# 微信观看小程序SDK 3.10

## DEMO地址

demo下载地址

## 产品介绍

### 概述

polyv小程序SDK为微信小程序提供了直播播放、点播播放、文档绘制等功能。且提供了一套组件，供用户灵活组合自己的业务逻辑。

### 配置微信直播权限

SDK 播放直播使用微信 `live-player`，连麦使用 `live-pusher`。接入前需通过微信当前开放类目的审核，再进入小程序管理后台的“开发 - 接口设置”开通对应组件权限。具体要求详见微信观看小程序 SDK 类目申请资质要求，并以微信 live-player 组件开通要求及实际审核结果为准。

### 功能特性

| 功能 | 表述 |
| --- | --- |
| 视频 | 支持直播视频和点播视频观看。暂不支持加密的视频 |
| 文档 | 文档以及画笔展示，暂不支持ppt动画 |
| 教学连麦 | 支持语音和视频连麦功能。支持1v1连麦 |
| 在线聊天 | 支持在线聊天 |

### 阅读对象

本文档为技术文档，需要阅读者：

- 拥有基本的小程序开发能力
- 准备接入polyv视频云或已接入的客户

- 对polyv视频云使用方法有基础的了解

# 使用步骤

## 开发准备

### 获取Access Key

登录保利威直播后台 - 云直播 - 开发设置 - 身份认证

### 微信接口白名单配置

```
request合法域名
https://miniapp.agoraio.cn
https://uni-webcollector.agora.io
https://router.polyv.net
https://api.polyv.net
https://prtas.videocc.net
https://rtas.videocc.net
https://hls.videocc.net
https://player.polyv.net
https://livestatic.videocc.net
https://livejson.polyv.net
https://live.polyv.net
https://doc-2.polyv.net
https://doc.polyv.net
https://img.videocc.net
https://liveimages.videocc.net

使用连麦功能需加上以下域名：
https://uap-ap-web-1.agora.io
https://uap-ap-web-2.agoraio.cn
https://uap-ap-web-3.agora.io
https://uap-ap-web-4.agoraio.cn
https://report-ad.agoralab.co
https://rest-argus-ad.agoralab.co
https://uni-webcollector.agora.io

https://cloud.tencent.com
https://yun.tim.qq.com
https://webim.tim.qq.com

socket合法域名
wss://chat.polyv.net
wss://miniapp.agoraio.cn
```

## SDK使用方法

sdk用了TypeScript代码，所以需要下载最新的开发者工具才能支持。原生支持 TypeScript

sdk提供了自定义组件polyv提供一套完整的业务逻辑，供用户开箱即用。也提供了player播放组件、ppt文档组件、chatroom聊天室组件等供用户灵活组合自己的业务逻辑。

在使用之前需要在`app.js的onLaunch中`调用`setApp`方法

### 方法一：传入polyv云直播的access key

如果担心小程序代码中apiSecret是明文显示，可以自行通过接口加密传输，也可使用方法二

```
import plv from '*/sdk/core/index';
onLaunch() {
    plv.setApp({
        apiId: '',
        apiSecret: ''
    });
}
```

### 方法二：传入verifyUrl验证接口

```
import plv from '*/sdk/core/index';
onLaunch() {
    plv.setApp({
        apiId: '',
        verifyUrl: ''
    });
}
```

### verifyUrl验证接口规则

（1）小程序请求verifyUrl接口，会带上下列参数（带上参数数量/参数名不固定）。需将所有除参数sign外的参数按字母顺序排列后配合appSecret按规则进行MD5加密并返回给小程序

### 请求参数描述

| 参数名 | 类型 | 说明 |
| --- | --- | --- |
| appId | string | 账号appId【详见获取密钥】 |
| timestamp | number | 13位毫秒级时间戳 |

| 参数名 | 类型 | 说明 |
| --- | --- | --- |
| sign | string | verifyUrl 校验 sign 参考（2）verifyUrl校验 |

**响应参数描述**

| 参数名 | 类型 | 说明 |
| --- | --- | --- |
| code | Integer | 响应状态码，200为成功返回，非200为失败【详见全局错误说明】 |
| status | String | 响应状态文本信息 |
| message | String | 响应描述信息，当code为400或者500的时候，辅助描述错误原因 |
| data | Object | 响应成功时返回账号可用直播分钟数信息【详见data字段说明】 |

verifyUrl接口PHP代码示例：

```php
<?php
$appSecretKey = 'Your appSecretKey';
$sign = $_GET['sign'];
$appId = $_GET['appId'];
$timestamp = $_GET['timestamp'];

// 获取url query并转换成数组
parse_str($_SERVER["QUERY_STRING"], $params);

// 获取除sign外的其他参数的拼接字符串
$concated = sort_param($params);

// STEP 1
// 计算接口请求是否合法
$outPutData = '';

$verifyUrlSign = strtoupper(md5("plyMinApp".$concated."plyMinApp"));
if ($sign != $verifyUrlSign) {
  $outPutData = '{"code": 200, "message" : "invalid sign", "status": "error", "data": ""}';
  echo($outPutData);
  return;
}

// STEP 2
// 输出正确sign返回给小程序
$outPutSign = strtoupper(md5($appSecretKey.$concated.$appSecretKey));
$outPutData = '{"code": 200, "message" : "", "status": "success", "data": {"sign": "'.$outPutSign.'"}}';
echo($outPutData);


/**
 * 将参数按照ASCKII升序 key + value + key + value ... +value 拼接
 * @return [type] [description]
 */
function sort_param($params){

  ksort($params);

  $sort_result = "";

  foreach ($params as $key => $val) {
    if(!is_null($val) && $key != 'sign'){
      $sort_result=$sort_result.$key.$val;
    }
  }
  return $sort_result;
}

?>
```

(2)verifyUrl校验

校验sign规则：

1.concated 取值为将参数 appId、timestamp 及其他参数按照ASCKII升序 key + value + key + value ... +value 拼接

2.verifyUrlSign 取值 `plyMinApp${concated}plyMinApp` 字符串拼接后的大写MD5值

```
$verifyUrlSign = strtoupper(md5("plyMinApp".$concated."plyMinApp"));
```

成功示例

```
{
    "code":200,
    "status":"success",
    "message":"",
    "data":{
        "sign":"3DDE7222C4264F225931053A661889BA"
    }
}
```

异常示例

```
{
    "code": 400,
    "status": "error",
    "message": "invalid signature.",
    "data": ""
}
```

# 组件的使用

## 一、使用polyv组件。可参考demo的 `polyv` 目录

1. 拷贝sdk代码到自己的项目中，在使用到sdk的page的json文件中引入组件

```
{
  "usingComponents": {
     "polyv": "*/sdk/components/polyv/polyv"
  }
}
```

2. 在wxml中使用polyv组件

```
<view>
    <polyv />
</view>
```

3. 在页面的onload中调用init方法，在onUnload中调用destory方法

   init方法初始化观看，获取频道详情、初始化socket事件等。

```
import plv from '*/sdk/core/index';
// onLoad
onLoad() {
    const options = {
      channelId: '', // 频道ID
      openId: '', // 用户openId
      userName: '', // 用户名
      avatarUrl: '', // 用户头像
      param4: '', // 自定义参数
      param5: '', // 自定义参数
    };
    plv.init(options);
}
// onUnload
onUnload() {
   plv.destory();
 }
```

## 二、灵活组合组件。可参考demo的`polyv-sub`

1. 在使用到sdk的page的json文件中引入组件

```
{
  "usingComponents": {
     "player": "*/sdk/components/player/player",
        "ppt": "*/sdk/components/ppt/ppt",
     ...
  }
}
```

2. 在wxml中使用组件，传入必要的参数。

```
<view>
    <player
   videoOption="{{ videoOption }}"
   bind:onLiveStatusChange="playerLiveStatusChange"
 />
    <ppt />
</view>
```

3. 在页面的onload方法中调用init方法。

```
import plv from '*/sdk/core/index';
Page({
    onLoad() {
        const options = { ... };
        options.plvInsideUse = true; // 区分是学一学还是sdk，当为 true 时开启抽奖模块。
        plv.init(options)
            .then(data => {
                const { detail, chat } = data;
               // 处理业务逻辑
            })
            .catch(err => {
                        // 异常处理
            });
    },
    onUnload() {
       plv.destory();
    }
});
```

## 组件详解

使用以下组件之前，都需要先在json中通过`usingComponents`字段引入。

### 1. polyv组件

参数介绍

| 参数 | 类型 | 必填 | 默认 | 说明 |
| --- | --- | --- | --- | --- |
| userBanned | Event | 否 | - | 用户被踢出触发 |
| onError | Event | 否 | - | 出现异常 |
| allowDanmu | Boolean | 否 | true | 是否允许使用弹幕 |
| skinAlwaysShow | Boolean | 否 | false | 是否一直显示播放器皮肤 |

| 参数 | 类型 | 必填 | 默认 | 说明 |
| --- | --- | --- | --- | --- |
| usePlayerSkin | Boolean | 否 | true | 是否使用播放器皮肤 |

```
<polyv
  bind:userBanned="handleUserBanned"
  bind:onError="handlePolyvError"
  allowDanmu="{{ false }}"
  skinAlwaysShow="{{ true }}"
  usePlayerSkin="{{ false }}"
  hasAnswerCard="{{ true }}"
/>
```

### 2. player播放器组件

player组件可以播放直播和点播。直播使用live-player，`模拟器不能播放，查看效果请用真机。`点播模拟器不能播放`加密视频，查看效果请用真机。`

参数介绍

| 参数 | 类型 | 必填 | 默认 | 说明 |
| --- | --- | --- | --- | --- |
| videoOption | Object | 是 | 空 | 播放器初始化参数(详情见下面js代码) |
| vodSeek | Number | 否 | 0 | 回放视频seek操作时间点(mode为vod时，或者暂存才生效) |
| onLiveStatusChange | Event | 否 | - | 直播流状态（mode为live时触发） |
| onLiveStorageProgress | Event | 否 | - | 直播暂存当前播放时间 |
| onVodProgress | Event | 否 | - | 点播回放当前播放时间 |
| onVodEnd | Event | 否 | - | 点播回放播放结束 |
| onError | Event | 否 | - | 出现异常 |
| allowDanmu | Boolean | 否 | true | 是否允许使用弹幕 |

| 参数 | 类型 | 必填 | 默认 | 说明 |
| --- | --- | --- | --- | --- |
| skinAlwaysShow | Boolean | 否 | false | 是否一直显示播放器皮肤 |
| usePlayerSkin | Boolean | 否 | true | 是否使用播放器皮肤 |
| hasAnswerCard | Boolean | 否 | false | 是否使用答题卡 |

**注意：skinAlwaysShow的优先级大于usePlayerSkin，如果设置了skinAlwaysShow为true，usePlayerSkin参数失效**

```
<player
 videoOption="{{ videoOption }}"
 allowDanmu="{{ false }}"
 skinAlwaysShow="{{ true }}"
 usePlayerSkin="{{ false }}"
 bind:onLiveStatusChange="playerLiveStatusChange"
 bind:playerVodProgress="playerVodProgress"
 bind:onVodEnd="playerVodEnd"
 bind:onLiveVodEnd="playerVodEnd"
 bind:onError="playerError"
/>
```

```
// ###### 播放直播或者暂存视频 #######
videoOption = {
   mode: 'live',
   uid: userId, // 直播频道uid
   cid: channelId, // 直播频道channelId
   isAutoChange: true, // 自动切换直播和暂存。
   vodsrc: '', // 指定回放地址。有暂存视频的情况下，传入暂存视频的mp4或者m3u8。
   pipMode: '', // 是否使用小窗模式，默认为undefined。相关参数设置详情参考注意2.2
   forceVideo: false, // 是否强制使用video标签作为播放器(播放m3u8)，建议使用live-player
   statistics: { // 播放器自定义统计参数，如需添加param4、param5参数，详情见下面init方法详解
     param1: 'param1', // 用户ID
     param2: 'param2', // 用户昵称
   },
   // logoConfig: {
   //   enable: false, // 是否显示logo
   //   position: 'tl',// logo位置1 左上，2 右上（默认）3左下 4 右下
   //   opacity: 0.5, // 透明度
   //   src: '' // logo图片的url
   // }
};

// 直播状态改变：只有在mode为live时才会触发。
playerLiveStatusChange(e) {
    const status = e.detail.status;
    if (status === 'live') {
      // 开始直播
    }
    if (status === 'end') {
      // 结束直播
    }
}

// 获取回放播放进度
playerVodProgress(e) {
  console.info(e.detail.currentTime, '----currentTime---');
}

// 播放器异常捕获
playerError(e) {
  console.info(e.detail, '-----e-----');
}

/*
* 回放播放结束事件
* onVodEnd：点播回放列表播放结束触发，返回当前播放结束点播视频vid
* onLiveVodEnd：直播暂存播放结束时触发，返回当前暂存视频播放地址
*/
playerVodEnd(e) {
  console.info(e.detail.curVodVid, '---curVodVid---');
}

// ###### 播放点播视频 #######
videoOption = {
    mode: 'vod',
    vodVid: '' // 播回放时vodVid为videoPoolId
```

```
};
// 在mode为vod时，从点播切换到直播状态，云课堂和普通直播监听直播开始的方法不同。
// 1. 普通直播通过轮询api.getOrdinaryLiveStatus(stream)获取当前的状态
// 2. 云课堂通过chat.on(chat.events.SLICESTART, () => {})监听直播开始
```

**注意：**

### 2.1 skinAlwaysShow 和 usePlayerSkin 优先级

skinAlwaysShow的优先级大于usePlayerSkin，如果设置了skinAlwaysShow为true，usePlayerSkin参数失效

### 2.2 pipMode

参数范围与触发条件参考官方API live-player组件参数。官方说明基础库需升级到2.11.0才支持小窗功能，建议登录公众号平台 -> 设置 -> 基础库最低版本设置 进行升级。

### 3. ppt文档组件

参数介绍

| 参数 | 类型 | 必填 | 默认 | 说明 |
| --- | --- | --- | --- | --- |
| chatData | Object | 是 | - | 频道详情 |
| videoId | String | 否 | - | 当前播放回放的videoId |
| vidCurrentTime | Number | 否 | - | 当前回放播放时间 |
| pptSize | Object | 是 | - | 文档尺寸{ height , width } |

```
<ppt
   chatData="{{ detail }}"
   videoId="{{videoId}}"
   vidCurrentTime="{{vodPlayerProgress}}"
   pptSize='{{pptSize}}'
/>
```

```
// 直播时传入chatData
// 播放点播时传入回放的videoId和当前回放播放时间
```

### 4. concat连麦组件

参数介绍

| 参数 | 类型 | 必填 | 默认 | 说明 |
| --- | --- | --- | --- | --- |
| channelDetail | Object | 是 | - | 频道详情 |
| applyData | Object | 是 | {show：fale, txt: '申请连线'} | |
| show | Event | 是 | - | 房间连麦状态：开启/关闭 |
| refreshStatus | Event | 是 | - | 连麦状态更改：举手apply/等待允许cancel/挂断stop |
| stop | Event | 是 | - | 停止连麦 |

```
<concat
    id="test"
 channelDetail="{{ channelDetail }}"
    applyData="{{ applyData }}"
    bind:show="handleShowConcatApply"
    bind:refreshStatus="handleRefreshStatus"
    bind:stop="handleStop"
/>
```

```
//js
// 监听当前房间连麦状态
// 弃用
handleShowConcatApply(data) {
  // data.detail.status为open/close
  // open: 当前房间已开启连麦
  // close: 当前房间未开启连麦
},

// 监听当前用户连麦状态
// 只有在房间开启了连麦功能后，用户才能进行连麦
handleRefreshStatus(data) {
  // data.detail: {
  //  show: true/false, // 是否能连麦
  //  type: 'apply'/'cancel'/'stop', // 当前连麦类型：未举手/已举手/连麦中
  //  txt: '' //对应连麦类型：申请连线/取消申请/挂断连线
  //}
},

// 结束连麦
handleStop(data) {
  console.info(data.detail, '---stop----');
}
```

### 连麦组件相关方法说明

```
- apply

  说明：  连麦控制方法，“举手”/“取消申请”都需要调用此方法，调用此方法后组件会判断当前连麦状态，触发响应的事件。
- stop

  说明：挂断连线
```

## 5. chatroom聊天室组件

```
<chatroom bind:onTapBulletin="handleShowBulletin" />
```

```
handleShowBulletin() {
  console.info('===点击聊天室公告按钮触发====');
}
```

| 参数 | 类型 | 必填 | 默认 | 说明 |
| --- | --- | --- | --- | --- |
| showBulletin | Boolean | 否 | true | 是否显示公告按钮 |

| 参数 | 类型 | 必填 | 默认 | 说明 |
| --- | --- | --- | --- | --- |
| skin | String | 否 | black | 皮肤（black、white） |

### 6. quiz咨询提问组件

```
<quiz />
```

### 7. playback往期组件

参数介绍

| 参数 | 类型 | 必填 | 默认 | 说明 |
| --- | --- | --- | --- | --- |
| playbackList | Array | 否 | [ ] | 回放列表 |
| nextVod | String | 否 | '' | 当前播放的回放videoPoolId |
| onTapPlayback | Event | 否 | 空 | 点击回放回调函数 |

```
<playback
    playbackList="{{ playbackList }}"
    nextVod="{{ currentVodId }}"
 bind:onTapPlayback="handlePlayback"
/>
```

```
// 回放列表通过api.getPlayBackVideos(channelId)获取
// 播放下一个回放，nextVod传入当前的回放videoPoolId
// 点击某个回放时，通知player播放
handlePlayback(e) {
    const { videoPoolId, videoId } = e.detail;
}
```

### 8. chapter章节组件

参数介绍

| 参数 | 类型 | 必填 | 默认 | 说明 |
| --- | --- | --- | --- | --- |
| chapterList | Array | 是 | [ ] | 章节列表 |

| 参数 | 类型 | 必填 | 默认 | 说明 |
| --- | --- | --- | --- | --- |
| vodCurTime | Number | 是 | 0 | 当前播放时间 |
| onTapChapter | Event | 否 | 空 | 点击章节回调函数 |

```
<chapter
    bind:onTapChapter="handleChangeChapter"
 vodCurTime="{{ vodPlayerProgress }}"
 chapterList="{{ chapterList }}"
/>
```

```
// 回放列表通过api.getChapterRecords(channelId)获取
// vodCurTime: 当前播放器的播放时间
// onTapChapter
handleChangeChapter(e) {
 const chapter = e.detail.chapter;
}
```

### 9. menu-custom自定义菜单组件

参数介绍

| 参数 | 类型 | 必填 | 默认 | 说明 |
| --- | --- | --- | --- | --- |
| parseHtml | String | 是 | 空 | 富文本字符串 |

### 10. sign 互动功能签到组件

```
<sign bind:onSignShow="handleSignShow" />
```

```
handleSignShow() {
  console.info('====收到签到开始事件，显示签到弹窗时触发====');
}
```

### 11. question 互动功能问卷组件

```
<question zIndex="2000" />
```

| 参数 | 类型 | 必填 | 默认 | 说明 |
| --- | --- | --- | --- | --- |
| zIndex | Number | 否 | - | 设置弹窗层级 |

### 12. answer-card 互动功能答题卡组件

```
<answer-card
  class="c-answer-card"
  answerCardSize="{{ answerCardSize }}"
  bind:onAnswerCardShow="handleShowAnswerCard" />
```

```
handleShowAnswerCard() {
  console.info('=====收到答题事件，显示答题卡弹窗时触发====');
}
```

| 参数 | 类型 | 必填 | 默认 | 说明 |
| --- | --- | --- | --- | --- |
| answerCardSize | Object | 否 | 空 | 设置答题卡弹窗大小 { height: 400, width: 750 } |
| zIndex | Number | 否 | - | 设置弹窗层级 |
| class | String | 否 | 空 | 设置组件样式 |

### 13. lottery 互动功能抽奖组件

```
<lottery zIndex="{{ lotteryIndex }}"/>
```

| 参数 | 类型 | 必填 | 默认 | 说明 |
| --- | --- | --- | --- | --- |
| zIndex | Number | 否 | - | 设置弹窗层级 |

### 14. bulletin 互动功能公告组件

```
<bulletin
  show="{{ true }}"
  zIndex="2001"
  bulletinStr="公告显示内容"
  bind:onClose="handleHideBulletin"/>
```

```
handleHideBulletin() {
  console.info('===点击关闭公告按钮触发===');
}
```

| 参数 | 类型 | 必填 | 默认 | 说明 |
| --- | --- | --- | --- | --- |
| show | Boolean | 是 | false | 用于控制何时显示公告 |
| bulletinStr | String | 是 | '' | 公告内容 |

公告可以自定义显示内容，如果需要显示聊天室的公告消息，需要监听聊天室"BULLETIN"事件，获取公告内容显示

## 使用api

功能使用时核心方法为setApp, init、destory、api方法

### setApp

设置polyv云直播的appId、appSecret。一般在app.js的onLaunch方法中调用。

### init

初始化成功返回 频道详情：`detail` 、聊天室：`chat` 、网络请求：`api`

```
plv.init(options)
  .then(r => {
    // 初始化成功
    const { detail, chat } = r;
  })
  .catch(err => {
    // 初始化失败
    console.error(err);
  });
```

### 自定义统计参数设置

如果需要传入自定义互动统计数据param4、param5,需要在options里面加上参数; 如果是直接引用polyv组件，则直接在options里面添加param4、param5即可，如果是单独引用player组件，则需要设置player组件参数videoOption;

直接引用polyv组件示例：

```
plv.init(options = {..., param4: 'param4', param5: 'param5'})
  .then(r => {
    // 初始化成功
    const { detail, chat } = r;
  })
  .catch(err => {
    // 初始化失败
    console.error(err);
  });
```

单独使用player组件示例

```
  <player
    videoOption="{{ videoOption }}"
  />
```

```
videoOption = {
  statistics: {
    param4: 'param4',
    param5: 'param5'
  }
}
plv.init(options = {..., param4: 'param4', param5: 'param5'})
  .then(r => {
    // 初始化成功
    const { detail, chat } = r;
  })
  .catch(err => {
    // 初始化失败
    console.error(err);
  });
```

相关参数说明

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| appId | String | 是 | polyv云直播appId |
| appSecret | String | 是 | polyv云直播appSecret |
| channelId | String\|Number | 是 | 频道号 |
| openId | String | 是 | 小程序用户openId |
| userName | String | 是 | 用户昵称 |

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| avatarUrl | String | 是 | 用户头像 |

频道详情: detail

| 属性 | 说明 |
| --- | --- |
| channelMenus | 后台设置的页面菜单 |
| scene | 当前直播类型: ppt 云课堂 / alone 普通直播 |
| status | 直播状态: Y 直播中 / N 非直播 |
| name | 直播名称 |
| desc | 直播介绍 |
| publisher | 主持人 |
| userId | 用户ID |
| likes | 点赞数 |
| pageView | 观看数 |
| channelId | 频道号 |
| playbackEnabled | 回放功能是否开启 |
| hasPlayback | 是否有回放 |
| playbackList | 回放列表 |
| recordFileSimpleModel | 直播暂存 |
| warmUpImg | 暖场图片 |
| warmUpFlv | 暖场视频 |
| coverImage | 封面图 |
| stream | 直播流名 |
| startTime | 直播开始时间 |

| 属性 | 说明 |
| --- | --- |
| sessionId | 场次Id |
| chatToken | 聊天室鉴权token |

聊天功能：chat

- chat.events 所有事件

| 参数 | 说明 |
| --- | --- |
| CONNECT | 连接socket |
| DISCONNECT | 取消连接 |
| ERROR | socket错误 |
| RECONNECT_ATTEMPT | 重新连接开始 |
| CLOSE_ROOM | 房间关闭 |
| OPEN_ROOM | 房间打开 |
| SYSTEM_MESSAGE | 系统消息 |
| SPEAK | 用户发言 |
| SPEAK_ERROR | 发言错误 |
| SPEAK_CENSOR | 发言审核 |
| FLOWERS | 送花 |
| CHAT_IMG | 图片 |
| REWARD | 奖励信息 |
| CUSTOMER_MESSAGE | 自定义信息 |
| SERVER_ERROR | 后台出错 |
| KICK_USER | 用户被踢 |
| REMOVE_HISTORY | 清除聊天记录 |

| 参数 | 说明 |
| --- | --- |
| REMOVE_CONTENT | 清除某条聊天记录 |
| HISTORY_MESSAGE | 获取历史聊天信息 |
| SEND_MESSAGE | 发送消息成功 |
| PROHIBIT_TO_SPEAK | 禁止发言 |
| LOGIN | 登录 |
| LOGOUT | 退出登录 |
| LOGIN_REFUSE | 禁止登录 |
| SLICESTART | 云课堂上课 |
| MICROPHONE | 连麦 |
| ALLOW_MICROPHONE | 允许连麦 |
| SUCCESS_MICROPHONE | 连麦成功 |
| JOIN_CHANNEL_FAIL | 加入频道失败 |
| BAN_USER_ROOM | 禁止进入聊天室 |
| UPDATE_QUESTION_HISTROY | UPDATE_QUESTION_HISTROY |
| S_QUESTION | 学生提问 |
| T_ANSWER | 教师或助教或管理员回答问题 |

- chat.socket 获取socket对象
- chat.on 监听events事件
- chat.off 卸载events事件
- chat.trigger 触发事件
- chat.options 传入的options
- chat.roomClosed 房间是否关闭
- chat.teacherData 老师信息

- 聊天室api相关

| api | params | desc |
| --- | --- | --- |
| historyCount | 无 | 获取聊天室历史消息数量 |
| getHistoryMessage() | start I int: 开始行数<br>end \| int: 结束行数<br>请求示例：<br>chat.getHistoryMessage(end, start, data => {console.log(data)}) | 获取聊天室记录<br>return \| Array |
| hasMoreHistory() | 无 | 查询是否有历史记录 |
| getOnlineUserList() | 请求示例:<br>chat.getOnlineUserList().then(res => {console.log(res)}) | 获取频道在线列表<br>return \| Object |
| sendFlower() | 无 | 送花 |
| sendLike() | num \| int: 点赞次数<br>请求示例:<br>chat.sendLike(1) | 发送点赞 |
| send() | msg \| String: 文字消息<br>请求示例:<br>chat.send('Hello,World') | 发送消息 |

- 咨询室api相关
  - chat.getQuestionHistoryMessage() 获取咨询历史记录
  - chat.sendQuestion() 发送咨询消息
- 连麦
  - chat.checkCurrentStatus() 查询当前连麦状态
  - chat.cancelJoinChannel() 取消加入连麦

### destory

在页面的onUnload方法中调用该方法，重置数据

### api

- api.getUserId(openId) 获取userId

- api.getChannelDetail(channelId) 获取频道详情
- api.getOrdinaryLiveStatus(channelId) 获取普通直播的状态
- api.getPlayBackVideos(channelId) 获取回放列表数据。
- api.getChapterRecords(params) 获取章节信息
  - params.id 暂存文件id/回放视频的videoId（当type为record，id为暂存文件的fileId，当type为playback，id为回放视频的videoId）；
  - params.channelId 频道ID
  - params.type 类型（record：暂存类型；playback：回放类型）
- api.getChannelKey(channelId) 获取连麦密钥

### 错误消息说明

| 错误码 | 说明 |
| --- | --- |
| 31000 | 点播视频数据获取失败 |
| 31001 | vid不能为空 |
| 31002 | 点播过期 |
| 31003 | 账户没流量 |

## change log

### 3.0.0

- 目录调整，逻辑代码和界面代码分离。

### 3.1.0

- 增加跑马灯，水印，logo，暂停广告，倍速播放功能。

### 3.2.0

- 点赞接口加上userid参数。
- 修复用户名带特殊字符截断显示不全的bug。

### 3.3.0

- 修复连麦挂件初始化失败的问题。

### 3.5.0

- 修复TRTC连麦弹窗被画笔数据遮盖问题。

### 3.6.0

- 小程序聊天组件补齐展示引入回复消息（不支持发送引入回复）。

### 3.7.0

- 修复在连麦状态下开播，下播后，导致页面卡死。

### 3.8.0

- 聊天室对齐新增表情包
- 补充统计页面观看次数逻辑

### 3.9.0

- 修复聊天消息跳过严禁词

### 3.10.0

- 新增自定义统计参数