**webRTC 库**

**使用及开发说明**

# 功能与项目结构

## 功能

库仅进行前端封装，不涉及后端实现。提供如下功能：

1. 拍照：将预览的视频画面绘制到canvas标签。
2. 录像：录制音视频（至少选择其一）。允许设置时间片长度，当时间片结束或录制停止，订阅事件回调方法将得到一个作为输入参数的Blob对象。
3. 录屏：在启动录制之前，需要连接扩展程序（chrome 扩展可用， firefox 扩展未开发），请求用户授权。同录像一样，通过回调取得录制数据。
4. p2p 音视频通讯： 通过交换信令，建立点对点连接，传输流媒体。
5. p2p 数据传输(未开发完成)：用于传输文本、图像、视频等。根据不同的实现方式可能需要做向下兼容。

## 目录结构

src 扩展程序(extension)与webRTC封装源码(src)，原生js编写 :

src/src/core webRTC封装的核心部分：

1. base 文件夹提供硬件信息、流媒体相关的基础模块。
2. 基于base开发的进阶功能，例如录制等。

src/src/constants 常量管理：

1. errorCode: 错误码。
2. methods: 存储私有方法名。使部分方法对外透明。
3. property: 存储私有属性名。使部分属性对外透明。

src/src/utils 内部工具方法。

example 演示用例。基于vue 2.x编写。

## 类继承关系



# 使用指南

目前webRTC库还未做成单独打包的形式，暂不提供以js文件的形式引入。必须将源码复制到项目中，使用转码、打包工具同项目一起编译打包。

模块分离了需要做兼容的部分，但是尚未做向下兼容。部分代码考虑到chrome优先，使用了工作草案中提到，不受支持的方法。(firefox 55已经不推荐使用，会因此失去对码流的控制)。

## 扩展程序

扩展程序用户获取sourceId。目前仅完成chrome扩展。

## 音视频流方法相关说明

一切涉及到获取音视频流的方法，都是异步的。且调用者应提供从设备信息筛选出设备ID所需的参数。device类提供多种方式获取设备ID。从高优先级开始查找，找到目标返回结果，未找到使用次级参数继续查询。若最终没有找到目标，返回undefined。

取视频设备ID参数优先级: vid、pid > camNo 摄像头序号 > vLabel 设备标签 > facingMode 视频源方向

取音频设备ID 参数优先级：micNo 麦克风序号 > mLabel 设备标签

## 事件订阅

类向外暴露 \_rtcEvtsSubscribe 方法提供订阅事件，\_rtcEvtsUnsubscribe 解除订阅。支持批量订阅与批量解除，但一个事件只能绑定一个回调方法，先前的将被后绑定的覆盖。

\_rtcEvtsSubscribe({ pairs: obj }) obj是包含键名为事件名，值为回调方法的对象。

\_rtcEvtsUnsubscribe({ names: arr }) arr 是成员为事件名的数组。

## 演示用例

### 拍照

let tp = new TakePicture()

await tp.start({

canvasId: 'tpc', // canvas元素Id

facingMode: 'user',

videoId: 'tpv', // video元素Id

vLabel: this.camLabel

})

// 拍照

tp.takePicture({

w: 400, // 照片宽度

h: 300 // 照片高度

})

…

tp.close() // 关闭摄像头

### 录屏

let screen = new Screen()

/\*

1. 订阅 recDataAvailCallBack 事件获取录制数据

2. 订阅 recStopCallBack 事件进行预览操作(可选)

\*/

await screen.connExts({

cTimeout: 5, // 连接扩展超时时间(秒)

aTimeout: 15 // 请求授权超时时间(秒)

})

// 开始录制

await screen.start()

…

screen.close() // 关闭多媒体设备

### 录像

let recorder = new Rec()

/\*

1. 订阅 recDataAvailCallBack 事件获取录制数据

2. 订阅 recStopCallBack 事件进行预览操作(可选)

\*/

await recorder.start({

pid: '0401',

vid: '5986',

micNo: 2,

videoId: 'rec' // 预览video 标签ID

})

// 开始录制

recorder.rec({

duration: 6, // 录制时长

timeSlice: 3 // 时间片大小

})

…

recorder.close() // 关闭摄像头

### p2p监控

offer 端：

let offer = new MonOffer()

offer.init({

config: {

IceServers: [

{

urls: webRtcConfig.stunUrls

},

{

urls: webRtcConfig.turnUrls,

username: webRtcConfig.turnUser,

credential: webRtcConfig.credential

}

]

}

})

/\*

事件订阅

1. 订阅 pLocalSDPReady事件发送sdp 进行信令交换

2. 订阅 pOnIceCandidate事件发送ice candidate 进行信令交换

\*/

/\*

信令交换

1. websocket 收到sdp, 调用 offer. \_rtcPCAddSDP(data)方法设置远端sdp。

2. websocket 收到ice candidate, 调用 offer. \_rtcPCAddIceCandidate(data)方法添加ice candidate。

\*/

await offer.start({

vLabel: ‘Chicony USB 2.0 Camera (04f2:b5a7)’,

mLabel: ‘麦克风 (Realtek High Definition Audio)’,

frameRate: 10,

width: 480,

height: 360

})

…

offer.close() // 关闭多媒体设备及p2p连接

answer 端：

let answer = new MonAnswer()

answer.init({

config: {

IceServers: [

{

urls: webRtcConfig.stunUrls

},

{

urls: webRtcConfig.turnUrls,

username: webRtcConfig.turnUser,

credential: webRtcConfig.credential

}

]

}

})

/\*

事件订阅

1. 订阅 pLocalSDPReady事件发送sdp 进行信令交换

2. 订阅 pOnIceCandidate事件发送ice candidate 进行信令交换

\*/

/\*

信令交换

1. websocket 收到sdp, 调用 offer. \_rtcPCAddSDP(data)方法设置远端sdp。

2. websocket 收到ice candidate, 调用 offer. \_rtcPCAddIceCandidate(data)方法添加ice candidate。

\*/

…

answer.close() // 关闭多媒体设备及p2p连接

# 附录

## 订阅事件列表

### 录制事件

|  |  |
| --- | --- |
| 事件名 | 事件定义 |
| recDataAvail | 可获取录制数据 |
| recClosed | 录制连接已正常关闭 |
| recStop | 停止录制 |

事件参数及业务定义如下:

recDataAvail

|  |  |
| --- | --- |
| 接收参数 | 需要完成的工作 |
| Blob | 录制数据上传服务器 |

recClosed

|  |  |
| --- | --- |
| 接收参数 | 需要完成的工作 |
| 无 | 无 |

recStop

|  |  |
| --- | --- |
| 接收参数 | 可选的工作 |
| 无 | 合并文件进行本地预览（可选） |

### p2p事件

|  |  |
| --- | --- |
| 事件名 | 事件定义 |
| pLocalSDPReady | 本地sdp 设置完毕 |
| pOnIceCandidate | 正在采集本地ice candidate |
| pIceConnCompleted | ice 连接搜集 ice 完毕 |
| pIceConnConnected | ice 连接完成 |
| pIceConnDisconnected | ice 连接中断，有可能恢复 |
| pOnAddStream | 接收到远程流媒体 |

事件参数及业务定义如下:

pLocalSDPReady

|  |  |
| --- | --- |
| 接收参数 | 需要完成的工作 |
| 本地 sdp 对象 | 发送本地sdp到信令服务器 |

pOnIceCandidate

|  |  |
| --- | --- |
| 接收参数 | 需要完成的工作 |
| 一个ice candidate | 发送ice candidate到信令服务器 |

pIceConnCompleted

|  |  |
| --- | --- |
| 接收参数 | 需要完成的工作 |
| 无 | 无 |

pIceConnConnected

|  |  |
| --- | --- |
| 接收参数 | 需要完成的工作 |
| 无 | video标签绑定流媒体 |

pIceConnDisconnected

由于chrome 暂不支持查询连接状态，需要手动实现对连接异常中断的判断，主动关闭的一端应该对远端进行通知。

|  |  |
| --- | --- |
| 接收参数 | 需要完成的工作 |
| 无 | 非正常断开，连接中断报错 |

pOnAddStream

|  |  |
| --- | --- |
| 接收参数 | 需要完成的工作 |
| 流媒体数组 | 无 |

## 错误信息表

订阅errHandler 事件以获取关键性错误上报。

### MediaRecorder 错误

暂无

### RTCPeerConnection 错误

|  |  |  |  |
| --- | --- | --- | --- |
| code | 类型 | 错误信息 | 错误名 |
| -2000 | 错误 | 建立连接超时 | P2P\_ICECONN\_ESTABLISH\_TIMEOUT |
| -2001 | 错误 | 发起 sdp 协商失败 | P2P\_STARTNGO\_FAILED |
| -2002 | 错误 | ice 连接异常关闭 | P2P\_ICECONN\_FAILED |