MimicMe tasks Analysis

Joe Minichino

Feature Points

The `face` object received back from the Affectiva API contains a property called `featurePoints` consisting of an array of Point objects (containing coordinates `x` and `y` for each `featurePoint`).

Displaying the `feautrePoints` is done by executing the following line in the `drawFeaturePoints` function:

```
ctx.arc(featurePoint.x, featurePoint.y, 2, 0, 2 * Math.PI);
```

This draws a circle centered on the feature point.

Dominant emoji displayed correctly

The dominant emoji is retrieved from the `face` object through the path `face.emojis.dominantEmojis`
I chose the first `featurePoint` as it represents the outer corner of the right eye (so the emoji lies close to the face itself but not on it).
This is accomplished by drawing to canvas with the following line in `drawEmoji`:

```
ctx.fillText(face.emojis.dominantEmoji, face.featurePoints[0].x,
face.featurePoints[0].y)
```

Show random emoji to mimic

Unicode codes for the 13 supported `emojis` is provided in the boilerplate code. I created a function `chooseEmoji` that chooses a random emoji among the 13 emojis, and updates the

UI accordingly. It always incresaes the total counter of emojis chosen for the purpose of the game.
This is the code of `chooseEmoji`:

```
const chooseEmoji = () => {
  tgt = emojis[Math.floor(Math.random() * 13)];
  console.log(`choosing emoji ${tgt}`)
  total += 1;
  setTargetEmoji(tgt);
  setScore(correct, total);
};
```

Note that `tgt`, `total` and `correct` are global variables.

Match with current player expression

Matching is performed by comparing the Unicide representation of `dominantEmoji` with the current game target represented by the `tgt` variable.
This is done with the `processImage` function which is called at each “"`onImageResultsSuccess`" event.

```
const processImage = (face) => {
  console.log(toUnicode(face.emojis.dominantEmoji), tgt);
  if (toUnicode(face.emojis.dominantEmoji) === tgt) {
    console.log('MATCH');
    correct += 1;
    resetInterval();
  }
};
```

Once a match is found, the interval that periodically chooses an emoji is reset, so a new emoji can be chosen and a new timeout set.

Reset and shows a new emoji

The program has four global variables:

- `tgt`: current target emoji
- `correct`: number of emojis matched
- `total`: total number of emojis displayed

- `intv`: setInterval ID to be used as a reference to clear and reset intervals

Game initialization and reset is performed by the `initGame` function:

```
const initGame = () => {
  total = -1;
  correct = 0;
  chooseEmoji();
  if (intv) {
    clearInterval(intv);
  }
  intv = setInterval(chooseEmoji, TIMEOUT);
};
```

The above function sets the total number of emojis to -1 (this is because chooseEmoji increases total by 1 immediately, correct matches to 0 and sets the global intv to the id of interval that randomly picks and emoji every 10 seconds.

Reset of the interval is preformed by `resetInterval`:

```
const resetInterval = () => {
  clearInterval(intv);
  chooseEmoji();
  intv = setInterval(chooseEmoji, TIMEOUT);
};
```

The above function clears the interval stored in intv, chooses and emoji and resets the interval. This function is called after a correct match, so the current interval is cleared, a new emoji picked and the next emoji switch set to happen in 10 seconds time.

Game Reset

Resetting the game is easily done by calling initGame again, as all values are set to their default (total -1, correct 0, interval cleared and reset, a new emoji picked).