Bouw zelf een PWA met gezichts-herkenning en spraakherkenning

Deel 3

Dit is een vervolg op de tweede tutorial over PWA (zie links onderaan dit artikel voor de voorgaande tutorials). Je kan deze tutorial ook volgen als je de tweede niet hebt gevolgd.

We gaan ons richten op enkele nieuwe Web API's, zoals:

- Face detection API, voor gezichtsherkenning in de browser **https://justadudewhohacks.github.io/face-api.js/docs/index.html**
- Web Speech API, om tekst in te kunnen spreken **https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API/Using_the_Web_Speech_API**

Deze API's voegen we toe aan onze bestaande PWA voor het maken van selfies. Met face detection voorspellen we je emotie, je geslacht en je leeftijd. Om een begeleidende tekst in te spreken ('Speech to text') voor je selfie, kan je eenvoudig de Speech API inzetten.

Experimental Web Platform features

Bovenstaande API's werken alleen als je 'Experimental Web Platform features' aangezet hebt in je Chrome browser (zie **afbeelding 1**) via de url: **chrome://flags**

Project Setup

Als uitgangspunt voor de tutorial, clone je de volgende Github repository:

```
git clone https://github.com/
petereijgermans11/progressive-web-app
```

Ga vervolgens in je terminal naar de directory:

```
cd pwa-article/pwa-app-native-
features-rendezvous-init
```

Installeer de dependencies middels: `npm i && npm start` en open de webapp op (zie **afbeelding 2**):

```
http://localhost:8080
```

Public url voor je mobiel

Er zijn veel manieren om toegang te krijgen tot onze **localhost:8080** vanaf een mobiel apparaat op afstand. Je kan hiervoor **ngrok** gebruiken. Zie**: https://ngrok.com/**

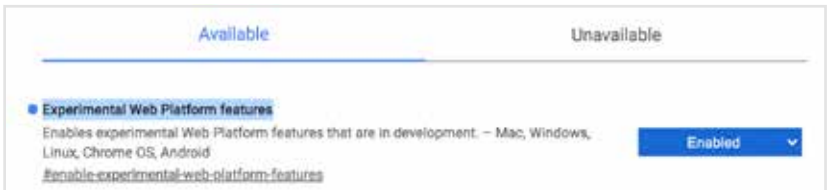
Installeer ngrok middels:`npm install -g ngrok`

En run het volgende commando in je terminal: `ngrok http 8080`. Dit commando genereert een public url voor je. Browse vervolgens naar de gegenereerde url op je mobiel in Chrome.

Gezichtsherkenning middels JavaScript

Gezichtsherkenning is een van de meest gebruikte toepassingen van Artificial Intelligence. Het gebruik van gezichtsherkenning is de afgelopen jaren toegenomen.

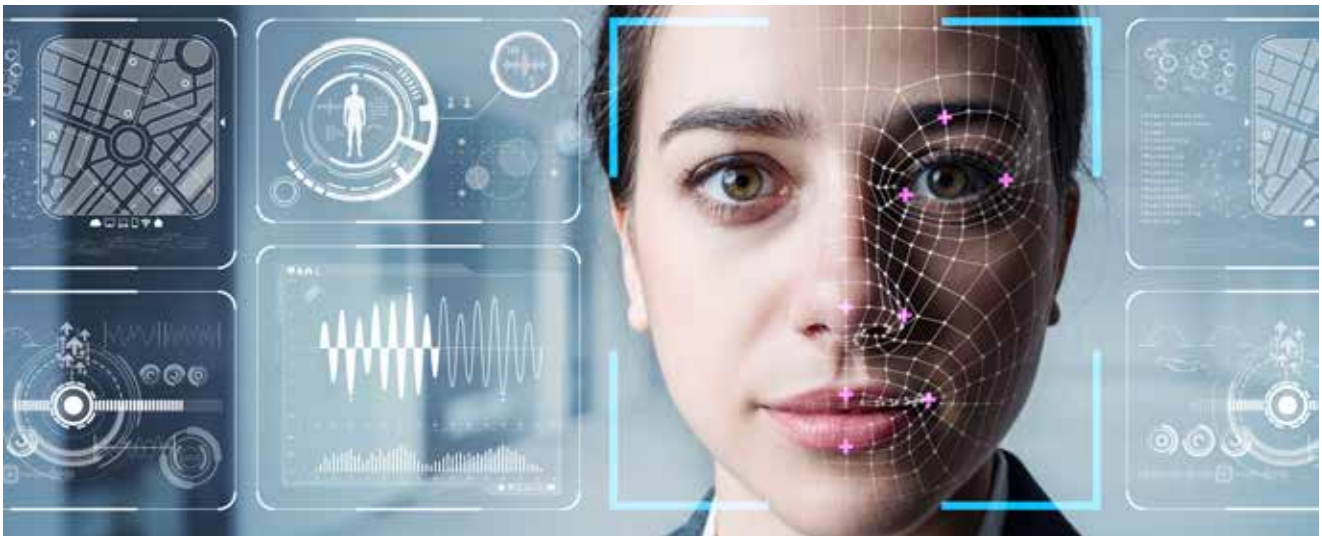
Peter Eijgermans is Frontend Developer en Codesmith bij Ordina JTech. Hij deelt graag zijn kennis met anderen middels workshops en presentaties.



Afbeelding 1.

Afbeelding 2.

Afbeelding 3.



In deze tutorial bouwen we de bestaande app uit voor gezichtsherkenning die zelfs in de browser werkt. Aan de hand van je gemaakte selfie voorspellen we je emotie, je geslacht en je leeftijd. We maken hier gebruik van de **Face-api.js**. Face-api.js bevat een JavaScript-API voor gezichtsherkenning in de browser. Deze API is geïmplementeerd bovenop de API van tensorflow.js. De output van deze app ziet er uit zoals in **afbeelding 3**.

De stappen om bovenstaande te implementeren zijn als volgt:

Stap 1: face-api
Face-api.js bevat een JavaScript-API voor gezichtsherkenning in de browser. Deze face-api staat reeds in de folder: public/src/lib.

Stap 2: modellen
Modellen zijn de getrainde gegevens die we zullen gebruiken om kenmerken van je selfie te detecteren. Deze modellen staan al klaar in de folder: public/src/models.

Stap 3: index.html
In het index.html-bestand importeren we:
- bestaande facedetection.css voor de styling (zie **Listing 1**);
- face-api.min.js, dit is de Face detection API voor het verwerken van de modelgegevens en het extraheren van de features (zie **Listing 3**);
- facedetection.js waar we onze logica zullen schrijven.

Importeer eerst de styling in de index.html

Plaats de code van listing 2 in het index.html-bestand, direct onder de tag:

```
<link rel="stylesheet" href="src/css/facedetection.css">
```

Listing 1.

```
<video id="player" autoplay></video>
<div class="container-faceDetection">
</div>
<canvas id="canvas" width="320px" height="240px"></canvas>
<div class="result-container">
   <div id="emotion">Emotion</div>
   <div id="gender">Gender</div>
   <div id="age">Age</div>
</div>
```

Listing 2.

```
<script src="src/lib/face-api.min.js"></script>
<script src="src/js/facedetection.js"></script>
```

Listing 3.

```
const startVideo = () => {
   navigator.mediaDevices.getUserMedia({video: {facingMode: 'user'},
   audio: false})
       .then(stream => {
           videoPlayer.srcObject = stream;
           videoPlayer.style.display = 'block';
           videoPlayer.setAttribute('autoplay', '');
           videoPlayer.setAttribute('muted', '');
           videoPlayer.setAttribute('playsinline', '');
       })
       .catch(error => {
           console.log(error);
       });
}
```

Listing 4.

```
Promise.all([
   faceapi.nets.tinyFaceDetector.loadFromUri("/src/models"),
   faceapi.nets.faceLandmark68Net.loadFromUri("/src/models"),
   faceapi.nets.faceRecognitionNet.loadFromUri("/src/models"),
   faceapi.nets.faceExpressionNet.loadFromUri("/src/models"),
   faceapi.nets.ageGenderNet.loadFromUri("/src/models")
]).then(startVideo);
```

Listing 5.

We maken gebruik van de bestaande video-tag om een selfie te kunnen maken. (zie Listing 2). En een 'result-container' voor het voorspellen van je emotie, je geslacht en je leeftijd.

Plaats de code van listing 3 onderaan de index.html, zodat we gebruik kunnen maken van de Face-detection API.

Stap 4: inlezen modellen in de PWA
Eerst maken we een aparte functie aan in feed.js voor het starten van de video-streaming (zie **Listing 4**). Verplaats de volgende bestaande code uit de initializeMedia-functie in een aparte functie onder de naam startVideo. Deze functie is verantwoordelijk voor de video-streaming.

In het bestaande feed.js gebruiken we Promise.all om de modellen voor de face-API asynchroon in te laden. Zodra deze modellen goed zijn ingeladen, roepen we de aangemaakte startVideo()-functie aan (zie Listing 5).

Plaats de code van listing 5 onderaan in de initializeMedia-functie.

Stap 5: Implementeer de facedetection.js
Hieronder staan de functies van de Face-detection API, die we in onze app gebruiken:

faceapi.detectSingleFace
faceapi.detectSingleFace maakt gebruik van de SSD Mobilenet V1 Face Detector. Je kunt de gezichtsherkenning specificeren door het videoPlayer object een opties object door te geven. Om meerdere gezichten te detecteren, vervang dan de functie detectSingleFace door detectAllFaces.

withFaceLandmarks
Deze functie wordt gebruikt voor het detecteren van 68 gezichtsoriëntatiepunten.

withFaceExpressions
Deze functie detecteert alle gezichten in een afbeelding en herkent gezichtsuitdrukkingen van elk gezicht en retourneert een array.

withAgeAndGender
Deze functie detecteert alle gezichten in een afbeelding, schat de leeftijd en herkent het geslacht van elk gezicht en retourneert een array.

Plaats onderstaande code in de bestaande file met de naam: facedetection.js, onder de reeds aanwezige code (zie **Listing 6**). Hierin worden bovenstaande functies aangeroepen, om de gezichtsherkenning uit te voeren.

```
videoPlayer.addEventListener("playing", () => {
 const canvasForFaceDetection = faceapi.createCanvasFromMedia(video
 Player);
 let containerForFaceDetection = document.querySelector(".container-
 faceDetection");
 containerForFaceDetection.append(canvasForFaceDetection);

 const displaySize = { width: 500, height: 500};
 faceapi.matchDimensions(canvasForFaceDetection, displaySize);

 setInterval(async () => {
   const detections = await faceapi
     .detectSingleFace(videoPlayer, new faceapi.
TinyFaceDetectorOptions())
     .withFaceLandmarks()
     .withFaceExpressions()
     .withAgeAndGender();

   const resizedDetections = faceapi.resizeResults(detections,
   displaySize);
     canvasForFaceDetection.getContext("2d").clearRect(0, 0, 500,
     500);

   faceapi.draw.drawDetections(canvasForFaceDetection,
   resizedDetections);
   faceapi.draw.drawFaceLandmarks(canvasForFaceDetection,
   resizedDetections);
   if (resizedDetections && Object.keys(resizedDetections).length > 0)
{
     const age = resizedDetections.age;
     const interpolatedAge = interpolateAgePredictions(age);
     const gender = resizedDetections.gender;
     const expressions = resizedDetections.expressions;
     const maxValue = Math.max(...Object.values(expressions));
     const emotion = Object.keys(expressions).filter(
       item => expressions[item] === maxValue
     );
     document.getElementById("age").innerText = `Age
     ${interpolatedAge}`;
     document.getElementById("gender").innerText = `Gender -
     ${gender}`;
     document.getElementById("emotion").innerText = `Emotion -
     ${emotion[0]}`;
   }
 }, 100);
});
```

Listing 6

```
<link rel="stylesheet" href="src/css/speech.css">
```

Listing 7.

Ten eerste wordt een playing event handler toegevoegd aan de videoPlayer, die reageert als de videocamera actief is. De variabele videoPlayer bevat het HTML-element <video>. Hierin worden je videotracks gerenderd. Vervolgens wordt er een canvasElement aangemaakt onder de naam canvasForFaceDetection. Deze wordt gebruikt voor de gezichtsherkenning. Deze canvasForFaceDetection wordt geplaatst in de containerfaceDetection.

De functie setInterval() herhaalt de functie faceapi.detectSingleFace per tijdsinterval van 100 milliseconden. Deze functie wordt asynchroon aangeroepen middels async/await en uiteindelijk worden de resultaten van de gezichtsherkenning weergegeven in de velden: emotion, gender en age.

Web Speech API

De interface die we gaan bouwen voor de Web Speech API ziet er uit zoals hieronder weergegeven (zie afbeelding 4). Zoals je ziet bevat het scherm een invoerveld om tekst in te voeren. Maar het is ook mogelijk om tekst in te spreken middels 'Speech to text'. Hiervoor moet je op de microfoon icoon klikken in het invoerveld.

Onder dit invoerveld kan je ook nog een gewenste taal uitkiezen via een select box.

De stappen om bovenstaande te implementeren zijn als volgt:

Stap 1: index.html
In het index.html-bestand importeren we:
- de bestaande speech.css voor de styling (zie **Listing 7**);
- speech.js waar we onze logica zullen schrijven voor de 'Speech to Text' feature (zie **Listing 9**).

Importeer eerst de styling in index.html
Plaats de onderstaande code in de index.html, direct onder de tag: `<form>` (zie **Listing 8**).
In het `<div id="info">` - sectie zijn de info teksten geplaatst die getoond kunnen worden, zodra gebruik gemaakt wordt van deze API.

Middels de startButton onclick-event, kan je de API opstarten en gebruiken. Tenslotte kan je met updateCountry onchange-event een gewenste taal selecteren via een select box.

Plaats de code van listing 9 onderaan index.html, zodat we gebruik kunnen maken van de Web Speech API.

Stap 2: Implementeer de Web Speech API
Deze code in de speech.js initialiseert de Web Speech Recognition API (zie **Listing 10**).

Eerst wordt gecontroleerd of de API van de 'webkitSpeechRecognition' beschikbaar is in het window object. Het window object representeert de browser window (Javascript is onderdeel van het window object).

```
<div id="info">
   <p id="info_start">Click on the microphone icon and begin
   speaking.</p>
   <p id="info_speak_now">Speak now.</p>
   <p id="info_no_speech">No speech was detected. You may need to
   adjust your
       <a href="//support.google.com/chrome/bin/answer.
       py?hl=en&amp;answer=1407892">
           microphone settings</a>.</p>
   <p id="info_no_microfoon" style="display:none">
       No microphone was found. Ensure that a microphone is installed
       and that
       <a href="//support.google.com/chrome/bin/answer.
       py?hl=en&amp;answer=1407892">
           microphone settings</a> are configured correctly.</p>
   <p id="info_allow">Click the "Allow" button above to enable your
   microphone.</p>
   <p id="info_denied">Permission to use microphone was denied.</p>
   <p id="info_blocked">Permission to use microphone is blocked. To
   change, go to chrome://settings/contentExceptions#media-stream</p>
   <p id="info_upgrade">Web Speech API is not supported by this
   browser. Upgrade to <a href="//www.google.com/chrome">Chrome</a>
   version 25 or later.</p>
</div>

<div class="right">
   <button id="start_button" onclick="startButton(event)">
       <img id="start_img" src="./src/images/mic.gif" alt="Start">
</
<div class="input-section mdl-textfield mdl-js-textfield mdl-
textfield--floating-label div_speech_to_text">
   <span id="title" contenteditable="true" class="final"></span>
   <span id="interim_span" class="interim"></span>
   <p>
</div>
<div class="center">
   <p>
   <div id="div_language">
       <select id="select_language" onchange="updateCountry()">
select>
       <select id="select_dialect"></select>
   </div>
</div>
```

Listin

```
<script sr  rc/js/speech.js"></script>
```

Listin

Als de 'webkitSpeechRecognition' beschikbaar is in het window object, dan wordt er een 'webkitSpeechRecognition' object aangemaakt via: recognition = new webkitSpeechRecognition();

We stellen vervolgens de volgende properties in van de Speech API: **recognition.continuous = true**. Deze property bepaalt of er continu resultaten worden geretourneerd voor elke recognition.
recognition.interimResults = true. Deze property bepaalt of de Speech recognition tussentijdse resultaten retourneert.

Event handlers

recognition.onstart(event)
Deze event handler wordt uitgevoerd wanneer de SpeechRecognition API is opgestart (zie **Listing 10**). Er wordt de volgende info tekst getoond: 'Speak now'. Vervolgens wordt een animatie getoond van een pulserende microfoon: mic-animate.gif.

recognition.onresult(event)
Deze event handler wordt uitgevoerd wanneer de SpeechRecognition API een resultaat retourneert. De SpeechRecognitionEvent 'results property" retourneert een 2-dimensionale SpeechRecognitionResultList. Middels de isFinal property wordt in een 'loop' bepaalt of een resultaat-tekst uit de lijst 'final' of 'interim' (tussentijds) is. Tevens wordt met de transcript property de resultaten omgezet naar een string.

recognition.onend(event)
Deze event handler wordt uitgevoerd wanneer de Speech Recognition beëindigd wordt.
Er wordt dan geen info tekst getoond, alleen de default icoon van een microfoon.

recognition.onerror(event)
Deze event handler vangt de errors af. Tevens wordt een bijpassende info tekst getoond bij de error melding.

Starten van de speech recognition

Voeg bovenin de reeds bestaande speech.js de volgende code toe, voor het starten van de Web Speech API, middels de startButton-event (zie **Listing 11**):

Deze code start de Web Speech Recognition API op middels recognition.start(). Deze start()-functie activeert een start-event en wordt afgehandeld in de event handler recognition.onstart() (zie **Listing 10**). Verder wordt de gekozen taal ingesteld met recognition.lang. En tenslotte wordt een pulserende microfoon image getoond.

Tenslotte

Na deze introductie kun je verder gaan met een uitgebreide tutorial die je kan vinden in: **https://github.com/petereijgermans11/progressive-web-app/tree/master/pwa-workshop.** In een volgend artikel ga ik verder in op andere API's zoals Web Streams API, Push API (voor het ontvangen van push notifications), Bluetooth API. ■

```
if ('webkitSpeechRecognition' in window) {
    start_button.style.display = 'inline-block';
    recognition = new webkitSpeechRecognition();
    recognition.continuous = true;
    recognition.interimResults = true;

    recognition.onstart = () => {
       recognizing = true;
       showInfo('info_speak_now');
       start_img.src = './src/images/mic-animate.gif';
    };

    recognition.onresult = (event) => {
       let interim_transcript =
       for (let i = event.resultIndex; i < event.results.length; ++i)
       {
          if (event.results[i].isFinal) {
             final_transcript += event.results[i][0].transcript;
          } else {
             interim_transcript += event.results[i][0].transcript;
          }
       }
       final_transcript = capitalize(final_transcript);
       title.innerHTML = linebreak(final_transcript);
       interim_span.innerHTML = linebreak(interim_transcript);
     };


    recognition.onerror = (event) => {
     ....
    };

    recognition.onend = () => {
      ...
    };
}
```

Listing 10.

```
const startButton = (event) => {
   if (recognizing) {
       recognition.stop();
       return;
   }
   final_transcript =
   recognition.lang = select_dialect.value;
   recognition.start();
   ignore_onend = false;
   title.innerHTML = '';
   interim_span.innerHTML = '';
   start_img.src = './src/images/mic-slash.gif';
   showInfo('info_allow');
   start_timestamp = event.timeStamp;
};
```

Listing 11.

LINKS:

1e tutorial: How to Build a Progressive Web App (PWA)
https://dzone.com/articles/how-to-build-a-progressive-web-app-pwa-with-javasc

2e tutorial: Enable Background Sync, Media Capture, and Geolocation APIs in Your PWA
https://dzone.com/articles/how-to-build-a-progressive-selfies-web-app