video的简单操作及在IOS和Android下的差异

这次小编与大家分享一下，如何对video做一些简单基本的操作，包括了播放器的播放、暂停，音量的读取、设置的相关操作。

**一、获取影片总时长**

对video操作进行操作之前，应先给video标签加个id，便于我们获取video元素，这样我们就可以开启对video的探索之旅。首先要得到的是影片的一些信息，其中一个就是总时长。

<video id="video" controls preload="auto" width="100%" x-webkit-airplay="true" webkit-playsinline="true" poster="images/cover1.jpg" src="mp4/1.mp4"></video>

获取video视频总时长，要用到video的一个事件loadedmetadata，这个事件的触发表示元数据（媒体的一些基本信息）已经加载完成，用addEventListener监听事件

var video = document.getElementById('video');//获取video元素

myVideo.addEventListener("loadedmetadata", function(){

//要执行的代码

});

这时已经监听了，那么接下来要做的就是获取总时长，其实就是用到了video的一个属性：duration

var video = document.getElementById('video');//获取video元素

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

totalTime = video.duration;//获取总时长

});

需要注意的是获取到的在总时长的单位为秒，显示的时候可根据需要去转换。

**二、播放、暂停**

我们经常会需要对video设置播放或者暂停，播放或者暂停用到的video的两个方法就是play和pause

//播放

function play(){

video.play();

}

//暂停

function pause(){

video.pause();

}

需要注意的是，当播放结束后再运行play方法将会从头播放。

三、**获取影片的播放时间和设置播放点**

 很多时候我们都想知道video视频播到哪了，这需要监听一个事件和获取一个属性的值，用到的是video的timeupdate事件和currentTime属性

//更新播放时间点

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

var currentTime = video.currentTime; //获取当前播放时间

console.log(currentTime); //在调试器中打印

// 如果播放时间等于视频总时间，就暂停播放

if(currentTime == totalTime){

pause();

}

});

设置播放点，设置播放点用得还是currentTime属性，currentTime属性是可读写的，要注意的是设置值的单位是秒，如果播放点不是秒为单位那就要进行转换

//设置播放点

function playBySeconds(num){

video.currentTime = num;

}

playBySeconds(30); // 从30秒开始播放

**四、音量的获取和设置**

获取音量可以直接用volume属性，但是在这里还要介绍的是音量改变的触发事件，方面以后需要自定义UI使用，那就是volumechange事件

//改变音量

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

var videoVolume = video.volume;//获取当前音量

console.log(videoVolume);//在调试器中打印

});

要注意的是音量的范围值为0~1，一般在UI中都是用百分比，所以需要的时候要进行转换。

音量是可以通过改变属性来设置的，跟播放的时间点是相似的，只不过音量设置的是volume属性。

//设置音量

function setVol(num){

video.volume = num;

}

setVol(0.5);

案例演示：

二维码

**五、一些常用且需要重点关注的video事件**

|  |  |  |
| --- | --- | --- |
| **event** | **iOS** | **Android** |
| play | 只是要播放视频，响应的是video.play()方法，并不代表已经开始播放 | 和iOS一样，仅是响应video.play()方法 |
| durationchange | 会执行一次，一定会获取到视频的duration | 可能会执行多次，只有最后一次才能获取到真实的duration，前面的duration都是0；但低版本Android可能获取到的duration是0或1；（本文提到的低版本Android大部分是4.1以下） |
| canplay | 可以认为是视频元素没有问题，可以运行，没有更多含义了，基本用不上 | 同iOS |
| canplaythrough | 会有明确的缓冲，表示可以流畅播放了； | 没有什么用，视频仍然会卡住，数据可能还没有开始加载； |
| playing | 明确表示播放开始了； | 依然没有用，视频可能并没有开始播放； |
| progress | 有明确的下载，可以获取到当前的buffer，并且全部下载完毕后不在触发； | 不一定有明确的数据下载，并且全部下载完毕后依然继续触发； |
| timeupdate | 会有明确的进度变化，可以获取到currentTime； | 进度不一定变化，currentTime可能总是0，但是第一次有currentTime变化的timeupdate事件一定代表了视频开始播放了； |
| error | iOS中会有明确的错误抛出； | Android中某些浏览器会莫名其妙的抛出error； |
| stalled | 网络状况不佳，导致视频下载中断； | 在没有play之前，也可能会抛出该事件。 |

**六、属性差异**

|  |  |  |
| --- | --- | --- |
| **attributes** | **iOS** | **android** |
| poster 封面图片 | 支持，但是加载速度明显比在<img>中要慢； | 不一定支持（浏览器厂商的实现标准不统一）； |
| preload 预加载 | iPhone不支持； | 可能支持； |
| autoplay 自动播放 | iPhone Safari中不支持，但在webview中可能被开启；iOS开发文档明确说明蜂窝网络下不允许autoplay； | 可能支持； |
| loop 循环播放 | 支持 | 可能支持； |
| controls 控制条 | 支持，但是需要开始播放了才显示 | 基本都支持显示或者不显示 |
| width和height | 一定给出明确的属性设置，且不能为0； | 如果不设置，仅仅通过CSS样式去控制视频大小，可能会导致标签失效。 |

### 七、其他怪异bug和不友好表现

|  |  |
| --- | --- |
| **iOS** | **android** |
| 物理位置覆盖在video区域上的元素，click和touch等事件会失效，比如一个a链接如果覆盖在video上，那么点击后没有任何效果。 | - |
| iOS8.0+中，单页面播放视频超过16个，再播放的视频全部MediaError解码异常无法播放。 | - |
| iPhone的Safari会弹出一个全屏的播放器来播放视频，iPad则支持内联播放。iOS7+ 如果webview（比如微信）开启了webview.allowsInlineMediaPlayback = YES;，可以通过设置webkit-playsinline属性支持内联播放； | 支持内联播放，但某些厂商会用自己的播放器劫持原生的视频播放； |
| 下载视频时，会先发送一个2字节的请求来获取视频元数据（比如时长），然后再不断的发送分包续传（206）请求来下载视频，抓包显示请求数和请求量至少有一倍的冗余（x2），这个严重的bug在iOS8中有明显的修复，但是分包的206请求仍然会有冗余数据的下载，浪费了流量。 | 比iOS的处理方式好，没有第一个2字节请求，没有流量损耗； |
| - | 低版本Android（<=4.0.4）中，video如果在有相对和决定定位的层中，可能会导致整个页面错位。 |
| - | 某些浏览器厂商会劫持video，用其“自己”的播放器来播放视频，“破坏”了产品本身的播放体验，那么只能case by case的解决了。 |
| 加载视频时没有进度提示，视觉上看不出是播放完了还是卡住了； | 加载视频时，大都会显示一个自带的loading UI。 |