# HTML5 第二天笔记

## 音视频处理

### 1. 视频处理

#### 1）基础内容

##### a. 目前实现网页视频播放的技术 - Flash

* Flash并不是浏览器原生支持(第三方组件)
* Flash的性能并不好
* 移动端不支持Flash技术

##### b. HTML 5 提供的视频处理技术 - <video>

* 提供了相对应的基本处理方式
* 提供了高级编程自定义方式

##### c. <video>元素所支持的视频格式

| **视频格式** | **类型** | **描述** |
| --- | --- | --- |
| MP4格式 | .mp4 | 目前最主流的视频格式。 |
| OGV格式 | .ogv |  |
| WEBM格式 | .webm | 是由Google公司推出的，目前唯一一个支持超高清的视频格式。 |

#### 2）如何使用<video>元素

##### a. 引入单个视频格式

<video> 标签的编写结构如下：

<video src="视频文件的路径" autoplay>

浏览器不支持的提示内容

</video>

例如以下示例代码：

<video src="../DATA/oceans-clip.mp4" autoplay width="640px" height="400px" style="background:black;">

非常抱歉,你的浏览器不支持该视频!

</video>

| **属性名称** | **类型** | **描述** |
| --- | --- | --- |
| width | Number | 设置视频的宽度。 |
| height | Number | 设置视频的高度。 |
| style |  | 设置CSS样式属性。 |
| class |  | 设置CSS样式属性。 |
| autoplay | Boolean | 设置是否自动播放视频。 |

##### b. 引入多个视频格式

<video> 标签和 <source> 标签的编写结构如下：

<video autoplay>

<source src="视频文件的路径" />

<source src="视频文件的路径" />

<source src="视频文件的路径" />

浏览器不支持的提示内容

</video>

例如以下示例代码：

<!-- 解决了浏览器对视频格式的兼容问题 -->

<video autoplay>

<!--

<source>元素

\* 引入视频文件(一个<video>元素允许包含多个<source>)

-->

<source src="../DATA/oceans-clip.mp4" />

<source src="../DATA/oceans-clip.ogv" />

<source src="../DATA/oceans-clip.webm" />

</video>

#### 3）<video>元素的属性

* autoplay属性：自动播放
* controls属性：提供控制面板

<video src="../DATA/oceans-clip.mp4" controls></video>

* loop属性：循环播放

<video src="../DATA/oceans-clip.mp4" autoplay loop></video>

* poster属性：播放之前实现一张图片

<video src="../DATA/oceans-clip.mp4" controls poster="../DATA/oceans-clip.png"></video>

* preload属性：预加载视频
  + none：不预加载
  + auto：默认值,尽快预加载
  + metadata：预加载除视频之外的内容(宽度、高度等)

#### 4）视频高级编程

##### a. 事件

| **事件名称** | **作用** |
| --- | --- |
| play | 视频播放时触发。 |
| pause | 视频暂停时触发。 |
| ended | 视频播放结束时触发。 |
| error | 视频播放错误时触发。 |

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>video元素的高级事件</title>

<style>

img {

position: absolute;

top : 66px;

left : 160px;

display : none;

}

</style>

</head>

<body>

<video id="mmedia" src="../DATA/oceans-clip.mp4" controls></video>

<img id="adv" src="../DATA/oceans-clip.png" width="320">

<script>

var mmedia = document.getElementById("mmedia");

var adv = document.getElementById("adv");

mmedia.addEventListener("pause",function(){

adv.style.display = "block";

});

mmedia.addEventListener("play",function(){

adv.style.display = "none";

});

</script>

</body>

</html>

##### b. 方法

| **方法名称** | **作用** |
| --- | --- |
| play() | 用于播放视频。 |
| pause() | 用于暂停视频。 |
| load() | 用于加载视频。 |
| canPlayType() | 判断当前浏览器是否支持当前视频格式。 |

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>video元素的高级方法</title>

</head>

<body>

<video id="mmedia" src="../DATA/oceans-clip.mp4"></video>

<br>

<input type="button" id="btn" value="播放">

<script>

var btn = document.getElementById("btn");

var mmedia = document.getElementById("mmedia");

btn.onclick = function(){

if(mmedia.paused){// 播放

mmedia.play();

this.value = "暂停";

}else{// 暂停

mmedia.pause();

this.value = "播放";

}

}

</script>

</body>

</html>

##### c. 属性

| **属性名称** | **作用** |
| --- | --- |
| paused | 如果视频为暂停或未播放时,返回true。 |
| ended | 如果视频播放完毕时,返回true。 |
| duration | 返回当前视频的时长。 |
| currentTime | 获取或设置视频的当前位置。 |

**参考内容：**[https://developer.mozilla.org/en-US/docs/Web/HTML/Supportedmediaformats](https://developer.mozilla.org/en-US/docs/Web/HTML/Supported_media_formats)

### 2. 音频处理

#### 1）基础内容

##### a. 目前音频处理技术

* Flash技术也可以音频处理
* Media Player播放器允许嵌入在网页中

##### b. HTML5 提供的音频处理 - <audio>

* 浏览器原生支持
* 性能很好
* 移动端支持

##### c. <audio>元素支持的音频格式

| **音频格式** | **类型** | **描述** |
| --- | --- | --- |
| MP3格式 | .mp3 | 目前最主流的音频格式。 |
| OGG格式 | .ogg |  |
| WAV格式 | .wav |  |

#### 2）如何使用<audio>元素

##### a. 引入单个音频格式

<audio src="音频文件的路径" autoplay>

浏览器不支持的提示内容

</audio>

##### b. 引入多个音频格式

<audio autoplay>

<source src="音频文件的路径" />

<source src="音频文件的路径" />

<source src="音频文件的路径" />

浏览器不支持的提示内容

</audio>

#### 3）<audio>元素的特有属性

| **属性名称** | **作用** |
| --- | --- |
| autoplay | 设置是否自动播放。 |
| controls | 提供控制面板。 |
| loop | 设置循环播放。 |
| preload | 预加载视频。 |

#### 4）<audio>元素的高级编程

##### a. 事件

| **事件名称** | **作用** |
| --- | --- |
| play | 视频播放时触发。 |
| pause | 视频暂停时触发。 |
| ended | 视频播放结束时触发。 |
| error | 视频播放错误时触发。 |

##### b. 方法

| **方法名称** | **作用** |
| --- | --- |
| play() | 用于播放视频。 |
| pause() | 用于暂停视频。 |
| load() | 用于加载视频。 |
| canPlayType() | 判断当前浏览器是否支持当前视频格式。 |

##### c. 属性

| **属性名称** | **作用** |
| --- | --- |
| paused | 如果视频为暂停或未播放时,返回true。 |
| ended | 如果视频播放完毕时,返回true。 |
| duration | 返回当前视频的时长。 |
| currentTime | 获取或设置视频的当前位置。 |

## Canvas 画布

### 1. 基础内容

#### 1）Canvas技术允许在HTML页面直接绘制图形

* 不再需要引入外部图片(图形),HTML页面性能有所提高。
* 可以实现一些比较复杂的图形绘制工作。

#### 2）Canvas技术主要应用方向

* Web应用方面主要实现图表类。
* 网页游戏方面 - 主要实现2D效果。

#### 3）HTML5 提供有关图形方面的技术

* Canvas - 主要以2D为主
* WebGL - 主要以3D为主
* SVG - 矢量图

#### 4）Canvas的特点

* 绘制的图形与HTML页面之间是无关系的
* 通过Canvas绘制的图形不能使用DOM API
* 通过Canvas绘制的图形不能绑定事件
* Canvas画布最终是以图片(png|jpg)形式出现
* 绘制图形默认的颜色为黑色

### 2. 如何使用Canvas画布

#### 1）在HTML页面中定义<canvas>元素

* 设置<canvas>元素宽度和高度使用属性方式。
* 使用CSS样式方式或属性方式设置<canvas>元素的宽度和高度。

<!--

在HTML页面中,定义<canvas>元素

\* 默认只定义<canvas>元素时

\* 效果非常类似于<div>元素,但不一样的地方:

\* <div>元素在默认情况下,不具有高度和宽度的

\* <canvas>元素在默认情况下,具有高度和宽度的

\* 宽度 - 300px

\* 高度 - 150px

\* 设置<canvas>元素的高度和宽度

\* (建议)使用属性width和height

\* 使用CSS中的属性width和height

\* 绘制的图形会被拉伸

-->

<canvas id="canvas" width="500px" height="500px" style="background:pink;"></canvas>

<!--

<canvas id="canvas" style="width:500px;height:500px;"></canvas>

-->

#### 2）在JS代码中实现

* 获取<canvas>元素
* 通过<canvas>元素,创建画布对象
  + getContext('2d')函数
  + 返回画布对象
* 利用画布对象进行图形的绘制

// 获取HTML页面中的<canvas>元素

var canvas = document.getElementById("canvas");

/\*

通过<canvas>元素,创建画布对象

使用getContext(type)函数,创建画布对象

\* 该函数返回画布对象

\* type参数

\* 设置当前创建的画布是2d还是3d

\* 注意

\* 参数选项是2d(3d效果使用WebGL)

\* 必须写成"2d"

\*/

var context = canvas.getContext("2d");

// 利用画布对象,进行绘制图形

context.fillRect(10,10,100,100);

### 3. 绘制图形

#### 1）绘制矩形

##### a. 绘制实心(填充)矩形

fillRect(x,y,width,height)

| **参数名称** | **作用** |
| --- | --- |
| x和y | 绘制矩形的左上角的坐标值 |
| width | 设置绘制矩形的宽度(单位为px) |
| height | 设置绘制矩形的高度(单位为px) |

##### b. 绘制空心(描边)矩形

strokeRect(x,y,width,height)

| **参数名称** | **作用** |
| --- | --- |
| x和y | 绘制矩形的左上角的坐标值 |
| width | 设置绘制矩形的宽度(单位为px) |
| height | 设置绘制矩形的高度(单位为px) |

##### c. 清除指定区域的矩形

clearRect(x,y,width,height)

| **参数名称** | **作用** |
| --- | --- |
| x和y | 绘制矩形的左上角的坐标值 |
| width | 设置绘制矩形的宽度(单位为px) |
| height | 设置绘制矩形的高度(单位为px) |

<canvas id="canvas" width="500px" height="500px"></canvas>

<script>

// 1. 获取<canvas>元素

var canvas = document.getElementById("canvas");

// 2. 创建画布对象

var context = canvas.getContext('2d');

// 3. 绘制图形

// a. 绘制实心矩形

context.fillRect(10,10,100,100);

// b. 绘制空心矩形

context.strokeRect(120,10,100,100);

// c. 清除指定区域的矩形

context.fillRect(230,10,100,100);

context.clearRect(240,20,80,80);

</script>

#### 2）设置样式

| **属性名称** | **作用** |
| --- | --- |
| fillStyle | 设置填充样式 |
| strokeStyle | 设置描边样式 |
| globalAlpha | 设置透明度(0-1) |

**值得注意的是：**

* 一定要先设置样式(颜色),再绘制图形。
* 每次改变样式(颜色),重新设置。

<canvas id="canvas" width="500px" height="500px"></canvas>

<script>

var canvas = document.getElementById("canvas");

var context = canvas.getContext('2d');

// a. 设置填充样式

context.fillStyle = "pink";

// b. 绘制实心矩形

context.fillRect(10,10,100,100);

context.fillStyle = "blue";

context.fillRect(10,120,100,100);

// 设置描边样式

context.strokeStyle = "red";

context.strokeRect(120,10,100,100);

context.strokeStyle = "green";

context.strokeRect(120,120,100,100);

// 设置透明度

context.globalAlpha = 0.5;

context.fillRect(230,10,100,100);

context.fillStyle = "black";

context.globalAlpha = 0.1;

context.fillRect(230,120,100,100);

</script>

#### 3）设置渐变

##### a. 线性渐变

createLinearGradient(x1,y1,x2,y2)

线性渐变主要依靠基准线概念，所谓基准线就是设置线性渐变的标准。

| **参数名称** | **作用** |
| --- | --- |
| x1和y1 | 基准线的起点坐标值 |
| x2和y2 | 基准线的终点坐标值 |

<canvas id="canvas" width="500px" height="500px"></canvas>

<script>

var canvas = document.getElementById("canvas");

var context = canvas.getContext('2d');

/\*

设置线性渐变

createLinearGradient(x1,y1,x2,y2)方法

\* 该方法具有返回值,是渐变对象

\*/

var grd = context.createLinearGradient(0,0,100,100);

/\*

设置线性渐变的颜色和位置

addColorStop(position,color)

\* position - 设置颜色的位置

\* 值的范围为 0 - 1

\* color - 设置颜色

\*/

grd.addColorStop(0,"red");

grd.addColorStop(1,"blue");

grd.addColorStop(0.5,"yellow");

// 将设置的线性渐变,赋值给样式(fillStyle和strokeStyle)

context.fillStyle = grd;

// 绘制矩形

context.fillRect(0,0,100,100);

</script>

##### b. 射线(扇形)渐变

createRadialGradient(x1,y1,r1,x2,y2,r2)

射线渐变主要依靠基准圆概念，所谓基准圆就是设置射线渐变的标准。

| **参数名称** | **作用** |
| --- | --- |
| x1和y1 | 第一个基准圆的圆心 |
| r1 | 第一个基准圆的半径 |
| x2和y2 | 第二个基准圆的圆心 |
| r2 | 第二个基准圆的半径 |

<canvas id="canvas" width="500px" height="500px"></canvas>

<script>

var canvas = document.getElementById("canvas");

var context = canvas.getContext("2d");

/\*

设置射线渐变

createRadialGradient(x1,y1,r1,x2,y2,r2)

\* 该方法返回渐变对象

\*/

var grd = context.createRadialGradient(100,100,100,canvas.width,canvas.height,200);

/\*

设置渐变颜色

\*/

grd.addColorStop(0,"red");

grd.addColorStop(1,"blue");

// 将渐变对象赋值给样式

context.fillStyle = grd;

// 绘制矩形

context.fillRect(0,0,canvas.width,canvas.height);

</script>

##### c. 设置渐变颜色

addColorStop(position,color)

| **参数名称** | **作用** |
| --- | --- |
| position | 设置渐变颜色的位置，值的范围必须是0-1。 |
| color | 设置渐变的颜色 |

#### 4）绘制文字

| **属性名称** | **作用** |
| --- | --- |
| font | 设置文字的属性（用法同 CSS 属性font）。 |
| textAlign | 水平对齐方式。（left：基准线在左边；center：基准线在中间；right：基准线在右边） |
| textBaseline | 垂直对齐方式。（top：基准线在上边；middle：基准线在中间；bottom：基准线在下边；hanging：悬挂基线；alphabetic：字母基线） |

##### a. 绘制实心文字

fillText(text,x,y)

| **参数名称** | **作用** |
| --- | --- |
| text | 绘制的文字内容。 |
| x和y | 绘制文字的坐标值。 |

##### b. 绘制空心文字

strokeText(text,x,y)

| **参数名称** | **作用** |
| --- | --- |
| text | 绘制的文字内容。 |
| x和y | 绘制文字的坐标值。 |

<canvas id="canvas" width="500px" height="500px"></canvas>

<script>

var canvas = document.getElementById("canvas");

var context = canvas.getContext('2d');

// 设置文字样式

context.font = "bold 48px 宋体";

// 基准线

context.beginPath();

context.moveTo(100,0);

context.lineTo(100,400);

context.stroke();

// 设置水平对齐

context.textAlign = "right";

// 绘制文字

context.fillText("text",100,50);

// 设置水平对齐

context.textAlign = "center";

// 绘制文字

context.fillText("text",100,100);

// 设置水平对齐

context.textAlign = "left";

// 绘制文字

context.fillText("text",100,150);

// 基准线

context.beginPath();

context.moveTo(0,300);

context.lineTo(500,300);

context.stroke();

// 设置垂直对齐

context.textBaseline = "top";

context.strokeText("text",0,300);

context.textBaseline = "middle";

context.strokeText("text",100,300);

context.textBaseline = "bottom";

context.strokeText("text",200,300);

context.textBaseline = "hanging";

context.strokeText("text",300,300);

context.textBaseline = "alphabetic";

context.strokeText("text",400,300);

</script>

#### 5）设置阴影

| **属性名称** | **作用** |
| --- | --- |
| shadowColor | 设置阴影颜色。 |
| shadowOffsetX | 设置水平方向阴影。 |
| shadowOffsetY | 设置垂直方向阴影。 |
| shadowBlur | 设置阴影程度。 |

**原创声明：**如发布或转载请注明以下作者信息，谢谢！

本文作者 - 金云龙

联系方式 - QQ：2080954535；微信：imlonge；微信公众号：longestory

个人网站：<http://www.longestory.com>