**Reliable, browser-based audio recording and processing**

The research group has a number of project ideas that involve the recording and processing of audio data. To make deployment of the systems practical, it must be possible to run them on the most common desktop computers and phones. Those target computing platforms are: iOS with the default version of Safari browser, Android with the default version of Chrome, Windows running Chrome (and ideally Microsoft Edge), macs running Safari (and ideally Chrome). This must be possible without requiring special plug-ins that are OS-dependent, and without requiring native code for any of the devices. On the phones, it is important that someone be able to click on a link from an email, that pulls up the default browser, and then use the website to record and process audio (after giving the browser permission to use audio, of course).

Recording in the browser has not been possible until very recently because Apple did not allow recording and the new HTML 5 capabilities are not fully or properly implemented in many browsers. It still seems like it is tricky to get it working reliably, especially on Apple devices, due to browser bugs and inconsistencies. But, it looks like it is possible, with some careful and clever programming. We think that because there are proof-of-concept examples that show recording from the various devices/browsers can be done. We also know it still cannot be done with some system/browser combinations.

**Project Step 1:**

The first step is take existing example code and modify it to record and playback audio.

We want to create JavaScript code that has/does the following:

* “Start recording” button – Press it and audio recording starts. Button name changes to “Stop recording”. If “Play audio” and “Save audio” buttons were showing, they disappear during recording.”
* When button “Stop recording” is pressed, audio stops recording and button changes back to “Start recording”. Two new buttons pop up, “Play audio” and “Save audio.”
* If “Play audio” is pressed, last recorded audio sample plays. While it plays, this button should change to “Stop playing audio”. If that is pressed, the audio should stop playing.
* If “Save audio” is pressed, it pops up a window that allows saving the audio file to the local computer’s drive.

The file could be saved/recorded as a wav file. However, but far better is to record/save the audio in the OPUS audio format, which is very compact for voice recording, which is what we care about (<https://en.wikipedia.org/wiki/Opus_(audio_format))>. It might be necessary to convert from a wav format to the opus format using javascript, depending upon what the HTML 5 defaults are. An important note is that we want all audio processing to happen locally, in the browser. We do not want to rely on servers to do the computation, because that will get expensive very quickly. Only processed audio will ultimately be shipped off to a server.

Step 1 is completed when there is a working demo that can be run on all the devices listed above, and that allows recording of files over and over without having to do anything funny to the browser (e.g., kill and restart it). Having this working on all the platforms above, with the smallest amount of well-organized code possible, would be outstanding.

**Project Step 2:**

The second step is to do some processing of the audio before it is saved or sent to a server.

Compute the duration of the audio file. Hopefully this is nearly instantaneous. Display the duration immediately after the audio is recorded (i.e., “Stop recording” is pressed). One way to do it is to change the name of the “Play audio” button to “Play 10.3s audio file”, where 10.3s is the length in seconds.

Next, add a checkbox to the sample code: “Clip audio” (with check of on/off). If this is selected, then immediately after the audio is recorded (i.e., “Stop recording” is pressed), the code should indicate that it is “Processing audio…” (disabling the appropriate buttons”) and process the audio for silence at the start and end, and clip off that silence. When this processingis done, the “Processing audio…” message should go away, and the appropriate buttons should be active again. When the audio is played, silence will have been clipped off the start and end.

Next, add another checkbox to “Adjust volume.” If this is selected, process the audio to adjust the audio amplitude, so that a particularly loud sample will be made software, and a soft sample will be made louder. There are a variety of techniques to do this we can talk about if you get this far.

Finally, if all of the above works, there is another checkbox to add that would enable additional processing that tries to remove clicks and pops from the audio.

**Project Step 3:**

The last step is to get the sample code working from Google App Engine. Alternatively, for those more interested Cloud computing than audio processing, this could become Step 2 (before audio processing).

We want to make this work from Google App Engine, so we can build systems that scale. The goal here is to setup a Google App Engine based webpage that serves the sample code. Add a new button that appears after recording, “Upload sample” and make that button send the processed data file to Google App Engine and save it. Create another webpage that shows all audio files that have been uploaded to Google, and allows clicking on each one to play the audio. Also, add a button next to each file (“Delete”) that will delete the file from the cloud storage.

**Resources:**

You will need to ask many questions. Please do!

Listed below, in no particular order, are some of the websites we have found that have led us to believe this is possible. Please review them all before starting coding. There is much sample code out there to use as a starting point. But, it is also important to understand that many examples work on one type of browser but will not work on Apple! (You can find some conversations about this below)

<https://www.webrtc-experiment.com/RecordRTC/simple-demos/audio-recording.html>

<https://github.com/muaz-khan/RecordRTC/issues/324#issuecomment-378994332>

<https://github.com/muaz-khan/RecordRTC/issues/392#issuecomment-373816392>

<https://github.com/muaz-khan/RecordRTC/issues/351>

<https://kaliatech.github.io/web-audio-recording-tests/dist/#/>

<https://danielstorey.github.io/WebAudioTrack/index.html>

<https://danielstorey.github.io/webrtc-audio-recording/>

<https://www.sitepoint.com/5-libraries-html5-audio-api/>

<https://developer.mozilla.org/en-US/Apps/Fundamentals/Audio_and_video_manipulation>

<https://webaudiodemos.appspot.com/AudioRecorder/index.html>

<https://www.createjs.com/demos/soundjs/webaudionodeinsertion>

<https://github.com/daaain/JSSoundRecorder>

<https://github.com/mido22/recordOpus>

<https://github.com/plucked/html5-audio-editor>

<https://github.com/unconed/ThreeAudio.js>

<http://www.danieldemmel.me/JSSoundRecorder/>

<https://stackoverflow.com/questions/16413063/html5-record-audio-to-file>

<https://mido22.github.io/MediaRecorder-sample/>

<https://blog.rillke.com/opusenc.js/>

<https://aws.amazon.com/blogs/machine-learning/capturing-voice-input-in-a-browser/>

If you find additional examples that are helpful, be sure to make a note of them and email them to me.