main_gum.js

```
'use strict';

// Put variables in global scope to make them available to the browser console.
const constraints = window.constraints = {
  audio: true,
  video: true
};

function handleSuccess(stream) {
  const video = document.querySelector('video');
  const videoTracks = stream.getVideoTracks();
  console.log('Got stream with constraints:', constraints);
  console.log(`Using video device: ${videoTracks[0].label}`);
  window.stream = stream; // make variable available to browser console
  video.srcObject = stream;
}

function handleError(error) {
  if (error.name === 'ConstraintNotSatisfiedError') {
    let v = constraints.video;
    errorMsg(`Die Aufl&ouml;sung ${v.width.exact}x${v.height.exact} px wird von Ihrem Ge-
rät nicht unterst&uuml;tzt.`);
  } else if (error.name === 'PermissionDeniedError') {
    errorMsg('Der Zugriff auf die Kamera und das Mikrofon wurde nicht erlaubt. ' +
      'Damit dieser Test funktioniert, müssen Sie &uuml;ber Ihren Browser ' +
      'Kamera und Mikrofon frei schalten.');
  }
  errorMsg(`getUserMedia error: ${error.name}`, error);
}

function errorMsg(msg, error) {
  const errorElement = document.querySelector('#errorMsg');
  errorElement.innerHTML += `<p>${msg}</p>`;
  if (typeof error !== 'undefined') {
    console.error(error);
  }
}

async function init(e) {
  try {
    const stream = await navigator.mediaDevices.getUserMedia(constraints);
    handleSuccess(stream);
```

soundmeter_main.js

```
'use strict';

const instantMeter = document.querySelector('#instant meter');
const slowMeter = document.querySelector('#slow meter');
const clipMeter = document.querySelector('#clip meter');

const instantValueDisplay = document.querySelector('#instant .value');
const slowValueDisplay = document.querySelector('#slow .value');
const clipValueDisplay = document.querySelector('#clip .value');

try {
  window.AudioContext = window.AudioContext || window.webkitAudioContext;
  window.audioContext = new AudioContext();
} catch (e) {
  alert('Web Audio API not supported.');
}
//  Put variables in global scope to make them available to the browser console.
//- const constraints = window.constraints = {
//-  audio: true,
//-  video: false
};
function handleSuccess(stream) {
  // Put variables in global scope to make them available to browser console.
  window.stream = stream;
  const soundMeter = window.soundMeter = new SoundMeter(window.audioContext);
  soundMeter.connectToSource(stream, function(e) {
    if (e) {
      alert(e);
      return;
    }
    setInterval(() => {
      instantMeter.value = instantValueDisplay.innerText =
        soundMeter.instant.toFixed(2);
      slowMeter.value = slowValueDisplay.innerText =
        soundMeter.slow.toFixed(2);
      clipMeter.value = clipValueDisplay.innerText =
        soundMeter.clip;
    }, 200);
  });
}
function handleError(error) {
  console.log('navigator.getUserMedia error: ', error);
```

soundmeter.js

```
'use strict';

function SoundMeter(context) {
  this.context = context;
  this.instant = 0.0;
  this.slow = 0.0;
  this.clip = 0.0;
  this.script = context.createScriptProcessor(2048, 1, 1);
  const that = this;
  this.script.onaudioprocess = function(event) {
    const input = event.inputBuffer.getChannelData(0);
    let i;
    let sum = 0.0;
    let clipcount = 0;
    for (i = 0; i < input.length; ++i) {
      sum += input[i] * input[i];
      if (Math.abs(input[i]) > 0.99) {
        clipcount += 1;
      }
    }
    that.instant = Math.sqrt(sum / input.length);
    that.slow = 0.95 * that.slow + 0.05 * that.instant;
    that.clip = clipcount / input.length;
  };
}

SoundMeter.prototype.connectToSource = function(stream, callback) {
  console.log('SoundMeter connecting');
  try {
    this.mic = this.context.createMediaStreamSource(stream);
    this.mic.connect(this.script);
    // necessary to make sample run, but should not be.
    this.script.connect(this.context.destination);
    if (typeof callback !== 'undefined') {
      callback(null);
    }
  } catch (e) {
    console.error(e);
    if (typeof callback !== 'undefined') {
      callback(e);
    }
  }
```

```
  e.target.disabled = true;
 } catch (e) {
  handleError(e);
 }
}
document.querySelector('#showVideo').addEventListener('click', e => init(e));
```

```
}
navigator.mediaDevices.getUserMedia(constraints).then(handleSuccess).catch(handleEr-
ror);
```

```
};

SoundMeter.prototype.stop = function() {
 this.mic.disconnect();
 this.script.disconnect();
};
```