```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Welcome to Firebase Hosting</title>

    <!-- update the version number as needed -->
    <script defer
src="/__/firebase/9.22.2/firebase-app-compat.js"></script>
    <!-- include only the Firebase features as you need -->
    <script defer
src="/__/firebase/9.22.2/firebase-auth-compat.js"></script>
    <script defer
src="/__/firebase/9.22.2/firebase-database-compat.js"></script>
    <script defer
src="/__/firebase/9.22.2/firebase-firestore-compat.js"></script>
    <script defer
src="/__/firebase/9.22.2/firebase-functions-compat.js"></script>
    <script defer
src="/__/firebase/9.22.2/firebase-messaging-compat.js"></script>
    <script defer
src="/__/firebase/9.22.2/firebase-storage-compat.js"></script>
    <script defer
src="/__/firebase/9.22.2/firebase-analytics-compat.js"></script>
    <script defer
src="/__/firebase/9.22.2/firebase-remote-config-compat.js"></script>
    <script defer
src="/__/firebase/9.22.2/firebase-performance-compat.js"></script>
    <!--
      initialize the SDK after all desired features are loaded, set
useEmulator to false
      to avoid connecting the SDK to running emulators.
    -->
    <script defer src="/__/firebase/init.js?useEmulator=true"></script>

    <style media="screen">
```

```
      body { background: #ECEFF1; color: rgba(0,0,0,0.87); font-family:
Roboto, Helvetica, Arial, sans-serif; margin: 0; padding: 0; }
      #message { background: white; max-width: 360px; margin: 100px auto
16px; padding: 32px 24px; border-radius: 3px; }
      #message h2 { color: #ffa100; font-weight: bold; font-size: 16px;
margin: 0 0 8px; }
      #message h1 { font-size: 22px; font-weight: 300; color:
rgba(0,0,0,0.6); margin: 0 0 16px;}
      #message p { line-height: 140%; margin: 16px 0 24px; font-size:
14px; }
      #message a { display: block; text-align: center; background:
#039be5; text-transform: uppercase; text-decoration: none; color: white;
padding: 16px; border-radius: 4px; }
      #message, #message a { box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px
2px rgba(0,0,0,0.24); }
      #load { color: rgba(0,0,0,0.4); text-align: center; font-size: 13px;
}
      @media (max-width: 600px) {
        body, #message { margin-top: 0; background: white; box-shadow:
none; }
        body { border-top: 16px solid #ffa100; }
      }
    </style>
  </head>
  <body>
    <div id="message">
      <h2>Welcome</h2>
      <h1>Firebase Hosting Setup Complete</h1>
      <p>You're seeing this because you've successfully setup Firebase
Hosting. Now it's time to go build something extraordinary!</p>
      <a target="_blank"
href="https://firebase.google.com/docs/hosting/">Open Hosting
Documentation</a>
    </div>
    <p id="load">Firebase SDK Loading&hellip;</p>

    <div>Teachable Machine Image Model</div>
```

```
    <button type="button" onclick="init()">Start</button>
    <div id="webcam-container"></div>
    <div id="label-container"></div>

    <script>
      document.addEventListener('DOMContentLoaded', function() {
        const loadEl = document.querySelector('#load');
        // //
🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥
        // // The Firebase SDK is initialized and available here!
        //
        // firebase.auth().onAuthStateChanged(user => { });
        // firebase.database().ref('/path/to/ref').on('value', snapshot =>
{ });
        // firebase.firestore().doc('/foo/bar').get().then(() => { });
        // firebase.functions().httpsCallable('yourFunction')().then(() =>
{ });
        // firebase.messaging().requestPermission().then(() => { });
        // firebase.storage().ref('/path/to/ref').getDownloadURL().then(()
=> { });
        // firebase.analytics(); // call to activate
        // firebase.analytics().logEvent('tutorial_completed');
        // firebase.performance(); // call to activate
        //
        // //
🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥

        try {
          let app = firebase.app();
          let features = [
            'auth',
            'database',
            'firestore',
            'functions',
            'messaging',
            'storage',
            'analytics',
```

```
            'remoteConfig',
            'performance',
          ].filter(feature => typeof app[feature] === 'function');
          loadEl.textContent =  ` Firebase SDK loaded with
${features.join(', ')} ` ;
        } catch (e) {
          console.error(e);
          loadEl.textContent = 'Error loading the Firebase SDK, check the
console.';
        }
      });
    </script>

    <script
src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest/dist/tf.min.js">
</script>
    <script
src="https://cdn.jsdelivr.net/npm/@teachablemachine/image@latest/dist/teac
hablemachine-image.min.js"></script>
    <script type="text/javascript">
      // More API functions here:
      //
https://github.com/googlecreativelab/teachablemachine-community/tree/maste
r/libraries/image

      // the link to your model provided by Teachable Machine export panel
      const URL =
"https://teachablemachine.withgoogle.com/models/EMSASWR-F/"; //ให้นำแชร์ลิงก์
teachable machine ของตนเองมาวางแทน

      let model, webcam, labelContainer, maxPredictions;

      // Load the image model and setup the webcam
      async function init() {
          const modelURL = URL + "model.json";
          const metadataURL = URL + "metadata.json";
```

```
        // load the model and metadata
        // Refer to tmImage.loadFromFiles() in the API to support files
from a file picker
        // or files from your local hard drive
        // Note: the pose library adds "tmImage" object to your window
(window.tmImage)
        model = await tmImage.load(modelURL, metadataURL);
        maxPredictions = model.getTotalClasses();

        // Convenience function to setup a webcam
        const flip = true; // whether to flip the webcam
        webcam = new tmImage.Webcam(200, 200, flip); // width, height,
flip
        await webcam.setup(); // request access to the webcam
        await webcam.play();
        window.requestAnimationFrame(loop);

        // append elements to the DOM

document.getElementById("webcam-container").appendChild(webcam.canvas);
        labelContainer = document.getElementById("label-container");
        for (let i = 0; i < maxPredictions; i++) { // and class labels
            labelContainer.appendChild(document.createElement("div"));
        }
    }

    async function loop() {
        webcam.update(); // update the webcam frame
        await predict();
        window.requestAnimationFrame(loop);
    }

    // run the webcam image through the image model
    async function predict() {
        // predict can take in an image, video or canvas html element
        const prediction = await model.predict(webcam.canvas);
        for (let i = 0; i < maxPredictions; i++) {
```

```
            const classPrediction =
                prediction[i].className + ": " +
prediction[i].probability.toFixed(2);
            labelContainer.childNodes[i].innerHTML = classPrediction;
        }
    }
  </script>
 </body>
</html>
```