# WEATHER API

<https://openweathermap.org/current>

<https://www.youtube.com/watch?v=ecT42O6I_WI>

## APPLICATION PROGRAMMING INTERFACE

- Need to get severs to talk to eachother (openweathermap is a server and I want it to talk to my server)

- Check out examples of API calls

<https://samples.openweathermap.org/data/2.5/weather?q=London,uk&appid=b6907d289e10d714a6e88b30761fae22>

### API KEY

- Identifies you so you can use it in the code and make an API query

- Need it to use the data (identifies you as user)

### API DOCUMENTATION

- Most sites will let you know how to use it

#### URL QUERY STRING

- You need to describe where you want weather from, what time, etc

- query string comes at the end of URL (/?city=london&APIKEY=keyvalue)

- Start with questinong mark (?)

- This tells you want to give some parameters (name=value pairs)

- Ampersand(&) strings together multiple queries

- Then need API key at end

#### JSONVIEW

- Chrome extension that automatically formats the data

### fetch()

<https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch>

- Retrieve data from the API

- Part of the window object and not JS language

- Most modern browsers support

- Returns a promise since asynchronous data

- For node, need specific module to use

<https://www.youtube.com/watch?v=tc8DU14qX6I&t=64s>

#### API ENDPOINT

- There will be certain endpoints you can fetch

api.openweathermap.org/data/2.5/weather?q={city name},{country code}

let url = ‘https://samples.openweathermap.org/data/2.5/weather’ ,

qString = ’?q=Calgary,uk&appid=’ + apiKey + ””

fetch(url+qString);

.then((response)=>{

return response.json();

}

.then((data)=>{

// Act on the promise returned by json method

console.log(data)

}

- returns promise, need to convert to JSON data

##### PROMISE

###### .then()

- .then() is used to handle asynchronous response

- A second .then() will then turn the promise into usable data, this is where we can use the data (for example use events to use it in html (document.getElementById().textContent(respinseData))

##### HANDLE ERRORS (.catch())

- Put this at the end of promise handling to handle any errors

**fetch(url+qString);**

**.then((response)=>{**

**return response.json()**

**}**

**.then((data)=>{**

**// Act on the promise returned by json method**

**console.log(data)**

**}**

**. catch(err =>{ console.log(err)})**

###### async/await

- Makes more readable

- Can only use ‘await’ in the context of an ‘async’ function

async function catchRainbow() {

const response = await fetch(‘rainbow.jpeg’); // Get response, wait till occurs and save to response variable

const blob = await json(response); (json blob) // Can now use the data in blob

document.getElementById().textContent(blob))

}

- Then just need to call this method to use it

- Handle error

catchRainbow()

. catch(err =>{ console.log(err)})

#### json()

- Converts fetched data into JSON object

- Returns a promise as well

<https://developer.mozilla.org/en-US/docs/Web/API/Body/json>

#### GET ICONS FROM OPENWEATHERAPP

<https://openweathermap.org/weather-conditions>

<https://www.w3schools.com/jsref/prop_img_src.asp>

- In the json object 🡪 *let* icon *= data.weather[0].icon*;

**async function getWeatherIcon(data){**

**let icon = data.weather[0].icon;**

**let imageUrl = `http://openweathermap.org/img/wn/${icon}@2x.png`;**

**let iconImage = document.getElementById('temperature-icon');**

**iconImage.src = imageUrl;**

**}**

#### GET 5 DAY FORECAST

<https://openweathermap.org/forecast5>

**async function getFiveDayForecast(location){**

**let latitude = parseFloat(location[0]);**

**let longitude = parseFloat(location[1]);**

**let url = 'https://api.openweathermap.org/data/2.5/forecast';**

**let qString = "?lat=" + latitude + "&lon=" + longitude + "&units=metric&appid=";**

**let apiKey = '400411a71373f243c1576f706181dfd0';**

**const response = await fetch(url + qString + apiKey);**

**let data = await response.json();**

**console.log(data);**

**}**

#### DATE OBJECT

- Had to use getUTCDate to get day, was giving tomorrows day

<https://www.w3schools.com/jsref/jsref_obj_date.asp>

#### ADDING ELEMENTS THROUGH DOM

<https://www.w3schools.com/jsref/met_node_appendchild.asp>

### GEOLOCATION WEB APP

<https://www.youtube.com/watch?v=3ls013DBcww>

<https://developer.mozilla.org/en-US/docs/Web/API/Navigator>

app.js = the server

index.html = static file

#### SERVER

- The purpose of the server is to host static files (like html)

- It will send them to the client, to be rendered in their browser

- Client never sees ‘app.js’

- Server saves to database

- Authentication

- API keys

- Clients can geolocate itself and return it to the server

#### NAVIGATIOR (GEOLOCATION)

- Check if the browser has geolocation

*if*('geolocation' *in* navigator){

console.log('geolocation available');

}*else*{

console.log('geolocation not available');

}

- Then to get current position

*navigator*.*geolocation*.*getCurrentPosition*((position)*=>*{

console.log(position)

*getWeather*([*position*.*coords*.*latitude*, *position*.*coords*.*longitude*]);

})

<https://www.programmableweb.com/api/reddit>