

37 | 如何使用 video.js 播放多媒体文件？

2019-10-08 李超

从0打造音视频直播系统

[进入课程 >](#)



讲述：李超

时长 14:48 大小 13.57M



在上一篇文章中，我们介绍了 flv.js 播放器。那今天我们再来介绍另一款非常有名的 JavaScript 播放器——video.js。

我们首先来比较一下这两款播放器，看看它们之间有什么不同？在我看来，flv.js 更聚焦在多媒体格式方面，其主要的是将 FLV 格式转换为 MP4 格式，而对于播放器的音量控制、进度条、菜单等 UI 展示部分没有做特别的处理。而 video.js 对音量控制、进度条、菜单等 UI 相关逻辑做了统一处理，对媒体播放部分设计了一个插件框架，可以集成不同媒体格式的播放器进去。所以**相比较而言，video.js 更像是一款完整的播放器。**

video.js 对大多数的浏览器做了兼容。它设计了自己的播放器 UI，接管了浏览器默认的<video>标签，提供了统一的 HTML5/CSS 皮肤。因此，通过 video.js 实现的播放器，在大多数浏览器上运行时都是统一的风格和操作样式，这极大地提高了我们的开发效率。

除了上面介绍的特点外，video.js 还有以下优势：

开源、免费的。不管你是学习、研究，还是产品应用，video.js 都是不错的选择。

轻量。浏览器 UI 的展现全部是通过 HTML5/CSS 完成，没有图片的依赖。

完善的 API 接口文档，让你容易理解和使用。

统一的 UI 设计，不管在哪个浏览器，你都看不出任何差异。

皮肤可以任意更换，很灵活。

开放灵活的插件式设计，让你可以集成各种媒体格式的播放器。

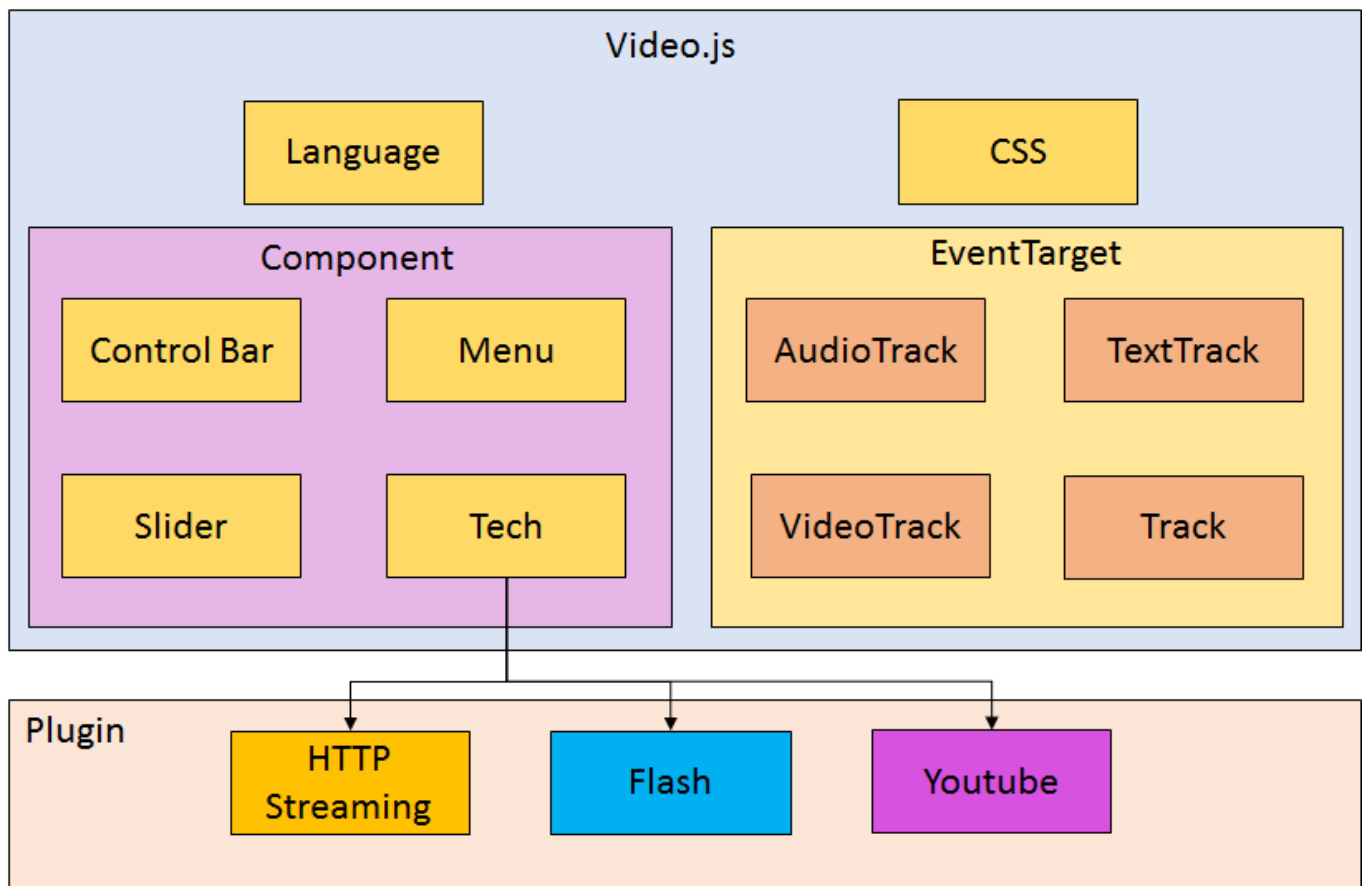
支持多种文字语言，如中文、英文等。

既然有这么多优势，那接下来我们就来详细讲解一下 video.js 的相关内容吧。

video.js 的架构

HTML5 为媒体播放新增了很多新的元素，比如<audio>、<video>、<source>等，这些内置标签基本上可以满足我们日常的需求。而 video.js 把这些组件统统都实现了一遍，其主要目的是为了适配不同浏览器的差异，为各浏览器提供统一的 UI 展示和个性化定制。

接下来，我们来看看 video.js 都包含了哪些主要组件，如下图所示：



video.js 架构图

通过该图可以看到，video.js 主要包括**对多种文字语言支持、CSS 样式定制、控件部分、媒体内嵌元素部分和外部插件**五大部分。下面我们来简要介绍下这每一部分的相关信息。

第一部分是 Language。它在 video.js/language 目录下面，支持多种文字语言切换。

第二部分是 CSS 样式。video.js 的 CSS 样式是可以更换的，支持个性化定制。

第三部分是 Component。Component 是 video.js 中 UI 控件的抽象类，在 Component 中封装了 HTML 元素。Control Bar、Menu、Slider、Tech 都是继承自 Component，叫做子组件，子组件也可以有子组件，这样就形成了一棵树。这样设计的目的就是**将播放器相关控件模拟成 DOM 树模型**。下面是子组件的功能：

Control Bar，播放器的控制模块。调节音量、播放进度拖动等都由该模块完成。

Menu，播放器右键菜单的实现。

Slider，滚动条控件。可以是垂直滚动条，也可以是水平滚动条。音量滚动条、进度滚动条都是它的子类。

Tech, 是 Technique 的缩写, 表示采用的播放器技术。其实它就是为播放器插件提供抽象接口。video.js 默认使用的是 HTML5 播放器。

第四部分是 EventTarget。HTML5 除了提供了 `<audio>`、`<video>`、`<source>` 这些可见元素, 还包括了 Media Source 的概念。像 AudioTrack、VideoTrack、TextTrack、Track 都继承自 Media Source, video.js 把它们也都进行了抽象, 这些对象都统一实现了 EventTarget 接口。这几个 Track 的作用如下:

AudioTrack, 音频轨, 也是音频媒体源。

VideoTrack, 视频轨, 也是视频媒体源。

TextTrack, 文字轨, 也是文字媒体源。比如给视频添加字幕就可以使用它, 对应于 `<track>` 标签。

Track, 媒体源的公共抽象。你一定要将它与 `<track>` 区分开来, 这两个不是同一回事。这一点需要你注意一下。

第五部分是插件。video.js 支持播放器插件开发, 目前已经有很多插件实现了。在上图中我们只列举了 3 个插件:

HTTP Streaming, 可以播放 HLS 协议、DASH 协议的媒体流。

Flash, 用于播放 RTMP 媒体流。但目前各大浏览器默认都是禁止使用 Flash 的。经测试, Chrome 浏览器和 IE 新版本浏览器都已不能使用 Flash 播放 RTMP 流了。


YouTube, 是企业定制插件。

video.js 安装部署

[video.js 的文档](#) 非常详细, 包括了安装、部署、API 说明、FAQ 等, 只要按照文档手册的步骤进行安装部署就可以了。使用 video.js 主要有以下三种方式。


第一种方式, 通过源码安装部署。

首先, 从 GitHub 下载源码。命令如下:

 复制代码


```
1 git clone https://github.com/videojs/video.js.git
```

然后，安装 video.js 依赖的文件。命令如下：

 复制代码

```
1 cd video.js
2 npm install
```


最后，构建 video.js。运行命令如下：

 复制代码

```
1 npm run-script build
```


通过以上三步，在 video.js/dist 目录下面就会生成 video.min.js、video.min.css、语言、字体等相关文件，你只需要将它们引入到你的 JavaScript 工程中即可。

第二种方式，从 npm 仓库获取。

 复制代码

```
1 npm install video.js
```

第三种方式，通过 CDN 直接下载官方构建好的文件。


 复制代码

```
1 <link href="https://unpkg.com/video.js/dist/video-js.min.css" rel="stylesheet">
2 <script src="https://unpkg.com/video.js/dist/video.min.js"></script>
```

总之，你可以根据自己的喜好，选择其中一种方式使用 video.js。这里需要说明一下，本文后续的例子都是采用的第三种方式。

video.js 播放 MP4

接下来我们就来实战一下，使用 video.js 播放一个本地 MP4 文件，具体代码如下：

 复制代码

```
1 // 引入 video.js 库
2 <link href="https://unpkg.com/video.js/dist/video-js.min.css" rel="stylesheet">
3 <script src="https://unpkg.com/video.js/dist/video.min.js"></script>
4
5 // 使用 video 标签描述 MP4 文件
6 <video
7     id="local_mp4"
8     class="video-js"
9     controls
10    preload="auto"
11    poster="//vjs.zencdn.net/v/oceans.png"
12    data-setup='{ }'>
13    <source src="d:/test.mp4" type="video/mp4"></source>
14 </video>
```

下面我们来对上面的代码片段做下简要说明：

`<link>` 标签，从 CDN 获取 video.js 的 CSS 文件。

`<script>` 标签，从 CDN 获取 video.js 文件。注意，这种方式获取的是最新发布版本。

`<video>` 标签，对于 video.js 来讲，需要设置 id 属性。

data-setup 属性，是 video.js 特有的，此标签用于播放器的自动加载。这里我们的代码中传入的是一个空的 json 对象。你也可以参考[官网文档选项参数](#)，按照需求传入配置参数对象。

`<source>` 标签，指定播放 URL 和媒体类型。我们的代码中指定的是本地磁盘 MP4 文件，test.mp4 是预先准备好的文件，类型是 video/mp4。

我们可以预先准备好一个 MP4 文件，然后再将上面的代码片段拷贝在一个 HTML 文件中，比如play_local_mp4.html，最后在浏览器中打开，就可以看到播放画面了。

本地流媒体服务器搭建


接下来，我们再来介绍一下如何使用 video.js 播放 HLS。对于推流工具我们可以使用 FFmpeg 或 OBS 进行推流，对于这些工具体的使用，我在前面的文章中都有向你做过详细介绍，所以这里就不再赘述了。

对于流媒体服务器，你可以使用 Nginx 在你的本机上搭建流媒体服务器，也可以通过前面文章所介绍的 CDN 网络做为流媒体服务器。对于实验来说，它们都是可行的方案。

对于流媒体服务器的搭建的过程，前面我们已经通过两篇文章做了详细介绍。这里同样也不再赘述了。

video.js 播放 HLS

在使用 video.js 播放 HLS 媒体流之前，我们需要先创建一个 HTML5 文件，如 play_hls.html，在 HTML5 文件中的内容如下：

 复制代码

```
1 // 引入 video.js 库
2 <link href="https://unpkg.com/video.js/dist/video-js.min.css" rel="stylesheet">
3 <script src="https://unpkg.com/video.js/dist/video.min.js"></script>
4
5 // 设置 video 标签
6 <video id="my-hls-player" class="video-js">
7   <source src="http://localhost:8000/live/test/index.m3u8" type="application/x-mpegURL"
8 </video>
9
10 <script>
11 // 创建 HLS 播放器实例
12 var player = videojs('my-hls-player', {
13   controls:true,
14   autoplay:true,
15   preload:'auto'
16 });
17
18 player.ready(function(){
19   console.log('my-hls-player ready...');
20 });
21 </script>
```

video.js 的官方文档对如何使用 video.js 描述得非常详细，你可以自行查阅使用手册。这里我们只简要分析一下代码片段中用到的接口：

从官方指定的 CDN 获取 video.min.js 和 video-js.min.css 文件。需要注意的是，从 video.js 7 开始，HLS 插件默认包含在 video.js 里了。此时，我们就不需要再单独引入 HLS 插件了。

`<video>` 标签中需要指定 ID，这个 ID 是可以随便起的，我们这里设置的是 my-hls-player。 `<video>` 标签的 CSS 样式采用的是官方提供的默认样式 video-js。当然，你也可以定制自己喜欢的样式。

`<source>` 标签中的 src 属性，指定了 m3u8 播放地址。我们这里设置的地址是 <http://localhost:8000/live/test/index.m3u8>。需要注意的是，type 属性必须是 application/x-mpegURL。

在代码的最后实现了 player 实例的 ready 回调函数，这样当播放器加载完成后触发 ready 事件时，player 的 ready 函数就会被调用。

现在我们就来测试一下吧。通过 FFmpeg 工具向地址 rtmp://IP/live/test 推流，而 HLS 协议媒体流的播放地址为 <http://IP:port/live/test/index.m3u8>，再通过浏览器打开 play_hls.html 页面，就可以看到 HLS 协议的画面了。

小结

本文我们首先对 video.js 的架构做了简要介绍，了解了 video.js 的几个重要组成模块，以及这些模块的主要作用。紧接着，我们展示了如何通过 video.js 播放本地 MP4 文件。最后，我们还介绍了如何通过 video.js 播放 HLS 直播流。

可以说 video.js 是目前在浏览器上最好用、最著名的开源流媒体播放器。它的功能非常强大，既可以处理多种多媒体格式，如 MP4、FLV 等；又可以支持多种传输协议，如 HLS、RTMP。因此，它是播放音视频直播媒体流必不可少的播放工具。

在本文中，我们为了简单，只在本地建了一个使用 video.js 的 Demo。但在真实的直播系统中，我们应该实现一个直播客户端，在直播客户端中引入 video.js 来播放直播流。此外，该直播客户端还应该通过 WWW 服务发布。当用户想进入房间观看节目时，首先从 WWW 服务上下载客户端，在浏览器将其加载后，通过信令服务获取到直播地址，并最终拉取直播流进行展示。这样就将 video.js 播放器与我们前面讲的万人直播系统组合到一起，最终实现可商用的直播系统了。

思考时间

今天留给你的思考题是：video.js 是如何通过 data-setup 属性完成自动加载的呢？

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。



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 36 | 如何使用 flv.js 播放 FLV 多媒体文件呢？

下一篇 38 | 实战推演：带你实现一个支持万人同时在线的直播系统

精选留言 (1)

写留言



刘丹

2019-10-08

请问flv.js、video.js是否都支持2倍速、0.5倍速播放音视频？

作者回复: flv.js 和 video.js 最终使用的还是HTML5 的video标签。在 H5的 <video>标签中有 playbackRate 属性，你通过它就可以实现倍速播放了。

