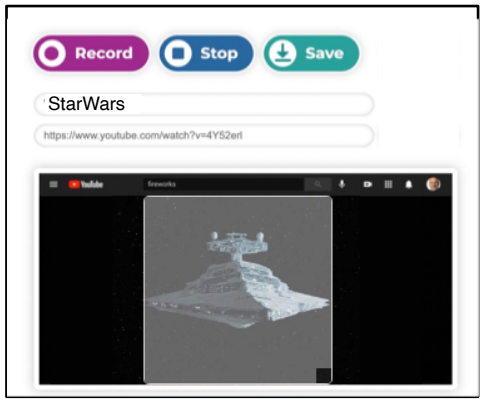
Reflections Recorder

Version: December 5, 2021

fcohen@starlingwatch.com, (408) 364-5508
Licensed under Creative Commons license



host://reflections/recorder

Javascript-based HTTP Post to host://reflections/recorder
Uploads starwars.mjpeg

Stores starswars.mjpeg to the file system

Uses ffmpeg to split the audio into its own file, including downsampling the audio to:

sample rate of 8000 kHz
16 bits
stereo format
WAV format

ffmpeg -i "$i" -c:a libfdk_aac -b:a 8k "starwars.m4a"

and down-converting video to:
240x240 pixels, uncompressed

ffmpeg -i "$i" -vf "fps=15,scale=240:240:flags=lanczos,crop=240:in_h:(in_w-240)/2:0" -q:v 9 "starwars.mjpeg"

Notes:
This page uses the WebRTC API to capture video
https://www.w3.org/TR/mediacapture-streams/

Example code to capture video is at:
https://simpl.info/mediarecorder/

Note: Chrome only allows this to work over HTTPS https://developer.mozilla.org/docs/Web/API/MediaDevices/getUserMedia#Browser_compatibility. Reflections will use a self-signed certificate for this part of the project. Frank will buy an SSL certificate in the near future.

The user will be given a prompt to share screen before recording, just like the one you see in video conferences before sharing the screen.

The whole video might need be uploaded then cropped in the server.

Target is Chrome/Mac and Windows

Creates TAR archive of video file, audio file, and show file (JSON format)

```
{
  "ReflectionsShow": {
    "title": "Reflections Recorder",
    "showname": "recorder",
      "version": "1",
         "events": {

     "1": {
       "type": "event",
       "name": "First sequence",
       "comment": "Plays audio and video",
       "sequence": {
          "playaudio": "StarWars.m4a",
          "playvideo": "StarWars.mjpeg"
           }
        }
     }
  }
}
```

http://starlingwatch.com/reflections/getshow

StarWars.tar

Record any browser content and play on a Reflections Seuss display.

Use case:

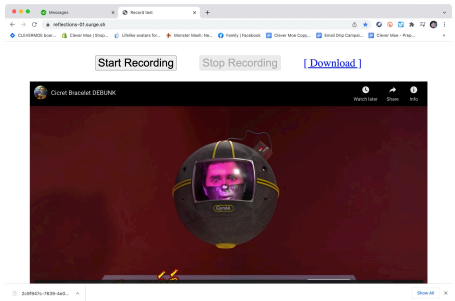
1) Browse https://starlingwatch.com/reflections/recorder
2) Enter a URL to any Web page. For example, https://www.youtube.com/watch?v=vLgsf8Pei6Q
3) A grey user changeable box appears in the capture area. Use the mouse to drag the box to determine the recording area. Use the mouse to click and drag the size icon (in the lower right corner of the capture area) to resize the capture area. The capture area is always square.
4) Click Record
5) Later click Stop
6) Click Save. The browser executes an HTTP Post to the Cloud City REST service to upload an MJPEG file (containing video and audio). REST service stores the MJPEG file to the server local file system.
7) REST service splits the audio into its own file (starwars.m4a) and video file (starwars.mjpeg).
8) Service creates a TAR file containing the audio, video, and show script.
9) Wrist watch makes HTTP REST getshow service requests. Cloud City replies with the newest TAR file (starwars.tar)
10) Wrist watch decodes the tar file and runs the audio and video files.

fcohen@starlingwatch.com, (408) 364-5508
Licensed under Creative Commons license

Example of WebRTC API use:

```
<html>
  <head>
    <title>Record test</title>
    <style>
      body {
 text-align: center;
}

button, a {
 display: inline-block;
 margin: 1em 1em;
 font-size: 2em;
 cursor: pointer;
}

iframe {
 border: 1px solid lightgray;
 width: 90%;
 height: 800px;
 background-color: #eee;
}

    </style>

  </head>
  <body>

      <button id="start">
       Start Recording
      </button>
      <button id="stop" disabled>
       Stop Recording
      </button>
      <a id="video-link" target="_blank">[ Download ]</a>
      <iframe src="https://www.youtube.com/embed/KbgvSi35n6o"></iframe>
      <script>
        const start = document.getElementById("start");
       const stop = document.getElementById("stop");
       const video = document.getElementById("video-link");
       let recorder, stream;

       async function startRecording() {
        stream = await navigator.mediaDevices.getDisplayMedia({
         video: { mediaSource: "screen" }, audio: true
        });

        let options = { mimeType: "video/webm\;codecs=h264" };
        recorder = new MediaRecorder(stream, options);

        const chunks = [];
        recorder.ondataavailable = e => chunks.push(e.data);
        recorder.onstop = e => {
         const completeBlob = new Blob(chunks, { type: chunks[0].type });
         video.setAttribute('href', URL.createObjectURL(completeBlob));
        };

        recorder.start();
       }

       start.addEventListener("click", () => {
        start.setAttribute("disabled", true);
        stop.removeAttribute("disabled");

        startRecording();
       });

       stop.addEventListener("click", () => {
        stop.setAttribute("disabled", true);
        start.removeAttribute("disabled");

        recorder.stop();
        stream.getVideoTracks()[0].stop();
       });

          </script>


  </body>
</html>
```

Demo at https://reflections-01.surge.sh/



1) You choose the language, server environment (PHP, Tomcat, Python, etc)
2) ou need to create and implement the XHTML.
3) Final deliverable is the running Web app on an AWS host that I provide, and a document explaining all steps needed to set-up and run the app.
4) You choose the recording format. For example, since the browser cannot record with MJPEG format, record in vp9 h264. and convert to MJPEG in the server.