Emotion Recognition Computer Vision Model

Final Presentation by Nicholas Bornkamp

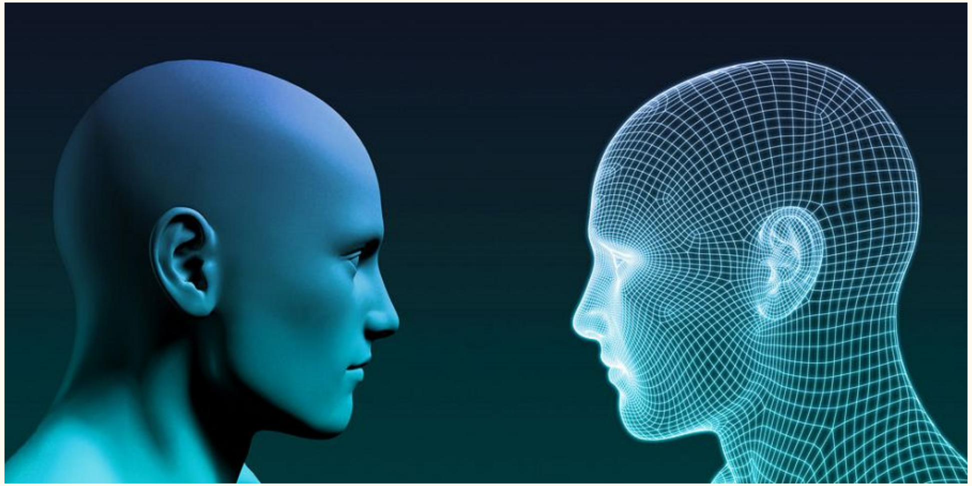
How are you feeling?

Emotion Detection Model

- Model that detects emotion
- This was an interesting project to work on due to learning how facial feature detection compares to pose detection.
- This project was inspired by the face recognition websites shown in class and by Lab5, although COCO-SSD was not used for this project.

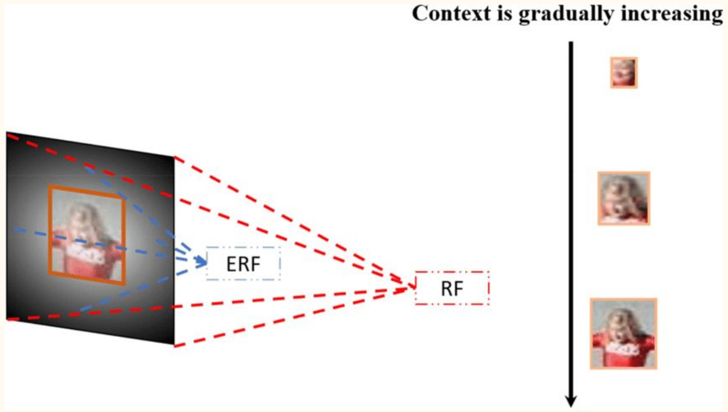
What’s the point?

- Facial detection is used for security purposes.
- Emotion detection practical uses:
    - Advertisements and focus groups
    - Artificial intelligence interaction with humans.



Algorithms and Techniques Used

- One model used by face-api.js model uses CNN based on MobileNet V1 with additional prediction layers.
- The one I used for my project is Tiny Face Detector, which has depth-wise separable convolutions instead of regular ones.
  - This makes the model faster but less accurate.

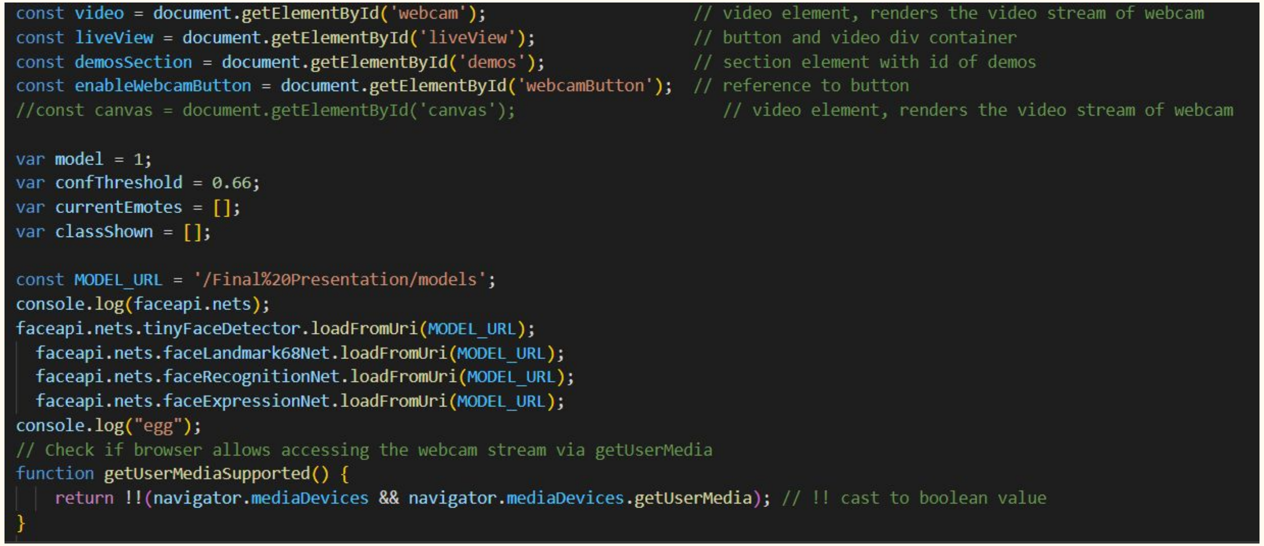


Imported JS Code and Models

- tfjs
  - Tensorflow.js, the one used in this class throughout the semester
- face-api.js
  - Downloaded from GitHub
- models
  - Folder also downloaded from GitHub
  - Each model consists of a shard file and a JSON weights file.
  - Trained on databases lie FGNET, Chalearn, Wiki, and IMDB

JavaScript code

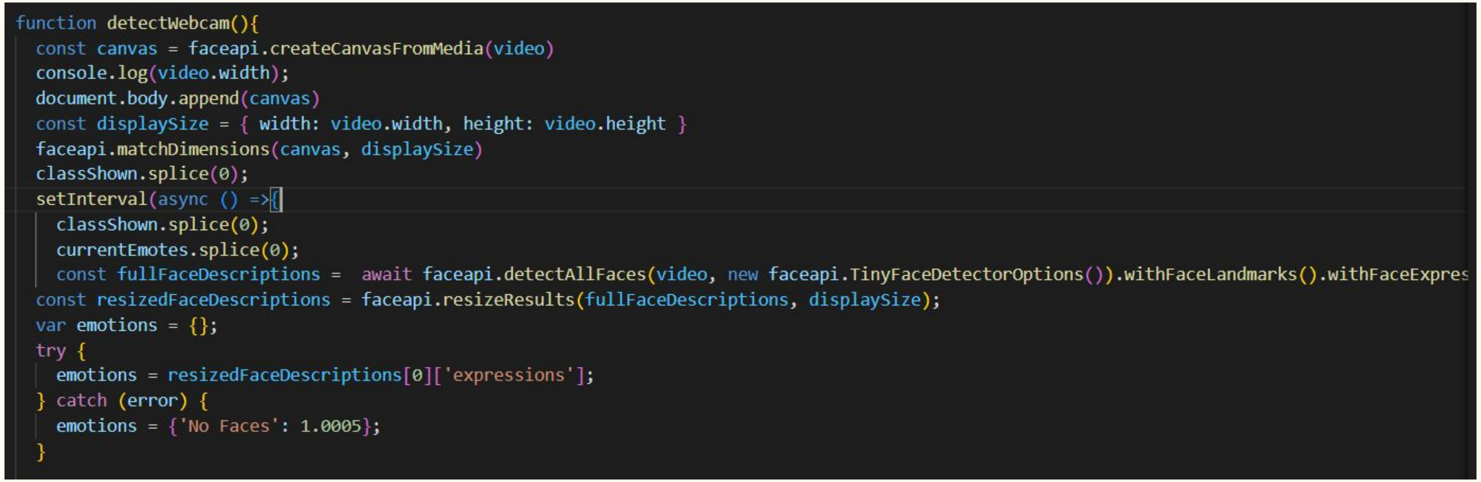
- Similar setup to Lab5
- faceapi.nets need to load the model from the models directory.
- Once the model is loaded, it logs “egg” to the console
  - Helped me debug so I kept it in.



```
const video = document.getElementById('webcam');                  // video element, renders the video stream of webcam
const liveView = document.getElementById('liveView');             // button and video div container
const demosSection = document.getElementById('demos');            // section element with id of demos
const enableWebcamButton = document.getElementById('webcamButton'); // reference to button
//const canvas = document.getElementById('canvas');                   // video element, renders the video stream of webcam

var model = 1;
var confThreshold = 0.66;
var currentEmotes = [];
var classShown = [];

const MODEL_URL = '/Final%20Presentation/models';
console.log(faceapi.nets);
faceapi.nets.tinyFaceDetector.loadFromUri(MODEL_URL);
  faceapi.nets.faceLandmark68Net.loadFromUri(MODEL_URL);
  faceapi.nets.faceRecognitionNet.loadFromUri(MODEL_URL);
  faceapi.nets.faceExpressionNet.loadFromUri(MODEL_URL);
console.log("egg");
// Check if browser allows accessing the webcam stream via getUserMedia
function getUserMediaSupported() {
    return !!(navigator.mediaDevices && navigator.mediaDevices.getUserMedia); // !! cast to boolean value
}
```

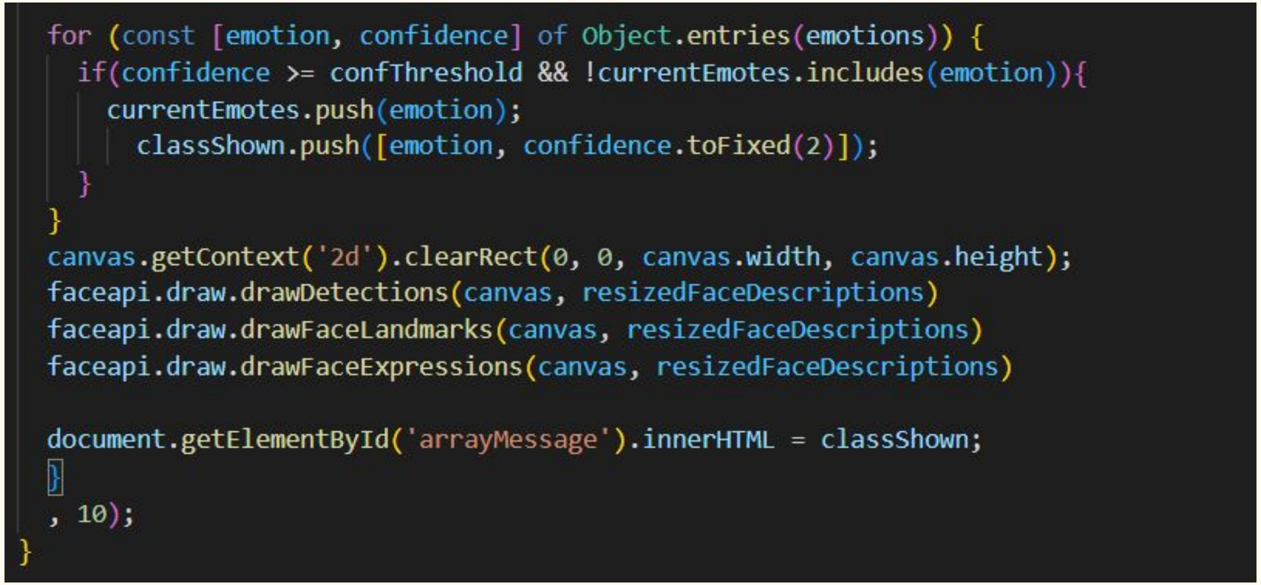
detectWebcam Function

- Creates a canvas from the video element.
- Creates an async function for the detection
- detectAllFaces generates the face detection output.
    - If no faces, then placeholder dictionary is used.



```
function detectWebcam(){
  const canvas = faceapi.createCanvasFromMedia(video)
  console.log(video.width);
  document.body.append(canvas)
  const displaySize = { width: video.width, height: video.height }
  faceapi.matchDimensions(canvas, displaySize)
  classShown.splice(0);
  setInterval(async () =>{
    classShown.splice(0);
    currentEmotes.splice(0);
    const fullFaceDescriptions =  await faceapi.detectAllFaces(video, new faceapi.TinyFaceDetectorOptions()).withFaceLandmarks().withFaceExpres
  const resizedFaceDescriptions = faceapi.resizeResults(fullFaceDescriptions, displaySize);
  var emotions = {};
  try {
    emotions = resizedFaceDescriptions[0]['expressions'];
  } catch (error) {
    emotions = {'No Faces': 1.0005};
  }
```

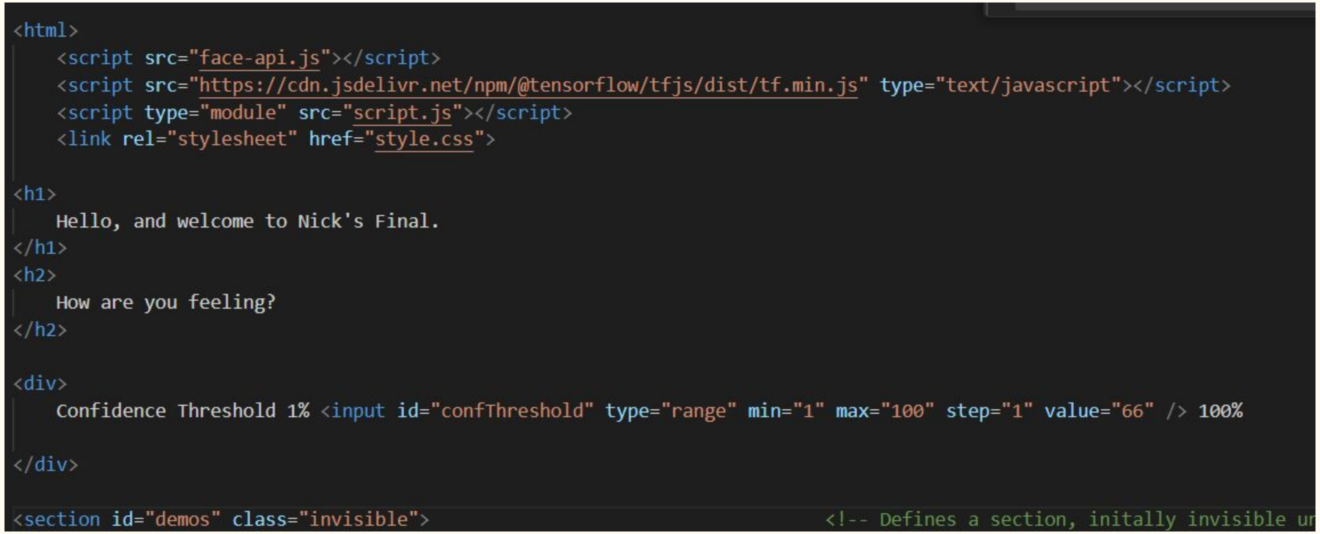
detectWebcam Function (cont)



```javascript
    for (const [emotion, confidence] of Object.entries(emotions)) {
      if(confidence >= confThreshold && !currentEmotes.includes(emotion)){
        currentEmotes.push(emotion);
          classShown.push([emotion, confidence.toFixed(2)]);
      }
    }
    canvas.getContext('2d').clearRect(0, 0, canvas.width, canvas.height);
    faceapi.draw.drawDetections(canvas, resizedFaceDescriptions)
    faceapi.draw.drawFaceLandmarks(canvas, resizedFaceDescriptions)
    faceapi.draw.drawFaceExpressions(canvas, resizedFaceDescriptions)

    document.getElementById('arrayMessage').innerHTML = classShown;
    }
    , 10);
  }
```

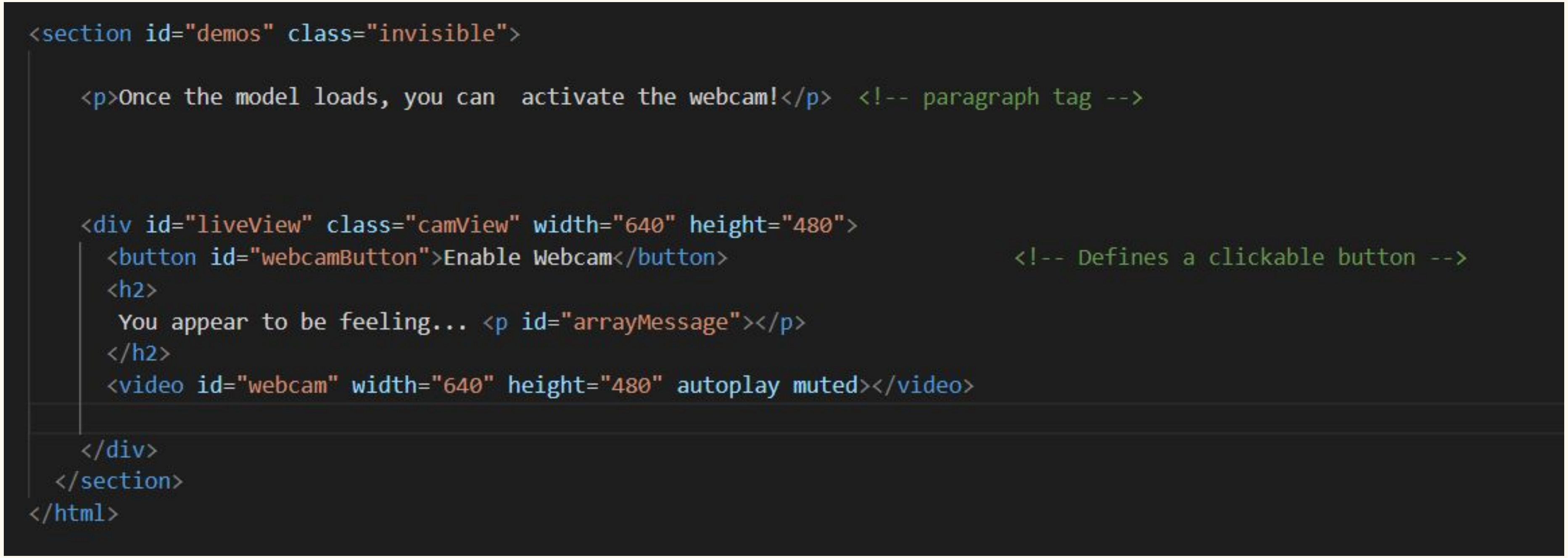
HTML Code

- Loads the face-api.js script from the directory along with the css.
- Confidence threshold slider.



```html
<html>
    <script src="face-api.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs/dist/tf.min.js" type="text/javascript"></script>
    <script type="module" src="script.js"></script>
    <link rel="stylesheet" href="style.css">

<h1>
    Hello, and welcome to Nick's Final.
</h1>
<h2>
    How are you feeling?
</h2>

<div>
    Confidence Threshold 1% <input id="confThreshold" type="range" min="1" max="100" step="1" value="66" /> 100%

</div>

<section id="demos" class="invisible">                                    <!-- Defines a section, initally invisible ur
```

HTML code (cont)



```
<section id="demos" class="invisible">

    <p>Once the model loads, you can  activate the webcam!</p>  <!-- paragraph tag -->



    <div id="liveView" class="camView" width="640" height="480">
      <button id="webcamButton">Enable Webcam</button>                      <!-- Defines a clickable button -->
      <h2>
       You appear to be feeling... <p id="arrayMessage"></p>
      </h2>
      <video id="webcam" width="640" height="480" autoplay muted></video>

    </div>
  </section>
</html>
```

Live Demonstration

py -m http.server to get things started

Sources used:

https://itnext.io/face-api-js-javascript-api-for-face-recognition-in-the-browser-with-tensorflow-js-bcc2a6c4cf07

So, how do you feel about it?

Questions/Comments