



IS216 Web Application Development II

Session 6

JavaScript – Part 3

(APIs, Axios, JSON)

K. J. Shim

Sections: G1 / G2 / G3 / G11



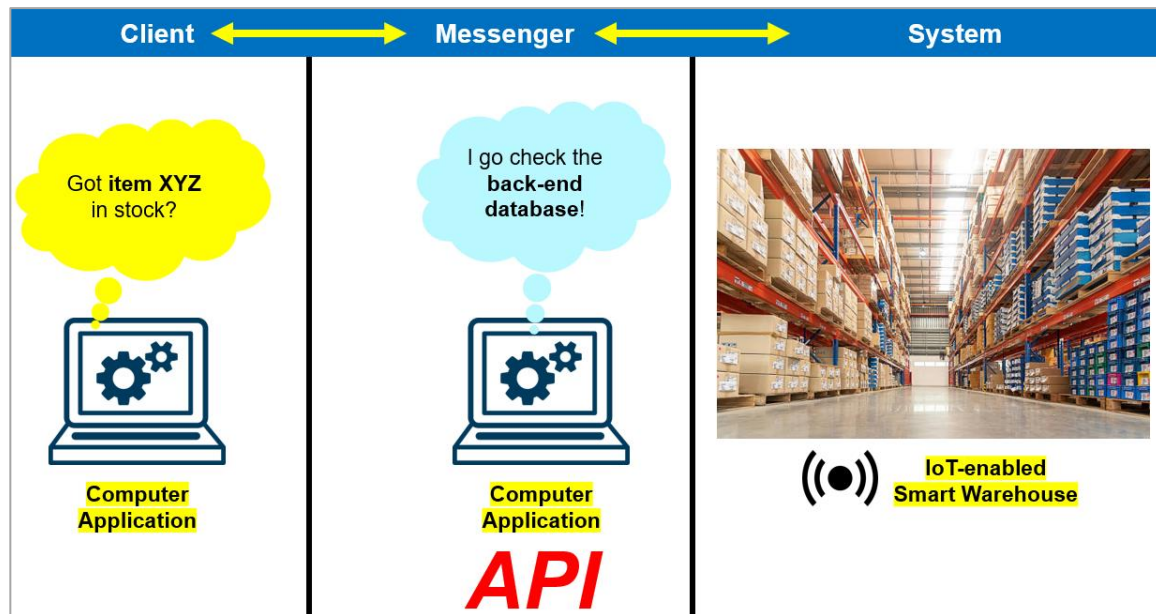
Agenda

AXIOS {JSON}



APIs

- Axios & Async HTTP requests
- JSON Data Processing
- jQuery



Project (25%) - *Milestones*

- **Initial** “Draft” Proposal Submission

- (Week 8) 8-OCT-2021 (FRI) 12PM SG Time
- Fill out **ALL** sections
- Teaching team will review and provide feedback by **Week 9 Monday 12PM SG Time**.

- **Final** “Draft” Proposal Submission

- (Week 9) 17-OCT-2021 (SUN) 12PM SG Time
- Address and resolve **ALL feedback comments/questions** by the teaching team.

- **Consult Teaching Team** (between **NOW** and **Initial Draft Proposal Submission**)

- Look for us for **discussion** if you’re UNSURE about any aspect of **proposal writing**.
- You can add your **questions** as **comments** in your **Proposal Google Doc** – and simply **Slack DM** us to nudge us for help.
- We can provide feedback/answers in the **Proposal Google Doc** or via online consultation.
- Let us know how we can help you! Don’t wait until the last minute please ~~

Source Code Files

eLearn → Content → Session 6 → In Class → **Week6.zip**

- **Unzip** it into your **webroot** (*any meaningful sub-directory*), for example:
 - (WAMP) C:\wamp64\www\is216\...\b>Week6
 - (MAMP) /Applications/MAMP/htdocs/is216/...\b>Week6
- You don't have to follow the above path – it's just an example.
 - But we DO strongly encourage you to keep source code files **organized** so that you can easily **search** them during **Lab Tests**

Things to Finish Before Session 7

1. Complete:
 - [JavaScript – Part 3 \(API, JSON & AJAX\) – Challenges 13, 14, 15](#)
 - **Resource files (*Bootstrap 5.1 version !!!*)** can be downloaded from [this link](#)
2. eLearn → Content → Session 7 → Before Class
 - Watch the video
 - Complete [Session 7] Pre-Class Quiz (Vue.js – Basics)



KrazyMatch

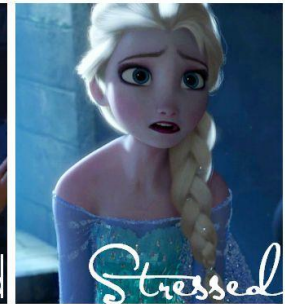
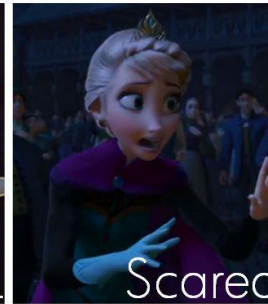
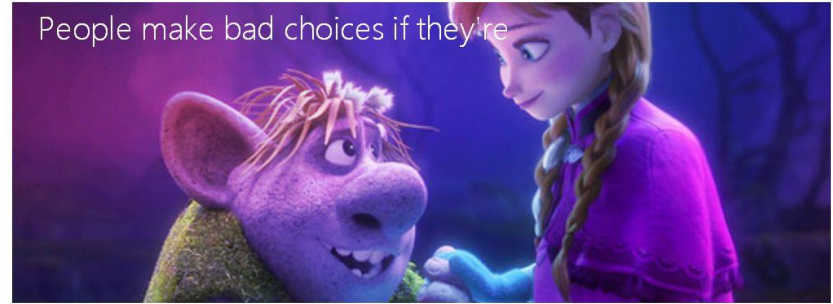
Business Problem



- Singaporeans are **busy**. A [2020 news article](#) showed that Singapore was world's 2nd most **overworked** city.
- Singaporeans are **stressed**. A [2019 Straits Times article](#) reported that **18%** of national healthcare expenditure was on stress-related illnesses annually.

Business Problem

- Recent studies have shown that **too many choices** exhaust people – making us **unhappy** and leading us to sometimes *abscond from making a decision all together*.
- In the 21st century, we have so many options that we get **stressed** every time we have to make a decision... because we're *worried about making the wrong decision*.



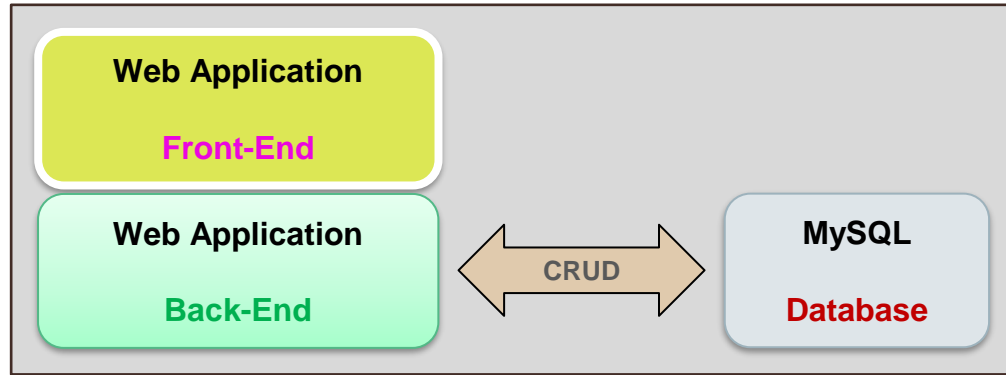
Solution

- Are you **busy**?
- Are you **stressed**?
- Are you **single** and looking for **the right someone**?
- Do you find yourself screaming
"Goodness! **Don't flood me** with **choices**!
Just **find me someone**!"
- **KrazyMatch** will *make that decision* **FOR YOU**
- All you have to do is **Sign Up** today for **FREE**



Solution Architecture *(in IS113)*

KrazyWoman, Inc.

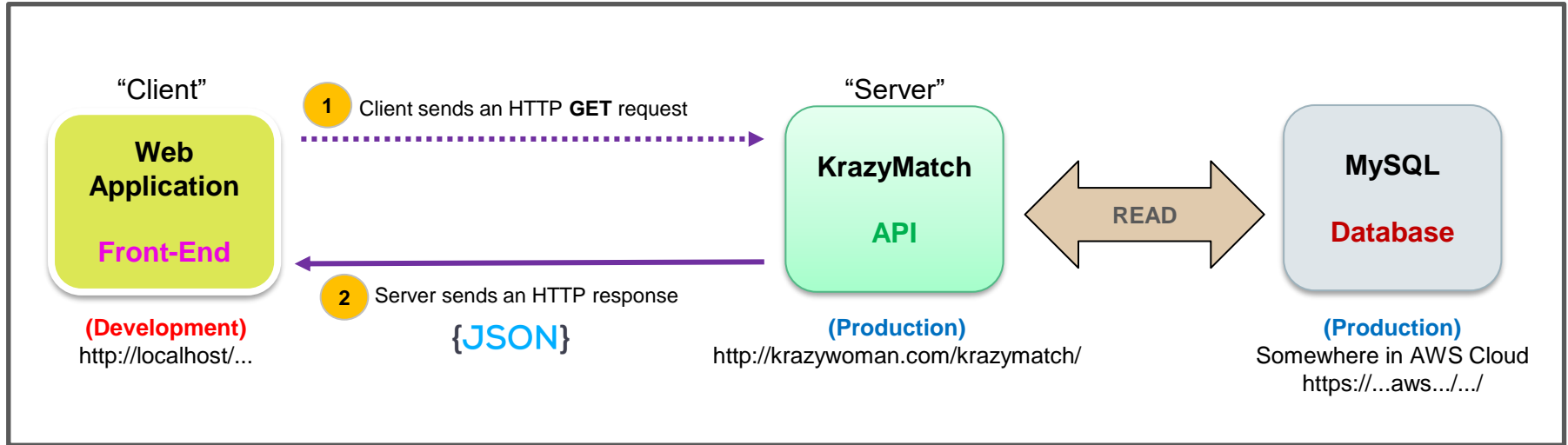


(Development)
<http://localhost/...>

Front-End & **Back-End** are **tightly coupled** (belong to the same code base)
Web App & **Database** reside on the same server (yikes! Single point of failure...)

Solution Architecture *(Phase 1)*

KrazyWoman, Inc.

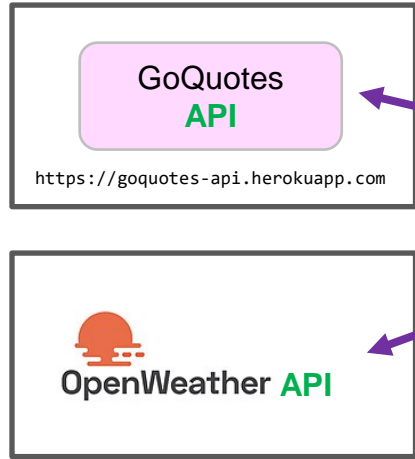


Front-End Web App & API App are de-coupled

Front-End Web App & API App & Database are hosted in 3 different locations (no single point of failure)

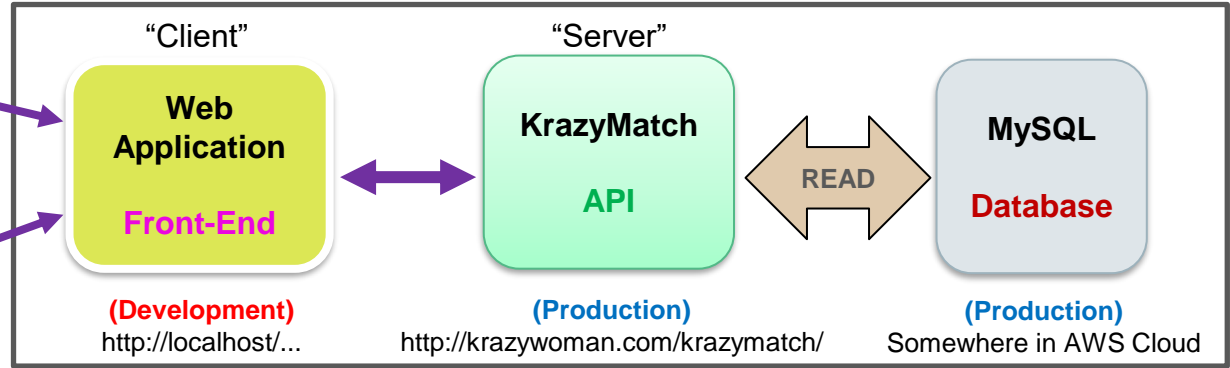
Solution Architecture *(Phase 2)*


External to KrazyWoman, Inc.



{JSON}

KrazyWoman, Inc.






KrazyMatch

(Phase 1)

Architecture Diagram



Tools & Dev Env Setup

- **Download & Install Postman**

- <https://www.postman.com/downloads/>
- **Sign Up** (*create an account*)

- **FollowMe → krazymatch**

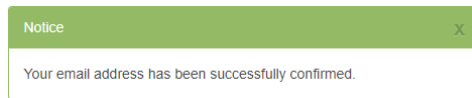
- Move **krazymatch** folder directly under your **Webroot**

Windows	C:\wamp64\www\krazymatch
Mac	/Applications/MAMP/htdocs/krazymatch

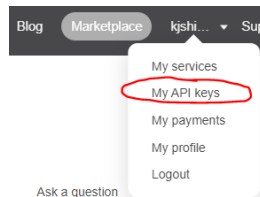
- Follow me ... to set up this **API** internally on **localhost**
 - **krazymatch → api → db → load.sql**
 - Go to <http://localhost/phpmyadmin>, log in, and import load.sql (or run the SQL statements inside the file by going to **SQL** tab)
 - **Krazymatch → api → config → database.php**
 - (If necessary) Change **username** / **password** / **port**
- **Don't WORRY about the PHP code** – it's NOT tested in IS216. Treat it as a **black box** for the purposes of front-end web app development.
- We WILL eventually make **KrazyMatchWebApp** (front-end app) call **krazymatch API** hosted **externally** at <http://krazywoman.com/krazymatch/>

Tools & Dev Env Setup *(continued...)*

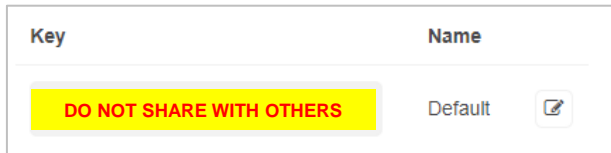
1. Go to <https://openweathermap.org/> and **Sign Up** → **Verify Your Email**



2. Sign In → (menu) → Click on your username → **My API keys**



3. You will see the **default key** → this is your **API Key**



It is NOT activated yet! It takes ~30 minutes to 1 hour.



LATER today... You will need **this API Key** to programmatically **CALL** OpenWeatherMap.org's weather API endpoints

Copy this API Key into a Sticky Note *(or somewhere on your computer for easy access – Do NOT share it with others)*

Follow Me (Code Together)

- **FollowMe → KrazyMatchWebApp**

1. home.html → Click **Sign In**
2. match.html
3. match.js → code up
 - display_default()
 - call_krazymatch_api()
 - populate_today_pick_box()
 - process_not_my_type()
 - process_totally_my_type()



KrazyMatch

(Phase 2)

Architecture Diagram





Follow Me (Code Together)

- **FollowMe → krazymatch**

1. home.html → Click **Sign In**
2. match.html
3. match.js → code up:
 - display_default()
 - call_krazymatch_api()
 - populate_today_pick_box()
 - process_not_my_type()
 - process_totally_my_type()
 - call_quotes_api() → [Activity 1](#)
 - call_weather_api() → [Activity 2](#)

Activity 1: Quotes API (★ ★)

1. Complete `call_quotes_api()` such that it makes a request to the below **API endpoint**:
`https://goquotes-api.herokuapp.com/api/v1/random?count=1`
2. Use **Postman** to **test** this API endpoint. Inspect the **response**. What does it have? What do you need to retrieve from the response?

match.html	match.html (AFTER clicking "Totally My Type" button)																																								
<div data-bbox="81 434 898 464">KrazyMatchHello, Suzy B.</div> <div data-bbox="81 485 241 518">Your Profile</div> <div data-bbox="81 524 892 693"><table><tr><td>Name</td><td>Suzy Bae</td></tr><tr><td>Age</td><td>25</td></tr><tr><td>Gender</td><td>Female</td></tr><tr><td>Occupation</td><td>Student</td></tr><tr><td>City</td><td>Singapore</td></tr></table></div> <div data-bbox="81 737 241 769">Today's Pick</div> <div data-bbox="81 775 892 966"><table><tr><td>Name</td><td>Hugh Simpson</td></tr><tr><td>Age</td><td>35</td></tr><tr><td>Gender</td><td>Male</td></tr><tr><td>Occupation</td><td>Musician</td></tr><tr><td>City</td><td>Sydney</td></tr></table></div> <div data-bbox="396 977 589 999"><div>Not My Type</div><div>Totally My Type</div></div> <div data-bbox="724 982 801 1021">Click</div>	Name	Suzy Bae	Age	25	Gender	Female	Occupation	Student	City	Singapore	Name	Hugh Simpson	Age	35	Gender	Male	Occupation	Musician	City	Sydney	<div data-bbox="1085 434 1806 464">KrazyMatchHello, Suzy B.</div> <div data-bbox="1085 485 1226 518">Your Profile</div> <div data-bbox="1085 524 1796 671"><table><tr><td>Name</td><td>Suzy Bae</td></tr><tr><td>Age</td><td>25</td></tr><tr><td>Gender</td><td>Female</td></tr><tr><td>Occupation</td><td>Student</td></tr><tr><td>City</td><td>Singapore</td></tr></table></div> <div data-bbox="1085 704 1226 737">Today's Pick</div> <div data-bbox="1085 742 1796 889"><table><tr><td>Name</td><td>Hugh Simpson</td></tr><tr><td>Age</td><td>35</td></tr><tr><td>Gender</td><td>Male</td></tr><tr><td>Occupation</td><td>Musician</td></tr><tr><td>City</td><td>Sydney</td></tr></table></div> <div data-bbox="1107 917 1787 1021"><div>A Random Quote You Can Share with Today's Pick</div><div>So it's the kind of business where you can't wait to get up in the morning and read the papers, or listen to what's on the news, and you know, how the world's going to change. — Sanford I. Weill</div></div>	Name	Suzy Bae	Age	25	Gender	Female	Occupation	Student	City	Singapore	Name	Hugh Simpson	Age	35	Gender	Male	Occupation	Musician	City	Sydney
Name	Suzy Bae																																								
Age	25																																								
Gender	Female																																								
Occupation	Student																																								
City	Singapore																																								
Name	Hugh Simpson																																								
Age	35																																								
Gender	Male																																								
Occupation	Musician																																								
City	Sydney																																								
Name	Suzy Bae																																								
Age	25																																								
Gender	Female																																								
Occupation	Student																																								
City	Singapore																																								
Name	Hugh Simpson																																								
Age	35																																								
Gender	Male																																								
Occupation	Musician																																								
City	Sydney																																								

Activity 2: Weather API (☆☆)

1. Sign into <https://openweathermap.org/>
2. By now, you SHOULD have gotten your **API Key** and saved it on your local computer somewhere
3. Go to (menu) **API** → **Current Weather Data** → Click **API doc**



4. We will retrieve the **current weather info** for a certain **city** (e.g. **Singapore**).

By city name

You can call by city name or city name, state code and country code. Please note that searching by states available only for the USA locations.

API call

```
api.openweathermap.org/data/2.5/weather?q={city name}&appid={API key}
```

API Endpoint

`http://api.openweathermap.org/data/2.5/weather?q=Singapore&appid=YOUR_API_KEY`

5. In **Postman**, test the above **API endpoint** with **city** "Singapore" and with **your own API Key**.
 - What **response** do you receive?

Go to the next slide for more instructions

[Back to Menu](#)

Activity 2: Weather API (☆☆)


Complete `call_weather_api()` such that it retrieves the **current weather info** of **Singapore** from OpenWeatherMap's API

match.html

KrazyMatch

Hello, **Suzy B.**

Your Profile



Name

Suzy Bae

Age

25

Gender

Female


Occupation

Student

City

Singapore

Today's Pick



Name

Ross Frenz

Age

32

Gender

Male

Occupation

Football Coach

City

Chicago

Not My Type

Totally My Type


match.html

(**AFTER** clicking "Totally My Type" button)

KrazyMatch

Hello, **Suzy**

Your Profile



Name

Suzy Bae

Age

25

Gender

Female


Occupation

Student

City

Singapore

Today's Pick



Name

Ross Frenz

Age

32

Gender

Male

Occupation

Football Coach

City

Chicago

A Random Quote You Can Share with Today's Pick

I am as bad as the worst, but, thank God, I am as good as the best.
— Walt Whitman

My City

Temperature: **28.68**
Humidity: **71**

My Date's City

Temperature: **9.95**
Humidity: **61**

Suzy's city (Singapore) weather info

[Back to Menu](#)

Asynchronous HTTP Requests

What is Asynchronous Request?

- Asynchronous requests can be used to:
 - read data (e.g. JSON, XML, Text) from backend server - after the page has loaded
 - update a web page without reloading the page
 - send data to the server - in the background
- Compared to Synchronous requests (e.g. requests sent via HTML Forms),
 - less data usage, faster, more interactive

Axios

Promise based API for sending Asynchronous HTTP requests to backend server / REST endpoints and doing create/read/update/delete (CRUD) operations

Can be used with plain JavaScript or with frameworks like Vue.js

What is promise-based?
Shortly.

Basic Axios Methods

- Basic Axios Methods

`axios.get(url[, config])`

`axios.post(url[, data[, config]])`

- Axios Response Object

- After sending an async. request, server returns a response.
- Axios response object consists of:
 - data – payload
 - status – HTTP code (2xx, 3xx, 4xx)
 - statusText – HTTP status message ('OK', 'Moved Permanently', 'Client Error')
 - headers – information about the response, e.g. content type, cookies, etc.

How Does Axios Work?

1. import AXIOS library
2. execute GET/POST request to the endpoint of your backend server
3. Use the *then* method to act on the response result

Example: *wk6example1.html*

```
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
<button type="button" onclick="getAdvice()">Click Me!</button>
<script>
    function getAdvice() {
        axios.get(url)
            .then(response => {
                // process response.data object
            })
            .catch(error => {
                // process error object
            });
    }
</script>
```

How Does Axios Work?

Sending **GET** requests with parameters

Example: *wk6example2.html*

```
axios.get(url, {  
    params: {  
        name1 : value1,  
        name2 : value2  
    }  
})  
    .then(response => {  
        // process response.data object  
    })  
    .catch(error => {  
        // process error object  
    });
```

How Does Axios Work?

Sending **POST** requests with parameters

Example: *wk6example3.html*

```
axios.post(url, {  
    name1 : value1,  
    name2 : value2  
})  
    .then(response => {  
        // process response.data object  
    })  
    .catch(error => {  
        // process error object  
    });
```

JavaScript Arrow Functions

Arrow functions allow us to write shorter function syntax.

An arrow function is a function which has no name and is used once only.

Example

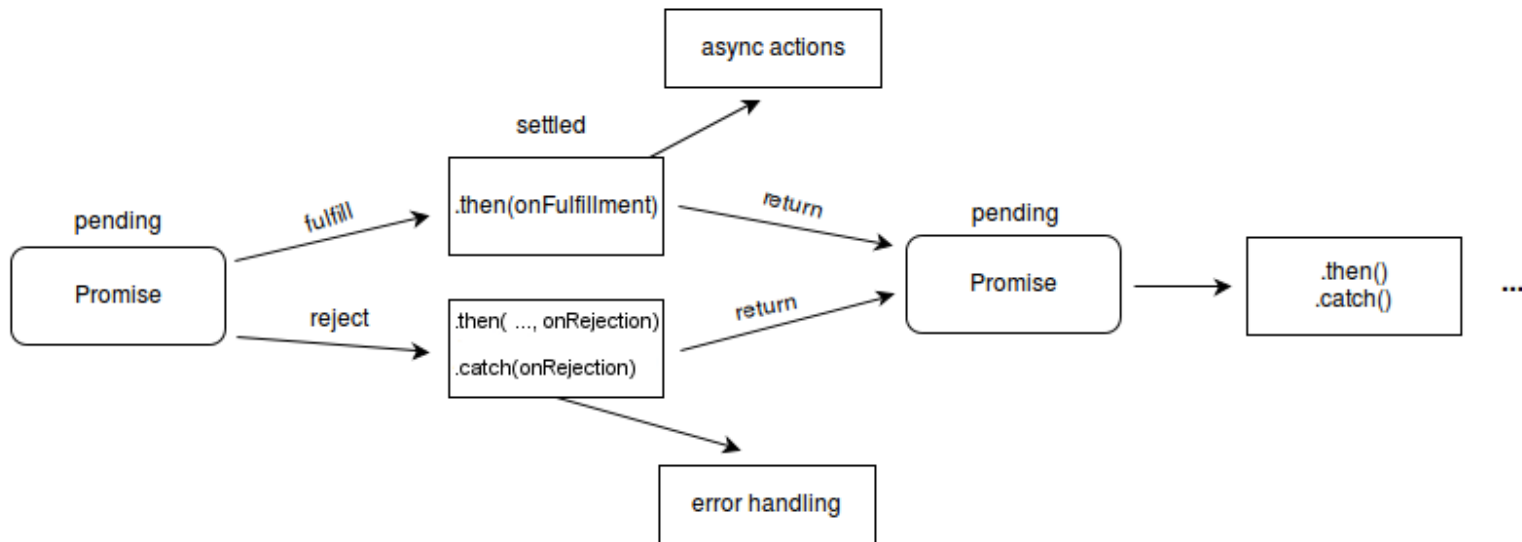
```
function onceFunc (msg) {  
    return "Hello World!" + msg;  
}
```

is equivalent to

```
msg => {  
    return "Hello World!" + msg;  
}
```

Promise

In JavaScript, a Promise is an object used as a proxy for a value not yet known (similar to Future in Java).



then() and catch()

Given a Promise object x,

x.then(fn) executes the function fn if the x is in a fulfilled state; and returns another Promise object; otherwise x is returned.

x.catch(fn) execute the function fn if x is in a rejected state; and returns another Promise object; otherwise x is returned.

Example: *wk6example1.html*

```
//returns a promise
axios.get('http://api.advice Slip.com/advice')

//call then to handle the good case
.then(response => {
  console.log(response.data.slip.advice)
})

//call catch to handle the bad case
.catch(error => {
  console.log(error.message)
})
```

Exercise 1: Search

- Given the resources: search.html, getHint.php (endpoint that returns Text data)
- Modify **search.html** such that when a user types a character in the input field (**onkeyup** event), a function called **showHint()** is executed, which suggests the possible names that the user is searching for.
- **showHint()** uses Asynchronous GET request to getHint.php to obtain those possible names

Type a name in the input field below:

Name:

Suggestions: Anna, Amanda

Type a name in the input field below:

Name:

Suggestions: Amanda

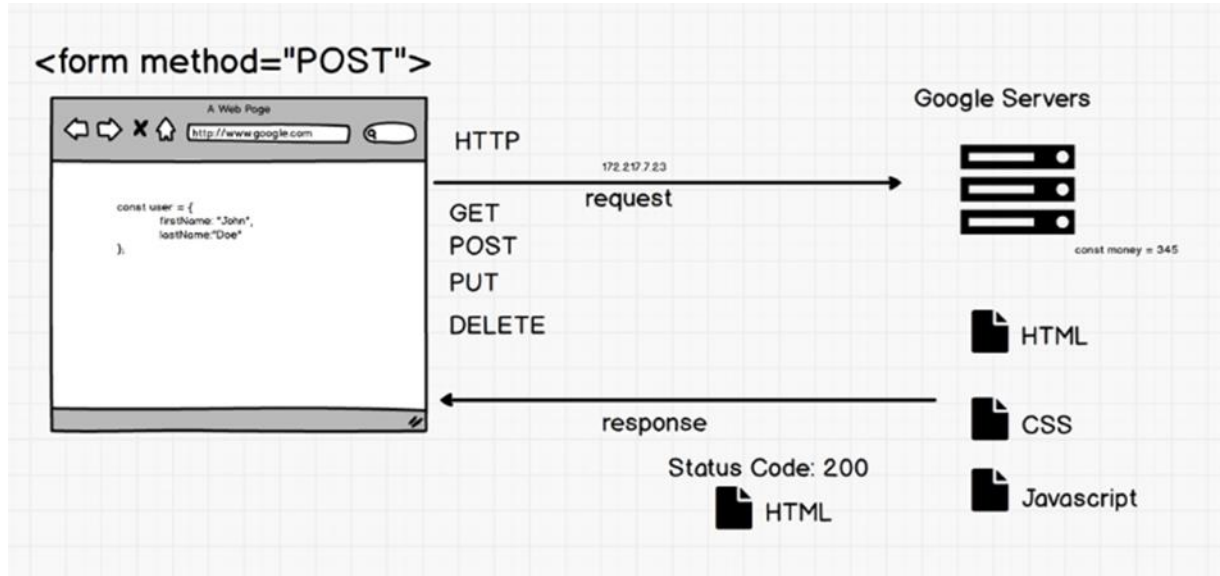
Type a name in the input field below:

Name:

Suggestions: no suggestion

JSON

Why JSON?



You can't just send JS objects directly to servers. Servers only understand Text. JSON or XML can be used to transfer data in text format.

JSON

What is JSON?

Lightweight data-interchange format for storing and exchanging data.

JSON has a JavaScript syntax but is text only.

JavaScript values/objects can be converted into JSON and send it to the server; JSON received from the server can be converted into JavaScript values/objects.

Language independent; can be used by any programming language.

Example:

```
{  
  
  "coord": { "lon": 139, "lat": 35 },  
  
  "weather": [  
    {  
      "id": 800,  
      "main": "Clear",  
      "description": "clear sky",  
      "icon": "01n"  
    }  
  ],  
  
  "base": "stations"  
}
```

JSON Syntax

JSON objects are written in key/value pairs, separated by commas and surrounded by curly braces:

In JSON, keys must be strings, written with double quotes.

The file type for JSON files is ".json".

Example:

```
{  
  "name" : "Mary",  
  "age" : 26,  
  "hobby" : [ "swimming", "badminton" ],  
  "isMarried" : false  
}
```

JSON Data Types

In JSON

There are 6 types: null, string, number, Boolean, JSON object, and array.

In Javascript

Function `typeof` returns 6 different values.

Object, boolean, function, number, string, and undefined

```
typeof([1,2,3]); // object!
```

Example: *wk6example4.html*

```
{  
  "employee": {  
    "name" : "Mary",  
    "age" : 26,  
    "hobby" : [ "swimming" ],  
    "isMarried" : false  
  }  
}
```

"..." is string

26 is a number

[...] is an array

false / true: boolean

{ ... }: object

Nested JSON

Nested Objects

```
person = {  
  "name": "John",  
  "age": 30,  
  "cars": {  
    "car1": "Ford",  
    "car2": "BMW",  
    "car3": "Fiat"  
  }  
}
```

Nested Arrays

```
person = {  
  "name": "John",  
  "Age": 30,  
  "cars": [  
    { "name": "Ford",  
      "models": [ "Fiesta",  
                  "Mustang" ]  
    },  
    { "name": "BMW",  
      "models": [ "320",  
                  "X3",  
                  "X5" ]  
    },  
    {  
      "name": "Fiat",  
      "models": [ "500",  
                  "Panda" ]  
    }  
  ]  
}
```

JSON vs Form Data

Form Data

Typically, we exchange data between client and server using forms (<form> tag) in HTML via POST or GET request

But this is a synchronous activity – request/response protocol

JSON

With JSON, we can also now grab the contents of the <input> in a form and submit those with JSON instead of form data

You can submit to server whenever you want through [Asynchronous HTTP requests](#)

Exercise 2: Dynamic Table

- Given the resources: info.html and getData.php (endpoint that returns JSON)
- Modify **info.html** so that it generates the HTML table dynamically based on the value of a drop down menu and the limit value
- Requirement: use Async. POST request to getData.php

Choose an option:

Limit: 1

- ✓ Customers
- Products
- Suppliers

Choose an option:

Limit: 1 Products

products:

name	price
iPhone	2049

Choose an option:

Limit: 3 Products

products:

name	price
iPhone	2049
Samsung	1699
Huawei	1499

Exercise 2 ctd.

- Modify **info.html** so that it generates the HTML table dynamically based on the value of a dropdown menu and the limit value:

Choose an option:

Limit:

customers:

name	age	city
Jack	30	London
Mary	24	Paris
Dan	18	Prague
Olav	32	Moscow
Billie	43	Barcelona

Choose an option:

Limit:

suppliers:

name	age	city
Bane	35	Tokyo
Joker	44	Seoul
Penguin	28	KL
Dent	38	Singapore

Extra Exercise (Optional): A Blog Service

In **Week6** folder, there is a sub-folder called **blog**.

- Copy this **blog** folder under your web server's **webroot**
- (Mac) **/Applications/MAMP/htdocs/blog**
- (Win) **C:\wamp64\www\blog**

Next, do the following:

- 1) Start your WAMP/MAMP server
- 2) Go to **<http://localhost/phpmyadmin>** and log in
 - (Win) Username: root, Password: <empty>
 - (Mac) Username: root, Password: root
- 3) After logging in, import **create.sql** into MySQL
- 4) Go to **ConnectionManager.php** and edit **password** and **port number** accordingly as per your local computer setup

Test Case

<http://localhost/blog/getPosts.php>

Go to the above URL in your web browser.

Make sure some JSON data is received.

Extra Exercise (Optional): A Blog Service ct.

Complete the TODO in extra-addpost.html so that it uses the service provided by addPost.php to add a new blog post.

Add a New Blog Post

Subject:

Entry:

Mood: ▼

Post added successfully

Click [here](#) to return to Main Page

Using Google Map APIs

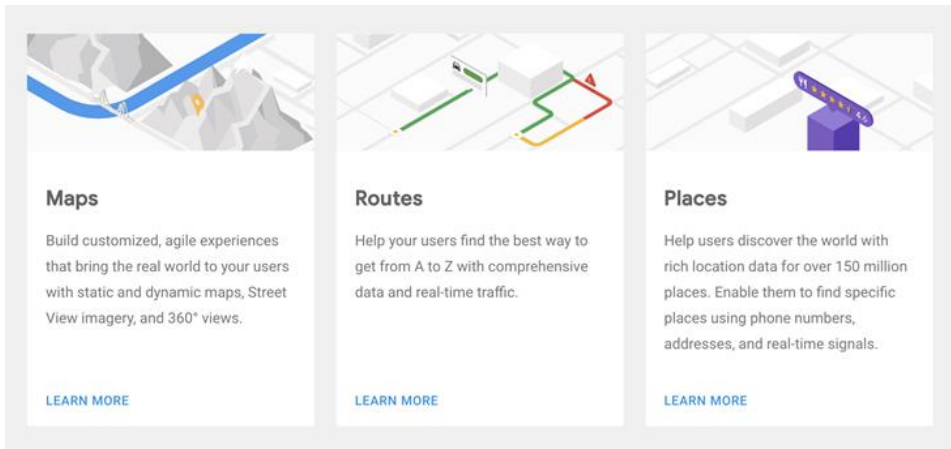
Google Map API

Google Map API

<https://cloud.google.com/maps-platform/>

API: Maps, Routes, Places, Gaming Solution
and more

[Google Maps JavaScript API V3 Reference](#)



Google Map API

Have a look at this example.

Make sure you use your own Google Maps API key.

Given that you've keyed in a valid Google Maps API key, you should be able to see a map after the webpage has loaded in a web browser.

Example

wk6example5.html

Geolocation and Google Map

The HTML Geolocation API is used to locate a user's position.

[HTML Geolocation API](#)

[Geolocation API - Web APIs | MDN](#)

Example: *wk6example6.html*

https://developers.google.com/maps/documentation/javascript/examples/map-geolocation#maps_map_geolocation-javascript

Exercise 3: Preparation

Enable Geocoding API.

Google Cloud Platform

IS113

Google Maps

Overview

APIs

Metrics

Support

Enabled APIs

Select an API to view details. Figures are for the last 30 days.

API ↑	Requests	Errors
Distance Matrix API	0	0
Geocoding API	11	10
Geolocation API	0	0
Maps JavaScript API	22	0
Roads API	0	0

Check your API key works using
wk6example6.html

Try the following URL in your browser and
make sure that some JSON data is received.
Analyze the data a bit.

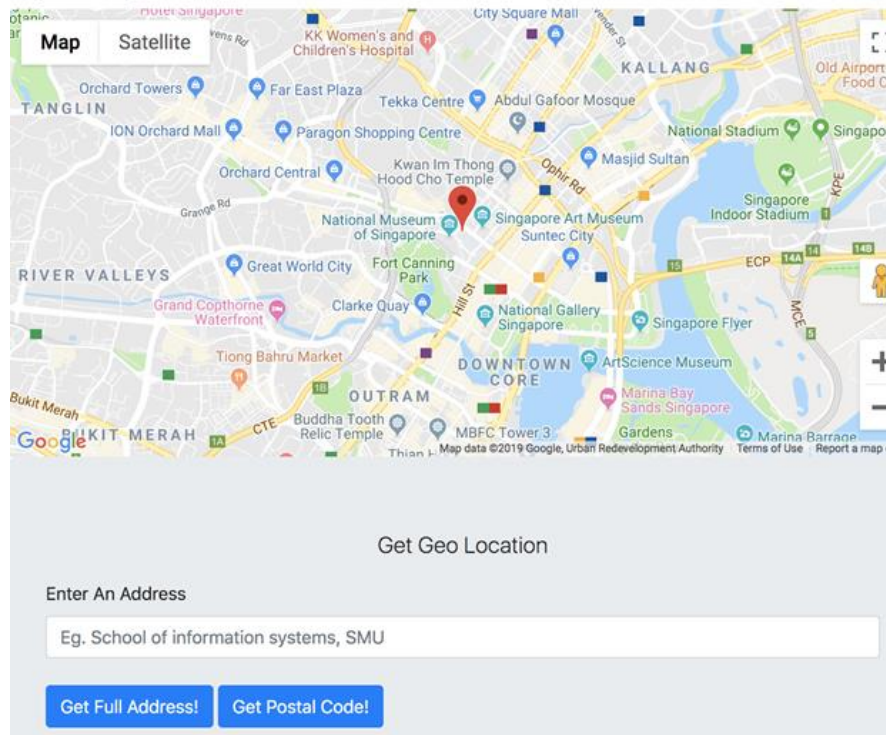
[https://maps.googleapis.com/maps/api/geocode/
json?address=1600+Amphitheatre+Parkway,+M
ountain+View,+CA&key=yourapikey](https://maps.googleapis.com/maps/api/geocode/json?address=1600+Amphitheatre+Parkway,+Mountain+View,+CA&key=yourapikey)

Exercise 3

Given ex3.html, implement a client app (for geocoding) such that given a street address, the app should show its geolocation on the map and put a mark.

The app should make use of Google's Geocoding API provided as web services.

My Google Map



Exercise 3 (Cont'd)

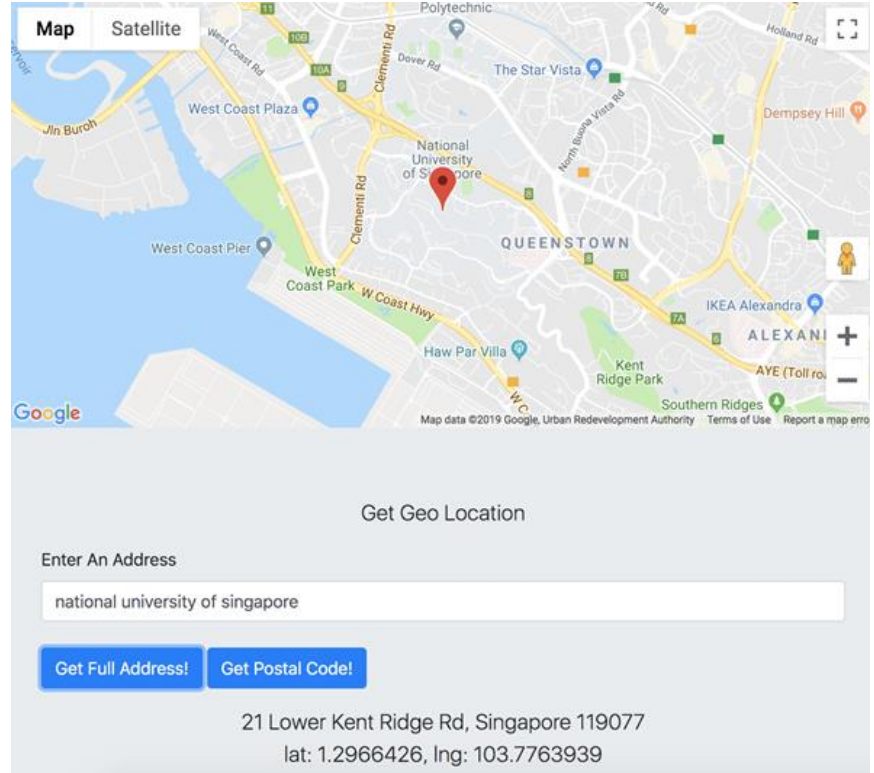
On page load, the app shows a map with a marker at a default address

It has an input field where the user can either enter a partial address or a postal code.

Upon clicking the button, the app shows the corresponding information.

Also the map automatically refreshes to the new address that the user just provided.

Your code should properly deal with invalid addresses.





Using Firebase Database



This topic is relevant to your **IS216 Project**
It will NOT be tested in Lab Tests/Final Exam

Firestore

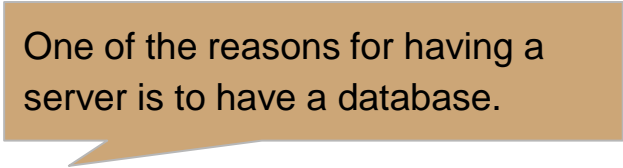
Firestore

Firestore is a platform developed by Google for creating mobile and web applications.

It supports multiple databases.

Firestore Realtime Database

A realtime database which organizes data in the form of JSON.



One of the reasons for having a server is to have a database.

Firestore Realtime Database

No server required: Firestore Database can be accessed directly from a web browser; there's no need for an application server.

Realtime: data is synchronized across multiple devices efficiently.

Offline: Firestore database persists your data to disk when offline. Once connectivity is reestablished, the client device receives any changes it missed, synchronizing it with the current server state.

Firestore Realtime Database

Data are organized as a JSON tree

No table; no SQL.

Add/delete/update data by modifying the JSON tree.

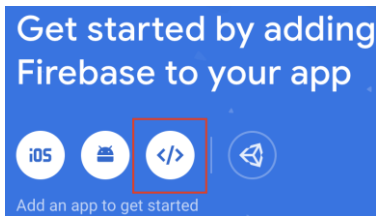
Retrieve data by retrieving one sub-tree at a time (so no mindful of the size of the subtree).



Firestore Realtime Database

Follow the steps [here](#) to enable Firestore Realtime Database

1. [Sign-in](#) Firestore using your Google account
2. Create a Firestore project
3. Register an APP



Now is a good time to set it up.
Make sure that you are using a free plan (i.e., Spark plan)

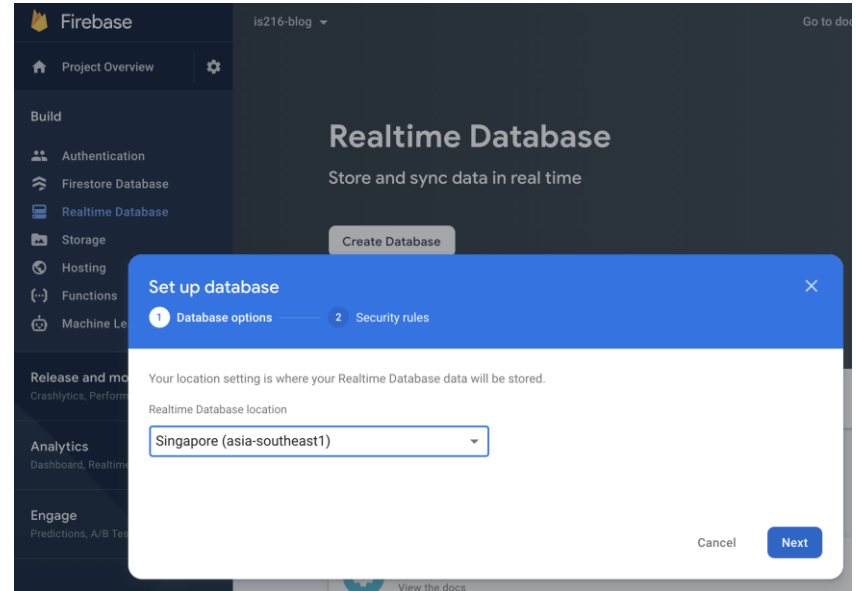
Firestore Realtime Database ctd.

Build -> Real Time Database -> Create

Choose **test mode** instead of **locked mode**

(The Database URL will take a while to show up in your Firebase Config)

[Follow the instructions here to setup Firestore in your JavaScript](#)



Firestore Realtime Database

Store data

Add data into the database by growing the JSON tree with new nodes: using the set function.

API reference

<https://firebase.google.com/docs/reference/js>

Example: *wk6example7.html*

```
function writeUserDataWithCompletion(userId, name, email) {  
  firebase.database().ref('users/' + userId).set({  
    username: name, email: email  
  }, function(error) {  
    if (error) {  
      ... //if error happens  
    }  
    else {  
      ... //if no error  
    }  
  });  
}
```

Firestore Realtime Database

Retrieve data

Retrieve data by obtaining a reference to the subtree and get a dataSnapshot **once**.

API reference

<https://firebase.google.com/docs/reference/js>

Example: *wk6example7.html*

```
function checkIfUserExists(userId) {  
  var user = firebase.database().ref('users/' + userId);  
  user.once('value').then((snapshot) => {  
    if(snapshot.exists()) {  
      ...  
    }  
    else {  
      ...  
    }  
  });  
}
```

Firestore Realtime Database

Retrieve data

React **on** certain data changes at a particular location.

API reference

<https://firebase.google.com/docs/reference/js/firebase.database.Reference#on>

Example: *wk6example7.html*

```
var users = firebase.database().ref('users');
users.on('child_added', (snapshot) =>{
  var today = new Date();
  var dd = String(today.getDate()).padStart(2, '0');
  var mm = String(today.getMonth() + 1).padStart(2, '0');
  var yyyy = today.getFullYear();
  today = mm + '/' + dd + '/' + yyyy;
  document.getElementById("latest").innerHTML = "Last user registered at " + today;
});
```

Firestore Realtime Database

Delete data

Delete data by obtaining a reference to the subtree and remove().

API reference

<https://firebase.google.com/docs/reference/js>

Example: *wk6example7.html*

```
function deregister(userId) {  
  
    firebase.database().ref('users/' + userId).remove()  
    .then(function() {  
        ... //if success  
    })  
    .catch(function(error) {  
        ... //if error occurs  
    });  
  
}
```

Firebase Realtime Database

Update data

Update data by obtaining a reference to the database and update().

API reference

<https://firebase.google.com/docs/reference/js>

Example

```
function updateEmail() {  
  var userId = document.getElementById("id").value;  
  var name = document.getElementById("name").value;  
  var email = document.getElementById("email").value;  
  var updates = {};  
  
  updates['/users/' + userId + "/" + 'name'] = name;  
  updates['/users/' + userId + "/" + 'email'] = email;  
  firebase.database().ref().update(updates);  
}
```

Exercise 4

Given *wk6example7.html*,

1. Update the project information to yours.
2. Enrich the page to display the total number of users who have registered. Note that the number should be updated accordingly when new users register or a user de-registers.

Hint: listen to database change through *reference.on(event, ...)*

Firebase Authentication

You should never store password in Firebase Realtime Database.

Use Firebase authentication instead.

Multiple authentication options are available.

<https://firebase.google.com/docs/auth>

Easiest option:

<https://firebase.google.com/docs/auth/web/firebaseui>

Example: *wk6example8.html*

Using Google authentication

(To run the above file, start your WAMP server and access the file in your browser with the right URL, e.g., localhost/wk6example8.html)

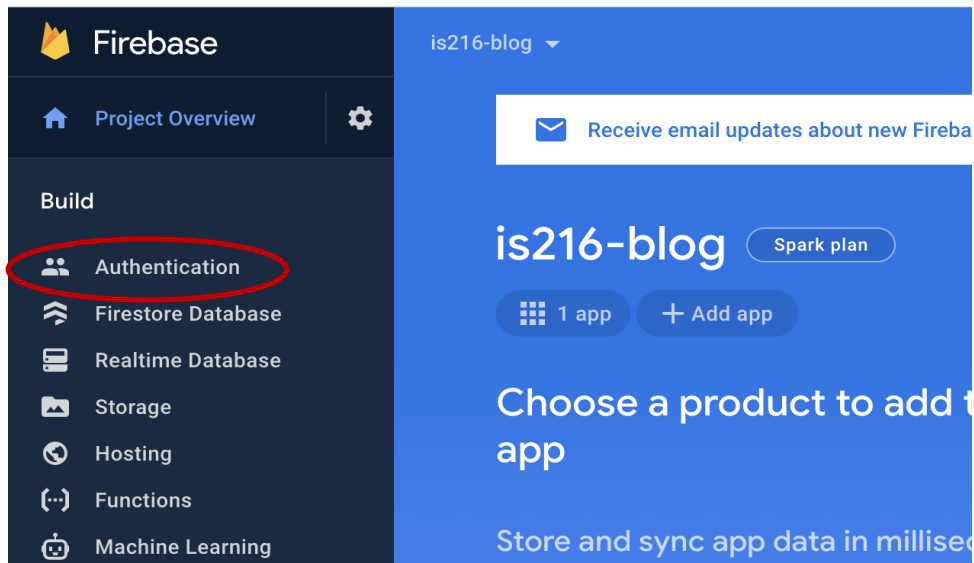
Firebase Authentication

You need to enable “Authentication” in your firebase project

Go to your project console:

<https://console.firebase.google.com/u/0/>

Select “Authentication”



Firebase Authentication




Choose your preferred “Authentication” option

is216-blog ▾

Authentication

Users Sign-in method Templates Usage

Sign-in providers

Provider	Status
 Email/Password	Enabled
 Phone	Disabled
 Google	Enabled

Exercise 5

Given *wk6example8.html*,

1. Update the project information to yours.
2. Re-direct to the *wk6example7.html* page once the user logs in.

Hint: You will need to enable Firebase Authentication for Google Sign-in for this to work.

Firestore: More

Firestore

A newer version which is more scalable and reliable.

Firestore Realtime Database or Firestore?

<https://firebase.google.com/docs/database/rtdb-vs-firestore>

Or do we need to have a server?

Take-Away Message

JSON - a structured text for server-client communication

Axios – send asynchronous HTTP GET/POST requests

```
axios.get( url, { params: { ... } } )
```

```
.then( response => { ... } )
```

```
.catch( error => { ... } )
```