Ear to Eye

Karen Bowman

Introduction

This is a sound visualization project, which takes the format of a website. Its purpose is to encourage its users to interact with sound in new ways and learn something new about audio. It has several informational pages on aspects of sound, as well as interactive pages where the user can make sound and watch how a visual react to that sound. It also has frequency and pitch displays that display statistics in real time.

>> PLEASE OPEN IN CHOME ONLY. Other browsers may be unsupported. The site may be launched by opening “index2.html” in the browser.

Table of Contents

Problem Description

Sound is part of everyone's lives. We hear noise every day; some sounds we ignore, others tell us something important and some are for enjoyment. In most people worlds however visual stimulus takes precedence over auditory stimulus. Though we might hear certain noises every day, the more repetitive they become, the less typical it is to try to determine their subtle differences

I believe that people will enjoy and benefit from taking time to enjoy small background and unintentional noise as well as music. There is something mesmerizing to the sound of paper and pop of a soda can - I would like my users to experience some curiosity relating to the background noise around them.

The problem is to create an interactive site or application that will interpret audio in a new and interesting way, through using visuals to catch a user's attention and teach them something about sound.

Project Objectives

---

1. Inform the user about frequency, pitch, and other aspects of sound.
    a. Include information popups and sections that explain the science of sound and definitions.
2. Provide animations that 'forces' the user to make noise, and pay attention to the type of noise they are making
    a. Create responsive animations that use some aspect of sound to create a changing visual - something interesting enough that the user feels they have to make a sound in order to manipulate it in the way they are tempted to.
3. Gain the users interest on their first visit to the site.
    a. Use color and design to gain user interest.
4. Cause the user to think about background sound more carefully.
    a. Keep the user engaged long enough with animations that they leave thinking about the ways they tried manipulating sound in new ways.
5. Be easily navigable.
    a. Don't be confusing and keep the user from leaving out of frustration or boredom.

Overview of Approaches

I took a number of different approaches in order to access audio and create the animations I wanted. I also had to experiment a good deal with menus and page layout.

Menus

I originally wanted a top bar navigational menu, as it is more classic and easily navigable. However, the pages ended up being rather plain and so I experimented with other menu types, including the dynamic and “middle of the page” menus. I settled on the circle menu because it added the most visual interest. It is slightly glitchy sometimes, so if given a chance I would aim for a more smooth alternative.

Animation

I originally intended to create a jpg stream for animation; essentially to create jpg and then display them at a fast rate. I soon realized that it would take far too much time to render images of the right size and resolution in this way, especially in custom animations.

Research indicated that canvases or SVG elements were best for fast animation. I decided to choose SVG displays because as a “scalable vector graphic”, you pass not large visual files, but more compact instruction of what to draw to the browser. This theoretically makes for better loading speed. It was worked fairly well for me; though the speed is not great, it is very easy to manipulate svg elements with javascript, so they are easily modified en masse in animation.

A lot of animation work also involved tweaking speed and size constants so movement meshed well with the type of audio typical on a laptop.

Audio

The only audio library that I could get to work on my laptop was the audioContext library, by Mozilla. It is supported on pretty much all browsers and has a good interface for fetching microphone permissions, which was my issue with other interfaces.

Product Design

It was very important to me to have excellent visuals in this site since much of the point was to gain and keep user interest through aesthetic appeal. In that light, the following were goals for Product Design:

- Easy Navigation
- Clear Purposes for Each Page
- Eye-Catching Color
- Consistency in Style Across Pages
- Simple Design

Each page has the same menu, to promote simplicity, ease of navigation, and consistency. There is also a back button on all non-home pages so that the user can return to the home menu where there are more directions if confused.

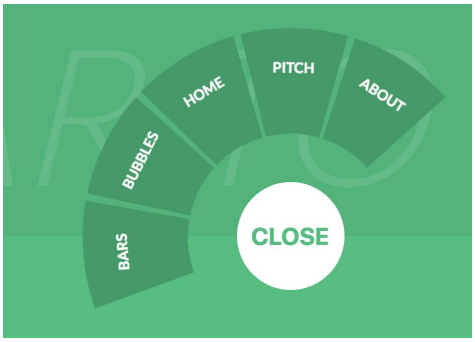
Each page also has a unique color palette to help catch the user's interest and promote appreciation of the sound visuals displayed. There are instructions for each page so that the user is (hopefully) never confused.

Features

Overview

The overall construction of this project is in webpage form- meaning the user is meant to explore the different web pages and capabilities of each one via the menu. As aligns with this format, there are a number of different features; a page with pitch and frequency detection, a "bubble" animation based on volume, a bar animation based on a Fourier transform of audio frequencies, a home page, and an about page. All pages have an identical responsive circular menu. Everything is developed in Javascript, Jquery, Html, and CSS.

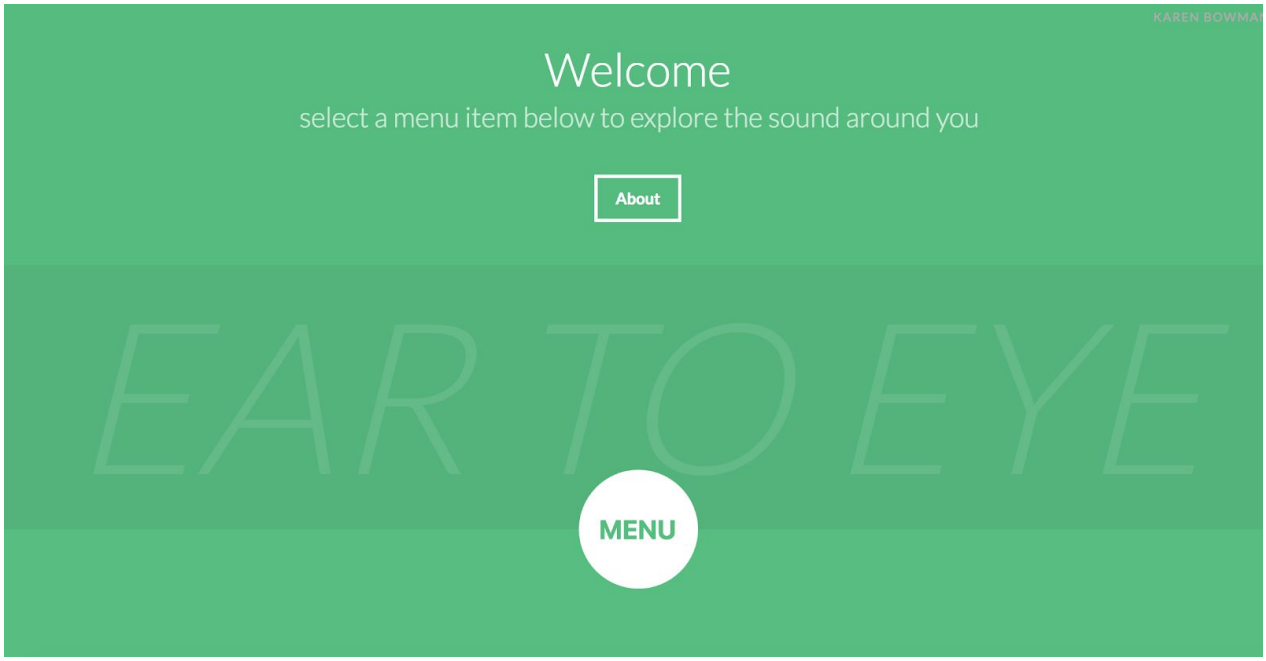
Menu



This menu is a circular menu and uses css animations to create dynamic interaction. The menu button when clicked expands into 5 option, radiating out from the first popout menu item. When clicked again, the menu will collapse. Each menu option leads to another page. The menu is

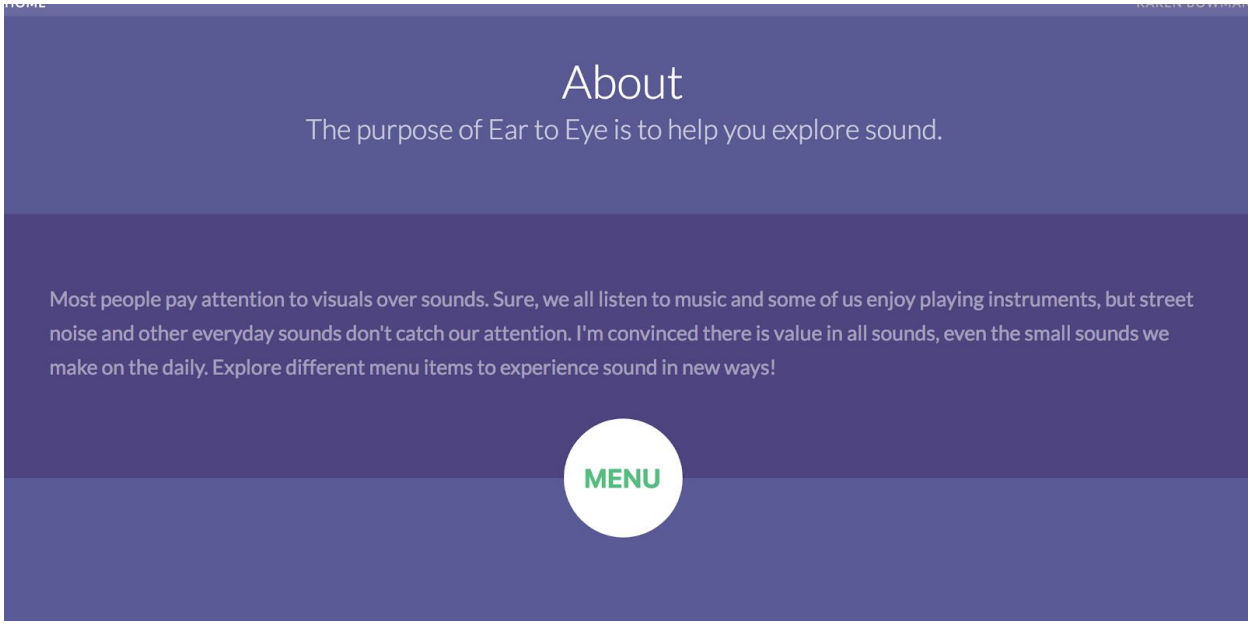
formatted at the bottom middle of the page and floats above other content using position: absolute.

Home Page



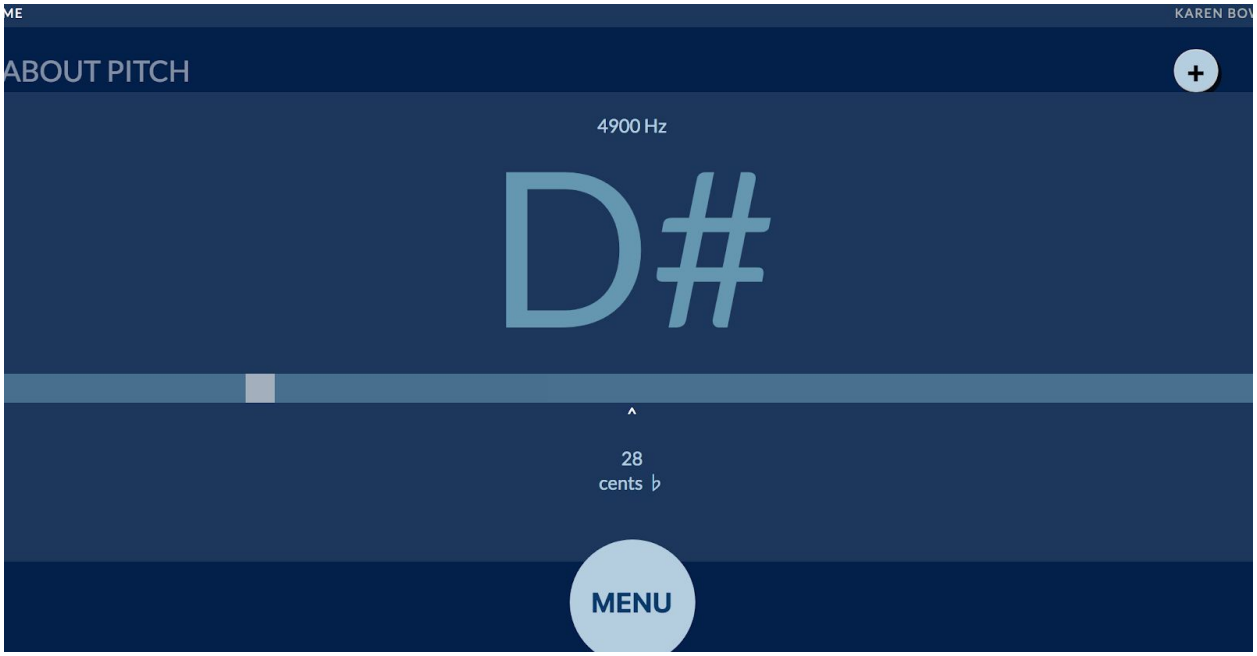
The home page is a simple page with a welcome banner, a faint background label “ear to eye,” button, and menu. The button leads to the about page and turns white when hovered over. The menu behaves as specified above.

About Page



The about page is another text-only page, with a menu, title, and paragraph content. It also has a back button that returns to the home page, for users more used to that style of navigation. The back button is highlighted on hover so it is easier to find and select.

Pitch Detection



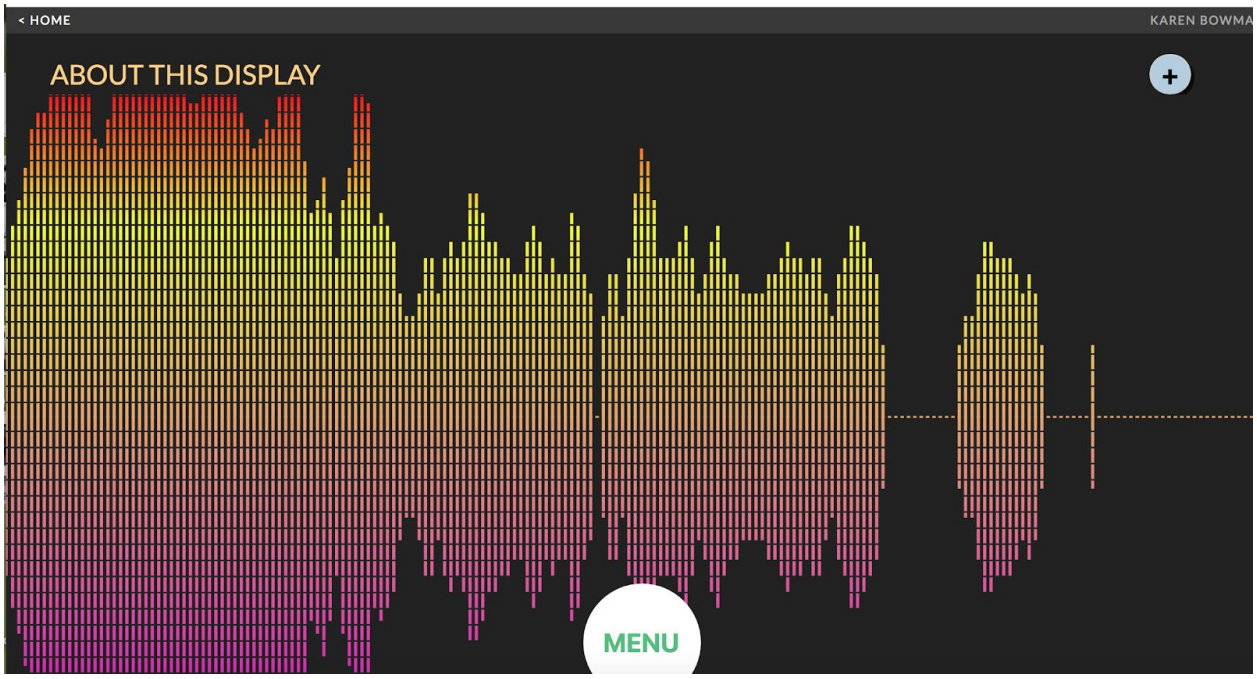
The pitch detection page prompts the user to make noise and then displays statistics after registering an adequately strong noise through the microphone. It then displays the calculated pitch, frequency in Hz, and how close the user's noise is to a perfect pitch/note. It also has a pop-up section that is displayed after pressing the “+” icon and hidden after pressing the “-” icon. The popup explains what the page content is.

The pitch is computed in pitch2.js, where the set of frequency data (an array) is analyzed to find the highest correlating value (not the average, which would be inaccurate!). This frequency is then compared the the known frequencies of specific pitches/notes. It is then categorized as a flat or sharp note depending on the accuracy. The function below takes the frequency and transforms it into a note.

```
function noteFromPitch( frequency ) {
  var noteNum = 12 * (Math.log( frequency / 440 )/Math.log(2) );
  return Math.round( noteNum ) + 69;
}
```

Lastly, the final information is displayed, and the “accuracy,” or counts from a perfect note, is displayed as a statistic and as a visual. The slider bar indicates if a note is flat or sharp, the square sliding accordingly, with the center being an exact note, neither flat nor sharp.

Bar Animation



The bar animation page displays the frequency data from real-time audio input as a symmetrical bar graph that “expands” when noise is made. It also has a popup section, a menu, and a back button.

The main animation uses an svg, or scalable vector graphic, and draws each bar as a path element of that svg, with a length corresponding to the size of that particular frequency (how common it is). After a Fourier transform, the frequency of certain sized “peaks” in sound waves is accessible as an array. Only the first 200 of 1024 values are displayed since those are the most frequent and the most meaningful.

Through Javascript, each stroke is created and formatted in real time. The animation is updated by continuously requesting new animation frames, in a recursive format. The function

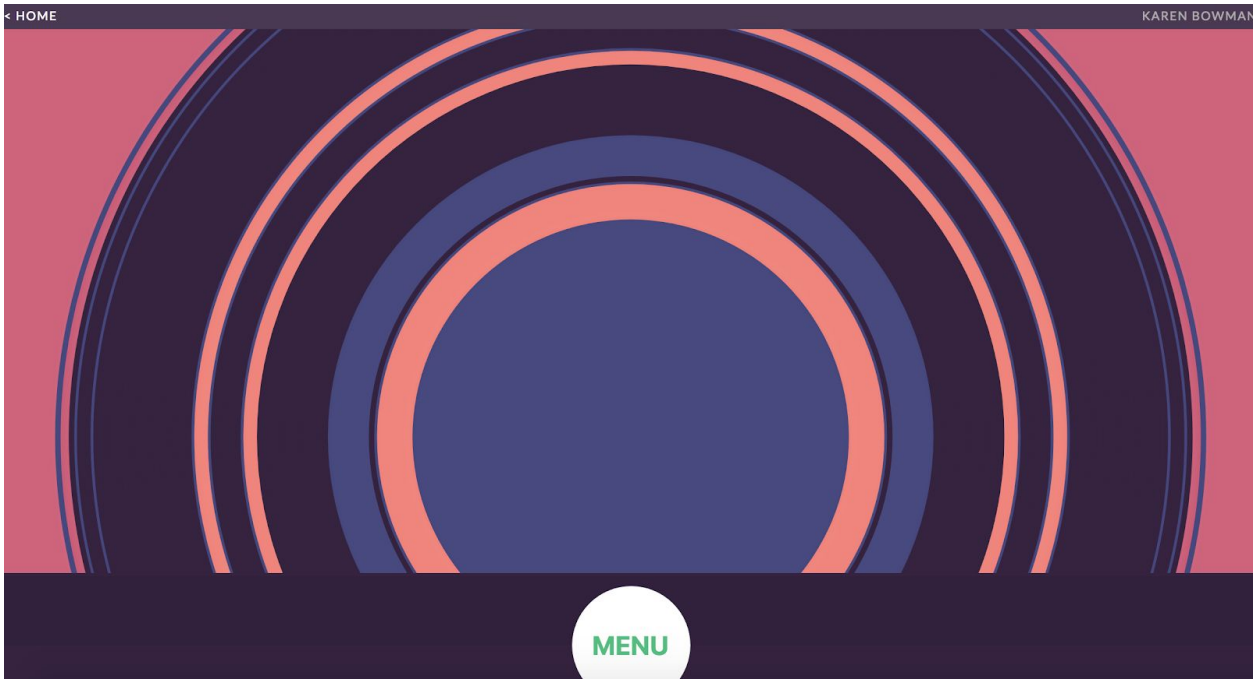
doDraw below updates each frame.

```
for (var i = 0 ; i < 400; i++) {
    path = document.createElementNS('http://www.w3.org/2000/svg', 'path');
    path.setAttribute('stroke-dasharray', '4,1');
    mask.appendChild(path);
}
var doDraw = function () {
    requestAnimationFrame(doDraw);
    analyser.getByteFrequencyData(frequencyArray);
    var adjustedLength;
    for (var i = 0 ; i < 200; i++) {
        adjustedLength = Math.floor(10*Math.sqrt(frequencyArray[i] - (Math.floor(frequ
        paths[i].setAttribute('d', 'M '+ (i) +',100 l 0,-' + adjustedLength);
        paths[200+i].setAttribute('d', 'M '+ (i) +',100 l 0,' + adjustedLength);
    }

}
doDraw();
```

The popup section expands dynamically (in a smooth fashion) when the user clicks “+” and can collapse if the user then presses the “-” icon. It contains a text explanation.

Bubble Animation



The bubble animation creates circular patterns based on the recent audio history; when noises of different loudness are created, it spawns circles of different colors. The result is usually the ring pattern displayed above, especially in music. There is also a back button and a menu navigator.

The circles are automatically spawned and added as children of the scalable vector graphic display when the average frequency of the overall sound (a measure of volume, or loudness) reaches a certain level. Depending on that level, the circle is assigned a color, lighter for lower volume, and darker for higher volume. The circle then expands at a slow rate until it reaches a size where it fills the screen. The code snippet below shows how the circles are modified and eventually removed.

```
for (var i=0; i<circles.length; i++){
  var current=circles[i].getAttribute('r');
  circleStore[i]=circleStore[i]+.99;
  circles[i].setAttribute('r', Math.floor(circleStore[i]));
  if (current > 500){
    visualizer.removeChild(circles[i]);
    circleStore.shift();
  }
```

Issues

Bubble Animation Glitches: The bubble in the bubble page sometimes overlap, and in general are spawned with a lagging pace, likely due to the number of circles drawn each frame (about 50-100).

Note Inconsistencies: The pitch detector changes very rapidly, which makes it hard for the user to understand how the pitch and noise are changing unless they are singing a very steady note.

Slider Animation: The slider ideally would be animated to smoothly transition between flat and sharp, but it is instead very jerky which makes it harder to use.

Div Animation: The div animation expands smoothly only on the first click- it does not animate successfully on later clicks unless the page is refreshed.

Conclusion

This approach worked adequately for the problem, but there were a few key issues. While the visuals and the exploratory structure were a good choice overall, the CSS animations did not align well with the data-heavy rapid fire functions needed to support them.

Given that the entire goal was to help people explore their own environment, it makes sense that there were different pages, each with an exploratory purpose. I think the color capabilities and small animated touches add a smoothness to the site that will be successful in interesting people and engaging them. Overall, the visuals have the general aesthetic appeal that I needed.

Despite the success, I ran into huge issues when it came to meshing data processing with css animation. I hoped that SVG rendering would be lightweight enough for the browser, but the animations are glitchy and more importantly, tend to lag behind the audio stream. It is hard to tell if that is solely the fault of visual rendering, or if the audio stream itself is not efficient in real time (aka takes a long time to fetch the audio data). Either way, there are issues in the way the circles expand in the bubble animation and in the way the bars respond to real time audio in the bar animation.

To improve this project, I would look into more efficient ways of rendering and displaying the animations for a more seamless look. I would also be interested in exploring more complex forms of animation, that take into account longer-term changes in tone in music, such as mood prediction. I have so far only explored making displays with straight frequency and volume data.

Appendix

Libraries / Interfaces

Audio Context Interface by Mozilla

Link: https://developer.mozilla.org/en-US/docs/Web/API/AudioContext

This was the library I used to fetch audio data. All other libraries had a great deal of difficulty getting the right audio permissions.

Through a new audioContext() object, I could access audio and create a media stream.

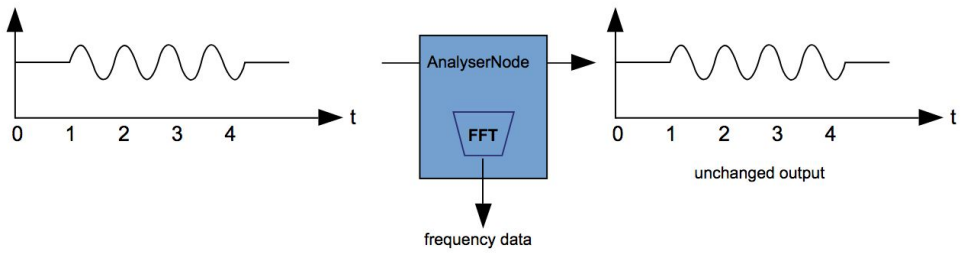
```
var audioContent = new AudioContext();
var audioStream = audioContent.createMediaStreamSource( stream );
```

AnalyzerNode by Mozilla

Link: https://developer.mozilla.org/en-US/docs/Web/API/AnalyserNode

Perfroms a Fourier transform on audio data, giving access to the frequency of different types of peaks.

```
analyser.getByteFrequencyData(frequencyArray);
```



Classie from bonzo

Link: https://github.com/ded/bonzo

Class helper functions, help with child management and inheritance of objects in the menu.

Tutorials

W3 Schools HTML SVG Tutorial

Link: https://www.w3schools.com/hTml/html5_svg.asp

This tutorial walks through using svg's and what you can draw with them, which I used to create the two animations.

Pitch Detection

https://github.com/cwilso/PitchDetect

Repository by cwilso on github was a great resource for a reliable method of calculating average pitch and frequency from an audioContext via a real-time audio stream.

Div Animation

Link: https://css-tricks.com/using-css-transitions-auto-dimensions/

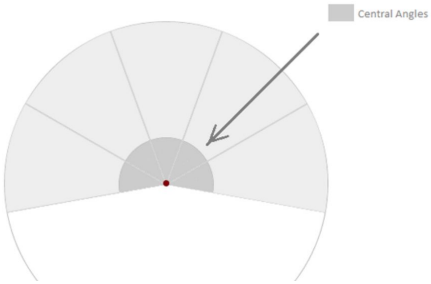
Explains how to animate dividers, as seen on the info expansion on the bubbles and pitch pages.

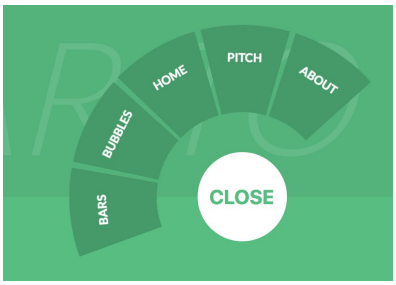
Circular Menu

Link: https://tympanus.net/codrops/2013/08/09/building-a-circular-navigation-with-css-transforms/

This tutorial explains how to create a dynamic expandable menu with a circular shape, and is the resource I used to create the menu for this site.

The pictures below show how a circular transformation and certain visible/not visible div elements can be used to create the style of menu on Ear to Eye.



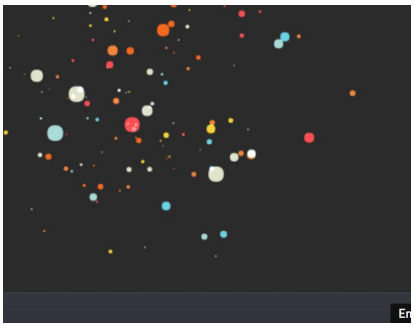


Content Inspiration

Flare Codepen by Justin Wilden

Link: https://codepen.io/soulwire/pen/foktm

This codepen has independent moving circles that spawn around a mouse. It is the inspiration for my bubbles animation since I liked the idea of independently moving circles. My final animation was much more basic.



Audio Visualizer from Codepen

Link: ???

I can't find the exact codepen, but it was a codepen that used path strokes in an svg to create a responsive bar graph with a gradient mask overlay, which I really liked the look of.

Color-Hex Palettes

Link: https://www.color-hex.com/color-palette/69389

Each page of my site has a unique color combination. This was the source of those colors; several users came up with the color palettes see in this project.