

## 14 | 如何打开/关闭音视频？

2019-08-15 李超

从0打造音视频直播系统

[进入课程 >](#)



讲述：李超

时长 10:08 大小 9.29M



在实时互动直播系统中，打开 / 关闭音视频流是很常见的需求。作为一个直播用户，你至少会有下面几种需求：

**将远端的声音静音。**比如来了一个电话，此时，应该先将直播中远端的声音关掉，等接完电话再将远端的声音打开，否则电话的声音与直播远端的声音会同时播放出来。

**将自己的声音静音。**比如老板要找你谈话，这时你应该将直播中自己的声音静音，否则你与老板的一些私密谈话会被远端听到。比如被老板骂了，要是被远端听到可就尴尬了。

**关闭远端的视频。**这个与远端声音静音差不多，只不过将声音改为视频了。比如当机子性能比较差的时候，为了节省资源，你可能会选择将远端的视频关闭掉。不过这种情况不是很多。

**关闭自己的视频。**当你不想让对方看到自己的视频时，就可以选择关闭自己的视频。比如今天你的状态特别不好，你又特别在乎你的形象，此时你就可以选择关闭自己的视频。

这几个功能是实时互动直播中的必备功能。因此，在开发实时互动直播系统时一定要将这些功能添加到你的系统中，那该如何实现它们呢？

## 基本逻辑

针对上面的问题，本节我们就讨论一下如何才能实现这几个功能。下面我们就按需求分别对这几个功能做详细的分析。

### 1. 将远端的声音静音

要实现这个功能，你可以通过在**播放端控制**和**发送端控制**两种方式实现。

在播放端有两种方法，一种是**不让播放器播出来**，另一种是**不给播放器喂数据，将收到的音频流直接丢弃**。在播放端控制的优点是实现简单；缺点是虽然音频没有被使用，但它仍然占用网络带宽，造成带宽的浪费。

在发送端控制也可以细分成两种方法实现，即**停止音频的采集**和**停止音频的发送**。对于 1 对 1 实时直播系统来说，这两种方法的效果是一样的。但对于多对多来说，它们的效果就大相径庭了。因为停止采集音频后，所有接收该音频的用户都不能收到音频了，这显然与需求不符；而停止向某个用户发送音频流，则符合用户的需求。

### 2. 将自己的声音静音

无论是 1 对 1 实时互动，还是多人实时互动，它的含义都是一样的，就是所有人都不能听到“我”的声音。因此，你只需**停止对本端音频数据的采集**就可以达到这个效果。

### 3. 关闭远端的视频

它与将“远端的声音静音”是类似的，要实现这个功能也是分为从播放端控制和从发送端控制两种方式。

不过它与“将远端的声音静音”也是有区别的，那就是：

从播放端控制**只能使用不给播放器喂数据这一种方法**，因为播放器不支持关闭视频播放的功能；

从发送端控制是**通过停止向某个用户发送视频数据这一种方法**来实现的。而另一个停止采集则不建议使用，因为这样一来，其他端就都看不到你的视频了。

## 4. 关闭自己的视频

其逻辑与“将自己的声音静音”相似。但你不应该关闭视频的采集，而应该通过**关闭所有视频流的发送**来实现该需求。之所以要这样，是因为视频还有本地预览，只要视频设备可用，本地预览就应该一直存在。所以，“关闭自己的视频”与“将自己的声音静音”的实现是不一样的。

## 代码实现


了解了以上如何打开 / 关闭音视频的基本逻辑后，下面我们就具体实操起来，来看一下如何通过代码实现上面那些功能吧！

### 1. 将远端的声音静音

前面在分析基本逻辑时，我们讲过这个功能可以从播放端控制和发送端控制这两种方式来实现，而每种方式又对应两种方法，所以一共有四种方法。下面我们就来逐个分析和实现。

#### (1) 播放端控制：不播放声音


在播放端控制的代码特别简单，你只需要在 `<video>` 标签中设置 **muted** 即可，代码如下：

 复制代码

```
1 <HTML>
2 ...
3 <video id=remote autoplay muted playsinline/>
4 ...
5 </HTML>
```

当`<video>`标签设置了 `muted` 属性后，你会发现虽然将远端获取到音视频流赋值给 `<video>` 标签进行播放，但最后只有视频被显示出来了，声音不播放。这样也就达到你的预期了。


其实，要想让音频播放出来也很容易，只需写一行 JavaScript 代码，将 muted 属性设置为假即可。

 复制代码

```
1 ...
2 var remotevideo = document.querySelector('video#remote');
3 remotevideo.muted = false;
4 ...
```

## (2) 播放端控制：丢掉音频流

当然在播放端还有另外一种办法实现远端的静音，即在收到远端的音视频流后，将远端的 AudioTrack 不添加到要展示的 MediaStream 中，也就是让媒体流中不包含音频流，这样也可以起到静音远端的作用。具体代码如下：


 复制代码

```
1 ...
2 var remoteVideo = document.querySelector('video#remote');
3 ...
4 {
5     // 创建与远端连接的对象
6     pc = new RTCPeerConnection(pcConfig);
7     ...
8     // 当有远端流过来时，触发该事件
9     pc.ontrack = getRemoteStream;
10    ...
11 }
12 ...
13
14 function getRemoteStream(e){
15     // 得到远端的音视频流
16     remoteStream = e.streams[0];
17     // 找到所有的音频流
18     remoteStream.getAudioTracks().forEach((track)=>{
19         if (track.kind === 'audio') { // 判断 track 是类型
20             // 从媒体流中移除音频流
21             remoteStream.removeTrack(track);
22         }
23     });
24     // 显示视频
25     remoteVideo.srcObject = e.streams[0];
26 }
27 ...
```

在上述代码中，实现了 ontrack 事件的处理函数。在该函数中首先保存远端传来的音视频流，然后将其中的音频轨去掉，最后将流赋值给<video>标签的 srcObject 域，这样播放器就只能播放视频了。

### (3) 发送端控制：不采集音频

通过远端不采集音频的方法也可以达静音的效果。那如何才能让远端知道你想让它静音呢？这就要通过信令通知了。本地想让远端静音时，首先向信令服务器发送一条静音指令，然后信令服务器向远端转发该指令，远端收到指令后就执行下面的代码：

 复制代码

```
1 ...
2
3 // 获取本地音视频流
4 function gotStream(stream) {
5     localStream = stream;
6     localVideo.srcObject = stream;
7 }
8
9 // 获得采集音视频数据时限制条件
10 function getUserMediaConstraints() {
11
12     var constraints = {
13         "audio": false,
14         "video": {
15             "width": {
16                 "min": "640",
17                 "max": "1280"
18             },
19             "height": {
20                 "min": "360",
21                 "max": "720"
22             }
23         }
24     };
25
26     return constraints;
27 }
28
29 ...
30 // 采集音视频数据
31 function captureMedia() {
32     ...
```

```


33         if (localStream) {
34             localStream.getTracks().forEach(track => track.stop());
35         }
36         ...
37         // 采集音视频数据的 API
38         navigator.mediaDevices.getUserMedia(getUserMediaConstraints())
39             .then(gotStream)
40             .catch(e => {
41                 ...
42             });
43     }
44     ...

```

上面的代码非常简单，captureMedia 函数用于采集音视频数据，在它里面实际是调用的浏览器 API getUserMedia 进行具体操作的。由于这里强调的是不采集音频数据，所以你可以看到在 getUserMediaConstraints 函数中，将音频关掉了，所以最后获取到的流中只有视频数据。

#### (4) 发送端控制：关闭通道

通过远端关闭通道的方式也可以达到静音的效果。与方法 3 不采集音频类似，本地想让远端静音时，向信令服务器发送一条静音指令，信令服务器进行转发，远端收到指令后执行下面的代码：

 复制代码

```

1     ...
2     var localStream = null;
3
4     // 创建 peerconnection 对象
5     var pc = new RTCPeerConnection(server);
6     ...
7
8     // 获得流
9     function gotStream(stream){
10         localStream = stream;
11     }
12     ...
13
14     //peerconnection 与 track 进行绑定
15     function bindTrack() {
16         //add all track into peer connection
17         localStream.getTracks().forEach((track)=>{
18             if(track.kind !== 'audio') {

```

```
19         pc.addTrack(track, localStream);
20     }
21     });
22 }
23
24 ...
```

在上面的代码中，在 `getUserMedia` 函数的回调函数中获得本地媒体流，然后在将其与 `RTCPeerConnection` 对象进行绑定时，对 `track` 做判断，如果是音频就不进行绑定，关闭了通道，这样对方就收不到音频数据了，从而达到远端静音的效果。

以上就是将远端静音的四种方法，接下来我们再来看看如何将自己的声音静音。

## 2. 将自己的声音静音

将自己的声音静音只需要在采集时停止对音频数据进行采集就可以了。它与上面“将远端声音静音”中的方法 3（不采集音频）是一样的，只需将 `constraints` 中的 `audio` 属性设置为 `false` 就好了。这里我就不再赘述了。

## 3. 关闭远端的视频

在前面讲解基本逻辑时，我们分析过关闭远端的视频有两种方法，一种是在显示端不将视频数据给 `video` 标签来达到不显示视频的效果，另一种是控制远端不发送数据。

实际上这两种方式与将远端声音静音中的方法 2 和方法 4 是一样的，只不过在做类型判断时，需要将 `'audio'` 修改为 `'video'` 就好了。因此，这里我也不再进一步介绍了。

## 4. 关闭本地视频

最后一个关闭本地视频，因不同的需求有不同的实现，一般情况下由于还涉及到本地视频的预览，所以在关闭本地视频时不是直接在采集的时候就不采集视频数据，而是不将视频数据与 `RTCPeerConnection` 对象进行绑定。具体的代码参考“将远端声音静音”中的方法 4。

## 小结



通过上面的介绍，我想你应该已经对直播系统中如何打开 / 关闭音视频有了非常系统而又清楚的认识。在你的产品中具体该使用哪种方法来实现音视频的打开与关闭，主要还得看你的产品需求。

如果想为用户节省流量，那就尽可能在远端进行控制。如果觉得流量不是特别重要的问题，为了实现简单，直接在本地处理也是没有任何问题的。

音视频的打开 / 关闭逻辑上虽并不太复杂，但在直播系统是却是必不可少的功能。另一方面，如果你这个逻辑不清楚的话，也就是不知道分为本地静音、远端静音等逻辑的话，在多人实时互动的场景中，实现这部分代码时就很容易被绕进去。所以这部分的知识你最好还是要提前搞清楚、学透彻。

## 思考时间

上面我向你介绍了打开 / 关闭音视频的各种情况和实现方法，但列得还并不全面，你还能想到其他可以打开 / 关闭音视频的方法吗？

欢迎在留言区与我分享你的想法，也欢迎你在留言区记录你的思考过程。感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。



# 从 0 打造音视频直播系统

手把手教你打造实时互动音视频直播系统

李超

新东方音视频直播技术专家  
前沪江音视频架构师



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。



上一篇 13 | 在WebRTC中如何控制传输速率呢？

下一篇 15 | WebRTC中的数据统计原来这么强大（上）

## 精选留言 (5)

写留言



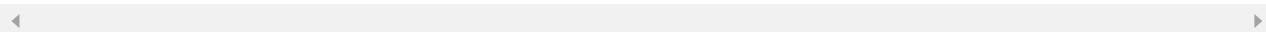
frank

2019-08-17

SFU服务临时静音，都是服务器端停止转发该rtp实现

展开

作者回复: 是的



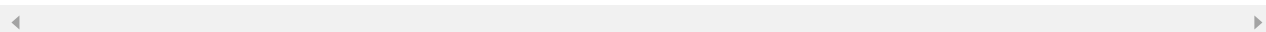
hao11111205

2019-08-16

将自己的声音静音的其它方法：将本地的麦克风静音；将远端的声音静音的其它方法：将本地的喇叭静音。主要是通过控制音频设备来实现。

展开

作者回复: 将本地的麦克静音后，是对方听不到了，不是你的喇叭不响哈，你再仔细想想！



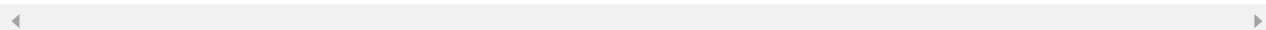
君

2019-08-15

老师，请问下是什么原因导致我的ice回调方法didChangeIceGatheringState这么慢呢，从Gathering到Complete的过程需要5秒至10秒

展开

作者回复: 其实这个状态你不必特别关心的，它是由底层自己完成的，不是说只有 complete之后才进行连接。你只要每次收到回调，就将它发送给对方，这样双方是在 gathering的状态下就开始尝试连接的建立的。对于本地收集 candidate 来说是特别快的，你的收集之所以慢，应该是你使用了 stun 服务导致的，主要还是你的 stun服务器的与你的主机之间的网络问题





**许童童**

2019-08-15

你还能想到其他可以打开 / 关闭音视频的方法吗？

我能想到通过物理的方式，比如拔掉开关，关闭电源。或者通过硬件控制器。

展开 ∨

作者回复: 关闭电源...,兄弟你够猛，哈哈！



**Jason**

2019-08-15

其他可以打开 / 关闭音视频的方法：通过更新sdp的视频、音频信息，是不是也可以呢？

作者回复: 是的，具体的做法呢？

