```
<meta name="viewport" content="width=device-width, initial-scale=1">


<style>
body {font-family: Arial;}

.tab {
  overflow: hidden;
  border: 1px solid #ccc;
  background-color: #f1f1f1;
}


.tab button {
  background-color: inherit;
  float: left;
  border: none;
  outline: none;
  cursor: pointer;
  padding: 14px 16px;
  transition: 0.3s;
  font-size: 17px;
}

/* Change background color of buttons on hover */
.tab button:hover {
  background-color: #FFCC00;
}

/* Create an active/current tablink class */
.tab button.active {
  background-color: #FFFFCC;
}

/* Style the tab content */
.tabcontent {
  display: none;
  padding: 6px 12px;
  border: 1px solid #ccc;
  border-top: none;
}
</style>
```

```
<div class="tab">
  <button class="tablinks" onclick="openCity(event, 'Audio')">Audio</button>
  <button class="tablinks" onclick="openCity(event, 'Video')">Video</button>
</div>



<div id="Audio" class="tabcontent">


<!doctype html>
<html lang="en">
    <head>
        <title>Audio Recorder</title>
        <link rel="canonical" href="https://markjivko.com/tutorials/B3wWIsNHPk4/">
        <meta charset="UTF-8">
        <meta name="apple-mobile-web-app-capable" content="yes">
        <meta name="apple-mobile-web-app-status-bar-style" content="black-translucent">
        <meta name="viewport" content="initial-scale=1.0, width=device-width">
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <link rel="icon" type="image/ico" href="https://markjivko.com/favicon.ico">
        <style type="text/css">

.holder {
                background-color: #4c474c;
                background-image: -webkit-gradient(linear, left bottom, left top, from(#4c474c),
to(#141414));
                background-image: -o-linear-gradient(bottom, #4c474c 0%, #141414 100%);
                background-image: linear-gradient(0deg, #4c474c 0%, #141414 100%);
                border-radius: 50px;
            }
            [data-role="controls"] > button {
                margin: 50px auto;
                outline: none;
                display: block;
                border: none;
                background-color: #D9AFD9;
                background-image: -webkit-gradient(linear, left bottom, left top, from(#D9AFD9),
to(#97D9E1));
                background-image: -o-linear-gradient(bottom, #D9AFD9 0%, #97D9E1 100%);
                background-image: linear-gradient(0deg, #D9AFD9 0%, #97D9E1 100%);
                width: 100px;
                height: 100px;
                border-radius: 50%;
                text-indent: -1000em;
                cursor: pointer;
                -webkit-box-shadow: 0px 5px 5px 2px rgba(0,0,0,0.3) inset,
                   0px 0px 0px 30px #fff, 0px 0px 0px 35px #333;
                      box-shadow: 0px 5px 5px 2px rgba(0,0,0,0.3) inset,
                   0px 0px 0px 30px #fff, 0px 0px 0px 35px #333;
            }
            [data-role="controls"] > button:hover {
                background-color: #ee7bee;
                background-image: -webkit-gradient(linear, left bottom, left top, from(#ee7bee),
to(#6fe1f5));
                background-image: -o-linear-gradient(bottom, #ee7bee 0%, #6fe1f5 100%);
                background-image: linear-gradient(0deg, #ee7bee 0%, #6fe1f5 100%);
            }
            [data-role="controls"] > button[data-recording="true"] {
                background-color: #ff2038;
                background-image: -webkit-gradient(linear, left bottom, left top, from(#ff2038),
to(#b30003));
                background-image: -o-linear-gradient(bottom, #ff2038 0%, #b30003 100%);
                background-image: linear-gradient(0deg, #ff2038 0%, #b30003 100%);
            }
            [data-role="recordings"] > .row {
                width: auto;
                height: auto;
                padding: 20px;
            }
            [data-role="recordings"] > .row > audio {
                outline: none;
            }
            [data-role="recordings"] > .row > a {
                display: inline-block;
                text-align: center;
                font-size: 20px;
                line-height: 50px;
                vertical-align: middle;
                width: 50px;
                height: 50px;
                border-radius: 20px;
                color: #fff;
                font-weight: bold;
                text-decoration: underline;
                background-color: #0093E9;
                background-image: -webkit-gradient(linear, left bottom, left top, from(#0093E9),
to(#80D0C7));
                background-image: -o-linear-gradient(bottom, #0093E9 0%, #80D0C7 100%);
                background-image: linear-gradient(0deg, #0093E9 0%, #80D0C7 100%);
                float: right;
                margin-left: 20px;
                cursor: pointer;
            }
            [data-role="recordings"] > .row > a:hover {
                text-decoration: none;
            }
            [data-role="recordings"] > .row > a:active {
                background-image: -webkit-gradient(linear, left top, left bottom, from(#0093E9),
to(#80D0C7));
                background-image: -o-linear-gradient(top, #0093E9 0%, #80D0C7 100%);
                background-image: linear-gradient(180deg, #0093E9 0%, #80D0C7 100%);
            }
        </style>
        <script type="text/javascript" src="https://code.jquery.com/jquery.min.js"></script>
        <script src="https://markjivko.com/dist/recorder.js"></script>
        <script>
            jQuery(document).ready(function () {
                var $ = jQuery;
                var myRecorder = {
                   objects: {
                      context: null,
                      stream: null,
                      recorder: null
                   },
                   init: function () {
                      if (null === myRecorder.objects.context) {
                          myRecorder.objects.context = new (
                                  window.AudioContext || window.webkitAudioContext
                                  );
                      }
                   },
                   start: function () {
                      var options = {audio: true, video: false};
                      navigator.mediaDevices.getUserMedia(options).then(function (stream) {
                          myRecorder.objects.stream = stream;
                          myRecorder.objects.recorder = new Recorder(
                                  myRecorder.objects.context.createMediaStreamSource(stream),
                                  {numChannels: 1}
                          );
                          myRecorder.objects.recorder.record();
                      }).catch(function (err) {});
                   },
                   stop: function (listObject) {
                      if (null !== myRecorder.objects.stream) {
                          myRecorder.objects.stream.getAudioTracks()[0].stop();
                      }
                      if (null !== myRecorder.objects.recorder) {
                          myRecorder.objects.recorder.stop();

                          // Validate object
                          if (null !== listObject
                                  && 'object' === typeof listObject
                                  && listObject.length > 0) {
                              // Export the WAV file
                              myRecorder.objects.recorder.exportWAV(function (blob) {
                                  var url = (window.URL || window.webkitURL)
                                          .createObjectURL(blob);

                                  // Prepare the playback
                                  var audioObject = $('<audio controls></audio>')
                                          .attr('src', url);

                                  // Prepare the download link
                                  var downloadObject = $('<a>▼</a>')
                                          .attr('href', url)
                                          .attr('download', new Date().toUTCString() + '.wav');

                                  // Wrap everything in a row
                                  var holderObject = $('<div class="row"></div>')
                                          .append(audioObject)
                                          .append(downloadObject);

                                  // Append to the list
                                  listObject.append(holderObject);
                              });
                          }
                      }
                   }
                };

                // Prepare the recordings list
                var listObject = $('[data-role="recordings"]');
```

```
            // Prepare the record button
            $('[data-role="controls"] > button').click(function () {
                // Initialize the recorder
                myRecorder.init();

                // Get the button state
                var buttonState = !!$(this).attr('data-recording');

                // Toggle
                if (!buttonState) {
                    $(this).attr('data-recording', 'true');
                    myRecorder.start();
                } else {
                    $(this).attr('data-recording', '');
                    myRecorder.stop(listObject);
                }
            });
        });
    </script>
</head>
<body>
    <div class="holder">
        <div data-role="controls">
            <button>Record</button>
        </div>
        <div data-role="recordings"></div>
    </div>
</body>
</html>






</div>




<div id="Video" class="tabcontent">
```

```
<style>

.button {
  background-color: #66CC00;
  border: none;
  color: white;
  padding: 15px 32px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 16px;
  margin: 4px 2px;
  cursor: pointer;
  border-radius: 4px;
  transition-duration: 0.4s;
}

  .button:hover {
  border: 2px solid #66CC00;
  background-color: #ffffff;
  color: #333333;
}

  .button2 {
  background-color: #f44336;
  border: none;
  color: white;
  padding: 15px 32px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 16px;
  margin: 4px 2px;
  cursor: pointer;
  border-radius: 4px;
  transition-duration: 0.4s;
}

  .button2:hover {
  border: 2px solid #f44336;
  background-color: #ffffff;
  color: #333333;
}

p.capitalize {
  text-transform: capitalize;
}
</style>
    <meta charset="UTF-8">
    <title>MediaCapture and Streams API</title>
    <meta name="viewport" content="width=device-width">
            <link rel="stylesheet" href="main.css">




<center>
<h2 style="color:white">Record your practice.</h2>
</center>

    <center><video id="record" controls="" muted="muted"></video></center>
        <center>
</center>

        <center>

<button class="button" id="btnStart">START RECORDING ⏺️</button>  <button class="button
button2" id="btnStop">STOP RECORDING ⏹️ </button>

</center>





<center><video id="vid2" controls=""></video></center>
        <center>
<h4 style="color:white" class="capitalize">Watch your practice. Click to download.</h4>
</center>

    <script>

        let constraintObj = {
            audio: true,
            video: {
```

```
            facingMode: "user",
            width: { min: 250, ideal: 560, max: 560 },
            height: { min: 250, ideal: 315, max: 315 }
        }
    };
    // width: 100%, height: 315  -- preference only
    // facingMode: {exact: "user"}
    // facingMode: "environment"

    //handle older browsers that might implement getUserMedia in some way
    if (navigator.mediaDevices === undefined) {
        navigator.mediaDevices = {};
        navigator.mediaDevices.getUserMedia = function(constraintObj) {
            let getUserMedia = navigator.webkitGetUserMedia || navigator.mozGetUserMedia;
            if (!getUserMedia) {
                return Promise.reject(new Error('getUserMedia is not implemented in this
browser'));
            }
            return new Promise(function(resolve, reject) {
                getUserMedia.call(navigator, constraintObj, resolve, reject);
            });
        }
    }else{
        navigator.mediaDevices.enumerateDevices()
        .then(devices => {
            devices.forEach(device=>{
                console.log(device.kind.toUpperCase(), device.label);
                //, device.deviceId
            })
        })
        .catch(err=>{
            console.log(err.name, err.message);
        })
    }

    navigator.mediaDevices.getUserMedia(constraintObj)
    .then(function(mediaStreamObj) {
        //connect the media stream to the first video element
        let video = document.querySelector('video');
        if ("srcObject" in video) {
            video.srcObject = mediaStreamObj;
        } else {
            //old version
            video.src = window.URL.createObjectURL(mediaStreamObj);
        }

        video.onloadedmetadata = function(ev) {
            //show in the video element what is being captured by the webcam
            video.play();
        };

        //add listeners for saving video/audio
        let start = document.getElementById('btnStart');
        let stop = document.getElementById('btnStop');
        let vidSave = document.getElementById('vid2');
        let mediaRecorder = new MediaRecorder(mediaStreamObj);
        let chunks = [];

        start.addEventListener('click', (ev)=>{
            mediaRecorder.start();
            console.log(mediaRecorder.state);
        })
        stop.addEventListener('click', (ev)=>{
            mediaRecorder.stop();
            console.log(mediaRecorder.state);
        });
        mediaRecorder.ondataavailable = function(ev) {
            chunks.push(ev.data);
        }
        mediaRecorder.onstop = (ev)=>{
            let blob = new Blob(chunks, { 'type' : 'video/mp4;' });
            chunks = [];
            let videoURL = window.URL.createObjectURL(blob);
            vidSave.src = videoURL;
        }
    })
    .catch(function(err) {
        console.log(err.name, err.message);
    });

    /*********************************
    getUserMedia returns a Promise
    resolve - returns a MediaStream Object
    reject returns one of the following errors
    AbortError - generic unknown cause
    NotAllowedError (SecurityError) - user rejected permissions
    NotFoundError - missing media track
    NotReadableError - user permissions given but hardware/OS error
```

```
        OverconstrainedError - constraint video settings preventing
        TypeError - audio: false, video: false
        *********************************/
    </script>







</div>



<script>
function openCity(evt, cityName) {
  var i, tabcontent, tablinks;
  tabcontent = document.getElementsByClassName("tabcontent");
  for (i = 0; i < tabcontent.length; i++) {
    tabcontent[i].style.display = "none";
  }
  tablinks = document.getElementsByClassName("tablinks");
  for (i = 0; i < tablinks.length; i++) {
    tablinks[i].className = tablinks[i].className.replace(" active", "");
  }
  document.getElementById(cityName).style.display = "block";
  evt.currentTarget.className += " active";
}
</script>
```