<!DOCTYPE html>-

<html>

<head>

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

<style>

body

{

margin: 0;

min-width: 250px;

}

/\*The following HTML coding is for MY TO DO LIST only which is downloaded from

https://www.w3schools.com/howto/howto\_js\_todolist.asp\*/

/\* Include the padding and border in an element's total width and height \*/

{

box-sizing: border-box;

}

/\* Remove margins and padding from the list \*/

ul

{

margin: 0;

padding: 0;

}

/\* Style the list items \*/

ul li

{

cursor: pointer;

position: relative;

padding: 12px 8px 12px 40px;

list-style-type: none;

background: #e7e7e7;

font-size: 18px;

transition: 0.2;

/\* make the list items unselectable \*/

-webkit-user-select: none;

-moz-user-select: none;

-ms-user-select: none;

user-select: none;

}

/\* Set all odd list items to a different color (zebra-stripes) \*/

ul li:nth-child(odd)

{

background: #f9f9f9;

}

/\* Darker background-color on hover \*/

ul li:hover

{

background: #ddd;

}

/\* When clicked on, add a background color and strike out text \*/

ul li.checked

{

background: #888;

color: #fff;

text-decoration: line-through;

}

/\* Add a "checked" mark when clicked on \*/

ul li.checked::before

{

content: '';

position: absolute;

border-color: #fff;

border-style: solid;

border-width: 0 2px 2px 0;

top: 10px;

left: 16px;

transform: rotate(45deg);

height: 15px;

width: 7px;

}

/\* Style the close button \*/

.close

{

position: absolute;

right: 0;

top: 0;

border: 1px solid green;

padding: 12px 16px 12px 16px;

}

.close:hover

{

background-color:#92a8d1;

color: white;

}

/\* Style the header \*/

.header

{

background-color: #00ffbf;

padding: 30px 40px;

color: white;

text-align: center;

}

/\* Clear floats after the header \*/

.header:after

{

content: "";

display: table;

clear: both;

}

/\* Style the input \*/

input

{

margin: 0;

border: none;

border-radius: 0;

width: 75%;

padding: 10px;

float: left;

font-size: 16px;

}

/\* Style the "Add" button \*/

.addBtn

{

padding: 10px;

width: 25%;

background:#e7e7e7;

color: #ffbf00;

float: left;

text-align: center;

font-size: 16px;

cursor: pointer;

transition: 0.3s;

border-radius: 0;

}

//when the user hover around the add button

.addBtn:hover

{

background-color:#e7e7e7;

}

/\*drop down box width\*/

#drop

{

width:100%;

}

</style>

</head>

<div id="myDIV" class="header">

<h2 style="margin:5px">My To Do List</h2>

<p align="right">

<span onclick="speechconvt()" class="addBtn" >Speech</span>

<span onclick="newElement()" class="addBtn">ADD</span>

</p>

<input type="text" id="myInput" placeholder="Title..." color:#e7e7e7 width="48">

<select id="drop" onmousedown="this.value='';" onchange="listboxFunction(this.value)" Color:#e7e7e7;>

<option value="Meet Mom">Meet Mom</option>

<option value="Go to library ">Go to library</option>

<option value="Hair cut">Hair cut</option>

<option value="Wash my car">Wash my car </option>

</select>

</div>

<ul id="myUL">

<li>Hit the gym</li>

<li class="checked">Pay bills</li>

<li class="a">Meet George</li>

<li>Buy eggs</li>

<li>Read a book</li>

<li>Organize office</li>

</ul>

<head>

<style>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0">

<meta name="apple-mobile-web-app-status-bar-style" content="black-translucent">

<input type="button" onclick="Speech to text()">Speech to text</input>

/\* The following HTML coding is for the compas and the Geolocation,which is downloaded from

https://developers.arcgis.com/javascript/3/jssamples/mobile\_compass.html\*/

body

{

padding: 5;

margin: 5;

overflow: hidden;

}

#map

{

height: 100%;

width: 100%;

position: centerAndZoom;

z-index: 20;

}

#compassHousing

{

margin-left: 5px;

margin-top: 5px;

background-color: #ddd;

border-style: black;

border-width: 1px;

border-radius: 62.5px;

padding: 2px;

position: absolute;

z-index: 2;

opacity: 0.77;

-moz-box-shadow: 2px 3px 10px 2px #333;

-webkit-box-shadow: 2px 3px 10px 2px #333;

box-shadow: 2px 3px 10px 2px #333;

}

#compassFace

{

z-index: -1;

position: absolute;

}

#compassNeedle

{

position: absolute;

-webkit-transition-property: -webkit-transform;

-webkit-transition-duration: .5s;

-webkit-transition-timing-function: ease-out;

}

@-webkit-keyframes pulse

{

0%

{

opacity: 1.0;

}

45%

{

opacity: 0.20;

}

100%

{

opacity: 1.0;

}

}

@-moz-keyframes pulse

{

0%

{

opacity: 1.0;

}

45%

{

opacity: 0.20;

}

100%

{

opacity: 1.0;

}

}

#map\_graphics\_layer

{

-webkit-animation-duration: 3s;

-webkit-animation-iteration-count: infinite;

-webkit-animation-name: pulse;

-moz-animation-duration: 3s;

-moz-animation-iteration-count: infinite;

-moz-animation-name: pulse;

}

/\* compass \*/

@media(orientation: landscape)

{

#compass

{

margin-top: 20px;

margin-left: 20px;

opacity: 0.85;

filter: alpha(opacity=85);

position: absolute;

z-index: 2;

}

}

@media(orientation: landscape)

{

#compass

{

margin-top: 20px;

margin-left: 20px;

opacity: 0.85;

filter: alpha(opacity=85);

position: absolute;

z-index: 2;

}

}

</style>

<link rel="stylesheet" href="https://js.arcgis.com/3.25/esri/css/esri.css" />

<script src="https://js.arcgis.com/3.25compact/">

</script>

<script>

/\*The following script is written for the compas and the Geo location\*/

require([

"esri/Color",

"dojo/dom",

"dojo/dom-geometry",

"dojo/has",

"dojo/on",

"dojo/parser",

"dojo/ready",

"dojo/window",

"esri/geometry/Point",

"esri/graphic",

"esri/map",

"esri/symbols/SimpleLineSymbol",

"esri/symbols/SimpleMarkerSymbol"

], function(Color, dom, domGeom, has, on, parser, ready, win, Point, Graphic, Map, SimpleLineSymbol, SimpleMarkerSymbol) {

/\*Following are the variables used for compass and Geo location\*/

var map;

var COMPASS\_SIZE = 100;

var pt;

var graphic;

var watchId;

var compassFaceRadius, compassFaceDiameter;

var needleAngle, needleWidth, needleLength, compassRing;

var renderingInterval = -1;

var currentHeading;

var hasCompass;

var compassHousing;

var containerX;

var containerY;

var compassNeedleContext;

//ready function is called as soon as page's Document Object Model (DOM) becomes safe to manipulate

ready(function()

{

//

parser.parse();

var supportsOrientationChange = "onorientationchange" in window,

orientationEvent = supportsOrientationChange ? "orientationchange" : "resize";

//The addEventListener() method is called upon orientation change.

window.addEventListener(orientationEvent, function ()

{

orientationChanged();

}, false);

map = new Map("map",

{

basemap: "gray",

center: [-117.708, 33.523],

zoom: 16,

slider: false

});

on(map, "load", mapLoadHandler);

loadCompass();

});

// The HTML5 geolocation API is used to get the user's current position.

function mapLoadHandler()

{

on(window, 'resize', map, map.resize);

// check if geolocaiton is supported

if (navigator.geolocation)

{

navigator.geolocation.getCurrentPosition(zoomToLocation, locationError);

// retrieve update about the current geographic location of the device

watchId = navigator.geolocation.watchPosition(showLocation, locationError);

}

else

{

alert("Browser doesn't support Geolocation. Visit http://caniuse.com to discover browser support for the Geolocation API.");

}

}

function zoomToLocation(location)

{

pt = esri.geometry.geographicToWebMercator(new Point(location.coords.longitude, location.coords.latitude));

addGraphic(pt);

map.centerAndZoom(pt, 17);

}

function showLocation(location)

{

pt = esri.geometry.geographicToWebMercator(new Point(location.coords.longitude, location.coords.latitude));

if (!graphic)

{

addGraphic(pt);

}

else

{

//move the graphic if it already exists

graphic.setGeometry(pt);

}

map.centerAt(pt);

}

function locationError(error)

{

//error occurred so stop watchPosition

if (navigator.geolocation)

{

navigator.geolocation.clearWatch(watchId);

}

switch (error.code)

{

case error.PERMISSION\_DENIED:

alert("Location not provided");

break;

case error.POSITION\_UNAVAILABLE:

alert("Current location not available");

break;

case error.TIMEOUT:

alert("Timeout");

break;

default:

alert("unknown error");

break;

}

}

// Add a pulsating graphic to the map

function addGraphic(pt)

{

var symbol = new SimpleMarkerSymbol(SimpleMarkerSymbol.STYLE\_CIRCLE, 12, new SimpleLineSymbol(SimpleLineSymbol.STYLE\_SOLID, new Color([210, 105, 30, 0.5]), 8), new Color([210, 105, 30, 0.9]));

graphic = new Graphic(pt, symbol);

map.graphics.add(graphic);

}

function loadCompass()

{

compassHousing = dom.byId("compassHousing");

// assign the compass housing dimensions

compassHousing.style.height = compassHousing.style.width = COMPASS\_SIZE + "px";

// return the absolute position of the compass housing

containerX = domGeom.position(compassHousing).x;

containerY = domGeom.position(compassHousing).y;

currentHeading = 0;

needleAngle = 0;

if (!buildCompassFace())

{

return;

}

drawCompassFace();

drawCompassNeedle();

hasWebkit();

}

// Creates the diameter of the compass face

// Creates the radius

function buildCompassFace()

{

// compass housing diameter and radius

compassFaceDiameter = COMPASS\_SIZE;

compassFaceRadius = compassFaceDiameter / 2;

// needle length

needleLength = compassFaceDiameter;

// needle width

needleWidth = needleLength / 10;

// tick marks

compassRing = compassFaceDiameter / 50;

return true;

}

var compassFaceContext;

// Draw the coppass face, text labels and font, and tick marks

function drawCompassFace()

{

var compassFaceCanvas = dom.byId("compassFace");

compassFaceCanvas.width = compassFaceCanvas.height = compassFaceDiameter;

compassFaceContext = compassFaceCanvas.getContext("2d");

compassFaceContext.clearRect(0, 0, compassFaceCanvas.width, compassFaceCanvas.height);

// draw the tick marks and center the compass ring

var xOffset, yOffset;

xOffset = yOffset = compassFaceCanvas.width / 2;

for (var i = 0; i < 360; ++i)

{

var x = (compassFaceRadius \* Math.cos(degToRad(i))) + xOffset;

var y = (compassFaceRadius \* Math.sin(degToRad(i))) + yOffset;

var x2 = ((compassFaceRadius - compassRing) \* Math.cos(degToRad(i))) + xOffset;

var y2 = ((compassFaceRadius - compassRing) \* Math.sin(degToRad(i))) + yOffset;

compassFaceContext.beginPath();

compassFaceContext.moveTo(x, y);

compassFaceContext.lineTo(x2, y2);

compassFaceContext.closePath();

compassFaceContext.stroke();

i = i + 4;

}

// The measureText method returns an object, with one attribute: width.

// The width attribute returns the width of the text, in pixels.

compassFaceContext.font = "10px Arial";

compassFaceContext.textAlign = "center";

var metrics = compassFaceContext.measureText('N');

compassFaceContext.fillText('N', compassFaceRadius, 15);

compassFaceContext.fillText('S', compassFaceRadius, compassFaceDiameter - 10);

compassFaceContext.fillText('E', (compassFaceRadius + (compassFaceRadius - metrics.width)), compassFaceRadius);

compassFaceContext.fillText('W', 10, compassFaceRadius);

}

// Draw the compass needle

function drawCompassNeedle()

{

var compassNeedle = dom.byId("compassNeedle");

compassNeedle.width = compassNeedle.height = compassFaceDiameter;

compassNeedle.style.left = Math.floor(compassFaceContext.width / 2) + "px";

compassNeedle.style.top = Math.floor(compassFaceContext.height / 2) + "px";

compassNeedleContext = compassNeedle.getContext("2d");

compassNeedleContext.translate(compassFaceRadius, compassFaceRadius);

compassNeedleContext.clearRect((compassNeedleContext.canvas.width / 2) \* -1, (compassNeedleContext.canvas.height / 2) \* -1, compassNeedleContext.canvas.width, compassNeedleContext.canvas.height);

// The first step to create a path is calling the beginPath method. Internally, paths are stored as a list of sub-paths

// (lines, arcs, etc) which together form a shape. Every time this method is called, the list is reset and we can start

// drawing new shapes.

// SOUTH

compassNeedleContext.beginPath();

compassNeedleContext.lineWidth = 1;

compassNeedleContext.moveTo(0, 5);

compassNeedleContext.lineTo(0, compassFaceRadius);

compassNeedleContext.stroke();

// circle around label

compassNeedleContext.beginPath();

compassNeedleContext.arc(0, compassFaceRadius - 15, 8, 0, 2 \* Math.PI, false);

compassNeedleContext.fillStyle = "#FFF";

compassNeedleContext.fill();

compassNeedleContext.lineWidth = 1;

compassNeedleContext.strokeStyle = "black";

compassNeedleContext.stroke();

// S

compassNeedleContext.beginPath();

compassNeedleContext.moveTo(0, 0);

compassNeedleContext.font = "normal 10px Verdana";

compassNeedleContext.fillStyle = "#000";

compassNeedleContext.textAlign = "center";

compassNeedleContext.fillText("S", 0, compassFaceRadius - 10);

// needle

compassNeedleContext.beginPath();

compassNeedleContext.fillStyle = "#000";

compassNeedleContext.moveTo(0, 0);

compassNeedleContext.lineTo(0, needleLength / 4);

compassNeedleContext.lineTo((needleWidth / 4) \* -1, 0);

compassNeedleContext.fill();

compassNeedleContext.beginPath();

compassNeedleContext.fillStyle = "#000";

compassNeedleContext.moveTo(0, 0);

compassNeedleContext.lineTo(0, needleLength / 4);

compassNeedleContext.lineTo(needleWidth / 4, 0);

compassNeedleContext.fill();

// NORTH

compassNeedleContext.beginPath();

compassNeedleContext.lineWidth = 1;

compassNeedleContext.moveTo(0, 0);

compassNeedleContext.lineTo(0, - compassFaceRadius);

compassNeedleContext.stroke();

// circle

compassNeedleContext.beginPath();

compassNeedleContext.arc(0, - (compassFaceRadius - 16), 8, 0, 2 \* Math.PI, false);

compassNeedleContext.fillStyle = "#FFF";

compassNeedleContext.fill();

compassNeedleContext.lineWidth = 1;

compassNeedleContext.strokeStyle = "black";

compassNeedleContext.stroke();

// N

compassNeedleContext.beginPath();

compassNeedleContext.moveTo(0, 0);

compassNeedleContext.font = "normal 10px Verdana";

compassNeedleContext.fillStyle = "#000";

compassNeedleContext.textAlign = "center";

compassNeedleContext.fillText("N", 0, - (compassFaceRadius - 20));

// needle

compassNeedleContext.beginPath();

compassNeedleContext.fillStyle = "#000";

compassNeedleContext.moveTo(0, 0);

compassNeedleContext.lineTo(0, (needleLength / 4) \* -1);

compassNeedleContext.lineTo((needleWidth / 4) \* -1, 0);

compassNeedleContext.fill();

compassNeedleContext.beginPath();

compassNeedleContext.fillStyle = "#000";

compassNeedleContext.moveTo(0, 0);

compassNeedleContext.lineTo(0, (needleLength / 4) \* -1);

compassNeedleContext.lineTo(needleWidth / 4, 0);

compassNeedleContext.fill();

// center pin color

compassNeedleContext.beginPath();

compassNeedleContext.arc(0, 0, 10, 0, 2 \* Math.PI, false);

compassNeedleContext.fillStyle = "rgb(255,255,255)";

compassNeedleContext.fill();

compassNeedleContext.lineWidth = 1;

compassNeedleContext.strokeStyle = "black";

compassNeedleContext.stroke();

compassNeedleContext.beginPath();

compassNeedleContext.moveTo(0, 0);

compassNeedleContext.arc(0, 0, (needleWidth / 4), 0, degToRad(360), false);

compassNeedleContext.fillStyle = "#000";

compassNeedleContext.fill();

}

var orientationHandle;

function orientationChangeHandler()

{

// An event handler for device orientation events sent to the window.

orientationHandle = on(window, "deviceorientation", onDeviceOrientationChange);

// The setInterval() method calls rotateNeedle at specified intervals (in milliseconds).

renderingInterval = setInterval(rotateNeedle, 100);

}

var compassTestHandle;

function hasWebkit()

{

if (has("ff") || has("ie") || has("opera"))

{

hasCompass = false;

orientationChangeHandler();

alert("Your browser does not support WebKit.");

} else if (window.DeviceOrientationEvent)

{

compassTestHandle = on(window, "deviceorientation", hasGyroscope);

} else

{

hasCompass = false;

orientationChangeHandler();

}

}

// Test if the device has a gyroscope.

// Instances of the DeviceOrientationEvent class are fired only when the device has a gyroscope and while the user is changing the orientation.

function hasGyroscope(event)

{

dojo.disconnect(compassTestHandle);

if (event.webkitCompassHeading !== undefined || event.alpha != null)

{

hasCompass = true;

}

else

{

hasCompass = false;

}

orientationChangeHandler();

}

// Rotate the needle based on the device's current heading

function rotateNeedle()

{

var multiplier = Math.floor(needleAngle / 360);

var adjustedNeedleAngle = needleAngle - (360 \* multiplier);

var delta = currentHeading - adjustedNeedleAngle;

if (Math.abs(delta) > 180)

{

if (delta < 0)

{

delta += 360;

} else

{

delta -= 360;

}

}

delta /= 5;

needleAngle = needleAngle + delta;

var updatedAngle = needleAngle - window.orientation;

// rotate the needle

dom.byId("compassNeedle").style.webkitTransform = "rotate(" + updatedAngle + "deg)";

}

function onDeviceOrientationChange(event)

{

var accuracy;

if (event.webkitCompassHeading !== undefined)

{

// Direction values are measured in degrees starting at due north and continuing clockwise around the compass.

// Thus, north is 0 degrees, east is 90 degrees, south is 180 degrees, and so on. A negative value indicates an invalid direction.

currentHeading = (360 - event.webkitCompassHeading);

accuracy = event.webkitCompassAccuracy;

} else if (event.alpha != null)

{

// alpha returns the rotation of the device around the Z axis; that is, the number of degrees by which the device is being twisted

// around the center of the screen

// (support for android)

currentHeading = (270 - event.alpha) \* -1;

accuracy = event.webkitCompassAccuracy;

}

if (accuracy < 11)

{

compassNeedleContext.fillStyle = "rgba(0, 205, 0, 0.9)";

} else if (accuracy >= 15 && accuracy < 25)

{

compassNeedleContext.fillStyle = "rgba(255, 255, 0, 0.9)";

} else if (accuracy > 24)

{

compassNeedleContext.fillStyle = "rgba(255, 0, 0, 0.9)";

}

compassNeedleContext.fill();

if (renderingInterval == -1)

{

rotateNeedle();

}

}

// Convert degrees to radians

function degToRad(deg)

{

return (deg \* Math.PI) / 180;

}

// Handle portrait and landscape mode orientation changes

function orientationChanged()

{

if (map)

{

map.reposition();

map.resize();

}

}

});

</script>

</head>

<body>

<article id="compassHousing">

<canvas id="compassFace"></canvas>

<canvas id="compassNeedle"></canvas>

</article>

<div id="map">

</div>

</body>

<script>

// Timer updates every 1000 milli seconds which is downloaded from W3 schools.

var myVar = setInterval(myTimer, 1000);

// Timer function downloaded from w3 schools to display current local time and the date.

function myTimer()

{

var d = new Date();

document.getElementById("demo").innerHTML = d.toLocaleTimeString();

document.getElementById("demo").innerHTML = d;

}

// list box function is my contribution

function listboxFunction(value)

{

//value in the `drop down box is assiged to the text box.

document.getElementById("myInput").value = value;

//selected drop down box value is converted to text

var txt = document.createTextNode(value);

//Then the text is appended to the list in the app

  li.appendChild(txt);

document.getElementById("myUL").appendChild(li);

}

// Create a "Delete" button and append it to each list item

//Delete button is my contribution

var myNodelist = document.getElementsByTagName("LI");

var loop;

//Delete button is added to all the list items in the list

for (loop = 0; loop < myNodelist.length; loop++)

{

var span = document.createElement("BUTTON");

var txt = document.createTextNode("Delete");

span.className = "close";

span.appendChild(txt);

myNodelist[loop].appendChild(span);

}

// Click on a Delete button to hide the current list item

var close = document.getElementsByClassName("close");

var loop;

for (loop = 0; loop < close.length; loop++)

{

close[loop].onclick = function()

{

var div = this.parentElement;

div.style.display = "none";

}

}

// Add a "checked" symbol when clicking on a list item

var list = document.querySelector('ul');

list.addEventListener('click', function(ev)

{

if (ev.target.tagName === 'LI')

{

ev.target.classList.toggle('checked');

}

}, false);

// my contribution for "Enter" event handler in the text box

var input = document.getElementById("myInput");

//when Enter key is pressed add add event listener is called.

input.addEventListener("keyup", function(event)

{

event.preventDefault();

//Key code 13 refers to Enter key,if enetr key is pressed newelement function is called.

if (event.keyCode === 13)

{

newElement();

}

});

// Create a new list item when clicking on the "Add" button

function newElement()

{

//list box element is created and assigned to li variable

var li = document.createElement("li");

//Text box element is assigned to input value variable

var inputValue = document.getElementById("myInput").value;

//input value is converted into text and appeneded to the list

var txt = document.createTextNode(inputValue);

li.appendChild(txt);

//If there is no value entered in the text box and user clicks the add button then the following alert is displayed.

if (inputValue === '')

{

alert("You must write something!");

}

//If value is entered in the text box and then appended to the unordered list(myUL)

else

{

   document.getElementById("myUL").appendChild(li);

}

//Delete button is added to the newly added list item.

var span = document.createElement("BUTTON");

var txt = document.createTextNode("Delete");

span.className = "close";

span.appendChild(txt);

li.appendChild(span);

for (loop = 0;loop< close.length; loop++)

{

close[loop].onclick = function()

{

var div = this.parentElement;

div.style.display = "none";

}

}

}

//The following coding is to convert speech into text that enables user to add in the list by voice command.which is partly downloaded from https://www.bouvet.no/bouvet-deler/utbrudd/getting-started-with-html5-speech-recognition-on-google-chrome and

//https://gist.github.com/szolotykh/e57476ff257bf1ae01b0.The original code has been restructured according to the need of this app by myself with my own contribution.

//WebkitspeechRecognition java script API is used for recording and transcribing the speech.

  var recognition = new webkitSpeechRecognition();

//speech is continous,so we set it ti TRUE

  recognition.continuous = true;

//If we want interim results then we set it as TRUE

  recognition.interimResults = true;

// Language is set to Australian English here

recognition.lang = 'en-AU'

//Below function is called once the user clicks the Speech button in the app

//The variable recognition is webkitspeech recognition's instance.

//getusermedia() API that allows the app to access user's micro phone,which is downoaded from https://developers.google.com/web/fundamentals/media/recording-audio/

navigator.mediaDevices.getUserMedia({ audio: true, video: false })

.then(handleSuccess);

  function speechconvt()

{

//users speech is transcripted and returned to the recognition varaiable.

    recognition.onresult = function(event)

{

      console.log(event);

      for(var loop=0; loop<event.results.length; loop++)

{

var inputValue= event.results[loop][0].transcript;

}

//users speech is been delivered to the text box in the app so that user can maually add it to the list to do.

document.getElementById("myInput").value=inputValue;

}

// Kick off the Speech to Text recognition process

   recognition.start();

// Ending after success or failed parsing

recognition.onend = function()

{

console.log("onend");

}

//creating a static web server for the web page.

var connect = require('connect');

var serveStatic = require('serve-static');

connect().use(serveStatic(\_\_dirname)).listen(8080);

}

// References

// https://developers.arcgis.com/javascript/3/jssamples/mobile\_compass.html

//https://www.w3schools.com/howto/howto\_js\_todolist.asp

//https://www.bouvet.no/bouvet-deler/utbrudd/getting-started-with-html5-speech-recognition-on-google-chrome

//https://gist.github.com/szolotykh/e57476ff257bf1ae01b0

//https://developers.google.com/web/fundamentals/media/recording-audio

</script>

<p id="demo"></p>

</body>