

数字媒体资源管理实验报告

**2019/2020(1)**



作业题目 实验 2. 基于WebRTC的视频语音聊天系统

学生姓名 林英琮

学生学号 201706062412

学生班级 数媒1702

任课教师 简琤峰

提交日期 2019/10/22

**计算机科学与技术学院**

目录

[1. 实验内容 3](#_Toc23799823)

[2. 实验目的 3](#_Toc23799824)

[3. 实验流程 3](#_Toc23799825)

[3.1. 实验前必须的准备与说明 3](#_Toc23799826)

[3.2. 系统下nodejs环境搭建 3](#_Toc23799827)

[3.3. 一个简单的示例 3](#_Toc23799828)

[3.4. 综合实验 4](#_Toc23799829)

[3.4.1. 样例：SkyRTC-demo 5](#_Toc23799830)

[3.4.2. 自己编写的聊天室（期末总网站的一部分） 7](#_Toc23799831)

[4. 问题与难点 9](#_Toc23799832)

[4.1. 相关WebRTC的熟悉 9](#_Toc23799833)

[4.2. 流媒体数据的传输和接收 9](#_Toc23799834)

[5. 附录 10](#_Toc23799835)

[5.1. 代码仓库： 10](#_Toc23799836)

[5.2. 调用摄像头基础js代码： 10](#_Toc23799837)

[5.3. 参考文档 11](#_Toc23799838)

# 实验内容

* 安装配置nodejs环境
* 完成一个基于WebRTC获取来自摄像头视频的小实验
* 完成两（多）人之间的基于webRTC的前端视频语音聊天系统

# 实验目的

* 掌握Nodejs安装及环境配置
* 熟悉并了解WebRTC协议及相关API使用
* 掌握使用WebRTC来进行流媒体传输

# 实验流程

## 实验前必须的准备与说明

* 计算机需要连接Internet以安装软件。
* 实验需要用firefox浏览器以达到WebRTC实验的预期效果。
* 需要准备USB摄像头。

## 系统下nodejs环境搭建

由于我曾经做过web开发，本地环境已经搭建好node.js。



图 3‑1 nodejs环境

## 一个简单的示例

捕捉摄像头视频在HTML5的video标签显示。

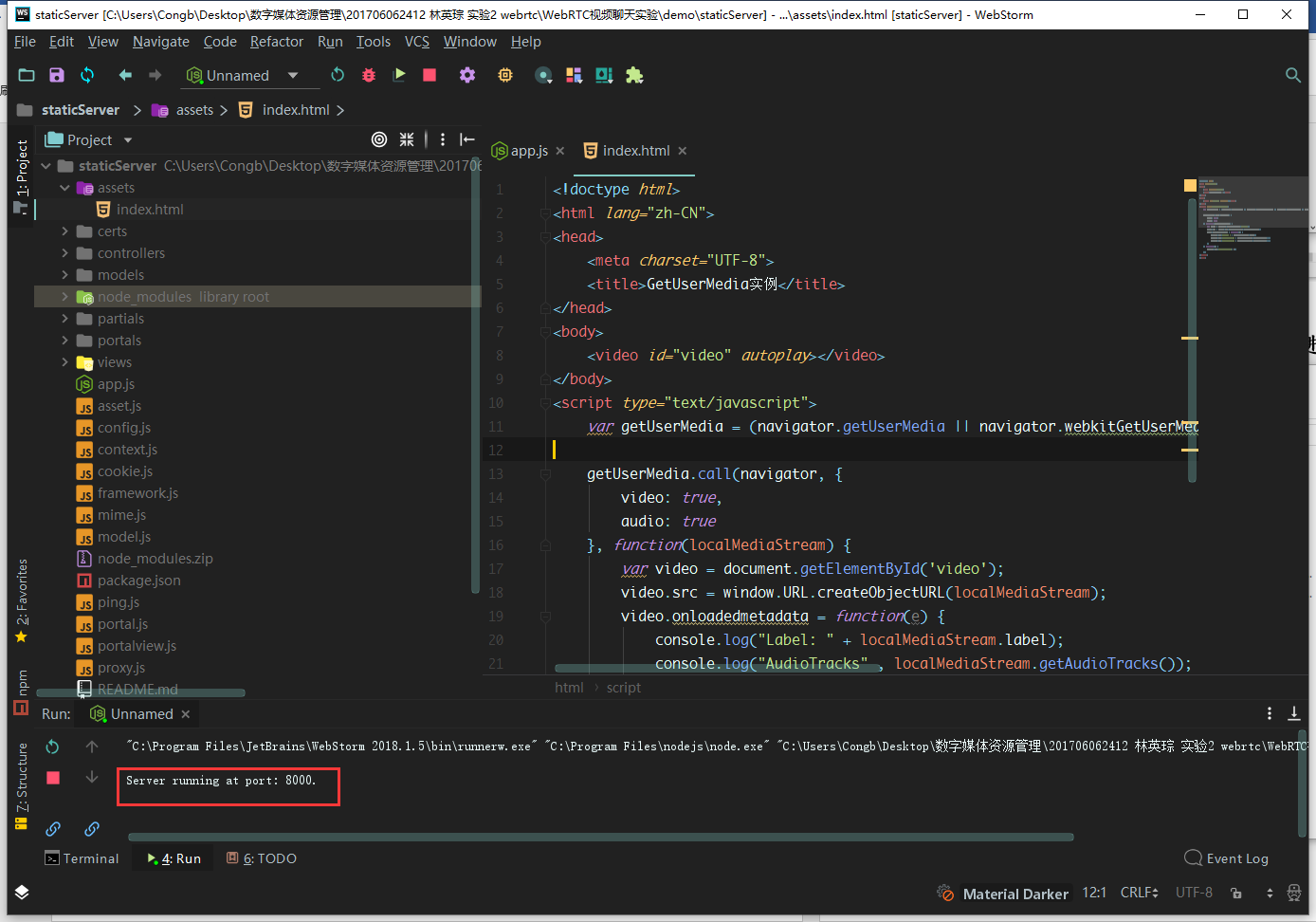


图 3‑2 启动

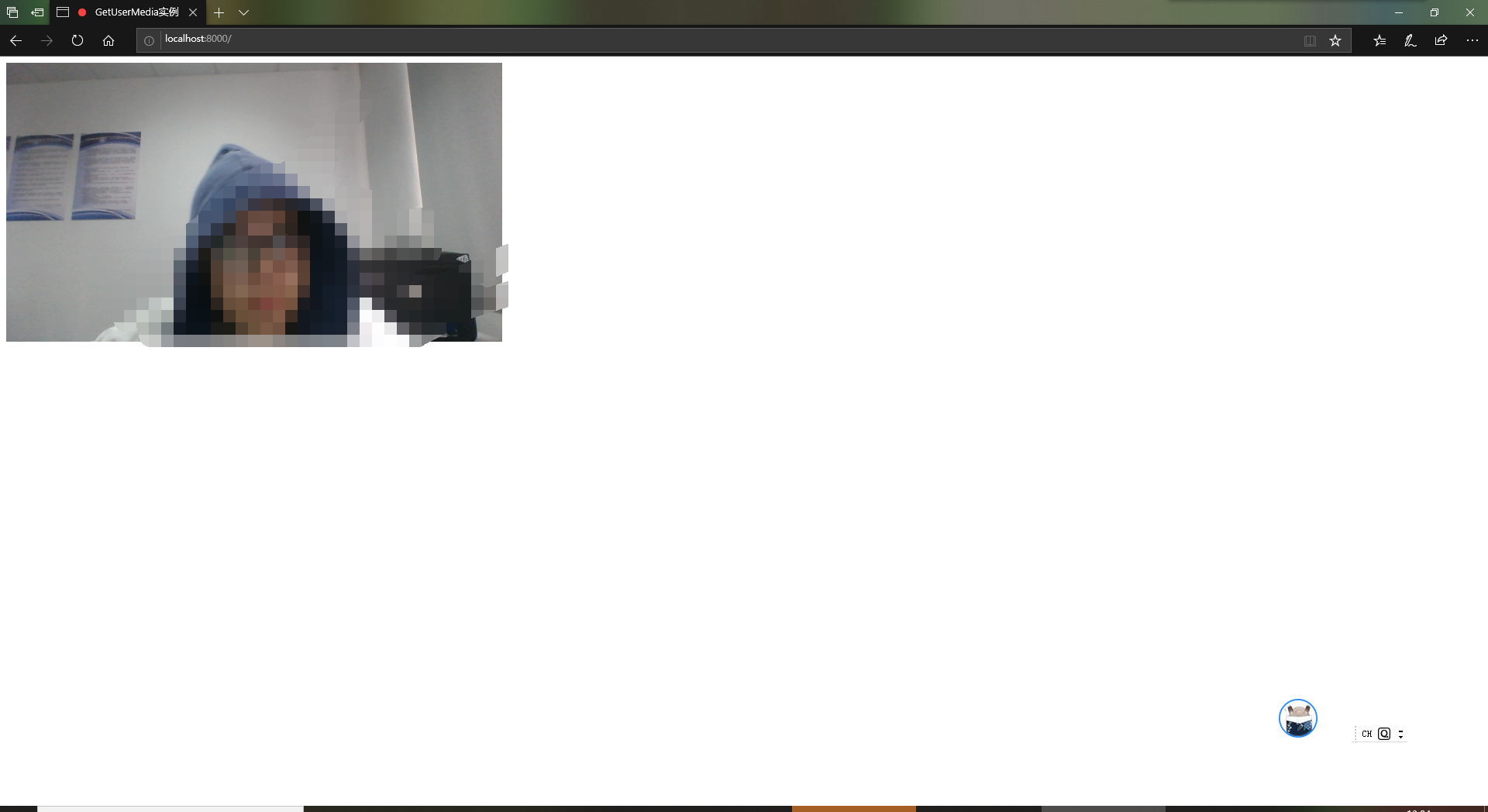


图 3‑3 效果

## 综合实验

核心内容为使用Webrtc API，构建支持划分房间的在线音频、视频、文字聊天，提供房间内文件共享功能。

WebRTC (Web Real-Time Communications) 是一项实时通讯技术，它允许网络应用或者站点，在不借助中间媒介的情况下，建立浏览器之间点对点（Peer-to-Peer）的连接，实现视频流和（或）音频流或者其他任意数据的传输。WebRTC包含的这些标准使用户在无需安装任何插件或者第三方的软件的情况下，创建点对点（Peer-to-Peer）的数据分享和电话会议成为可能。

WebRTC包含了若干相互关联的API和协议以达到这个目标。

### 样例：SkyRTC-demo

经过查找，由老师提供的SkyRTC-demo的原代码地址为：

<https://github.com/LingyuCoder/SkyRTC-demo>

这是一个基于已经编写好的SkyRTC服务器端库：

<https://github.com/LingyuCoder/SkyRTC>

和SkyRTC-client前端库：

<https://github.com/LingyuCoder/SkyRTC-client>

共同编写的基于WebRTC和WebSocket技术的在线音频、视频聊天室demo。

* 本地启动测试

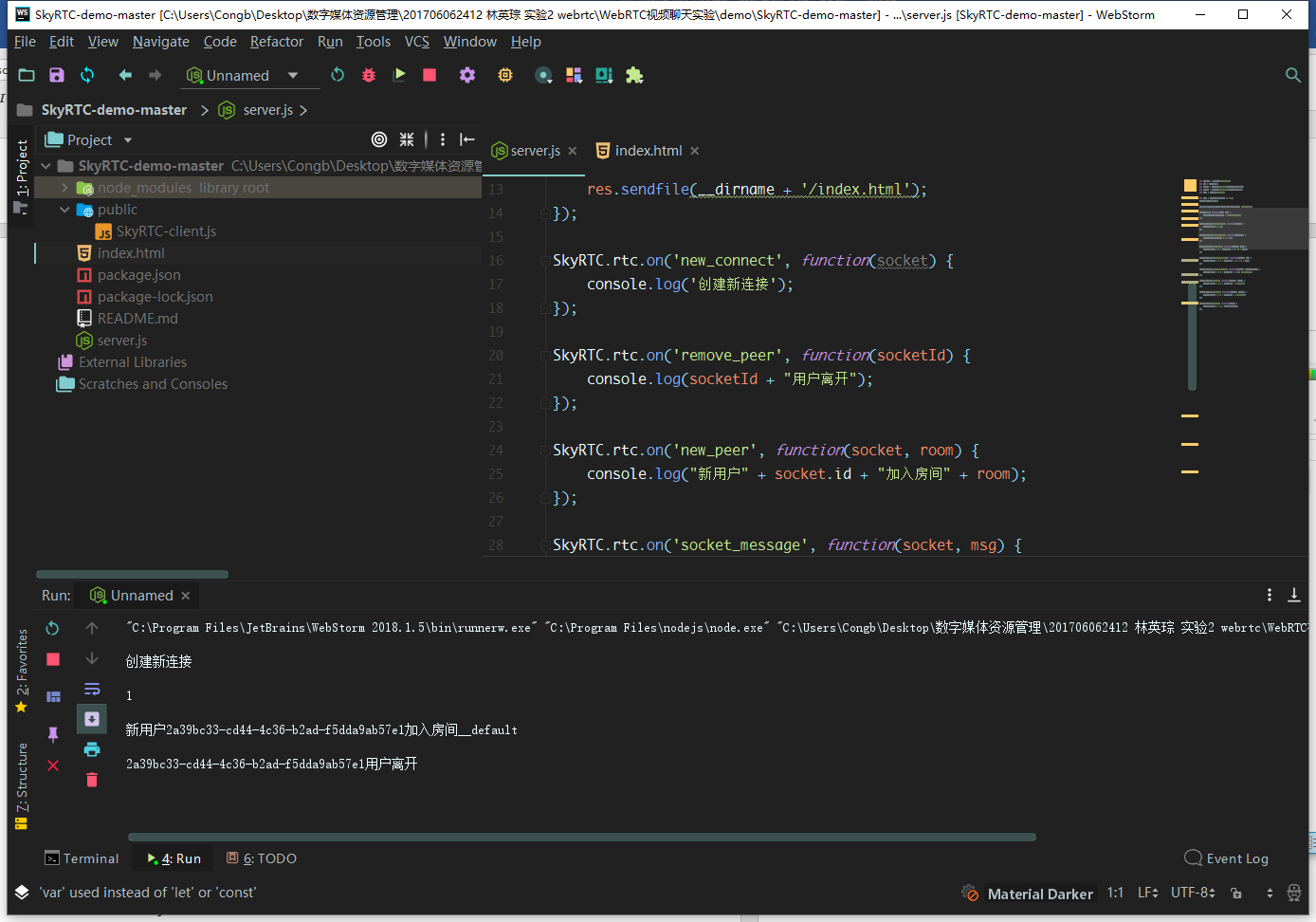


图 3‑4 启动

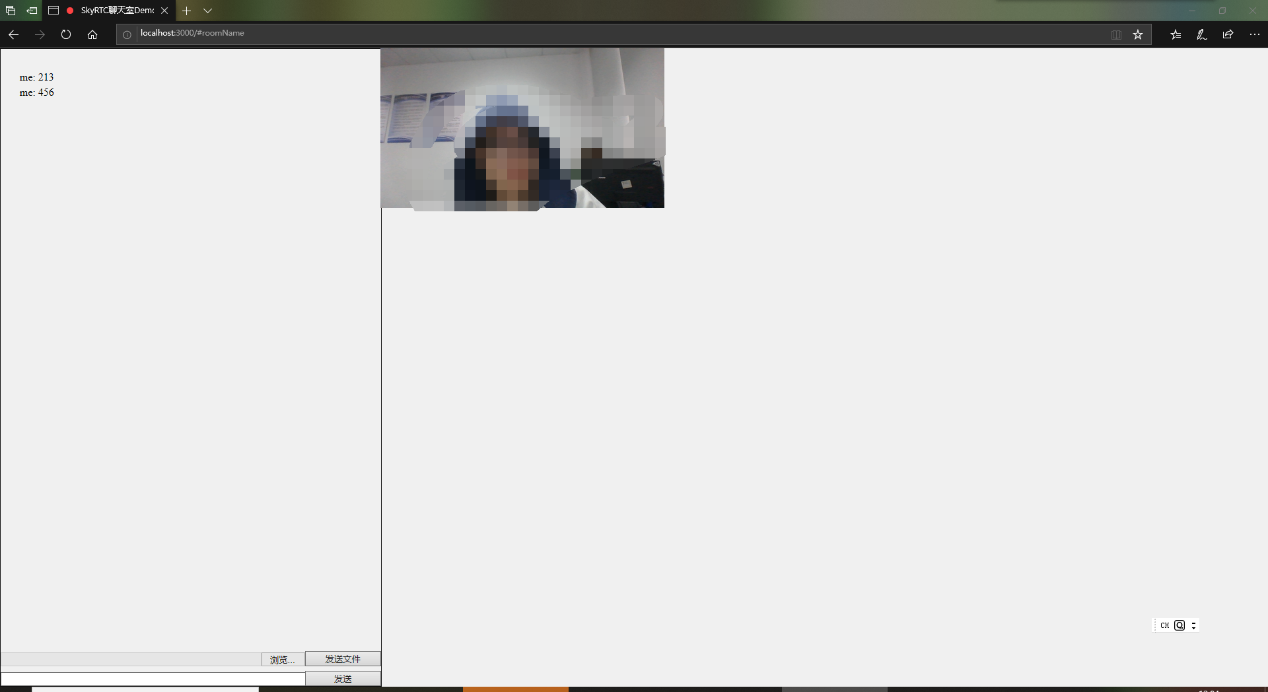


图 3‑5 效果

* 部署到服务端测试

我将SkyRTC部署到自己的服务器上以后，并不能正确地调用摄像头获取视频流，估计和我自己的个人服务器没有特殊配置有关，暂时搁置。

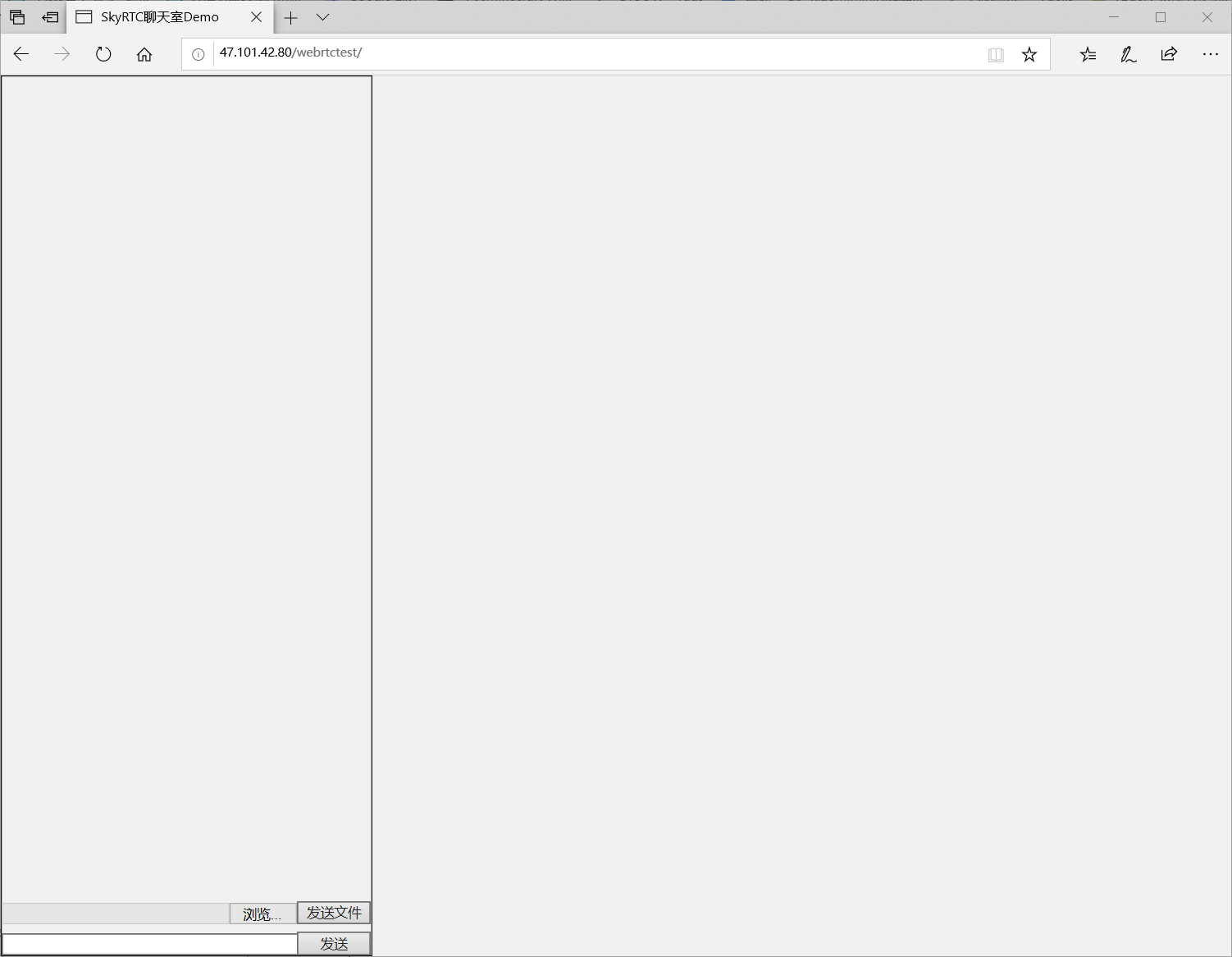


图 3‑6

### 自己编写的聊天室（期末总网站的一部分）

由于服务端SkyRTC和前端SkyRTC-client两个必须配合使用，年代久远且代码冗长、可读性一般，难以修改，且期末作业好像需要整合，因此决定自己编写一个完整的网站。

网站的外框架修改于我大二参加服务外包比赛的作品。

* 首页



图 3‑7 首页

* 视频流
* 聊天室

进入房间：

浏览器打开两个窗口，扮演两个用户

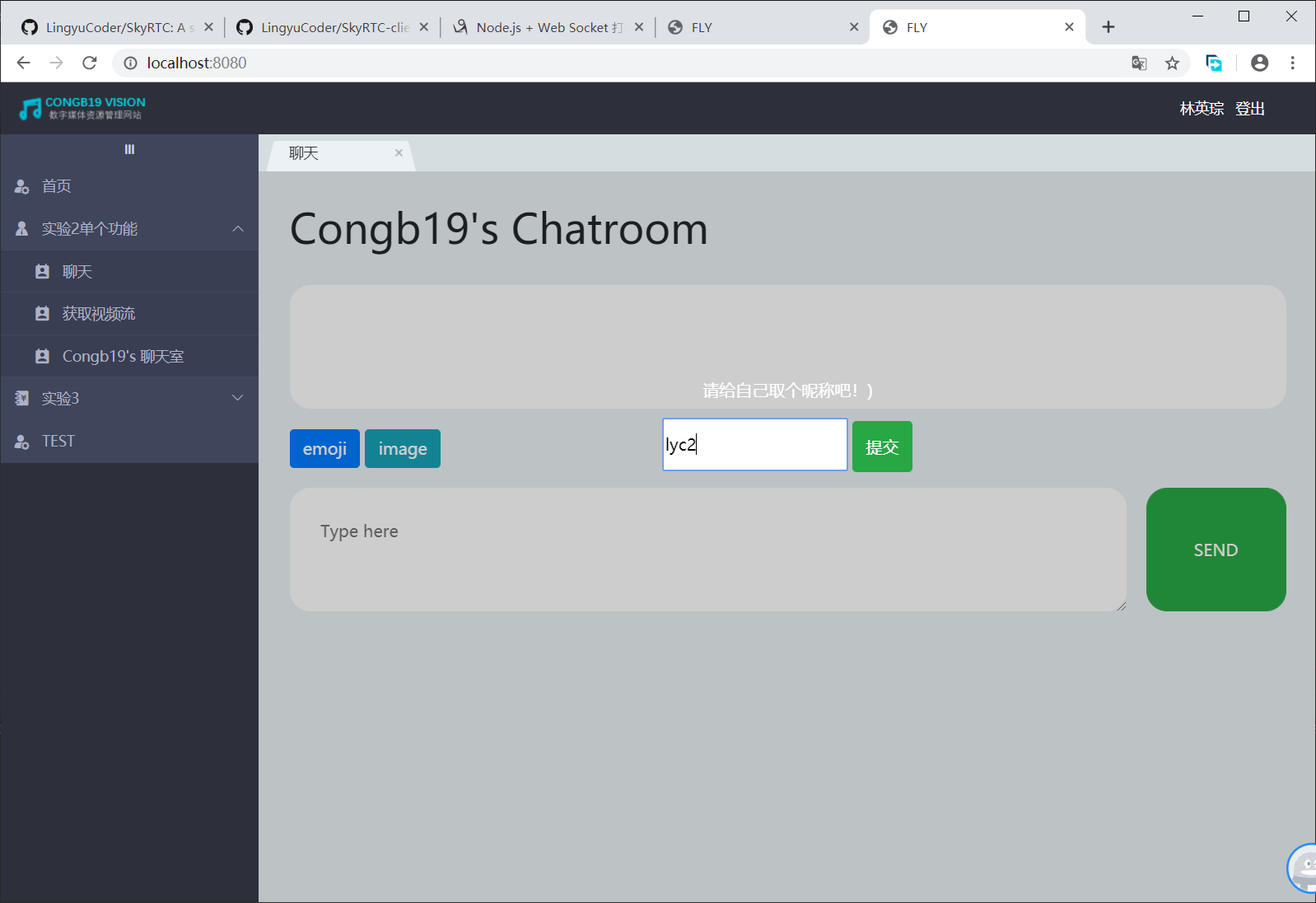


图 3‑8 进入房间

开始愉快的聊天：

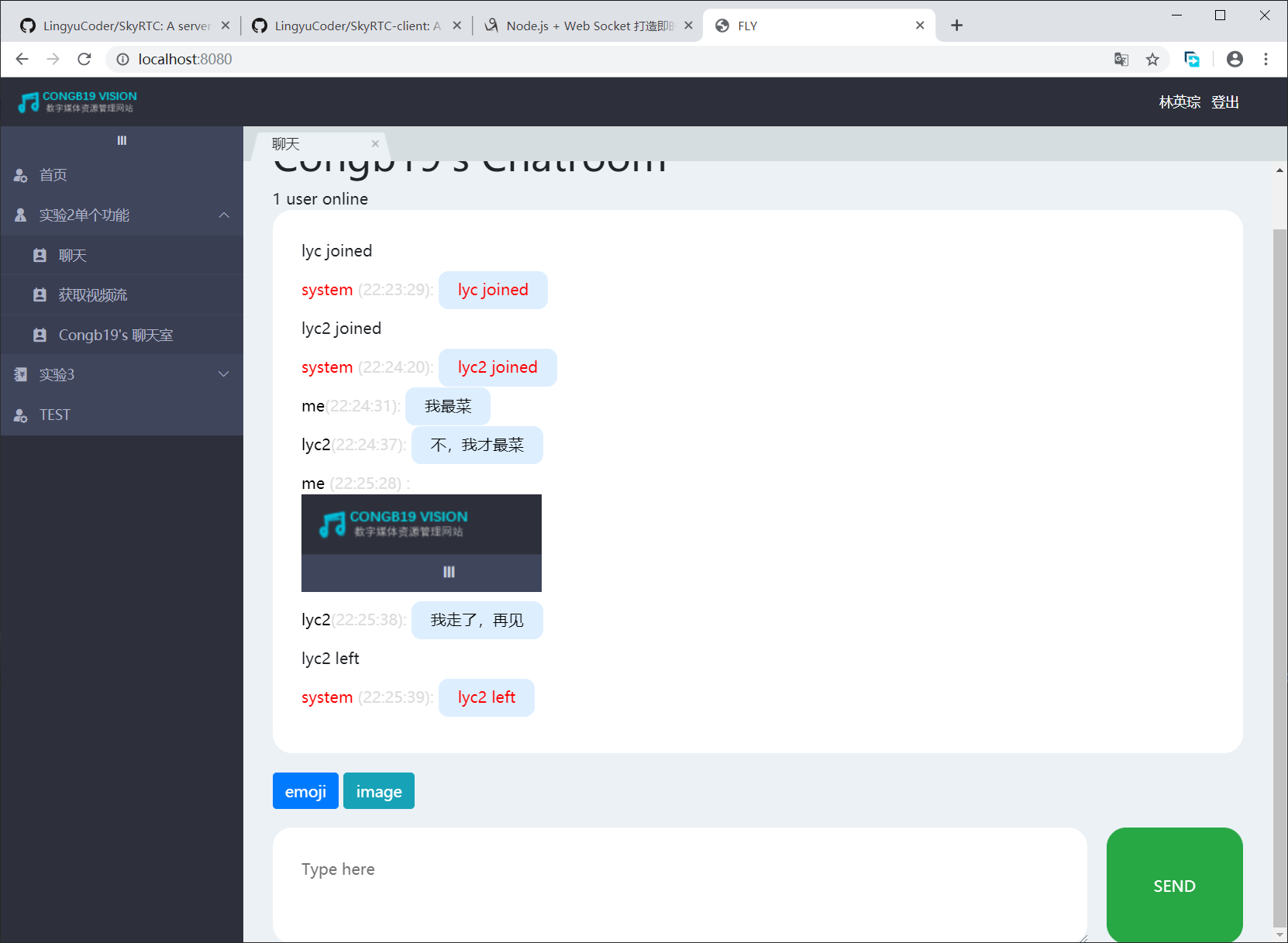


图 3‑9 聊天

# 问题与难点

## 相关WebRTC的熟悉

WebRTC (Web Real-Time Communications) 是一项实时通讯技术，它允许网络应用或者站点，在不借助中间媒介的情况下，建立浏览器之间点对点（Peer-to-Peer）的连接，实现视频流和（或）音频流或者其他任意数据的传输。

## 流媒体数据的传输和接收

* 事件：

MediaStream.onaddtrack：

这是addtrack事件在这个对象上触发时调用的事件处理器[EventHandler]，这时一个MediaStreamTrack对象被添加到这个流。

MediaStream.onended：

这是当流终止[ended]时触发的事件。

MediaStream.onremovetrack：

这是removetrack事件在这个对象上触发事调用的事件处理器[EventHandler]，这时一个对象从流上移除。

* 主要方法函数：

MediaStream.addTrack()：

存储传入参数 MediaStreamTrack 的一个副本。如果这个轨道已经被添加到了这个媒体流，什么也不会发生; 如果目标轨道为“完成”状态（也就是已经到尾部了），一个INVALID\_STATE\_RAISE异常会产生。

MediaStream.clone()：

返回这个MediaStream对象的克隆版本。返回的版本会有一个新的ID。返回给定ID的轨道。如果没有参数或者没有指定ID的轨道，将返回null。如果有几个轨道有同一个ID，将返回第一个。

MediaStream.removeTrack()

移除作为参数传入的 MediaStreamTrack。 如果这个轨道不在MediaStream对象中什么也不会发生； 如果目标轨道为“完成”状态，一个INVALID\_STATE\_RAISE异常会产生。

# 附录

## 代码仓库：

我将网站的代码开源至github仓库：

<https://github.com/Congb19/WEBRTC_DEMO_CONGB19>

## 调用摄像头基础js代码：

Navigator.getUserMedia()方法提醒用户需要使用音频（0或者1）和（0或者1）视频输入设备，比如相机，屏幕共享，或者麦克风。如果用户给予许可，successCallback回调就会被调用，MediaStream对象作为回调函数的参数。

但是，该特性已经从 Web 标准中删除，虽然一些浏览器目前仍然支持它，但也许会在未来的某个时间停止支持。

这里改用MediaDevices.getUserMedia()。它会提示用户给予使用媒体输入的许可，媒体输入会产生一个MediaStream，里面包含了请求的媒体类型的轨道。此流可以包含一个视频轨道（来自硬件或者虚拟视频源，比如相机、视频采集设备和屏幕共享服务等等）、一个音频轨道（同样来自硬件或虚拟音频源，比如麦克风、A/D转换器等等），也可能是其它轨道类型。

它返回一个 Promise 对象，成功后会resolve回调一个 MediaStream 对象。若用户拒绝了使用权限，或者需要的媒体源不可用，promise会reject回调一个 PermissionDeniedError 或者 NotFoundError 。

MediaStream 接口是一个媒体内容的流。一个流包含几个轨道，比如视频和音频轨道。

另外，更换原代码中的getElementById()为更新的querySelector()，用来返回文档中匹配指定 CSS 选择器的一个元素。该方法比getElementById()具有更高的效率。

$(function(){

module.messageCallBack.getInfo = function(){

console.log(this);

}

});

var constraints = { audio: true, video: { width: 1280, height: 720 } };

navigator.mediaDevices.getUserMedia(constraints)

.then(function(mediaStream) {

var video = document.querySelector('video');

video.srcObject = mediaStream;

video.onloadedmetadata = function(e) {

video.play();

};

})

.catch(function(err) { console.log(err.name + ": " + err.message); }); // 总是在最后检查错误

## 参考文档

<https://developer.mozilla.org/zh-CN/docs/Web/API/WebRTC_API>

<https://developer.mozilla.org/en-US/docs/Web/API/Document/querySelector>

<https://developer.mozilla.org/zh-CN/docs/Web/API/WebSocket>