Class #12

Welcome to Monday Night

Today’s objectives

- Generate API specific events and request data from a web service.
- Implement a geolocation API to request a location.
- Process a third-party API response and share location data on your website.
- Make a request and ask another program or script to do something.

Running a simple server

- In a lot of instances, we need a local server running serving our code
- In the past we have not been running a server, we've just been viewing local files in our browser.
- What we *want* is "http://localhost:3000"
- This will allow us to play with API's in this class

Node.js provides this tool

- We'll use a simple tool called http-server
- To install, please run this in your terminal (anywhere will do)
    - `sudo npm i -g http-server`
- `i = install`
- `-g = globally (we can use it in our terminal)`

Running the server

- Make sure you’re in the same directory as the HTML you want to be serving through your new server.
- Run:
    - **`touch index.html`**
- And then run
    - `$ http-server -p 3000`

Server Along

Quick Recap

- API's are a set of endpoints that you can make requests to to retrieve or change data.
- They are commonly provided by SaaS companies (Github, 500px, Namely)
- They commonly use JSON to communicate the data back and forth.

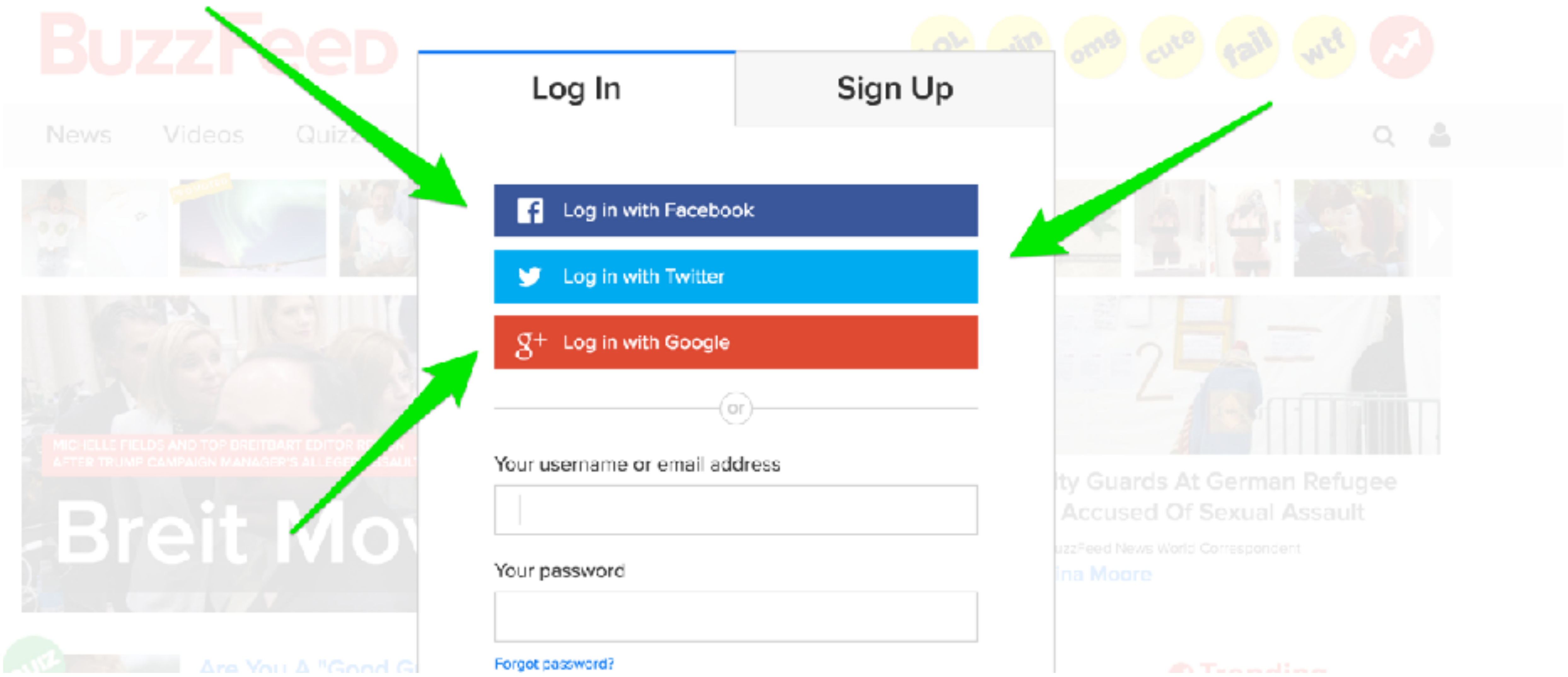
But what about sensitive endpoints…

- Sometimes these API’s have endpoints that expose things like user data.
- It would be bad if everyone could just hit these API endpoints and retrieve whatever data they want.
- To mitigate this, API’s commonly implement a authorization framework called “OAuth(2)”.

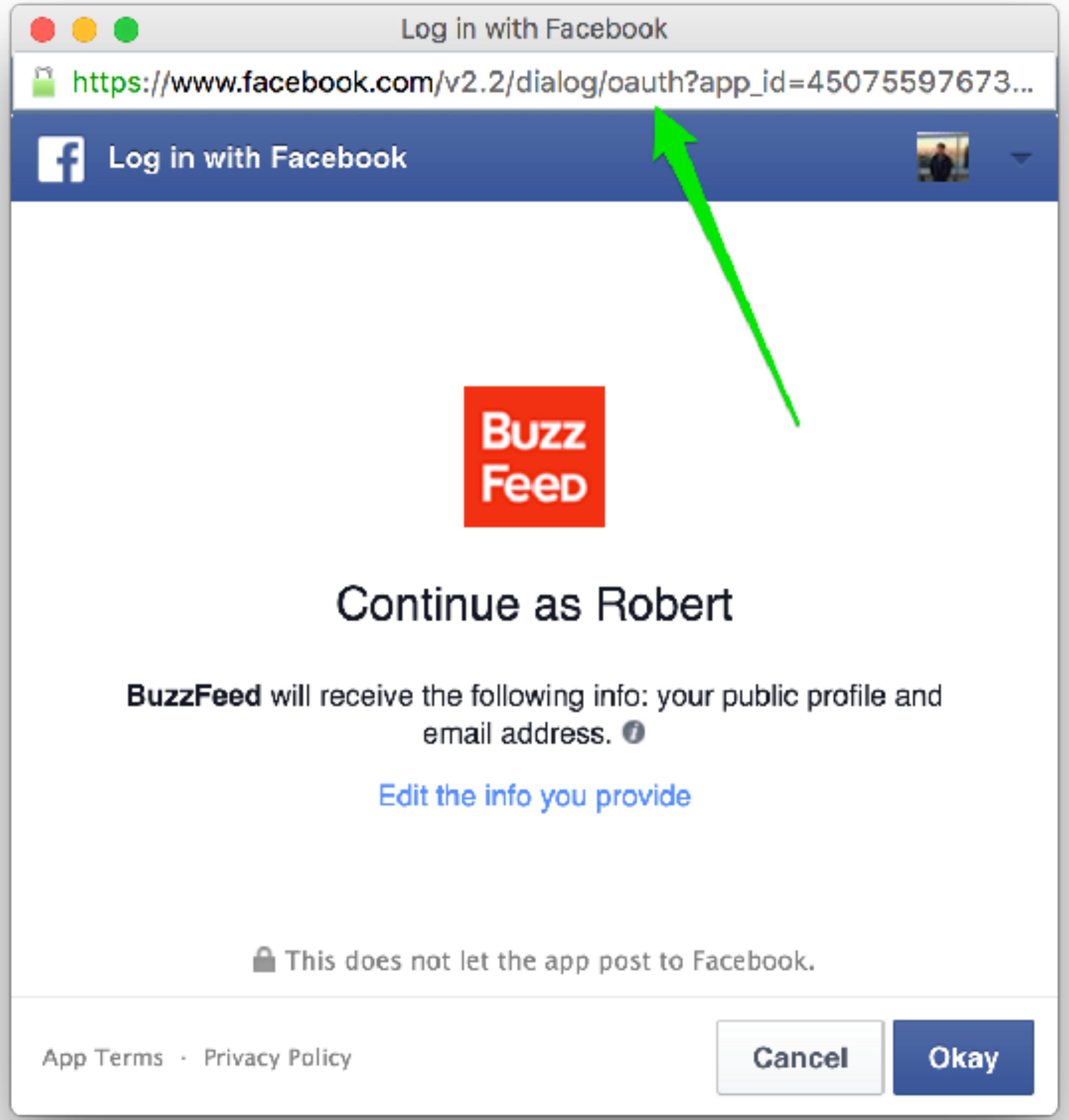
OAuth in a nutshell

- User attempts to visit protected resource
- Server says “no!” and asks you to authenticate.
- You’re presented options, “Login with email” or “Login with Facebook”
- The “Login with Facebook” option is using OAuth2 to provide the website with authentication.

OAuth is all around you



OAuth is all around you



Log in with Facebook

https://www.facebook.com/v2.2/dialog/oauth?app_id=45075597673...

Log in with Facebook

Continue as Robert

BuzzFeed will receive the following info: your public profile and email address.

Edit the info you provide

This does not let the app post to Facebook.

OAuth2 Flow

- User is redirected from our application to the third-party API we wish to authenticate with (let's go with Instagram).

- User logs into Instagram.

- Instagram redirects user back to our application with an access token.

- User can now make requests for resources from the third-party API with the provided access token. The access token is used by the Instagram server to verify User's authentication and authority to make requests.

- User receives resources from Instagram.

Quick note

- Having an access token does not mean you can perform any HTTP method on the resource provider's API
- Nor does it mean your request for any type of data will be granted. Each access token is accompanied with a "scope" of authority.

Example with 500px

- User is redirected from our application to the third-party API we wish to authenticate with (let's go with 500px).
- User logs into 500px.
- 500px redirects user back to our application with an **access token**.
- User can now make requests for resources from the third-party API with the provided access token. The access token is used by the 500px server to verify a user's authentication and authority to make requests.
- User receives resources from 500px.

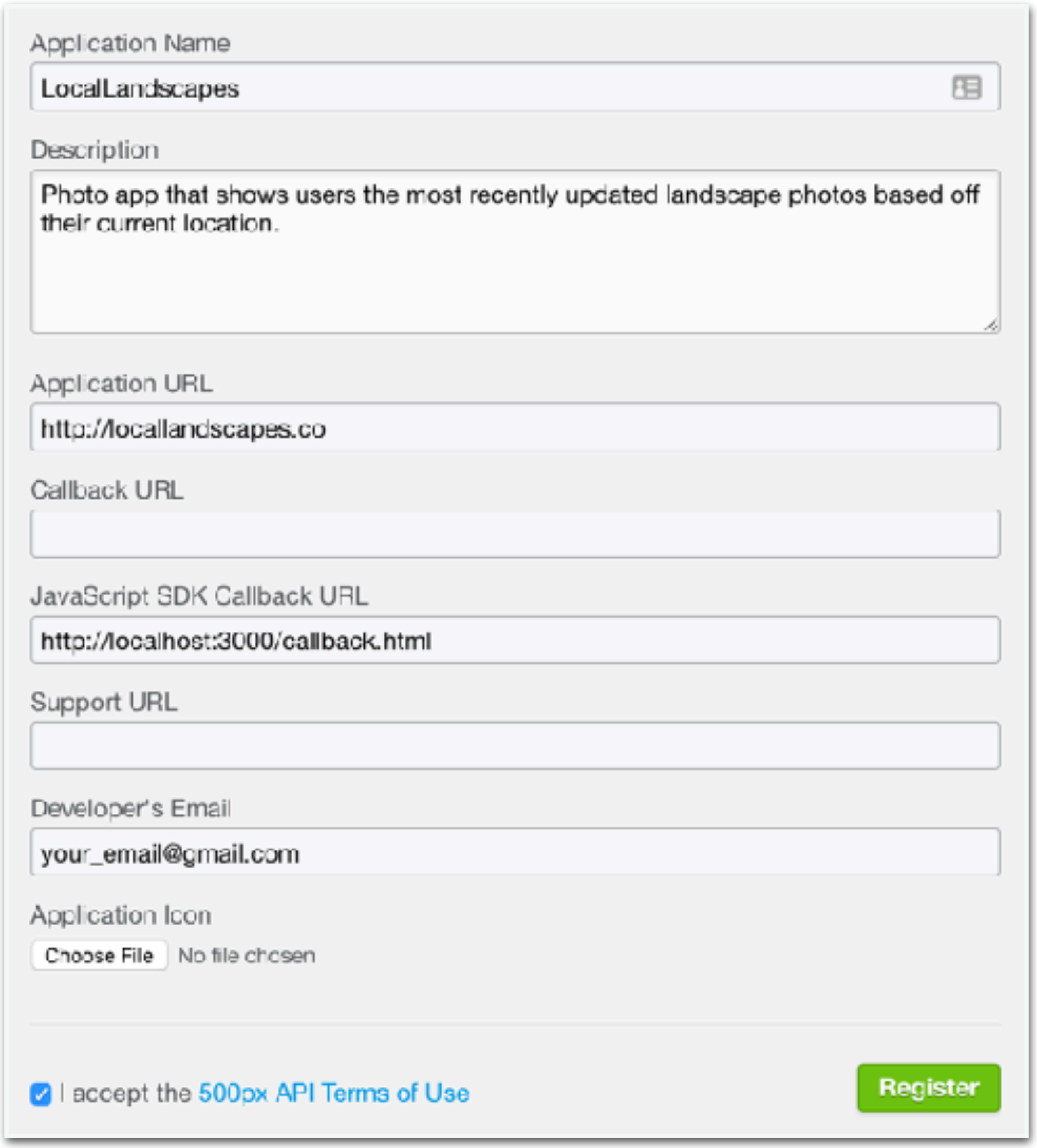
We’ll be using 500px’s API

- Before we can get API keys for instagram’s API, we need to tell them a bit about us.
- To do this, please travel to https://www.instagram.com/developer/ right now in your browser
- Here you will see an option in the navbar to "Manage Clients". Click it.
- Once here register your application by clicking on the "Register a New Client" button.

First Off

- Please go to: https://500px.com/signup and sign up if you do not have an account
- After signing in, go to https://500px.com/settings/applications (Linked under your settings page)

Create an Application



Application Name

LocalLandscapes

Description

Photo app that shows users the most recently updated landscape photos based off their current location.

Application URL

http://locallandscapes.co

Callback URL

JavaScript SDK Callback URL

http://localhost:3000/callback.html

Support URL

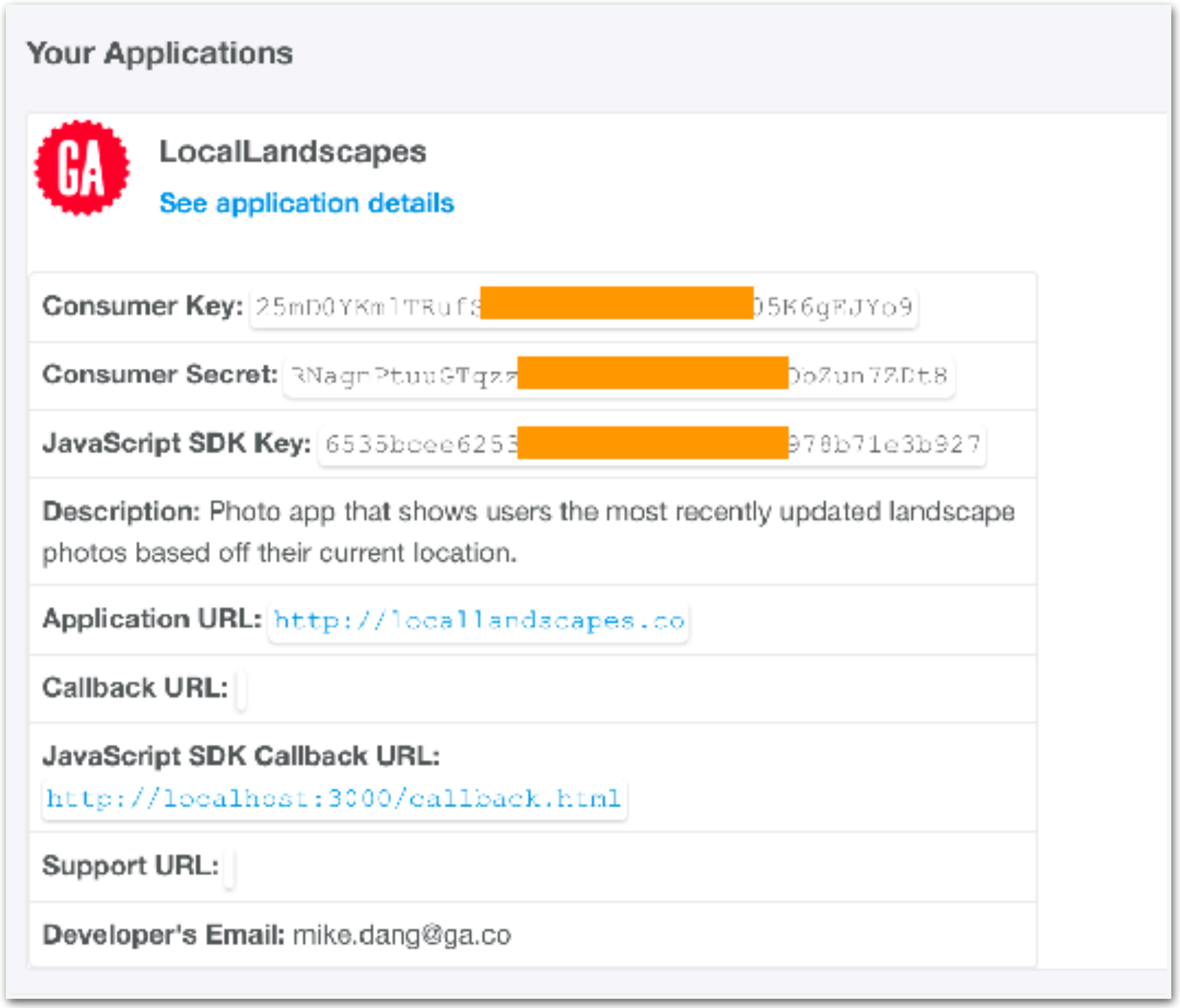
Developer's Email

your_email@gmail.com

Application Icon

☑ I accept the 500px API Terms of Use

Your New Credentials



Your Applications

LocalLandscapes

See application details

Consumer Key: 25mD0YKm1TRufS[redacted]05K6gEJYo9

Consumer Secret: 3NagrPtuuGTqzz[redacted]DoZun7ZDt8

JavaScript SDK Key: 6535bcee6253[redacted]978b71e3b927

Description: Photo app that shows users the most recently updated landscape photos based off their current location.

Application URL: http://locallandscapes.co

Callback URL:

JavaScript SDK Callback URL: http://localhost:3000/callback.html

Support URL:

Developer's Email: mike.dang@ga.co

What these are

- Client ID and Secret are like “email” and “password”. People know your email but not your password (hopefully!).

- You use your client ID when sending people to 500px to authenticate.

- You use your redirect URI to tell 500px where they get sent after they say “yay” or “nay”.

```
https://api.instagram.com/oauth/authorize/?client_id=CLIENT-ID&redirect_uri=REDIRECT-URI&response_type=token
```

Download the starter code

http://bit.ly/js58-12-starter

What we’re building

The app we will be making will show our user an aggregate of the most popular 500px landscape pictures based off their location. Let's refer to our app as LocalLandscapes. Fun times ahead!

Steps we’ll be taking

- Get our 500px developer credentials
- Create our initial view which will have a way for our user to perform OAuth
- Get User's location
- Ping the 500px endpoint with User's location and access token
- Parse through API response for images and put them into view

Lets find the docs

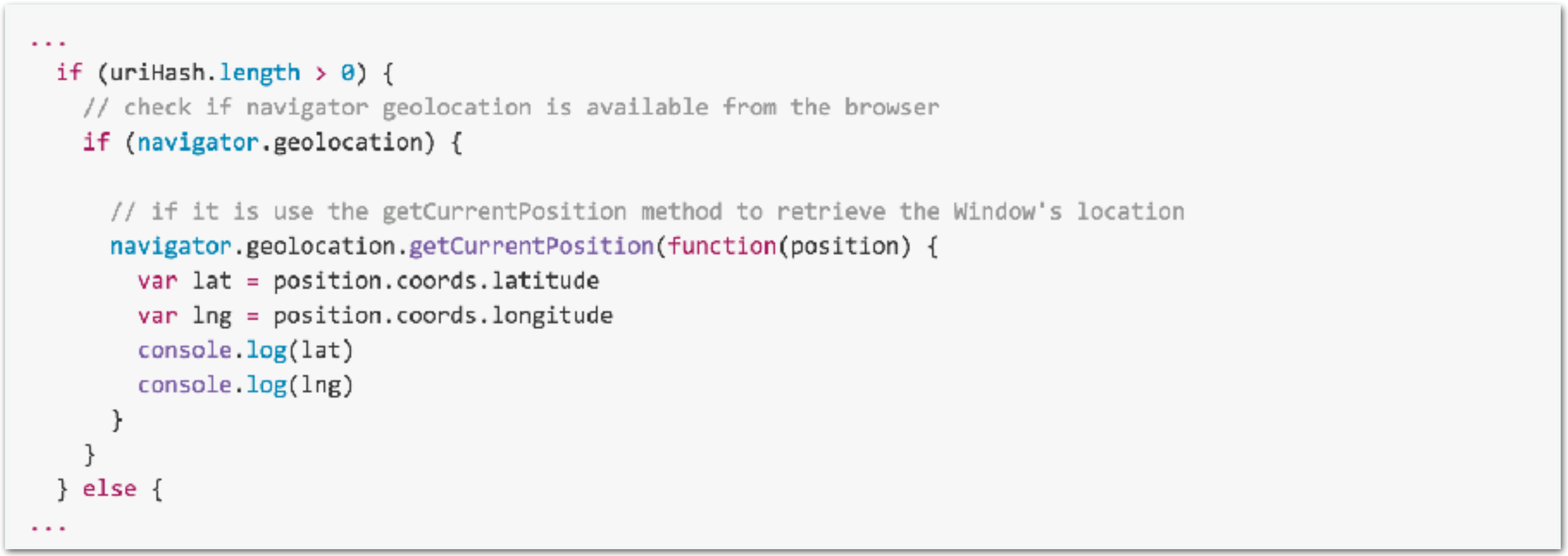
- Lets find the 500px API documentation
- Try googling “500px API documentation”
- We’re looking for the “photo search” endpoint

Class Code Along

Getting a users location

- Many modern browsers allow you retrieve the users location via Javascript.
- This is possible using the Navigator Object.
    - https://developer.mozilla.org/en-US/docs/Web/API/Navigator
- The user needs to allow it, but it gives us the coordinates in latitude and longitude, which is perfect for our needs.

For Example…

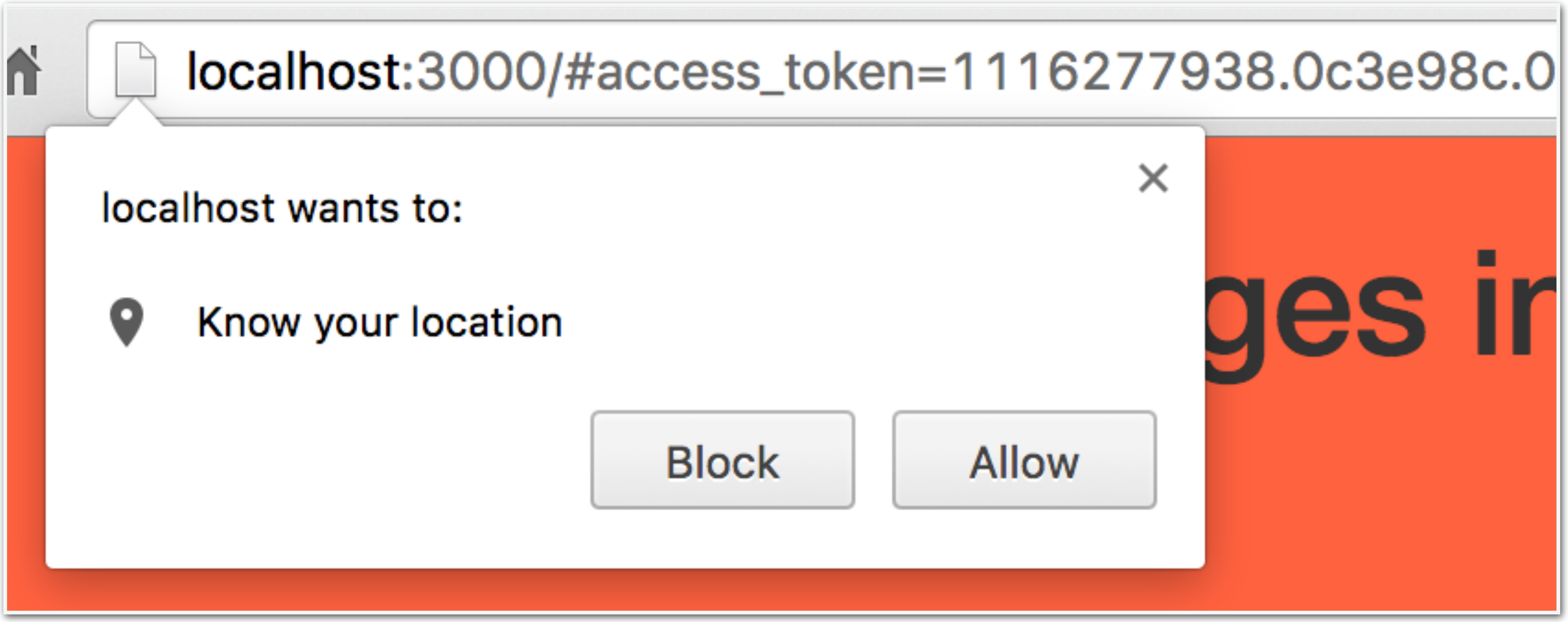


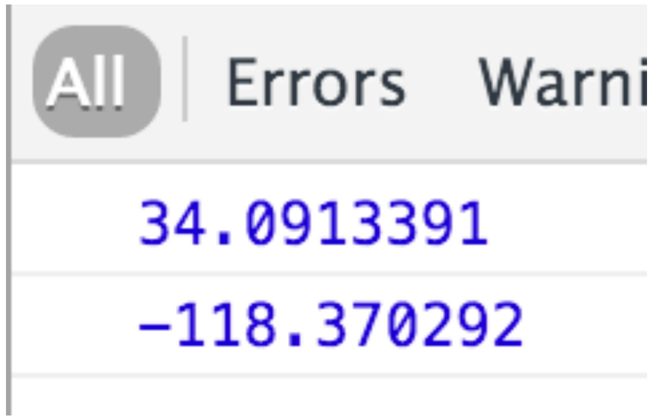
```
...
  if (uriHash.length > 0) {
    // check if navigator geolocation is available from the browser
    if (navigator.geolocation) {

      // if it is use the getCurrentPosition method to retrieve the Window's location
      navigator.geolocation.getCurrentPosition(function(position) {
        var lat = position.coords.latitude
        var lng = position.coords.longitude
        console.log(lat)
        console.log(lng)
      }
    }
  } else {
...
```

- First we check if the browser has the navigator object
- If it does, then we perform “getCurrentPosition()” with a callback function.
- The callback function is given an argument containing coordinates

You should see something like when it works





```
34.0913391
-118.370292
```