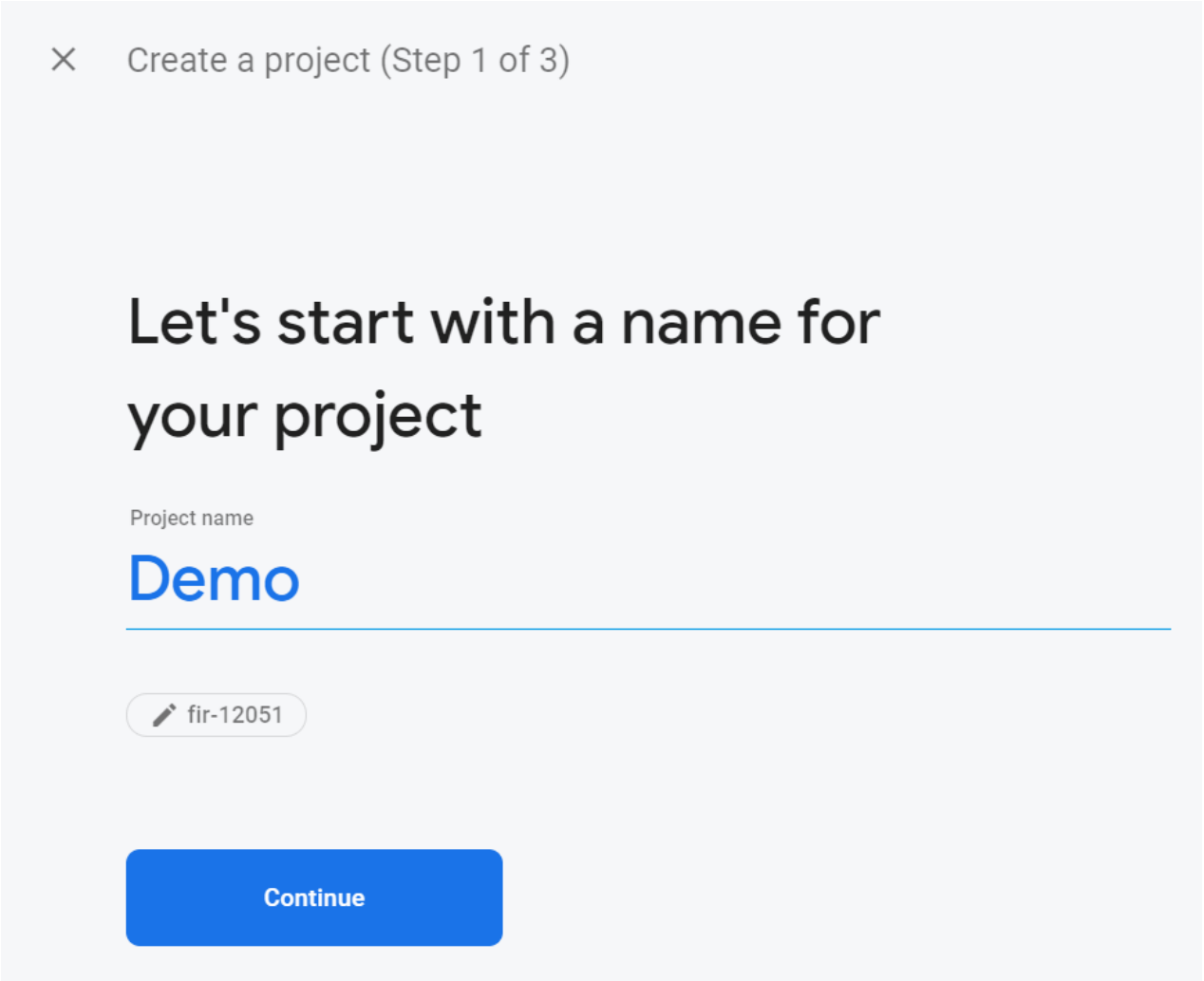
Create a Firebase account

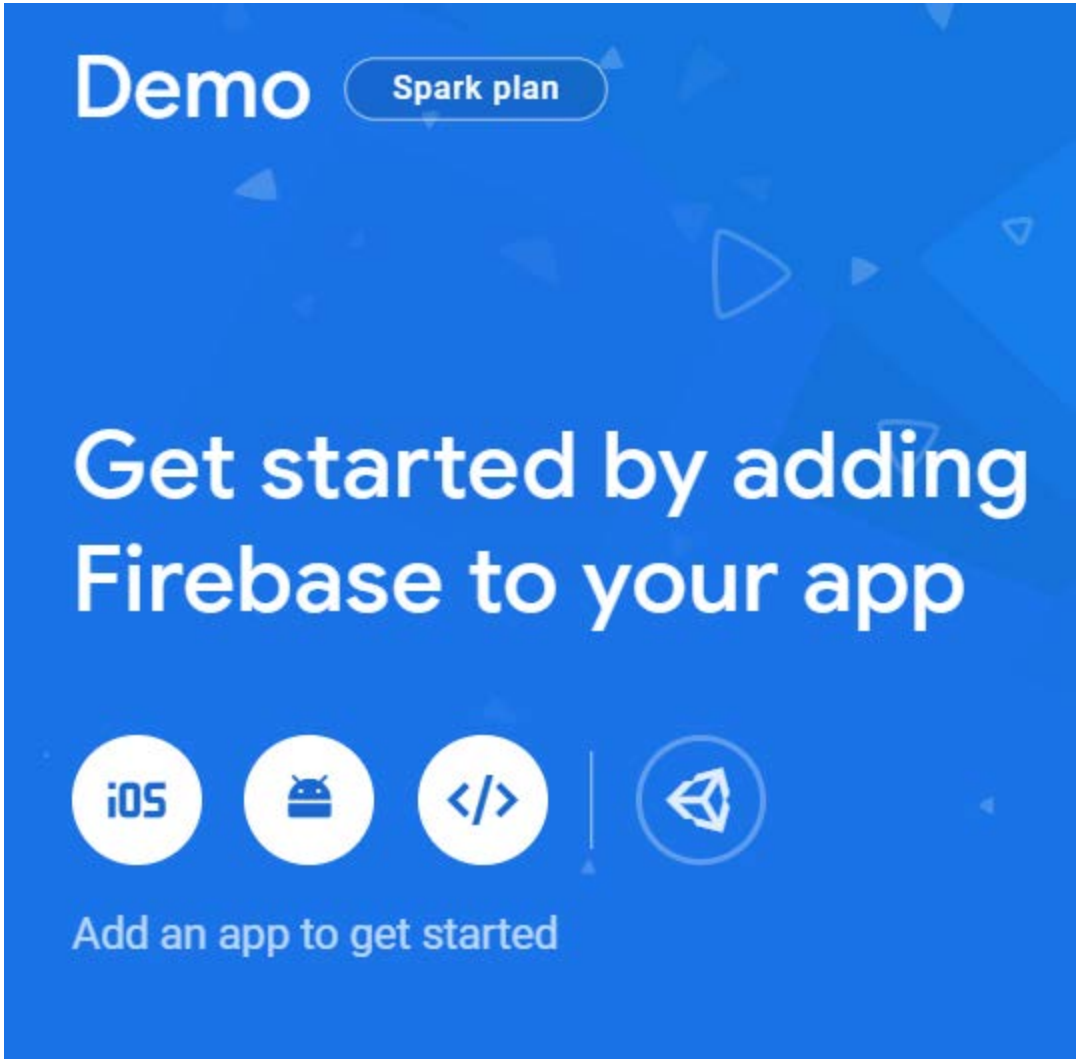
- You may use your Google account

- Go to Firebase console:
  - https://console.firebase.google.com/

Click on "Add project"

Create a Firebase project




Demo
Spark plan
Get started by adding Firebase to your app
iOS
Add an app to get started

Add Firebase to your web app

<script src="https://www.gstatic.com/firebasejs/7.7.0/firebase-database.js"></script>

Register app

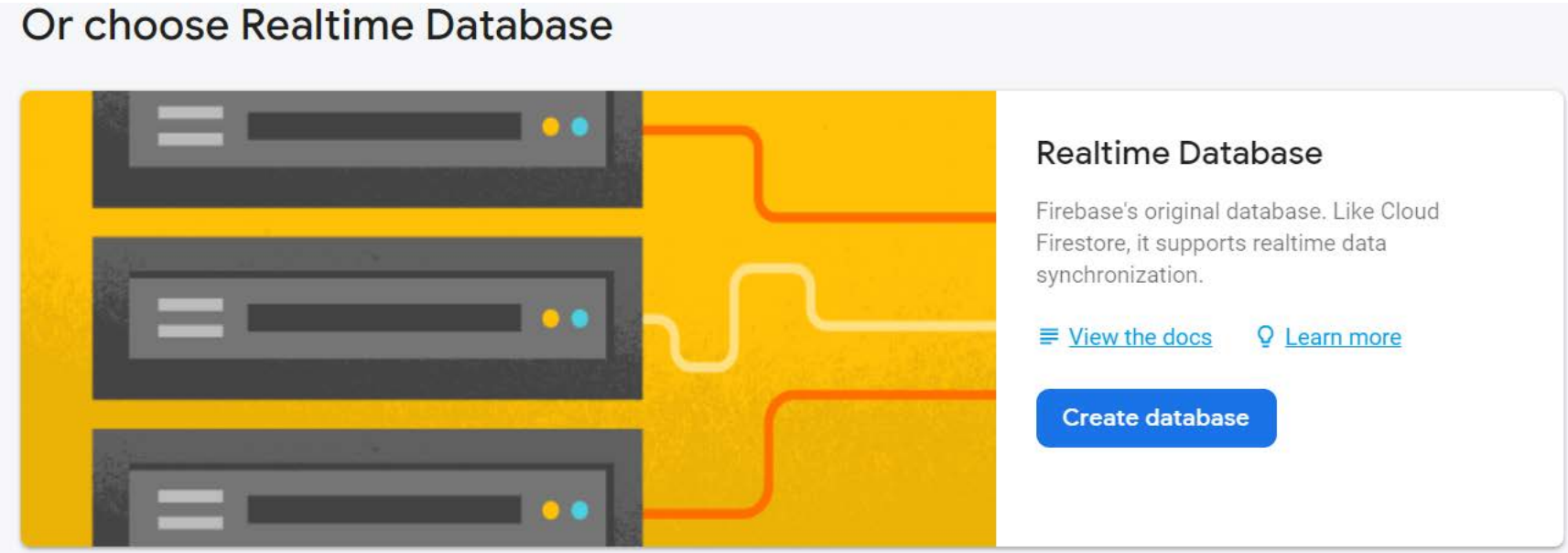
2 Add Firebase SDK

Add this

Copy and paste these scripts into the bottom of your <body> tag, but before you use any Firebase services:

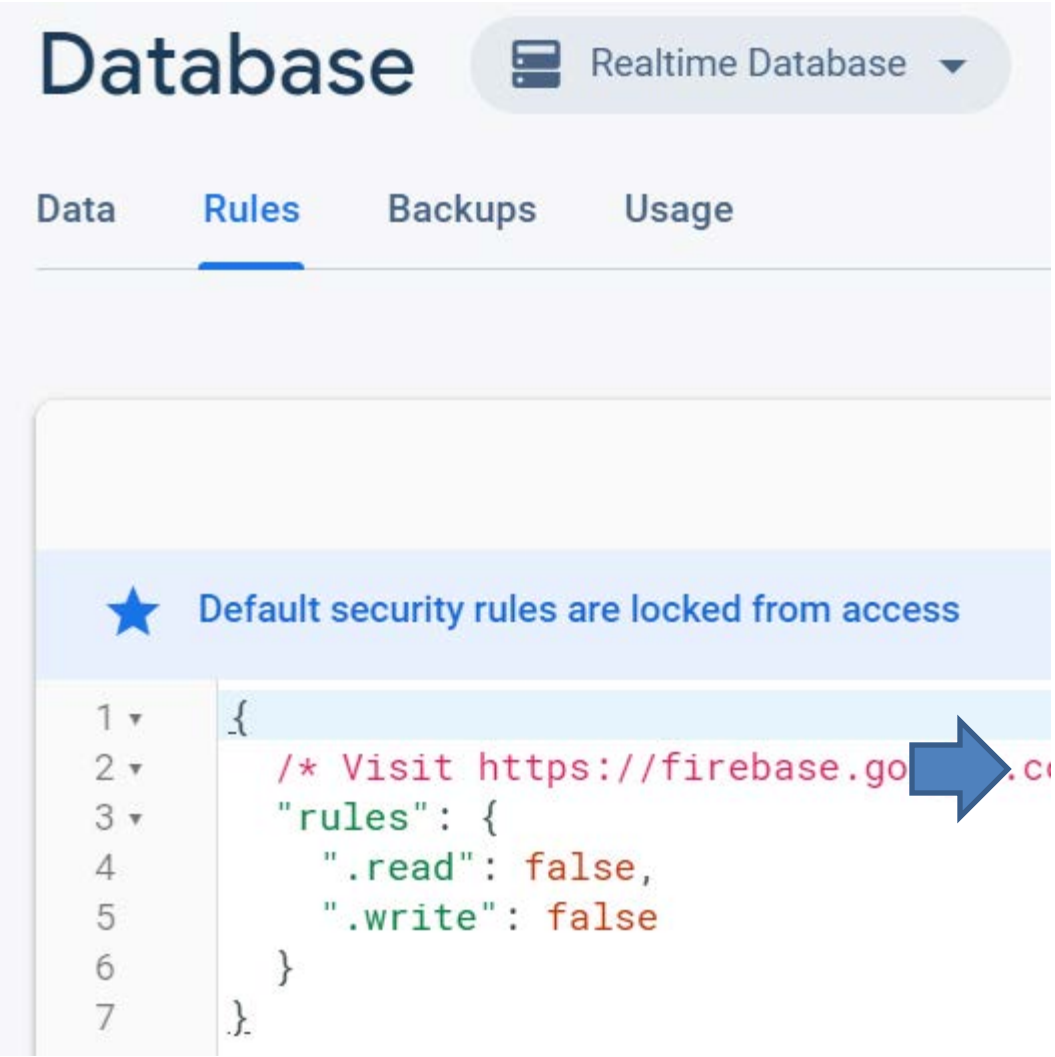
```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/7.7.0/firebase-app.js"></script

<!-- TODO: Add SDKs for Firebase products that you want to use
     https://firebase.google.com/docs/web/setup#available-libraries -->

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: "AIzaSyC5_JqxzEePNfplBB4LrzOAQu2GaVF3Mas",
    authDomain: "fir-12051.firebaseapp.com",
    databaseURL: "https://fir-12051.firebaseio.com",
    projectId: "fir-12051",
    storageBucket: "fir-12051.appspot.com",
    messagingSenderId: "692453711883",
    appId: "1:692453711883:web:0a96ec021817c21ed8e70e"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
</script>
```

Realtime database



Realtime database

Opening up the access



Firebase pricing plans

Spark	Blaze
Free $0/month	Pay as you go
Usage quotas for Database, Firestore, Storage, Functions, Phone Auth, Hosting, and Test Lab	Includes free usage, calculated daily. After, pay only for what your project uses.
Ability to extend your project with Google Cloud Platform	Ability to extend your project with Google Cloud Platform
Included in all plans Analytics, Notifications, Crash Reporting, support, and more	**Included in all plans** Analytics, Notifications, Crash Reporting, support, and more
See full plan details	See full plan details
Current Plan	Select plan

JSON (Javascript Object Notation)

- Light-weight data exchange format
  - Much simpler than XML
  - Language-independent
  - Inspired by syntax of JavaScript object literals

- Some differences from JavaScript objects, e.g.,
  - String in JSON must be double-quoted
  - Ok to single-quote in JavaScript (& Python)

Syntax of JSON

These are actual values

- value = string|number|object|array|**true**|**false**|**null**

- object = {} | { members }
  - members = pair | pair, members
  - pair = string : value

- array = [] | [elements]
  - elements = value | value, elements

Valid JSON or not?

- []
- {}
- {[]}
- [{}]
- {"name": john}
- {name: "john"}
- {"name": 25}
- "name"
- 25
- {25}
- [25]

JSON is case-sensitive

- Valid or not?
  - True
  - true
  - TRUE
  - Null
  - false

Example JSON

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```

Value is an object

Value is an array

Stored in Firebase

```
inf551-1b578
  examples
    address
      city: "New York
      postalCode: "10021-3100
      state: "NY"
      streetAddress: "21 2nd Street
    age: 25
    firstName: "John'
    height_cm: 167.6
    isAlive: true
    lastName: "Smith'
    phoneNumbers
      0
        number: "212 555-1234
        type: "home'
      1
```

Note: array stored as an object
Key = index of element in array

Check syntax of JSON

- JSON validator
  - http://jsonlint.com/

Roadmap

- Firebase REST API

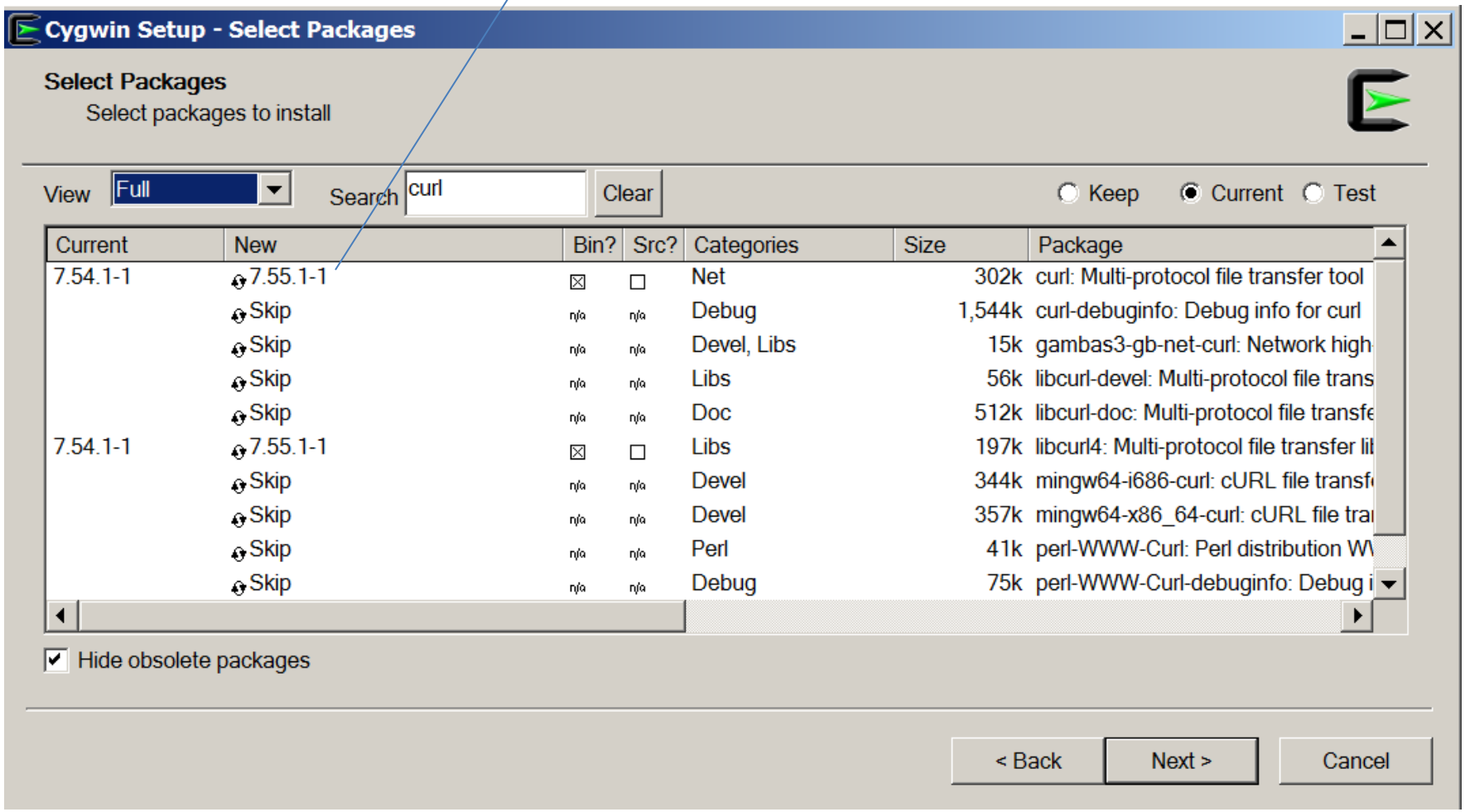
- Firebase Javascript API
  - Useful for your project

curl

- Command line tool for data transfer
- Download from here (has Windows & Mac OS versions):
  - https://curl.haxx.se/download.html
- You may easily grab a copy of curl in Cygwin (see next slide)

Install curl in Cygwin

Select to install this one



Firebase REST API

- PUT & POST (C in CRUD)
- GET (R)
- PATCH (U)
- DELETE (D)

All request commands
are case sensitive (all uppercases)

GET

- curl 'https://inf551-1b578.firebaseio.com/weather.json'

- Or
  - curl -X GET 'https://inf551-1b578.firebaseio.com/weather.json'

Another example

- curl -X GET 'https://inf551-1b578.firebaseio.com/examples/phoneNumbers/0.json'
  - {"number":"212 555-1234","type":"home"}

Note: refer to array element by index

PUT

- curl -X PUT 'https://inf551-1b578.firebaseio.com/weather.json' -d '"hot"'
  - "hot"

- PUT: write a given value (e.g., "hot") to the specify node (e.g., "weather")
  - Overwrite if node already has value

PUT

- curl -X PUT 'https://inf551-1b578.firebaseio.com/users/100.json' -d '{"name": "john"}'

- This will add a new node "users" (assuming it does not exist yet) and a child of this node with key "100" and content: {"name": "john"}

Example

- Is the previous command the same as this?
  - curl -X PUT -d '{"100": {"name": "John"}}' 'https://inf551-1b578.firebaseio.com/users.json'

Note we now write to the "users" node

- Can you think of a situation where two commands give different results?

POST

- curl -X POST -d '{"name": "John"}' 'https://inf551-1b578.firebaseio.com/users.json'

- Note post automatically generates a new key & then stores the value for the new key
  - In contrast, PUT will simply overwrite the value

PATCH

- curl -X PATCH -d '{"name": "John Smith", "age": 25}' 'https://inf551-1b578.firebaseio.com/users/100.json'

- PATCH performs the update if value already exists (e.g., name) ; otherwise, it inserts the new value (e.g., age)
  - So... an upsert

DELETE

- curl -X DELETE 'https://inf551-1b578.firebaseio.com/users/100.json'

- What does this do?
  - curl -X DELETE 'https://inf551-1b578.firebaseio.com/users.json'

Query: filtering by key

- curl 'https://inf551-1b578.firebaseio.com/users.json?orderBy="$key"&equalTo="200"'

Must be a string. Why?

- This returns:
  - {"200":{"age":25,"name":"David"}}

Another example

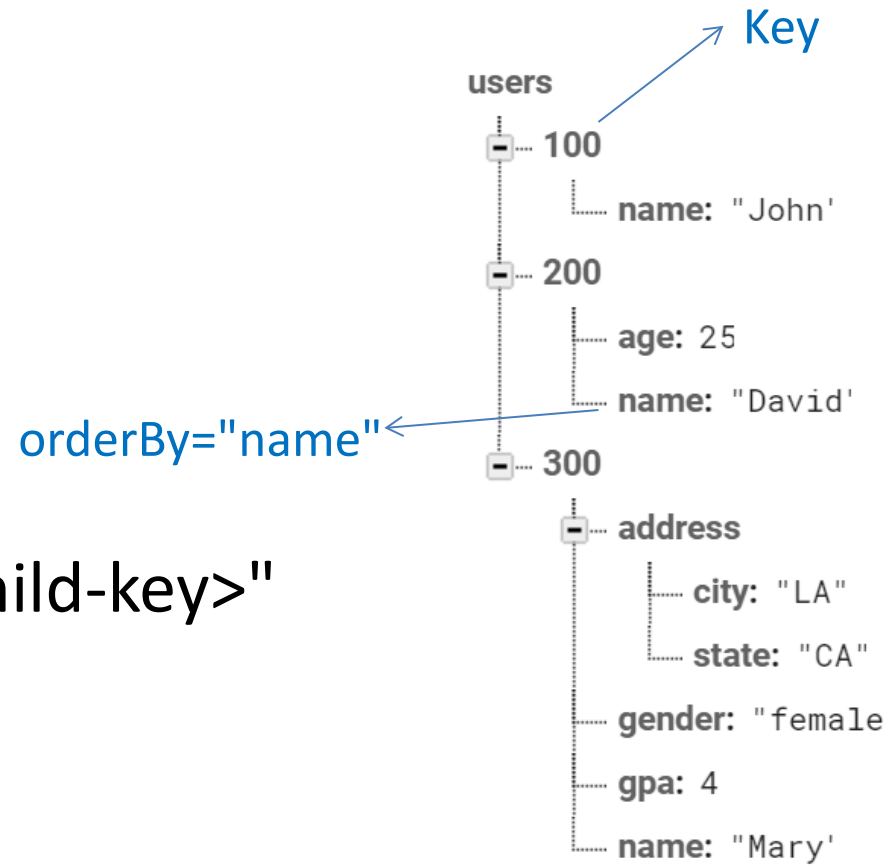
- curl 'https://inf551-1b578.firebaseio.com/users.json?orderBy="$key"&startAt="200"'

Users with keys >= "200"

- This returns:
  - {"200":{"age":25,"name":"David"},"300":{"gender":"female","gpa":4.0,"name":"Mary"}}

Ways of filtering data

- By key:
  - orderBy="$key"
- By child key:
  - orderBy="<path-to-child-key>"
- By value:
  - orderBy="$value"



Parameters

- startAt
- endAt
- equalTo
- limitToFirst
- limitToLast

Example: filtering by child key

- curl 'https://inf551-1b578.firebaseio.com/users.json?orderBy="name"&limitToFirst=1&print=pretty'

- What will this return?

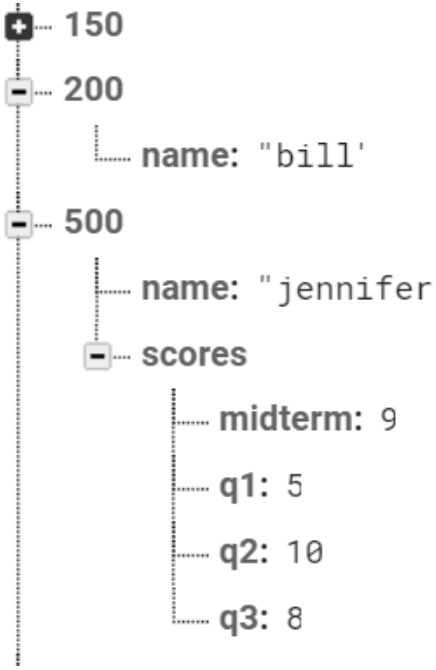
Example for orderBy="$value"

- curl -X PUT 'https://inf551-1b578.firebaseio.com/users/500.json' -d '{"name": "jennifer", "scores": {"q1": 5, "q2": 10, "q3": 8, "midterm": 9}}'

Example: filtering by value

- curl 'https://inf551-1b578.firebaseio.com/users/500/scores.json?orderBy="$value"&limitToFirst=1&print=pretty'

- What will this return?



Creating index for value/child key

- Specified in database rules:
  - https://firebase.google.com/docs/database/security/indexing-data

- Only required

  for REST API

```
{
  "rules": {
    ".read": true,
    ".write": true,
    "users": {
      ".indexOn": ["name", "age"],
      "500": {
        "scores": {".indexOn": ".value"}}
    }
  }
}
```

Ordering

orderBy

When using `orderBy` with the name of a child key, data that contains the specified child key will be ordered as follows:

1. Children with a `null` value for the specified child key come first.
2. Children with a value of `false` for the specified child key come next. If multiple children have a value of `false`, they are sorted lexicographically by key.
3. Children with a value of `true` for the specified child key come next. If multiple children have a value of `true`, they are sorted lexicographically by key.
4. Children with a numeric value come next, sorted in ascending order. If multiple children have the same numerical value for the specified child node, they are sorted by key.
5. Strings come after numbers, and are sorted lexicographically in ascending order. If multiple children have the same value for the specified child node, they are ordered lexicographically by key.
6. Objects come last, and sorted lexicographically by key in ascending order.

The filtered results are returned unordered. If the order of your data is important you should sort the results in your application after they are returned from Firebase.

Watch out…

- https://firebase.google.com/docs/database/rest/retrieve-data

Filtered data is returned unordered: When using the REST API, the filtered results are returned in an undefined order since JSON interpreters don't enforce any ordering. If the order of your data is important you should sort the results in your application after they are returned from Firebase.

Using REST in Python

- import requests
  - May need to "pip install requests" first

- url = 'https://inf551-1b578.firebaseio.com/users.json'
- response = requests.get(url)
- response.json()
  - {u'200': {u'age': 25, u'name': u'David'},…

Writing

- url1 = 'https://inf551-1b578.firebaseio.com/users/888.json'

- data = '{"name": "jimmy", "gender": "male"}'

- response = requests.put(url1, data)

Update, delete & post

- Updating
  - requests.patch(url, data)
- Deleting
  - requests.delete(url)
- Posting
  - Requests.post(url, data)

Pretty printing

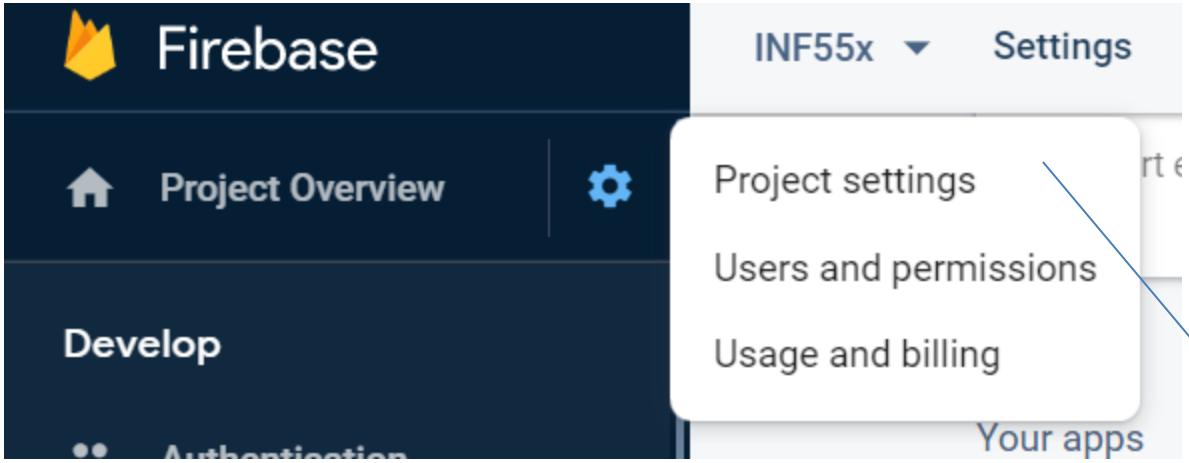
- import json

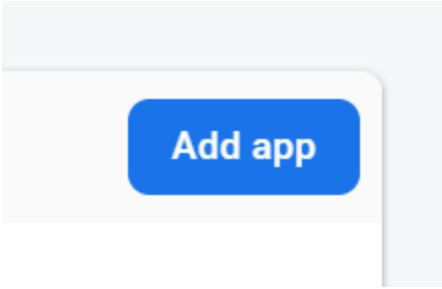
- print json.dumps(response.json(), indent=4)

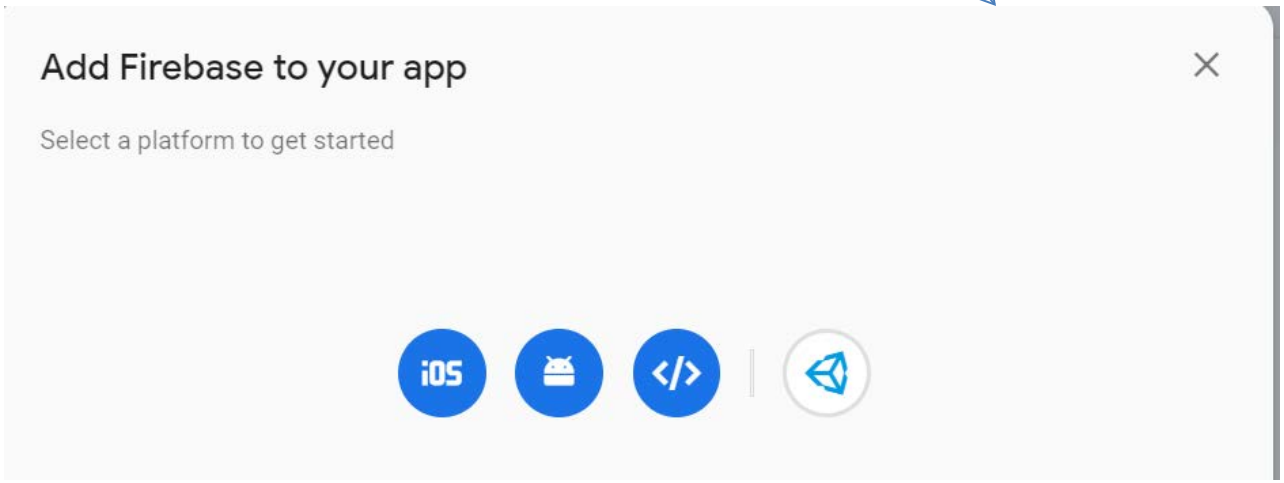
```
{
    "200": {
        "age": 25,
        "name": "David"
    },
…
```

Roadmap

- Firebase REST API
- Firebase Javascript/Web API
  - Useful for your project


Firebase
INF55x
Settings
Project Overview
Project settings
Users and permissions
Usage and billing
Develop
Your apps


Add app


Add Firebase to your app
Select a platform to get started
iOS

Add Firebase to your web app

<script src="https://www.gstatic.com/firebasejs/7.7.0/firebase-database.js"></script>

Register app

2 Add Firebase SDK

Add this

Copy and paste these scripts into the bottom of your <body> tag, but before you use any Firebase services:

```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/7.7.0/firebase-app.js"></script

<!-- TODO: Add SDKs for Firebase products that you want to use
     https://firebase.google.com/docs/web/setup#available-libraries -->

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: "AIzaSyC5_JqxzEePNfplBB4LrzOAQu2GaVF3Mas",
    authDomain: "fir-12051.firebaseapp.com",
    databaseURL: "https://fir-12051.firebaseio.com",
    projectId: "fir-12051",
    storageBucket: "fir-12051.appspot.com",
    messagingSenderId: "692453711883",
    appId: "1:692453711883:web:0a96ec021817c21ed8e70e"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
</script>
```

Demo html page

```html
<html>
<head><title>Test Firebase</title></head>
<body>

It is <span id="value"></span> today!!

<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/7.7.0/firebase-app.js"></script>
<script src="https://www.gstatic.com/firebasejs/7.7.0/firebase-database.js"></script>

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: "AIzaSyDSyDdbCB6m9JXqgCDvMMHbNqY0L8WixiI",
    authDomain: "inf55x.firebaseapp.com",
    databaseURL: "https://inf55x.firebaseio.com",
    projectId: "inf55x",
    storageBucket: "inf55x.appspot.com",
    messagingSenderId: "163182188596",
    appId: "1:163182188596:web:ca7ccfc2221ef4f4db5261"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);

  var value = document.getElementById("value");
  var dbRef = firebase.database().ref().child("weather");

  // query example: a single value
  dbRef.on('value', function(snapshot) {
     console.log("weather value" + ": " + JSON.stringify(snapshot.val()));
     value.innerText = snapshot.val()
  });
```

val() returns a Javascript object representing content of snapshot

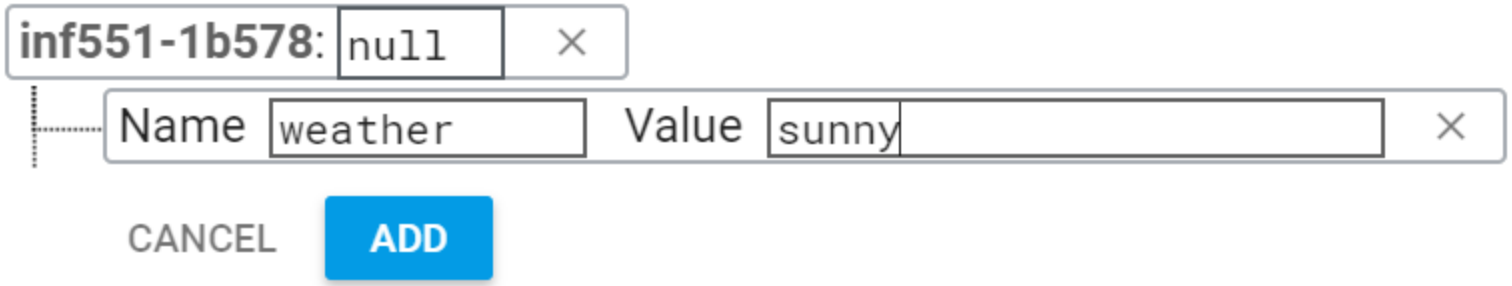
Database reference

- firebase.database() returns a reference to the firebase database as specified by "firebaseConfig"

- ref(): returns a reference to the root node of the database
- ref("weather") returns a reference to the "weather" child of the root
  - same as ref().child("weather")

Modify the data in database

- Observe the data automatically changed in the browser



Write data using set()

- function writeUserData(userId, name, email) {
  firebase.database().ref("users/" + userId).set({
    name: name,
    email: email
  });
}

Setting/overwriting the data of user 123

- writeUserData("123", "John", "john@usc.edu");

Write data using push() and set()

- firebase.database().ref("users").push().set({name: "John", email: "john@usc.edu"});

- push() will automatically generate a key
  - In this case, id for the new user

- Which REST command is this similar to?

Update data

- function updateUserData(userId, phone) {

```
  firebase.database().ref("users/"+userId).update({
    phone: phone
  });
}
```

Note this does not remove other data of user 123
What if you replace "update" with "set"?

- updateUserData("123", "(626)123-0000");

Retrieve a list of values

- userRef = firebase.database().ref("users");
- userRef.on("value", function(snapshot) {

```
  snapshot.forEach(function(child) {
   console.log(child.key + ": " + child.val());
  });
});
```

Press Ctrl+Shift+J in Chrome for console window

Listening to child events instead

- userRef.on("value", function(snapshot) {…
  - Will retrieve a list of values in the path specified by userRef
  - Not efficient, since entire list will be retrieved whenever changes occur

- userRef.on("child_added", function(…)) {…
  - Firebase will callback for every existing child and new child added to the path userRef
  - Other events: child_changed, child_removed

Filtering data

- queryRef = firebase.database().ref("users").orderByChild("name").equalTo("David");
- queryRef.on("value", function(snapshot) {

```
snapshot.forEach(function(child) {
  console.log(child.key + ": " + child.val());
 });
});
```

Filtering data

- It also supports:
  - orderByKey()
  - orderByValue()

orderByValue() example

```
queryRef = firebase.database().ref("users/500/scores")
        .orderByValue();
 queryRef.on("value", function(snapshot) {
        snapshot.forEach(function(child) {
                console.log(child.key + ": " + child.val());
  });
 });
```

```
q1: 5
q3: 8
midterm: 9
q2: 10
```

Resources

- Add Firebase to your JavaScript Project
  - https://firebase.google.com/docs/web/setup
- Getting Started with Firebase on the Web
  - https://www.youtube.com/watch?v=k1D0_wFlXgo&feature=youtu.be
- Realtime Database: Installation & Setup in JavaScript, Read & Write Data …
  - https://firebase.google.com/docs/database/web/start

Resources

- Firebase REST API
  - https://firebase.google.com/docs/reference/rest/database/

- Requests for Python
  - http://docs.python-requests.org/en/master/user/quickstart/#make-a-request