

Lesson

Weekend

Intermediate JavaScript (/intermediate-javascript)

/ Asynchrony and APIs (/intermediate-javascript/asynchrony-and-apis)

/ API Documentation and Keys

Text

Just like we covered in the previous lesson, there are many different APIs we can incorporate into our applications. These range from APIs provided by Twitter and YouTube to APIs for the weather forecast, sports scores, and movie databases. For instance, if you wanted information about movies in your application, you could try the IMDB or Rotten Tomatoes API.

While the specifications for each API will be different, there's a couple things that most APIs will have in common: documentation and API keys. We'll cover both of these topics in this lesson, and use the OpenWeather API as an example.

API Keys

Some APIs are free while others are premium, though many premium APIs have free tiers. A good place to start is verifying if the API is accessible and at what cost.

In most cases, you'll need to create a developer account with the API you wish to use and then get an API key. An **API key** is like a password and should always be protected — otherwise, someone else can access (and use) your API account.

The APIs that we'll use during this section use token-based authentication. In our case, that's just a fancy way of saying that we need to attach our API key to every call we make to that server. The API server will check our token (the API key) and verify whether we can have access to the information we are asking for. This verification process is known as **authentication**. We take advantage of authentication all the time, like when we log into our email, Epicenter, or any other accounts we have online.

Some APIs use tokens that expire every 24 hours. That means we need to get a new token every day. Other sites use a more complex authentication process known as OAuth. OAuth is beyond the scope of what we cover, but it's good to know it exists. We'll focus on sites that provide an API key that doesn't expire.

One such site is OpenWeather (<https://openweathermap.org/>), which provides information about weather conditions around the world. Its API has both free and premium tiers. We'll use the free tier which will still allow us to make plenty of API calls.

Working with API Documentation

Many APIs have extensive documentation, but not always. When you're getting started with a new API, look for a guide, API page, or quick start page either in the navbar or on the splash page (the "introductory" page of a site). It's also good to briefly review the contents of the documentation to what it covers, and locate important information like pricing and how to get started.

For the OpenWeather API, we can find information about signing up for an account and getting an API key at the OpenWeather API guide (<https://openweathermap.org/guide>).

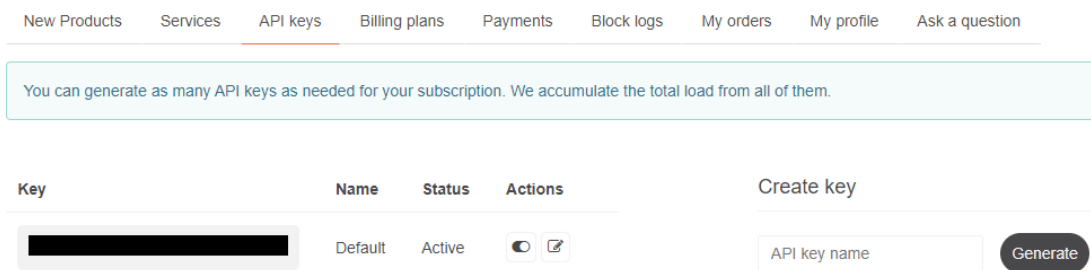
Getting Started with the OpenWeather API

To get an API key for OpenWeather, follow these steps:

1. Sign up for an account, and sign in.
2. Once you are signed in, click on the *API keys* tab.
3. Locate the API key called *Default*. You'll use this key in every request you make to the OpenWeather API.

Take note that you can generate a new key with a unique name instead of using *Default*. Generally, you'd only need multiple keys if you are using the API in multiple locations (such as on different sites) and you want to keep track of the API usage in each location.

Keep the information about your key handy since we'll be using it in the next lesson to make our first API call.



The screenshot shows the 'API keys' tab selected in a navigation bar. Below the navigation bar, a light blue message box states: 'You can generate as many API keys as needed for your subscription. We accumulate the total load from all of them.' Below this, there is a table with columns: 'Key', 'Name', 'Status', and 'Actions'. The first row shows a blacked-out key, the name 'Default', the status 'Active', and two icons (an eye and a copy icon). To the right of the table is a 'Create key' section with a text input field labeled 'API key name' and a 'Generate' button.

Key	Name	Status	Actions
[Redacted]	Default	Active	[Eye icon] [Copy icon]

Create key

API key name Generate

API Limits

Also, note the message just above our API key information:

You can generate as many API keys as needed for your subscription. We accumulate the total load from all of them.

The total load is the number of API calls we make. Almost all APIs limit the number of calls we make, especially for free tiers. We can usually see the limits by going to a page that lists different membership tiers (in this case, by clicking on the *Billing Plans* tab) or by looking more closely at the documentation.

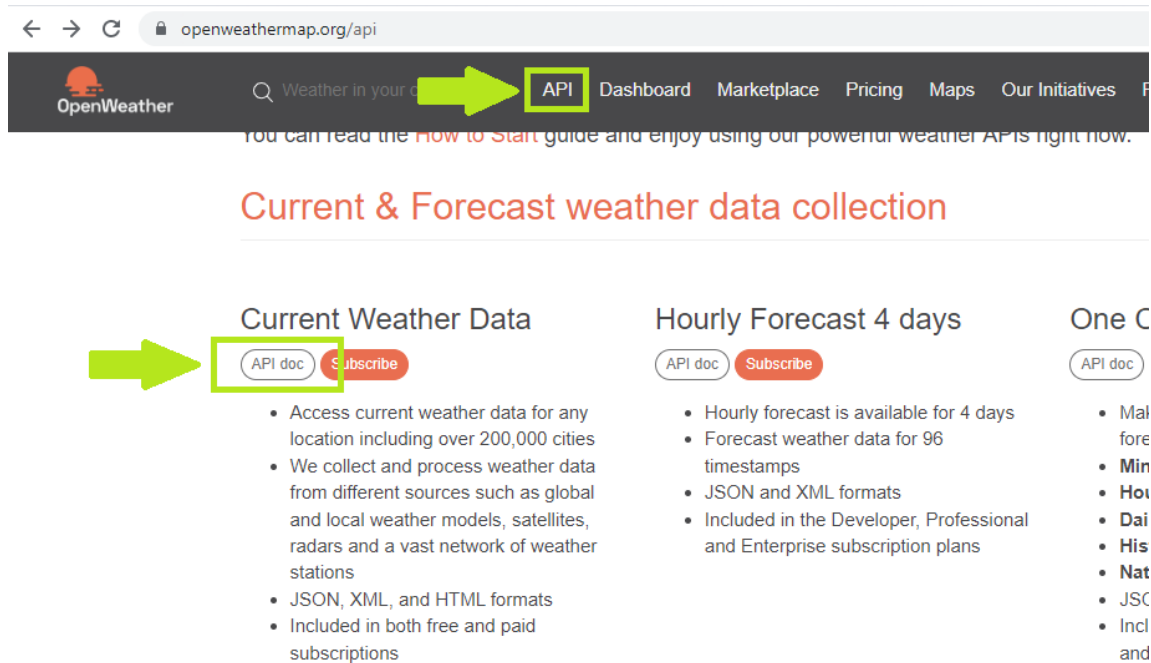
Limits on API calls can be monthly, weekly, daily, and even by the minute. In the case of the OpenWeather API, the free tier is one million calls per month — we don't have to worry about hitting that limit any time soon! However, other APIs you might work with may have introductory free plans that much more severely limit the number of API calls you can make, so you should always check the limits before you decide to interface with an API in your application.

Documentation on Endpoints

A well-documented API will have information about all of its **endpoints**. An endpoint is just a specific URL we can query for information. In the case of the OpenWeather API, there's a *lot* of information we can query. Working with APIs is just one more example of how important it is to get really used to working with documentation. In a future job, you may well have to dig deep into existing API documentation to find the exact API endpoint that you're looking for.

In our case, we want current weather conditions. This could be useful for a wide range of sites. Let's say, for instance, that we're working on a site that displays local parks. Wouldn't it be nice if a user could see the local forecast for that park on the same page?

We can click on the *API* tab to get information about all the endpoints OpenWeather provides, or navigate to this webpage (<https://openweathermap.org/api>). If you scroll down on this page, you'll find a section called *Current & Forecast weather data collection*. In this section, we'll find an API called *Current Weather Data*, which is exactly what we need. You'll also see a button called *API Docs*. Go ahead and click that and it will take you to the documentation on the Current Weather Data API (<https://openweathermap.org/current>).



In the documentation for the Current Weather Data API, we'll see that we can search the current weather by a range of different search parameters including city name, longitude and latitude, or zip code. We'll go with city name for now.

Well documented APIs provide example URLs and OpenWeather is no different. We'll use the option that also includes a state code so OpenWeather knows we mean Portland, Oregon, not another Portland. Here's the example URL:

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

We can update this to include the specific information we want:

```
api.openweathermap.org/data/2.5/weather?q=portland,oregon&appid=[YOUR-API-KEY]
```

Note that your API key needs to go in place of [YOUR-API-KEY] . You should not add any syntax around your API key such as square or curly brackets. Brackets are included above for emphasis only.

Making an API Call

Next, take that URL and enter it in the browser search bar. We'll see the following:



The screenshot shows a web browser window with the address bar containing the URL: `api.openweathermap.org/data/2.5/weather?q=portland,oregon&appid=c7174ca7b461d...`. The browser's developer tools or console displays a JSON object representing weather data for Portland, Oregon. The JSON includes coordinates, weather conditions (few clouds), temperature, wind speed, and other meteorological details.

```
{
  "coord": {
    "lon": -122.6762,
    "lat": 45.5234
  },
  "weather": [
    {
      "id": 801,
      "main": "Clouds",
      "description": "few clouds",
      "icon": "02d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 305.58,
    "feels_like": 305.34,
    "temp_min": 303.4,
    "temp_max": 308.08,
    "pressure": 1017,
    "humidity": 36,
    "visibility": 10000,
    "wind": {
      "speed": 5.66,
      "deg": 290
    },
    "clouds": {
      "all": 20
    },
    "dt": 1658357571,
    "sys": {
      "type": 2,
      "id": 2008548,
      "country": "US",
      "sunrise": 1658320878,
      "sunset": 1658375547,
      "timezone": -25200,
      "id": 5746545,
      "name": "Portland",
      "cod": 200
    }
  }
}
```

As we can see, we can actually make our API call right in the browser! The response we receive is in JSON (JavaScript Object Notation) (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON), a format we are already familiar with. But why is the format in JSON if our API call itself has nothing to do with JavaScript?

Well, the whole point of an API is to allow different applications to communicate with each other even if the applications use completely different programming languages. For that reason, many APIs use a more universal format to communicate with each other. JSON is the most common while XML is another less common format. Because XML is less common, we won't be covering it in this course, but if you do end up working with XML data, there is plenty of documentation online to help you.

Most languages, including Ruby, C#, and others, have methods to parse JSON. So while JSON is written in the same syntax as JavaScript objects, it's not really JavaScript. It's just written in that notation. The great news about this is that we are already pretty familiar with the notation that most APIs use!

Vetting an API

During the classwork for this section, you'll have an opportunity to pick your own APIs to work with. As we've outlined in this lesson, you'll need to do a little research to determine whether it's the right API to work with. When you find an API that you want to work with, make sure to answer the following questions before proceeding:

- Is the API well-documented? If not, **that's a bad sign**. It may be hard to work with as a result — and poor documentation can be a sign of a poorly-built API.
- Is there a free tier? If so, how many API calls can you make on this tier? Is it enough for your project?
- Does the API provide the data you need for your project? You might find yourself building an application around what the API *does* offer — but you may also want to build an application that needs specific data. In that case, make sure that the API's endpoints provide the data you need.
- What kind of authentication does the API use? An API key (token-based authentication) is best for now. Steer clear of OAuth unless you want a challenge.
- Does the API allow CORS? We'll discuss CORS in a future lesson (<https://www.learnhowtoprogram.com/intermediate-javascript/asynchrony-and-apis/sop-and-cors>). If it does allow CORS, you're good to go. If it doesn't, you should use a different API.

Summary

In this lesson, we've walked through the process of setting up a developer account and getting an API key from OpenWeather. In the process, we've also talked about some general points for reading through API documentation. Remember to always take the time to review an API's documentation before working with it to verify that it meets your needs.

At this point, we're ready to start trying out API calls in Postman!

[Previous \(/intermediate-javascript/asynchrony-and-apis/introduction-to-apis\)](#)

[Next \(/intermediate-javascript/asynchrony-and-apis/testing-api-calls-with-postman\)](#)

Lesson 4 of 33

Last updated more than 3 months ago.

[disable dark mode](#)



© 2023 Epicodus (<http://www.epicodus.com/>), Inc.