```html
<!doctype html>
<html>
<!-- @author :  bilibili:一只斌  /   mail: tuduweb@qq.com-->
<head>
    <title>Camera</title>
    <style>
        #user-list>li {
            font-size: 24px;
        }
    </style>
</head>

<body>
    <h1 id="user-id">用户名称</h1>
    <ul id="user-list">
        <li>用户12</li>
        <li>用户23</li>
        <li>用户34</li>
    </ul>
    <video id="video-local" controls autoplay></video>

    <canvas id="capture-canvas" style="display: none;"></canvas>

    <button id="capture">拍照</button>

    <ul id="capture-list"></ul>


    <div id="videos"></div>

    <script src="//cdn.bootcdn.net/ajax/libs/socket.io/3.0.4/socket.io.js"></script>
    <script src="//cdn.bootcdn.net/ajax/libs/jquery/3.4.1/jquery.js"></script>
    <script>
        //封装一部分函数
        function getUserMedia(constrains, success, error) {
            if (navigator.mediaDevices.getUserMedia) {
                //最新标准API
                promise = navigator.mediaDevices.getUserMedia(constrains).then(success).catch(error);
            } else if (navigator.webkitGetUserMedia) {
                //webkit内核浏览器
                promise = navigator.webkitGetUserMedia(constrains).then(success).catch(error);
            } else if (navigator.mozGetUserMedia) {
                //Firefox浏览器
                promise = navagator.mozGetUserMedia(constrains).then(success).catch(error);
            } else if (navigator.getUserMedia) {
                //旧版API
                promise = navigator.getUserMedia(constrains).then(success).catch(error);
            }
        }

        function canGetUserMediaUse() {
            return !!(navigator.mediaDevices.getUserMedia || navigator.webkitGetUserMedia || navigator.mozGetUserMedia || navigator.msGetUserMedia);
        }


        const localVideoElm = document.getElementById("video-local");
        $('document').ready(() => {


            $('#capture').click(() => {
                let video = localVideoElm//原生dom
                let isPlaying = !(video.paused || video.ended || video.seeking || video.readyState < video.HAVE_FUTURE_DATA)

                if (isPlaying) {
                    let canvas = $('#capture-canvas')
                    canvas.attr('width', localVideoElm.clientWidth);//设置canvas的宽度
                    canvas.attr('height', localVideoElm.clientHeight);//设置canvas的高度

                    let img = $('<img>')
                    img.attr('width', localVideoElm.clientWidth);//设置图像的宽度
                    img.attr('height', localVideoElm.clientHeight);//设置图像的高度

                    //canvas[0] //jQuery对象转dom
                    var context = canvas[0].getContext('2d');
                    //在canvas上绘图，其绘图坐标为0,0;
                    //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, localVideoElm.clientHeight);
                    //根据canvas内容进行编码，并赋值到图片上
                    var data = canvas[0].toDataURL('image/png');
                    img.attr('src', data);
                    //插入到id为capture-list的有序列表里
                    $('#capture-list').append($('<li></li>').html(img));
                }
            })

        });

        //STUN,TURN服务器配置参数
        const iceServer = {
            iceServers: [{ urls: ["stun:ss-turn1.xirsys.com"] }, { username: "CEqIDkX5f51sbm7-pXxJVXePoMk_WB7w2J5eu0Bd00YpiONHlLHrwSb7hRMDDrqGAAAAAF_OT9V0dWR1d2Vi", credential: "446118be-38a4-11eb-9ece-0242ac140004", urls: ["turn:ss-turn1.xirsys.com:80?transport=udp", "turn:ss-turn1.xirsys.com:3478?transport=udp"] }]
        };

        //PeerConnection
        var pc = [];
        var localStream = null;

        function InitCamera() {

            if (canGetUserMediaUse()) {
                getUserMedia({
                    video: true,
                    audio: false
                }, (stream) => {
                    localStream = stream;
                    localVideoElm.srcObject = stream;
                    $(localVideoElm).width(800);
                }, (err) => {
                    console.log('访问用户媒体失败：', err.name, err.message);
                });
            } else {
                alert('您的浏览器不兼容');
            }

        }

        function StartCall(parterName, createOffer) {

            pc[parterName] = new RTCPeerConnection(iceServer);

            //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。
            if (localStream) {

                localStream.getTracks().forEach((track) => {
                    pc[parterName].addTrack(track, localStream);//should trigger negotiationneeded event
                });

            }else{
                //否则需要重新启动摄像头并获取
                if (canGetUserMediaUse()) {
                    getUserMedia({
                        video: true,
                        audio: false
                    }, function (stream) {
                        localStream = stream;

                        localVideoElm.srcObject = stream;
                        $(localVideoElm).width(800);

                    }, function (error) {
                        console.log("访问用户媒体设备失败：", error.name, error.message);
                    })
                } else { alert('您的浏览器不兼容'); }

            }

            //如果是呼叫方,那么需要createOffer请求
            if (createOffer) {
                //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。它的工作是创建和发送一个请求，给被叫方，要求它与我们联系。
                pc[parterName].onnegotiationneeded = () => {
                    //https://developer.mozilla.org/zh-CN/docs/Web/API/RTCPeerConnection/createOffer

                    pc[parterName].createOffer().then((offer) => {
                        return pc[parterName].setLocalDescription(offer);
                    }).then(() => {
                        //把发起者的描述信息通过Signal Server发送到接收者
                        socket.emit('sdp', {
                            type: 'video-offer',
                            description: pc[parterName].localDescription,
                            to: parterName,
                            sender: socket.id
                        });
                    })
                };
            }

            //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 }) => {
                socket.emit('ice candidates', {
                    candidate: candidate,
                    to: parterName,
                    sender: socket.id
                });
            };

            //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 。
            pc[parterName].ontrack = (ev) => {
                let str = ev.streams[0];

                if (document.getElementById(`${parterName}-video`)) {
                    document.getElementById(`${parterName}-video`).srcObject = str;
                } else {
                    let newVideo = document.createElement('video');
                    newVideo.id = `${parterName}-video`;
                    newVideo.autoplay = true;
                    newVideo.controls = true;
                    //newVideo.className = 'remote-video';
                    newVideo.srcObject = str;

                    document.getElementById('videos').appendChild(newVideo);
                }
            }



        }

        var socket = io();

        socket.on('connect', () => {
            InitCamera();

            //输出内容 其中 socket.id 是当前socket连接的唯一ID
            console.log('connect ' + socket.id);

            $('#user-id').text(socket.id);

            pc.push(socket.id);

            socket.emit('new user greet', {
                sender: socket.id,
                msg: 'hello world'
            });

            socket.on('need connect', (data) => {

                console.log(data);
                //创建新的li并添加到用户列表中
                let li = $('<li></li>').text(data.sender).attr('user-id', data.sender);
                $('#user-list').append(li);
                //同时创建一个按钮
                let button = $('<button class="call">通话</button>');
                button.appendTo(li);
                //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
                $(button).click(function () {
                    //$(this).parent().attr('user-id')
                    console.log($(this).parent().attr('user-id'));
                    //点击时，开启对该用户的通话
                    StartCall($(this).parent().attr('user-id'), true);
                });

                socket.emit('ok we connect', { receiver: data.sender, sender: socket.id });
            });
            //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);

                $('#user-list li[user-id="' + socket_id + '"]').remove();
            })
            //链接吧..
            socket.on('ok we connect', (data) => {
                console.log(data);

                $('#user-list').append($('<li></li>').text(data.sender).attr('user-id', data.sender));
                //这里少了程序，比如之前的按钮啊，按钮的点击监听都没有。
            });

            //监听发送的sdp事件
            socket.on('sdp', (data) => {
                //如果时offer类型的sdp
                if (data.description.type === 'offer') {
                    //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
                    StartCall(data.sender, false);
                    //把发送者(offer)的描述，存储在接收者的remoteDesc中。
                    let desc = new RTCSessionDescription(data.description);
                    //按1-13流程走的
                    pc[data.sender].setRemoteDescription(desc).then(() => {

                        pc[data.sender].createAnswer().then((answer) => {
                            return pc[data.sender].setLocalDescription(answer);
                        }).then(() => {
                            socket.emit('sdp', {
                                type: 'video-answer',
                                description: pc[data.sender].localDescription,
                                to: data.sender,
                                sender: socket.id
                            });

                        }).catch();//catch error function empty

                    })
                } else if (data.description.type === 'answer') {
                    //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);
                    pc[data.sender].setRemoteDescription(desc);
                }
            })

            //如果是ice candidates的协商信息
            socket.on('ice candidates', (data) => {
                console.log('ice candidate: ' + data.candidate);
                //{ candidate: candidate, to: partnerName, sender: socketID }
                //如果ice candidate非空（当candidate为空时，那么本次协商流程到此结束了）
                if (data.candidate) {
                    var candidate = new RTCIceCandidate(data.candidate);
                    //讲对方发来的协商信息保存
                    pc[data.sender].addIceCandidate(candidate).catch();//catch err function empty
                }
            })

```

```

        });


    </script>
</body>


</html>
```