《Web 前端开发》课程理论教学部分

第九讲：音频与视频

学时计划：2 学时 理论，1 学时 实验

（实验四：在网页中使用表单和多媒体）

教学大纲：

1、从网络上的视频说起

2、video 和 audio 元素

3、video 和 audio 的属性

4、video 和 audio 的方法

5、video 和 audio 的事件

6、案例：拍个 DV 让大家看

7、讨论与思考

互联网的飞速发展和网络速度的改善，使得视频和音频已经成为了互联网内容的重要形式之一，并且视频和音频的网络传输，也极大地推动了互联网应用的发展。今天通过优酷、土豆、乐视、CNTV 等网站观看电影、电视剧和综艺节目，成为人们使用互联网的主要应用之一。

视频和音频的广泛应用，使得在网络中使用视频和音频技术变得越加重要。本讲将讨论视频和音频的容器文件和编解码器，以及 HTML 5 对视频和音频方面的新技术，以帮助读者创建丰富多彩的多媒体互联网应用服务。

一、从网络上的视频说起

在 HTML 5 规范之前，如果希望在网页上播放视频、音频，通畅都需要借助第三方插件，比如 Flash。或者需要自主开发使用多媒体播放插件。但是无论哪种方式，都需要在浏览器或者计算机中安装插件或者播放器。对于视频和音频的支持，并不是由浏览器自身来提供支持的，这种做法复杂且麻烦，还容易产生兼容性和安全性的问题。

HTML 5 定义了一个在网页中嵌入视频和音频的标准方式，就是

使用<video>和<audio>元素。虽然目前此部分的工作还不够完善，但是已经取得了浏览器的广泛支持和认可。

1.1 视频容器

不论是视频文件还是音频文件，例如.avi、.flv 等，实质上都是一个容器文件。这个容器文件就如同.zip 文件一样。视频文件（视频容器）包含了音频轨道、视频轨道和其他一些元数据。当进行视频播放时，音频轨道和视频轨道是绑定在一起的。元数据部分包含了视频的封面图片、标题、子标题、字幕等信息。

视频容器有很多中，较为常见的如下所示：

（1）MPEG-4（.mp4）

（2）Flash Video（.flv）

（3）Ogg（.ogv）

（4）WebM（.webm）

（5）Audio Video Interleave（.avi）

（6）Matroska（.mkv）

1.2 音频和视频的编解码器

音频和视频的编码/解码器是一组算法，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。

当观看一个视频的时候，视频播放器（解码器）需要完成的工作有：

第一步：解析容器格式以找出可以使用的视频和音频轨道，并分析它们的存储结构，以便接下来的解码工作。

第二步：对视频流解码，并在屏幕上显示一幅幅的图像。

第三步：对音频流解码，同时给扬声器传输声音信号。

编解码器包括有损和无损两种。无损视频文件一般太大了，在网页中没有优势，所以在网络上传送的视频采用的都是有损编解码器。有损视频编解码器中，信息在编码过程的丢失时无法避免的。这就好比从一个磁带复制音频，每次复制都会丢失一些原来音频的信息，复制音频的质量也会越来越差。因此如果希望编码后的视频能够清晰，

且希望得到最好的编码效率，需要有一个良好的视频源、优秀的编码算法、高性能的编码软件和恰当的编码参数。

音频和视频的编码算法很多，常见的有：

音频编解码器有：AAC、MPEG-3、Ogg Vorbis。

视频编解码器有：H.264、VP8、Ogg Theora。

有些编码器是受专利保护的，有些则是免费的。例如 Vorbis、Theora 是免费的，可以随意使用，而 H.264、MPEG-4 就需要支付专利费用。

延伸阅读：WebM

WebM 由 Google 于 2010 年 5 月发布，是一个开放、免费的媒体文件格式。现在主流的 Web 媒体都比较模糊，作为新的音频和视频格式，WebM 旨在改善这种现状，让 Web 视频清晰化。WebM 的后缀名是.webm。

WebM 格式其实是以 Matroska（即 MKV）容器格式为基础开发的新容器格式，里面包括了 VP8 编码器和 Ogg Vorbis 音频编码器，其中 Google 将其拥有的 VP8 视频编码技术以类似 BSD 授权开源，Ogg Vorbis 本来就是开放格式。Google 以不受限制许可的方式发布了 WebM 的规范和软件，包括源码和专利权。

WebM 标准的编解码器更加偏向于开源并且是基于 HTML5 标准的,WebM 项目旨在为对每个人都开放的网络开发高质量、开放的视频格式，其重点是解决视频服务这一核心的网络用户体验。

1.3HTML 5 Audio 和 Video 的限制

虽然 HTML 5 增加了音频和视频的元素，但是有些功能目前仍然没有得到 HTML 5 的支持。

（1）流式音频和视频。目前 HTML 5 中没有视频比特率切换的标准，所以 HTML 5 对视频的支持只限于加载的全部媒体文件。

（2）HTML 5 的媒体受到 HTTP 跨源资源共享的限制。

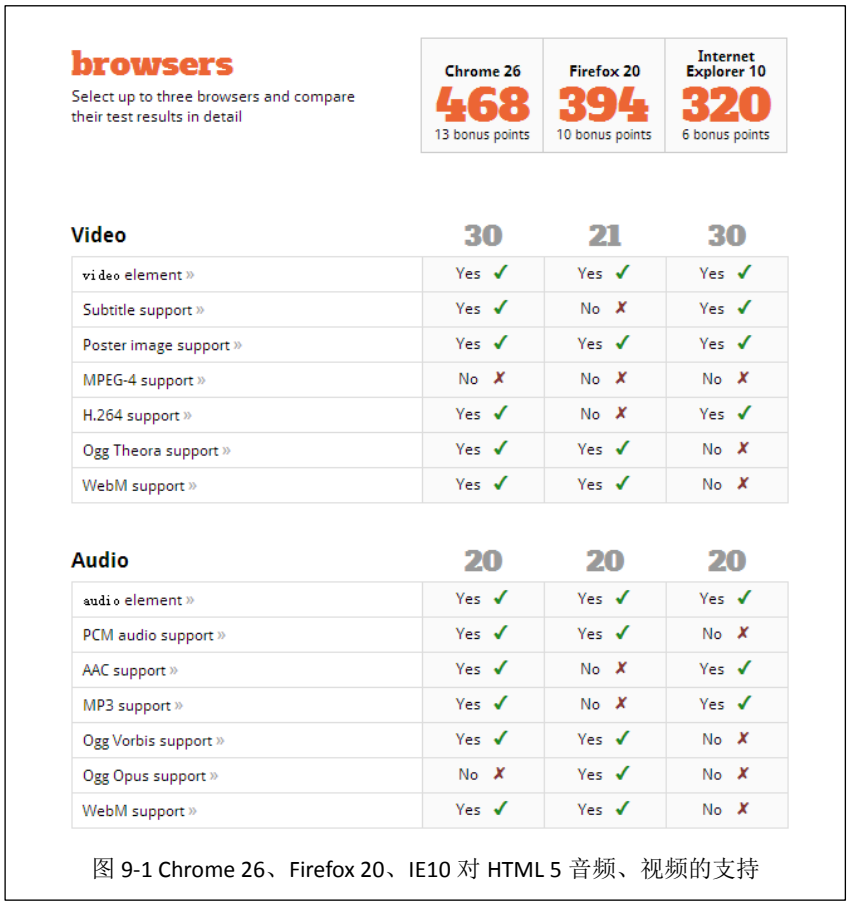
（3）HTML 5 自身无法在视频全屏播放时进行脚本控制。这个功能的缺失是出于安全的考虑，但是可以通过浏览器自身的功能进行解决。

1.4 浏览器对 HTML 5 的 Audio 和 Video 元素的支持

由于音频和视频的解码是有浏览器来进行的，因为除了了解浏览器对 Audio 和 Video 元素的支持外，还需要知道浏览器支持的编解码

器。

具体的支持情况可以通过网站查询：http://html5test.com。



browsers

Select up to three browsers and compare their test results in detail

	Chrome 26	Firefox 20	Internet Explorer 10
	468	394	320
	13 bonus points	10 bonus points	6 bonus points
Video	30	21	30
video element »	Yes ✓	Yes ✓	Yes ✓
Subtitle support »	Yes ✓	No ✗	Yes ✓
Poster image support »	Yes ✓	Yes ✓	Yes ✓
MPEG-4 support »	No ✗	No ✗	No ✗
H.264 support »	Yes ✓	No ✗	Yes ✓
Ogg Theora support »	Yes ✓	Yes ✓	No ✗
WebM support »	Yes ✓	Yes ✓	No ✗
Audio	20	20	20
audio element »	Yes ✓	Yes ✓	Yes ✓
PCM audio support »	Yes ✓	Yes ✓	No ✗
AAC support »	Yes ✓	No ✗	Yes ✓
MP3 support »	Yes ✓	No ✗	Yes ✓
Ogg Vorbis support »	Yes ✓	Yes ✓	No ✗
Ogg Opus support »	No ✗	Yes ✓	No ✗
WebM support »	Yes ✓	Yes ✓	No ✗

图 9-1 Chrome 26、Firefox 20、IE10 对 HTML 5 音频、视频的支持

二、video 和 audio 元素

2.1 在 HTML 4 页面中播放音频和视频的方法

在 HTML 4 中，没有音频和视频的元素，只能够使用<object>和<embed>元素来进行。

案例：

示例 9-1：在 HTML 4 页面中播放音频或视频

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>在 HTML 4 页面中播放音频或视频文件</title>
</head>
```

```
<body>
<center>
<h1>播放视频</h1>
<object                                    type="application/x-shockwave-flash"
data="FLVPlayer_Progressive.swf" width="673" height="378">
      <param name="quality" value="high">
      <param name="wmode" value="opaque">
      <param name="scale" value="noscale">
      <param name="salign" value="lt">
      <param name="FlashVars"

   value="&amp;MM_ComponentVersion=1&amp;skinName=Corona_Skin_
3&amp;streamName=medias/QQ-AD-1&amp;autoPlay=false&amp;autoRe
wind=false"
      />
      <param name="swfversion" value="8,0,0,0">
</object>
<h1>播放音频</h1>
<embed
   src="medias/GeGe-ChangShilei.mp3"
   width="500"
   height="82">
</embed>
<br><br>
</center>
</body>
</html>
```

通过这个案例，可以看到在 HTML 4 页面中播放音频和视频的不足是非常明显的。

（1）冗长的代码，且非常不简洁。

（2）需要使用第三方插件，如果用户的计算机没有安装相应的插件，则视频就无法播放。

（3）需要结合使用<object>和<embed>元素，并且需要添加大量的属性和参数，代码的书写难度非常高。

2.2 在 HTML 5 页面中播放音频和视频的方法

在 HTML 5 中，新增加了 video 和 audio 两个元素。video 元素专门用来播放网络上的视频，audio 元素用来播放网络上的音频。使用这两个元素后，就不需要其他插件了，只要浏览器能够支持 HTML 5

的这两个元素并且能够支持音频、视频文件的编解码器即可。同时，代码的书写方法也非常简单、清晰、简洁。

案例：

示例 9-2：在 HTML 5 页面中播放音频或视频

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>在 HTML 4 页面中播放音频或视频文件</title>
</head>

<body>
<center>
<h1>播放视频</h1>
<video src="medias/WoWeiZiJiDaiYan-AD.mp4" controls>
    您的浏览器不支持 video 元素。
</video>
<h1>播放音频</h1>
<audio src="medias/GeGe-ChangShilei.mp3" controls>
  您的浏览器不支持 audio 元素。
</audio>
</center>
</body>
</html>
```

由于目前浏览器能够支持的编解码器不一致，为了确保一个视频能够同时被所有支持 HTML 5 的浏览器支持，则可以通过<source>元素来为同一个视频指定多个源，供不同的浏览器来选择适合自己的。

案例：

示例 9-3：使用多个源，在 HTML 5 页面中播放音频或视频

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>使用多个源，在 HTML 5 页面中播放音频或视频文件</title>
</head>

<body>
<center>
<h1>播放视频</h1>
```

```
<video>
    <source src="medias/WoWeiZiJiDaiYan-AD.mp4" type="video/mp4">
    <source src="medias/WoWeiZiJiDaiYan-AD.webm"
type="video/webm">
    <source src="medias/WoWeiZiJiDaiYan-AD.ogv" type="video/ogv">
    您的浏览器不支持 video 元素。
</video>

<h1>播放音频</h1>
<audio controls>
  <source src="medias/GeGe-ChangShilei.mp3" type="audio/mp3">
    <source src="medias/GeGe-ChangShilei.ogg" type="audio/ogg">
  您的浏览器不支持 audio 元素。
</audio>
</center>
</body>
</html>
```

三、video 和 audio 的属性

video 元素和 audio 元素的属性大致相同。

3.1src

元素介绍：

src 属性用来指定媒体数据的 URL 地址。

3.2preload

元素介绍：

该属性中指定视频或者音频是否进行预加载。如果使用预加载，浏览器将会预先将视频或者音频进行缓冲，这样可以加快播放的速度。因为在进行数据播放时，可能数据已经缓冲一部分或者缓冲完成。

preload 属性值有三种：

（1）none：表示不进行预先加载。

（2）metadata：表示只预先加载媒体的元数据，主要包括媒体字节数、第一帧、播放列表、持续时间等信息。

（3）auto：表示预加载全部视频或音频。该值是默认值。

3.3poster

元素介绍：

该属性为 vedio 元素的独有属性，用来在视频不可用时，向用户展示一张替代图片，从而避免视频不可用时，页面出现一片空白。

3.4loop

元素介绍：

该属性中指定是否循环播放视频或音频。

3.5controls

元素介绍：

该属性中指定是否为视频或音频添加浏览器自带的播放用的控制条。控制条中具有播放、暂停等按钮。不同的浏览器，自带的控制条的样式不同。

可以通过撰写脚本的方式自定义控制条，而不使用浏览器默认的控制条。

3.6width 和 height

元素介绍：

该属性为 vedio 元素的独有属性。用来指定视频的宽度和高度。

3.7error

元素介绍：

在读取、使用媒体数据的过程中，在正常情况下，video 和 audio 元素的 error 属性为 null。当出现错误时，error 属性将返回一个 MediaError 对象，该对象通过 code 的方式将错误状态提供出来。错误状态值为只读属性，且有 4 个可能值：

（1）1：MEDIA_ERR_ABORTED，数据在下载中因为用户操作的原因而被中止。

（2）2：MEDIA_ERR_NETWORK，确认媒体资源可用，但是在下载时出现网络错误，媒体数据的下载过程被中止。

（3）3：MEDIA_ERR_DECODE，确认媒体资源可用，但是解码时发生错误。

（4）4：MEDIA_ERR_SRC_NOT_SUPPORTER，媒体格式不被支持。

案例：

示例 9-4：读取错误状态

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>读取错误状态</title>
</head>

<body>
<video id="ShowVideo" src="medias/WoWeiZiJiDaiYan-AD.mp4"
controls></video>
<script>
var videoerror = document.getElementById("ShowVideo");
videoerror.addEventListener("error",function()
{
   var errorinfo = videoerror.error;
   switch(errorinfo.code)
   {
        case 1:
             alert("用户取消了视频的载入。");
             break;
        case 2:
             alert("网络故障，造成数据载入失败。");
             break;
        case 3:
             alert("解码错误，请重新访问。");
             break;
        case 4:
             alert("浏览器不支持获得的视频格式。");
             break;
   }
},false);
</script>
</body>
</html>
```

3.8networkState

元素介绍：

媒体数据在加载过程中可以使用 video 元素或 audio 元素的 networkState 属性读取当前网络状态，其可能值有 4 个，该属性值为只读属性。

（1）0：NETWORK_EMPTY：初始状态。

（2）1：NETWORK_IDLE：浏览器已经选择好用什么编码格式来播放媒体，但尚未建立网络连接。

（3）2：NETWORK_LOADING：媒体数据加载中。

（4）3：NETWORK_NO_SOURCE：没有支持的编码格式，不进行加载。

3.9currentSrc

元素介绍：

currentSrc 属性用来读取 video 元素和 audio 元素中正在播放中的媒体数据的 URL 地址。

3.10buffered

元素介绍：

buffered 属性用来返回一个对象,该对象实现 TimeRange 接口，以确认浏览器是否已缓存媒体数据。

buffered 属性值为只读属性。

3.11readyState

元素介绍：

该属性返回 video 或 audio 元素中媒体当前播放位置的就绪状态，readyState 属性值为只读属性，其有 5 个可能值。

（1）0：HAVE_NOTHING：没有获得任何媒体的信息，当前播放位置没有播放数据。

（2）1：HAVE_METADATA：已经获得到足够的媒体信息，但是当

前播放位置没有有效的媒体数据，暂时不能够播放。

（3）2：HAVE_CURRENT_DATA：当前播放位置已经有数据可以播放，但没有获取到可以让播放器前进的数据。如果是视频，就是说当前帧数据已经获得，下一帧数据没有获得。

(4)3:HAVE_FUTURE_DATA:当前播放位置已经有数据可以播放，而且也获取到了可以让播放器前进的数据。当媒体为视频时，意思是当前帧的数据已经获得，且下一帧的数据也已经获得。

(5)4:HAVE_ENOUGH_DATA:当前播放位置已经有数据可以播放，下一帧数据已经获得,且浏览器确认媒体数据以某一种速度进行加载，可以保证有足够的后续数据进行播放。

3.12seeking 和 seekable

元素介绍：

seeking 属性返回一个布尔值，表示浏览器是否正在请求某一特定播放位置的数据，true 表示浏览器正在请求数据，false 表示浏览器已经停止请求。

seekable 属性返回一个 TimeRange 对象，该对象表示请求到的数据的时间范围。

seeking 和 seekable 属性值均为只读属性。

3.13currentTime、startTime 和 duration

元素介绍：

currentTime 属性来读取媒体的当前播放位置，通过修改该属性值可以修改当前播放位置。如果修改的位置上没有可用的媒体数据，将产生 INVALID_STATE_ERR 异常,如果修改的位置上的数据浏览器上没有获得，将产生 INDEX_SIZE_ERR 异常。

startTime 属性来读取媒体播放的开始时间，通常为 0。

duration 属性来读取媒体文件总的播放时间。

currentTime、startTime、duration 属性的值为时间，单位为秒。currentTime 的属性值为读写属性，startTime、duration 属性值为只读属性。

3.14played、paused、ended

元素介绍：

played 属性可以返回一个 TimeRange 对象，该对象中可以读取媒体文件已经播放部分的时间段。该时间段的开始时间为已播放部分的开始时间，结束时间为已播放部分的结束时间。

paused 属性可以返回一个布尔值,表示是否处于暂停播放状态。true 表示目前已经暂停播放，false 表示媒体正在播放。

ended 属性可以返回一个布尔值，表示是否已经播放完毕。true 表示媒体播放完毕，false 表示媒体没有播放完毕。

played、pause、ended 属性值均为只读属性。

3.15defaultPlaybackRate、playbackRate

元素介绍：

defaultPlaybackRate 属性读取或修改媒体默认的播放速率。

playbackRate 属性读取或修改媒体当前的播放速率。

3.16volume、muted

元素介绍：

volume 属性读取或修改媒体播放的音量，范围为 0 到 1，0 为静音，1 为最大音量。

muted 属性读取或修改媒体的静音状态,该属性值为布尔值,true 表示处于静音状态，false 表示处于非静音状态。

四、video 和 audio 的方法

video 元素和 audio 元素具有四种方法。

4.1play

元素介绍：

使用 play 方法来播放媒体,自动将元素的 paused 值变为 false。

4.2pause

元素介绍：

使用 pause 方法来暂停播放，自动将元素的 paused 值变为 true。

4.3load

元素介绍：

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

4.4canPlayType

元素介绍：

使用 canPlayType 方法来测试浏览器是否支持指定的媒体类型，该方法的定义是：

```
var supportTypeInfo = videoElement.canPlayType(type)
```

其中参数 type 的指定方法是使用播放文件的 MIME 类型来指定，可以在指定的字符串中加上表示媒体编码格式的 codes 参数。

该方法返回 3 个可能值：

（1）空字符串：表示浏览器不支持此种媒体类型。

（2）maybe：表示浏览器可能支持此种媒体类型。

（3）probably：表示浏览器确定支持此种媒体类型。

案例：

示例 9-5：媒体播放器

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8"/>
<title>视频播放器</title>
<style type=”text/css”>
body{
   margin:30px auto;
   padding:0px;
```

```
    text-align:center;
    font-size:10px;
    font-family:微软雅黑;
    background:#F2F2F2;
}
progress{
    width:552px;
    border:1px solid #333;
    background:#FFF;
}
button{
    margin-top:10px;
    border:none;
    background-color:#41A81E;
    color:#F2F2F2;
    width:68px;
    height:22px;
    text-align:center;
    vertical-align:middle;
    border-radius:8px;
    border:1px #078111 solid;
}
#showTime{
    margin-top:10px;
    color:#333;
}
#progressValue{
        width:0%;
        height:20px;
        background:#FFF;
        cursor:default;
}
</style>
</head>
<body>
    <h2>构建自己的视频播放器</h2>
    <!--视频播放窗口-->
    <div id="PlayVideo">
            <video id="VideoPlayer" width="552" height="311">
                <source src="medias/Wo-ShangWenjie.mp4">
                <source src="medias/Wo-ShangWenjie.ogv">
                <source src="medias/Wo-ShangWenjie.webm">
                您的浏览器不支持 video 元素
            </video>
        </div>
        <div id="buttonDiv">
```

```
        <!--播放进度条-->
        <progress id="playPercent" max=100>
          <div id="progress">
                <div id="progressValue"></div>
            </div>
        </progress>
        <!--播放控制按钮-->
        <div id="PlayButton">
            <button id="btnPlay" onclick="PlayOrPause()" disabled/>播放
</button>
            <button id="btnSpeedUp" onclick="SpeedUp()" disabled/>加速
播放</button>
            <button id="btnSpeedDown" onclick="SpeedDown()"
disabled/>减速播放</button>
            <button id="btnSlowPlay" onclick="SlowPlay()" disabled/>慢动
作</button>
            <button id="btnMute" onclick="setMute()" disabled/> 静 音
</button>
            <button id="btnVolumeUp" onclick="VolumeUp()" disabled/>
增大音量</button>
            <button id="btnVolumeDown" onclick="VolumeDown()"
disabled/>降低音量</button>
        </div>
    </div>
    <div id="showTime"></div>
</body>
<script type="text/javascript">
var speed=1;
var volume=1;
var muted=false;
var video = document.getElementById("VideoPlayer");
var showTime = document.getElementById("showTime");
if(video.canPlayType)
{
    video.addEventListener('loadedmetadata',loadedmetadata,false);
    video.addEventListener('ended',videoEnded,false);
    video.addEventListener('play',videoPlay,false);
    video.addEventListener('pause',videoPause,false);
    video.addEventListener('timeupdate',updateProgress,false);
    video.addEventListener("error",catchError,false);
    progress=document.getElementById("progress");
    progress.onmouseover=progress_mouseover;
    progress.onmouseout=progress_mouseout;
    progress.onclick=progress_click;
    playPercent=document.getElementById("playPercent");
    playPercent.onmouseover=progress_mouseover;
```

```
    playPercent.onmouseout=progress_mouseout;
    playPercent.onclick=playPercent_click;
}
function catchError()
{
    var error = video.error;
    switch(error.code)
    {
        case 1:
            alert("视频的载入过程以为用户操作，已经被中止。");
            break;
        case 2:
            alert("网络发生故障，视频的载入过程被中止。");
            break;
        case 3:
            alert("浏览器对视频的解码错误，无法播放视频。");
            break;
        case 4:
            alert("视频不可访问或视频编码器浏览器不支持。");
            break;
      }
}

function loadedmetadata()
{
    var btnPlay=document.getElementById("btnPlay");
    btnPlay.innerHTML="暂停";
    btnPlay.disabled="";
    video.play();
    var buttonDiv=document.getElementById("buttonDiv");
    buttonDiv.style.display="white";
}
function PlayOrPause()
{
    if(video.paused)
    {
        video.play();
        video.playbackRate=speed;
        video.muted=muted;
        video.volume=volume;
    }
    else
        video.pause();

}
function videoEnded(ev)
```

```
{
	video.currentTime=0;
	this.pause();
}
function videoPlay(ev)
{
	var btnPlay=document.getElementById("btnPlay");
	btnPlay.innerHTML="暂停";
	document.getElementById("btnSpeedUp").disabled="";
	document.getElementById("btnSpeedDown").disabled="";
	document.getElementById("btnSlowPlay").disabled="";
	document.getElementById("btnMute").disabled="";
	document.getElementById("btnVolumeUp").disabled="";
	document.getElementById("btnVolumeDown").disabled="";
}
function videoPause(ev)
{
	var btnPlay=document.getElementById("btnPlay");
	btnPlay.innerHTML="播放";
	document.getElementById("btnSpeedUp").disabled="disabled";
	document.getElementById("btnSpeedDown").disabled="disabled";
	document.getElementById("btnSlowPlay").disabled="disabled";
	document.getElementById("btnMute").disabled="disabled";
	document.getElementById("btnVolumeUp").disabled="disabled";
	document.getElementById("btnVolumeDown").disabled="disabled";
}
function updateProgress()
{
   var
value=Math.round((Math.floor(video.currentTime)/Math.floor(video.duration)
)*100,0);
   var progress = document.getElementById('playPercent');
	progress.value = value;
	var progressValue=document.getElementById("progressValue");
	progressValue.style.width = value+"%";

showTime.innerHTML=calcTime(Math.floor(video.currentTime))+'/'+calcTim
e(Math.floor(video.duration));
}
function calcTime(time)
{
	var hour;
	var minute;
	var second;
	hour=String(parseInt(time/3600,10));
	if (hour.length == 1)	hour	= '0' + hour;
```

```
    minute=String(parseInt((time%3600)/60,10));
    if (minute.length == 1)    minute    = '0' + minute;
    second=String(parseInt(time%60,10));
    if (second.length == 1)    second    = '0' + second;
    return hour+":"+minute+":"+second;
}
function progress_mouseover(ev)
{

showTime.innerHTML=calcTime(Math.floor(video.currentTime))+'/'+calcTim
e(Math.floor(video.duration));
    ev.stopPropagation();
}
function progress_mouseout(ev)
{
    showTime.innerHTML="";
}
function playPercent_click(evt)
{
    if(evt.offsetX)
    {
        playPercent=document.getElementById("playPercent");
        video.currentTime    =    video.duration    *    (evt.offsetX    /
playPercent.clientWidth);
    }
}
function progress_click(evt)
{
    progress=document.getElementById("progress");
    if(evt.offsetX)
        video.currentTime    =    video.duration    *    (evt.offsetX
/progress.clientWidth);
    else
        video.currentTime    =    video.duration    *    (evt.clientX
/progress.clientWidth);
}
function SpeedUp()
{
    video.playbackRate+=1;
    speed=video.playbackRate;
}
function SpeedDown()
{
    video.playbackRate-=1;
    if(video.playbackRate<0)
        video.playbackRate=0;
```

```
    speed=video.playbackRate;
}
function SlowPlay()
{
    var btnSlowPlay=document.getElementById("btnSlowPlay");
    if(btnSlowPlay.innerHTML=="慢动作")
    {
  video.playbackRate=0.5;
        btnSlowPlay.innerHTML="正常速度";
        document.getElementById("btnSpeedUp").disabled="disabled";
        document.getElementById("btnSpeedDown").disabled="disabled";
    }
    else
    {
  video.playbackRate=1;
        btnSlowPlay.innerHTML="慢动作";
        document.getElementById("btnSpeedUp").disabled="";
        document.getElementById("btnSpeedDown").disabled="";
    }
    speed=video.playbackRate;
}
function setMute()
{
    if(!video.muted)
    {
        video.muted=true;
        document.getElementById("btnMute").innerHTML="取消静音";
    }
    else
    {
        video.muted=false;
        document.getElementById("btnMute").innerHTML="暂时静音";
    }
    muted=video.muted;
}
function VolumeUp()
{
    if(video.volume<1)
        video.volume+=0.1;
    volume=video.volume;
}
function VolumeDown()
{
    if(video.volume>0)
         video.volume-=0.1;
    volume=video.volume;
```

```
}
</script>
</html>
```

五、video 和 audio 的事件

5.1 事件处理方式

使用 video 元素或 audio 元素读取或者播放媒体数据时，会触发一系列的事件，如果使用 JavaScript 脚本来捕捉这些事件，就可以对事件进行处理。

对事件的捕捉，有两种方式：

方式一：监听。

使用 video 元素或 audio 元素的 addEventListener 方法来对事件的发生进行监听，该方法的定义如下：

```
videoElement.addEventListener(type,listener,userCapture);
```

方式二：使用 JavaScript 脚本中的获取事件句柄。

例如：

```
<video id=”videodemo”
        src=”medias/video.webm” onplay=”begin_playing()；”></video>
<script>
function begin_playing()
{
…
}
</script>
```

5.2 事件

video 和 audio 在浏览器开始请求媒体数据、下载媒体数据、播放媒体数据到播放结束，触发的事件如下表所示。

表 7-1 input 的属性

事件	描述
abort	当音频/视频的加载已放弃时
canplay	当浏览器可以播放音频/视频时
canplaythrough	当浏览器可在不因缓冲而停顿的情况下进行播放时
durationchange	当音频/视频的时长已更改时

emptied	当目前的播放列表为空时
ended	当目前的播放列表已结束时
error	当在音频/视频加载期间发生错误时
loadeddata	当浏览器已加载音频/视频的当前帧时
loadedmetadata	当浏览器已加载音频/视频的元数据时
loadstart	当浏览器开始查找音频/视频时
pause	当音频/视频已暂停时
play	当音频/视频已开始或不再暂停时
playing	当音频/视频在已因缓冲而暂停或停止后已就绪时
progress	当浏览器正在下载音频/视频时
ratechange	当音频/视频的播放速度已更改时
seeked	当用户已移动/跳跃到音频/视频中的新位置时
seeking	当用户开始移动/跳跃到音频/视频中的新位置时
stalled	当浏览器尝试获取媒体数据，但数据不可用时
suspend	当浏览器刻意不获取媒体数据时
timeupdate	当目前的播放位置已更改时
volumechange	当音量已更改时

六、案例：拍个 DV 让大家看

演示：

第一步：使用手机拍摄视频。
第二步：获得视频文件。
第三步：对视频文件进行编码，以支持 HTML 5。
第四步：制作网页，并发布。

七、讨论与思考

7.1 如何进行视频编辑？

（1）使用手机、DV 等拍摄的视频，如何进行编辑？例如添加背景音乐、字幕等。

（2）视频编辑的常用软件有哪些？

7.2 视频分辨率

（1）经常听到的标清、高清、超清有哪些区别？分别采用了那些编码器？

（2）视频的分辨率和清晰度有关系么？