

AVD SDK

Web 版开发指南
(版本 v3.0)

目 录

1	修订记录.....	3
2	概述.....	3
2.1	简介.....	3
2.2	面向的读者.....	4
2.3	获取 SDK DEMO.....	4
2.4	技术支持.....	4
3	编写说明.....	4
4	环境准备.....	4
4.1	工程准备.....	4
4.1.1	开发工具 HBUILDER.....	4
4.1.2	SDK DEMO 工程导入.....	5
4.1.3	SDK DEMO 工程运行.....	6
4.2	非工程环境运行.....	8
5	音视频基本功能开发流程.....	8
5.1	获取授权信息.....	8
5.2	引擎初始化.....	8
5.3	加入房间.....	9
5.4	开启/关闭音视频.....	10
5.5	房间回调事件.....	10
5.6	用户回调事件.....	11
5.7	订阅/取消订阅视频.....	14
5.8	音视频渲染.....	14
5.9	离开/关闭房间.....	15
6	附录.....	15
6.1	名词解释.....	15
6.2	其它开发工具.....	16
6.2.1	VSCODE.....	16
6.2.1.1	SDK DEMO 工程导入.....	16
6.2.1.2	SDK DEMO 工程运行.....	17

1 修订记录

修订日期	描述	作者	版本号
2019.6.3	初始文档	王群	3.0.0

2 概述

Open-AVD SDK 是支持多种终端(浏览器,Andorid,Ios, 嵌入 linux,Mac,Electron 等)进行音视频互动的解决方案,支持 H.264/H.265/VP8/VP9 多种视频编码,支持 ISAC, opus, ,G722, PCMA, PCMU 等多种音频编码,支持桌面共享能力,完全兼容 WEBRTC 音视频协议。

Open-AVD SDK 网络音视频平台由前 Cisco,Webex 等资深音视频工程师架构和设计,用于解决安防,医疗,金融,教育等各个行业的音视频互动需求。

1.1 简介

Open-AVD SDK 为移动、桌面和互联网应用提供一个完善的音视频及数据互动开发框架,屏蔽掉互动系统的复杂细节,对外提供较为简洁的 API 接口,方便第三方应用快速集成互动功能。

Open-AVD SDK Web 版提供如下功能:

- 浏览器全平台支持
- 实时高清语音通话
- 实时高清视频通话
- 桌面共享服务
- SIP、H323、RTSP、RTMP 等集成功能
- 服务器端旁路直播

1.2 面向的读者

本指南是提供给具有一定的 web 编程经验和了解面向对象概念的产品经理及程序员使用, Open-AVD SDK 已很好的封装了音视频相关的底层技术细节,因而读者不需要具备音视频开发方面的经验。

1.3 获取 SDK Demo

公司的 github 网址(<https://github.com/3tee>)上会提供各类基于 Open-AVD SDK 的实例 Demo,包括基本音视频能力 Demo,桌面共享直播 Demo 等。

1.4 技术支持

您在使用本 SDK 的过程中，遇到任何困难，请与我们联系，我们将热忱为您提供帮助。
您可以通过如下方式与我们联系。

技术支持工程师： 186XXXXXXXX

2 编写说明

本指南编写目的是为了帮助使用 AVD SDK 的用户快速搭建 SDK 的开发环境、熟悉开发流程、掌握 SDK 开发功能接口而编写的。

本指南基于 Web 开发 IDE HBuilder，导入 baseVideo Demo 进行最简单的音视频能力进行编写，如果需要更多的功能，请参阅 SDK API 接口开发手册。

3 环境准备

3.1 工程准备

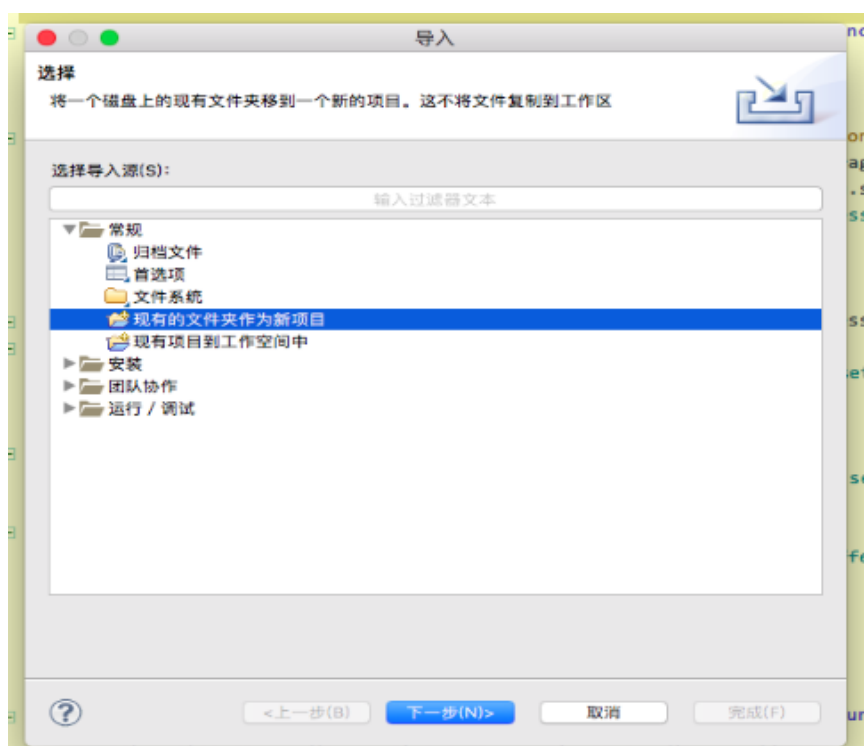
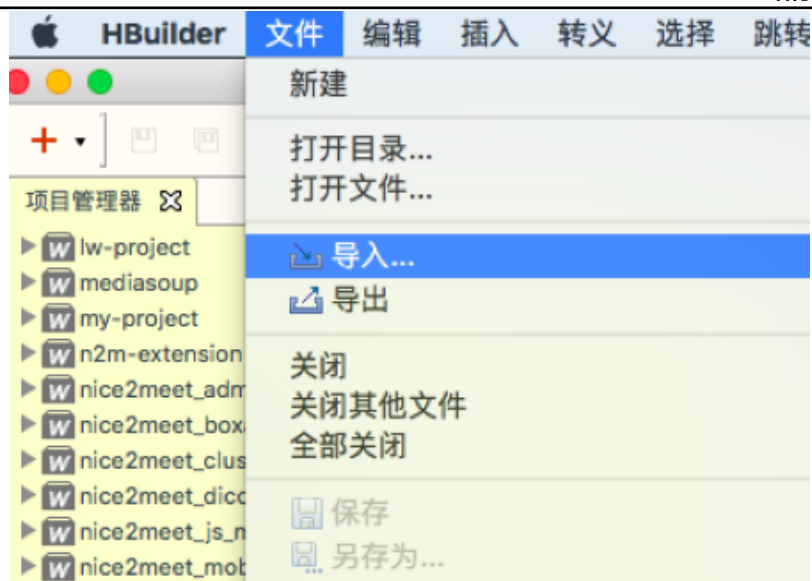
3.1.1 开发工具 HBuilder

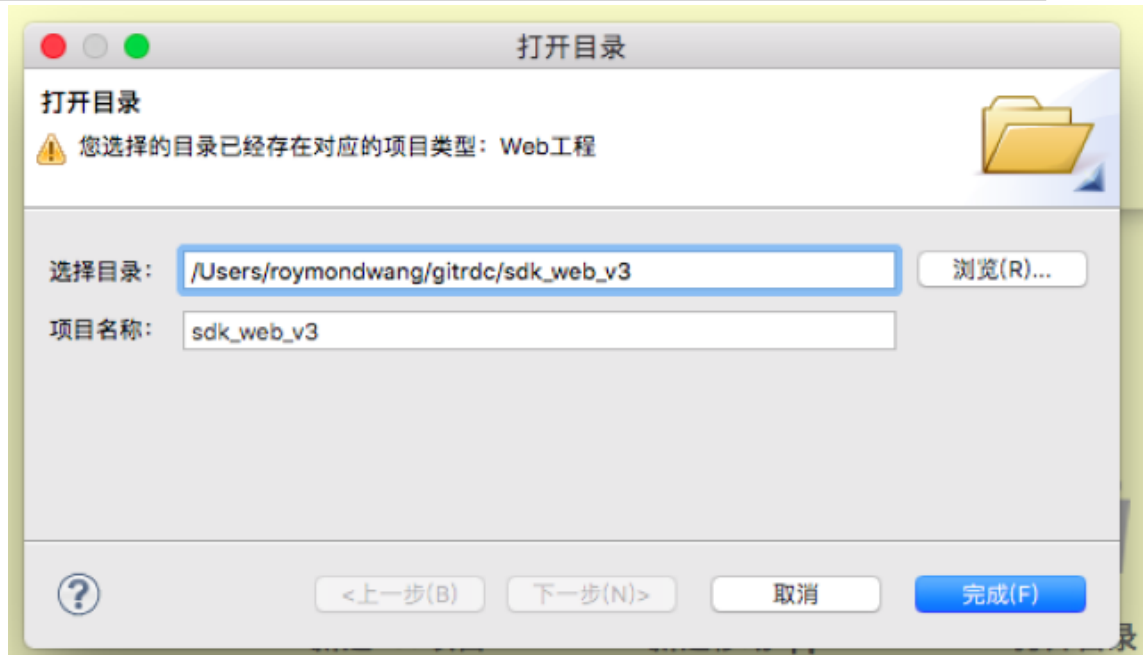
本指南默认的开发工具为 HBuilder(<http://www.dcloud.io/>), 它是DCloud推出的一款支持HTML5的Web开发IDE，其内置了 web 服务器，运行SDK Demo 非常方便。

如果您正在使用 vscode，可参考附录中“其它开发工具”节。

3.1.2 SDK Demo 工程导入

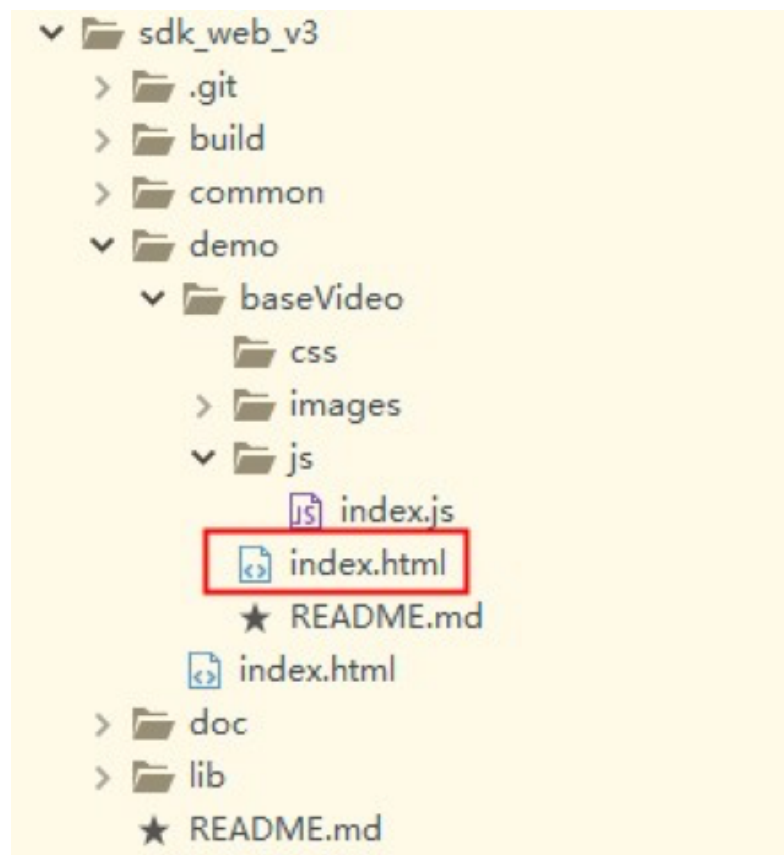
2.3 中下载的 SDK Demo，通过下面截图导入：

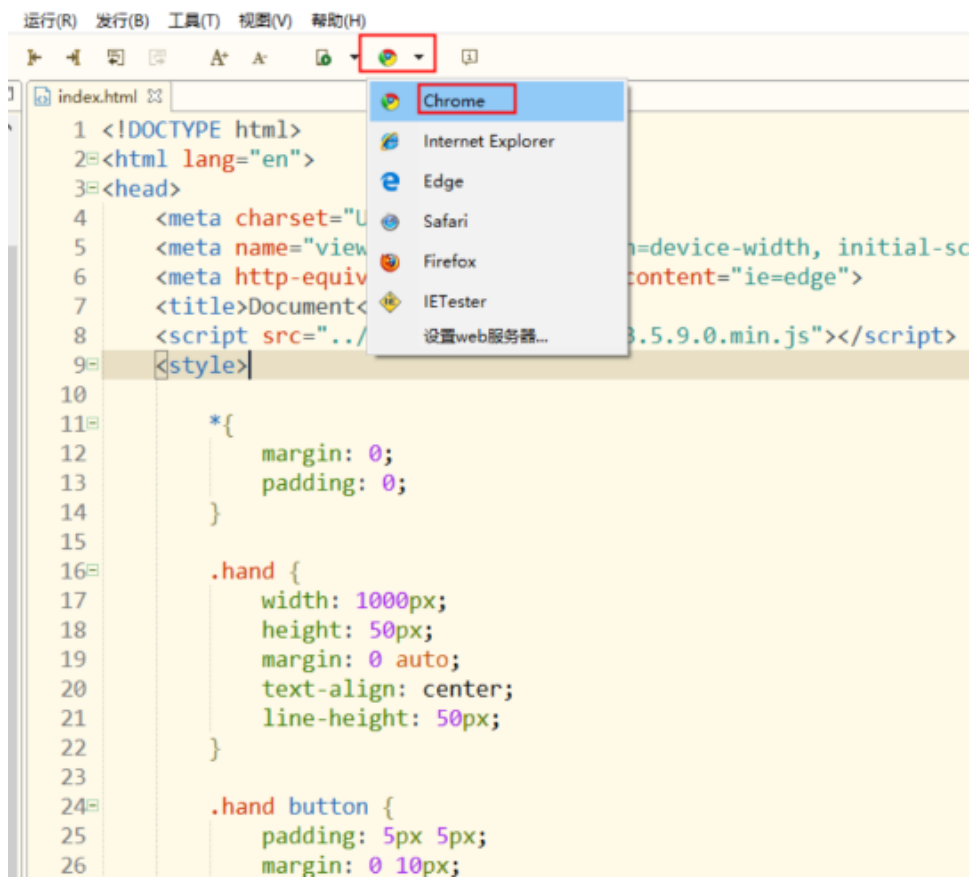




3.1.3 SDK Demo 工程运行

SDK Demo 导入工程后，打开页面（demo>baseVideo>index.html），通过其内置了 web 服务器，选择 Chrome 浏览器很方便运行，截图如下：





3.2 非工程环境运行

下载的 SDK Demo 不导入开发工具，也可以运行，如通过第三方 web 容器（tomcat,nginx 等）或通过 nodejs 运行，具体见 SDK Demo 的 readme.md 说明。

4 音视频基本功能开发流程

本开发流程只涉及到音视频的基本功能，以 baseVideo Demo 的能力加以说明，关于其它的功能，如桌面共享、多摄像头的支持，可以参阅 SDK API 接口开发手册或咨询我们的技术支持工程师。

4.1 获取授权信息

针对正式客户及测试客户，公司会提供一对有效的 Access Key 和 Secret Key，这对密钥可以产生访问令牌（24 小时有效），用于生成会议房间号，及加会房间等操作。

目前 SDK Demo 提供的测试的授权信息及服务器地址如下，公司会不定期的更新，有问题请咨询。

```
//服务器地址，依据当前 web 服务协议(http,https)指定对应的端口，默认是 https
var serverURI = "dev.3tee.cn:441";
```

```
var accessKey = "SDKdemo_access";
var secretKey = "SDKdemo_secret";
```

通过调用 rest api, 可以产生访问令牌（accessToken），参考代码如下：

```
var restApi = new RestApi (serverURI);
restApi.getAccessToken(accessKey, secretKey).then((res) => {
    console.log( '获取访问令牌成功.accessToken:', res.token);
}).catch((error) => {
    console.error( '获取访问令牌失败。error:', error);
});
```

4.2 引擎初始化

```
var AVDEngine = ModuleBase.use(ModulesEnum.avdEngine);
var avdEngine = new AVDEngine();
avdEngine.initDevice();
avdEngine.init(serverURI, accessToken).then(initSuccess).otherwise(initError);
function initSuccess (){
    console.log( '引擎初始化成功');
}
function initError(error){
    console.error( '引擎初始化失败。error:', error);
}
```



```
}
```

4.3 创建房间

房间如果没有，通过调用 `rest api`，可以创建房间，参考代码如下：

```
var restApi = new RestApi (serverURI);
restApi.createRoom(accessToken, topic, userId).then( (res) => {
    console.log( '获取房间成功.roomId:', res.room_id);
}).catch((error) => {
    console.error( '获取房间失败。error:', error);
});
```

注：参数说明如下：

`accessToken`: 访问令牌
`topic`: 会议主题
`userId`: 房间创建者 ID

4.4 加入房间

```
var roomId = "143213123231";
room = avdEngine.obtainRoom(roomId);
room.join(userId, userName, userData,
password).then(joinSuccess).otherwise(joinError);
function joinSuccess (){
    console.log( '加入房间成功');
}
function joinError(error){
    console.error( '加入房间失败。error:', error);
}
```

注：各种参数说明如下：

`userId`: 用户 ID,会议中需要保证唯一，否则后加会者会踢除前加会者。
`userName`: 用户名，可随便填写
`userData`: 用户扩展内容，可不填

4.5 开启/关闭音视频

打开视频：

```
var videos = room.selfUser.videos;
if(videos && videos.length > 0){
    var video = videos[0];
    video.previewAndPublish(localVideo).otherwise(alertError);
```

```
}
```

关闭视频:

```
var video = room.selfUser.videos[0];
if(video){
    video.unpreview();
    video.unpublish();
}
```

打开音频:

```
var audio = room.selfUser.audio;
if(audio){
    audio.openMicrophone().otherwise(alertError);
}
```

关闭音频:

```
var audio = room.selfUser.audio;
if(audio){
    audio.closeMicrophone();
}
```

4.6 房间回调事件

在加会成功 `joinSuccess` 中注册房间回调事件,完成远端用户加会,退会等回调操作。

```
function joinSuccess (){
    registerRoomCallback();
}

function registerRoomCallback (){
    room.addCallback(RoomCallback.user_join_notify, onUserJoinNotify);
    room.addCallback(RoomCallback.user_leave_notify, onUserLeaveNotify);
    room.addCallback(RoomCallback.leave_indication, onLeaveIndication);
}

/**
 * @desc 参会者加会回调
 * @param {Object} users — 参会者数组
 */
function onUserJoinNotify(users) {
    participantsHandle(users);
}
```

```

/**
 * @desc 参会者退会回调
 * @param {int} opt - 退会类型
 * @param {int} reason - 退会原因
 *
 *      807: 服务器端报 807 错误, 说明 UDP 不通或 UDP 连接超时
 * @param {Object} user - 退会用户
 */
function onUserLeaveNotify(opt, reason, user) {
}

/**
 * @desc 被踢出会议室
 * @param {Object} reason - 被踢原因
 *
 *      804: 同一个 userid 加会, 把前一个人踢下线
 *
 *      808: 调用 rest api 接口把人踢下线
 *
 *      809: 无授权 15 分钟以后强制踢人
 * @param {Object} userId - 踢人的操作者
 */
function onLeaveIndication(reason, userId) {
}

```

4.7 用户回调事件

在加会成功 `joinSuccess` 中注册用户回调事件, 完成用户的音视频状态, 订阅等回调操作。

```

function joinSuccess (){
    onPublishCameraNotify(room.pubVideos); //加会登陆前, 会议中已经发布的视频
    资源,采取订阅处理
    participantsHandle(room.getParticipants());
}

```

```

function participantsHandle(participants) {
    participants.forEach(function(user) {
        user.addCallback(UserCallback.microphone_status_notify,onMicrophoneStatusNotify);
        user.addCallback(UserCallback.camera_status_notify, onCameraStatusNotify);

```

```

        user.addCallback(UserCallback.publish_camera_notify, onPublishCameraNotify);

```

```

    user.addCallback(UserCallback.unpublish_camera_notify, onUnpublishCameraNotify);
    user.addCallback(UserCallback.subscribe_camera_result, onSubscribeCameraResult);
user.addCallback(UserCallback.unsubscribe_camera_result,
onUnsubscribeCameraResult);

```

```

user.addCallback(UserCallback.subscribe_microphone_result,
onSubscribeMicrophoneResult);
user.addCallback(UserCallback.unsubscribe_microphone_result,
onUnsubscribeMicrophoneResult);
}
}

```

```

/**
 * 麦克风状态更新
 * @param {Object} status — 状态
 * @param {Object} microphoneId — 麦克风设备 Id
 * @param {Object} microphoneName — 麦克风设备名称
 * @param {Object} userId — 麦克风设备所有者 I D
 */
function onMicrophoneStatusNotify(status, microphoneId, microphoneName, userId) {
}

```

```

/**
 * 摄像头状态更新
 * @param {Object} status — 状态
 * @param {Object} cameraId — 摄像头设备 Id
 * @param {Object} cameraName — 摄像头设备名称
 * @param {Object} userId — 摄像头设备所有者 I D
 */
function onCameraStatusNotify(status, cameraId, cameraName, userId) {
}

```

```

/**
 * 房间中发布的视频回调
 * @param {Object} videos — 发布的视频集
 */

```

```

function onPublishCameraNotify(videos) {

}

/**
 * 房间中取消发布的视频回调
 * @param {Object} video — 取消发布的视频
 */
function onUnpublishCameraNotify(video) {
}

/**
 * 订阅远端视频流反馈
 * @param {Object} stream — 远端视频流
 * @param {Object} userId — 所属用户 I D
 * @param {Object} userName — 所属用户名称
 * @param {Object} cameraId — 摄像头设备 I D
 */
function onSubscribeCameraResult(stream, userId, userName, cameraId) {
}

/**
 * 取消订阅远端视频流反馈
 * @param {Object} userId — 所属用户 I D
 * @param {Object} userName — 所属用户名称
 * @param {Object} cameraId — 摄像头设备 I D
 */
function onUnsubscribeCameraResult(userId, userName, cameraId) {
}

/**
 * 订阅远端音频流反馈
 * @param {Object} stream — 远端音频流
 * @param {Object} userId — 所属用户 I D
 * @param {Object} userName — 所属用户名称
 */
function onSubscribeMicrophoneResult(stream, userId, userName) {
}

```

```

/**
 * 取消订阅远端音频流反馈
 * @param {Object} userId — 所属用户 I D
 * @param {Object} userName — 所属用户名称
 */
function onUnsubscribeMicrophoneResult(userId, userName) {
}

```

4.8 订阅/取消订阅视频

订阅视频：

在 5.6 的房间中已发布的视频集回调 onPublishCameraNotify 中，进行订阅处理：

```

videos.forEach(function(video) {
    video.subscribe();
});

```

取消订阅：

在 5.6 的房间中取消发布的视频回调 onUnpublishCameraNotify 中，进行取消订阅处理：

```

video.unsubscribe();

```

4.9 音视频渲染

视频渲染：

在 5.6 的订阅远端视频流反馈回调 onSubscribeCameraResult 中，进行视频渲染处理：

```

room.selfUser.attachVideoElementMediaStream( videoElement, stream);

```

视频取消渲染：

在 5.6 的取消订阅远端视频流反馈回调 onUnsubscribeCameraResult 中，进行视频渲染取消处理：

```

room.selfUser.attachVideoElementMediaStream(videoElement, null);

```

其中 videoElement 为视频控件 video 对象。

4.10 离开/关闭房间

离开房间：

```

var reason = 1; //退会原因

```

```
room.leave(reason).then(function() {  
    console.log( '离开房间成功');  
})
```

关闭房间：

```
room.closeRoom(room.selfUser.id);
```

5 附录

5.1 名词解释

名词	解释	备注
room	房间对象，是实时沟通功能的一个管理单元，房间中会有多个沟通参与者即用户，房间有各种沟通功能，如文字聊天、语音视频等，沟通是基于房间的。不同房间沟通是隔离的	
user	用户对象，每个加入到房间的客户端作为一个房间用户，用户将会根据权限和设备情况执行房间中各种沟通功能。 用户 Id：唯一标示一个房间用户的 Id，由应用层来设置。	

5.1.2 iframe

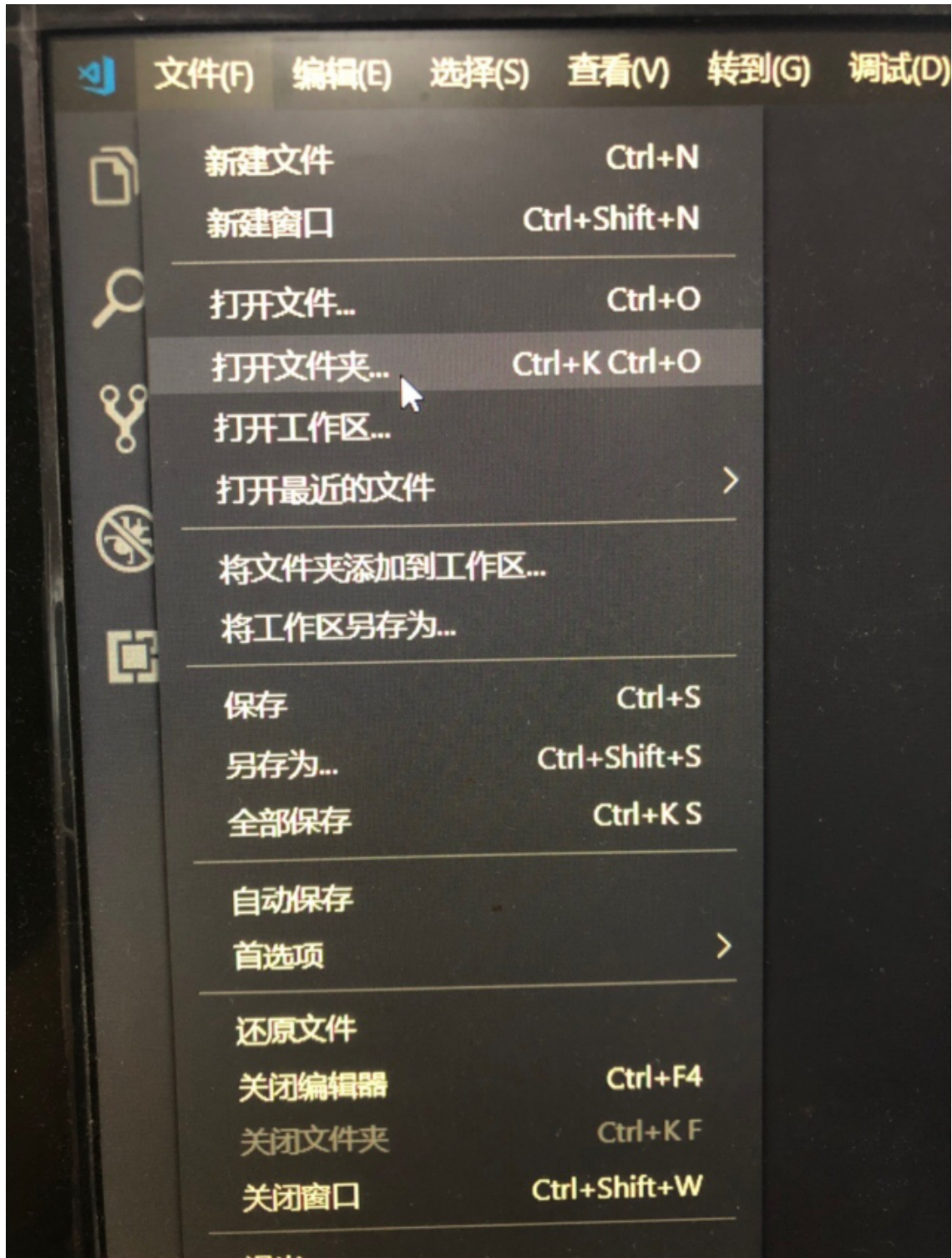
使用iframe访问dome页面时打不开音视频时，iframe要使用https，还要设置权限：

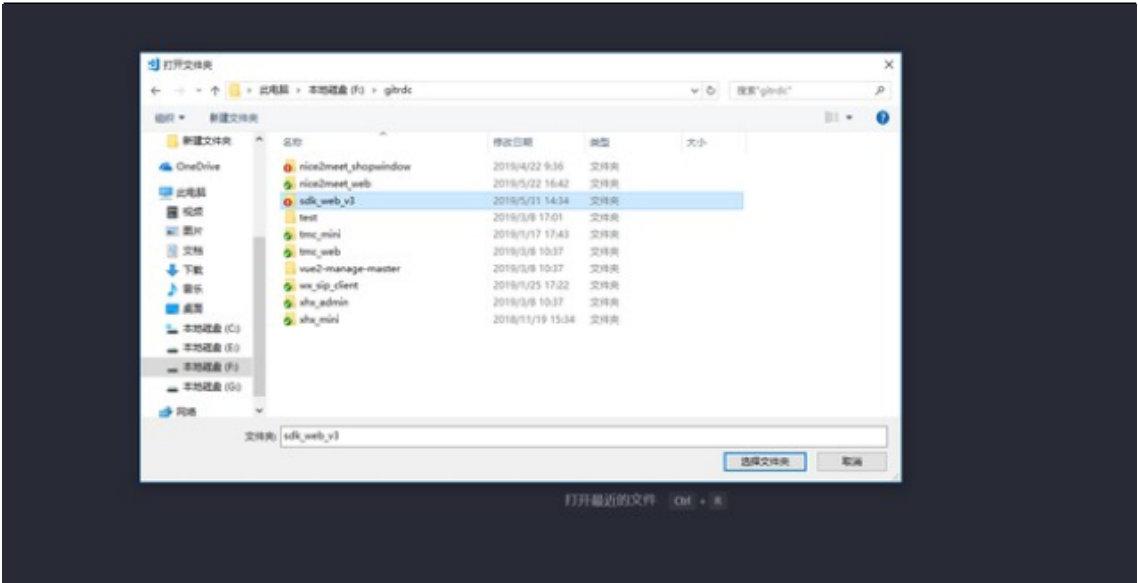
```
allow = "geolocation;microphone;camera;"
```

5.2 其它开发工具

5.2.1 vscode

5.2.1.1 SDK Demo 工程导入





5.2.1.2 SDK Demo 工程运行

