Script.js

```
/**
 * @license
 * Copyright 2018 Google LLC. All Rights Reserved.
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 *
=============================================================================
========
 */


const video = document.getElementById("webcam");
const liveView = document.getElementById("liveView");
const demosSection = document.getElementById("demos");
const enableWebcamButton = document.getElementById("webcamButton");

// Check if webcam access is supported.
function getUserMediaSupported() {
  return !!(navigator.mediaDevices && navigator.mediaDevices.getUserMedia);
}

// If webcam supported, add event listener to button for when user
// wants to activate it to call enableCam function which we will
// define in the next step.
if (getUserMediaSupported()) {
  enableWebcamButton.addEventListener("click", enableCam);
} else {
  console.warn("getUserMedia() is not supported by your browser");
}

// Enable the live webcam view and start classification.
function enableCam(event) {
  // Only continue if the COCO-SSD has finished loading.
  if (!model) {
    return;
  }
  // Hide the button once clicked.
  event.target.classList.add("removed");

  // getUsermedia parameters to force video but not audio.
  const constraints = {
    video: true,
  };

  // Activate the webcam stream.
  navigator.mediaDevices.getUserMedia(constraints).then(function (stream) {
    video.srcObject = stream;
    video.addEventListener("loadeddata", predictWebcam);
  });
}

// Placeholder function for next step.
function predictWebcam() {}

// Pretend model has loaded so we can try out the webcam code.
var model = true;
demosSection.classList.remove("invisible");
// Store the resulting model in the global scope of our app.
```

```
var model = undefined;


// Before we can use COCO-SSD class we must wait for it to finish
// loading. Machine Learning models can be large and take a moment
// to get everything needed to run.
// Note: cocoSsd is an external object loaded from our index.html
// script tag import so ignore any warning in Glitch.
cocoSsd.load().then(function (loadedModel) {
  model = loadedModel;
  // Show demo section now model is ready to use.
  demosSection.classList.remove("invisible");
});


var children = [];


function predictWebcam() {
  // Now let's start classifying a frame in the stream.
  model.detect(video).then(function (predictions) {
    // Remove any highlighting we did previous frame.
    for (let i = 0; i < children.length; i++) {
      liveView.removeChild(children[i]);
    }
    children.splice(0);
```

```
// Now lets loop through predictions and draw them to the live view if
// they have a high confidence score.
for (let n = 0; n < predictions.length; n++) {
  // If we are over 66% sure we are sure we classified it right, draw it!
  if (predictions[n].score > 0.66) {
    const p = document.createElement("p");
    p.innerText =
      predictions[n].class +
      " - with " +
      Math.round(parseFloat(predictions[n].score) * 100) +
      "% confidence.";
    p.style =
      "margin-left: " +
      predictions[n].bbox[0] +
      "px; margin-top: " +
      (predictions[n].bbox[1] - 10) +
      "px; width: " +
      (predictions[n].bbox[2] - 10) +
      "px; top: 0; left: 0;";

    const highlighter = document.createElement("div");
    highlighter.setAttribute("class", "highlighter");
```

```
        highlighter.style =
          "left: " +
          predictions[n].bbox[0] +
          "px; top: " +
          predictions[n].bbox[1] +
          "px; width: " +
          predictions[n].bbox[2] +
          "px; height: " +
          predictions[n].bbox[3] +
          "px;";

        liveView.appendChild(highlighter);
        liveView.appendChild(p);
        children.push(highlighter);
        children.push(p);
      }
    }

    // Call this function again to keep predicting when the browser is ready.
    window.requestAnimationFrame(predictWebcam);
  });
}
```

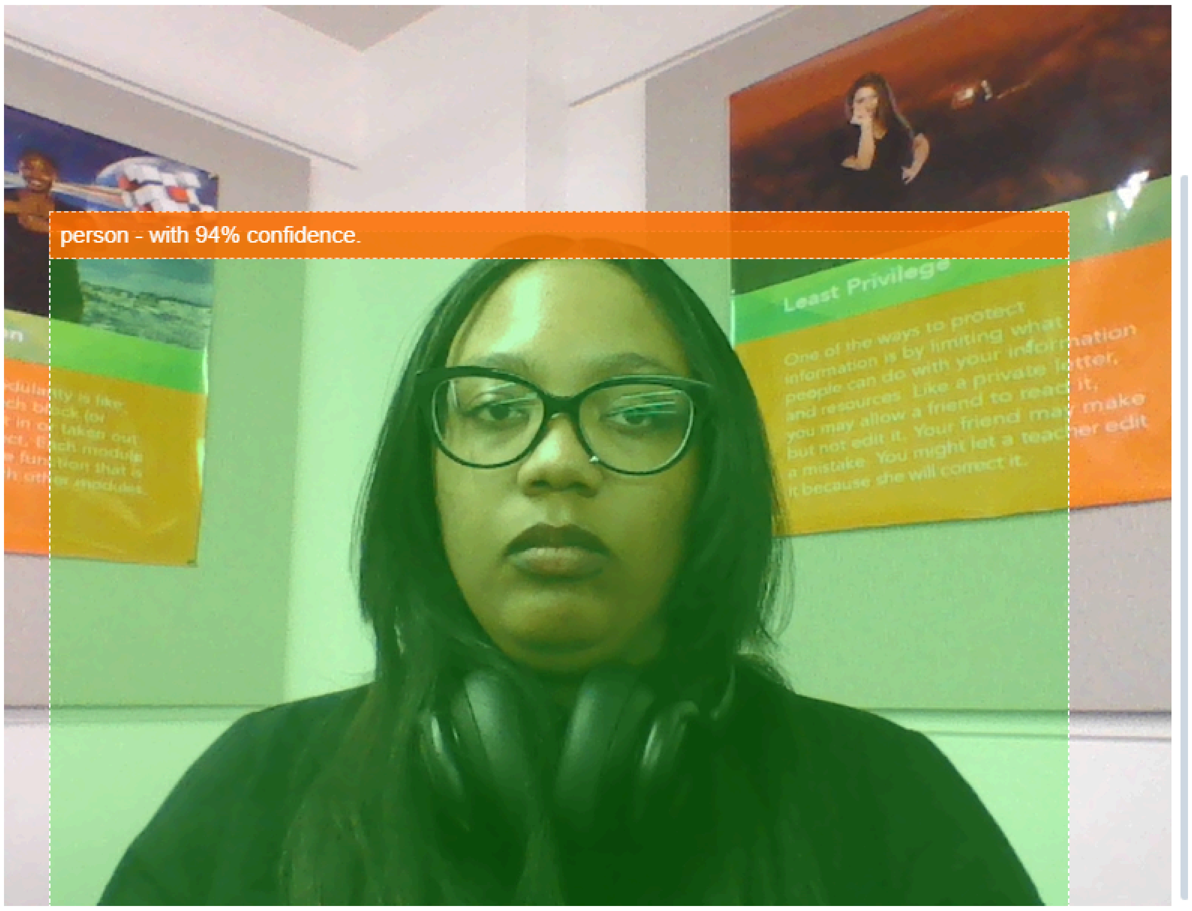
Index.html

```
<!DOCTYPE html>
<!-- Import TensorFlow.js library -->
<script
  src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs/dist/tf.min.js"
  type="text/javascript"
></script>
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>
      Multiple object detection using pre trained model in TensorFlow.js
    </title>
    <meta charset="utf-8" />
    <!-- Import the webpage's stylesheet -->
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <h1>Shirdora's Webcam</h1>
    <p>
      Wait for the model to load before clicking the button to enable the webcam
```

```
 - at which point it will become visible to use.
</p>

<section id="demos" class="invisible">
 <p>
   Hold some objects up close to your webcam to get a real-time
   classification! When ready click "enable webcam" below and accept access
   to the webcam when the browser asks (check the top left of your window)
 </p>

 <p>
   This is Shirdora Ashe's assignment for AIT-440 class at Capitol
   Technology University.
 </p>

 <div id="liveView" class="camView">
   <button id="webcamButton">Enable Webcam</button>
   <video id="webcam" autoplay muted width="640" height="480"></video>
 </div>
</section>

<!-- Import TensorFlow.js library -->
<script
```

```
      src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs/dist/tf.min.js"
      type="text/javascript"
    ></script>
    <!-- Load the coco-ssd model to use to recognize things in images -->
    <script src="https://cdn.jsdelivr.net/npm/@tensorflow-models/coco-ssd"></script>


    <!-- Import the page's JavaScript to do some stuff -->
    <script src="script.js" defer></script>
  </body>
</html>
```

ScreenShot


person - with 94% confidence.
Least Privilege
One of the ways to protect information is by limiting what people can do with your information and resources. Like a private letter, you may allow a friend to read it, but not edit it. Your friend may make a mistake. You might let a teacher edit it because she will correct it.