EXERCISE 5: API FOR THE WIN

What is an API?

An API is an Application Programming Interface, which as a simple definition enables the retrieval of data from one place (usually a server) for use in a different place.

Why do I need one?

I wanted to create a simple, visually pleasing webpage that would allow a user to input the name of a song and in return receive an image of the lyrics placed into an automatic word cloud. This meant using an API to retrieve the lyrics of the song based on the title, and then use the lyrics in another API to retrieve the word cloud image. I needed two APIs to retrieve data of both the lyrics to the song as well as the image.

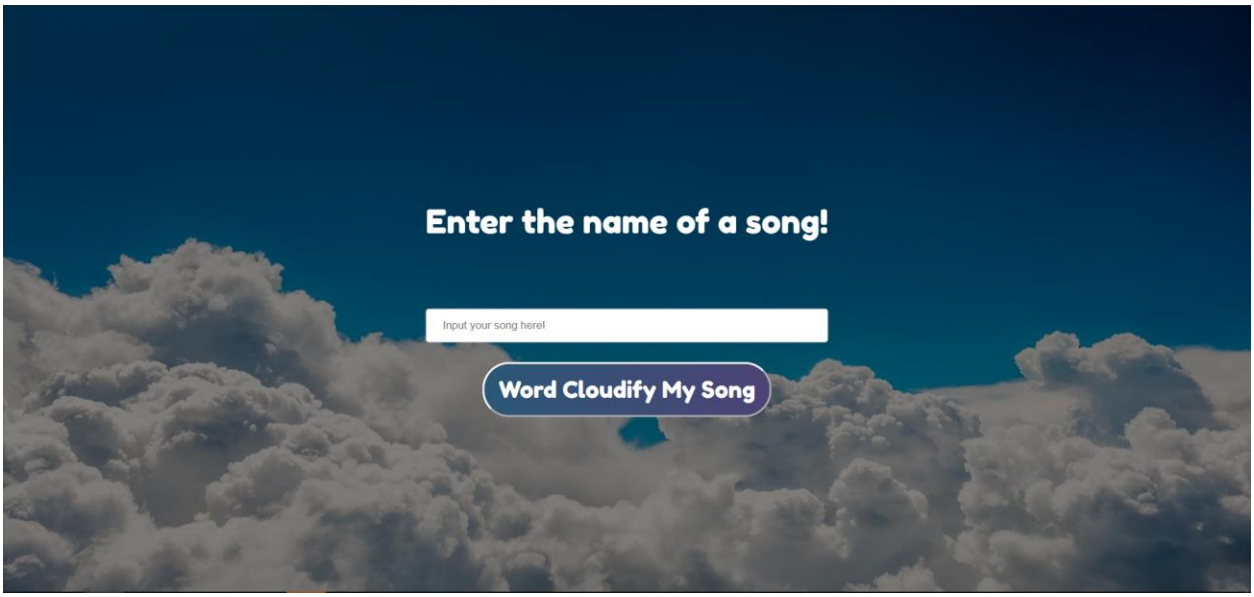
The API's I used

I used two APIs to create this website. The documentation (or the information on how to use the API in your own web development site) for these APIs can be found at:

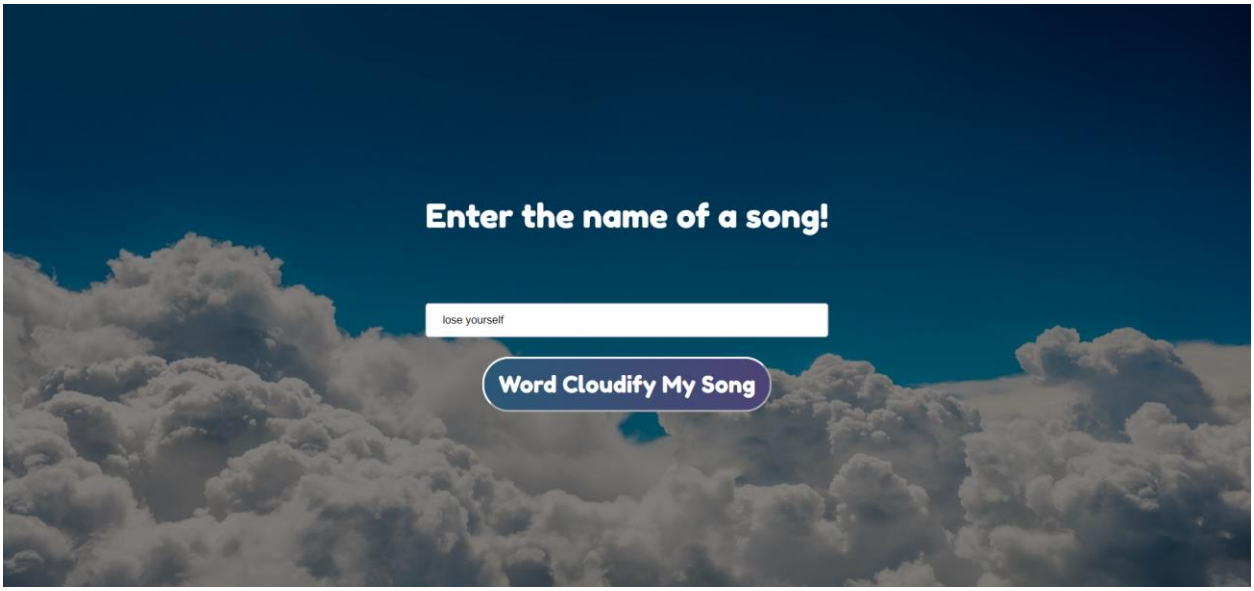
https://developer.musixmatch.com/documentation (To get lyrics from song name)

https://quickchart.io/documentation/word-cloud-api/#getting-started (to get word cloud from lyrics)

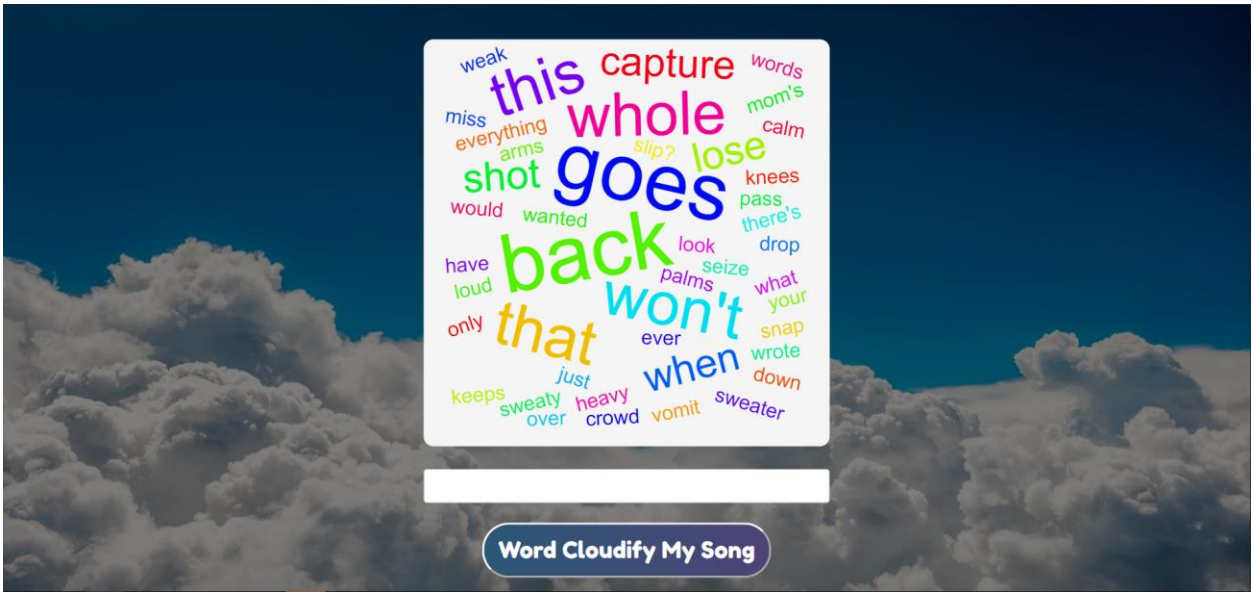
Final Webpage (then onto the nitty-gritty)



The initial page a user will see.



The user then inputs their desired song name.



The webpage gives the user the song lyrics in a word cloud.

API implementation

The first API call I needed to accomplish was getting the song lyrics from the song name. The explanations of my lines of code can be found in these screenshots as comments.

API Background information & setup

```
<script>
  var api_key = "7b44de807c853ff21acb8cfd49b0b787" // API key needed to utilize musixmatch API
  corsURL = "https://eric-cors-proxy.herokuapp.com/"; //CORS anywhere proxy server I utilized instead of the chrome extension (found online)
  $(document).ready(function(){
    $(".btn").click(function(){   //on button click clear all information already on the screen
      $(".failed").html("")       //^^("song not found" text)
      $(".pic").attr("src", "")   //^^ (word cloud image)
      $(".failed").html("Loading...") // (add text that tells the user that the webpage is loading the information)
      var song_name = $(".a input").val(); // sets variable song_name to the users input
      $(".a").html("<input value=''>");
      var musicURL = "https://api.musixmatch.com/ws/1.1/" //sets musicURL variable to the musixmatch API url because I use it multiple times
      data = "matcher.track.get?" + "apikey=" + api_key + "&q_track=" + encodeURIComponent(song_name) //sets the rest of the endpoint url + parameters (q_track)
      URL = musicURL + data //final url
      var track_id = 0; //creates track_id variable
```

First API call (GET track_id from user input song name)

```
    $.ajax({ //first API call
      url: corsURL,
      type: "GET",
      dataType: "json",
      headers: {"Target-URL": URL},
      success: function(response){ //upon call success:
        console.log(response) //^^print the response to the console &
        if (response.message.header.status_code != 200) { //if there was anything accept a 200 code(success code) when retrieving data,
          $(".failed").html("Song not found! Try another one.") //print "song not found"
          $(".failed").attr("style", "font-size: 40px;"); //make font size larger

          return; //and then stop all further calls
        }
        else { //basically if it works and finds the lyrics then
          $(".failed").html("Still Loading....."); //tell the user the website is still loading
          $(".failed").attr("style", "font-size: 40px;");
          //it then also continue onto the second API call
```

Second API call (GET lyrics from track_id)

```
  track_id = response.message.body.track.track_id; //sets track_id variable to the track_id in the JSON Object response
  URL = musicURL + "track.lyrics.get?" + "apikey=" + api_key+ "&track_id=" + track_id //creates the API URL using musixmatch api url + endpoint + parameters
  $.ajax({ //second API call
    url: corsURL ,
    type: "GET",
    dataType: "json",
    headers: {"Target-URL": URL},
    success: function(res) { //upon success of this API call
      console.log(res) //print the response to the console
      if (res.message.header.status_code != 200) { //if there was anything accept a 200 code (success code) when retrieving data,
        $(".failed").html("Song not found! Try another one.") //print song not found
        $(".failed").attr("style", "font-size: 40px;");
        return; //and stop all further calls
      }
      var lyrics = res.message.body.lyrics.lyrics_body; //sets variable lyrics to the
      if (lyrics.length == 0) { //if there were no lyrics in the JSON response (successfully found lyrics but there are none in the song)
        $(".failed").html("Song not found! Try another one.") //print song not found
        $(".failed").attr("style", "font-size: 40px;");
        return; //stop all further calls
      }
      lyrics = lyrics.slice(0,lyrics.length-75) //the musixmatch API only allowed me to utilize the first 30% of the lyrics of the song, and then put copyright
      information with some asterisks. This line of JS removes that copyright information from being utilized within the word cloud.

      //it then moves onto the third call if successful
```

Third API call (wordcloud from lyrics)

```
      var baseURL = "https://quickchart.io/wordcloud"; //second API utilization & third API call (if all else works fine the website will move onto this call)
      data = "?text=" + encodeURIComponent(lyrics) + "&width=" + 500 + "&height=" + 500 + "&scale=linear&minWordLength=4&language=en&fontFamily=sans-serif" //super
      simple API implementation, based on the documentation for wordcloud API
      $(".pic").attr("src",  baseURL+data) //makes word cloud image appear
      $(".failed").html("") //removes failed text
```

HTML

```
<body>
  <div class="centered"> <!--pairs with CSS to center all elements-->
    <p class="failed" style= "font-size:40px">Enter the name of a song!</p> <!--will either display initial text, "song not found", or loading text-->
    <img class="pic" src=""/> <!--display word cloud image-->
    <p class="a"><input placeholder="Input your song here!"></p> <!-- user text input box for song name-->
    <button class="btn">Word Cloudify My Song</button> <!--submit song name button-->
  </div>
  <div id="overlay"></div> <!--makes the background image a little darker-->
</body>
```

CSS

Webpage setup

```
<style>
  body { /*applied to entire webpage*/
    background-color: black; /*black background color*/
    min-height: 100%; /*minimum webpage height*/
    background-image: url(background.jpg); /*background image set to clouds + background image set up*/
    background-size: cover;
    background-repeat: no-repeat;
    background-position: center center;
    background-attachment: fixed;
    font-family: 'Fredoka One', cursive; /*sets font family for website*/

    }
  #overlay { /*applies dark overlay to background image for depth (copied from online)*/
    position: fixed;
    width: 100%;
    height: 100%;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    background-color: rgba(0,0,0,0.5);
    z-index: -1;
    }
```

Button Format

```
button { /*button formatting (copied from online & altered)*/
  z-index: 2;
  color: white;
  font-size: 28px;
  line-height: 28px;
  padding: 17px;
  border-radius: 50px;
  font-family: 'Fredoka One', cursive;
  font-weight: normal;
  text-decoration: none;
  font-style: normal;
  font-variant: normal;
  text-transform: none;
  background-image: linear-gradient(to right, #2b5876 0%, #4e4376  51%, #2b5876  100%);
  display: inline-block;
  border-width: 0;
  transition: 0.5s;
  background-size: 200% auto;
  border-color: white;
  border-width: medium;

  }
button:hover {
  background-position: right center; /* change the direction of the change here */
  text-decoration: none;
  cursor: pointer;
  }
button:focus {
  outline: none;
```

Reflection

Creating this website was one of the first real accomplishments that I have felt in a long time. The first time it worked I literally screamed, I was so excited to see a website respond based on an input. At first, I really struggled with finding an idea for what to build a website, because I did not understand JavaScript as well as I needed to be able to recognize the capabilities it has. Once I decided on an idea, I didn't recognize at first that I was going to need two APIs to accomplish it (and three API calls), but I got so attached to the idea that I decided to just follow through with it. Because of that, I now feel like I understand JavaScript and what it does, I know way more about troubleshooting code, and I also learned a lot about how servers work due quite a bit of research into CORS errors and proxy servers (which I utilized instead of the chrome extension, a URL I found online and attached to the API URL). I am sort of interested in using JavaScript again in the future, and perhaps I have found something I really enjoy doing within UX. I absolutely love the idea of webpages crafted to be responsive to the user, as it makes the user feel like they are part of something (even if it is something small like music preference.) I know that there are a lot of problems I encountered throughout this exercise that could not be cleanly resolved, and I would love to make this a website that can be utilized from any song (and all of the lyrics of the song) one day.