

36 | 如何使用 flv.js 播放 FLV 多媒体文件呢？

2019-10-05 李超

从0打造音视频直播系统

[进入课程 >](#)



讲述：李超

时长 11:53 大小 13.62M



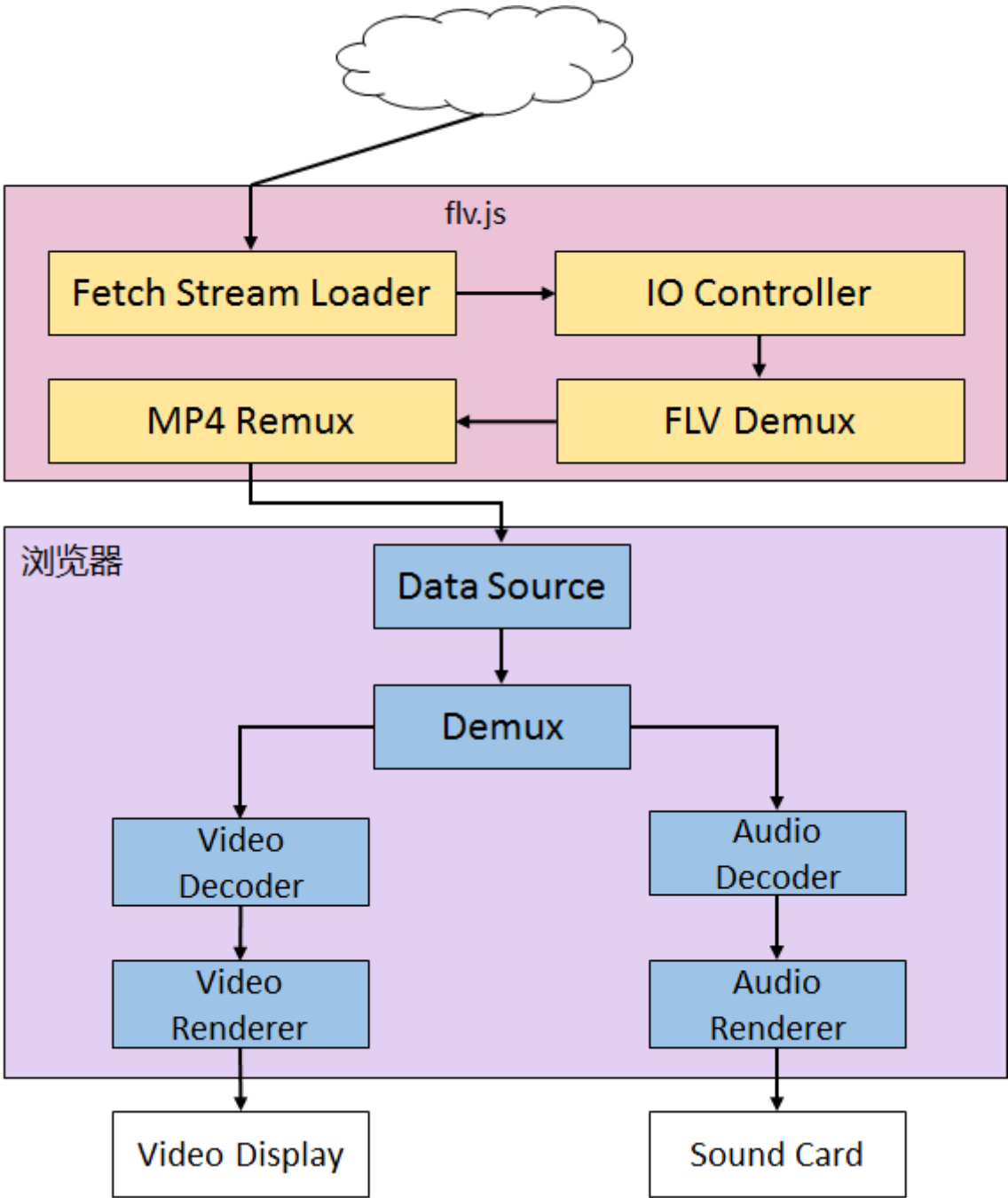
flv.js 是由 bilibili 公司开源的项目。它可以解析 FLV 文件，从中取出音视频数据并转成 BMFF 片段（一种 MP4 格式），然后交给 HTML5 的<video>标签进行播放。通过这种方式，使得浏览器在不借助 Flash 的情况下也可以播放 FLV 文件了。

目前，各大浏览器厂商默认都是禁止使用 Flash 插件的。之前常见的 Flash 直播方案，到现在已经遇到极大的挑战。因为它需要用户在浏览器上主动开启 Flash 选项之后才可以正常使用，这种用户体验是非常糟糕的，而 flv.js 的出现则彻底解决了这个问题。

flv.js 是由 JavaScript 语言开发的，该播放器的最大优势是，即使不安装 Flash 插件也可以在浏览器上播放 FLV 文件。虽说 Adobe 公司早已不再为 Flash 提供支持了，但 FLV 多媒体文件格式不会就此而消亡。因此，在没有 Flash 的时代里，能实现在浏览器上播放 FLV 文件就是 flv.js 项目的最大意义。

flv.js 基本原理

flv.js 的工作原理非常简单，它首先将 FLV 文件转成 ISO BMFF（MP4 片段）片段，然后通过浏览器的 Media Source Extensions 将 MP4 片段播放出来。具体的处理过程如下图所示：



flv.js 架构图

从上图我们可以看出，flv.js 播放器首先通过 Fetch Stream Loader 模块从云端获取 FLV 数据；之后由 IO Controller 模块控制数据的加载；数据加载好后，调用 FLV Demux 将 FLV 文件进行解封装，得到音视频数据；最后，将音视频数据交由 MP4 Remux 模块，重新对音视频数据封装成 MP4 格式。

将封装好的 MP4 片段交由浏览器的 Media Source Extensions 处理后，最终我们就可以看到视频并听到声音了。所以总体来说，flv.js 最主要的工作是做了媒体格式的转封装工作，具体的播放工作则是由浏览器来完成的。下面我们就对架构图中的每个模块分别做一下说明。

首先我们来看一下 flv.js 播放器，它包括以下四部分：

Fetch Stream Loader，指通过 URL 从互联网获取 HTTP-FLV 媒体流。其主要工作就是通过 HTTP 协议下载媒体数据，然后将下载后的数据交给 IO Controller。

IO Controller，一个控制模块，负责数据的加载、管理等工作。它会将接收到的数据传给 FLV Demux。

FLV Demux，主要的工作是去掉 FLV 文件头、TAG 头等，拿到 H264/AAC 的裸流。关于 FLV 文件格式，你可以参考 [《33 | FLV：适合录制的多媒体格式》](#) 一文。

MP4 Remux，它的工作是将 H264/AAC 裸流加上 MP4 头，采用的多媒体格式协议是 BMFF。它会将封装好格式的文件传给浏览器的 Data Source 对象。

经过以上四步，flv.js 就完成了自己的使命。

接下来的工作就是浏览器需要做的了，那我们再看一下浏览器各模块的主要作用。

Data Source，用来接收媒体数据的对象，收到媒体数据后传给 Demux 模块。

Demux，解封装模块，作用是去掉 MP4 头，获取 H264/AAC 裸流。

Video Decoder，视频解码模块，将压缩的视频数据解码，变成可显示的帧格式。

Audio Decoder，音频解码模块，将压缩的音频数据解码，变成可播放的格式。

Video Renderer，视频渲染模块，用于显示视频。

Audio Renderer，音频播放模块，用于播放音频。

Video Display，视频、图形显示设备的抽象对象。

Sound Card，声卡设备的抽象对象。

从上面的过程可以看出，flv.js 主要的工作就是进行了 FLV 格式到 MP4 格式的转换。之所以这样，是因为 flv.js 是通过 HTML5 的 `<video>` 标签播放视频，而此标签支持的是 MP4 格式。

关于 <video> 标签

<video> 标签是 HTML5 新支持的元素，用于在浏览器中播放音视频流。实际上，我们在介绍 WebRTC 知识的时候就已经对它做过一些介绍了。今天我们再来简单重温一下这部分知识。

<video> 标签的使用与其他 HTML5 标签的使用是一样的，我们这里就不多讲了，主要讲一下<video>标签支持的几个属性：

`autoplay`，如果设置该属性，则视频在就绪后就马上播放。

`src`，要播放的视频的 URL。

`srcObject`，用于播放实时音视频流，该属性与 `src` 互斥。

.....

<video>标签支持的属性比较多，你可以查看文末参考一节，那里将<video>标签的所有属性都做了详细说明。关于这些属性，我建议你写一个简单的页面分别测试一下，以加深你对它们的认知。


另外，对于该标签的测试有两点是需要你注意的：

媒体文件建议用 MP4 文件，你系统的本地文件就可以用来测试<video>标签。

在测试 `autoplay` 属性时候，如果发现没有效果，建议加上 `muted` 属性，浏览器保证静音状态是能 `autoplay` 的。

使用 flv.js

首先，我们需要将 `flv.js` 源码下载下来。主要有两种方式：一种是通过 `git clone` 命令从 GitHub 上拉取最新的代码；另一种是通过 NPM 命令从资源库中获取 `flv.js` 源码。这里我们采用的是第二种方式，具体命令如下：

 复制代码

```
1 npm install --save flv.js
```


源码下载下来后，我们还要将 flv.js 的依赖包下载下来，通过 gulp 工具将 flv.js 进行打包、压缩。具体步骤如下：

 复制代码

```
1 npm install          # 安装 flv.js 依赖的包
2 npm install -g gulp  # 安装 gulp 构建工具
3 gulp release         # 打包、压缩 工程 js 文件
```

其中，第 1 条命令的作用是下载 flv.js 的依赖包，第 2 条命令是安装 JavaScript 打包工具 gulp，第 3 条命令是使用 gulp 对 flv.js 进行构建。

需要注意的是，在执行 gulp release 命令时，有可能会遇到如下的错误：

 复制代码

```
1 gulp release[4228]: src\node_contextify.cc:633: Assertion `args[1]->IsString()' failed.
```

这时可以通过安装 Node.js 的 natives 模块来解决该问题，安装 natives 的命令如下：

 复制代码

```
1 npm install natives
```

通过上面的步骤，我们就可以构建出 flv.min.js 文件了。该文件就是我们在项目中需要引用的 flv.js 文件。

接下来，我们就展示一下该如何使用 flv.js。你可以在本地创建一个 HTML 文件，如 play_flv.html，其代码如下：

 复制代码

```
1 <!-- 引入 flv.js 库 -->
2 <script src="flv.min.js"></script>
3
4 <!-- 设置 video 标签 -->
5 <video id="flv_file" controls autoplay>
```



```
6   You Browser doesn't support video tag
7 </video>
8
9 <script>
10    // 通过 JavaScript 脚本创建 FLV Player
11    if (flvjs.isSupported()) {
12        var videoElement = document.getElementById('flv_file');
13        var flvPlayer = flvjs.createPlayer({
14            type: 'flv',
15            url: 'http://localhost:8000/live/test.flv'
16        });
17        flvPlayer.attachMediaElement(videoElement);
18        flvPlayer.load();
19        flvPlayer.play();
20    }
21 </script>
```

上面代码的逻辑非常简单，它主要做了三件事：第一是在 HTML 中引入了 flv.js 库；第二是设置了一个 `<video>` 标签，用于播放 FLV 文件；第三是一段 JavaScript 代码片段，用于创建 FLV Player 对象，并将这与上面定义的 `<video>` 绑定到一起，实现对 `<video>` 标签的控制。

flv.js 的实现架构以及暴露的 API 接口在 flv.js/docs 目录下面有一个简单的介绍，你可以参考一下。flv.js 暴露的接口不多，但由于没有暴露接口的任何文档，所以你只能直接查看源码去了解每个接口的具体作用与含义。幸运的是，flv.js 架构设计得非常合理，代码实现也非常优秀，所以对于 JavaScript 开发人员来说，查看 flv.js 源码不是太难的事儿。

小结

本文我们首先介绍了 flv.js 播放器的基本工作原理，通过该原理我们可以知道，它主要的工作就是将 FLV 文件格式转换成 MP4 文件格式。

紧接着我们通过一个例子向你介绍了如何使用 flv.js。flv.js 的使用还是比较简单的，你只要按文中的步骤就可以很快写出一个利用 flv.js 播放 FLV 文件的 Demo 出来。

另外需要注意的是，我们本文对 flv.js 的测试需要用于流媒体服务器，而流媒体服务器的搭建我们在前两篇文件中已经向你做过介绍了。你既可以使用 CDN 搭建流媒体服务器，也可以使用 Nginx 在自己的本机搭建流媒体服务器。其具体过程这里我就不再重复了。

至于推流工具，前面的文章中我们也详细介绍过，你可以任选 FFmpeg 或 OBS 作为你的推流工作，具体的操作方式请参考前面的文章。

总体来说，flv.js 是一个非常优秀的 Web 开源播放器，它的代码写得非常漂亮，可读性也高。但它的文档相对匮乏，所以我们在开发过程中如果遇到一些问题，建议直接查看其源码。

从本专栏的第三个模块开始到现在我们已经介绍了 FLV 文件格式、HLS 协议、流媒体服务器，今天我们又学习了 flv.js 播放器，至此我们就可以用 CDN 做流媒体服务转发、用 OBS 进行推流、用 flv.js 播放，构建出自己的直播系统了。

思考时间

今天留给你的思考题是：flv.js 中用到的 BMFF 格式是什么样的呢？

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。

参考

<video> 标签属性

属性	作用
autoplay	如果设置该属性，则视频在就绪后马上播放。
controls	如果设置该属性，则向用户显示控件，比如播放按钮。不设置该选项，浏览器窗口不显示播放器相关控件。
height	设置视频播放器的高度像素值。
width	设置视频播放器的宽度像素值。
loop	如果设置该属性，则视频文件播放完后从头再开始播放。
preload	如果出现该属性，则视频在页面加载时进行加载，并预备播放。如果使用“autoplay”，则忽略该属性。
srcObject	用于播放实时音视频流，该属性与 src 互斥。
src	要播放的视频的 URL。
muted	表示静音，就是播放的时候没有声音。



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。

精选留言

 写留言

由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。