

Face Recognition First Step

小林 統 *

2022 年 5 月 25 日

概要

顔認識ライブラリ clmtrackr.js を利用して、SNOW アプリを作ってみよう。

目次

1	はじめに	2
1.1	読み間違えないでね	2
1.2	注意	2
2	07-01.html ブラウザからカメラの情報を取得しよう	3
3	07-02.html 顔の特徴となる点を取得しよう	4
4	07-03.html 特徴となる点を数値で見よう	6
5	07-04.html 特徴となる点を元に SNOW にするよ	8
6	07-05.html 抽出された点から感情を導き出す関数を利用してみよう。	10
7	07-06.html 感情を元に画像を付け加えよう	13

* 帝京平成大学現代ライフ学部人間文化学科メディア文化コース

1 はじめに

1.1 読み間違えないでね

ソースコード 1 読み間違えないでね

```
1 数字: 0123456789
2 小文字:abcdefghijklmnopqrstuvwxyz
3 大文字:ABCDEFGHIJKLMNOPQRSTUVWXYZ
4
5 1:イチ
6 l:小文字のエル
7 i:小文字のアイ
8 |:ビックリマーク
9 |:バーティカルバー。Shift と ¥ を押したものの。
10
11 0:ゼロ
12 o:小文字のオー
13 O:大文字のオー
14
15 .:ピリオド
16 ,:コンマ
```

1.2 注意

- これから出てくるソースコードには、左に「行番号」と呼ばれる番号が出てくるけど、入力する必要はないからね。
- script タグの中で「//」で始まる文は、コメントで、プログラムは読み飛ばすよ。
- コピーできるところはコピーして効率よく入力して行こう
- 徐々に追加されていくから、量が多く見えるけど、平気だよ！
- 改行されていても、行番号が書かれていないところは、1 行だからね。表示上改行されて見えてるだけ

2 07-01.html ブラウザからカメラの情報を取得しよう

まずは、カメラの映像をホームページに写してみよう。

ソースコード 2 07-01.html

```
1 <!DOCTYPE html>
2 <html lang="ja">
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width,initial-scale=1">
6     <title>顔認識</title>
7     <style>
8       /* video 要素の上に canvas 要素をオーバーレイするための CSS */
9       /* コンテナ用の div について */
10      #container {
11        position: relative; /* 座標指定を相対値指定にする */
12      }
13      /* カメラ映像を流す video について */
14      #video {
15        transform: scaleX(-1); /* 左右反転させる */
16      }
17    </style>
18  </head>
19
20  <body>
21    <!-- video の上に canvas をオーバーレイするための div 要素 -->
22    <div id="container">
23      <!-- カメラ映像を流す video -->
24      <video id="video" width="400" height="300" autoplay></video>
25    </div>
26    <script>
27      // もろもろの準備
28      var video = document.getElementById("video"); // video 要素を取得
29      // getUserMedia によるカメラ映像の取得
30      var media = navigator.mediaDevices.getUserMedia({ // メディアデバイスを取得
31        video: {facingMode: "user"},
32        // カメラの映像を使う
33        audio: false
34        // マイクの音声は使わない
35      });
36      media.then((stream) => { // メディアデバイスが取得できたら
37        video.srcObject = stream;
38      });
39
40    </script>
41  </body>
42 </html>
```

3 07-02.html 顔の特徴となる点を取得しよう

clmtrackr.js を用いて、顔の特徴となる点を取得しよう。緑色の線で顔の輪郭や眉・目・鼻・口の位置を特定するよ。これは、あらかじめ多くの写真で学習させたモデル model_pca_20_svm.js があるからできるよ。

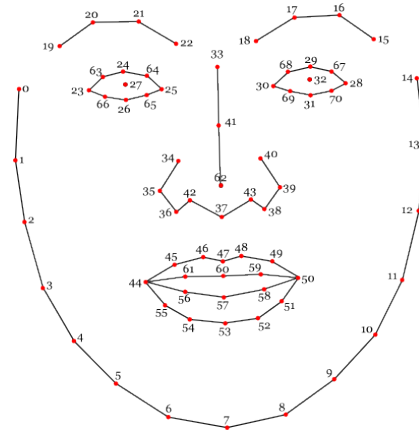
ソースコード 3 07-02.html

```
1 <!DOCTYPE html>
2 <html lang="ja">
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width,initial-scale=1">
6     <title>顔認識</title>
7     <style>
8       /* video 要素の上に canvas 要素をオーバーレイするための CSS */
9       /* コンテナ用の div について */
10      #container {
11        position: relative; /* 座標指定を相対値指定にする */
12      }
13      /* カメラ映像を流す video について */
14      #video {
15        transform: scaleX(-1); /* 左右反転させる */
16      }
17      #canvas { /* 描画用の canvas について */
18        transform: scaleX(-1); /* 左右反転させる */
19        position: absolute; /* 座標指定を絶対値指定にして */
20        left: 0; /* X 座標を 0 に */
21        top: 0; /* Y 座標を 0 に */
22      }
23    </style>
24  </head>
25
26  <body>
27    <!-- video の上に canvas をオーバーレイするための div 要素 -->
28    <div id="container">
29      <!-- カメラ映像を流す video -->
30      <video id="video" width="400" height="300" autoplay></video>
31      <!-- 重ねて描画する canvas -->
32      <canvas id="canvas" width="400" height="300"></canvas>
33    </div>
34
35    <!-- clmtrackr 関連ファイルの読み込み -->
36    <script src="js/clmtrackr.js"></script> <!-- clmtrackr のメインライブラリの読み込み -->
37
38    <script src="js/model_pca_20_svm.js"></script> <!-- 顔モデル(※)の読み込み -->
39
40    <script>
41      // もろもろの準備
42      var video = document.getElementById("video"); // video 要素を取得
43      var canvas = document.getElementById("canvas"); // canvas 要素の取得
44      var context = canvas.getContext("2d"); // canvas の context の取得
45
46      // getUserMedia によるカメラ映像の取得
47      var media = navigator.mediaDevices.getUserMedia({ // メディアデバイスを取得
48        video: {facingMode: "user"},
49        // カメラの映像を使う
50        audio: false
51      });
52      // マイクの音声は使わない
```

```
52     media.then((stream) => { // メディアデバイスが取得できたら
53         video.srcObject = stream;
54     });
55
56     // clmtrackr の開始
57     var tracker = new clm.tracker(); // tracker オブジェクトを作成
58     tracker.init(pModel); // tracker を所定のフェイスモデル(※)で初期化
59     tracker.start(video); // video 要素内でフェイストラッキング開始
60
61     // 描画ループ
62     function drawLoop() {
63         requestAnimationFrame(drawLoop); // drawLoop 関数を繰り返し実行
64         var positions = tracker.getCurrentPosition(); // 顔部品の現在位置の取得
65         context.clearRect(0, 0, canvas.width, canvas.height); // canvas をクリア
66         tracker.draw(canvas); // canvas にトラッキング結果を描画
67     }
68     drawLoop(); // drawLoop 関数をトリガー
69
70
71     </script>
72 </body>
73 </html>
```

4 07-03.html 特徴となる点を数値で見よう

clmtrackr.js で取得された点の配列を数値として表示してみよう。



ソースコード 4 07-03.html

```
1 <!DOCTYPE html>
2 <html lang="ja">
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width,initial-scale=1">
6     <title>顔認識</title>
7     <style>
8       /* video 要素の上に canvas 要素をオーバーレイするための CSS */
9       /* コンテナ用の div について */
10      #container {
11        position: relative; /* 座標指定を相対値指定にする */
12      }
13      /* カメラ映像を流す video について */
14      #video {
15        transform: scaleX(-1); /* 左右反転させる */
16      }
17      #canvas { /* 描画用の canvas について */
18        transform: scaleX(-1); /* 左右反転させる */
19        position: absolute; /* 座標指定を絶対値指定にして */
20        left: 0; /* X 座標を 0 に */
21        top: 0; /* Y 座標を 0 に */
22      }
23    </style>
24  </head>
25
26  <body>
27    <!-- video の上に canvas をオーバーレイするための div 要素 -->
28    <div id="container">
29      <!-- カメラ映像を流す video -->
30      <video id="video" width="400" height="300" autoplay></video>
31      <!-- 重ねて描画する canvas -->
32      <canvas id="canvas" width="400" height="300"></canvas>
33    </div>
34    <div id="dat"></div> <!-- データ表示用 div 要素 -->
35
36    <!-- clmtrackr 関連ファイルの読み込み -->
37    <script src="js/clmtrackr.js"></script> <!-- clmtrackr のメインライブラリの読み込み
```

```

-->
38 <script src="js/model_pca_20_svm.js"></script> <!-- 顔モデル(※)の読み込み -->
39
40 <script>
41     // もろもろの準備
42     var video = document.getElementById("video"); // video 要素を取得
43     var canvas = document.getElementById("canvas"); // canvas 要素の取得
44     var context = canvas.getContext("2d"); // canvas の context の取得
45
46     // getUserMedia によるカメラ映像の取得
47     var media = navigator.mediaDevices.getUserMedia({// メディアデバイスを取得
48         video: {facingMode: "user"},
49         // カメラの映像を使う
50         audio: false
51         // マイクの音声は使わない
52     });
53     media.then((stream) => {// メディアデバイスが取得できたら
54         video.srcObject = stream;
55     });
56
57     // clmtrackr の開始
58     var tracker = new clm.tracker(); // tracker オブジェクトを作成
59     tracker.init(pModel); // tracker を所定のフェイスモデル(※)で初期化
60     tracker.start(video); // video 要素内でフェイストラッキング開始
61
62     // 描画ループ
63     function drawLoop() {
64         requestAnimationFrame(drawLoop); // drawLoop 関数を繰り返し実行
65         var positions = tracker.getCurrentPosition(); // 顔部品の現在位置の取得
66         context.clearRect(0, 0, canvas.width, canvas.height); // canvas をクリア
67         tracker.draw(canvas); // canvas にトラッキング結果を描画
68         showData(positions); // データの表示
69     }
70     drawLoop(); // drawLoop 関数をトリガー
71
72     // 顔部品(特徴点)の位置データを表示する showData 関数
73     function showData(pos) {
74         var str = ""; // データの文字列を入れる変数
75         for(var i = 0; i < pos.length; i++) { // 全ての特徴点(71個)について
76             str += "特徴点" + i + ": ("
77                 + Math.round(pos[i][0]) + ", " // X座標 (四捨五入して整数に)
78                 + Math.round(pos[i][1]) + ")<br>"; // Y座標 (四捨五入して整数に)
79         }
80         var dat = document.getElementById("dat"); // データ表示用div 要素の取得
81         dat.innerHTML = str; // データ文字列の表示
82     }
83
84 </script>
85 </body>
86 </html>

```

5 07-04.html 特徴となる点を元に SNOW にするよ

取得した点をもとに耳と鼻 (ヒゲ) をつけてみよう。

ソースコード 5 07-04.html

```
1 <!DOCTYPE html>
2 <html lang="ja">0
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width,initial-scale=1">
6     <title>顔認識</title>
7     <style>
8       /* video 要素の上に canvas 要素をオーバーレイするための CSS */
9       /* コンテナ用の div について */
10      #container {
11        position: relative; /* 座標指定を相対値指定にする */
12      }
13      /* カメラ映像を流す video について */
14      #video {
15        transform: scaleX(-1); /* 左右反転させる */
16      }
17      #canvas { /* 描画用の canvas について */
18        transform: scaleX(-1); /* 左右反転させる */
19        position: absolute; /* 座標指定を絶対値指定にして */
20        left: 0; /* X 座標を 0 に */
21        top: 0; /* Y 座標を 0 に */
22      }
23    </style>
24  </head>
25
26  <body>
27    <!-- video の上に canvas をオーバーレイするための div 要素 -->
28    <div id="container">
29      <!-- カメラ映像を流す video -->
30      <video id="video" width="400" height="300" autoplay></video>
31      <!-- 重ねて描画する canvas -->
32      <canvas id="canvas" width="400" height="300"></canvas>
33    </div>
34    <div id="dat"></div> <!-- データ表示用 div 要素 -->
35
36    <!-- clmtrackr 関連ファイルの読み込み -->
37    <script src="js/clmtrackr.js"></script> <!-- clmtrackr のメインライブラリの読み込み -->
38
39    <script src="js/model_pca_20_svm.js"></script> <!-- 顔モデル(※)の読み込み -->
40
41    <script>
42      // もろもろの準備
43      var video = document.getElementById("video"); // video 要素を取得
44      var canvas = document.getElementById("canvas"); // canvas 要素の取得
45      var context = canvas.getContext("2d"); // canvas の context の取得
46
47      var stampNose = new Image(); // ★鼻のスタンプ画像を入れる Image オブジェクト
48      var stampEars = new Image(); // ★耳のスタンプ画像を入れる Image オブジェクト
49      stampNose.src = "img/nose.png"; // ★鼻のスタンプ画像のファイル名
50      stampEars.src = "img/ears.png"; // ★耳のスタンプ画像のファイル名
51
52      // getUserMedia によるカメラ映像の取得
53      var media = navigator.mediaDevices.getUserMedia({ // メディアデバイスを取得
```



```

53         video: {facingMode: "user"},
54         // カメラの映像を使う
55         audio: false
56         // マイクの音声は使わない
57     });
58     media.then((stream) => { // メディアデバイスが取得できたら
59         video.srcObject = stream;
60     });
61
62     // clmtrackr の開始
63     var tracker = new clm.tracker(); // tracker オブジェクトを作成
64     tracker.init(pModel); // tracker を所定のフェイスモデル(※)で初期化
65     tracker.start(video); // video 要素内でフェイストラッキング開始
66
67     // 描画ループ
68     function drawLoop() {
69         requestAnimationFrame(drawLoop); // drawLoop 関数を繰り返し実行
70         var positions = tracker.getCurrentPosition(); // 顔部品の現在位置の取得
71         context.clearRect(0, 0, canvas.width, canvas.height); // canvas をクリア
72         //tracker.draw(canvas); // canvas にトラッキング結果を描画
73         //showData(positions); // データの表示
74         drawStamp(positions, stampNose, 62, 2.5, 0.0, 0.0); // ★鼻のスタンプを描画
75         drawStamp(positions, stampEars, 33, 3.0, 0.0, -1.8); // ★耳のスタンプを描画
76     }
77     drawLoop(); // drawLoop 関数をトリガー
78
79     // 顔部品(特徴点)の位置データを表示する showData 関数
80     function showData(pos) {
81         var str = ""; // データの文字列を入れる変数
82         for(var i = 0; i < pos.length; i++) { // 全ての特徴点(71個)について
83             str += "特徴点" + i + ": ("
84                 + Math.round(pos[i][0]) + ", " // X座標(四捨五入して整数に)
85                 + Math.round(pos[i][1]) + ")<br>"; // Y座標(四捨五入して整数に)
86         }
87         var dat = document.getElementById("dat"); // データ表示用div要素の取得
88         dat.innerHTML = str; // データ文字列の表示
89     }
90
91     // ★スタンプを描く drawStamp 関数
92     // (顔部品の位置データ, 画像, 基準位置, 大きさ, 横シフト, 縦シフト)
93     function drawStamp(pos, img, bNo, scale, hShift, vShift) {
94         var eyes = pos[32][0] - pos[27][0]; // 幅の基準として両眼の間隔を求める
95         var nose = pos[62][1] - pos[33][1]; // 高さの基準として眉間と鼻先の間隔を求め
96             る
97         var wScale = eyes / img.width; // 両眼の間隔をもとに画像のスケールを決める
98         var imgW = img.width * scale * wScale; // 画像の幅をスケーリング
99         var imgH = img.height * scale * wScale; // 画像の高さをスケーリング
100         var imgL = pos[bNo][0] - imgW / 2 + eyes * hShift; // 画像のLeftを決める
101         var imgT = pos[bNo][1] - imgH / 2 + nose * vShift; // 画像のTopを決める
102         context.drawImage(img, imgL, imgT, imgW, imgH); // 画像を描く
103     }
104
105     </script>
106 </body>
107 </html>

```

6 07-05.html 抽出された点から感情を導き出す関数を利用してみよう。

clmtrackr.js の感情を分類する機能を使って、感情を読み取ってみよう。(割といい加減かも...)

ソースコード 6 07-05.html

```
1 <!DOCTYPE html>
2 <html lang="ja">
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width,initial-scale=1">
6     <title>顔認識</title>
7     <style>
8       /* video 要素の上に canvas 要素をオーバーレイするための CSS */
9       /* コンテナ用の div について */
10      #container {
11        position: relative; /* 座標指定を相対値指定にする */
12      }
13      /* カメラ映像を流す video について */
14      #video {
15        transform: scaleX(-1); /* 左右反転させる */
16      }
17      #canvas { /* 描画用の canvas について */
18        transform: scaleX(-1); /* 左右反転させる */
19        position: absolute; /* 座標指定を絶対値指定にして */
20        left: 0; /* X 座標を 0 に */
21        top: 0; /* Y 座標を 0 に */
22      }
23    </style>
24  </head>
25
26  <body>
27    <!-- video の上に canvas をオーバーレイするための div 要素 -->
28    <div id="container">
29      <!-- カメラ映像を流す video -->
30      <video id="video" width="400" height="300" autoplay></video>
31      <!-- 重ねて描画する canvas -->
32      <canvas id="canvas" width="400" height="300"></canvas>
33    </div>
34    <div id="dat"></div> <!-- データ表示用 div 要素 -->
35
36    <!-- clmtrackr 関連ファイルの読み込み -->
37    <script src="js/clmtrackr.js"></script> <!-- clmtrackr のメインライブラリの読み込み -->
38
39    <script src="js/model_pca_20_svm.js"></script> <!-- 顔モデル(※)の読み込み -->
40    <!-- clmtrackr 感情系ファイルの読み込み -->
41    <script src="js/emotionClassifier.js"></script> <!-- ★感情を分類する外部関数の読み込み -->
42
43    <script src="js/emotionModel.js"></script> <!-- ★感情モデル(※2)の読み込み -->
44
45    <script>
46      // もろもろの準備
47      var video = document.getElementById("video"); // video 要素を取得
48      var canvas = document.getElementById("canvas"); // canvas 要素の取得
49      var context = canvas.getContext("2d"); // canvas の context の取得
50
51      var stampNose = new Image(); // ★鼻のスタンプ画像を入れる Image オブジェクト
52      var stampEars = new Image(); // ★耳のスタンプ画像を入れる Image オブジェクト
53      stampNose.src = "img/nose.png"; // ★鼻のスタンプ画像のファイル名
```

```

52     stampEars.src = "img/ears.png"; // ★耳のスタンプ画像のファイル名
53
54     var emoji = ["怒り", "うんざり", "恐れ", "悲しみ", "驚き", "嬉しさ"]
55     // getUserMedia によるカメラ映像の取得
56     var media = navigator.mediaDevices.getUserMedia({// メディアデバイスを取得
57         video: {facingMode: "user"},
58         // カメラの映像を使う
59         audio: false
60         // マイクの音声は使わない
61     });
62     media.then((stream) => {// メディアデバイスが取得できたら
63         video.srcObject = stream;
64     });
65
66     // clmtrackr の開始
67     var tracker = new clm.tracker(); // tracker オブジェクトを作成
68     tracker.init(pModel); // tracker を所定のフェイスモデル(※)で初期化
69     tracker.start(video); // video 要素内でフェイストラッキング開始
70
71     // 感情分類の開始
72     var classifier = new emotionClassifier(); // ★emotionClassifier オブジェクトを
        作成
73     classifier.init(emotionModel); // ★classifier を所定の感情モデル(※2)で初期化
74
75
76     // 描画ループ
77     function drawLoop() {
78         requestAnimationFrame(drawLoop); // drawLoop 関数を繰り返し実行
79         var positions = tracker.getCurrentPosition(); // 顔部品の現在位置の取得
80         context.clearRect(0, 0, canvas.width, canvas.height); // canvas をクリア
81         //tracker.draw(canvas); // canvas にトラッキング結果を描画
82         //showData(positions); // データの表示
83         drawStamp(positions, stampNose, 62, 2.5, 0.0, 0.0); // ★鼻のスタンプを描画
84         drawStamp(positions, stampEars, 33, 3.0, 0.0, -1.8); // ★耳のスタンプを描画
85
86         var parameters = tracker.getCurrentParameters(); // ★現在の顔のパラメータを
            取得
87         var emotion = classifier.meanPredict(parameters); // ★そのパラメータから感
            情を推定して emotion に結果を入れる
88         showEmotionData(emotion); // ★感情データを表示
89     }
90     drawLoop(); // drawLoop 関数をトリガー
91
92     // 顔部品(特徴点)の位置データを表示する showData 関数
93     function showData(pos) {
94         var str = ""; // データの文字列を入れる変数
95         for(var i = 0; i < pos.length; i++) { // 全ての特徴点(71個)について
96             str += "特徴点" + i + ": ("
97                 + Math.round(pos[i][0]) + ", " // X座標 (四捨五入して整数に)
98                 + Math.round(pos[i][1]) + ")<br>"; // Y座標 (四捨五入して整数に)
99         }
100         var dat = document.getElementById("dat"); // データ表示用div 要素の取得
101         dat.innerHTML = str; // データ文字列の表示
102     }
103
104     // ★スタンプを描く drawStamp 関数
105     // (顔部品の位置データ, 画像, 基準位置, 大きさ, 横シフト, 縦シフト)
106     function drawStamp(pos, img, bNo, scale, hShift, vShift) {
107         var eyes = pos[32][0] - pos[27][0]; // 幅の基準として両眼の間隔を求める
108         var nose = pos[62][1] - pos[33][1]; // 高さの基準として眉間と鼻先の間隔を求め

```

```

109         var wScale = eyes / img.width; // 両眼の間隔をもとに画像のスケールを決める
110         var imgW = img.width * scale * wScale; // 画像の幅をスケーリング
111         var imgH = img.height * scale * wScale; // 画像の高さをスケーリング
112         var imgL = pos[bNo][0] - imgW / 2 + eyes * hShift; // 画像のLeftを決める
113         var imgT = pos[bNo][1] - imgH / 2 + nose * vShift; // 画像のTopを決める
114         context.drawImage(img, imgL, imgT, imgW, imgH); // 画像を描く
115     }
116
117     // ★感情データの表示
118     function showEmotionData(emo) {
119         var str = ""; // データの文字列を入れる変数
120         for(var i = 0; i < emo.length; i++) { // 全ての感情(6種類)について
121             str += emoji[i] + ": " // 感情名
122                 + emo[i].value.toFixed(1) + "<br>"; // 感情の程度(小数第一位まで)
123         }
124         var dat = document.getElementById("dat"); // データ表示用div要素の取得
125         dat.innerHTML = str; // データ文字列の表示
126     }
127
128     </script>
129 </body>
130 </html>

```

7 07-06.html 感情を元に画像を付け加えよう

読み取った感情を元に、画像を付加してみよう。

ソースコード 7 07-06html

```
1 <!DOCTYPE html>
2 <html lang="ja">
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width,initial-scale=1">
6     <title>顔認識</title>
7     <style>
8       /* video 要素の上に canvas 要素をオーバーレイするための CSS */
9       /* コンテナ用の div について */
10      #container {
11        position: relative; /* 座標指定を相対値指定にする */
12      }
13      /* カメラ映像を流す video について */
14      #video {
15        transform: scaleX(-1); /* 左右反転させる */
16      }
17      #canvas { /* 描画用の canvas について */
18        transform: scaleX(-1); /* 左右反転させる */
19        position: absolute; /* 座標指定を絶対値指定にして */
20        left: 0; /* X 座標を 0 に */
21        top: 0; /* Y 座標を 0 に */
22      }
23    </style>
24  </head>
25
26  <body>
27    <!-- video の上に canvas をオーバーレイするための div 要素 -->
28    <div id="container">
29      <!-- カメラ映像を流す video -->
30      <video id="video" width="400" height="300" autoplay></video>
31      <!-- 重ねて描画する canvas -->
32      <canvas id="canvas" width="400" height="300"></canvas>
33    </div>
34    <div id="dat"></div> <!-- データ表示用 div 要素 -->
35
36    <!-- clmtrackr 関連ファイルの読み込み -->
37    <script src="js/clmtrackr.js"></script> <!-- clmtrackr のメインライブラリの読み込み -->
38
39    <script src="js/model_pca_20_svm.js"></script> <!-- 顔モデル(※)の読み込み -->
40    <!-- clmtrackr 感情系ファイルの読み込み -->
41    <script src="js/emotionClassifier.js"></script> <!-- ★感情を分類する外部関数の読み込み -->
42
43    <script src="js/emotionModel.js"></script> <!-- ★感情モデル(※2)の読み込み -->
44
45    <script>
46      // もろもろの準備
47      var video = document.getElementById("video"); // video 要素を取得
48      var canvas = document.getElementById("canvas"); // canvas 要素の取得
49      var context = canvas.getContext("2d"); // canvas の context の取得
50
51      var stampNose = new Image(); // ★鼻のスタンプ画像を入れる Image オブジェクト
52      var stampEars = new Image(); // ★耳のスタンプ画像を入れる Image オブジェクト
53      stampNose.src = "img/nose.png"; // ★鼻のスタンプ画像のファイル名
```

```

52     stampEars.src = "img/ears.png"; // ★耳のスタンプ画像のファイル名
53
54     var stampTear = new Image(); // ★涙のスタンプ画像を入れる Image オブジェクト
55     var stampSurp = new Image(); // ★驚きのスタンプ画像を入れる Image オブジェクト
56     var stampEyes = new Image(); // ★ハートのスタンプ画像を入れる Image オブジェクト
57     stampTear.src = "img/tear.png"; // ★涙のスタンプ画像のファイル名
58     stampSurp.src = "img/surp.png"; // ★驚きのスタンプ画像のファイル名
59     stampEyes.src = "img/eyes.png"; // ★ハートのスタンプ画像のファイル名
60
61
62     var emoji = ["怒り", "うんざり", "恐れ", "悲しみ", "驚き", "嬉しさ"]
63     // getUserMedia によるカメラ映像の取得
64     var media = navigator.mediaDevices.getUserMedia({// メディアデバイスを取得
65         video: {facingMode: "user"},
66         // カメラの映像を使う
67         audio: false
68         // マイクの音声は使わない
69     });
70     media.then((stream) => {// メディアデバイスが取得できたら
71         video.srcObject = stream;
72     });
73
74     // clmtrackr の開始
75     var tracker = new clm.tracker(); // tracker オブジェクトを作成
76     tracker.init(pModel); // tracker を所定のフェイスモデル(※)で初期化
77     tracker.start(video); // video 要素内でフェイストラッキング開始
78
79     // 感情分類の開始
80     var classifier = new emotionClassifier(); // ★emotionClassifier オブジェクトを
        作成
81     classifier.init(emotionModel); // ★classifier を所定の感情モデル(※ 2)で初期化
82
83
84     // 描画ループ
85     function drawLoop() {
86         requestAnimationFrame(drawLoop); // drawLoop 関数を繰り返し実行
87         var positions = tracker.getCurrentPosition(); // 顔部品の現在位置の取得
88         context.clearRect(0, 0, canvas.width, canvas.height); // canvas をクリア
89         //tracker.draw(canvas); // canvas にトラッキング結果を描画
90         //showData(positions); // データの表示
91         drawStamp(positions, stampNose, 62, 2.5, 0.0, 0.0); // ★鼻のスタンプを描画
92         drawStamp(positions, stampEars, 33, 3.0, 0.0, -1.8); // ★耳のスタンプを描画
93
94         var parameters = tracker.getCurrentParameters(); // ★現在の顔のパラメータを
            取得
95         var emotion = classifier.meanPredict(parameters); // ★そのパラメータから感
            情を推定して emotion に結果を入れる
96         //showEmotionData(emotion); // ★感情データを表示
97
98         if(emotion[3].value > 0.4) { // ★感情 sad の値が一定値より大きければ
99             drawStamp(positions, stampTear, 23, 0.4, 0.0, 0.8); // ★涙のスタンプを
                描画(右目尻)
100             drawStamp(positions, stampTear, 28, 0.4, 0.0, 0.8); // ★涙のスタンプを
                描画(左目尻)
101         }
102         if(emotion[4].value > 0.8) { // ★感情 surprised の値が一定値より大きければ
103             drawStamp(positions, stampSurp, 14, 1.0, 0.7, 0.0); // ★驚きのスタンプ
                を描画(顔の左)
104         }
105         if(emotion[5].value > 0.4) { // ★感情 happy の値が一定値より大きければ

```

```

106         drawStamp(positions, stampEyes, 27, 0.6, 0.0, 0.0); // ★ハートのスタンプを描画(右目)
107         drawStamp(positions, stampEyes, 32, 0.6, 0.0, 0.0); // ★ハートのスタンプを描画(左目)
108     }
109
110 }
111 drawLoop(); // drawLoop 関数をトリガー
112
113 // 顔部品(特徴点)の位置データを表示する showData 関数
114 function showData(pos) {
115     var str = ""; // データの文字列を入れる変数
116     for(var i = 0; i < pos.length; i++) { // 全ての特徴点(71個)について
117         str += "特徴点" + i + ": ("
118             + Math.round(pos[i][0]) + ", " // X座標(四捨五入して整数に)
119             + Math.round(pos[i][1]) + ")<br>"; // Y座標(四捨五入して整数に)
120     }
121     var dat = document.getElementById("dat"); // データ表示用div要素の取得
122     dat.innerHTML = str; // データ文字列の表示
123 }
124
125 // ★スタンプを描く drawStamp 関数
126 // (顔部品の位置データ, 画像, 基準位置, 大きさ, 横シフト, 縦シフト)
127 function drawStamp(pos, img, bNo, scale, hShift, vShift) {
128     var eyes = pos[32][0] - pos[27][0]; // 幅の基準として両眼の間隔を求める
129     var nose = pos[62][1] - pos[33][1]; // 高さの基準として眉間と鼻先の間隔を求める
130     var wScale = eyes / img.width; // 両眼の間隔をもとに画像のスケールを決める
131     var imgW = img.width * scale * wScale; // 画像の幅をスケーリング
132     var imgH = img.height * scale * wScale; // 画像の高さをスケーリング
133     var imgL = pos[bNo][0] - imgW / 2 + eyes * hShift; // 画像のLeftを決める
134     var imgT = pos[bNo][1] - imgH / 2 + nose * vShift; // 画像のTopを決める
135     context.drawImage(img, imgL, imgT, imgW, imgH); // 画像を描く
136 }
137
138 // ★感情データの表示
139 function showEmotionData(emo) {
140     var str = ""; // データの文字列を入れる変数
141     for(var i = 0; i < emo.length; i++) { // 全ての感情(6種類)について
142         str += emoji[i] + ": " // 感情名
143             + emo[i].value.toFixed(1) + "<br>"; // 感情の程度(小数第一位まで)
144     }
145     var dat = document.getElementById("dat"); // データ表示用div要素の取得
146     dat.innerHTML = str; // データ文字列の表示
147 }
148
149 </script>
150 </body>
151 </html>

```

以上