

05 | 原来浏览器还能抓取桌面？

2019-07-25 李超

从0打造音视频直播系统

[进入课程 >](#)



讲述：李超

时长 15:29 大小 14.19M



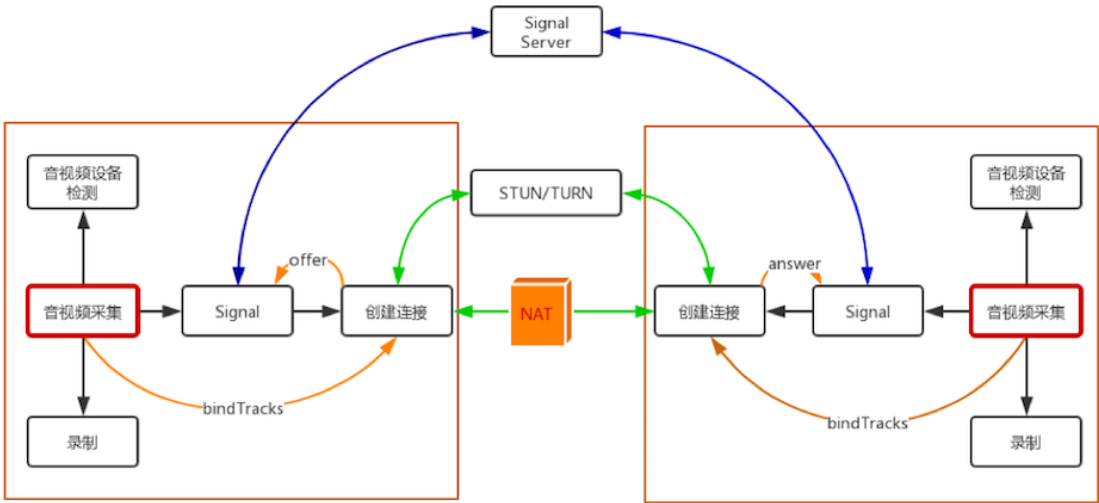
无论是做音视频会议，还是做远程教育，**共享桌面**都是一个必备功能。如果说在 PC 或 Mac 端写个共享桌面程序你不会有太多感受，但通过浏览器也可以共享桌面是不是觉得就有些神奇了呢？

WebRTC 的愿景就是要让这些看似神奇的事情，不知不觉地发生在我们身边。

你可以想象一下，假如浏览器有了共享桌面功能，这会使得浏览器有更广阔的应用空间，一个最直接的例子就是我们可以直接通过浏览器进行远程办公、远程协助等工作，而不用再下载共享桌面的应用了，这大大提高了我们的工作效率。

在 WebRTC 处理过程中的位置

在正式进行主题之前，我们还是来看看本文在整个 WebRTC 处理过程中的位置，如下图所示：



WebRTC 处理过程图

没错，它仍然属于音视频采集的范畴，但是这次采集的不是音视频数据而是桌面。不过这也没什么关系，**桌面也可以当作一种特殊的视频数据来看待。**

共享桌面的基本原理

共享桌面的基本原理其实非常简单，我们可以分“两头”来说明：

对于**共享者**，每秒钟抓取多次屏幕（可以是 3 次、5 次等），每次抓取的屏幕都与上一次抓取的屏幕做比较，取它们的差值，然后对差值进行压缩；如果是第一次抓屏或切幕的情况，即本次抓取的屏幕与上一次抓取屏幕的变化率超过 80% 时，就做全屏的帧内压缩，其过程与 JPEG 图像压缩类似（有兴趣的可以自行学习）。最后再将压缩后的数据通过传输模块传送到观看端；数据到达观看端后，再进行解码，这样即可还原出整幅图片并显示出来。

对于**远程控制端**，当用户通过鼠标点击共享桌面的某个位置时，会首先计算出鼠标实际点击的位置，然后将其作为参数，通过信令发送给共享端。共享端收到信令后，会模拟本地

鼠标，即调用相关的 API，完成最终的操作。一般情况下，当操作完成后，共享端桌面也发生了一些变化，此时就又回到上面共享者的流程了，我就不再赘述了。

通过上面的描述，可以总结出共享桌面的处理过程为：**抓屏、压缩编码、传输、解码、显示、控制**这几步，你应该可以看出它与音视频的处理过程几乎是一模一样的。

对于共享桌面，很多人比较熟悉的可能是**RDP (Remote Desktop Protocol)** 协议，它是 Windows 系统下的共享桌面协议；还有一种更通用的远程桌面控制协议——**VNC (Virtual Network Console)**，它可以实现在不同的操作系统上共享远程桌面，像 TeamViewer、RealVNC 都是使用的该协议。

以上的远程桌面协议一般分为桌面数据处理与信令控制两部分。

桌面数据：包括了桌面的抓取 (采集)、编码 (压缩)、传输、解码和渲染。

信令控制：包括键盘事件、鼠标事件以及接收到这些事件消息后的相关处理等。

其实在 WebRTC 中也可以实现共享远程桌面的功能。但由于共享桌面与音视频处理的流程是类似的，且**WebRTC 的远程桌面**又不需要远程控制，所以其**处理过程使用了视频的方式，而非传统意义上的 RDP/VNC 等远程桌面协议**。

下面我们就按顺序来具体分析一下，在桌面数据处理的各个环节中，WebRTC 使用的方式与 RDP/VNC 等真正的远程桌面协议的异同点吧。

第一个环节，共享端桌面数据的采集。WebRTC 对于桌面的采集与 RDP/VNC 使用的技术是相同的，都是利用各平台所提供的相关 API 进行桌面的抓取。以 Windows 为例，可以使用下列 API 进行桌面的抓取。

BitBlt：XP 系统下经常使用，在 vista 之后，开启 DWM 模式后，速度极慢。

Hook：一种黑客技术，实现稍复杂。

DirectX：由于 DirectX 9/10/11 之间差别比较大，容易出现兼容问题。最新的 WebRTC 都是使用的这种方式

GetWindowDC：可以通过它来抓取窗口。

第二个环节，共享端桌面数据的编码。WebRTC 对桌面的编码使用的是视频编码技术，即 H264/VP8 等；但 RDP/VNC 则不一样，它们使用的是图像压缩技术。使用视频编码技术的好处是压缩率高，而坏处是在网络不好的情况下会有模糊等问题。

第三个环节，传输。编码后的桌面数据会通过流媒体传输协议发送到观看端。对于 WebRTC 来说，当网络有问题时，数据是可以丢失的。但对于 RDP/VNC 来说，桌面数据一定不能丢失。

第四个环节，观看端解码。WebRTC 对收到的桌面数据通过视频解码技术解码，而 RDP/VNC 使用的是图像解码技术（可对比第二个环节）。

第五个环节，观看端渲染。一般会通过 OpenGL/D3D 等 GPU 进行渲染，这个 WebRTC 与 RDP/VNC 都是类似的。

通过以上的讲解，相信你应该已经对共享远程桌面有一个基本的认知了，并且也知道在浏览器下使用 WebRTC 共享远程桌面，你只需要会使用浏览器提供的 API 即可。


因此本文的目标就是：你只需要学会和掌握浏览器提供的抓取屏幕的 API 就可以了。至于编码、传输、解码等相关知识，我会在后面的文章中陆续为你讲解。

如何共享桌面

学习完共享桌面相关的理论知识，接下来，就让我们实践起来，一起来学习如何通过浏览器来抓取桌面吧！

1. 抓取桌面

首先我们先来了解一下在浏览器下抓取桌面的 API 的基本格式：

 复制代码

```
1 var promise = navigator.mediaDevices.getDisplayMedia(constraints);
```

这个 API 你看着是不是似曾相识？没错，它与前面[《01 | 原来通过浏览器访问摄像头这么容易》](#)一文中介绍的采集视频的 API 基本上是一样的，我们可以再看一下采集视频的 API 的样子：

```
1 var promise = navigator.mediaDevices.getUserMedia(constraints);
```

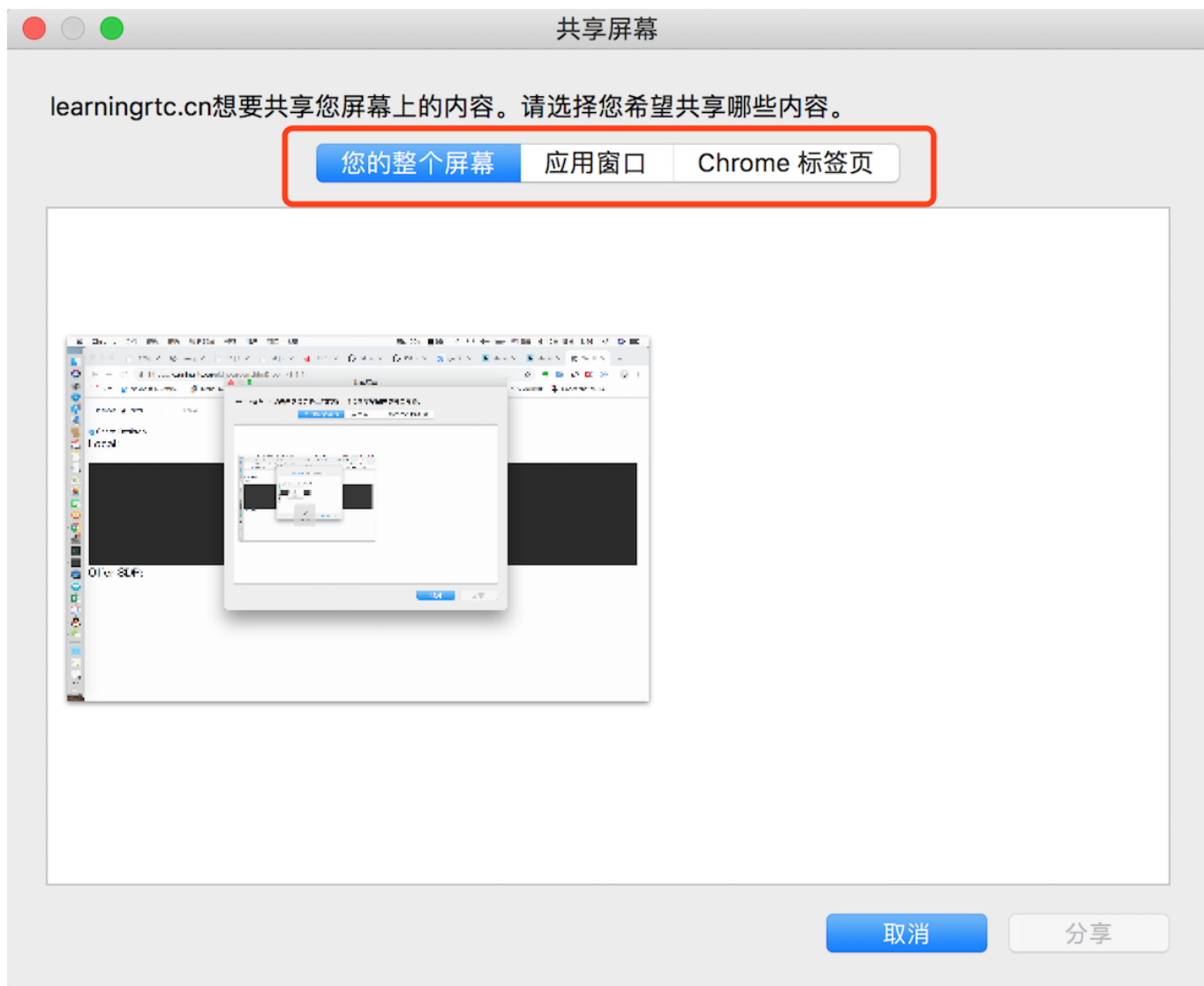
二者唯一的区别就是：一个是`getDisplayMedia`，另一个是`getUserMedia`。

这两个 API 都需要一个`constraints`参数来对采集的桌面 / 视频做一些限制。但需要注意的是，在采集视频时，参数`constraints`也是可以对音频做限制的，而在桌面采集的参数里却不能对音频进行限制了，也就是说，不能在采集桌面的同时采集音频。**这一点要特别注意。**

下面我们就来看一下**如何通过 `getDisplayMedia` API 来采集桌面**：

```
1 ...
2
3 // 得到桌面数据流
4 function getDeskStream(stream){
5     localStream = stream;
6 }
7
8 // 抓取桌面
9 function shareDesktop(){
10
11     // 只有在 PC 下才能抓取桌面
12     if(IsPC()){
13         // 开始捕获桌面数据
14         navigator.mediaDevices.getDisplayMedia({video: true})
15             .then(getDeskStream)
16             .catch(handleError);
17
18         return true;
19     }
20
21     return false;
22 }
23 }
24
25 ...
```

通过上面的方法，就可以获得桌面数据了，让我们来看一下效果图吧：




Chrome 浏览器共享桌面图

2. 桌面的展示

桌面采集后，就可以通过 HTML 中的`<video>`标签将采集到的桌面展示出来，具体代码如下所示。

首先，在 HTML 中增加下面的代码，其中`<video>`标签用于播放抓取的桌面内容：

 复制代码

```
1 ...  
2 <video autoplay playsinline id="deskVideo"></video>  
3 ...
```

下面的 JavaScript 则将桌面内容与`<video>`标签联接到一起：

```
1  ...
2  var deskVideo = document.querySelector("video/deskVideo");
3  ...
4  function getDeskStream(stream){
5      localStream = stream;
6      deskVideo.srcObject = stream;
7  }
8  ...
```

在 JavaScript 中调用 **getDisplayMedia** 方法抓取桌面数据，当桌面数据被抓到之后，会触发 **getDeskStream** 函数。我们再在该函数中将获取到的 **stream** 与 **video** 标签联系起来，这样当数据获取到时就从播放器里显示出来了。

3. 录制桌面

录制本地桌面与 [《04 | 可以把采集到的音视频数据录制下来吗？》](#) 一文中所讲的录制本地视频的过程是一样的。首先通过 **getDisplayMedia** 方法获取到本地桌面数据，然后将该流当作参数传给 **MediaRecorder** 对象，并实现 **ondataavailable** 事件，最终将音视频流录制下来。

具体代码如下所示，我们先看一下 HTML 部分：

```
1  <html>
2  ...
3  <body>
4      ...
5      <button id="record">Start Record</button>
6      ...
7  </body>
8  </html>
```

上面的 HTML 代码片段定义了一个开启录制的 **button**，当用户点击该 **button** 后，就触发下面的 JavaScript 代码：

```
1 ...
2
3 var buffer;
4
5 ...
6
7 function handleDataAvailable(e){
8     if(e && e.data && e.data.size > 0){
9         buffer.push(e.data);
10    }
11 }
12
13 function startRecord(){
14     // 定义一个数组，用于缓存桌面数据，最终将数据存储到文件中
15     buffer = [];
16
17     var options = {
18         mimeType: 'video/webm;codecs=vp8'
19     }
20
21     if(!MediaRecorder.isTypeSupported(options.mimeType)){
22         console.error(`${options.mimeType} is not supported!`);
23         return;
24     }
25
26     try{
27         // 创建录制对象，用于将桌面数据录制下来
28         mediaRecorder = new MediaRecorder(localStream, options);
29     }catch(e){
30         console.error('Failed to create MediaRecorder:', e);
31         return;
32     }
33
34     // 当捕获到桌面数据后，该事件触发
35     mediaRecorder.ondataavailable = handleDataAvailable;
36     mediaRecorder.start(10);
37 }
38
39 ...
```

当用户点击**Record**按钮的时候，就会调用**startRecord**函数。在该函数中首先判断浏览器是否支持指定的多媒体格式，如 webm。如果支持的话，再创建**MediaRecorder**对象，将桌面流录制成指定的媒体格式文件。

当从 localStream 获取到数据后，会触发**ondataavailable**事件。也就是会调用 handleDataAvailable 方法，最终将数据存放到 Blob 中。

至于将 Blob 保存成文件就比较容易了，我们在前面的文章[《04 | 可以把采集到的音视频数据录制下来吗？》](#)中都有讲解过，所以这里就不再赘述了！

小结

本文我向你讲解了如何通过浏览器提供的 API 来抓取桌面，并将它显示出来，以及如何通过前面所讲的**MediaRecorder**对象将桌面录制下来。

其实，真正的商用客户端录制并不是录制音视频流，而是录制桌面流。这样即使是多人互动的场景，在有多路视频流的情况下，录制桌面的同时就将桌面上显示的所有视频一起录制下来了，这样问题是不是一下就简单了？

比较遗憾的是，关于我们上述录制桌面的 API，目前很多浏览器支持得还不够好，只有 Chrome 浏览器相对比较完善。不过现在 WebRTC 1.0 规范已经出来了，相信在不久的将来，各浏览器都会实现这个 API 的。

思考时间

为什么使用视频的编码方式就容易出现桌面模糊的现象呢？有什么办法可以解决该问题吗？

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。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 04 | 可以把采集到的音视频数据录制下来吗？

下一篇 06 | WebRTC中的RTP及RTCP详解

精选留言 (10)

写留言

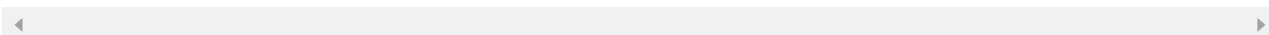


tommy_zhang

2019-07-29

老师，弹出的共享屏幕界面上内容能自定义吗？比如上面的文字"xxx想要共享共享您屏幕上的内容"

作者回复: 不能！



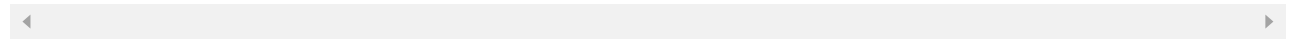
LongXiaJun

2019-07-26

打卡

展开 ▾

作者回复: 打卡！打卡！



K

2019-07-25

//第二部分

```
function playbackVideo(){
  blob = new Blob(buffer, {type:'video/webm'});
  rePlayVideo.src = window.URL.createObjectURL(blob);...
```

展开 ▾



K

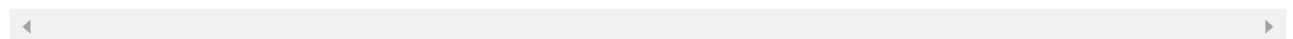
2019-07-25

//第一部分代码

```
let videoTypes = "video/webm\;codecs=vp8";
let userMediaSetting = { video: true };
let playVideo = document.querySelector('video#play');...
```

展开 ▾

作者回复: 赞！



恋着歌

2019-07-25

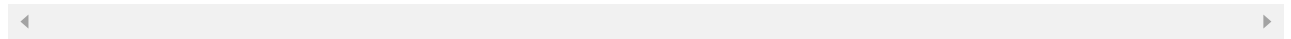
如果是解决网络引起的模糊，那么可能就要牺牲实时性，提高延迟，就像我们看视频时卡顿要缓冲一下。

具体的解决方法是：

1, 解决网络问题 🤖...

展开 ▾

作者回复: 赞!



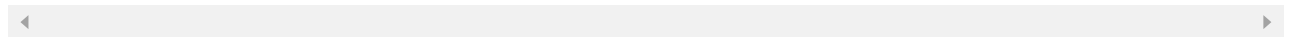
Jason

2019-07-25

思考题：我猜一下，wenrtc底层主要使用udp传输，网络不好的情况下，会有丢包。所以会有模糊的现象发生。

展开 ▾

作者回复: 如果是自适应码率，当发生丢包时，编码器会降低码率，当分辨率不变，码率变低时，就是模糊哈！

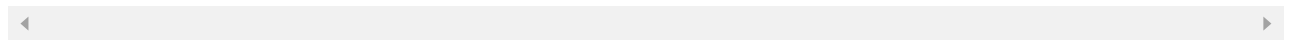


慢慢

2019-07-25

使用的chrome提示不支持navigator.mediaDevices.getDisplayMedia API，请问该如何解决？

作者回复: 是使用的https吗？



xiao豪

2019-07-25

简单demo

```
var deskVideo = document.querySelector("video#deskVideo");
function getDeskStream(stream){
    deskVideo.srcObject = stream;...
```

展开 ▾

作者回复: 赞！



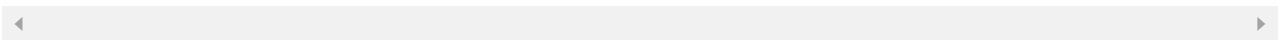
许童童

2019-07-25

老师能否给一个能够完整跑起来的Demo。

展开 ∨

作者回复: 这个主题结束之后，会有一个完整的 demo



流浪剑客

2019-07-25

看完打卡

展开 ∨

作者回复: 打卡！ 打卡！

