```
1 <!doctype html>
 2 <html>
 3
  k!-- @author : bilibili:一只斌 /
                                     mail: tuduweb@qq.com-->
 4
  <head>
 5
      <title>Camera</title>
 6
      <style>
 7
          #user-list>li {
8
              font-size: 24px;
9
10
      </style>
  </head>
11
12
13
  <body>
14
      <h1 id="user-id">用户名称</h1>
15
      ul id="user-list">
16
          用户12
17
          和 23
18
          利i>用户34
      19
20
      <video id="video-local" controls autoplay></video>
21
      <canvas id="capture-canvas" style="display: none;"></canvas>
22
23
24
      <button id="capture">拍照</button>
25
26
      ul id="capture-list">
27
28
29
      <div id="videos"></div>
30
      <script src="//cdn.bootcdn.net/ajax/libs/socket.io/3.0.4/socket.io.js"></script>
31
32
      <script src="//cdn.bootcdn.net/ajax/libs/jquery/3.4.1/jquery.js"></script>
33
      <script>
34
          //封装一部分函数
35
          function getUserMedia(constrains, success, error) {
36
              if (navigator.mediaDevices.getUserMedia) {
37
                  //最新标准API
38
                  promise =
   navigator.mediaDevices.getUserMedia(constrains).then(success).catch(error);
39
              } else if (navigator.webkitGetUserMedia) {
40
                  //webkit内核浏览器
41
                  promise =
   navigator.webkitGetUserMedia(constrains).then(success).catch(error);
42
              } else if (navigator.mozGetUserMedia) {
43
                  //Firefox浏览器
44
                  promise = navagator.mozGetUserMedia(constrains).then(success).catch(error);
              } else if (navigator.getUserMedia) {
45
46
                  //旧版API
                  promise = navigator.getUserMedia(constrains).then(success).catch(error);
47
48
              }
49
          }
50
          function canGetUserMediaUse() {
51
52
              return !!(navigator.mediaDevices.getUserMedia | navigator.webkitGetUserMedia
   | navigator.mozGetUserMedia | navigator.msGetUserMedia);
53
          }
54
55
          const localVideoElm = document.getElementById("video-local");
56
57
          $('document').ready(() => {
58
```

localhost:63808 1/6

```
59
 60
               $('#capture').click(() => {
 61
                   let video = localVideoElm//原生dom
 62
                   let isPlaying = !(video.paused || video.ended || video.seeking ||
    video.readyState < video.HAVE_FUTURE_DATA)</pre>
 63
                   if (isPlaying) {
 64
                       let canvas = $('#capture-canvas')
 65
                       canvas.attr('width', localVideoElm.clientWidth);//设置canvas的宽度
 66
 67
                       canvas.attr('height', localVideoElm.clientHeight);//设置canvas的高度
 68
                       let img = $('<img>')
 69
 70
                       img.attr('width', localVideoElm.clientWidth);//设置图像的宽度
 71
                       img.attr('height', localVideoElm.clientHeight);//设置图像的高度
 72
 73
                       //canvas[0] //jQuery对象转dom
 74
                       var context = canvas[0].getContext('2d');
 75
                       //在canvas上绘图,其绘图坐标为0,0;
76
                       //绘图大小为摄像头内容的宽度, 高度(全局绘制, 你可以改变这些值试试效果)。
                       context.drawImage(localVideoElm, 0, 0, localVideoElm.clientWidth,
 77
    localVideoElm.clientHeight);
 78
                       //根据canvas内容进行编码,并赋值到图片上
 79
                       var data = canvas[0].toDataURL('image/png');
                       img.attr('src', data);
 80
 81
                       //插入到id为capture-list的有序列表里
                       $('#capture-list').append($('').html(img));
 82
 83
               })
 24
 85
           });
 86
 87
 88
           //STUN, TURN服务器配置参数
 89
           const iceServer = {
 90
               iceServers: [{ urls: ["stun:ss-turn1.xirsys.com"] }, { username:
    "CEqIDkX5f51sbm7-pXxJVXePoMk_WB7w2J5eu0Bd00YpiONH1LHrwSb7hRMDDrqGAAAAAF_OT9V0dWR1d2Vi"
    credential: "446118be-38a4-11eb-9ece-0242ac140004", urls: ["turn:ss-turn1.xirsys.com:80?
    transport=udp", "turn:ss-turn1.xirsys.com:3478?transport=udp"] }]
 91
           };
 92
 93
           //PeerConnection
 94
           var pc = [];
 95
           var localStream = null;
 96
 97
           function InitCamera() {
 98
 99
               if (canGetUserMediaUse()) {
                   getUserMedia({
100
101
                       video: true,
                       audio: false
102
103
                   }, (stream) => {
                       localStream = stream;
104
                       localVideoElm.srcObject = stream;
105
                       $(localVideoElm).width(800);
106
107
                   }, (err) => {
108
                       console.log('访问用户媒体失败: ', err.name, err.message);
109
                   });
110
               } else {
111
                   alert('您的浏览器不兼容');
112
113
114
           }
```

localhost:63808 2/6

```
115
116
           function StartCall(parterName, createOffer) {
117
               pc[parterName] = new RTCPeerConnection(iceServer);
118
119
120
               //如果已经有本地流,那么直接获取Tracks并调用addTrack添加到RTC对象中。
121
               if (localStream) {
122
                  localStream.getTracks().forEach((track) => {
123
                      pc[parterName].addTrack(track, localStream);//should trigger
124
   negotiationneeded event
125
                  });
126
127
               }else{
128
                  //否则需要重新启动摄像头并获取
129
                  if (canGetUserMediaUse()) {
130
                      getUserMedia({
131
                          video: true,
132
                          audio: false
133
                      }, function (stream) {
134
                          localStream = stream;
135
136
                          localVideoElm.srcObject = stream;
                          $(localVideoElm).width(800);
137
138
                      }, function (error) {
139
140
                          console.log("访问用户媒体设备失败: ", error.name, error.message);
141
                      })
142
                  } else { alert('您的浏览器不兼容'); }
143
               }
144
145
146
               //如果是呼叫方,那么需要createOffer请求
147
               if (createOffer) {
148
                   //每当WebRTC基础结构需要你重新启动会话协商过程时,都会调用此函数。它的工作是创建和
        ·个请求,给被叫方,要求它与我们联系。
149
                  pc[parterName].onnegotiationneeded = () => {
                      //https://developer.mozilla.org/zh-
150
   CN/docs/Web/API/RTCPeerConnection/createOffer
151
152
                      pc[parterName].createOffer().then((offer) => {
153
                          return pc[parterName].setLocalDescription(offer);
                      }).then(() => {
154
155
                          //把发起者的描述信息通过Signal Server发送到接收者
156
                          socket.emit('sdp', {
157
                              type: 'video-offer',
158
                              description: pc[parterName].localDescription,
159
                              to: parterName,
160
                              sender: socket.id
161
                          });
                      })
162
163
                  };
               }
164
165
166
               //当需要你通过信令服务器将一个ICE候选发送给另一个对等端时,本地ICE层将会调用你的
   icecandidate 事件处理程序。有关更多信息,请参阅Sending ICE candidates 以查看此示例的代码。
               pc[parterName].onicecandidate = ({ candidate }) => {
167
                  socket.emit('ice candidates', {
168
169
                      candidate: candidate,
170
                      to: parterName,
171
                      sender: socket.id
```

localhost:63808 3/6

```
12/26/2020
```

```
172
                   });
173
               };
174
175
               //当向连接中添加磁道时,track 事件的此处理程序由本地WebRTC层调用。例如,可以将传入媒体
   连接到元素以显示它。详见 Receiving new streams 。
176
               pc[parterName].ontrack = (ev) => {
177
                   let str = ev.streams[0];
178
                   if (document.getElementById(`${parterName}-video`)) {
179
                       document.getElementById(`${parterName}-video`).srcObject = str;
180
181
                   } else {
182
                       let newVideo = document.createElement('video');
                       newVideo.id = `${parterName}-video`;
183
184
                       newVideo.autoplay = true;
185
                       newVideo.controls = true;
186
                       //newVideo.className = 'remote-video';
187
                       newVideo.srcObject = str;
188
189
                       document.getElementById('videos').appendChild(newVideo);
190
                   }
               }
191
192
193
194
195
196
197
           var socket = io();
198
199
           socket.on('connect', () => {
200
               InitCamera();
201
202
               //输出内容 其中 socket.id 是当前socket连接的唯一ID
203
               console.log('connect ' + socket.id);
204
               $('#user-id').text(socket.id);
205
206
207
               pc.push(socket.id);
208
209
               socket.emit('new user greet', {
210
                   sender: socket.id,
211
                   msg: 'hello world'
212
               });
213
               socket.on('need connect', (data) => {
214
215
216
                   console.log(data);
217
                   //创建新的Li并添加到用户列表中
218
                   let li = $('').text(data.sender).attr('user-id', data.sender);
219
                   $('#user-list').append(li);
220
                   //同时创建一个按钮
221
                   let button = $('<button class="call">通话</button>');
222
                   button.appendTo(li);
223
                   //监听按钮的点击事件,这是个demo 需要添加很多东西,比如不能重复拨打已经连接的用户
224
                   $(button).click(function () {
225
                       //$(this).parent().attr('user-id')
                       console.log($(this).parent().attr('user-id'));
226
227
                       //点击时,开启对该用户的通话
228
                       StartCall($(this).parent().attr('user-id'), true);
229
                   });
230
231
                   socket.emit('ok we connect', { receiver: data.sender, sender: socket.id });
```

localhost:63808 4/6

```
12/26/2020
 232
 233
  234
 235
```

```
});
               //某个用户失去连接时,我们需要获取到这个信息
               socket.on('user disconnected', (socket_id) => {
                   console.log('disconnect : ' + socket id);
236
                   $('#user-list li[user-id="' + socket_id + '"]').remove();
237
238
               })
239
               //链接吧..
240
               socket.on('ok we connect', (data) => {
241
                   console.log(data);
242
                   $('#user-list').append($('').text(data.sender).attr('user-id',
243
   data.sender));
244
                   //这里少了程序,比如之前的按钮啊,按钮的点击监听都没有。
245
               });
246
247
               //监听发送的sdp事件
               socket.on('sdp', (data) => {
248
249
                   //如果时offer类型的sdp
250
                   if (data.description.type === 'offer') {
251
                      //那么被呼叫者需要开启RTC的一套流程,同时不需要createOffer,所以第二个参数为
   false
252
                      StartCall(data.sender, false);
253
                      //把发送者(offer)的描述,存储在接收者的remoteDesc中。
                      let desc = new RTCSessionDescription(data.description);
254
255
                      //按1-13流程走的
256
                      pc[data.sender].setRemoteDescription(desc).then(() => {
257
258
                          pc[data.sender].createAnswer().then((answer) => {
259
                              return pc[data.sender].setLocalDescription(answer);
260
                          }).then(() => {
                              socket.emit('sdp', {
261
262
                                  type: 'video-answer',
263
                                  description: pc[data.sender].localDescription,
                                  to: data.sender,
264
                                  sender: socket.id
265
266
                              });
267
268
                           }).catch();//catch error function empty
269
270
                      })
                   } else if (data.description.type === 'answer') {
271
272
                      //如果使应答类消息 (那么接收到这个事件的是呼叫者)
                      let desc = new RTCSessionDescription(data.description);
273
274
                      pc[data.sender].setRemoteDescription(desc);
275
               })
276
277
278
               //如果是ice candidates的协商信息
279
               socket.on('ice candidates', (data) => {
                   console.log('ice candidate: ' + data.candidate);
280
                   //{ candidate: candidate, to: partnerName, sender: socketID }
281
282
                   //如果ice candidate非空 (当candidate为空时,那么本次协商流程到此结束了)
283
                   if (data.candidate) {
                      var candidate = new RTCIceCandidate(data.candidate);
284
285
                      //讲对方发来的协商信息保存
286
                      pc[data.sender].addIceCandidate(candidate).catch();//catch err function
   empty
287
                   }
               })
288
289
```

localhost:63808 5/6

```
12/26/2020
```

localhost:63808 6/6