Introduction to APIs and JSONs

APIs

- Application Programming Interface
- Protocols and routines
    - Building and interacting with software applications

OMDb API

The Open Movie Database

JSONs

- JavaScript Object Notation
- Real-time server-to-browser communication
- Douglas Crockford
- Human readable

JSONs

```
{'Actors': 'Samuel L. Jackson, Julianna Margulies, Nathan
Phillips, Rachel Blanchard',
 'Awards': '3 wins & 7 nominations.',
 'Country': 'Germany, USA, Canada',
 'Director': 'David R. Ellis',
 'Genre': 'Action, Adventure, Crime',
 'Language': 'English',
 'Rated': 'R',
 'Released': '18 Aug 2006',
 'Runtime': '105 min',
 'Title': 'Snakes on a Plane',
 'Type': 'movie',
 'Writer': 'John Heffernan (screenplay), Sebastian Gutierrez
(screenplay), David Dalessandro (story), John Heffernan (story)',
 'Year': '2006',
 'imdbID': 'tt0417148',
 'imdbRating': '5.6',
 'imdbVotes': '114,668'}
```

Loading JSONs in Python

```
In [1]: import json

In [2]: with open('snakes.json', 'r') as json_file:
   ...:     json_data = json.load(json_file)

In [3]: type(json_data)
Out[3]: dict
```

Exploring JSONs in Python

```
In [4]: for key, value in json_data.items():
   ...:     print(key + ':', value)

Title: Snakes on a Plane
Country: Germany, USA, Canada
Response: True
Language: English
Awards: 3 wins & 7 nominations.
Year: 2006
Actors: Samuel L. Jackson, Julianna Margulies
Runtime: 105 min
Genre: Action, Adventure, Crime
imdbID: tt0417148
Director: David R. Ellis
imdbRating: 5.6
Rated: R
Released: 18 Aug 2006
```

Let’s practice!

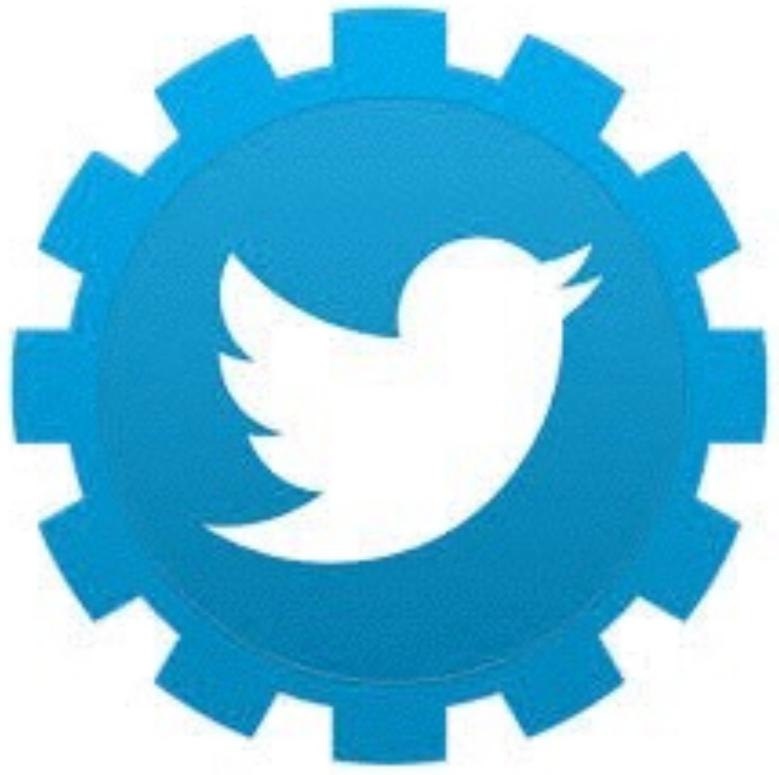
APIs and interacting with the world wide web

Herein, you'll learn

- What APIs are
- Why APIs are important
- In the exercises:
    - Connecting to APIs
    - Pulling data from APIs
    - Parsing data from APIs

What is an API?

- Set of protocols and routines
- Bunch of code
    - Allows two software programs to communicate with each other



APIs are everywhere

UBER Developers

Instagram API

Connecting to an API in Python

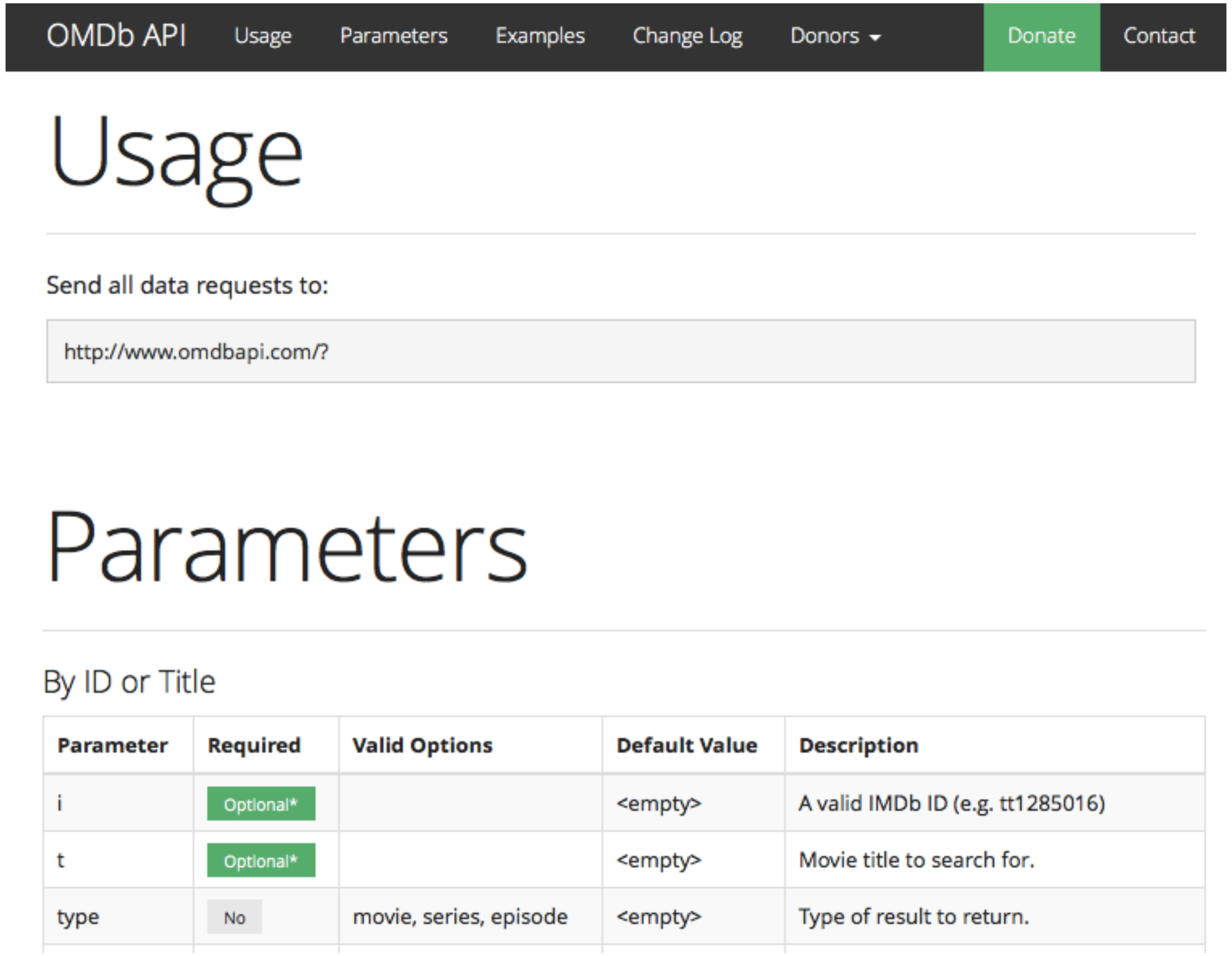
```
In [1]: import requests

In [2]: url = 'http://www.omdbapi.com/?t=hackers'

In [3]: r = requests.get(url)

In [4]: json_data = r.json()

In [5]: for key, value in json_data.items():
   ...:     print(key + ':', value)
```

What was that URL?

- http - making an HTTP request
- www.omdbapi.com - querying the OMDB API
- ?t=hackers
    - Query string
    - Return data for a movie with title (t) 'Hackers'

```
'http://www.omdbapi.com/?t=hackers'
```

OMDb API



Usage

Send all data requests to:

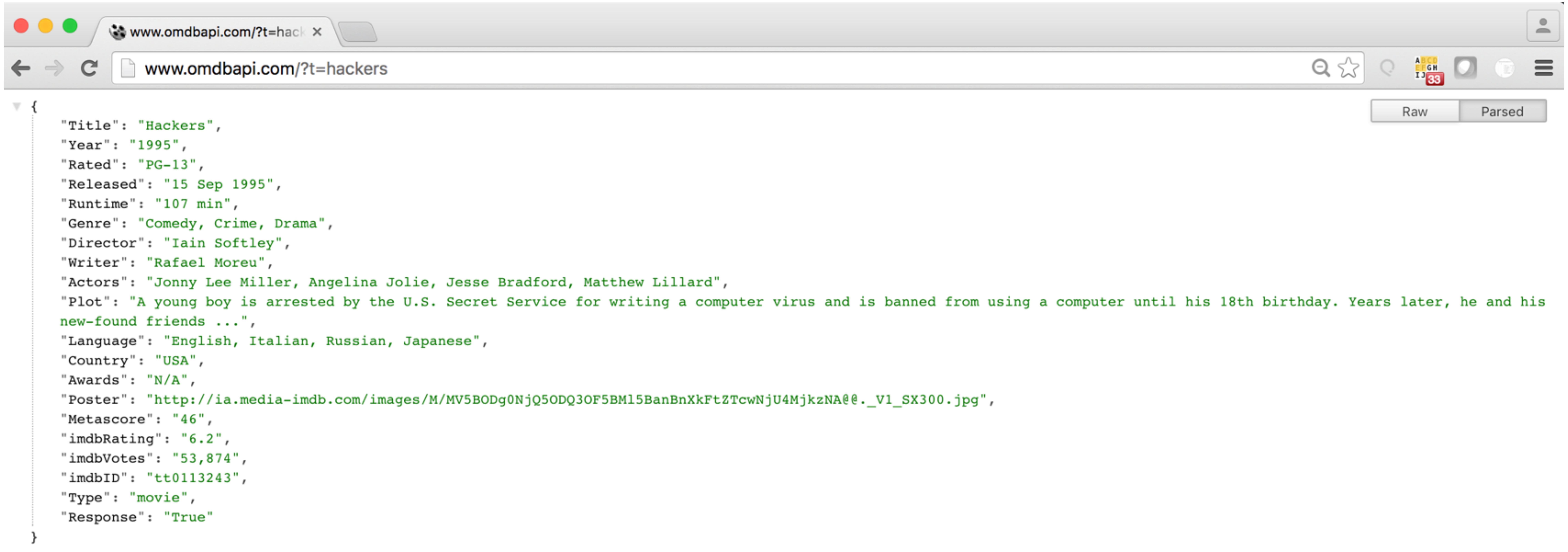
```
http://www.omdbapi.com/?
```

Parameters

By ID or Title

Parameter	Required	Valid Options	Default Value	Description
i	Optional*		<empty>	A valid IMDb ID (e.g. tt1285016)
t	Optional*		<empty>	Movie title to search for.
type	No	movie, series, episode	<empty>	Type of result to return.

It’s a regular URL!



```
{
    "Title": "Hackers",
    "Year": "1995",
    "Rated": "PG-13",
    "Released": "15 Sep 1995",
    "Runtime": "107 min",
    "Genre": "Comedy, Crime, Drama",
    "Director": "Iain Softley",
    "Writer": "Rafael Moreu",
    "Actors": "Jonny Lee Miller, Angelina Jolie, Jesse Bradford, Matthew Lillard",
    "Plot": "A young boy is arrested by the U.S. Secret Service for writing a computer virus and is banned from using a computer until his 18th birthday. Years later, he and his
    new-found friends ...",
    "Language": "English, Italian, Russian, Japanese",
    "Country": "USA",
    "Awards": "N/A",
    "Poster": "http://ia.media-imdb.com/images/M/MV5BODg0NjQ5ODQ3OF5BMl5BanBnXkFtZTcwNjU4MjkzNA@@._V1_SX300.jpg",
    "Metascore": "46",
    "imdbRating": "6.2",
    "imdbVotes": "53,874",
    "imdbID": "tt0113243",
    "Type": "movie",
    "Response": "True"
}
```

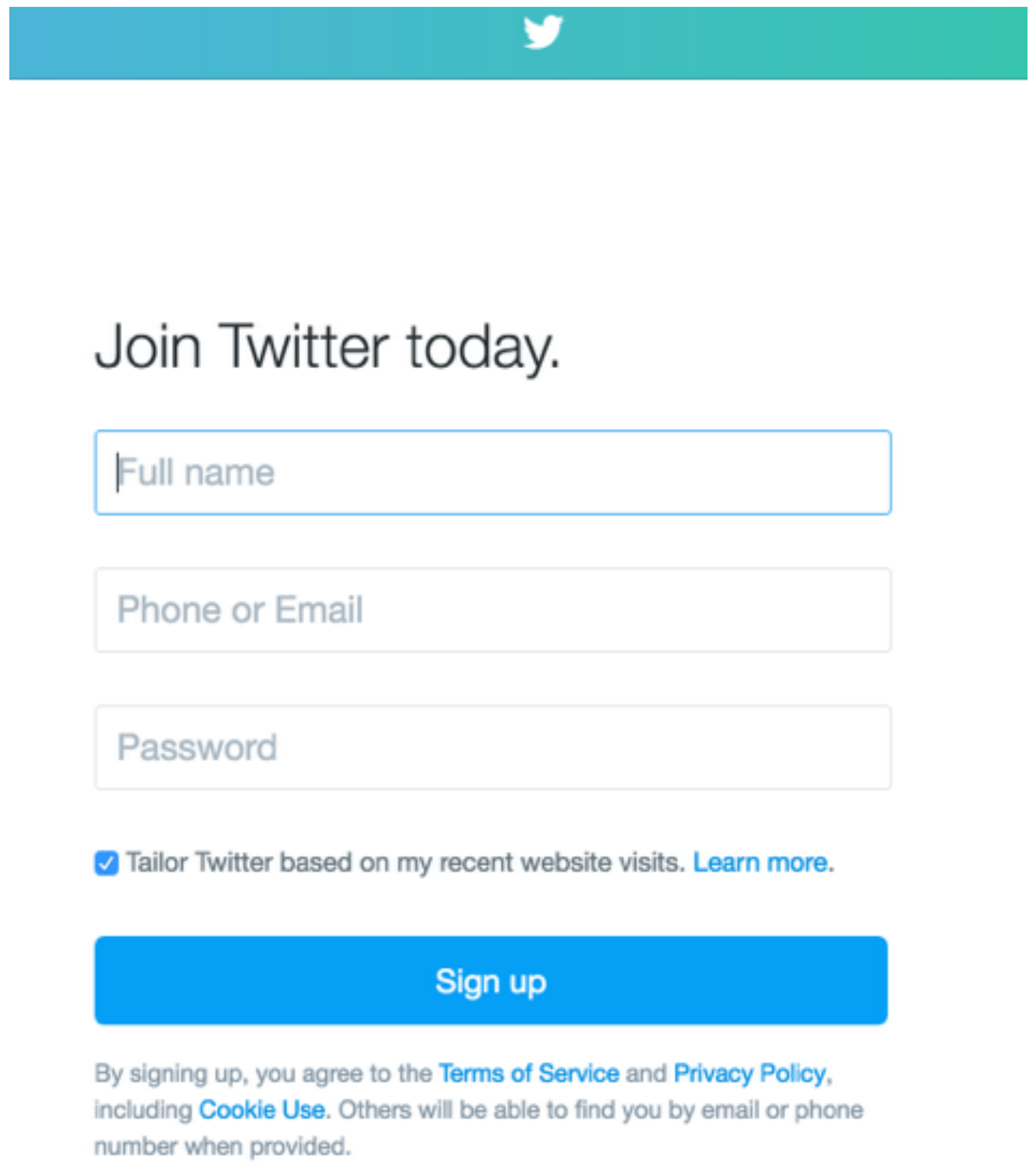
Let’s practice!

The Twitter API and Authentication

Herein, you'll learn

- How to stream data from the Twitter API
- How to filter incoming tweets for keywords
- About API Authentication and OAuth
- How to use the Tweepy Python package

Access the Twitter API



Join Twitter today.

Full name

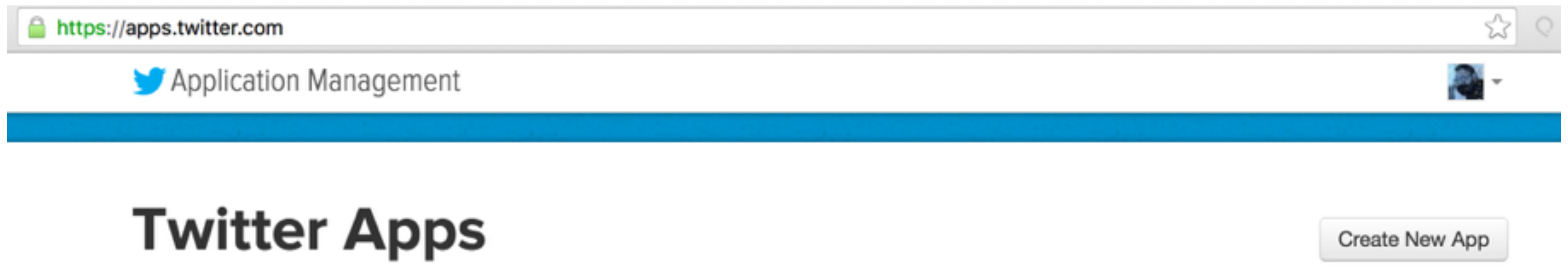
Phone or Email

Password

Tailor Twitter based on my recent website visits. Learn more.

Sign up

By signing up, you agree to the Terms of Service and Privacy Policy, including Cookie Use. Others will be able to find you by email or phone number when provided.



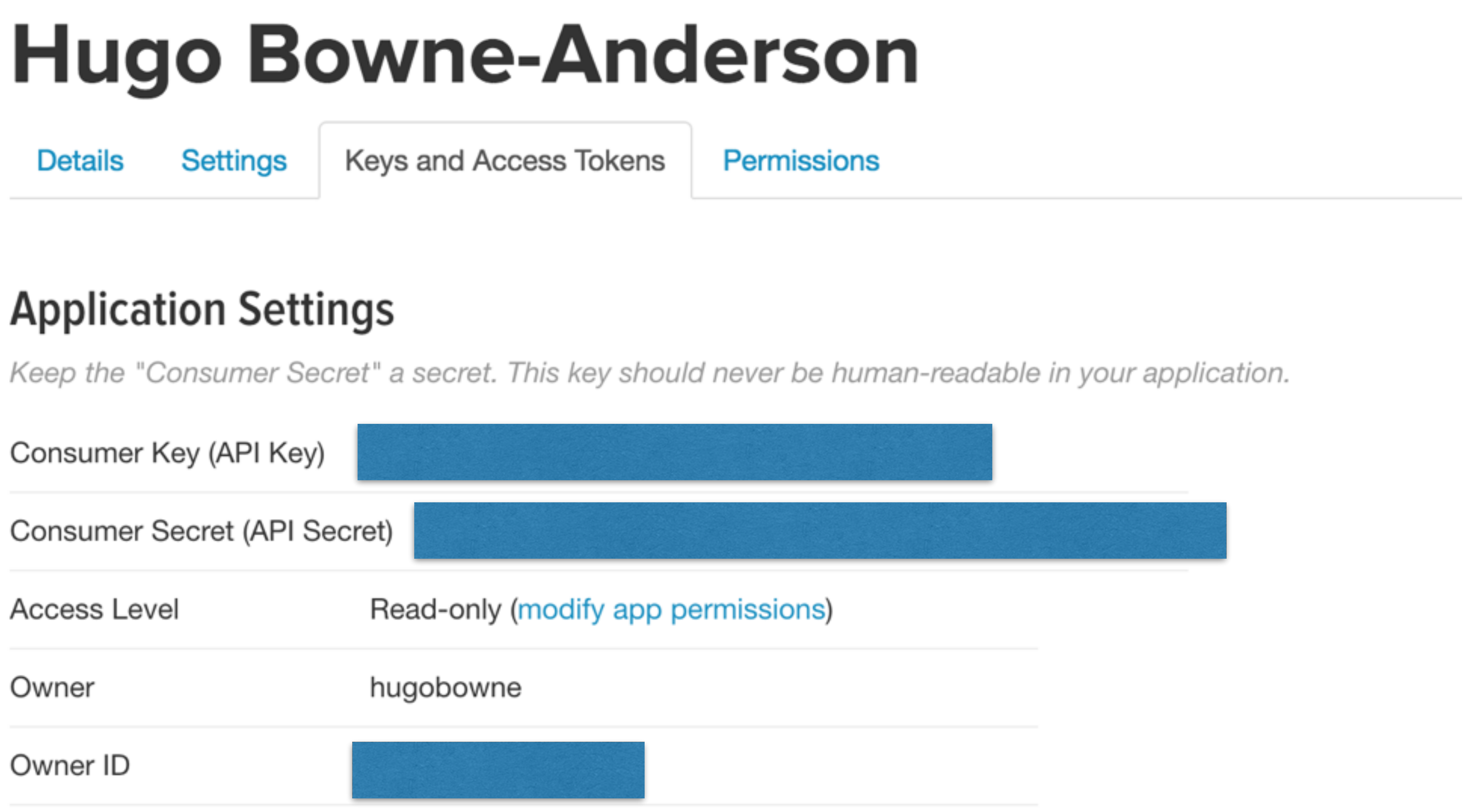
https://apps.twitter.com

Application Management

Twitter Apps

Create New App

Access the Twitter API



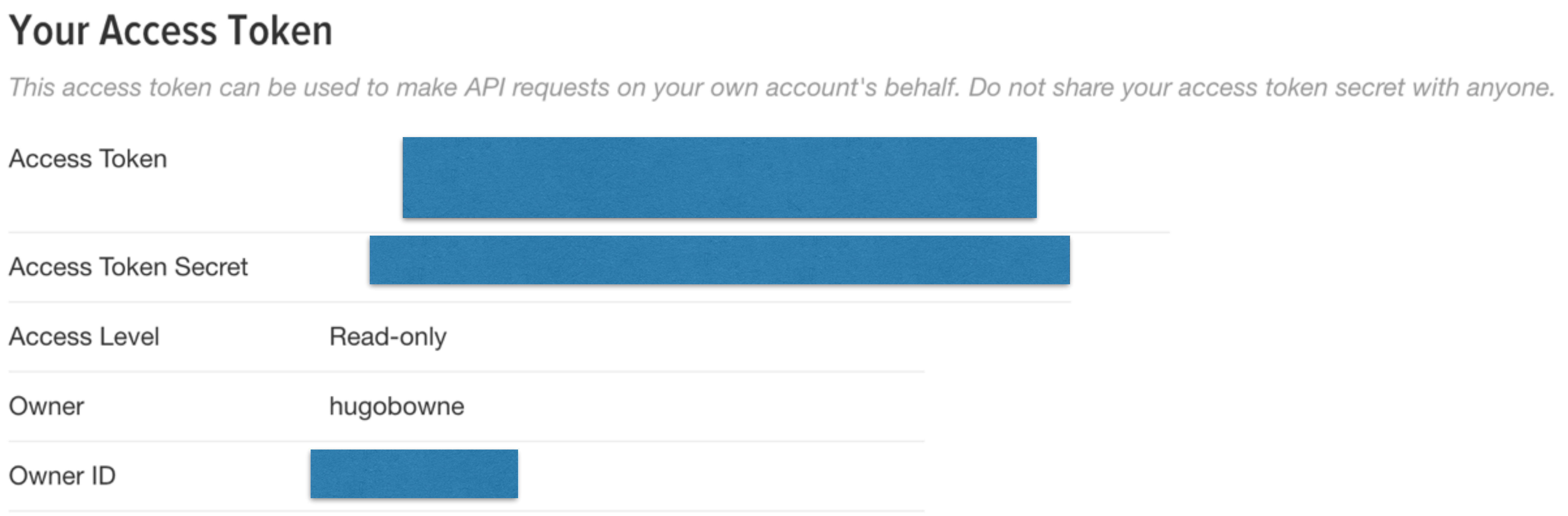
Hugo Bowne-Anderson

Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	[redacted]
Consumer Secret (API Secret)	[redacted]
Access Level	Read-only (modify app permissions)
Owner	hugobowne
Owner ID	[redacted]

Access the Twitter API

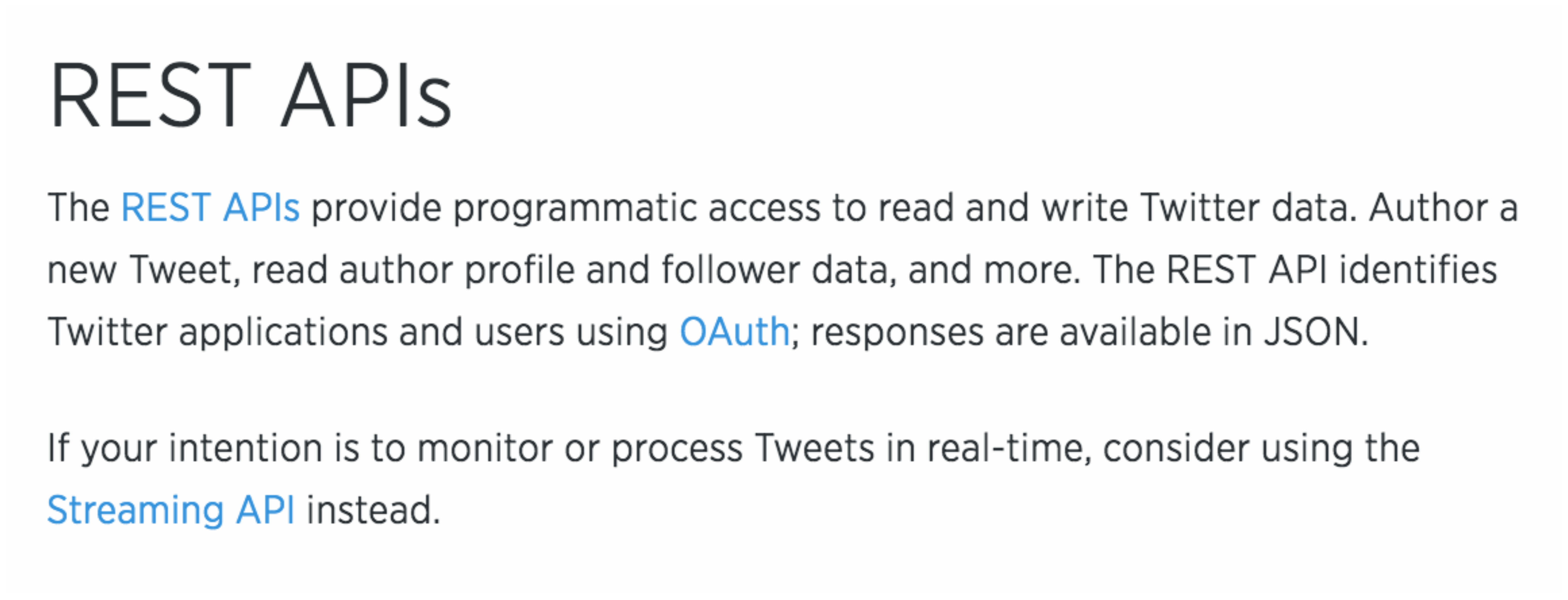


Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access token secret with anyone.

Access Token	[redacted]
Access Token Secret	[redacted]
Access Level	Read-only
Owner	hugobowne
Owner ID	[redacted]

Twitter has a number of APIs

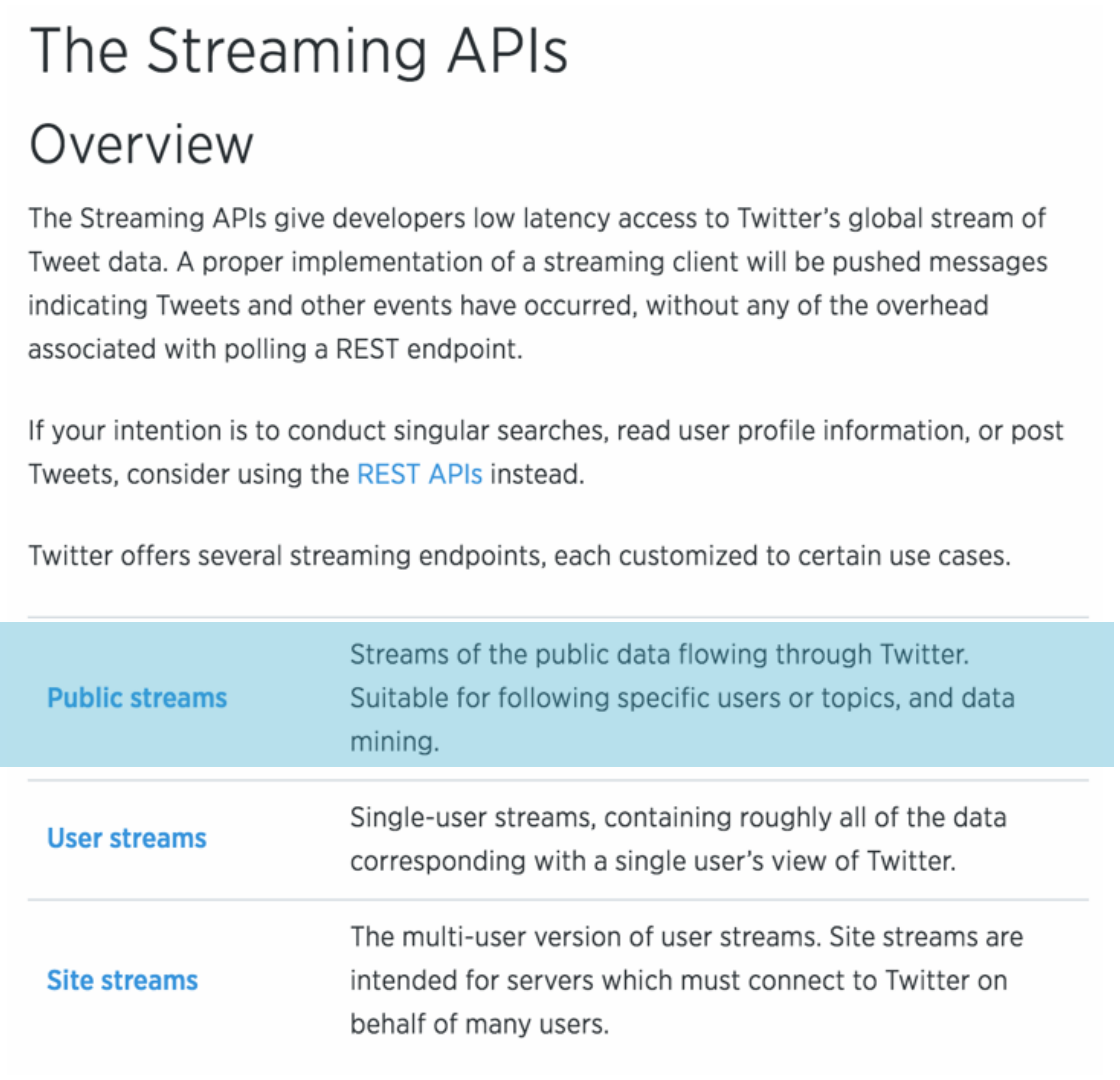


REST APIs

The REST APIs provide programmatic access to read and write Twitter data. Author a new Tweet, read author profile and follower data, and more. The REST API identifies Twitter applications and users using OAuth; responses are available in JSON.

If your intention is to monitor or process Tweets in real-time, consider using the Streaming API instead.

Twitter has a number of APIs



The Streaming APIs

Overview

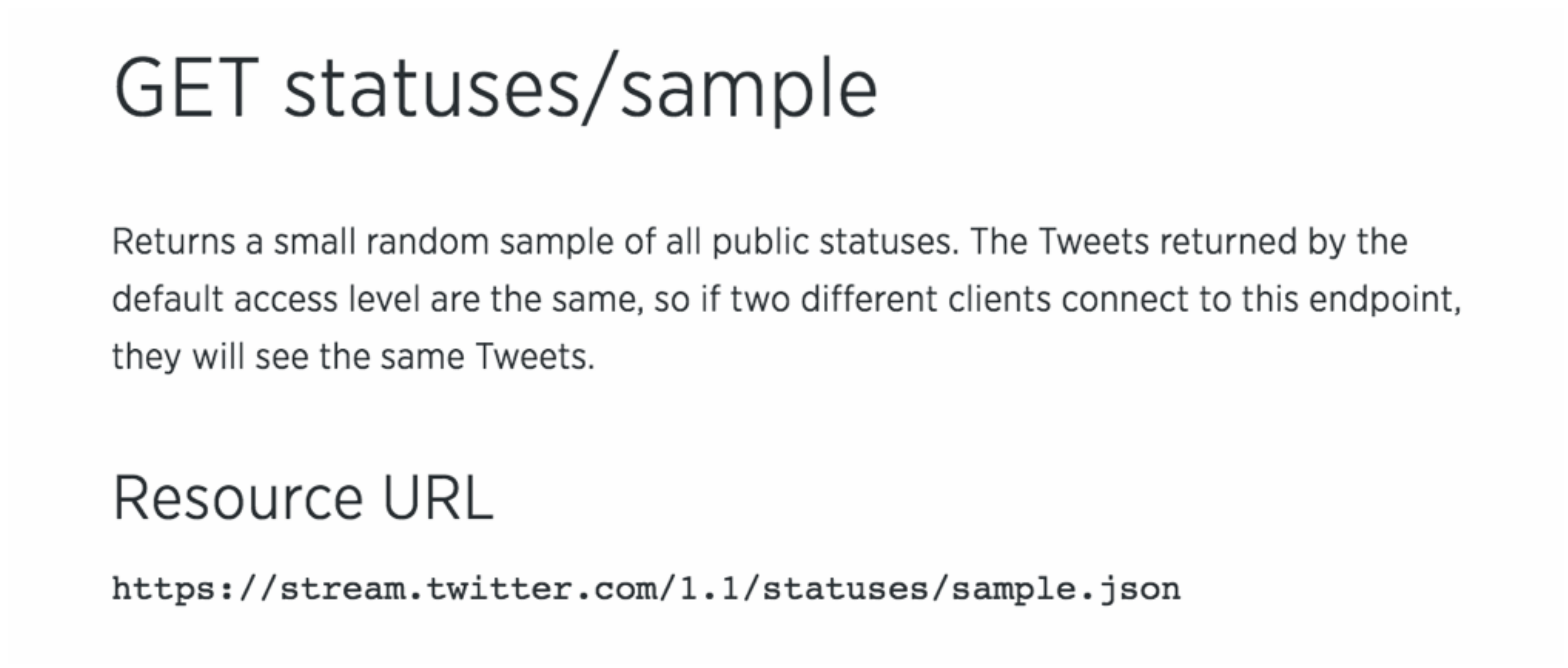
The Streaming APIs give developers low latency access to Twitter's global stream of Tweet data. A proper implementation of a streaming client will be pushed messages indicating Tweets and other events have occurred, without any of the overhead associated with polling a REST endpoint.

If your intention is to conduct singular searches, read user profile information, or post Tweets, consider using the REST APIs instead.

Twitter offers several streaming endpoints, each customized to certain use cases.

Public streams	Streams of the public data flowing through Twitter. Suitable for following specific users or topics, and data mining.
User streams	Single-user streams, containing roughly all of the data corresponding with a single user's view of Twitter.
Site streams	The multi-user version of user streams. Site streams are intended for servers which must connect to Twitter on behalf of many users.

Twitter has a number of APIs



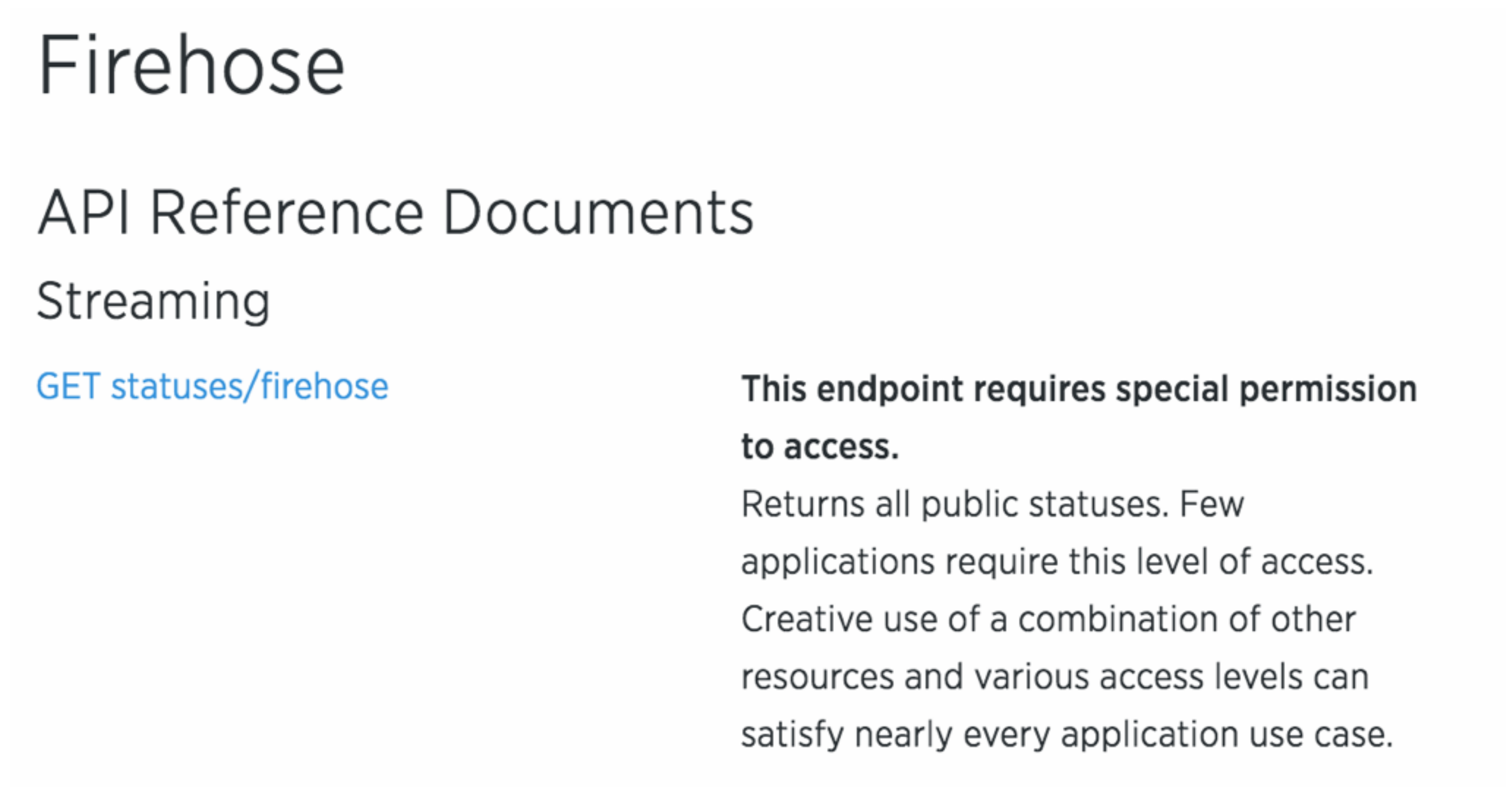
GET statuses/sample

Returns a small random sample of all public statuses. The Tweets returned by the default access level are the same, so if two different clients connect to this endpoint, they will see the same Tweets.

Resource URL

```
https://stream.twitter.com/1.1/statuses/sample.json
```

Twitter has a number of APIs


Firehose

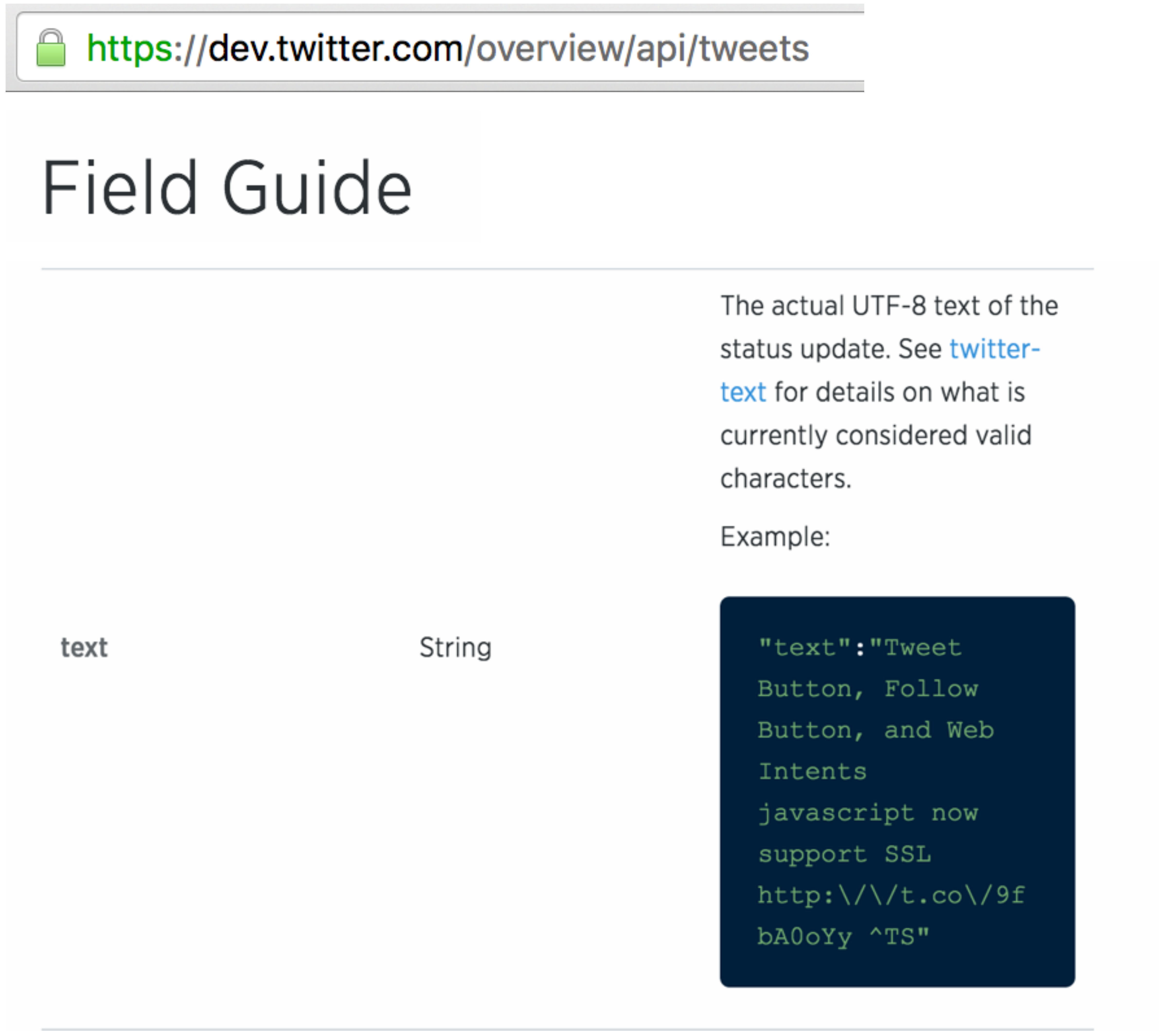
API Reference Documents

Streaming

GET statuses/firehose

This endpoint requires special permission to access.
Returns all public statuses. Few applications require this level of access. Creative use of a combination of other resources and various access levels can satisfy nearly every application use case.

Tweets are returned as JSONs

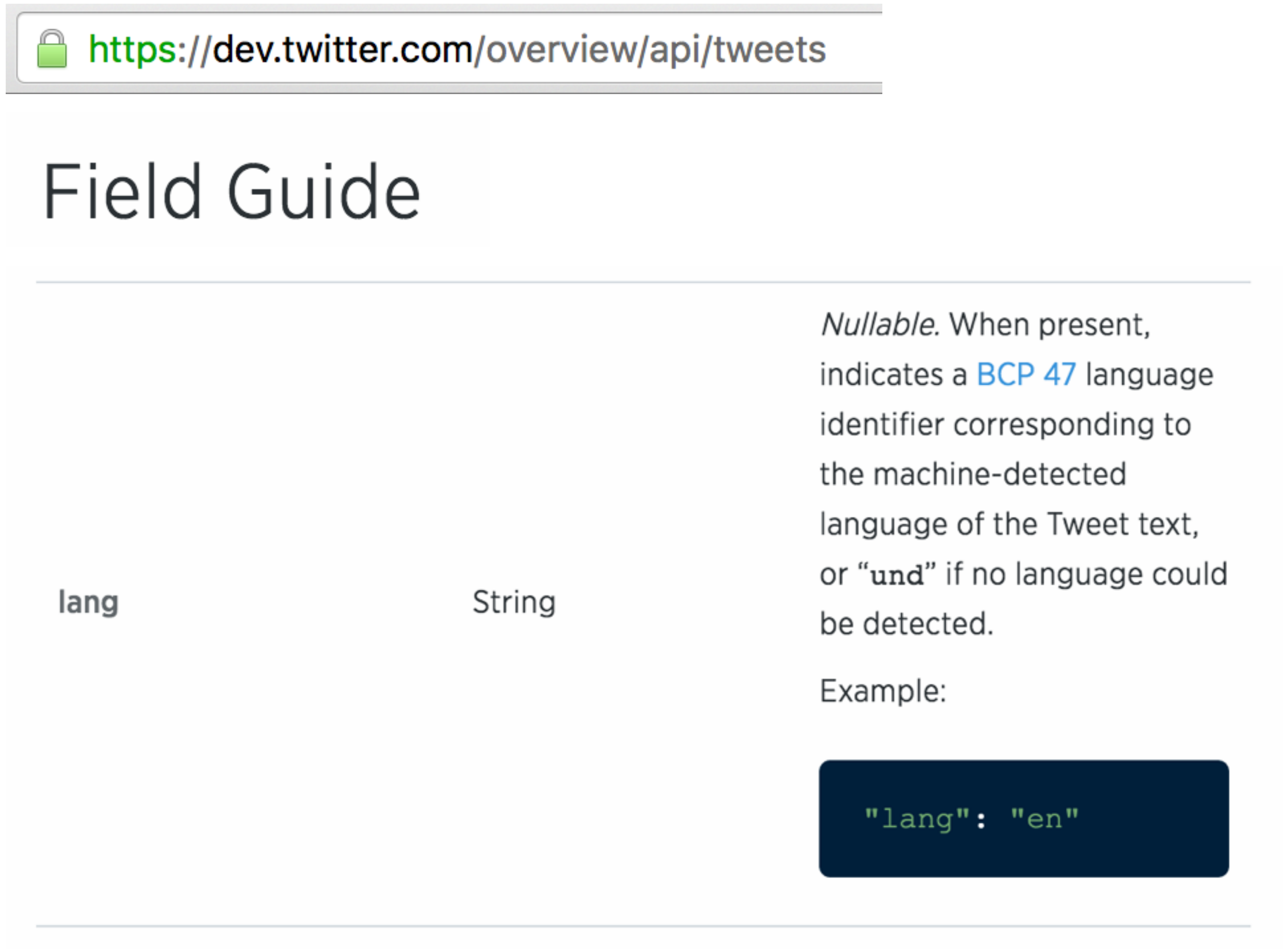


https://dev.twitter.com/overview/api/tweets

Field Guide

text	String	The actual UTF-8 text of the status update. See twitter-text for details on what is currently considered valid characters. Example: `"text":"Tweet Button, Follow Button, and Web Intents javascript now support SSL http:\/\/t.co\/9fbA0oYy ^TS"`

Tweets are returned as JSONs



https://dev.twitter.com/overview/api/tweets

Field Guide

lang	String	*Nullable.* When present, indicates a BCP 47 language identifier corresponding to the machine-detected language of the Tweet text, or "**und**" if no language could be detected. Example: `"lang": "en"`

Using Tweepy: Authentication handler

tw_auth.py

```
import tweepy, json

access_token = "..."
access_token_secret = "..."
consumer_key = "..."
consumer_secret = "..."

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
```

Tweepy: define stream listener class

st_class.py

```
class MyStreamListener(tweepy.StreamListener):
    def __init__(self, api=None):
        super(MyStreamListener, self).__init__()
        self.num_tweets = 0
        self.file = open("tweets.txt", "w")

    def on_status(self, status):
        tweet = status._json
        self.file.write(json.dumps(tweet) + '\n')
        tweet_list.append(status)
        self.num_tweets += 1
        if self.num_tweets < 100:
            return True
        else:
            return False
        self.file.close()
```

Using Tweepy: stream tweets!!

tweets.py

```
# Create Streaming object and authenticate
l = MyStreamListener()
stream = tweepy.Stream(auth, l)

# This line filters Twitter Streams to capture data by keywords:
stream.filter(track=['apples', 'oranges'])
```

Let’s practice!

Final Thoughts

What you've learned:

- Importing text files and flat files
- Importing files in other formats
- Writing SQL queries
- Getting data from relational databases
- Pulling data from the web
- Pulling data from APIs

Congratulations!