

Talking to Your Dog With Ember



Who Am I?

CEO of Ship Shape

- Ember.js Learning Core Team
- ember-math-helpers, ember-shepherd, Shepherd.js
- Love all dogs and have a Frenchie named Odie (Instagram: @odielafrenchie)

robbie@shipshape.io | <https://shipshape.io>

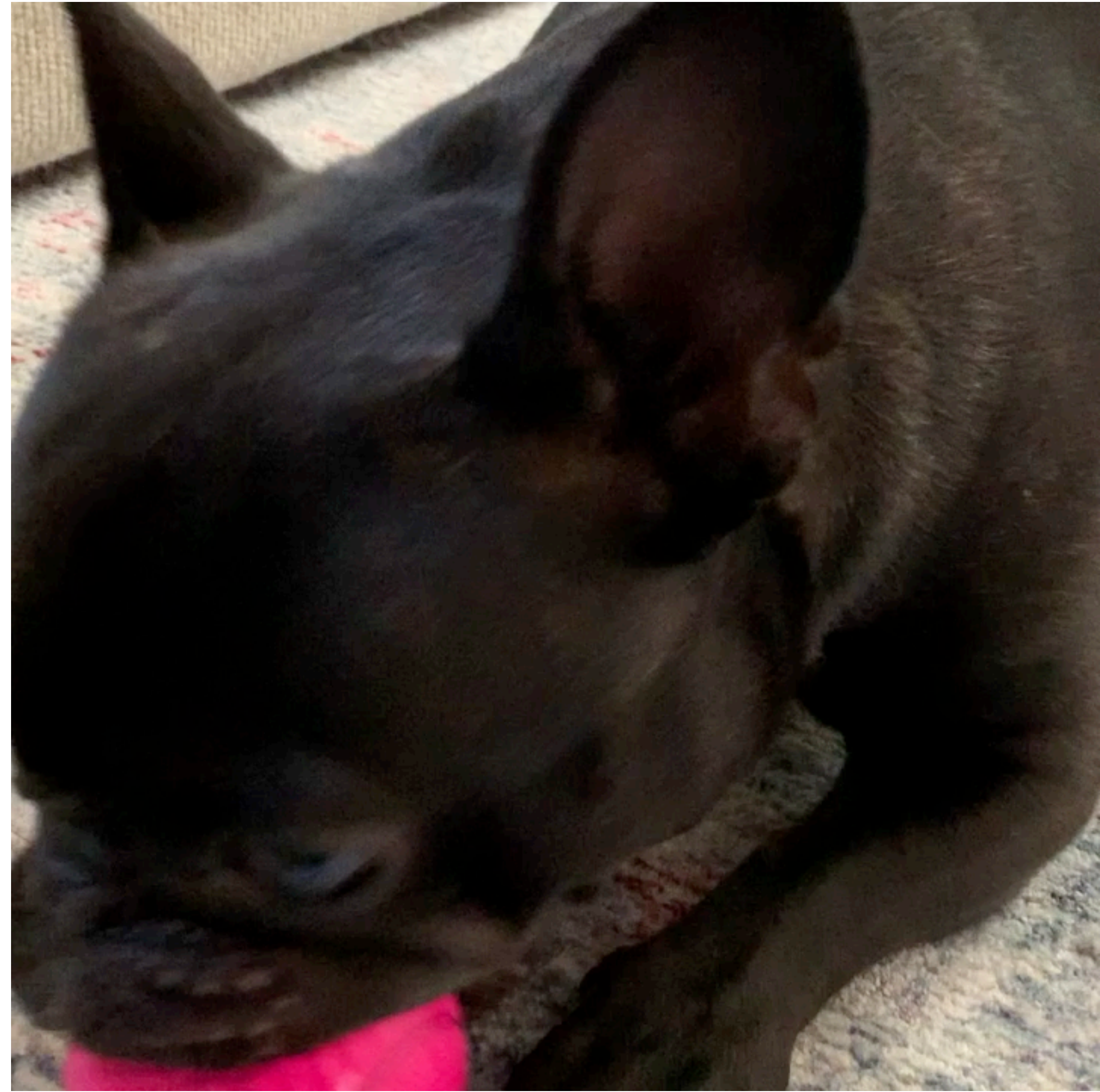
TWITTER: @RWWAGNER90

GITHUB: @RWWAGNER90

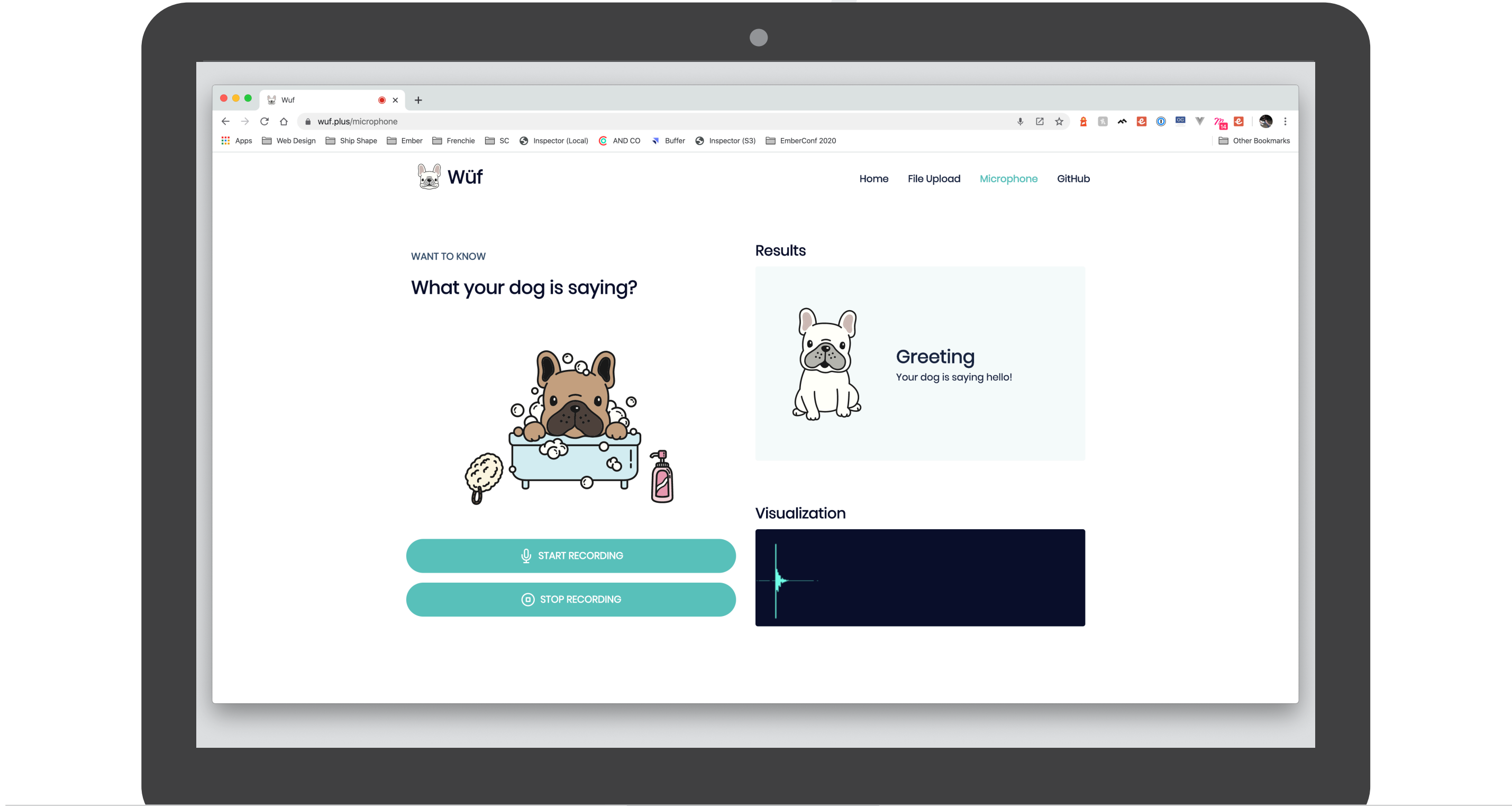
INSTAGRAM: @ODIELAFRENCHIE

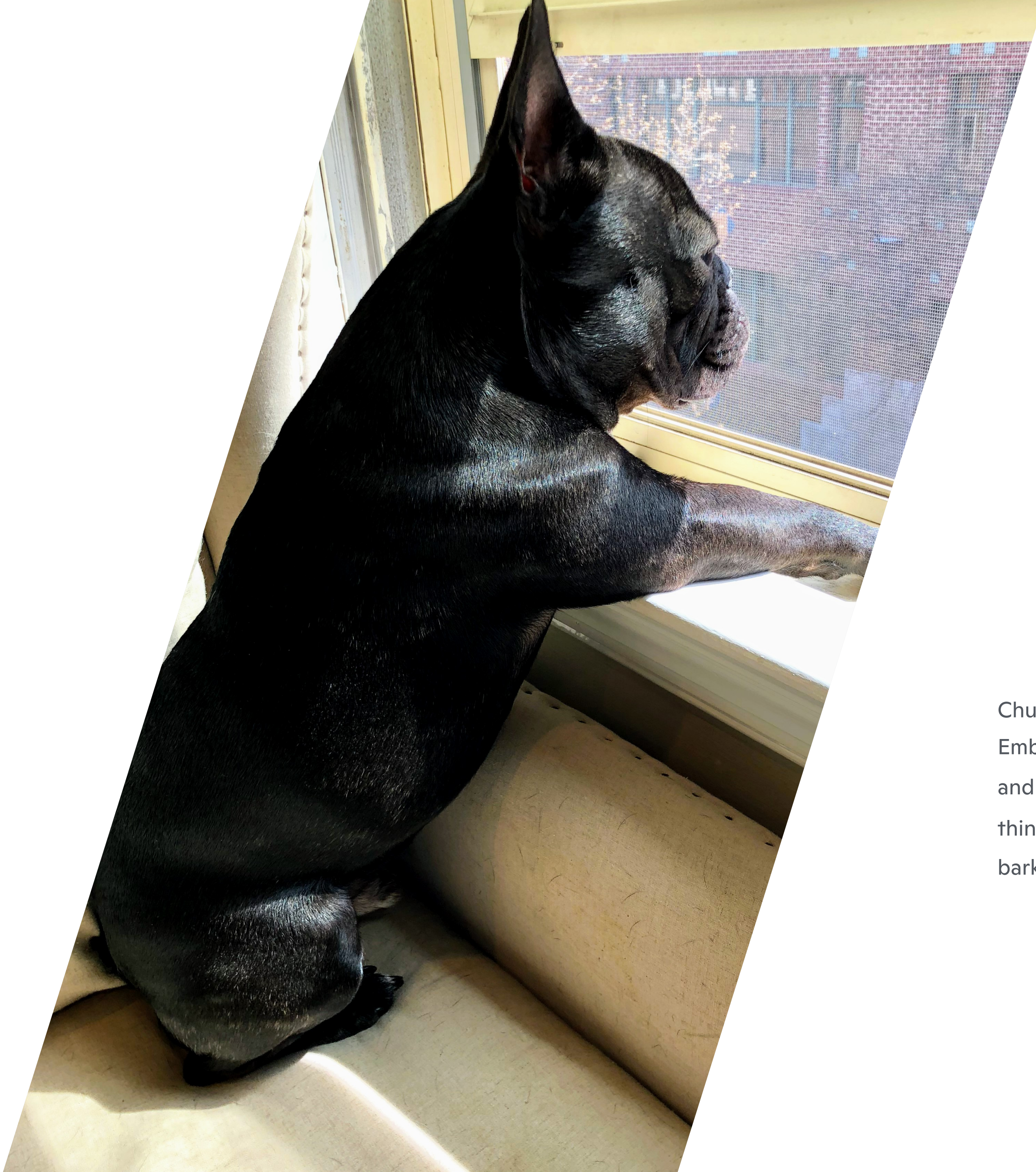
Motivation

Odie is constantly barking at something. He will bark playfully at the fireplace tools when he wants to be chased around, ferociously at anyone entering the house, and incessantly when squeaking his ball, as if he is trying to talk to it. Understanding why he barks the way he does was the major motivation behind exploring this talk.



Demo





The Idea

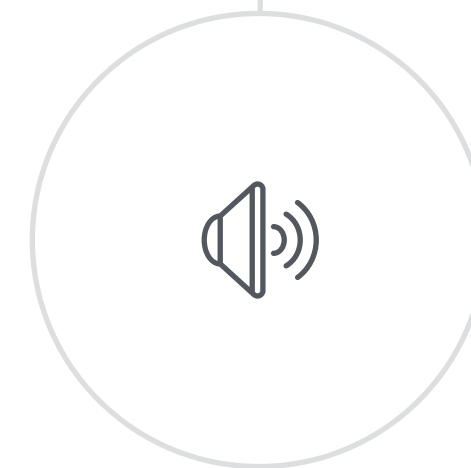
Chuck Carpenter, Ship Shape's COO, and I were working one day when the EmberConf CFP came out. We were brainstorming ideas and Odie kept barking and running around. We thought "it would be nice to know what Odie is thinking and why he is barking", and the idea to create an app to decode dog barks was born.

The Process



Web Audio API

The Web Audio API provides methods for working with audio and video in JavaScript.



Determining Dog Bark Types

Studies have shown that both dogs and humans can tell the difference between lonely, happy, and aggressive barks, and dogs have been shown to react differently to dogs they are familiar with.



Deciphering Audio Data

Since there is a scientifically proven difference between types of dogs barks, there must be some way we can use the Web Audio API to try to decode them.


What *is* Web Audio API?





When In Doubt, Google It


- Several real-time analysis examples
- Docs from Mozilla
- Stack Overflow posts


web audio api analyze sound


 All

 Videos

 Images

 News

 Shopping

 More

Settings

Tools


About 6,620,000 results (1.00 seconds)

Web Audio API is a high-level **JavaScript API** for processing and synthesizing **audio** in **web** applications. The aim of the **API** is to enable things like dynamic **sound** effects in games, **sound** processing in music production applications, and real-time **analysis** in music visualisers. Feb 28, 2013

ianreah.com › 2013/02/28 › Real-time-analysis-of-streaming-audio-data...
[Real-time analysis of streaming audio data with Web Audio API](#)

?

About Featured Snippets

 Feedback

webaudioapi.com › book › Web_Audio_API_Boris_Smus_html ▾

[Analysis and Visualization - Web Audio API](#)

The main way of doing **sound analysis** with the **Web Audio API** is to use `AnalyserNodes` Once this node is in your graph, it provides two main ways for you to inspect the **sound** wave: over the time domain and over the frequency domain. The results you get are based on FFT **analysis** over a certain buffer size.
You visited this page on 2/21/20.

developer.mozilla.org › Web technology for Developers › Web APIs ▾

[Web Audio API - Web APIs | MDN](#)

Oct 24, 2019 - The **Web Audio API** provides a powerful and versatile system for ... input of a destination (`AudioContext.destination`), which sends the **sound** to the ... able to provide real-time frequency and time-domain **analysis** information, ...

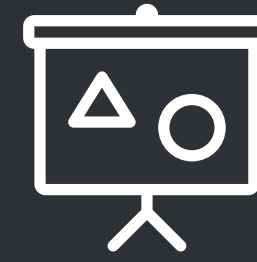
developer.mozilla.org › ... › Web APIs › Web Audio API ▾

[Visualizations with Web Audio API - Web APIs | MDN](#)

Mar 23, 2019 - One of the most interesting features of the **Web Audio API** is the ability to ... Another nice little **sound** visualization to create is one of those ...
You've visited this page 2 times. Last visit: 2/21/20

Web Audio API

The Web Audio API provides an `AnalyserNode`, which has several helpful methods for analyzing audio and getting frequency and waveform data from it.



`fftSize` and `frequencyBinCount`

`fftSize` represents the window size in samples that is used when performing a Fast Fourier Transform, and `frequencyBinCount` is always $1/2$ of `fftSize`.



`getByteFrequencyData()`

Copies the current frequency data into a `Uint8Array`. The frequency data is composed of integers on a scale from 0 to 255.



`getByteTimeDomainData()`

Copies the current waveform or time-domain data into a `Uint8Array`. The data is composed of integers 0-255 which map from -1 to +1, so 128 is 0.

First Attempt

01. createMediaElementSource

Initially, I was trying to use a media element as a source, which worked, and was intuitive, but it only provided the playing audio, not the full buffer.

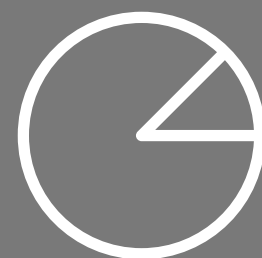
02. Frequency

I was incorrectly getting only the frequency data, which does not have a time component, so to get the data over time we had to switch to `getByteTimeDomainData`.

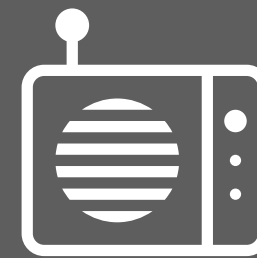
01.



03.



02.



04.



03. Small Data Snippets

The analyser methods only provide frequency and waveform data for the small section of audio they are called on, but we need data for the whole file.

04. How To Use The Data?

Once we do have the data, what does it mean and how do we use it?

Second Attempt

01. OfflineAudioContext

OfflineAudioContext allows us to use an entire audio file loaded into a buffer, rather than only analyzing the part of audio that is currently playing.

02. getByteTimeDomainData

We're using a script processor to run `getByteTimeDomainData` on `audioprocess`. This allows us to get time domain data for the whole file.

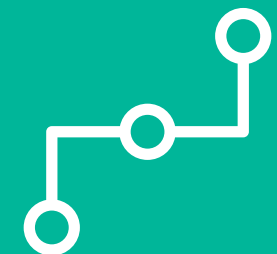
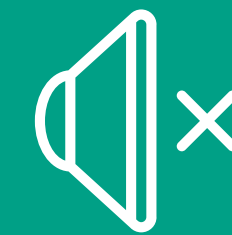
03. Upload Files / Use Microphone

Leveraging `ember-file-upload`, we can support file uploads with ease, and we can use `getUserMedia` to access microphone data.

04. Visualizations

Heavily borrowing from [Visualizing Audio #3 Time Domain Summary](#), I was able to add a visualization of the audio data, so we can see where it spikes.

01.



02.

04.



03.



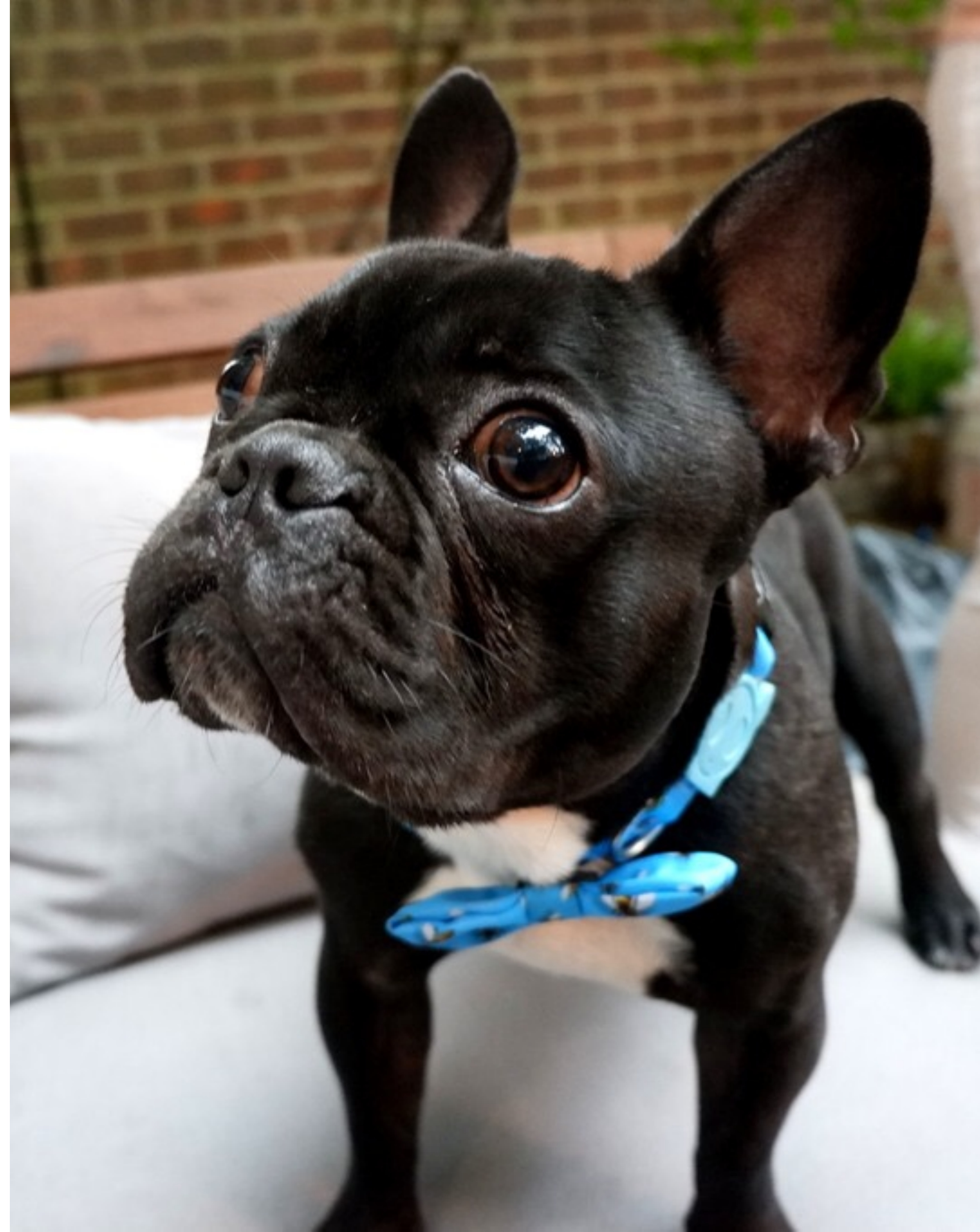


Dog Bark Science

Disclaimer: I am not an expert in dog barks. This project is based off of several different scientific studies, and is based on my personal interpretation of the results. It can certainly be improved, and may not be 100% accurate.

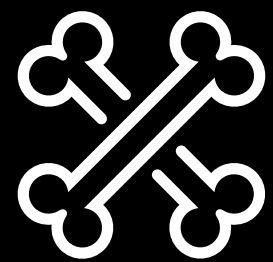
Dog Bark Science

Studies have shown that dog barks fall somewhere in the ~250-4000 Hz range at most shelters, all breeds seem to have some component of their bark registering in the ~1000-2000 Hz range, and barks register between 80-90 decibels from a distance of 5 meters.



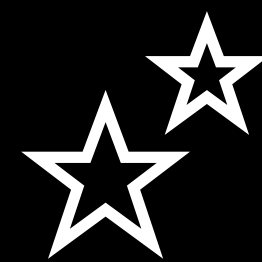
Bark Types

There are seemingly infinite combinations of potential types of dog barks. You could have any pitch, duration, and number of total barks, as well as subtly different inflections on each, so how do we determine meaning from such a huge set of possibilities?



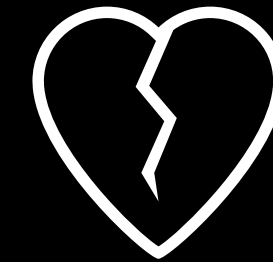
Alert

Rapid barking at a mid-range pitch can signal to the pack that there is a problem or something to investigate.



Greeting/Playful

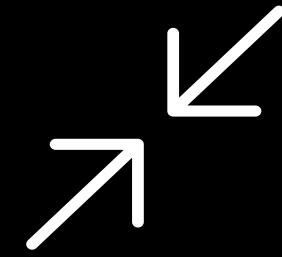
Stutter barks or rising pitch barks can signal a dog is happy and wanting to play. A single bark can be a greeting.



Distress

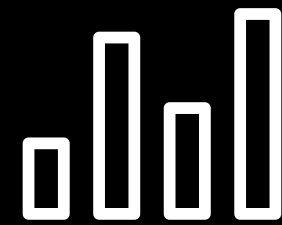
Barking with long periods of time between each utterance can mean “Is anyone there? I’m lonely and in need of companionship”.

Mapping Data To Bark Types



Limiting Frequency

Since dog barks only make sounds in the ~250-4000 Hz range, we can throw out data > 4000 Hz.



Finding dB Spikes

We can find spikes in the waveform data above 0.55 to try to assume the barks occurred there.



Modes

We can take the mode of the frequency range per bark occurrence to determine the pitch of that bark.

Let's apply what we learned about the different types of dog barks to what we learned about the science behind each bark type, in order to identify the type from the audio data.

Integrating With Ember

Ember helps us to take our vanilla JS implementation to the next level. We gain the flexibility to structure our code into components, the power of glimmer, and a huge set of available addons, which allow us to bolt on PWA functionality with ease.



ember-service-worker

Service workers allow our app to work offline, so it can continue to be used when connection is spotty and ember-service-worker makes installing them a breeze.



ember-web-app

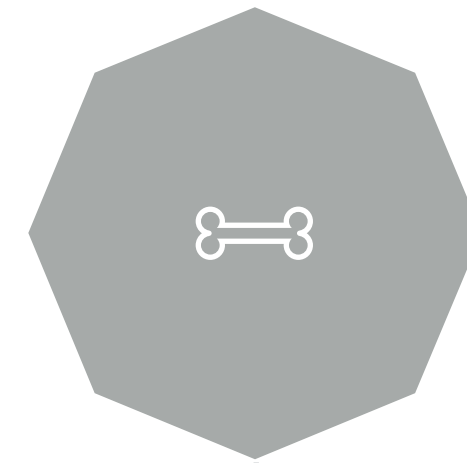
Ember-web-app creates a manifest that allows us to specify the name, description, icon, etc for the app and makes it installable on various devices.



Opinionated

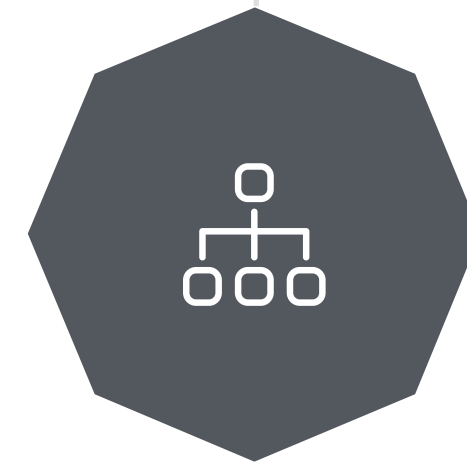
Ember's strong conventions help us to structure our app cleanly, quickly install packages as addons, and leverage the power of glimmer components.

Future Work



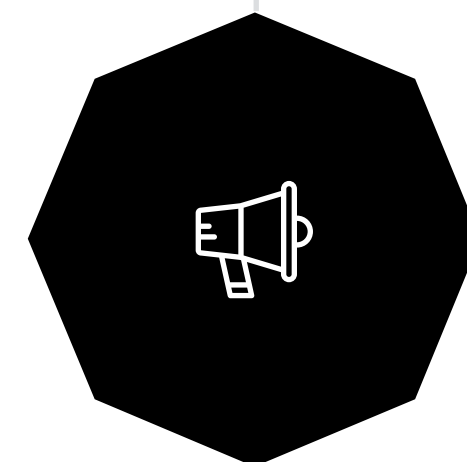
Add More Bark Types

We only have four rough categories currently, but there are 10 basic types at a minimum, as well as other nuanced bark types, which could be supported for more exact output. The one I particularly want to support is the lonely bark, to identify episodes of separation anxiety.



Refine Frequency Ranges

Our current low, mid, and high frequency calculation is a linear distribution of the three, but we should refine it to have more buckets, like mid-high, mid-low, etc. for better results.



Add Talk Back Feature

After fully mapping out the bark types and frequency ranges, it should also be possible to add a “talk back” feature, which could take barking audio samples and tweak them to respond to your dog.


A black French Bulldog is lying on a light-colored, textured couch. The dog is looking towards the camera with its large, upright ears. In the background, a window with a screen shows a view of a residential area with houses and trees. The image has a dark, semi-transparent overlay on the right side where the text is located.

Try Wüf Now

Visit <https://wuf.plus> to try it now!

Sources

- https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API
- <https://stackoverflow.com/questions/24083349/understanding-getbytetimedomaindata-and-getbytefrequencydata-in-web-audio>
- <https://www.petsafe.net/learn/10-translated-barks-know-what-your-dog-is-saying>
- <https://www.sciencemag.org/news/2014/08/dogs-glean-information-each-others-barks>
- <http://apprentice.craic.com/tutorials/32>

A photograph of two dogs, a small black dog on the left and a larger brown dog on the right, lying on a carpeted floor. They are both holding and playing with a thick, braided rope toy that has frayed ends. The black dog is wearing a blue and white checkered collar. The brown dog is wearing a red collar. In the background, there is a white knitted blanket and a blue and white striped basket. The image is dimly lit and has a dark overlay.

Thank you for your attention.

<https://shipshape.io/> / E-mail: robbie@shipshape.io