第 **9** 章 音频和视频  
学习要点：  
1.音频和视频概述  
2.video 视频元素  
3.audio 音频元素  
  
  
本章主要探讨 HTML5 中音频和视频元素， 通过这两个原生的媒体元素向 HTML 页面中  
嵌入音频和视频。  
一．音频和视频概述  
首先，我们要理解两个概念：容器（container）和编解码器(codec)。

**1.**视频容器  
音频文件或视频文件，都只是一个容器文件。视频文件包含了音频轨道、视频轨道和其  
他一些元数据。

视频播放时，音频轨道和视频轨道是绑定在一起的。元数据包含了视频的封面、 标题、 子标题、 字幕等相关信息。 主流视频容器支持的格式为： .avi、 .flv、 .mp4、 .mkv、  
.ogg、.webm。

2**.**编解码器  
音频和视频编码/解码是一组算法，用来对一段特定音频或视频进行解码和编码，以便  
音频和视频能够播放。原始的媒体文件体积非常巨大，如果不对其进行编码，那么数据量是  
非常惊人的，在互联网上传播则要耗费无法忍受的时间；如果不对其进行解码，就无法将编码后的数据重组为原始的媒体数据。主流的音频编解码器：AAC、MPEG-3、Ogg Voribs，  
视频编解码器：H.264、VP8、Ogg Theora。

**3.**浏览器支持情况  
起初，HTML5 规范本来打算指定编解码器，但实施起来极为困难。部分厂商已有自己的  
标准，不愿实现标准；而有一些编解码器受专利保护，需要支付昂贵费用。最终放弃了统一  
规范的要求，导致各个浏览器实现自己的标准。  
容器格式 视频编解码 音频编解码 IE9+ Firefox5+ Chrome13+  
WebM VP8 Vorbis × √ √  
OGG Theora Vorbis × √ √  
MPEG-4 H.264 AAC √ × 疑问？  
除了上面三款浏览器，还有 Safari5+支持 MPEG-4,Opera11 支持 WebM 和 OGG，通过  
这组数据，只要备好 MP4 和 OGG 格式的即可，但对于新的高清标准 WebM，当然是非常必要的。因为 WebM 不但清晰度高，而且免费，不受限制许可的使用源码和专利权。  
目前 Chrome 浏览器， 为了推广 WebM 格式的视频。 声称未来将放弃 H.264 编码的视频，所以有可能在以后的版本中无法播放 MP4 的视频。当然，目前演示的版本还是支持的。

二．**video** 视频元素  
以往的视频播放，需要借助 Flash 插件才可以实现。但 Flash 插件的不稳定性经常让浏览器导致崩溃， 因此很多浏览器或系统厂商开始抛弃它。 而取代它的正是 HTML5 的 video元素。  
<video>元素的属性  
属性名称 说明  
src 视频资源的 URL  
width 视频宽度  
height 视频高度  
autoplay 设置后，表示立刻开始播放视频  
preload 设置后，表示预先载入视频  
controls 设置后，表示显示播放控件  
loop 设置后，表示反复播放视频  
muted 设置后，表示视频处于静音状态  
poster 指定视频数据载入时显示的图片

**1.**嵌入一个 **WebM** 视频  
<video src="test.webm" width="800" height="600"></video>  
解释：<video>插入一个视频，主流的视频为.webm，.mp4，.ogg 等。src 表示资源  
URL；width 表示宽度；height 表示高度。

**2.**附加一些属性  
<video src="test.webm" width="800" height="600" autoplay controls loop  
muted></video>  
解释：autoplay 表示自动开始播放；controls 表示显示播放控件；loop 表示循环播放；muted 表示静音。

**3.**预加载设置  
<video src="http://li.cc/test.webm" width="800" height="600" controls  
preload="none"></video>  
解释：preload 属性有三个值：none 表示播放器什么都不加载；metadata 表示播放  
之前只能加载元数据（宽高、第一帧画面等信息）；auto 表示请求浏览器尽快下载整个视频。

**4.**使用预览图  
<video src="http://li.cc/test.webm" width="800" height="600" controls  
poster="img.png"></video>  
解释：poster 属性表示在视频的第一帧，做一张预览图。  
**5.**兼容多个浏览器  
<video width="800" height="600" controls poster="img.png">  
<source src="test.webm">  
<source src="test.mp4">  
<source src="test.ogg">  
<object>这里引入 flash 播放器实现 IE9 以下，但没必要了</object>  
</video>  
解释：通过<source>元素引入多种格式的视频，让更多的浏览器保持兼容。

二．**audio** 音频元素  
和 video 元素一样，audio 元素用于嵌入音频内容，而音频元素的属性和视频元素类  
似。音频的支持度和视频类似，使用<source>元素引入多种格式兼容即可。主流的音频格  
式有：.mp3，.m4a，.ogg，.wav。  
属性名称 说明  
src 视频资源的 URL  
autoplay 设置后，表示立刻开始播放视频  
preload 设置后，表示预先载入视频  
controls 设置后，表示显示播放控件

**1.**嵌入一个音频  
<audio src="test.mp3" controls autoplay></audio>  
解释：和嵌入视频一个道理。

**2.**兼容多个浏览器  
<audio controls>  
<source src="test.mp3">  
<source src="test.m4a">  
<source src="test.wav">  
</audio>

video元素

 --在HTML5中专门用来播放网络上的视频或者电影。

audio元素

 --在HTML5中专门用来播放网络上的音频。

 使用video和audio元素进行播放时就不在需要使用其他的插件了，只要我们的浏览器支持HTML5就可以了！

浏览器的支持：

 Safari3以上、Firefox4以上、0pera10以上、chrome3.0以上版本都对audio元素和video元素支持！

使用方法

audio元素只需要给他指定一个src属性：

<audio src="MP3.mp3" controls="controls"></audio>

对于不支持的浏览器我们可以在这对元素之间加入提示语句来代替

<audio src="MP3.mp3" controls="controls">您的浏览器不支持Audio元素</audio>

video元素要设定好长宽和src属性就可以了：

<video width="750" height="400" src="time.mp4"></video>

同样对于不支持video的浏览器可以在中间加入替换文字：

<video width="750" height="400" src="time.mp4">您的浏览器不支持video元素</video>

source元素指定多个播放格式与编码：

source元素可以为同一个媒体数据指定多个播放格式与编码方式，以确保浏览器可以从中选择一种自己支持的播放格式进行播放。选择顺序自上而下，直到选择到所支持的格式为止。

使用方法：

<video>

 <source src="video.m4v" type="video/mp4" />

 <source src="video.webm" type="video/webm" />

 <source src="video.ogv" type="video/ogg" />

 <source src="video.mp4" />

</video>

各种设备对编码格式的支持情况：

webm(.webm)格式的视频 火狐4.0+、chrome6.0+、opera10.6+

mp4(.m4v)格式的视频 IE9.0+ 、Safari3.1+ 、iso5.0 、Android4.0+

ogg(.ogv)格式的视频 火狐3.5+、chrome3.0+、opera10.5+

mp4(.mp4)格式的视频 IE9.0+ 、Safari3.1+ 、iso3.0 、Android2.3+

audio元素和video元素的常用属性

src属性：

在这个属性里面指定媒体数据的URL地址。

controls属性：

指定是否为视频或者音频数据添加浏览器自带的播放控制条，控制条中有播放按钮、暂停等按钮。

使用方法：

<video src="video.mp4" controls="controls"></video>

width和height属性（video独有）：

指定视频的宽度与高度。

使用方法：

<video src="video.mp4" width="650" height="450"></video>

autoplay属性：

这个属性指定是否当我们网页加载完成之后就开始自动播放。

preload属性：

这个属性指定是否对数据进行预加载，如果是的话，浏览器会将视频数据或者音频数据进行缓冲，这样做可以加快播放的速度。

  preload属性的三个值：

 none 表示不进行预加载。

 metadata 表示只预加载媒体的元数据。

 auto(默认值) 表示预加载全部的视频或者音频。

使用方法：

<video src="video.mp4" preload="auto" ></video>

poster属性（video独有）：

当视频不可以播放的时候，使用poster元素向用户展示一张图片代替视频。

使用方法：

<video src="video.mp4" poster="video.jpg"></video>

loop属性：

指定是否循环播放视频或者音频数据。

使用方法：

<video src="video.mp4" autoplay="auto" loop="loop"></video>

error属性：

读取过程中一旦发生错误，返回一个Media Error对象，这个对象的code返回对应的错误状态，默认情况下video和audio的error属性都是null。

 4种错误状态，返回一个数字值，它表示音频/视频的错误状态：

 1 = MEDIA\_ERR\_ABORTED - 取回过程被用户中止

 2 = MEDIA\_ERR\_NETWORK - 当下载时发生错误

 3 = MEDIA\_ERR\_DECODE - 当解码时发生错误

 4 = MEDIA\_ERR\_SRC\_NOT\_SUPPORTED - 媒体不可用或者不支持音频/视频

读取错误状态示例

<video id="video" src="video.mp4"></video>

<script type="text/javascript">

 var video = document.getElementById('video');

 video.addEventListener("error",function(){

 var error = video.error;

 switch (error.code){

    case 1:

        alert('取回过程被用户中止。');

        break;

    case 2:

        alert('当下载时发生错误。');

        break;

    case 3:

        alert('当解码时发生错误。');

        break;

    case 4:

        alert('媒体不可用或者不支持音频/视频。');

        break;

 }

 },false);

</script>

networkState属性：

networkState 属性返回音频/视频的当前网络状态（activity）

 4种错误状态，表示音频/视频元素的当前网络状态：

 0 = NETWORK\_EMPTY - 音频/视频尚未初始化

 1 = NETWORK\_IDLE - 音频/视频是活动的且已选取资源，但并未使用网络

 2 = NETWORK\_LOADING - 浏览器正在下载数据

 3 = NETWORK\_NO\_SOURCE - 未找到音频/视频来源

networkState属性：

networkState 属性返回音频/视频的当前网络状态（activity）

video元素和audio元素的4种方法

play方法：

使用play方法来播放媒体，自动将元素的paused属性的值变成false。

pause方法：

使用pause方法来暂停播放，自动将元素的paused属性的值变成true。

load方法：

使用load方法来重新载入媒体进行播放，自动将元素的playbackRate属性的值变成defaultPlaybackRate属性的值，自动把error的值变成null。

canPlayType方法：

使用canPlayType方法来测试浏览器是否支持指定的媒体类型。

使用方法如下：

 var support = videoElment.canPlayType(type);

canPlayType() 方法可返回下列值之一：

  "probably" - 浏览器最可能支持该音频/视频类型

  "maybe"   - 浏览器也许支持该音频/视频类型

  ""           - （空字符串）浏览器不支持该音频/视频类型