# 类

## 概述

> **PolyvLiveSdk** 是用于创建SDK实例的类

应用场景：当自己实现聊天室的功能时，可以使用到这两个属性

## 类属性

| 属性名 | 描述 | 返回数据类型 | 备注 |
| --- | --- | --- | --- |
| emotionLists | polyv 表情数据列表 | array | 已废弃 |
| emotionSearch | 键值对为 [`表情title`]: `表情url` 的对象 | object | 已废弃 |

> emotionLists 与 emotionSearch 已废弃不再维护，请使用 保利威表情 SDK 转换消息的表情内容。

全部表情 sprites 图片地址： https://livestatic.polyv.net/assets/images/em/default.png

```
/**
 * 获取表情数据列表
 *
 * @return {Array} - 表情数据列表
 */
// 示例代码
console.log(PolyvLiveSdk.emotionLists);
// 返回数据
[
    {
        url: "https://livestatic.polyv.net/assets/images/em/1.png",
        title: "微笑",
        position: "0px 0px"
    }
]
```

```
/**
 * 获取表情对象
 *
 * @return {Object} - 对象
 */
// 示例代码
console.log(PolyvLiveSdk.emotionSearch);
// 返回数据
{
    NO: {
        url: "https://livestatic.polyv.net/assets/images/em/89.png",
        title: "NO",
        position: "-4224px 0px"
    }
}
```

## 类方法

| 方法名 | 描述 | 参数 |
| --- | --- | --- |
| getChannelInfo | 获取频道信息 | 见下方示列代码块 |
| previewPPT | 预览指定的ppt | 见下方示列代码块 |

```
/**
 * 获取频道信息
 *
 * @param {string} channelId - 直播后台频道号
 * @param {string} appId - polyv 后台的 appId：开发设置 => 开发者信息 => AppID（应用 ID ）
 * @param {string} sign - 频道验证签名
 * @param {string} timestamp - 当前 13 位毫秒级时间戳
 * @return {Promise} - 标准 Promise 对象
 */
// 示例代码
// 注：自动请求的接口：/live/v3/applet/sdk/get-channel-detail
PolyvLiveSdk.getChannelInfo(channelId, appId, sign, timestamp)
    .then(res => {
        console.log(res);
    })
```

ppt操作demo

```
/**
 * 预览指定的ppt
 *
 * @param {object} config - 配置对象
 * @description config - config 列表:
 *       el {DOMSeletor|string} - 需要显示预览ppt的dom选择器
 *       autoId {number|string} - pptId
 *       type {string} - ppt类型
 *       width {number|string} - 可选，ppt宽，默认100%
 *       height {number|string} - 可选，ppt高，默认100%
 *       pptNav {boolean} - 可选，是否显示ppt控制控件，默认`true`
 *       pptNavBottom {string} - 可选，ppt控制栏距离底部距离，例：`pptNavBottom: '50px'
 * @return {function} - previewPPT 对象
 */
// 示例代码
// 需要给 el 元素设置相对定位和宽高
/*
#previewPPT{
    position: relative;
    width: 500px;
    height: 300px;
}
*/

var previewPPT = PolyvLiveSdk.previewPPT({
    el: '#previewPPT',
    autoId: 1329757,
    type: 'new'
});

// 示例事件
previewPPT.on('success', function () {
    console.log('预览成功');
});

previewPPT.on('error', function (err) {
    console.log('预览失败', err);
});

previewPPT.on('pptStatusChange', function (err) {
    console.log('当前页改变', err);
});

// 实例方法
previewPPT.nextPPTPage(); // 使 ppt 切换到下一页，跟随模式下不能切到讲师未讲解的 ppt
previewPPT.prevPPTPage(); // 使 ppt 切换到上一页

// 销毁ppt预览实例
//previewPPT.destroy()
```

# 实例

## 概述

> var **liveSdk** = new PolyvLiveSdk({}) // 创建sdk实列

## 实例属性

| 属性名 | 必选 | 类型 | 可选值 | 默认值 | 说明 |
| --- | --- | --- | --- | --- | --- |
| channelId | 是 | string | | | 频道id， 支持云课堂/直播助手/大班课发起的直播 |
| sign | 是 | string | | | POLYV 频道信息获取接口用到的签名 签名规则<br>请用 channelId、timestamp、appId 三个字段拼接sign |
| timestamp | 是 | number | | | POLYV 频道信息获取接口用到的时间戳<br>**注：该时间戳需要与服务器端获取的签名一样** |
| appId | 是 | string | | | POLYV账号appId，从POLYV 后台获取 |

| 属性名 | 必选 | 类型 | 可选值 | 默认值 | 说明 |
| --- | --- | --- | --- | --- | --- |
| user | 是 | object | {user: string; userName: string; pic: string} | | {<br>userId: '', // 用户id<br>userName: '', // 用户昵称<br>pic: '' // 用户头像，必须为是 http/https/ // 开头链接<br>}<br>`userId` 是聊天室中的用户唯一标识，两个相同的 `userId` 登入聊天室后，后者会将前者踢出聊天室，因此不同的聊天室用户的 `userId` 需要具有唯一性，生成建议如下：<br>1. 如果您有用户系统，则使用用户系统中的用户 id 作为登入聊天室的 `userId`<br>2. 如果您没有用户系统，则在后端生成高度随机的值作为登入聊天室的 `userId`，如：「时间戳 +5 位随机数」、uuid 等，并将该 `userId` 保存到 cookie 或本地存储（localStorage 或 sessionStorage）中，避免每次刷新页面后重新生成 `userId` |

| 属性名 | 必选 | 类型 | 可选值 | 默认值 | 说明 |
| --- | --- | --- | --- | --- | --- |
| updatePageview | 否 | boolean | | false | 设置后每当调用实例polyv的观看页上的累计观看人数会增加一个 |
| chat | 否 | boolean | | true | 是否连接聊天室，在直播模式下如果调用直播播放器必须连接聊天室，所以即使设置为`false`，在初始化直播播放器时会自动连接聊天室，回放模式下可以不连接 |
| keepChatDisabled | 否 | boolean | | false | 直播模式是否保持不连接聊天室，在`chat: false`设置后回放的场景会不连接聊天室，直播时会连接聊天室，如果需要在直播时需要保持聊天室关闭可设置该参数。需要注意的是设置后需要依赖聊天室信令的功能会不可用，如PPT、无延迟、连麦、互动功能等，`chat: false`设置后该参数才会生效 |
| lang | 否 | string | 'zh_CN' \| 'en' | 'zh_CN' | 设置播放器语言<br>'zh_CN' : 中文<br>'en': 英文 |

| 属性名 | 必选 | 类型 | 可选值 | 默认值 | 说明 |
| --- | --- | --- | --- | --- | --- |
| param4 | 否 | string | | | 播放器统计参数param1、param2，param3被预设为userId和userName以及直播状态，所以设置无效 |
| param5 | 否 | string | | | 播放器统计参数 |
| code | 否 | string | | | 跑马灯的授权参数 |
| socket | 否 | object | | | 此参数默认会创建socket，同时支持外部传入socket或聊天室sdk的socket **注：若使用聊天室sdk或者自行创建的聊天室，必须把实例出来的socket传入到此字段，不然会导致重复登录聊天室出现异常 使用方式：见如下示列** |
| lotteryEnabled | 否 | string | 'Y' \| 'N' | | 设置为'N'时，进行普通抽奖时会过滤当前用户 |

直播sdk&聊天室sdk 示例代码（以polyv聊天室SDK为例）

```
var chatroom = new PolyvChatRoom({  //实例聊天室SDK
  roomId: channelId,
  userId: userId,
  nick: userName,
  pic: pic,
  userType: 'slice'
});

var liveSDK = new PolyvLiveSdk({   //实例直播SDK
  channelId: channelId,
  appId: appId,
  sign: sign,
  timestamp: timestamp,
  socket: chatroom.chat.socket,    //传入聊天室的socket
  user: {
    userId: userId,
    userName: userName,
    pic: pic
  }
});
```

# 实例方法

## 1.初始化播放器

说明：初始化播放器，必须在频道信息事件触发后调用参数：`config` 播放器配置对象

**使用：liveSdk.setupPlayer(config: object)**

示例代码：

```
var config = {
    pptEl: '#ppt',
    el: '#player',
    type: 'auto',
    autoplay: false,
    audioMode: false,
    width: '100%',
    height: '100%',
    pptWidth: '100%',
    pptHeight: '100%',
    controllerPosition: 'ppt',
    controller: true,
    pptNav: true,
    // 回放模式需要fileId、url、sessionId
    fileId: undefined,
    url: undefined,
    sessionId: undefined
}
// 监听频道信息并初始化播放器
liveSdk.on(PolyvLiveSdk.EVENTS.CHANNEL_DATA_INIT, (event, data) => {
  liveSdk.setupPlayer(config);
});
```

参数说明：

| 参数名 | 必选 | 类型 | 可选值 | 默认值 | 说明 |
| --- | --- | --- | --- | --- | --- |
| el | 是 | string | id或class | | 讲师区域元素 |
| pptEl | 否 | string | id或class | | ppt 文档元素选择器，非云课堂（三分屏）可不填，<br>**注：仅在三分屏生效** |
| width | 否 | string \| number | | '100%' | 讲师区域宽度 |
| height | 否 | string \| number | | '100%' | 讲师区域高度 |
| pptWidth | 否 | string \| number | | '100%' | ppt区域宽度 |
| pptHeight | 否 | string \| number | | '100%' | ppt区域高度 |

| 参数名 | 必选 | 类型 | 可选值 | 默认值 | 说明 |
| --- | --- | --- | --- | --- | --- |
| type | 否 | string | 'auto' \| 'live' \| 'vod' \| 'record' | 'auto' | 播放器播放类型，选择`'auto'`后，播放器会以直播->回放列表视频->第一个暂存视频（优先级由大到小）的优先级播放<br>`'auto'`：根据频道实际设置自动选择播放类型<br>`'live'`：直播<br>`'vod'`：回放<br>`'record'`：暂存 |
| pptType | 否 | string | 'vod' \| 'record' | 'vod' | 设置 ppt 类型。用于`type: 'vod'`模式下指定ppt类型，可选`'record'`、`'vod'`,在使用指定暂存场景回放场次时可设置为`'record'`，回放列表可不设置，若调用`player.switchVod`切换回正常回放列表的场次需传入`'vod'`修改为回放列表类型<br>  |

| 参数名 | 必选 | 类型 | 可选值 | 默认值 | 说明 |
| --- | --- | --- | --- | --- | --- |
| vodType | 否 | string | 'playback' \| 'vod' | | 设置播放回放类型。<br>用于 `type: 'vod'` 模式下回放播放列表类型（`playback`-回放列表，`vod`-点播列表），设置为 `vod` 后将使用点播播放器播放，使用点播vid播放，流量将统计到点播平台，所以vid为必填值，若设置 `type: 'auto'`，如果当前播放的是回放则按点播列表播放 |
| vodToken | 否 | object | ((vid: string, next: (data: VodToken) => void) => void) \| VodToken | | 点播加密视频校验参数，若确认播放的点播视频存在加密视频则需要传入校验参数，可参考 播放加密视频 示列：<br>  |
| responseSettings | 否 | boolean | | fasle | 在设置自动播放模式 `type: 'auto'` 时播放回放是否按照Polyv后台设置播放视频 |

| 参数名 | 必选 | 类型 | 可选值 | 默认值 | 说明 |
| --- | --- | --- | --- | --- | --- |
| autoplay | 否 | boolean | | false | 是否自动播放，移动端因为浏览器限制，可能不生效 |
| audioMode | 否 | boolean | | false | 讲师画面以音频模式播放<br>**注：仅在移动端生效** |
| controller | 否 | boolean | | true | 是否显示控制栏<br>**注：因为设备兼容问题，在移动端中建议显示控制栏** |
| controllerPosition | 否 | string | 'ppt' \| 'player' | 'ppt' | 控制栏显示在哪个区域<br>`'ppt'`：ppt区域<br>`'player'`：讲师区域<br>**注：因为考虑到兼容的问题，该设置只在 PC 端的三分屏生效** |
| fixedController | 否 | boolean | | false | 固定显示控制栏，设置后控制栏不跟随鼠标离开隐藏<br>**注：仅在 PC 端生效** |
| pptNav | 否 | boolean | | true | 是否显示ppt控制控件<br>**注：仅在三分屏生效** |
| pptNavBottom | 否 | string | | | ppt控制栏距离底部距离，例：`pptNavBottom: '50px'`<br>**注：仅在三分屏生效** |

| 参数名 | 必选 | 类型 | 可选值 | 默认值 | 说明 |
| --- | --- | --- | --- | --- | --- |
| pptPlaceholder | 否 | boolean | | false | 是否在直播模式并且当前暂无直播时在ppt区域显示占位图<br>**注：仅在三分屏生效** |
| fileId | 否 | string | | | ppt数据id，**回放模式必填** |
| url | 否 | string | | | 回放视频链接，**回放模式必填** |
| sessionId | 否 | string | | | |
| clipSessionId | 否 | string | | | 裁剪合并视频id |
| vid | 否 | string | | | 回放id，回放模式下传入该参数，可不传`fileId`、`url`、`sessionId` |
| barrage | 否 | boolean | | false | 是否开启弹幕 |
| defaultBarrageStatus | 否 | boolean | | true | 播放器加载时弹幕转台是否显示开启 |
| switchPlayer | 否 | boolean | | false | 是否在控制栏显示ppt与player切换按钮，用于实现主副屏切换，需要监听player实例的`'switchPlayer'`事件做处理，移动端不响应该事件<br>**注：仅在三分屏生效** |

| 参数名 | 必选 | 类型 | 可选值 | 默认值 | 说明 |
| --- | --- | --- | --- | --- | --- |
| controllerEl | 否 | HTMLElement | | | 控制栏区域，设置后控制栏不跟随ppt与播放器的位置，设置后点击全屏会将控制栏所在元素全屏，可以实现全屏后同时显示ppt与讲师<br>**注：仅在三分屏生效** |
| useH5Page | 否 | boolean | | false | 是否开启微信端H5page设置 |
| fullscreenEl | 否 | HTMLELement | | | 可设置点击控制栏全屏按钮进入全屏的元素，也可以使用 `liveSdk.player.setFullscreenEl(el?: HTMLElement): void` 更新设置，el为空时则使用默认值<br>**注：仅在PC端非IE的场景下生效** |

| 参数名 | 必选 | 类型 | 可选值 | 默认值 | 说明 |
| --- | --- | --- | --- | --- | --- |
| theme | 否 | object | | | 设置部分播放器样式<br>{<br>playerBackgroundImage：'string' \| false，// 讲师背景图，false 去掉默认背景图<br>playerBackgroundColor：'', // 讲师背景颜色(css颜色，如'red'、'#ffffff'、'rgb(0,0,0)')，注意设置后可能被背景图遮挡<br>pptBackgroundImage： '', // ppt背景图<br>pptBackgroundColor：'' // ppt背景颜色<br>} |

| 参数名 | 必选 | 类型 | 可选值 | 默认值 | 说明 |
| --- | --- | --- | --- | --- | --- |
| showPPTFullscreenButton | 否 | boolean | | false | 是否在ppt中显示全屏按钮<br>该全屏效果只是实现页面全屏，可能会与页面本身样式有冲突导致全屏时样式异常，所以开发时如果用到该功能注意看看有没有样式冲突，有的话可监听 `liveSdk.player.on('PPTFullscreenChange')` 事件进行调整<br>**注：仅在移动端生效** |
| warmUpImg | 否 | string | | | 设置暖场图片 |
| warmUpImgType | 否 | string | 'contain' \| 'cover' | 'cover' | 设置暖场图片的适配方式<br>'cover'：拉伸覆盖<br>'contain'：保持图片比例 |
| warmUpImgBgColor | 否 | string | | '#212121' | 十六进制颜色值,当warmUpImgType设置为contain 时,设置图片未覆盖区域的颜色<br>**注：仅在移动端生效** |

| 参数名 | 必选 | 类型 | 可选值 | 默认值 | 说明 |
| --- | --- | --- | --- | --- | --- |
| skinConfig | 否 | object | | | 修改播放器非直播状态、音频模式、直播暂停时占位图<br>{<br>streamStop: '' , // 图片地址，用于修改暂无直播时居中图标<br>streamStopTxt: '', // 用于修改暂无直播时显示的文字，默认为'当前暂无直播'，需要streamStop有效<br>streamPause: '', // 图片地址，用于修改直播暂停状态时居中图标<br>streamPauseTxt: '', // 用于修改直播暂停时显示的文字，默认为'休息一会 稍后继续'，需要streamPause有效<br>audioMode: '', // 图片地址，用于修改音频模式时居中图标<br>audioModeTxt: '', // 用于修改音频模式时显示的文字，默认为'当前为音频模式'，需要audioMode有效<br>audioModeVod: '', // 用于修改 |

| 参数名 | 必选 | 类型 | 可选值 | 默认值 | 说明 |
| --- | --- | --- | --- | --- | --- |
| | | | | | 音频模式回放背景占位图，优先级低于`audioMode`<br>bgColor: '', `//`设置`streamStop`,`streamPause`,`audioMode`后自定义的背景颜色<br>} |
| webviewAutoplay | 否 | boolean | | false | webview场景支持自动播放(需要webview支持自动播放) 非webview场景也可以使用，强制自动播放，安卓端可能会有黑屏问题<br>**注：仅在直播以及直播回放生效** |
| speed | 否 | boolean | | false | 是否显示切换倍速控件 |
| webPageFullScreen | 否 | boolean | | false | 是否开启网页全屏<br>**注：仅在移动端生效** |
| fullScreenOrientation | 否 | string | 'portrait' \| 'landscape' \| 'none' | 'none' | 网页全屏方向<br>**注：仅在移动端生效** |

## 2.重新加载播放器

说明：目前支持播放器播放类型为`live`场景的使用，比如当前客户端没有推流，用户打开了页面，过一段时间后客户端开始推流可以调用该方法刷新播放器
**使用：`liveSdk.reloadPlayer(): void`**

示例代码：

```
liveSdk.setupPlayer({
  // ...
  type: 'live'
  // ...
});

// 监听流状态变化刷新播放器
liveSdk.on(PolyvLiveSdk.EVENTS.STREAM_UPDATE, function() {
  console.log('流状态改变');
  liveSdk.reloadPlayer();
});
```

### 3.设置token

说明：SDK设置接口token，用于一些互动的功能接口的请求，如点赞（调用点赞前需要调用此方法设置token）。

参数：`token` 获取方法

**使用：** **`liveSdk.setApiToken(token: string)`**

示例代码：

```
liveSdk.setApiToken('token');
```

### 4.切换回放场次

说明：换回放场次，用于切换具体播放场次视频

参数：`sessionData` 需要播放场次的场次数据

**使用：** **`liveSdk.switchVod(sessionData: object)`**

回放操作demo

示例代码：

```
/**
 * 切换回放场次，用于切换具体播放场次视频
 *
 * @param {object} sessionData - 需要播放场次的场次数据
 * @return
 */

liveSdk.setupPlayer({
  // ...
  type: 'vod'
  fileId,
  url,
  sessionId
});
// data 数据是该场次数据，可通过 liveSdk.getPlaybackLists(...) 获取
liveSdk.switchVod({
  fileId: data.videoId,
  url: data.url,
  sessionId: data.channelSessionId
});

//or
liveSdk.setupPlayer({
  // ...
  type: 'vod'
  vid
});
liveSdk.switchVod({
  vid: data.videoPoolId
});
```

## 5.获取回放列表

说明：换回放场次，用于切换具体播放场次视频

**使用：** `liveSdk.getPlaybackLists(page: number, pageSize: number, type: 'playback'|'vod' = 'playback): Promise`

示例代码：

```
liveSdk.getPlaybackLists(1, 5)
.then(function(data) {
    console.log(data);
});
```

参数说明：

| 参数名 | 类型 | 说明 |
| --- | --- | --- |
| page | number | 页数 |
| pageSize | number | 每页条数 |
| type | string | playback: 回放列表<br>vod: 点播列表，点播列表需要用点播播放器播放 |

## 6.获取章节列表

说明：获取章节列表，仅在三分屏生效

**使用：** `liveSdk.getChapterLists(fileId: string, type: string): Promise`

示列代码：

```
/**
 * 获取章节列表，注：仅在三分屏生效
 * @return {Promise} - 标准 Promise 对象
 */
// 示例代码
liveSdk.getChapterLists('28acd4bd4526a6daa8368949e7c31713', 'record')
    .then(res => {
        console.log(res);
    })
```

参数说明：

| 参数名 | 类型 | 说明 |
| --- | --- | --- |
| fileId | string | ppt数据id |
| type | string | 章节类型<br>playback：回放<br>record: 暂存 |

## 7.发送聊天消息

说明：发送聊天消息

**使用：** `liveSdk.send(message: string, callback?: (data: string) => void): void`

示列代码：

```
/**
 * 发送聊天信息
 */
// 示例代码
liveSdk.send('这是一条聊天信息')
```

参数说明：

| 参数名 | 类型 | 说明 |
| --- | --- | --- |
| message | string | 聊天内容 |

## 8.向讲师发送提问私聊信息

说明：向讲师发送提问私聊信息

**使用：** `liveSdk.sendQuestion(message: string): void`

示列代码：

```
/**
 * 向讲师发送提问私聊信息
 */
// 示例代码
liveSdk.sendQuestion('这是一条提问信息')
```

参数说明：

| 参数名 | 类型 | 说明 |
| --- | --- | --- |
| message | string | 私聊内容 |

## 9.发送答题卡回答内容

说明：发送答题卡回答内容

**使用：** `liveSdk.sendAnswer(option: string, questionId: string, callback: Function): void`

示列代码：

```
/**
 * 发送答题卡回答内容。
 * 1.结合 PolyvLiveSdk.EVENTS.GET_TEST_QUESTION_CONTENT 使用，获取到 questionId。
 * 2.结合 PolyvLiveSdk.EVENTS.GET_TEST_QUESTION_RESULT 获取到答题卡结果，需要讲师点击发送结果。
 * 3.结合 PolyvLiveSdk.EVENTS.STOP_TEST_QUESTION，了解到讲师停止答题。
 */
// 示例代码
liveSdk.sendAnswer(
    option,
    questionId,
    function (res) {
        console.log(res);
    }
)
```

参数说明：

| 参数名 | 类型 | 说明 |
| --- | --- | --- |
| option | string | 答题卡答案， 'A' \| 'ABC' |
| questionId | string | 答题卡题目id |
| callback | Function | 发送成功回调 |

## 10.发送自定义消息

说明：发送自定义消息，聊天室会广播这些信息，由用户自行定义数据结构，除观看端外，其他端收到消息不会解析

**使用：** `liveSdk.sendCustomMessage(messsage: object, callback?: (data: Object) => void): void`

示例代码:

```
/**
 * 发送自定义消息，聊天室会广播这些信息，由用户自行定义数据结构
 *
 * @param {object} - config 配置对象
 */
// 示例代码
liveSdk.sendCustomMessage({
    EVENT: 'myMessage', // 自定义消息类型，事件字符长度不大于 20
    version: '1.0', // 自定义消息版本
    emitMode: 0, // 可选，是否广播给自己，0表示广播所有人包括自己，1表示广播给所有人除了自己，2表示只发送给自己
    roomId: channelId, // 频道号
    data: { // 自定义消息内容，由客户定义
        message: value
    },
    tip: '我的自定义消息', // 提示语
});

// 收到自定义消息（仅限于sendCustomMessage发送）
liveSdk.on(PolyvLiveSdk.EVENTS.CUSTOM_MESSAGE, function (event, data) {
    console.log('监听自定义消息: ',data);
});
```

## 11.发送点赞消息

说明：发送点赞消息。【使用点赞方法前，需确保已经调用了 liveSdk.setApiToken()】

**使用：** `liveSdk.sendLike(times: number): void`

示例代码:

```
/**
 * 发送点赞数
 */
// 示例代码
liveSdk.sendLike(1)
```

参数说明：

| 参数名 | 类型 | 说明 |
| --- | --- | --- |
| times | number | 点赞数量，最大不能超过30 |

## 12.获取历史聊天记录

说明：获取历史聊天记录，参数非必传，默认10条，配合`PolyvLiveSdk.EVENTS.HISTORY_MESSAGE`事件使用，收到事件可往聊天列表前面增加数据
**使用：** `liveSdk.getHistoryMessage(pageNum: number, start: number): liveSdk`

示例代码:

```
// 点击查看更多
var more = true;
$more.on('click', function() {
  if (!more) return;
  // 消息返回后在 `PolyvLiveSdk.EVENTS.HISTORY_MESSAGE` 中返回信息
  liveSdk.getHistoryMessage();
})

// 监听历史内容更新
liveSdk.on(PolyvLiveSdk.EVENTS.HISTORY_MESSAGE, function (event, data, begin, end) {
  console.log(event, data, begin, end);
  // 渲染数据
  render(data);
  if (data.length < 10) {
    console.warn('已经没有更多历史消息');
    more = false;
  }
});
```

参数说明：

| 参数名 | 类型 | 说明 |
| --- | --- | --- |
| pageNum | number | 需要获取的消息数量 |
| start | number | 代表距离最新一条聊天消息往前的第几条消息开始 |

## 13.销毁播放器

说明：销毁播放器
**使用：** `liveSdk.destroy(disconnectSocket: boolean = true): void`

示例代码:

```
/**
 * 销毁播放器
 *
 * @param {string} disconnectSocket - 是否断开 socket
 */
// 示例代码
liveSdk.destroy(true);
```

参数说明：

| 参数名 | 类型 | 说明 |
| --- | --- | --- |
| disconnectSocket | boolean | 是否断开 socket |

## 14.发送签到

说明：发送签到

**使用：** `liveSdk.toSign(checkinId: string, callback: Function): void`

示例代码:

```
/**
 * 发送签到。
 * 结合 PolyvLiveSdk.EVENTS.SIGN_IN 使用，收到讲师发起签到消息和获取到 checkinId。
 * 结合 PolyvLiveSdk.EVENTS.STOP_SIGN_IN 使用，收到讲师停止签到消息，收到后不允许发送签到
 */
// 示例代码
liveSdk.toSign(
    checkinId,
    res => console.log(res)
);
```

参数说明：

| 参数名 | 类型 | 说明 |
| --- | --- | --- |
| checkinId | string | 签到id |

| 参数名 | 类型 | 说明 |
| --- | --- | --- |
| callback | function | 发送签到回调，返回的 code 的说明如下<br>200：签到成功<br>301：签到的用户没有登录聊天室<br>302：房间不在签到状态中<br>303：该用户已签到 |

## 15.发送问卷答案

说明：发送问卷答案

**使用：** `liveSdk.sendQuestionnaireAnswer(questionnaireId: string, answer: [], callback: Function): void`

示例代码：

```
/**
 * 发送问卷答案
 * 结合 PolyvLiveSdk.EVENTS.START_QUESTIONNAIRE 使用，收到问卷消息和获取到 questionnaireId、questionId
 * 结合 PolyvLiveSdk.EVENTS.STOP_QUESTIONNAIRE 使用，了解到讲师停止问卷
 */
// 示例代码
liveSdk.sendQuestionnaireAnswer(
    questionnaireId,
    [
        {
            questionId: questionId,
            answer: 'A'
        }
    ],
    function(res){
        console.log(res);
    }
)
```

参数说明：

| 参数名 | 类型 | 说明 |
| --- | --- | --- |
| questionnaireId | string | 文件id |

| 参数名 | 类型 | 说明 |
| --- | --- | --- |
| answer | array | 答案列表<br>与文档 ANSWER_QUESTIONNAIRE 中的answer一致 |
| callback | function | 发送回调<br>200：答题成功<br>301：问卷已结束<br>302：该学生已经提交过答案<br>303：用户未登录，也就是当前用户不在聊天室 |

## 16.查询当前频道直播状态

说明：查询当前频道直播状态，仅在开启直播录制时有效。

**使用：** `liveSdk.getCloudClassRoomStatus(): Promise`

示例代码：

```
liveSdk.getCloudClassRoomStatus()
.then(function(resp) {
  if (resp.status === 'success') {
    // liveStart（开始上课）、liveRecovery（恢复）、liveSuspend（暂停）、liveFinish（结束）
    console.log('当前直播状态为:' + resp.data.status);
  }
});
```

## 17.修改昵称

说明：修改昵称，可监听 `PolyvLiveSdk.EVENTS.SET_NICK_STATUS` 事件获取修改状态。

**使用：** `liveSdk.setNick(newName: string): void`

示例代码：

```
// 调用该方法设置新的昵称
liveSdk.setNick('新的昵称');

// 监听修改状态
liveSdk.on(PolyvLiveSdk.EVENTS.SET_NICK_STATUS, function(event, resp) {
    if (resp.status === 'success') {
      // 修改成功，需要页面记录设置的昵称以便下次进来用同样的昵称
          console.log(`修改昵称成功，新的昵称为：${resp.nick}`)
    } else {
        console.log(`修改昵称失败 ${resp.message}`)
    }
});
```

# 实例 player 对象

`liveSdk.player`：调用 `setupPlayer` 后创建的播放器实例，可用于控制播放器播放相关交互

## 对象属性

demo

对象属性，什么时候可以获取到?

- 当调用 setupPlayer 后创建的播放器示例，才可以获取到的对象属性：currentTime、playbackRate、cameraStatus、supportFullScreen、fullScreen、barrageStatus、levels、level
- 需要等 loadedmetadata 事件后，才可以获取到的对象属性：duration、volume、paused、lines、line、currentPPTPage、totalPPTPages

示列代码：

```
// 示例代码
liveSdk.player.on('loadedmetadata', function () {
    console.log('获得当前视频时长');
    console.log(liveSdk.player.duration);
});
```

| 属性名 | 类型 | 返回说明 |
| --- | --- | --- |
| duration | number | 视频时长，单位：秒 |
| currentTime | number | 当前播放的进度 |

| 属性名 | 类型 | 返回说明 |
| --- | --- | --- |
| playbackRate | number | 当前倍速 |
| volume | number | 当前音量 |
| cameraStatus | boolean | 当前讲师摄像头是否被关闭 |
| paused | boolean | 当前是否是暂停播放状态 |
| supportFullScreen | boolean | 是否支持全屏 |
| fullScreen | object | 播放器全屏实例，里面封装了进入全屏以及退出全屏的方法，可对于页面任意DOM操作 |
| lines | number | 当前频道有多少条线路，需要在`loadedmetadata`时间后访问 |
| line | number | 当前播放的线路`0/1` |
| currentPPTPage | number | 当前展示的ppt页码，若当前显示PDF或者白板则返回1 |
| totalPPTPages | number | 当前使用的ppt总页数，若当前使用PDF或者白板则返回1 |
| barrageStatus | boolean | 当前弹幕状态 |
| levels | number | 当前的码率数，移动端暂不支持 |
| level | number | 当前的清晰度，`0`/`1`/`2`递增，如果频道只有一个码率则返回`0`，移动端暂不支持 |

#####全屏示例说明 `liveSdk.player.fullScreen.request($dom: HTMLElement): void`： 将指定元素进入全屏状态移动端中只在video中生效，具体需要看浏览器支持度 `liveSdk.player.fullScreen.exit(): void`： 退出全屏状态

示列代码：

```
console.log('获取播放器全屏实例', liveSdk.player.fullScreen);
// 将指定元素进入全屏状态移动端中只在video中生效，具体需要看浏览器支持度，注意，参数是 DOM 元素
liveSdk.player.fullScreen.request(document.querySelector("#ppt > div"))
// 退出全屏状态
liveSdk.player.fullScreen.exit()
```

## 对象方法

demo 示列代码：

```
liveSdk.player.setVolume(1); // 设置播放器音量
```

| 方法名 | 参数 | 说明 |
| --- | --- | --- |
| setVolume | （volume: number） | 设置播放器音量，范围[0, 1]<br>**移动端无效，采用手机自带音量控制** |
| play | | 恢复播放视频 |
| pause | | 暂停播放视频 |
| togglePlay | | 交替恢复暂停播放视频 |
| resize | | 刷新ppt尺寸，播放期间如果对ppt容器尺寸有改变ppt尺寸可能异常，在改变尺寸后调用该方法恢复正常尺寸<br>直播中设置无效 |
| seek | (time：number) | 请求到指定播放进度，单位：秒 |
| setRate | (rate: number) | 切换倍速，可选0.5、1、1.25、1.5、2 （移动端受设备限制，不支持切换倍速）<br>直播中暂不支持切换倍数 |
| switchCamera | (hide: boolean) | 是否隐藏讲师摄像头 |
| switchLine | (line: number) | 切换当前线路，line由number `0` 递增，代表线路1、线路2... |

| 方法名 | 参数 | 说明 |
| --- | --- | --- |
| switchPPTDocMode | (control: boolean) | 直播时切换ppt模式，设置`true` ppt可由用户自己控制ppt内容，不跟随客户端控制（自由模式），反之。调用返回是否切换成功（跟随模式） |
| nextPPTPage | | 使ppt切换到下一页，跟随模式下不能切到讲师未讲解的ppt |
| prevPPTPage | | 使ppt切换到上一页 |
| sendBarrage | (text: string, color: string) | 发送弹幕`text`为弹幕内容，`color`为弹幕颜色，如`'#ffffff'`;<br>必须设置barrage: true |
| hideBarrage | | 隐藏弹幕 |
| showBarrage | | 恢复弹幕显示 |
| toggleBarrage | | 交替隐藏弹幕 |
| resizeBarrage | | 刷新弹幕尺寸，播放器尺寸更新时调用 |
| switchLevel | (level: number) | 切换清晰度，由`0`递增 |
| changePPTNavVisible | (hide: boolean) | 隐藏显示PPT翻页控件 |
| setFullscreenEl | (el: HTMLElement) | 更新全屏元素设置，el为空时则使用默认值 |

## 对象事件

demo 示列代码：

```
liveSdk.player.on('playing', function () {
    console.log('恢复播放');
});
```

| 事件名称 | 说明 |
| --- | --- |
| playing | 恢复播放 |
| pause | 暂停播放 |
| timeupdate | 播放进度更新 |
| loadedmetadata | 视频加载成功可以准备播放，可对播放器进行play()、pause()等操作 |
| ended | 播放结束 |
| ratechange | 倍速改变 |
| volumechange | 音量改变 |
| lineChanged | 线路改变 |
| barrageStatusChange | 弹幕状态改变，返回`true`(开启弹幕)、`false`(关闭弹幕) |
| switchPlayer | 点击控制栏切换按钮触发，可用于切换ppt与讲师位置，切换样式需自身去实现 |
| levelChanged | 清晰度改变，返回切换后清晰度 |
| error | 视频播放错误 |
| switchMainScreen | 切换主副屏播放器回调，初始化播放器时会收到一次回调是当前初始值<br>'player' \| 'ppt' |
| mutedAutoplay | 静音自动播放回调，因为受限于浏览器自动播放策略，在不支持自动播放的时候会采取静音自动播放策略，这时会触发该事件，可用于对用户进行相应提示，让用户恢复音量<br>`在非无延迟的pc端才支持` |

## PPT操作相关事件

示列代码：

```
liveSdk.player.on('pptStatusChange', function(data) {
  /*
  {
    page: 3,  // 当前是第几页
    total: 30, // 当前ppt总页数
    type: 'ppt' // ('ppt'|'whiteboard') 当前显示类型，ppt或白板
  }
  */
  console.log(data);
});
```

| 事件名称 | 说明 |
| --- | --- |
| pptStatusChange | ppt状态改变，包括当前页数改变（ppt更换、ppt与白板切换），翻页，注意白板状态时不能翻页 |
| whiteboardOpened | 讲师打开白板 |
| whiteboardClosed | 讲师关闭白板，恢复为PPT |
| PPTFullscreenChange | ppt全屏状态变化，设置`showPPTFullscreenButton: true`后才会触发 |

### 弹幕使用说明

- 需要自行监听聊天室的聊天消息事件，并调用`liveSdk.player.sendBarrage`方法发送弹幕
- 弹幕显示在控制栏所在播放器，移动端固定显示在讲师画面那里
- 若播放器尺寸发生变化需要调用`liveSdk.player.resizeBarrage()`刷新弹幕

## 实例事件

### 事件方法

#### `liveSdk.on(event: string, func: Function)` 事件监听方法

示列代码：

```
/**
 * 事件监听方法
 *
 * @param {string} event - 绑定事件名称
 * @param {Function} func(event: string, data: object) - 事件回调函数
 */
// 示例代码
liveSdk.on(PolyvLiveSdk.EVENTS.CHANNEL_DATA_INIT, function (event, data) {
    console.log('事件：',event);
    console.log('数据：',data);
});
```

## 绑定事件

示列代码：

```
/**
 * 绑定事件，只触发一次
 *
 * @param {string} event - 绑定事件名称
 * @param {Function} func(event: string, data: object) - 事件回调函数
 */
// 示例代码
liveSdk.once(PolyvLiveSdk.EVENTS.CHANNEL_DATA_INIT, function (event, data) {
    console.log('事件：',event);
    console.log('数据：',data);
});
```

## 解除事件绑定

示列代码：

```
/**
 * 解除事件绑定
 *
 * @param {string} event - 绑定事件名称
 * @param {Function} func - 事件回调函数
 */
// 示例代码
liveSdk.off(PolyvLiveSdk.EVENTS.CHANNEL_DATA_INIT, function () {
    console.log('解除事件绑定');
});
```

# 事件列表

## 1.频道信息获取完成

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.CHANNEL_DATA_INIT(event: string, data: object)
 */
// 示例代码
liveSdk.on(PolyvLiveSdk.EVENTS.CHANNEL_DATA_INIT, function (event,data) {
    console.log('频道信息获取完成', data);
});
```

data返回参数详情

| 属性名 | 类型 | 说明 |
| --- | --- | --- |
| channelId | number | 直播频道ID |
| userId | string | 直播用户ID |
| name | string | 直播频道名称 |
| desc | string | 直播频道描述 |
| publisher | string | 主持人姓名 |
| likes | number | 当前点赞数 |
| pageView | number | 累计观看人数 |
| coverImage | string | 直播图标 |
| status | string | 直播状态 Y(当前正在直播)、N(当前暂无直播) |
| startTime | string | 直播开始时间 |
| stream | string | 频道流名 |
| roomId | string | 聊天室的房间号，如果频道未分房间，则为频道号 |

| 属性名 | 类型 | 说明 |
| --- | --- | --- |
| scene | string | 直播场景，alone(活动拍摄)、ppt(三分屏)、topclass(大班课) |
| splashEnabled | string | 引导图开关，Y(开)、N(关) |
| splashImg | string | 引导图片 |
| warmUpImg | string | 暖场图片地址 |
| warmUpFlv | string | 暖场视频地址 |
| playbackEnabled | string | 后台回放开关，Y(开)、N(关) |
| hasPlayback | boolean | 是否有回放视频 |
| sessionId | string | 频道当前场次信息，若当前未直播，则返回上一场直播的场次id |
| watchStatus | string | 观看页状态，live(直播中)、playback(回放中)、end(已结束)、waiting(未开始) |
| playbackOrigin | string | 回放视频来源,后台设置的回放类型，record(暂存列表)、playback(回放列表)、vod(点播列表) |
| resolutionWidth | number | 分辨率宽度 |
| resolutionHeight | number | 分辨率高度 |
| country | string | 城市 |
| watchThemeModel | object | 观看页主题信息<br>{<br>pageSkin: string;<br>defaultTeacherImage: string;<br>watchLayout: 'ppt' \| 'player'<br>} |

| 属性名 | 类型 | 说明 |
| --- | --- | --- |
| recordFileSimpleModel | object | 频道录制文件， 没有录制视频时，该字段值为null<br>{<br>m3u8: string \| null<br>mp4: string \| null<br>fileId: string;<br>channelSessionId: string;<br>duration: number;<br>[propName: string]: string \| number \| null<br>} |
| channelMenus | Array | 自定义菜单列表<br>[{<br>menuId: string, // 菜单id<br>menuType: string, // 菜单类型 desc(直播介绍)、tuwen(图文直播)、quiz(咨询提问)、text(图文菜单)、chat(互动聊天)、iframe(推广外链)<br>name: string, // 菜单名称<br>ordered: number, // 菜单排序，从1开始<br>content: string // 菜单内容<br>}] |

## 2.回放章节初始化完成

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.PLAYBACK_INIT(event: string, data: object)
 */
// 示例代码
liveSdk.on(PolyvLiveSdk.EVENTS.PLAYBACK_INIT, function (event,data) {
    console.log('回放章节初始化完成', data);

    // data返回数据
    {
         sessionId: '123',
         fileId: '123'
    }
});
```

## 3.流状态更新，可用来判断当前有无直播在推流

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.STREAM_UPDATE(event: string, status: 'live'|'end')
 */
// 示例代码
liveSdk.on(PolyvLiveSdk.EVENTS.STREAM_UPDATE, function (event,status) {
    console.log('流状态更新', status);
    liveSdk.reloadPlayer();// 刷新播放器
});
```

## 4.非本人的用户发言

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.SPEAK(event: string, data: object)
 */
// 示例代码
liveSdk.on(PolyvLiveSdk.EVENTS.SPEAK, function (event,data) {
    console.log('收到非本人的用户发言', data);

    // data返回聊天信息数据
    {
        EVENT: 'speak',
        content: 'test',  // 聊天具体内容
        currentUser: false, // 是否为当前用户
        custom: false, // 是否是自定义消息
        formatTime: '2019-07-02 14:12',
        id: '65975130-9c90-11e9-94b0-fd61e3983999',  // 消息id
        reward: false,
        time: 1562047960506,
        user: { // 用户信息
          banned: false,
          channelId: '275682',
          clientIp: '61.144.146.199',
          nick: 'polyv',
          pic: '//livestatic.videocc.net/v_314/assets/wimages/missing_face.png',
          roomId: '275682',
          sessionId: 'f8qnsda4zq',
          uid: 'Ph4UtYfnxQo7pZK6AkbP',
          userId: '1562047951865',
          userType: 'slice',
        }
    }
});
```

## 5.本人的发言

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.SEND_MESSAGE(event: string, data: object)
 */
// 示例代码
liveSdk.on(PolyvLiveSdk.EVENTS.SEND_MESSAGE, function (event,data) {
    console.log('本人的发言', data);

    // data返回聊天信息数据，和 非本人的用户发言  返回的数据一致
})
```

## 6.发送提问消息回调

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.S_QUESTION(event: string, data: object)
 */
// 示例代码
liveSdk.on(PolyvLiveSdk.EVENTS.S_QUESTION, function (event,data) {
    console.log('发送提问消息回调', data);

    // data返回数据
    {
        EVENT: 'S_QUESTION',
        content: '1+1=多少',
        currentUser: true,
        custom: false,
        reward: false,
        roomId: '275682',
        user: {
          // ... 用户信息
        }
    }
})
```

## 7.管理员或讲师回答提问

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.T_ANSWER(event: string, data: object)
 */
// 示例代码
liveSdk.on(PolyvLiveSdk.EVENTS.T_ANSWER, function (event,data) {
    console.log('管理员或讲师回答提问', data);

    // data返回数据
    {
        EVENT: 'T_ANSWER'
        content: '等于2'
        roomId: '275682'
        s_userId: '1559030433536', // 被回答用户id
        user: {
          // ... 用户信息
        }
    }
})
```

## 8.收到自定义消息

代码示列：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.CUSTOMER_MESSAGE(event: string, data: object)
 * 一般由 https://help.polyv.net/#/live/api/v4/chat/send_custom_message 发起才会触发此回调
 */
// 示例代码
liveSdk.on(PolyvLiveSdk.EVENTS.CUSTOMER_MESSAGE, function (event,data) {
    console.log('收到自定义消息', data);

    // data返回数据,具体数据结构由用户自行定义
})
```

## 9.清除聊天记录回调

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.REMOVE_HISTORY
 * 清除聊天记录，页面可收到该消息后清除页面聊天内容
 */
// 示例代码
liveSdk.on(PolyvLiveSdk.EVENTS.REMOVE_HISTORY, function () {
    console.log('清除聊天记录回调');
})
```

## 10.清除某一条聊天信息回调

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.REMOVE_CONTENT(event: string, data: object)
 * 清除某一条聊天信息，页面可收到该消息后清除相应聊天内容
 */
// 示例代码
liveSdk.on(PolyvLiveSdk.EVENTS.REMOVE_CONTENT, function (event,data) {
    console.log('清除某一条聊天信息', data);

    // data返回数据,被删除消息内容
    {
        EVENT: 'removeContent',
        id: '65975130-9c90-11e9-94b0-fd61e3983999', // 被删除消息的id
        roomId: '275682'
    }
})
```

## 11.获取历史聊天信息完成

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.HISTORY_MESSAGE(event: string, data: object)
 * 获取历史聊天信息完成
 */
// 示例代码
liveSdk.on(PolyvLiveSdk.EVENTS.HISTORY_MESSAGE, function (event,data) {
    console.log('获取历史聊天信息完成', data);

    // data返回数据,历史聊天信息，内容为speak事件内容组成的数组
})
```

## 12.发言失败

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.PROHIBIT_TO_SPEAK
 * 发言失败，可能由于聊天室为连接但发送聊天消息等发送错误触发
 */
// 示例代码
liveSdk.on(PolyvLiveSdk.EVENTS.PROHIBIT_TO_SPEAK, function () {
    console.log('发言失败', data);
})
```

## 13.用户被禁止发言

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.ADD_SHIELD
 * 用户被禁止发言
 */
// 示例代码
liveSdk.on(PolyvLiveSdk.EVENTS.ADD_SHIELD, function (event, data) {
    console.log('用户被禁止发言', data);

     // data返回数据，可以根据data里面返回的userId来判断是哪个用户
})
```

## 14.用户被恢复发言

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.REMOVE_SHIELD
 * 用户被恢复发言
 */
// 示例代码
liveSdk.on(PolyvLiveSdk.EVENTS.REMOVE_SHIELD, function (event, data) {
    console.log('用户被恢复发言', data);

    // data返回数据，可以根据data里面返回的userId来判断是哪个用户
})
```

## 15.频道内用户登录事件

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.LOGIN(event: string, data: object)
 * 频道内用户登录事件
 */
// 示例代码
liveSdk.on(PolyvLiveSdk.EVENTS.LOGIN, function (event,data) {
    console.log('频道内用户登录事件', data);

    // data返回数据,用户信息
    {
        EVENT: 'login',
        onlineUserNumber: 3, // 当前在线人数
        timeStamp: 1562051459569,
        user: {
          // 用户信息
        }
    }
})
```

## 16.频道内用户退出事件

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.LOGOUT(event: string, data: object)
 * 频道内用户退出事件
 */
// 示例代码
liveSdk.on(PolyvLiveSdk.EVENTS.LOGOUT, function (event,data) {
    console.log('频道内用户退出事件', data);

    // data返回数据,用户信息
    {
        EVENT: 'loginOut',
        channelId: '275682',
        onlineUserNumber: 2,  //当前在线人数
        roomId: '275682',
        timeStamp: 1562051345281,
        uid: 'LVexVHpKvK0KOU_0Ak4L',
    }
})
```

## 17.频道内用户踢出房间

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.LOGIN_REFUSE(event: string, data: object)
 * 频道内有用户被踢出房间，包含用户以及本身
 * 一般应用于再次登录后需要获取是否被踢出房间
 */
// 示例代码
liveSdk.on(PolyvLiveSdk.EVENTS.LOGIN_REFUSE, function (event,data) {
    console.log('频道内有用户被踢出房间，包含用户以及本身', data);

    // data返回数据，被踢用户数据
    {
        EVENT: 'LOGIN_REFUSE',
        data: {referField: '1559030433537', type: 'userId'}
    }
})
```

## 18.当前用户被踢出房间

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.BAN_USER_ROOM
 * 当前用户被踢出房间，这时候会自动断开聊天室连接
 */
// 示例代码
liveSdk.on(PolyvLiveSdk.EVENTS.BAN_USER_ROOM, function () {
    console.log('当前用户被踢出房间');
})
```

## 19.收到讲师发起签到消息

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.SIGN_IN(event: string, data: object)
 * 收到讲师发起签到消息,可调用 `liveSdk.toSign` 方法发送签到
 */
// 示例代码
liveSdk.on(PolyvLiveSdk.EVENTS.SIGN_IN, function (event,data) {
    console.log('收到讲师发起签到消息', data);

    // data返回数据，签到数据
    {
        EVENT: 'SIGN_IN',
        data: {
          checkinId: '07ff9330-9c99-11e9-8aa1-37da87', // 签到id
          createTime: 1562051668963,
          limitTime: '30', // 签到时间，页面需要根据这个时间显示倒计时，结束后不允许再签到
          message: '各位同学开始签到了' // 签到提示文案
        }
        roomId: '275682'
    }
})
```

## 20.收到讲师停止签到消息

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.STOP_SIGN_IN
 * 收到讲师停止签到消息，收到后不允许发送签到
 */
// 示例代码
liveSdk.on(PolyvLiveSdk.EVENTS.STOP_SIGN_IN, function () {
    console.log('收到讲师停止签到消息');
})
```

## 21.收到公告消息

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.BULLETIN(event: string, data: object)
 */
liveSdk.on(PolyvLiveSdk.EVENTS.BULLETIN, function (event,data) {
    console.log('收到公告消息', data);

    // data返回数据，公告消息
    {
        EVENT: 'bulletin',
        content: '公告内容'
    }
})
```

## 22.删除公告消息

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.REMOVE_BULLETIN
 * 删除公告消息
 */
liveSdk.on(PolyvLiveSdk.EVENTS.REMOVE_BULLETIN, function () {
    console.log('删除公告消息');
})
```

## 23.收到问卷消息

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.START_QUESTIONNAIRE(event: string, data: object)
 * 收到问卷消息，回答后可使用`liveSdk.sendQuestionnaireAnswer` 方法发送答案
 */
liveSdk.on(PolyvLiveSdk.EVENTS.START_QUESTIONNAIRE, function (event,data) {
    console.log('收到问卷消息', data);

    // data返回数据，问卷内容,与polyv 聊天室 START_QUESTIONNAIRE 事件数据一致
})
```

## 24.讲师停止问卷

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.STOP_QUESTIONNAIRE(event: string, data: object)
 */
liveSdk.on(PolyvLiveSdk.EVENTS.STOP_QUESTIONNAIRE, function (event,data) {
    console.log('讲师停止问卷', data);

    // data返回数据，问卷ID
})
```

## 25.获取问题内容

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.GET_TEST_QUESTION_CONTENT(event: string, data: object)
 */
liveSdk.on(PolyvLiveSdk.EVENTS.GET_TEST_QUESTION_CONTENT, function (event,data) {
    console.log('获取问题内容', data);

    // data返回数据，答题卡消息
})
```

## 26.获取答题结果

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.GET_TEST_QUESTION_RESULT(event: string, data: object)
 */
liveSdk.on(PolyvLiveSdk.EVENTS.GET_TEST_QUESTION_RESULT, function (event,data) {
    console.log('获取答题结果', data);
})
```

## 27.停止答题

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.STOP_TEST_QUESTION(event: string, data: object)
 */
liveSdk.on(PolyvLiveSdk.EVENTS.STOP_TEST_QUESTION, function (event,data) {
    console.log('停止答题', data);
})
```

## 28.直播录制暂停

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.ON_PAUSE_RECORD(event: string, data: object)
 */
liveSdk.on(PolyvLiveSdk.EVENTS.ON_PAUSE_RECORD, function (event,data) {
    console.log('直播录制暂停（课间休息）', data);
})
```

## 29.恢复录制

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.ON_START_RECORD(event: string, data: object)
 */
liveSdk.on(PolyvLiveSdk.EVENTS.ON_START_RECORD, function (event,data) {
    console.log('恢复录制（休息结束，恢复直播）', data);
})
```

## 30.退出录制

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.ON_EXIT_RECORD(event: string, data: object)
 */
liveSdk.on(PolyvLiveSdk.EVENTS.ON_EXIT_RECORD, function (event,data) {
    console.log('退出录制（下课）', data);
})
```

## 31.下课/结束直播

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.FINISH_CLASS(event: string, data: object)
 */
liveSdk.on(PolyvLiveSdk.EVENTS.FINISH_CLASS, function (event,data) {
    console.log('讲师点击下课按钮下课/结束直播', data);
})
```

## 32.修改昵称状态回调

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.SET_NICK_STATUS(event: string, data: object)
 */
liveSdk.on(PolyvLiveSdk.EVENTS.SET_NICK_STATUS, function (event,data) {
    console.log('修改昵称状态回调', data);
})
```

## 33.点赞事件

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.LIKES(event: string, data: object)
 */
liveSdk.on(PolyvLiveSdk.EVENTS.LIKES, function (event,data) {
    console.log('收到点赞事件', data);
})
```

## 34.最大并发登录事件

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.MAX_VIEWER_LOGIN(event: string)
 */
liveSdk.on(PolyvLiveSdk.EVENTS.MAX_VIEWER_LOGIN, function (event) {
    console.log('收到最大并发登录事件', data);
    // TODO：跳出页面/页面提示
})
```

## 35.频道内同一用户重复登录事件

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.RELOGIN(event: string, data: object)
 * 频道内同一用户重复登录事件
 */
// 示例代码
liveSdk.on(PolyvLiveSdk.EVENTS.RELOGIN, function (event,data) {
    console.log('频道内同一用户重复登录事件', data);

    // data返回数据,用户信息
    {
        EVENT: 'RELOGIN',
        channelId: 'channelId',
        user: {
          // 用户信息
        }
    }
})
```

## 36.频道信息初始化失败回调

示列代码：

```
/**
 * 事件名称：PolyvLiveSdk.EVENTS.CHANNEL_DATA_INIT_ERROR(error: string, msgDetails: string)
 * 频道初始化失败的回调
 */
// 示例代码
liveSdk.on(PolyvLiveSdk.EVENTS.CHANNEL_DATA_INIT_ERROR, function (error, msgDetails) {
    console.log('频道初始化失败', error, msgDetails);
})
```