

根据本文指导快速集成 TTT Web SDK，基于本 SDK 实现音视频互动直播。

阅读本文档需要有es6、npm基础

本地调试仅localhost / 127.0.0.1可以正常使用，如需IP访问，请使用HTTPS部署本地环境

线上生产环境请确保使用HTTPS部署，以保证功能正常使用

DEMO 体验

我们在 [GitHub](#) 上提供一个开源的基础视频互动直播示例项目，在开始开发之前你可以通过该示例项目体验互动直播效果。示例代码包括两部分：

1. [demo](#) - 用户可使用该 demo 体验 websdk 的基本音视频功能；
2. [samples](#) - 开发者可以参考其中一系列的 sample，基于 TTTWebSDKDemo 开发。

注：以上 *demo* | *samples*，源代码均位于 [GitHub](#)

名词解释

在使用tttwebsdk时，你主要操作两种对象：

- Client - 一个客户端。Client类的方法提供了加入房间、发布Stream、接收房间事件等方法。
- Stream - 本地音视频流以及远端音视频流。Stream类的方法提供了调用本地音视频设备、播放远端音视频等方法。

集成 SDK

- 项目中引入tttwebsdk

有两种方式，引入 tttwebsdk：

1. npm node_module

安装tttwebsdk

```
npm install tttwebsdk --save
```

```
import TTRtcWeb from 'tttwebsdk';
```

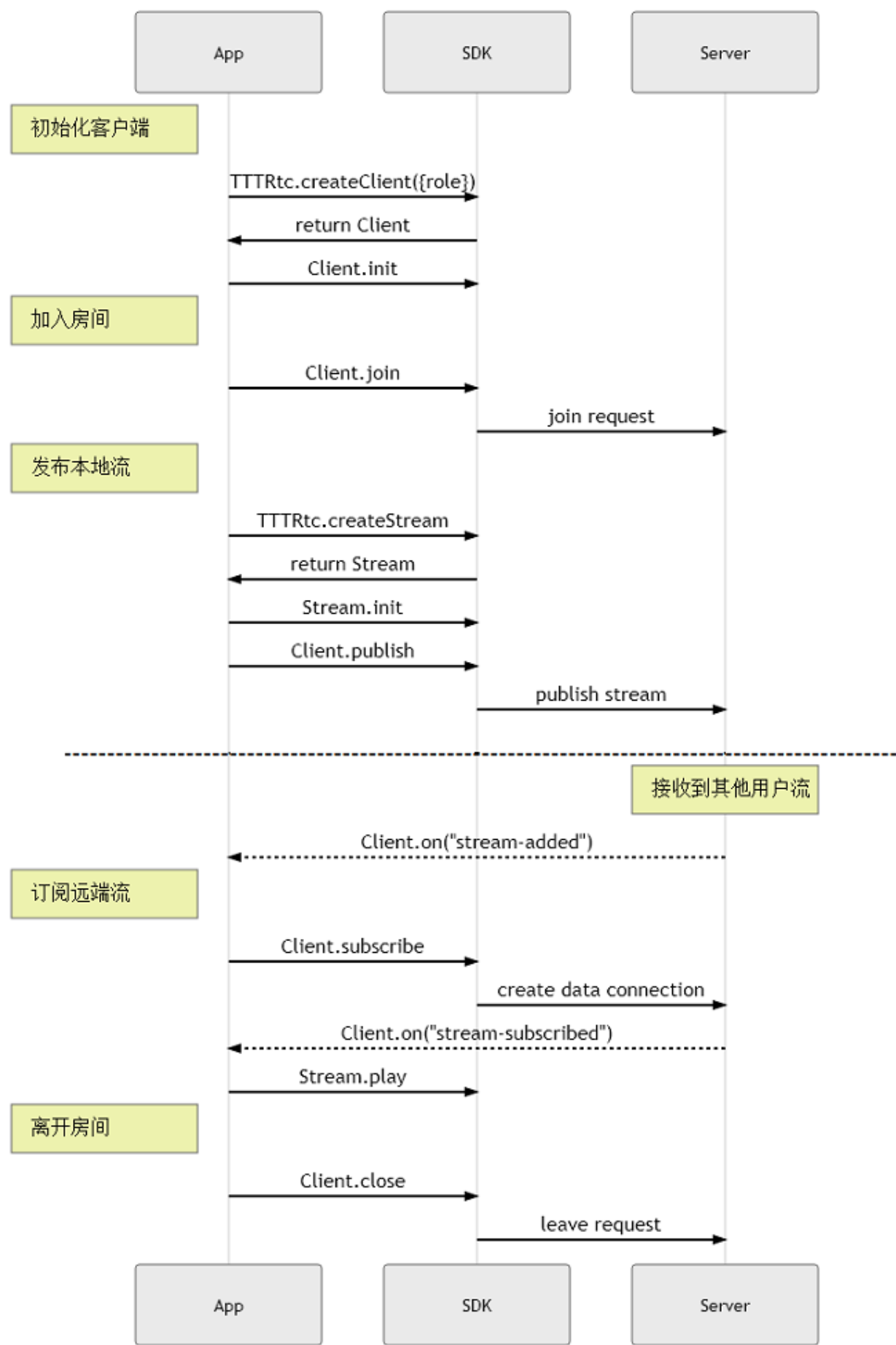
2. js文件

下载 tttwebsdk.js 文件，保存到目录（如 ./lib）

```
import TTRtcWeb from './lib/tttwebsdk';
```

加入直播房间

下图展示了加入直播房间的基本流程（图中的方法是对不同的对象调用的）



本文为快速入门指导，介绍 TTTWebSDK 基本用法。完整的 API 方法和回调详见 [WebSDK API 参考](#)

下面是代码具体实现

1. 加入房间

```
import TTTRtcWeb from 'ttswebsdk';
let TTTRtc = new TTTRtcWeb();

const APP_ID = 'xxxxxxxxxxxxxxxxxxxx';
const USER_ID = 91619846;
const ROOM_ID = 31819684;

/*创建本地客户端，并设置加入房间的角色为主播
 * role : 1 主播 | 2 副播 | 3 观众 )
 * rtmpUrl : 设置 cdn 推流地址
 */
let client = TTTRtc.createClient({role: 1, rtmpUrl: 'rtmp url'});

/*初始化客户端*/
client.init(APP_ID, USER_ID, ()=>{
  /*client.init success callback*/
  // 注: token 可以设置为 ''
  client.join(token, ROOM_ID, ()=>{
    console.log("join room successful")
  }, (error)=>{
    console.error("join room fail: ", error)
  })
}, (error) => {
  /*client.init fail callback*/
})
```

注：APP_ID: 类型为字符串，请联系三体云获取; USER_ID: 类型为整型，大于0且小于 2^{32} ，用户在该房间内的唯一标识; ROOM_ID: 类型为整型，大于0且小于 2^{32} ，要加入的房间号，如果该房间不存在，则会创建该房间。

2. 监听房间相关事件

这些事件是服务器收到其他用户的加入/离开房间、发布流等事件后，推送到client的事件。

```
client.on("peer-join", (evt) => {console.log("remote user joined: ", evt) })
// 有用户加入

client.on("peer-leave", (evt) => {console.log("remote user left: ", evt) })
// 有用户离开

// 有音频流加入 -- 可通过 client.subscribe 订阅该流播放
client.on('audio-added', (evt) => {
  var stream = evt.stream;
  if (!stream)
    return;
```

```

        client.subscribe(stream, (event) => {
            // successful doing someting, like play remote video or audio.
        }, (err) => {
            // some error occur
        });
    });

    client.on('video-added', (evt) => {
        var stream = evt.stream;
        if (!stream)
            return;

        client.subscribe(stream, (event) => {
            // successful doing someting, like play remote video or audio.
        }, (err) => {
            // some error occur
        });
    });

    // 某路流订阅成功，将执行 play
    client.on("stream-subscribed", (evt) => {
        // 订阅成功
        let { stream } = evt;

        if(stream.type === 'audio') { // 如果stream是纯音频流，不用传elementID
            stream.play()
        } else {
            let video = document.createElement('video');
            video.id = `3t_remote_${stream.innerStreamID}`;
            video.muted = false;
            video.autoplay = true;
            video.controls = true;
            video.setAttribute('playsinline', '');
            video.style.cssText = 'height: 300px; width: 300px; background: black; position:relative; display:inline-block;';
            videoEle.append(video);

            stream.play(video.id, true);
        }
    });

    client.on("stream-unsubscribed", (evt) => {}) // 已取消订阅的流

    client.on("disconnected", (evt) => {}) // 已经和服务端断开链接
    //... 更多事件请参考三体云websdk文档

```

注：我们建议在 createClient 成功后就监听这些事件。

3. 发布本地音视频流

在加入房间（client.join）成功后，创建一个本地音视频流。

```
let localStream = TTRtc.createStream({
  userId: USER_ID,
  audio: true,
  video: true
})
```

调用 `Stream.init` 方法初始化创建的流，成功后用`client.publish`发布这个流。

```
localStream.init(() => {
  console.log("localStream.init success")

  // 初始化本地流成功，使用client.publish发布这路流
  client.publish(localStream, () => {
    // 发布成功，房间内其他人会收到 video-added | audio-added 事件
    console.log("client.publish success")
  }, (error) => {
    console.log("client.publish fail", error)
  })
}, (error) => {
  console.log("localStream.init fail: ", error)
})
```

4. 离开房间

```
localStream.close();
client.leave();
client.close();
```