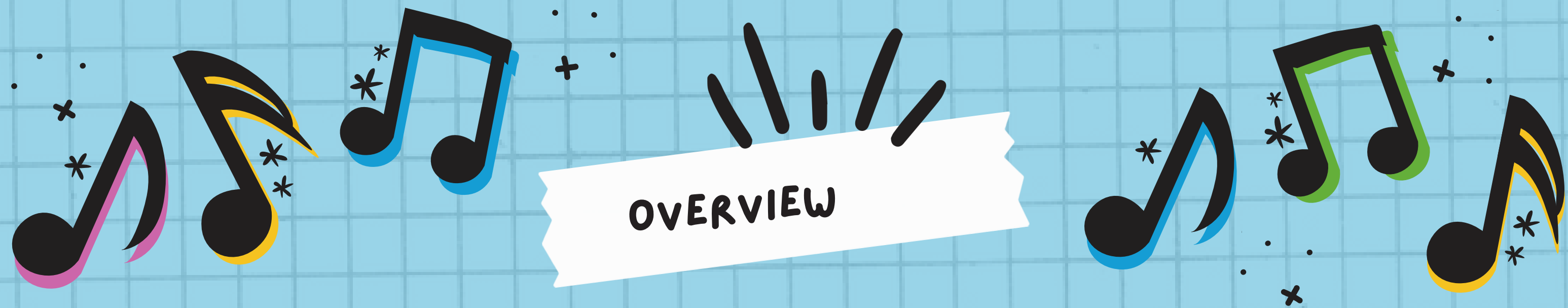




LIGHTNING TALK

# SPOTIFY API: OVERVIEW AND USE CASE

ALYSSA WARNICK-HESSE



A BRIEF INTRO TO SPOTIFY'S API DOCS AND SPOTIPY

FUN AND ALSO USEFUL! A SMALL REAL-LIFE EXAMPLE

# API OVERVIEW

DEVELOPER.SPOTIFY.COM

- Extremely thorough setup guides

DISCOVER

DOCS

CONSOLE

COMMUNITY

DASHBOARD

USE CASES

QUICK START

GUIDES

LIBRARIES

REFERENCE

## Web API Tutorial

Create a simple server-side application that accesses user related data through the Spotify Web API.

 **Note:** By using the Spotify Tools, you accept our [Developer Terms of Service](#).

Through the [Spotify Web API](#), external applications retrieve Spotify content such as album data and playlists. To access *user-related data* through the Web API, an application must be authorized by the user to access that particular information.

In this tutorial we create a simple application using Node.js and JavaScript and demonstrate how to:

- Register an application with Spotify
- Authenticate a user and get authorization to access user data
- Retrieve the data from a Web API endpoint

The authorization flow we use in this tutorial is the [Authorization Code Flow](#). This flow first gets a code from the Spotify Accounts Service, then exchanges that code for an access token. The code-to-token exchange requires a secret key, and for security is done through direct server-to-server communication.

In this example we retrieve data from the Web API `/me` endpoint, that includes information about the current user.

The complete source code of the app that will create in this tutorial is available on [GitHub](#).

## Set Up Your Account

To use the Web API, start by creating a Spotify user account (Premium or Free). To do that, simply sign up at [www.spotify.com](https://www.spotify.com).

# API OVERVIEW

```
Last login: Mon Oct 26 14:30:05 on ttys001
hessealy@Alyssas-MacBook-Pro ~ % cd src/Spotify\ Proj
hessealy@Alyssas-MacBook-Pro Spotify Proj % cd web-api-auth-examples
hessealy@Alyssas-MacBook-Pro web-api-auth-examples % cd authorization_code
hessealy@Alyssas-MacBook-Pro authorization_code % node app.js
Listening on 8888
(node:67256) [DEP0005] DeprecationWarning: Buffer() is deprecated due to security and usability issues. Please use
(Use 'node --trace-deprecation ...' to show where the warning was created)
{
  country: 'US',
  display_name: 'hessealy',
  email: 'hessealy@gmail.com',
  explicit_content: { filter_enabled: false, filter_locked: false },
  external_urls: { spotify: 'https://open.spotify.com/user/hessealy' },
  followers: { href: null, total: 0 },
  href: 'https://api.spotify.com/v1/users/hessealy',
  id: 'hessealy',
  images: [],
  product: 'premium',
  type: 'user',
  uri: 'spotify:user:hessealy'
}
```

Download authorization  
codes/files and run app.js  
on your local host

- Once logged in, you'll  
see your unique tokens

## Logged in as hessealy

Display name	hessealy
Id	hessealy
Email	hessealy@gmail.com
Spotify URI	<a href="https://open.spotify.com/user/hessealy">https://open.spotify.com/user/hessealy</a>
Link	<a href="https://api.spotify.com/v1/users/hessealy">https://api.spotify.com/v1/users/hessealy</a>
Profile Image	
Country	US

## oAuth info

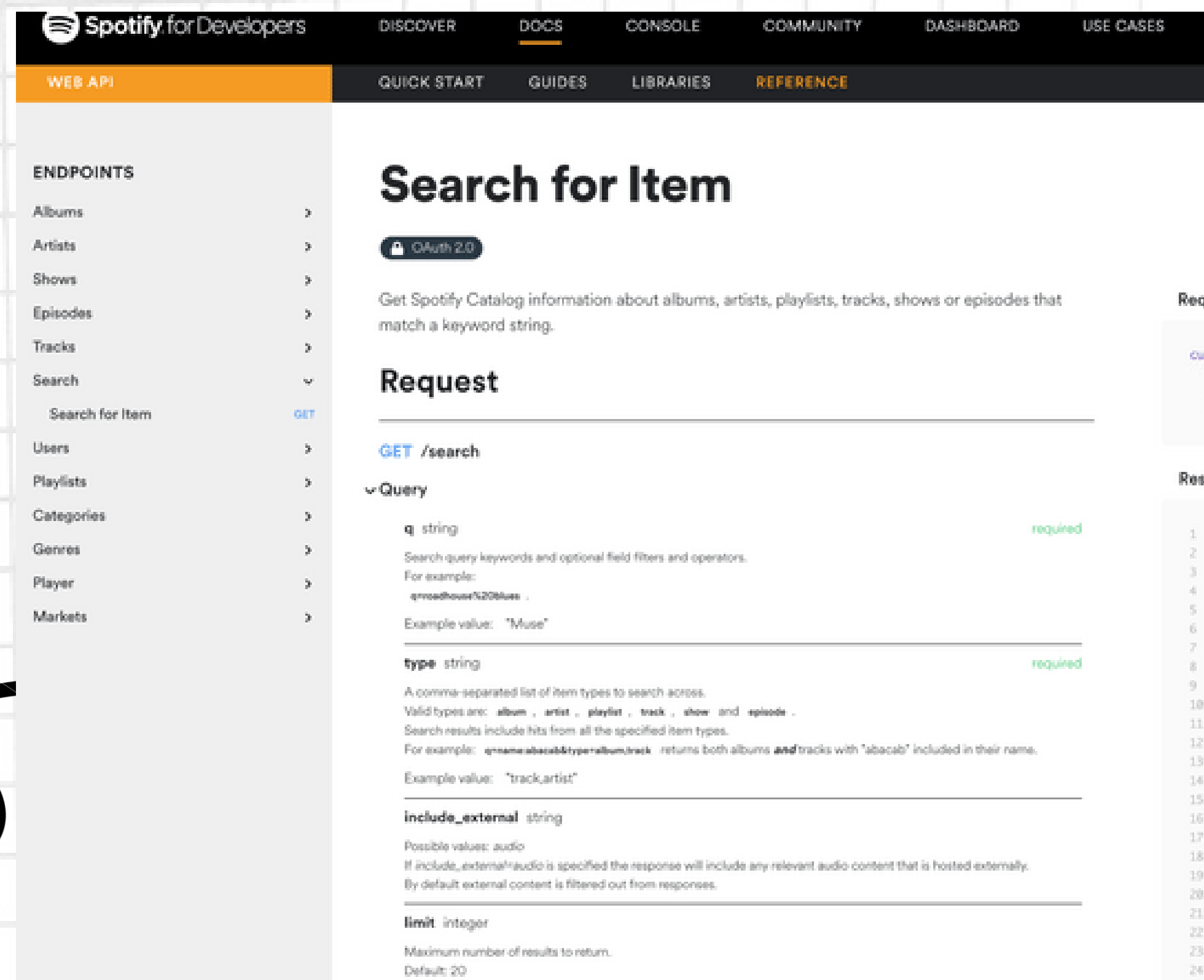
Access token	BODoeZRBH-veb54quVQYvli18DqkVXMSf
Refresh token	AQC10wTBEqa-iITsXDRmrdTL19Xu6fvsOI

Obtain new token using the refresh token

# API OVERVIEW

EASY TO BROWSE ENDPOINTS  
AND REQUESTS

Also easy to get lost in  
the huge JSON data  
structures!



The screenshot shows the Spotify for Developers Web API documentation. The top navigation bar includes links for DISCOVER, DOCS, CONSOLE, COMMUNITY, DASHBOARD, and USE CASES. The left sidebar lists various endpoints: Albums, Artists, Shows, Episodes, Tracks, Search (expanded), Users, Playlists, Categories, Genres, Player, and Markets. The 'Search for Item' endpoint is selected, showing a GET request. The main content area details the request parameters: 'q' (string, required) for search query keywords, 'type' (string, required) for item types, 'include\_external' (string) for audio content, and 'limit' (integer) for the number of results. The page is annotated with a green banner at the top left, a wavy line and a circle on the left side, and a large 'X' mark on the bottom right.

Spotify for Developers

DISCOVER DOCS CONSOLE COMMUNITY DASHBOARD USE CASES

WEB API QUICK START GUIDES LIBRARIES REFERENCE

## Search for Item

OAuth 2.0

Get Spotify Catalog information about albums, artists, playlists, tracks, shows or episodes that match a keyword string.

### Request

**GET** /search

Query

**q** string required

Search query keywords and optional field filters and operators.  
For example:  
`q=roadhouse%20blue`  
Example value: "Muse"

**type** string required

A comma-separated list of item types to search across.  
Valid types are: `album`, `artist`, `playlist`, `track`, `show` and `episode`.  
Search results include hits from all the specified item types.  
For example: `q=name:abacab&type=album,track` returns both albums **and** tracks with "abacab" included in their name.  
Example value: "track,artist"

**include\_external** string

Possible values: `audio`  
If `include_external=audio` is specified the response will include any relevant audio content that is hosted externally.  
By default external content is filtered out from responses.

**limit** integer

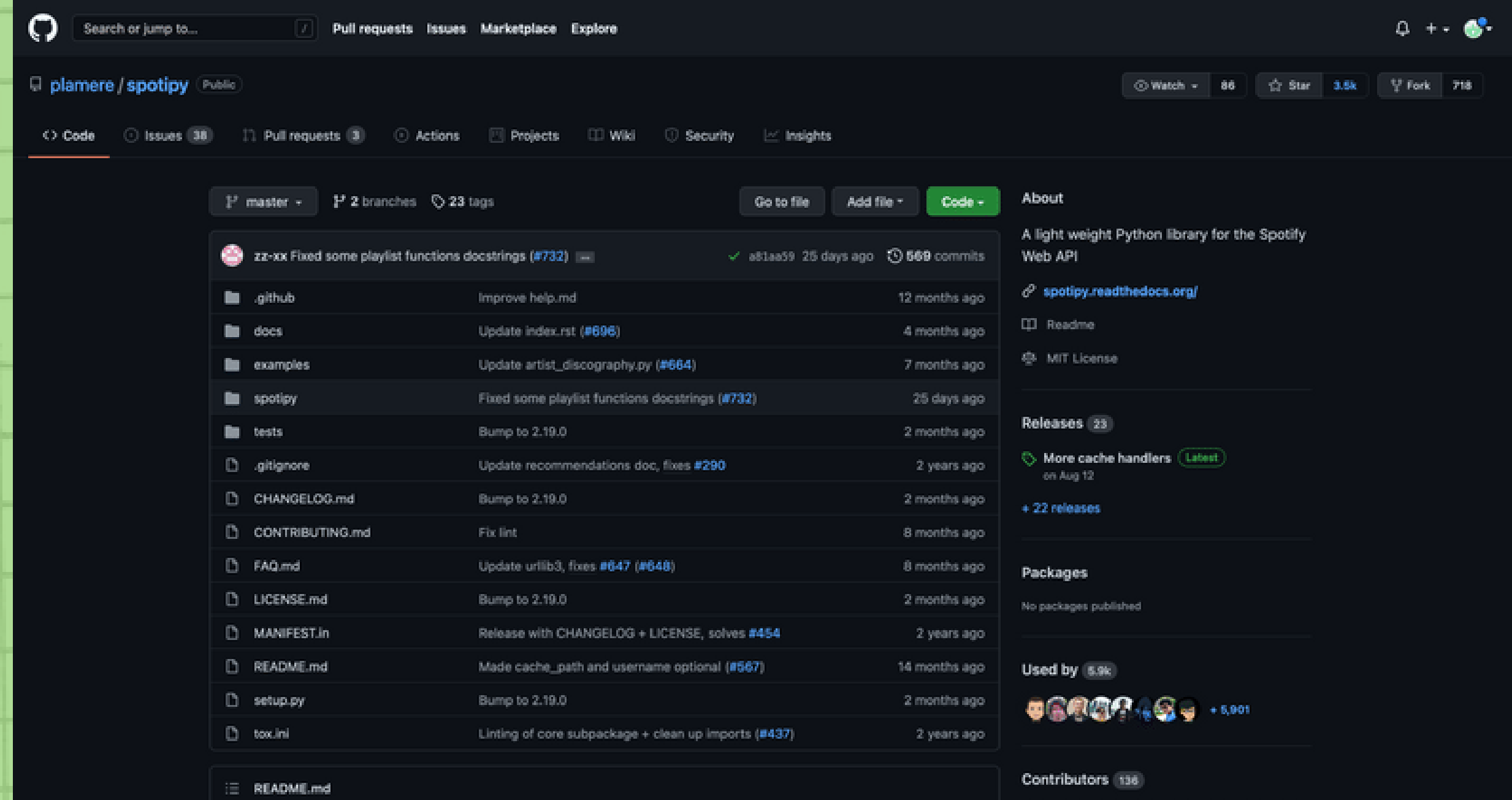
Maximum number of results to return.  
Default: 20



*SPOTIFY!!*



**A HUGE LIBRARY FOR  
YOUR HUGE DATA**

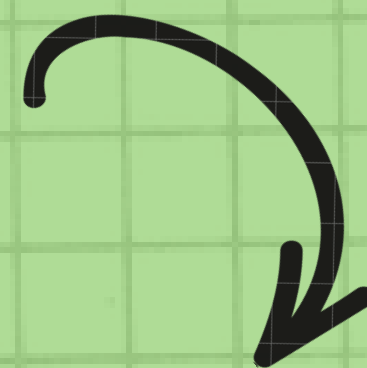
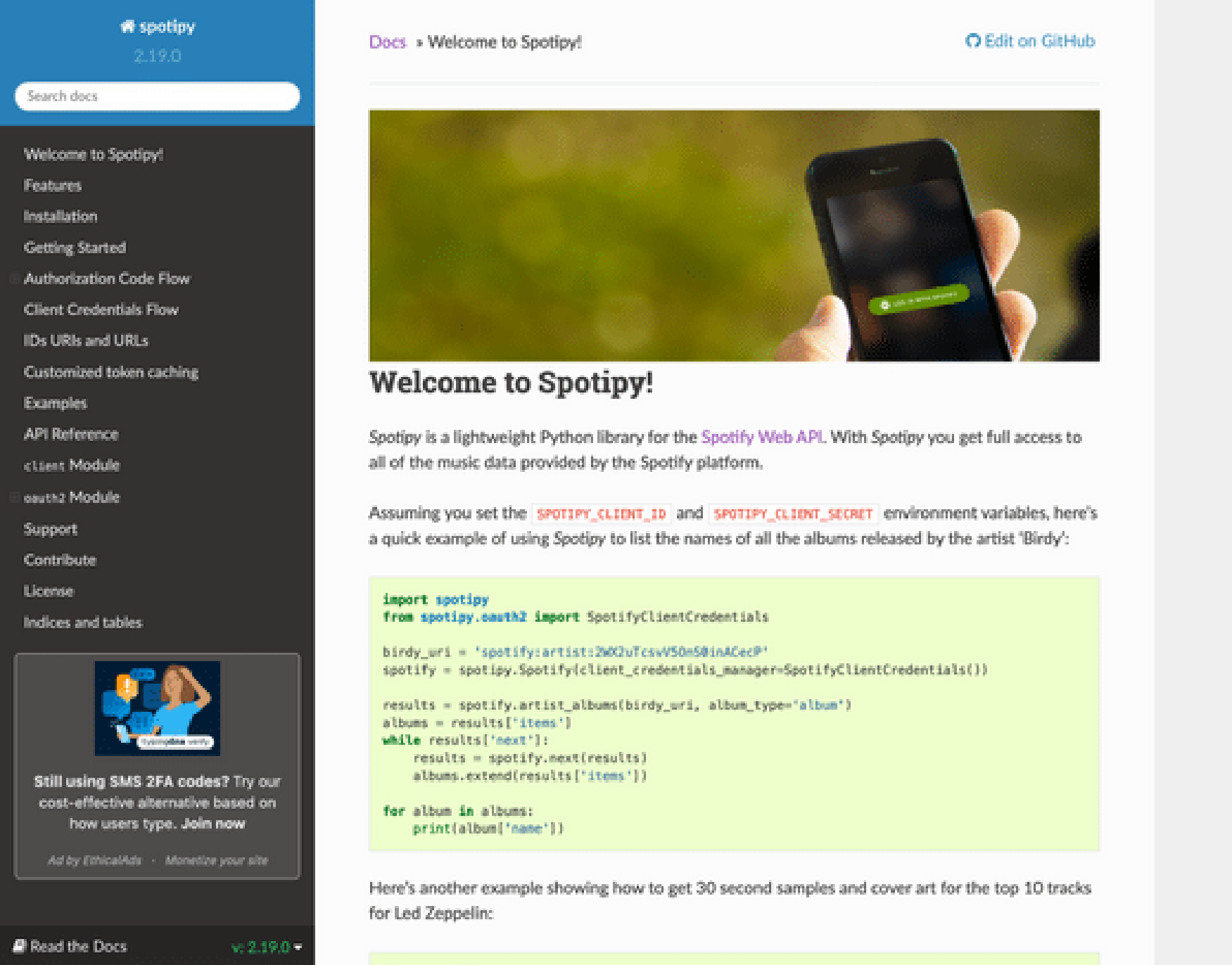


GITHUB.COM/PLAMERE/SPOTIPY

Tons of examples and walk-throughs

"client.py" has MANY pre-built functions to explore





SPOTIPY!!



spotipy.readthedocs.io/

Even more docs!

Examples and a little more reader-friendly



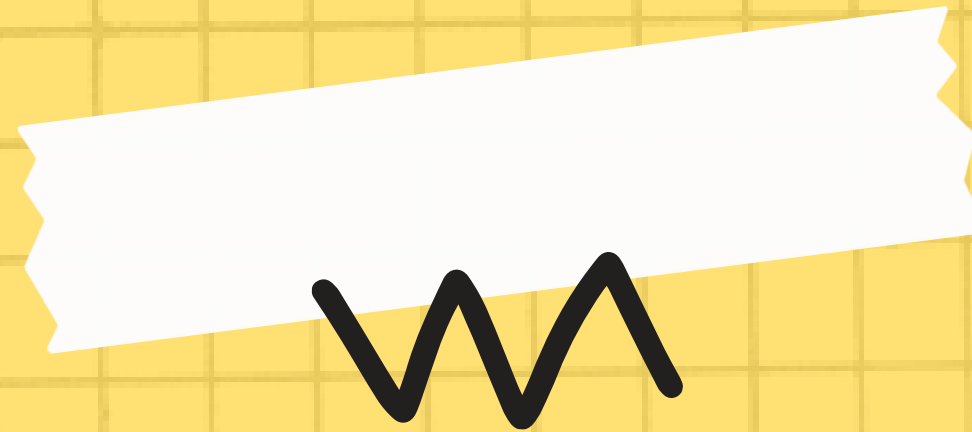


**CLIENT.PY**  
(OUR  
EXAMPLE)

```
537
538     def search(self, q, limit=10, offset=0, type="track", market=None):
539         """ searches for an item
540
541         Parameters:
542         - q - the search query (see how to write a query in the
543             official documentation https://developer.spotify.com/documentation/web-api/reference/search/) # noqa
544         - limit - the number of items to return (min = 1, default = 10, max = 50). The limit is applied
545             within each type, not on the total response.
546         - offset - the index of the first item to return
547         - type - the types of items to return. One or more of 'artist', 'album',
548             'track', 'playlist', 'show', and 'episode'. If multiple types are desired,
549             pass in a comma separated string; e.g., 'track,album,episode'.
550         - market - An ISO 3166-1 alpha-2 country code or the string
551             from_token.
552
553         """
554         return self._get(
555             "search", q=q, limit=limit, offset=offset, type=type, market=market
556         )
```



# USE CASE: SPEECH THERAPY IN K-12 SCHOOLS



1.

SCHOOL-BASED SLPS WORK WITH MANY STUDENTS WITH COMPLEX COMMUNICATION NEEDS

2.

ONE COMMON TOOL IS TO USE A "WORD OF THE DAY/WEEK" IN SONGS OR VIDEOS TO ENCOURAGE PRACTICE

3.

BUT IT'S A CHALLENGE TO FIND SONGS THAT ARE FUN, CHILD-APPROPRIATE, AND USE THE SAME WORD MANY TIMES

4.

API AND SPOTIFY CAN HELP!



# SEARCH FOR SONGS WITH A SPECIFIC WORD IN TITLE!



1.

YOU CAN SAMPLE IN  
THE API CONSOLE:

2.

BUT IT RETURNS A HUGE  
AMOUNT OF DATA

3.

AND THERE'S NO EASY  
WAY TO CROSS-  
REFERENCE DATA FROM  
ANOTHER ENDPOINT

The screenshot shows the Spotify API Console interface. On the left is a sidebar with a 'CONSOLE' menu containing links to Albums, Artists, Browse, Episodes, Follow, Library, Markets, Personalization, Player, Playlists, Search, Tracks, Shows, and Users Profile. The main panel is titled 'Search for Item' and contains a table with API details:

API Reference	Search for Item
Endpoint	https://api.spotify.com/v1/search
HTTP Method	GET
OAuth	Required

Below the table are input fields for query parameters:

- q \***: name:happy
- type \***: track
- market**: ES
- limit**: 10
- offset**: 5
- include\_external**: (empty)

At the bottom, there is a cURL command in a dark box:

```
curl -X "GET" "https://api.spotify.com/v1/search?q=name%3Ahappy&type=track" -H "Accept: application/json" -H "Content-Type: application/json" -H "Authorization: Bearer BQ0uWuIdNA390sn58k4nVcr4Xq8nz3f5q5R8MoY4Q_dR6fNPYd3GVIM202tkoGbXFTLUXAz-rqq8PBsA8wr95jcBW8s1lGkHj58tIUuU16rNS8h118YTdYU08M-p05P_9bB03oWK4s70"
```

The bottom status bar shows search filters: 'name', 'Highlight All', 'Match Case', 'Match Diacritics', 'Whole Words', and '13 of 107 matches'.



I used the search function included in spotipy, but wrote my own code for conditions and printing

```
test.py x
Users > hessealy > src > Spotify Proj > env > lib > python3.8 > site-packages > spotipy > test.py > ...
11
12
13
14 results = spotify.search(q="happy", type="track", limit=50)
15
16
17 for item in results['tracks']['items']:
18     if item['explicit']:
19         pass
20     else:
21
22         features = spotify.audio_features(tracks=item['id'])
23         if features[0]['danceability'] > 0.75:
24             print(item['name'], item['popularity'],
25                   item['artists'][0]['name'], features[0]['danceability'])
26
27
28
29
30
```

THIS RETURNS SONGS WITH "HAPPY" IN TITLE, NOT EXPLICIT, AND WITH "DANCEABILITY!"

## THE RESULTS!

RUN YOUR FILE IN TERMINAL

RESULTS FOR "HAPPY"

CHANGE DANCEABILITY TO  
VALENCE -

NEW RESULTS FOR "HAPPY"

```
Happy - From "Despicable Me 2" 88 Pharrell Williams 0.962
Happy Birthday to You 46 Happy Birthday TA 0.802
Happy Birthday 59 Stevie Wonder 0.96
Happy Birthday - Dance Mix 49 Happy Birthday 0.761
Happy Hour 64 Morgan Wallen 0.8
Happy Birthday/Cumpleaños Feliz 33 Dora The Explorer 0.866
Written in the Sand 68 Old Dominion 0.76
Happy Birthday Song 57 Cocomelon 0.79
Birthday - Remastered 2009 53 The Beatles 0.91
Hotel Key 67 Old Dominion 0.964
Happy Idiot 63 TV On The Radio 0.863
Happy 53 The Rolling Stones 0.965
(env) hessealy@Alyssas-MacBook-Pro spotify % python3 test.py
Happy Birthday 59 Stevie Wonder 0.779
Happy Birthday Song 57 Cocomelon 0.866
Happy Birthday Song (Trap Remix) 34 Pj Panda 0.768
Hotel Key 67 Old Dominion 0.758
Happy Now? 62 FINNEAS 0.821
Happy 48 Surface 0.792
Happy 46 Navino 0.897
(env) hessealy@Alyssas-MacBook-Pro spotify %
```

```
Happy - From "Despicable Me 2" 88 Pharrell Williams 0.962
Happy Birthday to You 46 Happy Birthday TA 0.802
Happy Birthday 59 Stevie Wonder 0.96
Happy Birthday - Dance Mix 49 Happy Birthday 0.761
Happy Hour 64 Morgan Wallen 0.8
Happy Birthday/Cumpleaños Feliz 33 Dora The Explorer 0.866
Written in the Sand 68 Old Dominion 0.76
Happy Birthday Song 57 Cocomelon 0.79
Birthday - Remastered 2009 53 The Beatles 0.91
Hotel Key 67 Old Dominion 0.964
Happy Idiot 63 TV On The Radio 0.863
Happy 53 The Rolling Stones 0.965
(env) hessealy@Alyssas-MacBook-Pro spotify %
```

A decorative header featuring stylized musical notes in green, yellow, and blue, some with asterisks. A white banner with a torn edge contains the text "WHAT DID WE LEARN?". Below the banner is a black wavy line.

## WHAT DID WE LEARN?

1. DETAILED DOCS CAN BE FRIEND OR FOE
2. DAUNTING CODE FILES ARE A GREAT WAY TO PRACTICE READING CODE AND SKIMMING
3. PERSEVERANCE IS MORE IMPORTANT THAN KNOWING IT ALL

A white circle icon.

### FURTHER STUDY:

- GETTING A LARGER LIST OF SONGS
- INTEGRATION WITH "GENIUS" TO SEARCH BY LYRICS
- HOW TO REMOVE INTERFERING DATA LIKE ALBUM NAME