SOUNDCLOUD API WITH JAVASCRIPT

CONNECTING TO THE API

To use the SC object and make requests, we must first initialize it with our client_id:

```
SC.initialize({
  client_id: '[YOUR_CLIENT_ID_HERE]'
});
```

SEARCHING FOR SONGS

To search for songs, we can use the `.get` method that SoundCloud has provided to us:

```
SC.get("/tracks").then(function(response) {
  // things to do after the tracks load...
  console.log( response );
});
```

SEARCHING FOR SONGS

To specify a specific search term, we can pass through a `q` option:

```
SC.get("/tracks",{
  q: "fish"
}).then(function(response) {
  // things to do after the tracks load...
  console.log( response );
});
```

SEARCHING FOR SONGS

Rather than hard-coding, we could dynamically pull the list of tracks from the user's search entry into an input box...

WHAT'S THAT `then` METHOD?

`then` is a JavaScript method for a `Promise` object. A `Promise` is a placeholder for the return value of a `function` that we want to do something with immediately after it finishes...

We need to wait until this value is available and `then` allows us to continue with our code after the `Promise` operation completes.

WHAT'S THAT `then` METHOD?

We use `then` here because we need to wait for the SoundCloud `get` method to finish loading the tracks before we do anything with the response.

WHAT'S THIS `then` METHOD?

`then` expects a callback argument; here we use an anonymous function as our callback.

The callback receives as its argument, the return value from the function we were waiting for. Here, we've called that return value from `SC.get`, response.

```
SC.get("/tracks").then(function(response) {
  // things to do after the tracks load...
  console.log( response );
});
```

STREAMING A SONG

To stream a song, we can pass through a `/tracks/[song id]` relative url to the `SC.stream` method.

```
SC.stream( '/tracks/216847995' ).then(function(player){
  console.log(player);
  player.play();
});
```

STREAMING A SONG

We can extract the stream urls from `SC.get("/tracks")` to pass them through to our player.

We need only dynamically add the ID of the song we wish to stream to the URL:

```
SC.stream( '/tracks/' + track.id ).then(function(player){
  console.log(player);
  player.play();
});
```

STREAMING A SONG

Note that the `play()` method may be called on the return value of `SC.stream`. We can tie the `play` and `pause` methods into our Jukebox programs from Tuesday.

DETECTING THE END OF A SONG

To detect the end of a song, we can either use the `duration` value returned within a `setTimeout` (annoyingly complex).

Or, we can listen for the event of the song ending...

```
SC.stream( '/tracks/216847995' ).then(function(player){
  player.play();
  player.on("finish",function(){
    console.log( "Done-zo" );
  });
});
```

ADVANCING BEYOND THE FIRST SONG

```
var songs = [];
var currentSong = 0;
//load Track objects into songs array elsewhere...

function playNext() {
  SC.stream( '/tracks/' + songs[currentSong].id ).then(function(player){
    player.play();
    player.on("finish",function(){
      currentSong += 1; //increase currentSong by 1
      playNext();       //call itself
    });
  });
}
```