Pollution API

The pollution section of the tmpst website is one of the more simpler of APIs to receive data from as there are no keys required to access the data from the website. The pollution API is provided by OpenAQ which provides information on a variation of data from the world such as:

- Cities
- Countries
- Pollution Tracking Locations
- Pollution Measurements

These are divided by endpoints which are where the information will be accessed from. Each endpoint can return data in JSON. For each endpoint you have a lot of customizability for querying the API. Parameters for the measurements endpoint include:

- Country
- City
- Location
- Pollution Type
- Coordinates
- And more...

In this tutorial:

Pre-requisites:
- Read the API documentation (https://docs.openaq.org/#api-Measurements)
- The endpoint that will be used will be the:
  - https://api.openaq.org/v1/measurements for pollution measurements
- A suitable method for making a request (Postman: https://www.getpostman.com/downloads/ or in a new tab)

Aims:
- Learn how to request data from one of the OpenAQ feeds in Postman
- Learn how to request data from one of the OpenAQ feeds in a new tab
- Learn how to request data from one of the OpenAQ feeds using AJAX call

Postman Tutorial

In this tutorial, we will create a request to a OpenAQ API feed using the postman application on Windows. Before proceeding with this tutorial, please ensure you have access to the Postman application or are using the online version.

Steps:

1. Choose endpoint to request data from
2. Construct request in Postman

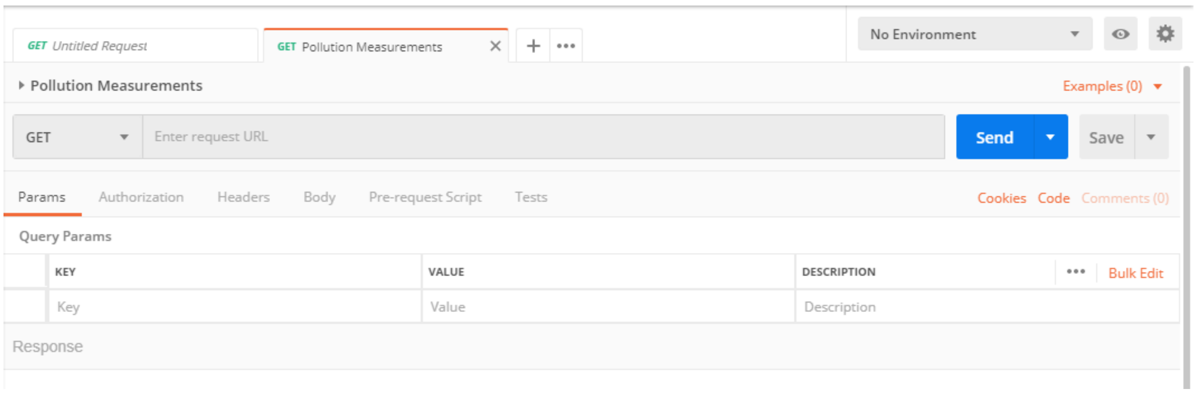
Step 1: Choose endpoint to request data from

As mentioned earlier, OpenAQ offers many pollution feed endpoints to choose from. In this tutorial, not all sources will be shown because the request format is identical for all. It is important to keep in mind that not all sources will have data at the time of the request. For example, making a request for the pollution of a country 2000 years in the future won't yield any results from the request. With that in mind, the endpoint that will be used for this tutorial is /measurements. This endpoint will always have data so it is good as an example. The endpoint is shown below:

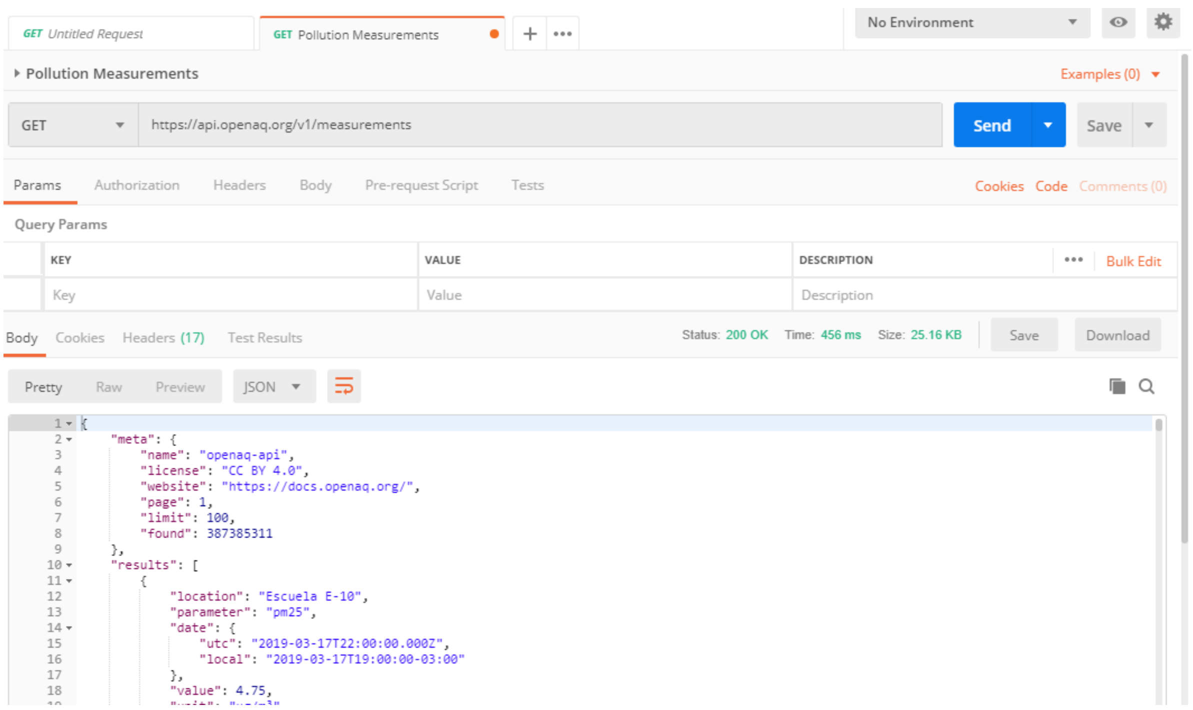
https://api.openaq.org/v1/measurements

Step 2: Construct request in Postman

This tutorial assumes that you have a basic understanding of how to use Postman. If not, we provide a tutorial on how to utilise Postman properly that you may wish to read before you continue with this tutorial. If you know how to use Postman, create a new GET request and give it an appropriate name. The name I will use in this tutorial is "Pollution Measurements". If you have setup the request properly you should see the following:



Nothing will happen if you attempt to send the request as the url box is empty. Now all that is required is to add the url mentioned in step 1 to the url box and hit send. You should see the following:



GET Untitled Request | GET Pollution Measurements

No Environment

Pollution Measurements

Examples (0)

GET https://api.openaq.org/v1/measurements

Send | Save

Params | Authorization | Headers | Body | Pre-request Script | Tests | Cookies | Code | Comments (0)

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Bulk Edit

Body | Cookies | Headers (17) | Test Results

Status: 200 OK Time: 456 ms Size: 25.16 KB

Save | Download

Pretty | Raw | Preview | JSON

```
{
    "meta": {
        "name": "openaq-api",
        "license": "CC BY 4.0",
        "website": "https://docs.openaq.org/",
        "page": 1,
        "limit": 100,
        "found": 387385311
    },
    "results": [
        {
            "location": "Escuela E-10",
            "parameter": "pm25",
            "date": {
                "utc": "2019-03-17T22:00:00.000Z",
                "local": "2019-03-17T19:00:00-03:00"
            },
            "value": 4.75,
```

Part of the JSON response is shown below in plain text:

```
{
    "meta": {
        "name": "openaq-api",
        "license": "CC BY 4.0",
        "website": "https://docs.openaq.org/",
        "page": 1,
        "limit": 100,
        "found": 387385311
    },
    "results": [
        {
            "location": "Escuela E-10",
            "parameter": "pm25",
            "date": {
                "utc": "2019-03-17T22:00:00.000Z",
                "local": "2019-03-17T19:00:00-03:00"
            },
            "value": 4.75,
            "unit": "µg/m³",
            "coordinates": {
                "latitude": -22.085518710314,
                "longitude": -70.188682515839
            },
            "country": "CL",
            "city": "Tocopilla"
        },
{
    "location": "Escuela E-10",
    "parameter": "pm25",
    "date": {
        "utc": "2019-03-17T21:00:00.000Z",
        "local": "2019-03-17T18:00:00-03:00"
    },
    "value": 4.43,
    "unit": "µg/m³",
    "coordinates": {
        "latitude": -22.085518710314,
        "longitude": -70.188682515839
    },
    "country": "CL",
    "city": "Tocopilla"
},
```

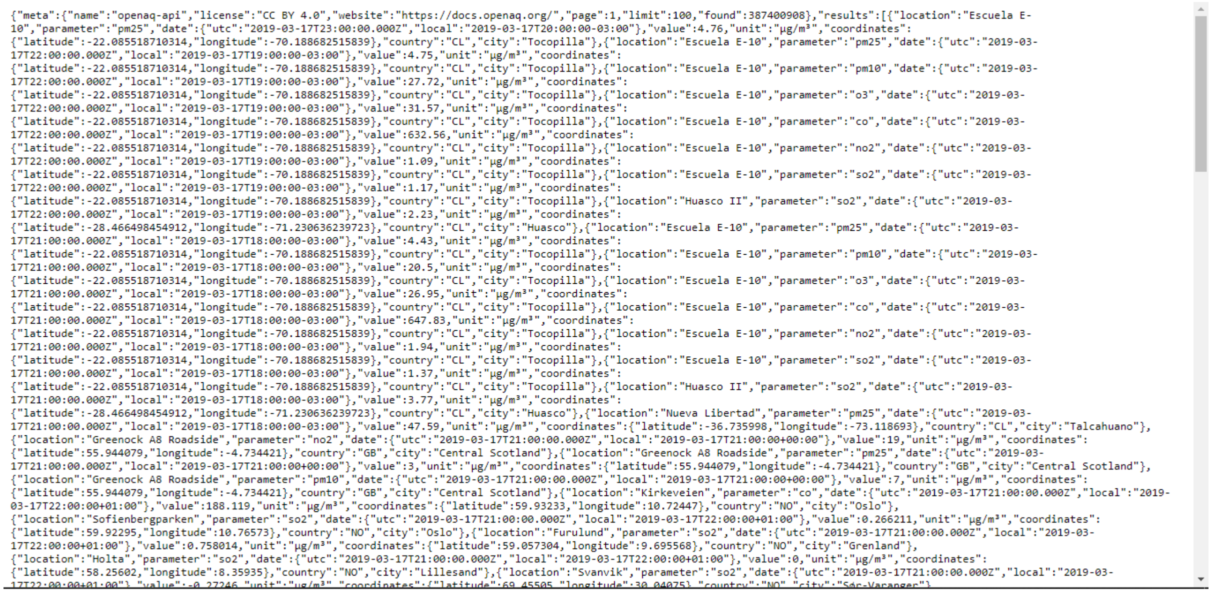
Not all the request will be show here as the response is too long for the document.

Browser Request Tutorial

In this tutorial, we will create a request to a OpenAQ API pollution feed using a new tab on a Google Chrome browser. For this tutorial, we will use the same endpoint as is used in the Postman example:

https://api.openaq.org/v1/measurements

Step 1: Add url to tab and run
This tutorial is easier than either the Postman method or the AJAX method (still to come) as it only involves using a tab in a browser as if you are navigating to a web page. To get a result, add the url to the address bar and hit enter and you should see the following:



```
{"meta":{"name":"openaq-api","license":"CC BY 4.0","website":"https://docs.openaq.org/","page":1,"limit":100,"found":387400908},"results":[{"location":"Escuela E-
10","parameter":"pm25","date":{"utc":"2019-03-17T23:00:00.000Z","local":"2019-03-17T20:00:00-03:00"},"value":4.76,"unit":"µg/m³","coordinates":
{"latitude":-22.085518710314,"longitude":-70.188682515839},"country":"CL","city":"Tocopilla"},{"location":"Escuela E-10","parameter":"pm25","date":{"utc":"2019-03-
17T22:00:00.000Z","local":"2019-03-17T19:00:00-03:00"},"value":4.75,"unit":"µg/m³","coordinates":
{"latitude":-22.085518710314,"longitude":-70.188682515839},"country":"CL","city":"Tocopilla"},{"location":"Escuela E-10","parameter":"pm10","date":{"utc":"2019-03-
17T22:00:00.000Z","local":"2019-03-17T19:00:00-03:00"},"value":27.72,"unit":"µg/m³","coordinates":
{"latitude":-22.085518710314,"longitude":-70.188682515839},"country":"CL","city":"Tocopilla"},{"location":"Escuela E-10","parameter":"o3","date":{"utc":"2019-03-
17T22:00:00.000Z","local":"2019-03-17T19:00:00-03:00"},"value":31.57,"unit":"µg/m³","coordinates":
{"latitude":-22.085518710314,"longitude":-70.188682515839},"country":"CL","city":"Tocopilla"},{"location":"Escuela E-10","parameter":"co","date":{"utc":"2019-03-
17T22:00:00.000Z","local":"2019-03-17T19:00:00-03:00"},"value":632.56,"unit":"µg/m³","coordinates":
{"latitude":-22.085518710314,"longitude":-70.188682515839},"country":"CL","city":"Tocopilla"},{"location":"Escuela E-10","parameter":"no2","date":{"utc":"2019-03-
17T22:00:00.000Z","local":"2019-03-17T19:00:00-03:00"},"value":1.09,"unit":"µg/m³","coordinates":
{"latitude":-22.085518710314,"longitude":-70.188682515839},"country":"CL","city":"Tocopilla"},{"location":"Escuela E-10","parameter":"so2","date":{"utc":"2019-03-
17T22:00:00.000Z","local":"2019-03-17T19:00:00-03:00"},"value":1.17,"unit":"µg/m³","coordinates":
{"latitude":-22.085518710314,"longitude":-70.188682515839},"country":"CL","city":"Tocopilla"},{"location":"Huasco II","parameter":"so2","date":{"utc":"2019-03-
17T22:00:00.000Z","local":"2019-03-17T19:00:00-03:00"},"value":2.23,"unit":"µg/m³","coordinates":
{"latitude":-28.466498454912,"longitude":-71.230636239723},"country":"CL","city":"Huasco"},{"location":"Escuela E-10","parameter":"pm25","date":{"utc":"2019-03-
17T21:00:00.000Z","local":"2019-03-17T18:00:00-03:00"},"value":4.43,"unit":"µg/m³","coordinates":
{"latitude":-22.085518710314,"longitude":-70.188682515839},"country":"CL","city":"Tocopilla"},{"location":"Escuela E-10","parameter":"pm10","date":{"utc":"2019-03-
17T21:00:00.000Z","local":"2019-03-17T18:00:00-03:00"},"value":20.5,"unit":"µg/m³","coordinates":
{"latitude":-22.085518710314,"longitude":-70.188682515839},"country":"CL","city":"Tocopilla"},{"location":"Escuela E-10","parameter":"o3","date":{"utc":"2019-03-
17T21:00:00.000Z","local":"2019-03-17T18:00:00-03:00"},"value":26.95,"unit":"µg/m³","coordinates":
{"latitude":-22.085518710314,"longitude":-70.188682515839},"country":"CL","city":"Tocopilla"},{"location":"Escuela E-10","parameter":"co","date":{"utc":"2019-03-
17T21:00:00.000Z","local":"2019-03-17T18:00:00-03:00"},"value":647.83,"unit":"µg/m³","coordinates":
{"latitude":-22.085518710314,"longitude":-70.188682515839},"country":"CL","city":"Tocopilla"},{"location":"Escuela E-10","parameter":"no2","date":{"utc":"2019-03-
17T21:00:00.000Z","local":"2019-03-17T18:00:00-03:00"},"value":1.94,"unit":"µg/m³","coordinates":
{"latitude":-22.085518710314,"longitude":-70.188682515839},"country":"CL","city":"Tocopilla"},{"location":"Escuela E-10","parameter":"so2","date":{"utc":"2019-03-
17T21:00:00.000Z","local":"2019-03-17T18:00:00-03:00"},"value":1.37,"unit":"µg/m³","coordinates":
{"latitude":-22.085518710314,"longitude":-70.188682515839},"country":"CL","city":"Tocopilla"},{"location":"Huasco II","parameter":"so2","date":{"utc":"2019-03-
17T21:00:00.000Z","local":"2019-03-17T18:00:00-03:00"},"value":3.77,"unit":"µg/m³","coordinates":
{"latitude":-28.466498454912,"longitude":-71.230636239723},"country":"CL","city":"Huasco"},{"location":"Nueva Libertad","parameter":"pm25","date":{"utc":"2019-03-
17T21:00:00.000Z","local":"2019-03-17T18:00:00-03:00"},"value":47.59,"unit":"µg/m³","coordinates":{"latitude":-36.735998,"longitude":-73.118693},"country":"CL","city":"Talcahuano"},
{"location":"Greenock A8 Roadside","parameter":"no2","date":{"utc":"2019-03-17T21:00:00.000Z","local":"2019-03-17T21:00:00+00:00"},"value":19,"unit":"µg/m³","coordinates":
{"latitude":55.944079,"longitude":-4.734421},"country":"GB","city":"Central Scotland"},{"location":"Greenock A8 Roadside","parameter":"pm25","date":{"utc":"2019-03-
17T21:00:00.000Z","local":"2019-03-17T21:00:00+00:00"},"value":3,"unit":"µg/m³","coordinates":{"latitude":55.944079,"longitude":-4.734421},"country":"GB","city":"Central Scotland"},
{"location":"Greenock A8 Roadside","parameter":"pm10","date":{"utc":"2019-03-17T21:00:00.000Z","local":"2019-03-17T21:00:00+00:00"},"value":7,"unit":"µg/m³","coordinates":
{"latitude":55.944079,"longitude":-4.734421},"country":"GB","city":"Central Scotland"},{"location":"Kirkeveien","parameter":"co","date":{"utc":"2019-03-17T21:00:00.000Z","local":"2019-
03-17T22:00:00+01:00"},"value":188.119,"unit":"µg/m³","coordinates":{"latitude":59.93233,"longitude":10.72447},"country":"NO","city":"Oslo"},
{"location":"Sofienbergparken","parameter":"so2","date":{"utc":"2019-03-17T21:00:00.000Z","local":"2019-03-17T22:00:00+01:00"},"value":0.266211,"unit":"µg/m³","coordinates":
{"latitude":59.92295,"longitude":10.76573},"country":"NO","city":"Oslo"},{"location":"Furulund","parameter":"so2","date":{"utc":"2019-03-17T21:00:00.000Z","local":"2019-03-
17T22:00:00+01:00"},"value":0.758014,"unit":"µg/m³","coordinates":{"latitude":59.057304,"longitude":9.695568},"country":"NO","city":"Grenland"},
{"location":"Holta","parameter":"so2","date":{"utc":"2019-03-17T21:00:00.000Z","local":"2019-03-17T22:00:00+01:00"},"value":0,"unit":"µg/m³","coordinates":
{"latitude":58.25602,"longitude":8.35935},"country":"NO","city":"Lillesand"},{"location":"Svanvik","parameter":"so2","date":{"utc":"2019-03-17T21:00:00.000Z","local":"2019-03-
17T22:00:00+01:00"},"value":-0.27246,"unit":"µg/m³","coordinates":{"latitude":69.45505,"longitude":30.04075},"country":"NO","city":"Sør-Varanger"},
```

The result is shown in plain text below:

{"meta":{"name":"openaq-api","license":"CC BY 4.0","website":"https://docs.openaq.org/","page":1,"limit":100,"found":387400908},"results":[{"location":"Escuela E-10","parameter":"pm25","date":{"utc":"2019-03-17T23:00:00.000Z","local":"2019-03-17T20:00:00-03:00"},"value":4.76,"unit":"µg/m³","coordinates":{"latitude":-22.085518710314,"longitude":-70.188682515839},"country":"CL","city":"Tocopilla"},{"location":"Escuela E-10","parameter":"pm25","date":{"utc":"2019-03-17T22:00:00.000Z","local":"2019-03-17T19:00:00-03:00"},"value":4.75,"unit":"µg/m³","coordinates":{"latitude":-22.085518710314,"longitude":-70.188682515839},"country":"CL","city":"Tocopilla"},{"location":"Escuela E-10","parameter":"pm10","date":{"utc":"2019-03-17T22:00:00.000Z","local":"2019-03-17T19:00:00-03:00"},"value":27.72,"unit":"µg/m³","coordinates":

Once again, the full response is not shown as it is too large.

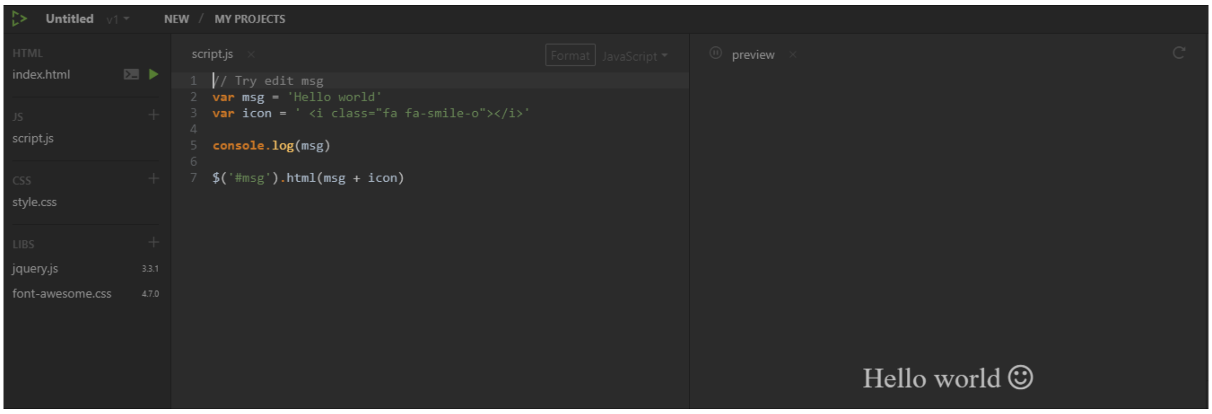
AJAX Tutorial

The above methods show you how to access the OpenAQ feed and view the data, however, neither method involves any coding and neither would work well in an application's context. So, in this tutorial we will connect to the monthly feed using AJAX which can then be put into the javascript of any application

Steps:

1. Create a new playcode project
2. Add the provided AJAX code and run

Step 1: Create a new playcode project
For this tutorial, playcode will be used to demonstrate the AJAX request. Although we are using playcode, almost any online JavaScript IDE would work (JsFiddle). If you search for playcode and click on the link you should see the following default project setup:



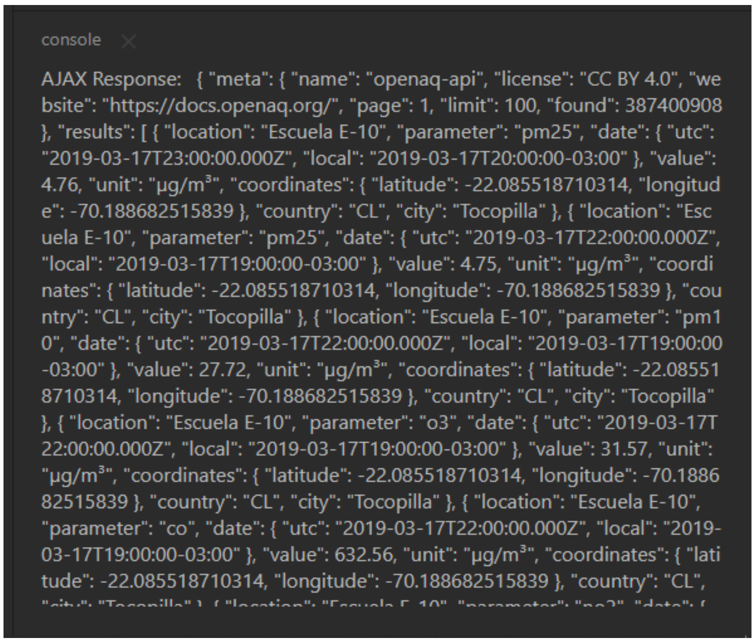
```
// Try edit msg
var msg = 'Hello world'
var icon = ' <i class="fa fa-smile-o"></i>'

console.log(msg)

$('#msg').html(msg + icon)
```

Playcode is ideal for this tutorial as we only need to change the js file and view the console as jquery is referenced as standard.

Step 2: Add the provided AJAX code and run
The AJAX required to fetch the data programatically is:

```
$.ajax({
        type: "GET",
        url: "https://api.openaq.org/v1/measurements",
        contentType: "application/json",
        success: function (result) {
            console.log('AJAX Response: ', result);
        },
        error: function (errorResult) {
            console.log('ERROR: ', errorResult);
        }
    });
```

If you replace the contents of script.js with the above AJAX and run the code, you should see the following:



```
AJAX Response:   { "meta": { "name": "openaq-api", "license": "CC BY 4.0", "we
bsite": "https://docs.openaq.org/", "page": 1, "limit": 100, "found": 387400908
}, "results": [ { "location": "Escuela E-10", "parameter": "pm25", "date": { "utc":
"2019-03-17T23:00:00.000Z", "local": "2019-03-17T20:00:00-03:00" }, "value":
4.76, "unit": "µg/m³", "coordinates": { "latitude": -22.085518710314, "longitud
e": -70.188682515839 }, "country": "CL", "city": "Tocopilla" }, { "location": "Esc
uela E-10", "parameter": "pm25", "date": { "utc": "2019-03-17T22:00:00.000Z",
"local": "2019-03-17T19:00:00-03:00" }, "value": 4.75, "unit": "µg/m³", "coordi
nates": { "latitude": -22.085518710314, "longitude": -70.188682515839 }, "cou
ntry": "CL", "city": "Tocopilla" }, { "location": "Escuela E-10", "parameter": "pm1
0", "date": { "utc": "2019-03-17T22:00:00.000Z", "local": "2019-03-17T19:00:00
-03:00" }, "value": 27.72, "unit": "µg/m³", "coordinates": { "latitude": -22.08551
8710314, "longitude": -70.188682515839 }, "country": "CL", "city": "Tocopilla"
}, { "location": "Escuela E-10", "parameter": "o3", "date": { "utc": "2019-03-17T
22:00:00.000Z", "local": "2019-03-17T19:00:00-03:00" }, "value": 31.57, "unit":
"µg/m³", "coordinates": { "latitude": -22.085518710314, "longitude": -70.1886
82515839 }, "country": "CL", "city": "Tocopilla" }, { "location": "Escuela E-10",
"parameter": "co", "date": { "utc": "2019-03-17T22:00:00.000Z", "local": "2019-
03-17T19:00:00-03:00" }, "value": 632.56, "unit": "µg/m³", "coordinates": { "lati
tude": -22.085518710314, "longitude": -70.188682515839 }, "country": "CL",
```

As you can see this result looks similar to the other results we have seen earlier in the project, however, this code can be added to an application and the data can be fetched and altered dynamically.

Note: If playcode hangs or feels unresponsive when running this code there is no need to worry, their is a lot of data to process from this feed.

Conclusion
That is the methods which can be used to request data from the OpenAQ feeds. The most useful of the methods is AJAX, however, the others can be used for testing and ensuring the quality of the data.

Common Issues:
Normally, implementing the methods discussed in this tutorial are relatively easy, however, some of the most common issues are detailed below:

- Lack of responsiveness - Depending on the feed chosen, the method used to fetch the data can seem slow because it has to process thousands of ages in the pollution sometimes. Normally, the request should complete quite quickly without too much impact on the user.

Outcomes
You should now be able to:
- Request data from one of the OpenAQ API pollution feeds in Postman
- Request data from one of the OpenAQ API pollution feeds in a new tab
- Request data from one of the OpenAQ API pollution feeds using AJAX call

References
Although we hope this tutorial has been all the help you need, here are some useful links that may be of use:

Useful Resources:

- OpenAQ API - https://openaq.org/
- Functional playcode AJAX request - https://playcode.io/268741?tabs=console&script.js&output

Tools:

- Postman - https://www.getpostman.com/downloads/
- Playcode - https://playcode.io/