WebRTC介绍

- WebRTC即web real-time communication，web实时通信技术。
- 谷歌的一个开源项目，目前正由w3c和IETF制定行业标准
- 它的设计目标就是让浏览器能够具有实时通信能力，而无需安装任何插件
- 封装了大量的多媒体处理和传输技术，并将这些技术以JavaScript接口的方式暴露给Web应用开发者，能够让开发者将精力更多的放到产品的设计上和实现细节上。
- 特性：开源、实时通信、音视频通信、基于web、P2P、安全、去插件、自适应算法、通话质量高。

移动终端编程

欧阳建权 教授
工科楼N211
Email：oyjq@xtu.edu.cn

助教：王彦斌　工科楼N504
Email:tuduweb@qq.com

WebRTC入门及实战

词汇介绍

WebRTC（Web Real-Time Communication）即网页即时通信，是一个支持网页浏览器进行实时语音对话或视频对话的API。封装在较新版本的浏览器中。

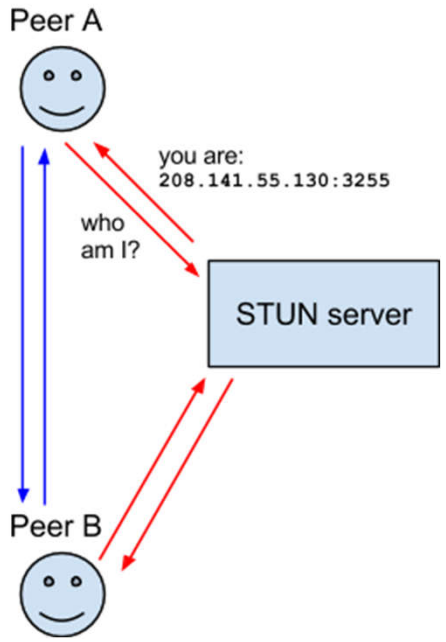
WebSocket是一种在单个TCP连接上进行全双工通信的协议。在WebSocket 中，浏览器和服务器只需要完成一次握手，两者之间就直接可以创建持久性的连接，并进行双向数据传输。

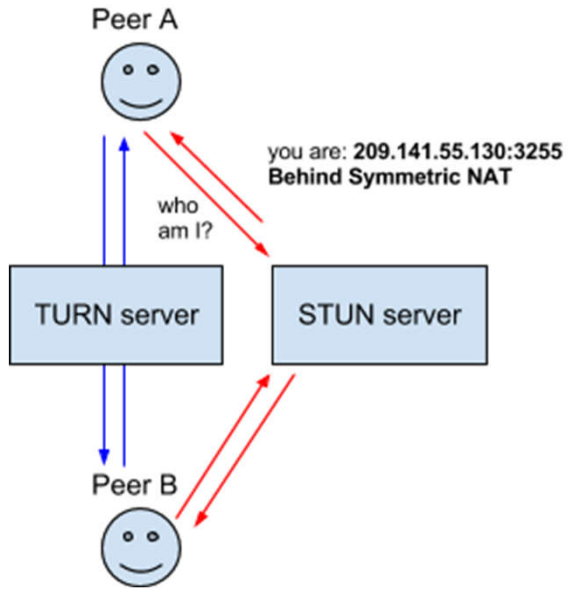
通话建立流程 https://segmentfault.com/a/1190000020780854

如用户A(浏览器) 想和用户 B (浏览器)进行音视频通话：
1.A、B 都连接信令服务器（ws）；
2.A 创建本地视频，并获取会话描述对象（offer sdp）信息；
3.A 将 offer sdp 通过 ws 发送给 B；
4.B 收到信令后，B 创建本地视频，并获取会话描述对象（answer sdp ）信息；
5.B 将 answer sdp 通过 ws 发送给 A；
6.A 和 B 开始打洞，收集并通过 ws 交换 ice 信息；
7.完成打洞后，A 和 B 开始为安全的媒体通信协商秘钥；
8.至此， A 和 B 可以进行音视频通话。

ws: WebSocket 一种长连接通信方式信令服务器
sdp:会话描述对象
ice: 用来在NAT上打洞的东西

WebRTC介绍


Peer A
you are:
208.141.55.130:3255
who
am I?
STUN server
Peer B


Peer A
you are: 209.141.55.130:3255
Behind Symmetric NAT
who
am I?
TURN server
STUN server
Peer B

WebRTC

词汇

SDP：

是一种描述连接的多媒体内容的标准，如分辨率、格式、编解码器、加密等，以便数据传输时双方能够相互理解。本质上，**这是描述内容的元数据**，而不是媒体内容本身。因此，从技术上讲，SDP不是真正的协议，而是用于描述设备之间共享媒体的连接的数据格式。

ICE：

由于Peer对Peer直连的方式受多重因素的限制，比如防火墙，公共IP,路由等。因此需要一个灵活的框架去除这些因素的影响，允许web浏览器与Peer节点连接的框架。而这个框架统称为ICE，ICE的背后使用的是STUN和TURN服务。

WebRTC

词汇

STUN（Session Traversal Utilities for NAT）

STUN主要解决两个问题：查询自己的公共IP和定位阻止Peer创建连接失败的原因。

PeerA要和PeerB通信，流程大致如下：

PeerA -> STUN : 我的公网IP是啥？

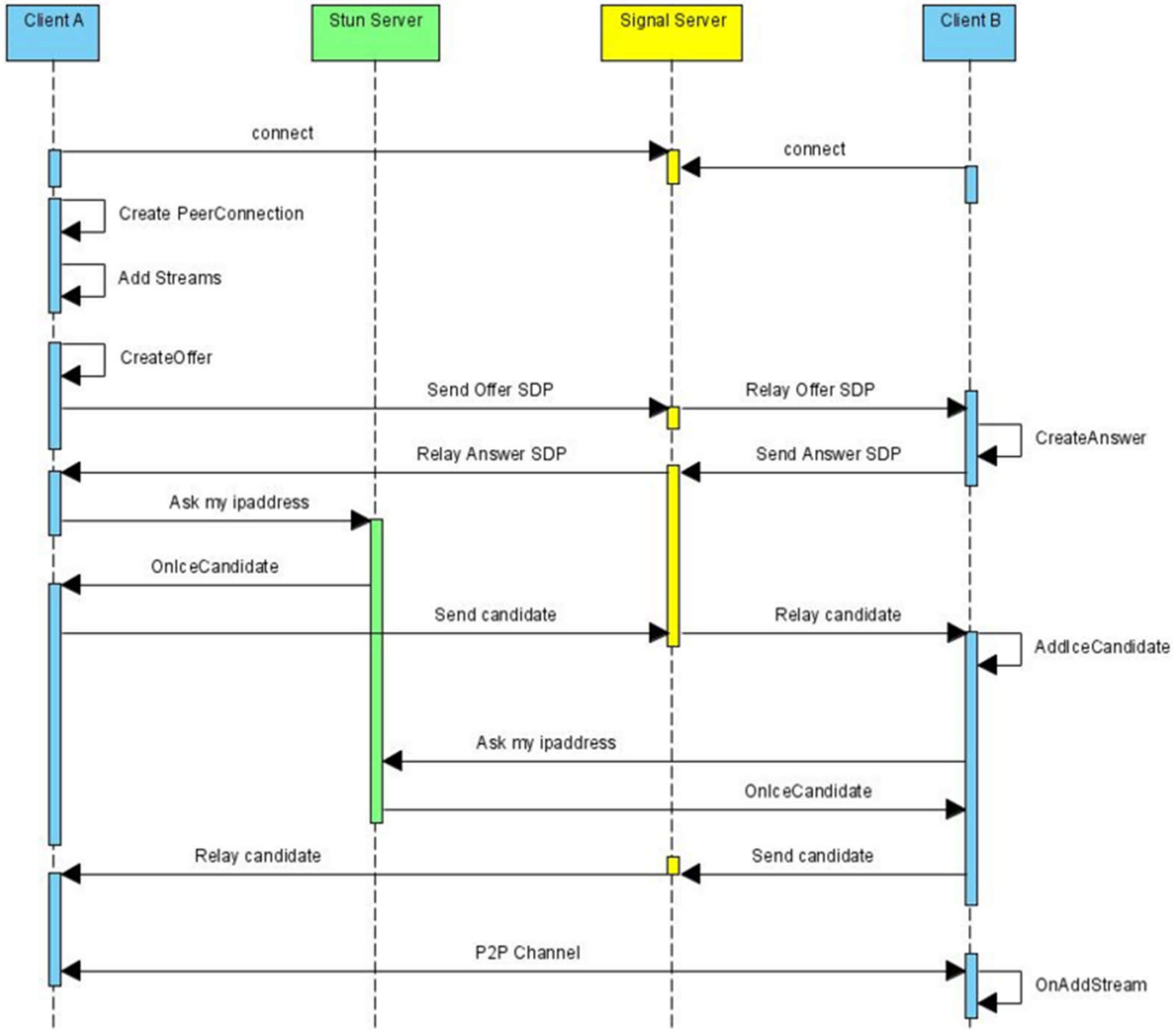
STUN -> PeerA: 你的公网IP是 42.49.109.169:3255

PeerB -> STUN : 我的公网IP是啥？

STUN -> PeerB: 你的公网IP是 42.49.109.169:3253

通过询问STUN后Peers就知道自己公网通信地址，就可以顺利的和其他Peer创建连接了。

TRUN(Traversal Using Relays around NAT)
使用TURN遍历意味着通过打开一个与TURN服务器的连接并通过该服务器转发所有信息来绕过Symmetric NAT限制。PeerA创建一个与TURN服务器的连接，并告诉其他的Peer发送数据包到服务器，然后


Client A
Stun Server
Signal Server
Client B
connect
connect
Create PeerConnection
Add Streams
CreateOffer
Send Offer SDP
Relay Offer SDP
CreateAnswer
Relay Answer SDP
Send Answer SDP
Ask my ipaddress
OnIceCandidate
Send candidate
Relay candidate
AddIceCandidate
Ask my ipaddress
OnIceCandidate
Relay candidate
Send candidate
P2P Channel
OnAddStream

信令服务器 环境安装

https://www.cnblogs.com/xiugeng/p/9611254.html

词汇

Node.js

Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行环境。

npm

Node.js的一种包管理器。类似Java语法中的maven，gradle，python中的pip

信令服务器 环境安装

词汇 用到的NPM包

express

简洁而灵活的 Node.js Web应用框架。

socket.io

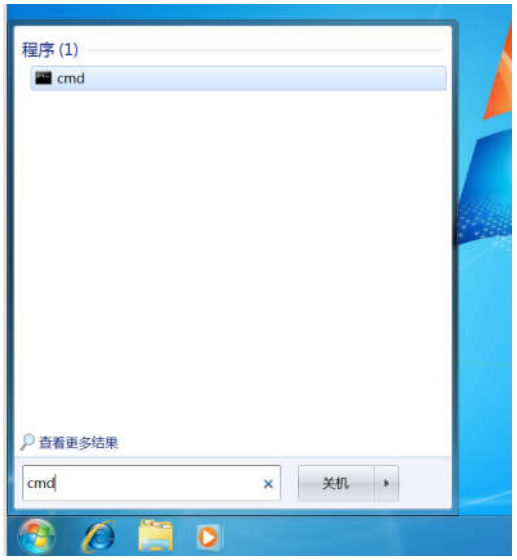
一个完全由JavaScript实现、基于Node.js、支持WebSocket的协议用于实时通信、跨平台的开源框架，它包括了客户端的JavaScript和服务器端的Node.js.

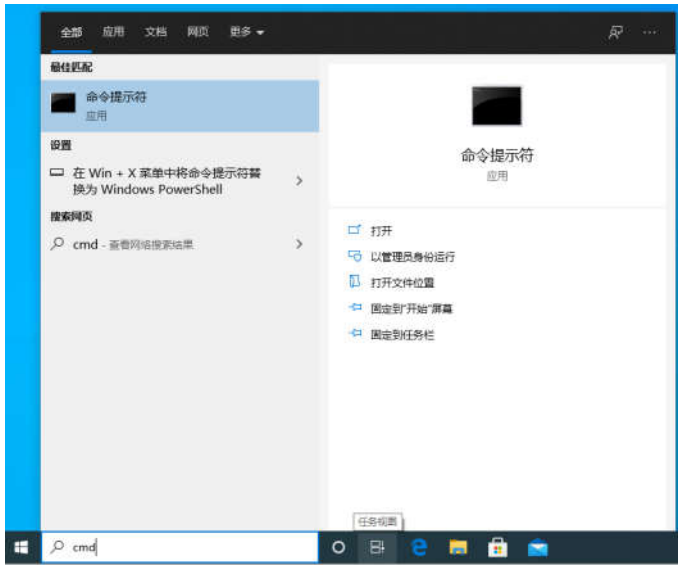
信令服务器 环境安装

基础概念

CMD工具（命令提示符）

打开左下角的开始菜单，输入cmd，回车，即可打开cmd工具。





命令

跟系统交互的一种方式。输入的命令必须都是英文字符，强烈建议在英文输入法下输入。

```
C:\Users\bin>node -v
'node' 不是内部或外部命令，也不是可运行的程序
或批处理文件。

C:\Users\bin>e r r o r
```

信令服务器 环境安装

1.查看系统是否安装过Node.js

2.安装Node.js

3.查看是否安装成功

4.通过NPM安装包:express

5.生成express工程

6.运行express项目

7.通过NPM安装包：socket.io

信令服务器 环境安装

查看系统是否安装过Node.js

打开CMD工具，输入 node –v 查看返回值。

```
C:\Users\bin>node -v
'node' 不是内部或外部命令，也不是可运行的程序
或批处理文件。

C:\Users\bin>e r r o r
```

```
Microsoft Windows [Version 10.0.19041.630]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\bin>node -v
v12.19.0
```

信令服务器 环境安装

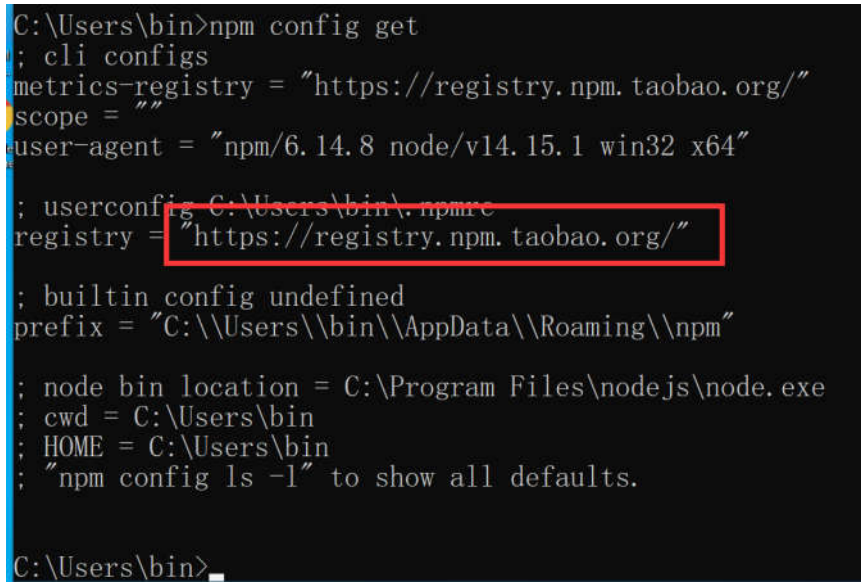
安装Node.js

修改npm源到国内地址，修改全局&缓存目录：

npm config set registry https://registry.npm.taobao.org

npm config set prefix "D:\nodejs\node_global"

npm config set cache "D:\nodejs\node_cache"



信令服务器 环境安装

安装Node.js & 查看是否安装成功

下载地址：

https://nodejs.org/zh-cn/

选择 长期支持版(LTS)

Win7的同学选择 其他下载>以往的版本> Node.js 10.x

再根据自己系统的架构选择x86 or x64

node-v10.23.0-x64.msi
node-v10.23.0-x86.msi

重新启动（关闭&再打开）CMD工具，

- 输入 node –v 查看返回值。
- 输入npm 查看返回值。

信令服务器 环境安装

创建express项目

新建一个目录：比如C盘（我的虚拟机只有一个盘）：

cd C:\

mkdir project

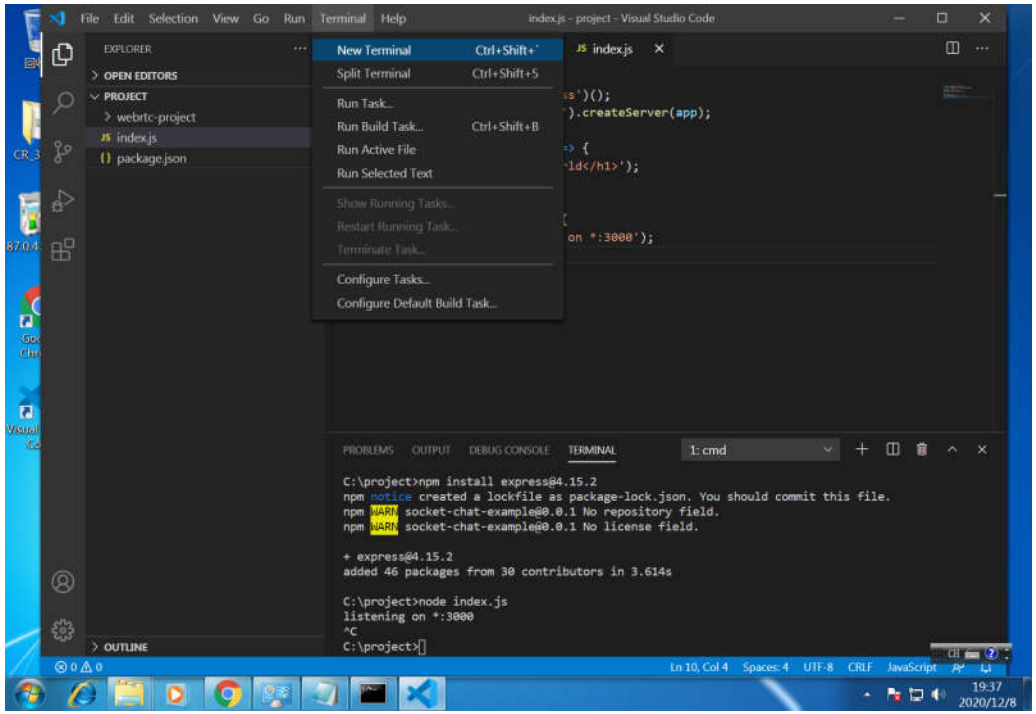
cd project

新建一个package.json文件：

```
{
  "name": "socket-chat-example",
  "version": "0.0.1",
  "description": "my first socket.io app",
  "dependencies": {}
}
```

Terminal->New Terminal

npm install express



信令服务器 环境安装

创建express项目

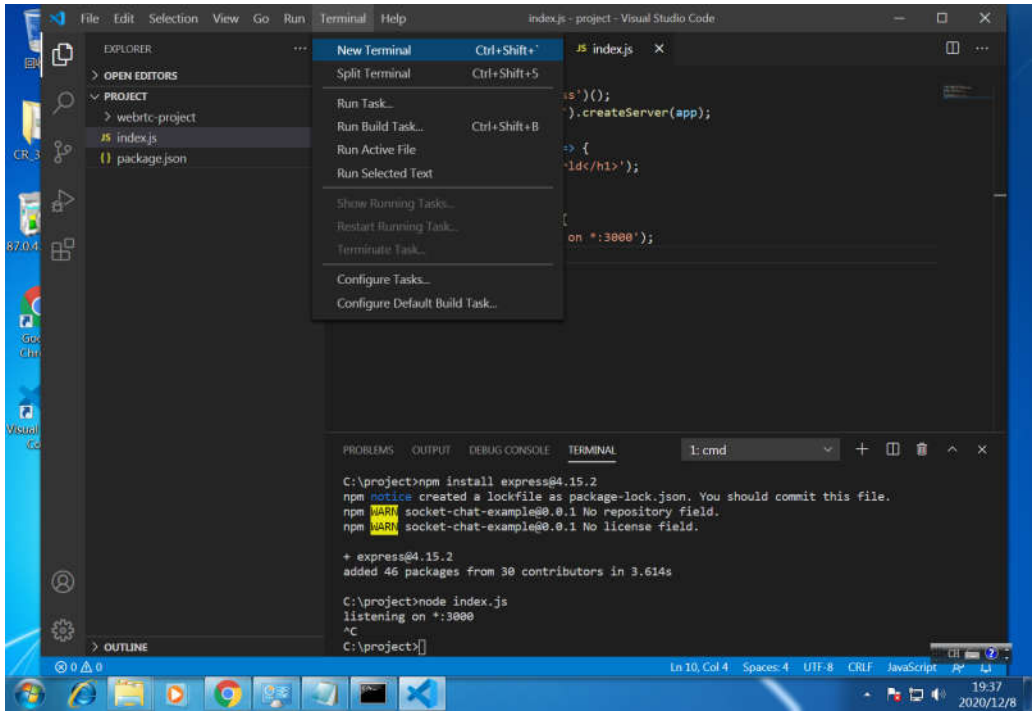
新建一个package.json文件：

```
{
 "name": "socket-chat-example",
 "version": "0.0.1",
 "description": "my first socket.io app",
 "dependencies": {}
}
```

菜单 Terminal->New Terminal

在下方弹出的界面中输入：

npm install express



信令服务器 环境安装

修改index显示内容

在当前目录新建一个文件index.html

```
<!doctype html>
<html>
<head>
<title>Socket.IO chat</title>
<style>
* { margin: 0; padding: 0; box-sizing: border-box; }
body { font: 13px Helvetica, Arial; }
form { background: #000; padding: 3px; position: fixed; bottom: 0; width:
100%; }
form input { border: 0; padding: 10px; width: 90%; margin-right: 0.5%; }
form button { width: 9%; background: rgb(130, 224, 255); border: none;
padding: 10px; }
#messages { list-style-type: none; margin: 0; padding: 0; }
#messages li { padding: 5px 10px; }
#messages li:nth-child(odd) { background: #eee; }
</style>
</head>
<body>
<ul id="messages"></ul>
<form action="">
<input id="m" autocomplete="off" /><button>Send</button>
</form>
</body>
</html>
```

信令服务器 环境安装

修改index显示内容

修改index.js的内容

```
// app.get('/', (req, res) => {
//   res.send('<h1>Hello world</h1>');
// });
app.get('/', (req, res) => {
res.sendFile(__dirname + '/index.html');
});
```

Ctrl+C 退出当前监听，按方向键↑（或直接输入node index.js）重新打开服务

刷新网页 127.0.0.1:3000

内容更新。

信令服务器 环境安装

通过NPM安装socket.io

Ctrl+C 退出当前监听

运行以下指令：

```
npm install socket.io
```

修改index.js：

```
var app = require('express')();
var http = require('http').createServer(app);
var io = require('socket.io')(http);

app.get('/', (req, res) => {
res.sendFile(__dirname + '/index.html');
});
io.on('connection', (socket) => {
console.log('a user connected');
});
http.listen(3000, () => {
console.log('listening on *:3000');
});
```

修改index.html，在</body>后添加：

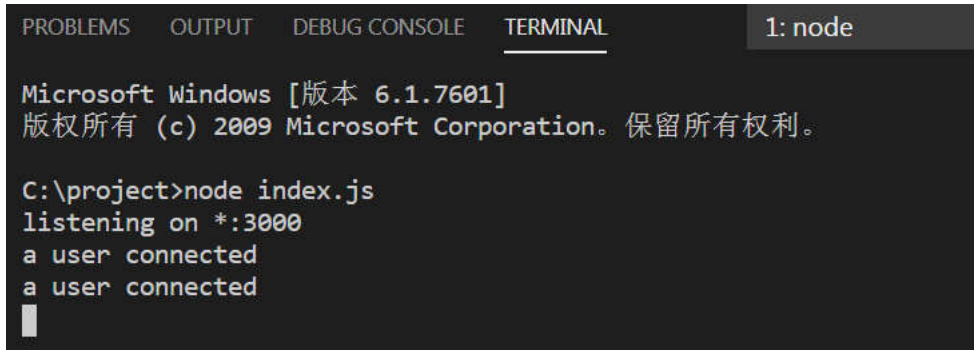
```
<script
src="//cdnjs.cloudflare.com/ajax/libs/socket.io/3.0.4/socket.io.js"></script>
<script>
var socket = io();
</script>
```

信令服务器 环境安装

通过NPM安装socket.io

使用以下命令执行:

node index.js



有连接那么也有失去连接:

```
io.on('connection', (socket) => {
  console.log('a user connected');
  socket.on('disconnect', () => {
    console.log('user disconnected');
  });
});
```

信令服务器 基础教程

发送消息到服务器

修改index.html

```
<script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
<script>
    $(function () {
    var socket = io();
    $('form').submit(function(e) {
        e.preventDefault(); // prevents page reloading
        socket.emit('message', $('#m').val());
        $('#m').val('');
        return false;
    });
    });
</script>
```

修改index.js，加入对’chat message’事件的监听

```
io.on('connection', (socket) => {
    socket.on('message', (msg) => {
        console.log('message: ' + msg);
    });
});
```

信令服务器 基础教程

消息互发：通过信令服务器转发

修改index.js，加入对'chat message'事件的监听

```
io.on('connection', (socket) => {
    socket.on('message', (msg) => {
        io.emit(message', msg);
    });
});
```

修改index.html

```
<script>
    $(function () {
        var socket = io();
        $('form').submit(function(e){
        e.preventDefault(); // prevents page reloading
        socket.emit('message', $('#m').val());
        $('#m').val('');
        return false;
        });
        socket.on('message', function(msg){
        $('#messages').append($('<li>').text(msg));
        });
    });
</script>
```

信令服务器 基础教程

其它方式

修改index.js，按以下方式修改，发送到除发送者外的端。

```
socket.on('message', (msg) => {
  //io.emit('message', msg);
  socket.broadcast.emit('message', msg);
  //console.log('message: ' + msg);
});
```

修改index.html，在index.html（前端）中，也能监听连接，失去连接事件。

```
var socketID;
socket.on('connect', function () {
    socketID = socket.io.engine.id;
    console.log('hello world!' + socketID);
});
```

Nginx反向代理服务器 环境安装

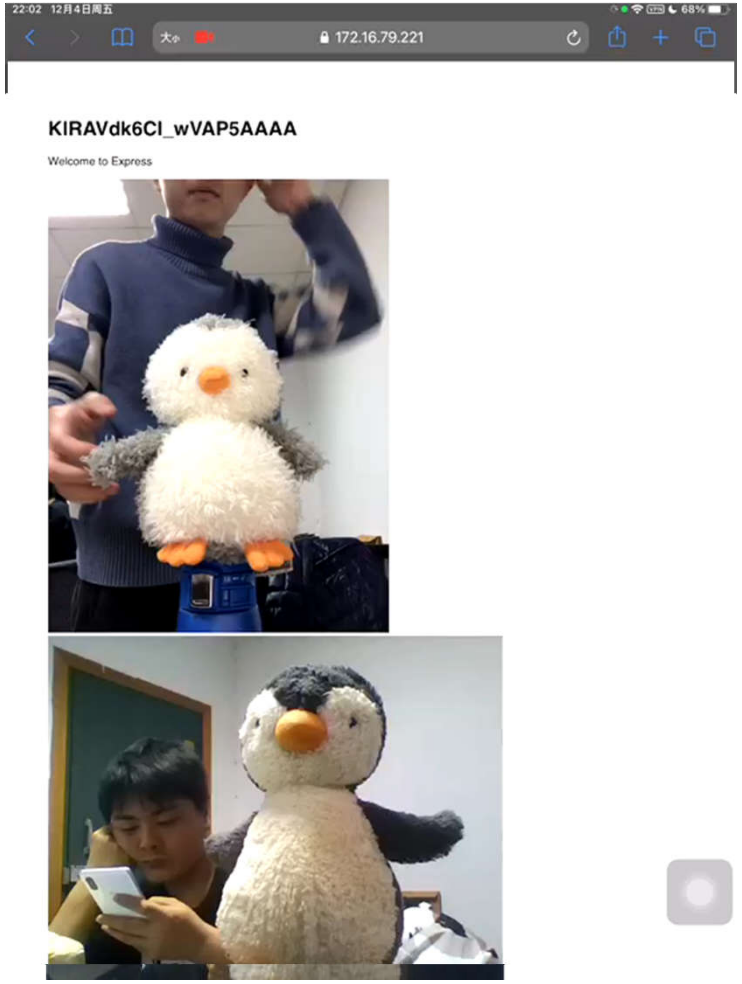
Nginx反向代理服务器安装

作用：

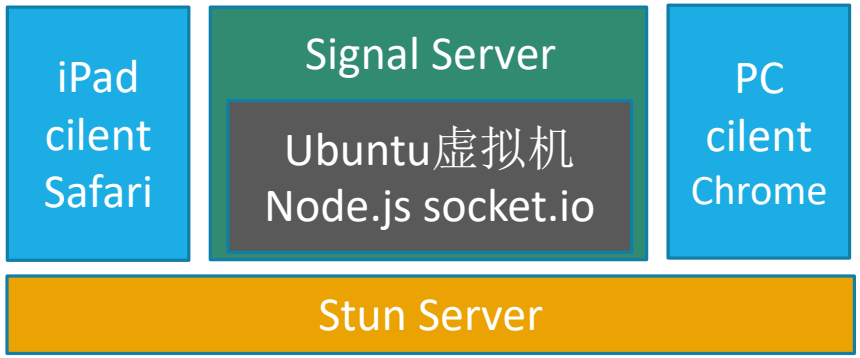
代理https请求，让我们的项目能在本地（127.0.0.1）外访问。

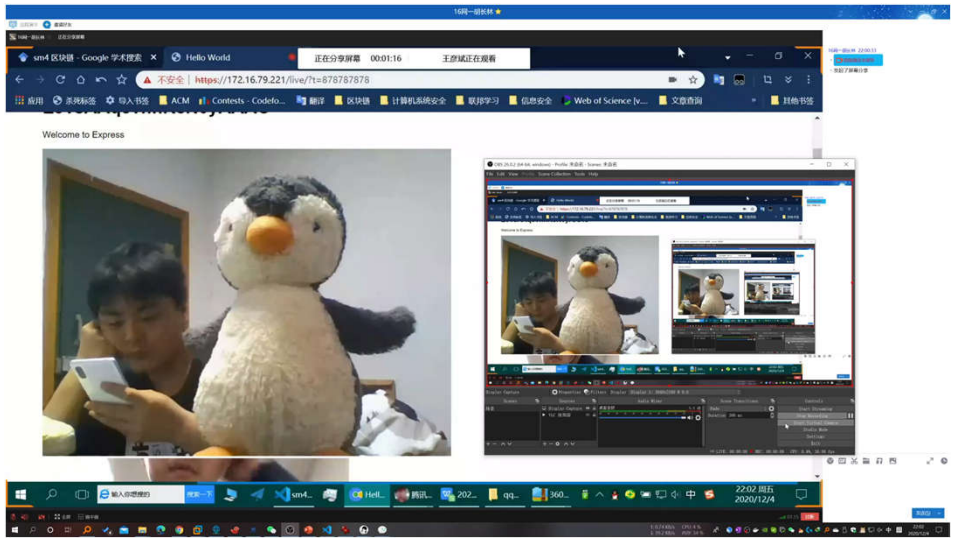
实战——一对一视频通话

效果展示



程序架构





创建WebRTC页面

- 1、设计index页面：打开views下面的index.ejs，添加video元素
  - <video autoplay controls id="video"></video>
  - <button onclick="showVideo()">打开摄像头</button>
- 2、检查兼容性：
  - function canGetUserMediaUse() {
  - return !!(navigator.mediaDevices.getUserMedia || navigator.webkitGetUserMedia || navigator.mozGetUserMedia || navigator.msGetUserMedia);
  - }

创建WebRTC页面

3、访问用户媒体设备的兼容方法

```
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);
        }
    }
```

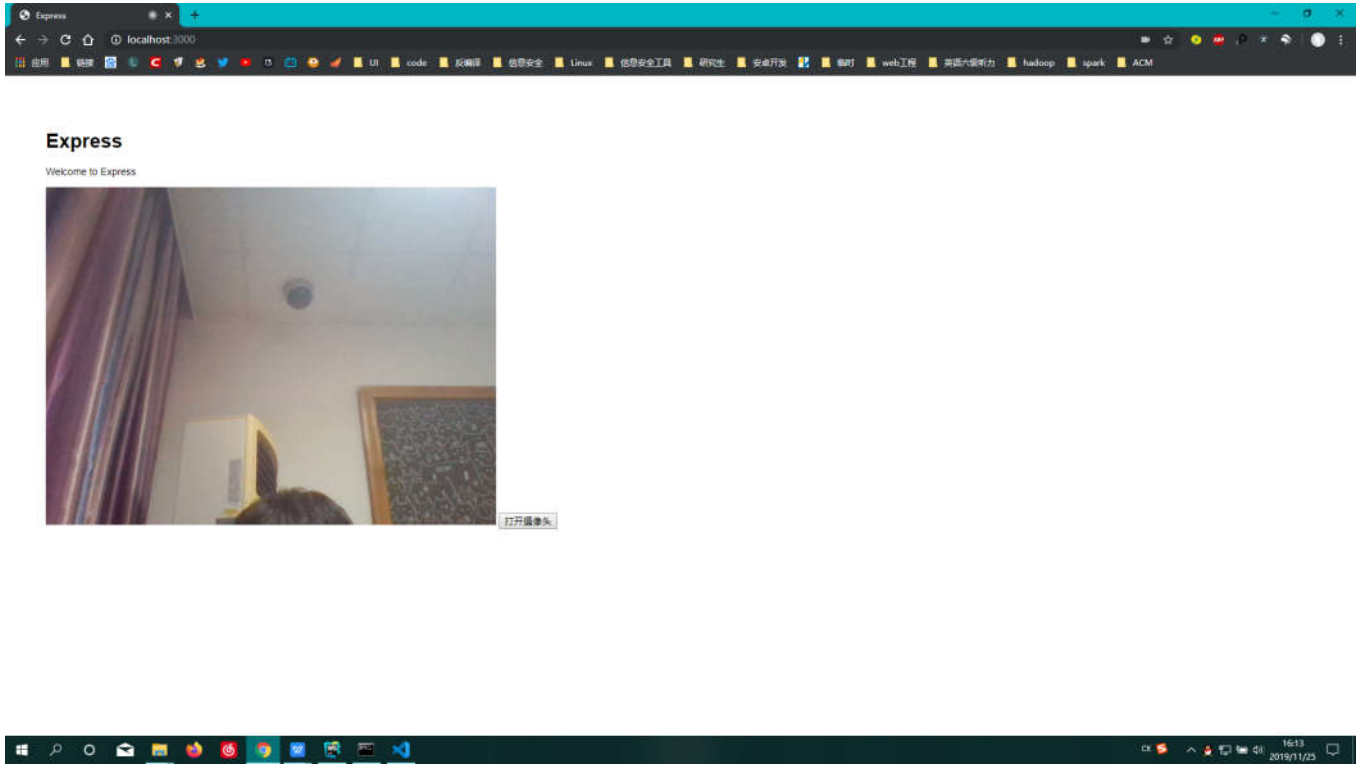
创建WebRTC页面

- 4、获取媒体流并在video中播放

```
function showVideo() {
      if (canGetUserMediaUse()) {
         getUserMedia({
            video:true,
            audio:true
         },function (stream) {
            mediaStream = stream;
            const video = document.getElementById('video');
            video.srcObject = stream;
            video.play();
         },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');  }
   }
```

创建WebRTC页面

*4、打开浏览器，访问http://localhost:3000/。点击打开摄像头，即可观看到结果。



*练习：打开设备的摄像头，观看到自己

媒体流的约束

- 可以在getUserMedia方法中添加参数来限制获取到的媒体
  - 设置video:false，可禁止浏览器获取视频图像
  - 设置audio:false，可禁止浏览器获取音频
- 注意：在开发WebRTC应用的时候可能会遇到音频反馈现象，当麦克风捕捉到你的声音再通过扬声器播放出来，会产生一个永无止境的回响。在开发过程中建议暂时关闭本地的音频或者设置video播放器为静音，即添加muted属性。
- 除了可以简单设置video和audio的开闭外，还可以进行更复杂的配置，如调整获取视频的最低分辨率、帧速率、视频宽高比等。

- 练习：采集音频和无声音的视频