# Audio Recording

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Audio Test</title>

<script>

</script>

</head>

<body>

<a id=*"download"*>Download</a>

<button id=*"stop"*>Stop</button>

<button id=*"start"* disabled>Start</button>

<br />

<div id=*"status"*>????</div>

</body>

<script>

**const** downloadLink = document.getElementById('download');

**const** stopButton = document.getElementById('stop');

**const** startButton = document.getElementById('start');

**const** statusText = document.getElementById('status');

//statusText.innerHTML = "waiting....";

//console.log (document.featurePolicy.allowedFeatures());

/// XXXXXXXXXXXXXXXXXXXXX

stopButton.addEventListener('click', **function**() {

stopRecording ();

startButton.disabled = **false**;

stopButton.disabled = **true**;

});

startButton.addEventListener('click', **function**() {

startRecording ();

startButton.disabled = **true**;

stopButton.disabled = **false**;

});

//==============================================================

// INIT

let shouldStop = **false**;

let stopped = **false**;

let recordedChunks = [];

//MediaDevice Stream Init

let thestream = **null**;

let media\_constraints = { video: **false**, audio: **true** };

//Init the Media Recorder

**const** media\_options = {mimeType : 'audio/webm'};

let mediaRecorder = **null**;

//==============================================================

**const** stopRecording = **function** () {

shouldStop = **true**;

status("clicked");

};

**const** initMediaRecorder = **function** () {

log("media device success...");

//const mediaRecorder = new MediaRecorder(stream, media\_options);

mediaRecorder = **new** MediaRecorder(thestream, media\_options);

mediaRecorder.start(1000); // maximale sample time

log("media recorder init/start done ...");

};

**const** ondataavailable = **function**(e) {

log("Handling on data available");

**if** (e.data.size > 0) {

recordedChunks.push(e.data);

console.log(e.data);

}

**if** (shouldStop === **true** && stopped === **false**) {

mediaRecorder.stop();

stopped = **true**;

status ("clicked... and ... stopped");

}

};

**const** onerror = **function**(e) {

log('Error: ' + e);

};

**const** onstart = **function**() {

log('Started, state = ' + mediaRecorder.state);

};

**const** onstop = **function**(e) {

downloadLink.href = URL

.createObjectURL(**new** Blob(recordedChunks));

downloadLink.download = 'acetest.wav';

log('media recorder stopped');

//statusText.innerHTML = "stopped";

status ("stopped");

};

**const** onlog = **function** (message) {

console.log(message);

};

**const** onstatus = **function** (html\_message) {

//####################################

//# TODO: change to on status change (outside)

statusText.innerHTML = html\_message;

};

**const** log = **function** (message) {

onlog (message);

};

**const** status = **function** (html\_message) {

onstatus (html\_message);

};

// MediaDevice successfully retrieved

**const** handleSuccess = **function**(stream) {

//######################################################

//# TODO: Create Stream should be done only once

thestream = stream

}

**const** reinitMediaRecorder = **function** () {

//=====================================================

// should be seperated

initMediaRecorder (thestream);

// Event handler

mediaRecorder.ondataavailable = ondataavailable;

mediaRecorder.onerror = onerror;

mediaRecorder.onstart = onstart;

mediaRecorder.onstop = onstop;

//console.log ("media recorder started...");

log ("media recorder started...");

};

**const** startRecording = **function** () {

status("waiting....");

log (document.featurePolicy.allowedFeatures());

recordedChunks = [];

shouldStop = **false**;

stopped = **false**;

//=====================================================

// If thestream is available, the next lines should not be called

**if** (thestream == **null**) {

navigator.mediaDevices.getUserMedia(media\_constraints)

.then(handleSuccess)

.**catch**(e => console.error(e));

}

reinitMediaRecorder ();

};

startRecording ();

</script>

<script>

</script>

</html>

# DataBlob Upload

<!DOCTYPE html>

<html lang=*"de"*>

<head>

<script src=*"http://code.jquery.com/jquery-1.11.0.min.js"*></script>

<title>JS - File Upload-1</title>

<script type=*"text/javascript"*>

<!--

//-->

**function** uploadSOund () {

**var** fd = **new** FormData();

fd.append('name', 'test.wav');

fd.append('data', soundBlob);

$.ajax(

{

type: 'POST',

url: '/upload.php',

data: fd,

processData: **false**,

contentType: **false**

}

).done(**function**(data) {

console.log(data);

}

);

};

// javascript function that uploads a blob to upload.php

**function** uploadBlob(){

// create a blob here for testing

**var** blob = **new** Blob(["i am a blob üüüüüääääääää"]);

console.log("create filereader...");

//var blob = yourAudioBlobCapturedFromWebAudioAPI;// for example

**var** reader = **new** FileReader();

// this function is triggered once a call to readAsDataURL returns

reader.onload = **function**(event){

console.log("this function is triggered once a call to readAsDataURL returns");

**var** fd = **new** FormData();

fd.append('name', 'test.txt');

//fd.append('file', event.target.result);

fd.append('datei', blob, 'test.xyz');

console.log(fd);

$.ajax({

type: 'POST',

url: 'http://audio.stichprobe.eu/upload.php',

data: fd,

dataType: "text",

crossDomain: **true**,

headers: {

"Access-Control-Allow-Origin" : "\*",

"Access-Control-Allow-Methods" : "GET,POST,PUT,DELETE,OPTIONS",

"Access-Control-Allow-Headers": "Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With"

},

processData: **false**,

//contentType: "Content-Type: audio/mpeg"

//contentType: 'text/html'

contentType: **false**

}).done(**function**(data) {

// print the output from the upload.php script

console.log(data);

});

};

//reader.onload = function() {

// console.log(reader.result);

//};

reader.onerror = **function**() {

console.log(reader.error);

};

console.log("trigger the read from the reader...");

// trigger the read from the reader...

reader.readAsDataURL(blob);

//reader.readAsText (blob);

console.log("... reader.readAsDataURL is triggered");

}

</script>

</head>

<body>

<form>

<button id=*"send"* type=*"button"*>Send</button>

</form>

</body>

<script type=*"text/javascript"*>

**const** sendButton = document.getElementById('send');

sendButton.addEventListener('click', **function**() {

uploadBlob ();

//statusText.innerHTML = "clicked";

});

</script>

</html>