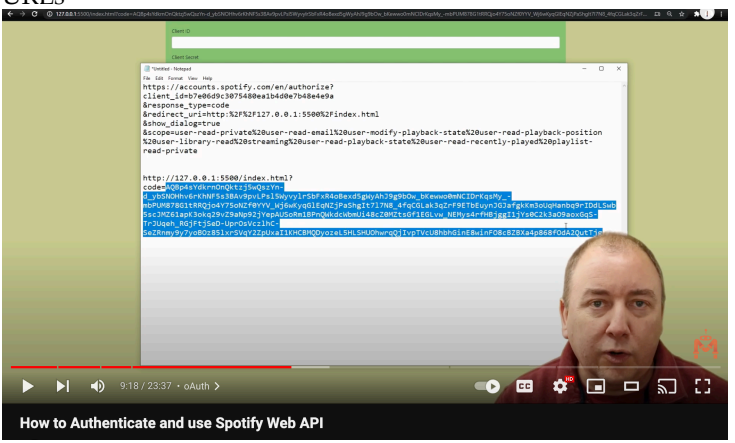
Okay I need help

Watching tutorial on how to auth into Spotify api

Hes going through the docs in order to avid having to use libraries, interact with the api directly.

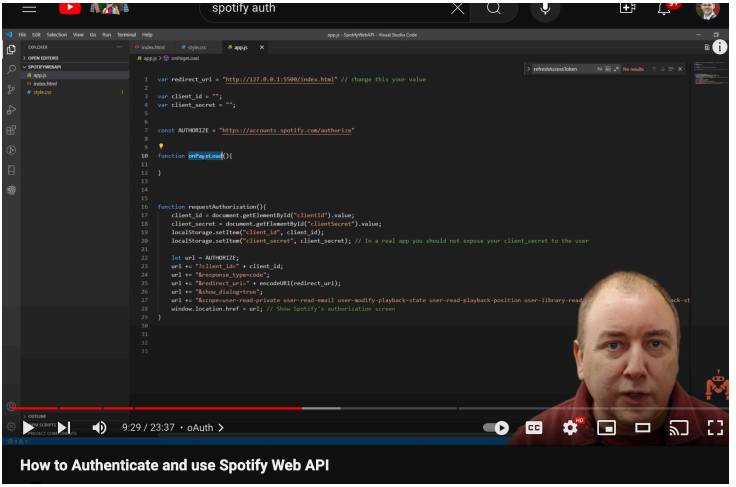
One of the first things I learned here is that you can use the name of a file, a route, in your local host as redirect uri. This guy has his as local host 3000/index.html the index.html part was new to me and is perhaps the reason why mine has been failing. I get from ChatGPT the /callback in both heroku and on streamlit. A possible solution is to when deploying have it go back to the home page, but now authenticated. How does the app know it's been authenticated though? Ah that's it with the two below URLs



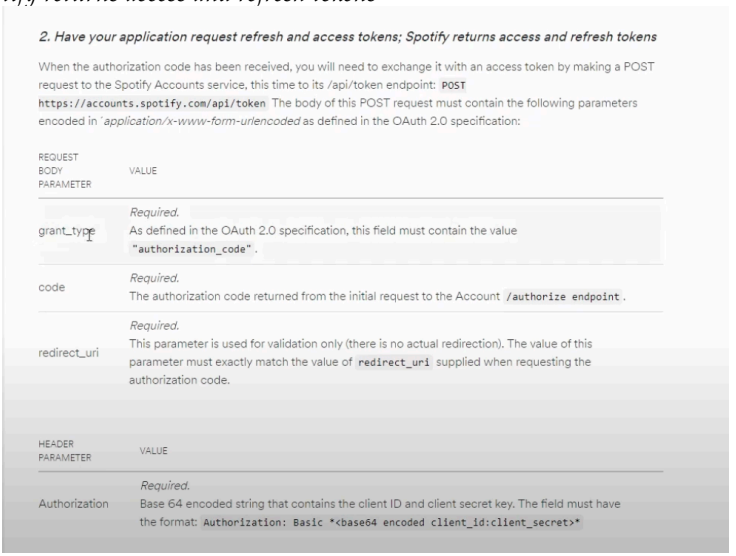
Meaning that the first one is what you send with a redirect uri encoded in html

Then the successful response begins with your redirect uri and adds code= along with an auth code. In streamlit via the internet I saw folks had a problem with streamlit not being able to handle this redirect successfully. With hacks showing ways users are prompted to paste in their own auth code (ie redirect themselves by repasting a link). This makes me think of those pages that pop up and saw you should be authenticated and automatically redirected. If that doesn't happen please click this link. Maybe that is a way around it.

Btw this is the code — vanilla javascript used to hit the Spotify api and auth without other libraries.



Step 2. *Have your application request refresh and access tokens; Spotify returns access and refresh tokens*



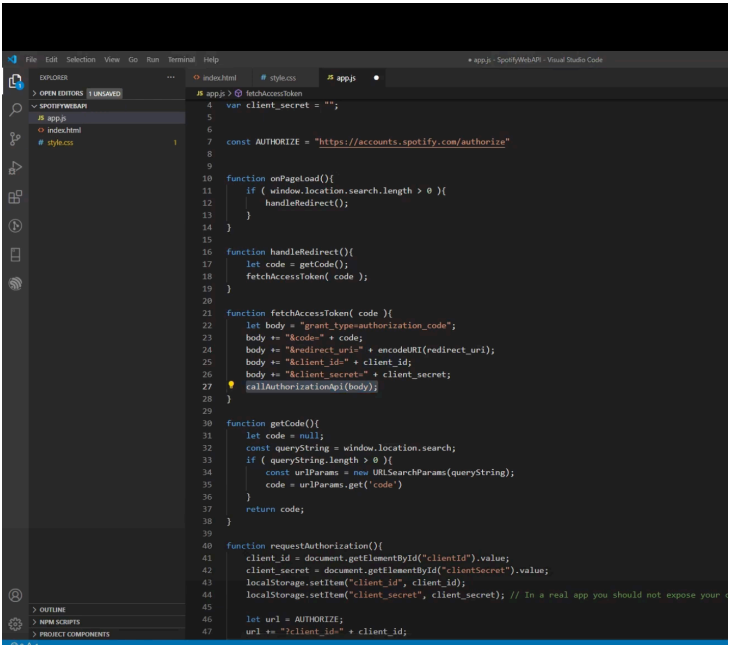
2. Have your application request refresh and access tokens; Spotify returns access and refresh tokens

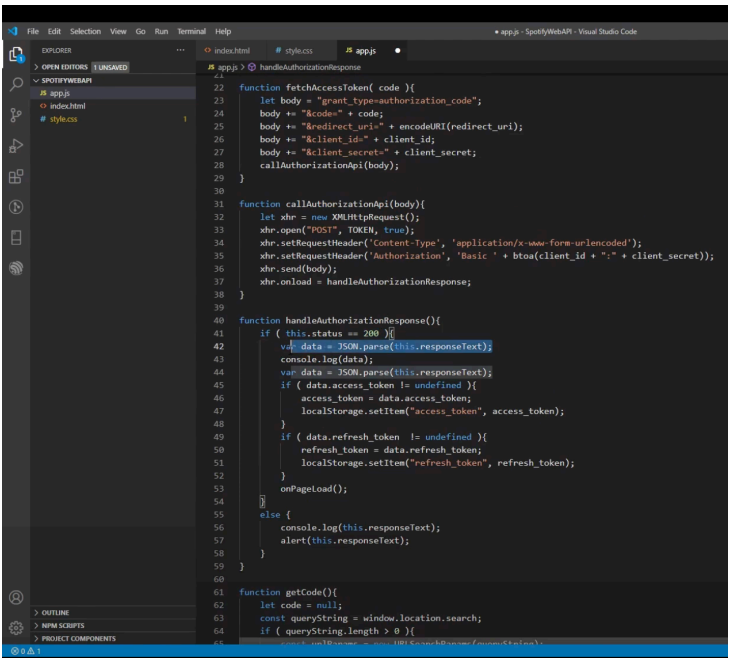
When the authorization code has been received, you will need to exchange it with an access token by making a POST request to the Spotify Accounts service, this time to its /api/token endpoint: `POST https://accounts.spotify.com/api/token` The body of this POST request must contain the following parameters encoded in *'application/x-www-form-urlencoded* as defined in the OAuth 2.0 specification:

REQUEST BODY PARAMETER	VALUE
grant_type	*Required.* As defined in the OAuth 2.0 specification, this field must contain the value `"authorization_code"`.
code	*Required.* The authorization code returned from the initial request to the Account `/authorize endpoint`.
redirect_uri	*Required.* This parameter is used for validation only (there is no actual redirection). The value of this parameter must exactly match the value of `redirect_uri` supplied when requesting the authorization code.

HEADER PARAMETER	VALUE
Authorization	*Required.* Base 64 encoded string that contains the client ID and client secret key. The field must have the format: `Authorization: Basic *<base64 encoded client_id:client_secret>*`

This below is the updated code for completing up to step 2.


File Edit Selection View Go Run Terminal Help
app.js - SpotifyWebAPI - Visual Studio Code
EXPLORER
OPEN EDITORS 1 UNSAVED
SPOTIFYWEBAPI
app.js
index.html
style.css
index.html
style.css
app.js
app.js > fetchAccessToken
var client_secret = "";
const AUTHORIZE = "https://accounts.spotify.com/authorize"
function onPageLoad(){
if (window.location.search.length > 0){
handleRedirect();
}
}
function handleRedirect(){
let code = getCode();
fetchAccessToken(code);
}
function fetchAccessToken(code){
let body = "grant_type=authorization_code";
body += "&code=" + code;
body += "&redirect_uri=" + encodeURI(redirect_uri);
body += "&client_id=" + client_id;
body += "&client_secret=" + client_secret;
callAuthorizationApi(body);
}
function getCode(){
let code = null;
const queryString = window.location.search;
if (queryString.length > 0){
const urlParams = new URLSearchParams(queryString);
code = urlParams.get('code')
}
return code;
}
function requestAuthorization(){
client_id = document.getElementById("clientId").value;
client_secret = document.getElementById("clientSecret").value;
localStorage.setItem("client_id", client_id);
localStorage.setItem("client_secret", client_secret); // In a real app you should not expose your
let url = AUTHORIZE;
url += "?client_id=" + client_id;
OUTLINE
NPM SCRIPTS
PROJECT COMPONENTS


File Edit Selection View Go Run Terminal Help
app.js - SpotifyWebAPI - Visual Studio Code
EXPLORER
OPEN EDITORS 1 UNSAVED
SPOTIFYWEBAPI
app.js
index.html
style.css
index.html
style.css
app.js
app.js > handleAuthorizationResponse
function fetchAccessToken(code){
let body = "grant_type=authorization_code";
body += "&code=" + code;
body += "&redirect_uri=" + encodeURI(redirect_uri);
body += "&client_id=" + client_id;
body += "&client_secret=" + client_secret;
callAuthorizationApi(body);
}
function callAuthorizationApi(body){
let xhr = new XMLHttpRequest();
xhr.open("POST", TOKEN, true);
xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
xhr.setRequestHeader('Authorization', 'Basic ' + btoa(client_id + ":" + client_secret));
xhr.send(body);
xhr.onload = handleAuthorizationResponse;
}
function handleAuthorizationResponse(){
if (this.status == 200){
var data = JSON.parse(this.responseText);
console.log(data);
var data = JSON.parse(this.responseText);
if (data.access_token != undefined){
access_token = data.access_token;
localStorage.setItem("access_token", access_token);
}
if (data.refresh_token != undefined){
refresh_token = data.refresh_token;
localStorage.setItem("refresh_token", refresh_token);
}
onPageLoad();
}
else {
console.log(this.responseText);
alert(this.responseText);
}
}
function getCode(){
let code = null;
const queryString = window.location.search;
if (queryString.length > 0){
OUTLINE
NPM SCRIPTS
PROJECT COMPONENTS

```
}

function callAuthorizationApi(body){
    let xhr = new XMLHttpRequest();
    xhr.open("POST", TOKEN, true);
    xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
    xhr.setRequestHeader('Authorization', 'Basic ' + btoa(client_id + ":" + client_secret));
    xhr.send(body);
    xhr.onload = handleAuthorizationResponse;
}

function handleAuthorizationResponse(){
    if ( this.status == 200 ){
        var data = JSON.parse(this.responseText);
        console.log(data);
        var data = JSON.parse(this.responseText);
        if ( data.access_token != undefined ){
            access_token = data.access_token;
            localStorage.setItem("access_token", access_token);
        }
        if ( data.refresh_token  != undefined ){
            refresh_token = data.refresh_token;
            localStorage.setItem("refresh_token", refresh_token);
        }
        onPageLoad();
    }
    else {
        console.log(this.responseText);
        alert(this.responseText);
    }
}

function getCode(){
    let code = null;
    const queryString = window.location.search;
    if ( queryString.length > 0 ){
        const urlParams = new URLSearchParams(queryString);
        code = urlParams.get('code')
    }
    return code;
}

function requestAuthorization(){
```

At 13:29 he gets the invalid client id bug. Same one it seems I am wrangling with. He says that those have not been properly saved in local storage and don't reauth on redirect. I will get the actual full transcript and post below.

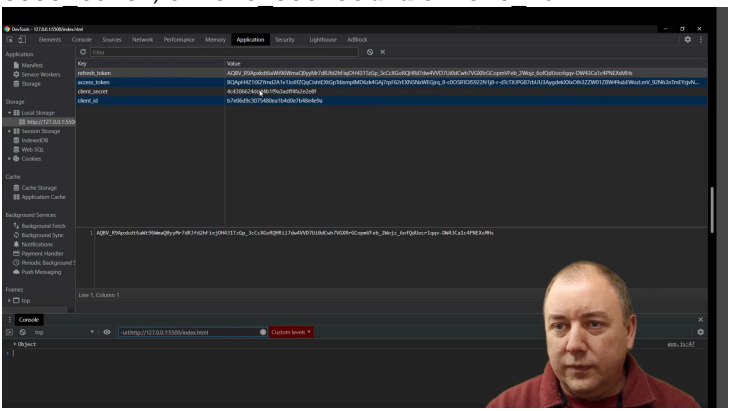
He attempts to fix below by adding local storage information on page load.

```
function onPageLoad(){
    client_id = localStorage.getItem("client_id");
    client_secret = localStorage.getItem("client_secret");

    if ( window.location.search.length > 0 ){
        handleRedirect();
    }
}
```

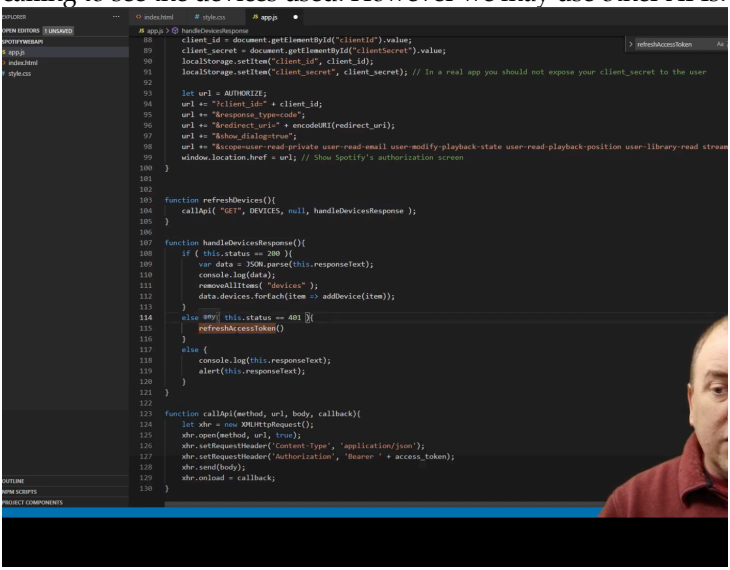
Also here ads a function that clears redirect each time

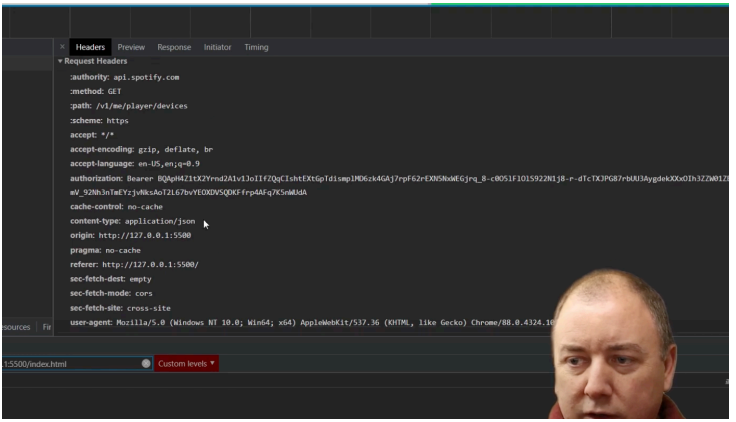
```
}

function handleRedirect(){
    let code = getCode();
    fetchAccessToken( code );
    window.history.pushState("", "", redirect_uri); // remove param from url
}
```

Wow that is smart he's using the dev tools in google chrome in local storage to see that he indeed has `refresh_token`, `access_token`, `client_secret` and `client_id`.



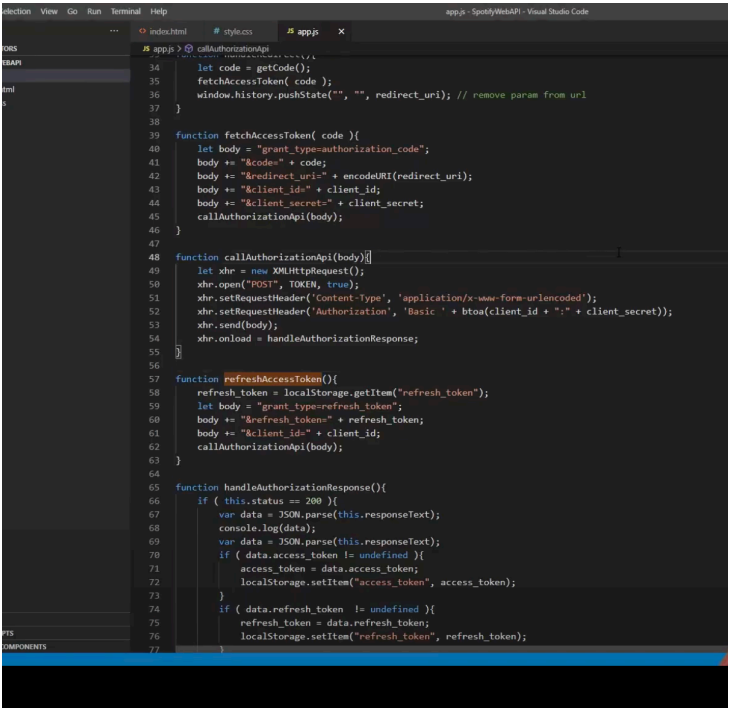
Now he's using those new tokens to make an api call. Here he is calling to see the devices used. However we may use other APIs.





Refresh access tokens

```
    return code;
}

function requestAuthorization(){
    client_id = document.getElementById("clientId").value;
    client_secret = document.getElementById("clientSecret").value;
    localStorage.setItem("client_id", client_id);
    localStorage.setItem("client_secret", client_secret); // In a real app you shou

    let url = AUTHORIZE;
    url += "?client_id=" + client_id;
    url += "&response_type=code";
    url += "&redirect_uri=" + encodeURI(redirect_uri);
    url += "&show_dialog=true";
    url += "&scope=user-read-private user-read-email user-modify-playback-state use
    window.location.href = url; // Show Spotify's authorization screen
}


function refreshDevices(){
    callApi( "GET", DEVICES, null, handleDevicesResponse );
}

function handleDevicesResponse(){
    if ( this.status == 200 ){
        var data = JSON.parse(this.responseText);
        console.log(data);
        removeAllItems( "devices" );
        data.devices.forEach(item => addDevice(item));
    }
    else if ( this.status == 401 ){
        refreshAccessToken()
    }
    else {
        console.log(this.responseText);
        alert(this.responseText);
    }
}


function addDevice(item){
```


app.js - SpotifyWebAPI - Visual Studio Code
index.html
style.css
app.js
app.js > callAuthorizationApi
let code = getCode();
fetchAccessToken(code);
window.history.pushState("", "", redirect_uri); // remove param from url
}
function fetchAccessToken(code){
let body = "grant_type=authorization_code";
body += "&code=" + code;
body += "&redirect_uri=" + encodeURI(redirect_uri);
body += "&client_id=" + client_id;
body += "&client_secret=" + client_secret;
callAuthorizationApi(body);
}
function callAuthorizationApi(body){
let xhr = new XMLHttpRequest();
xhr.open("POST", TOKEN, true);
xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
xhr.setRequestHeader('Authorization', 'Basic ' + btoa(client_id + ":" + client_secret));
xhr.send(body);
xhr.onload = handleAuthorizationResponse;
}
function refreshAccessToken(){
refresh_token = localStorage.getItem("refresh_token");
let body = "grant_type=refresh_token";
body += "&refresh_token=" + refresh_token;
body += "&client_id=" + client_id;
callAuthorizationApi(body);
}
function handleAuthorizationResponse(){
if (this.status == 200){
var data = JSON.parse(this.responseText);
console.log(data);
var data = JSON.parse(this.responseText);
if (data.access_token != undefined){
access_token = data.access_token;
localStorage.setItem("access_token", access_token);
}
if (data.refresh_token != undefined){
refresh_token = data.refresh_token;
localStorage.setItem("refresh_token", refresh_token);