HTML5 canvas与SVG

十、canvas

canvas的基本用法

canvas是HTML5中的新的标签，表示一个HTML画布对象。

定义canvas元素

```
<canvas id="canvas" width="500" height="500">对不起，你的浏览器不支持canvas标签</canvas>
```

canvas只有两个属性（width与height）。默认的width是300px，高度是150px。 注意：不要使用css样式指定canvas的大小。

渲染上下文

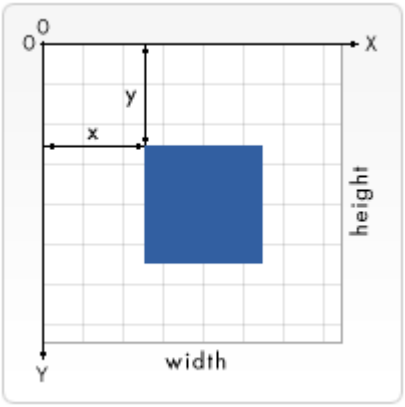
canvas初始化是空白的，要在上面用脚本画图首先需要其渲染上下文，可以通过canvas对象的 `getContext` 方法来获取，同时得到的还有一些画图用的函数。

这里来看看一个例子：

```
function draw() {
      var canvas = document.getElementById("canvas");
      if (canvas.getContext) {
        var ctx = canvas.getContext("2d");
        ctx.fillStyle = "rgb(200,0,0)";
        ctx.fillRect (10, 10, 55, 50);
        ctx.fillStyle = "rgba(0, 0, 200, 0.5)";
        ctx.fillRect (30, 30, 55, 50);
      }
    }
```

canvas绘制图形

我们先来看一下canvas的网格
在上面的例子中绘制了两个正方形，拿其中的红色正方形来讲。运行代码后，会发现它离左上角有一定的距离。这个距离是由传进的参数来决定的，是(10,10)。这表示这个正方形的原点的坐标。在canvas网格中，原点定在左上角的位置（坐标(0,0)）。所有canvas里所绘制的图形的位置都是相对这个原点的。看下图，这个蓝色的方块的位置就是距离左边x像素，距离上边y像素，即它的坐标是(x,y)。实际上，是(x-0,y-0)，由于canvas定原点坐标为(0,0)所以便是(x,y)。



下面我们来看看绘制的具体的方法

绘制矩形

以下单个函数是绘制矩形：

```
1. fillRect(x,y,width,height) //填充一个矩形
2. strokeRect(x,y,width,height) //绘制一个矩形的边框（但不填充矩形的内部）
3. clearRect(x,y,width,height) //擦出了指定的矩形，并且用一个透明的颜色填充
```

绘制路径
与绘制路径有关的函数：

```
1. beginPath()  //丢弃任何当前定义的路径并且开始一条新的路径。它把当前的点设置为(0,0)。当一个画布的环境第一次创建时，beginPath()方法会被显示地调用。
2. closePath()  //创建从当前点到开始点的路径。
3. stroke()  //方法绘制当前路径的边框。它意味着画轮廓，但是线条的可视化取决于strokeStyle、lineWidth、lineJoin、lineCap和miterLimit等属性。
4. fill()  //方法用于填充路径，默认是黑色。fill()方法使用fillStyle属性所指定的颜色、渐变和模式来填充当前路径。当调用该方法时，开放的路径会自动闭合。
```

moveTo()方法

```
1. moveTo(x,y) //用于定位绘画的位置，即将点移动到参数x,y所指定的坐标位置。
```

canvas初始化或者调用了beginPath()方法时，绘画开始的位置即原点(0,0)，使用moveTo()方法，我们可以将起始位置移动到任何我们想要的地方。

lineTo()方法

```
1. lineTo(x,y) //接受终点的坐标(x,y)作为参数。起始坐标取决于前一路径的终点坐标。当然，起始坐标也可以通过前面介绍的moveTo()方法来设置。
```

arc()方法

```
1. arc(x, y, radius, startAngle, endAngle, anticlockwise)
```

该方法接受6个参数。其中，x,y是圆心坐标；radius是圆的半径；startAngle和endAngle分别是起末弧度（以x轴为基准）；anticlockwise为true表示逆时针，反之为顺时针。

这边有一点是需要注意的，就是arc方法里的角度是以弧度为计算单位的，不是度。这么说吧，通常我们说的180度，就等价于PI。两者的计算公式是这样的：radians=(Math.PI/180)*degrees.(其中，radians代表弧度，degrees代表度)

贝塞尔与二次方曲线

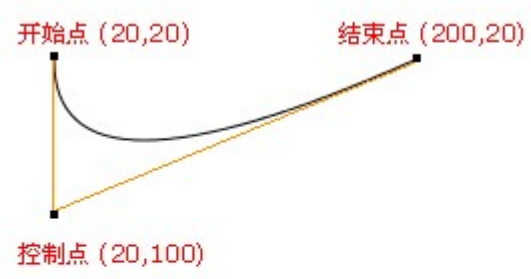
贝塞尔曲线(Bézier curve)，又称贝兹曲线或贝济埃曲线，是应用于二维图形应用程序的数学曲线

二次贝塞尔曲线

```
1. quadraticCurveTo(cpx, cpy, x, y)
```

其中，cpx指控制点的x坐标；cpy指控制点的y坐标；x指结束点的x坐标；y指结束点的y坐标。

看看下面的这个图片：



- 开始点：moveTo(20,20)
- 控制点：quadraticCurveTo(20,100,200,20)
- 结束点：quadraticCurveTo(20,100,200,20)

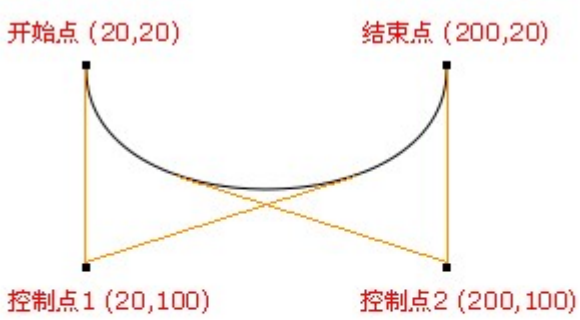
由上面的参数可以看出，二次贝赛尔曲线需要两个点：**第一个**点用于二次贝赛尔计算中的控制点，**第二个**点是曲线的结束点。曲线的开始点是当前路径中最后一个点。如果路径不存在，可使用beginPath()和moveTo()方法来定义。

三次贝塞尔曲线

```
1. bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)
```

其中，cp1x为第一个控制点的x坐标；cp1y为第一个控制点的y坐标；cp2x为第二个控制点的x坐标；cp2y为第二个控制点的y坐标；x为结束点的x坐标；y为结束点的y坐标。

再来看一张图片：



- 开始点：moveTo(20,20)
- 控制点 1：bezierCurveTo(20,100,200,100,200,20)
- 控制点 2：bezierCurveTo(20,100,200,100,200,20)
- 结束点：bezierCurveTo(20,100,200,100,200,20)

由上面可以看出，三次贝赛尔曲线需要三个点。前两个点用于三次贝赛尔计算中的控制点，第三个点是曲线结束的控制点。同上，曲线的开始点是当前路径中最后一个点。如果路径不存在，可使用beginPath()和moveTo()方法来定义。

canvas运用样式与颜色

色彩

```
1. fillStyle=color;
2. strokeStyle=color;
```

fillStyle用于设置填充的颜色，而strokeStyle用于设置图像轮廓的颜色。color可以是表示css颜色值的字符串、渐变对象或者图案对象。默认情况下，线条颜色和填充颜色都是黑色的。

透明度

globalAlpha

```
1. globalAlpha = transparencyValue
```

这个属性会影响canvas里 所有 在设置透明度后绘制的图形的透明度。它的取值区间是[0.0,1.0]。其中，0.0表示完全透明，1.0表示完全不透明。默认值是1.0。

rgba

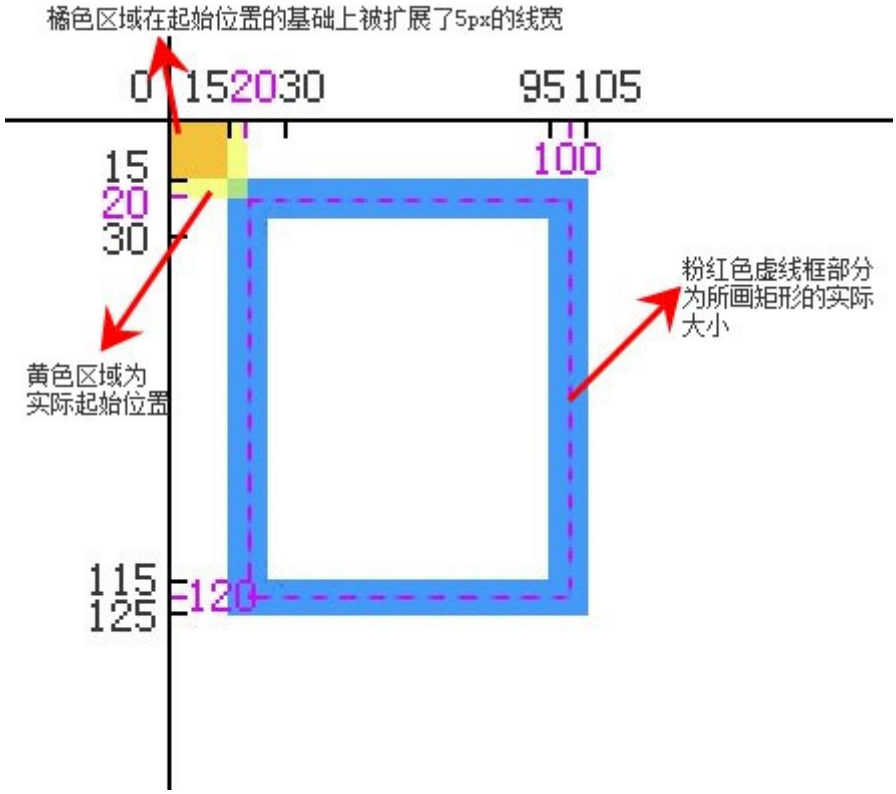
由于strokeStyle和fillStyle都接受符合css3规范的颜色值，所以还可以使用另一种设置透明度的方法，那就是rgba()方法，它的灵活性更大。

线型

lineWidth属性

lineWidth用于设置当前绘制的线的粗细。属性值必须为正数。默认值为1.0。

注意，这里的线宽是指给定路径的中心到两边的粗细。换句话说就是：在路径的两边各绘制线宽的一半。



lineCap属性

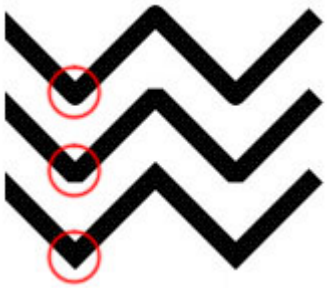
属性lineGap设置或返回了 线条末端 线帽的样式。

它有三个可选值，分别为：**butt**，这个值是默认的，表示向线条的每个末端添加平直的边缘；**round**，表示向线条的每个末端添加圆形线帽；**square**，表示向线条的每个末端添加正方形的线帽。

lineJoin属性

当一个路径包含了线段或曲线相交的交点时，lineJoin属性说明如何绘制这些交点。只有当绘制具有等宽线条的时候，这一属性的效果才能表现出来。

它的可选值有三个，分别是：**round**、**bevel**和**miter**。默认值是miter。
下图从上往下的三个小圆：



miterLimit属性

miterLimit 属性设置或返回最大斜接长度。
斜接长度指的是在两条线交汇处内角和外角之间的距离。

注意：只有当 lineJoin 属性为 “miter” 时，miterLimit 才有效。

边角的角度越小，斜接长度就会越大。为了避免斜接长度过长，我们可以使用 miterLimit 属性。

如果斜接长度超过 miterLimit 的值，边角会以 lineJoin 的 “bevel” 类型来显示（图解 3）：

语法：

```
1. miterLimit=number;
```

使用虚线

用 `setLineDash` 方法和 `lineDashOffset` 属性来制定虚线样式. setLineDash 方法接受一个数组，来指定线段与间隙的交替；lineDashOffset 属性设置起始偏移量.

```
1. setLineDash([5, 15])
```

一个跑马灯的例子：

```
var ctx = document.getElementById('canvas').getContext('2d');
var offset = 0;

function draw() {
  ctx.clearRect(0,0, canvas.width, canvas.height);
  ctx.setLineDash([4, 2]);
  ctx.lineDashOffset = -offset;
  ctx.strokeRect(10,10, 100, 100);
}

function march() {
  offset++;
  if (offset > 16) {
    offset = 0;
  }
  draw();
  setTimeout(march, 20);
}

march();
```

渐变

线性渐变

```
createLinearGradient(xStart, yStart, xEnd, yEnd)
```

线性渐变有四个参数，其中xStart,yStart为渐变起始点的坐标；xEnd,yEnd为渐变结束点的坐标。

径向渐变

```
createRadialGradient(xStart, yStart, radiusStart, xEnd, yEnd, radiusEnd)
```

径向渐变有六个参数，其中xStart,yStart为开始圆的圆心坐标；radiusStart为开始圆的半径；xEnd,yEnd为结束圆的圆心坐标；radiusEnd为结束圆的半径。

注意，以上两个方法都没有为渐变指定任何颜色。使用返回对象的`addColorStop()`来设置颜色。要使用一个渐变来勾勒线条或填充区域，只需要把`CanvasGradient`对象赋给`strokeStyle`属性或`fillStyle`属性即可。

addColorStop

```
addColorStop(offset,color)
```

offset是一个范围在0.0到1.0之间的浮点值，表示渐变开始点和结束点直接的一部分。offset值为0对应开始点，offset值为1对应结束点。

使用createPattern创建重复图片

语法：createPattern(image, type)
image是图片，type是图片的重复（repeat，repeat-x，repeat-y 和 no-repeat）
创建出一个 pattern 之后，赋给 fillStyle 或 strokeStyle 属性即可。
例如：

```
function draw() {
  var ctx = document.getElementById('canvas').getContext('2d');

  // 创建新 image 对象，用作图案
  var img = new Image();
  img.src = 'images/wallpaper.png';
  img.onload = function(){

    // 创建图案
    var ptrn = ctx.createPattern(img,'repeat');
    ctx.fillStyle = ptrn;
    ctx.fillRect(0,0,150,150);

  }
}
```

canvas填充规则

当我们用到 fill（或者 clip和isPointinPath ）你可以选择一个填充规则，该填充规则根据某处在路径的外面或者里面来决定该处是否被填充，这对于自己与自己路径相交或者路径被嵌套的时候是有用的。
两个可能值：nonzero（默认，一般没用） evenodd

绘制文本

两种绘制方式

`fillText(text,x,y[,maxWidth])`
在指定的(x,y)位置填充指定的文本，绘制的最大宽度是可选的.
`strokeText(text, x, y [, maxWidth])`
在指定的(x,y)位置绘制文本边框，绘制的最大宽度是可选的.

有样式的文本

`font`
当前我们用来绘制文本的样式. 这个字符串使用和 CSS font 属性相同的语法. 默认的字体是 10px sans-serif。
`textAlign`
文本对齐选项. 可选的值包括：start, end, left, right or center. 默认值是 start。
`textBaseline`
基线对齐选项. 可选的值包括：top, hanging, middle, alphabetic, ideographic, bottom。默认值是 alphabetic。
`direction` ：ltr,rtl

文本测量

`measureText()` 方法

```
1. function draw() {
2.   var ctx = document.getElementById('canvas').getContext('2d');
3.   var text = ctx.measureText("foo"); // TextMetrics object
4.   text.width; // 16;
5. }
```

阴影

shadowOffsetX
shadowOffsetY
shadowBlur
shadowColor

canvas应用图像

可以使用的图片源

- HTMLImageElemment：这些图片是由Image()函数构造出来的，或者任何的 `<img>` 元素
- HTMLVideoElement：用一个HTML的 `<video>` 元素作为你的图片源，可以从视频中抓取当前帧作为一个图像
- HTMLCanvasElement：可以使用另一个 `<canvas>` 元素作为你的图片源。
- ImageBitmap：这是一个高性能的位图，可以低延迟地绘制，它可以从上述的所有源以及其它几种源中生成。

drawImage

drawImage方法有三种状态`，下面为最简单的一种。

第一种

```
1. drawImage(img,x,y);
```

其中，img规定要使用的图像、画布或视频；x指在画布上放置图像的x坐标位置；y指在画布上放置的图像的y坐标位置。

注意：引用图片可是使用js的new Image()来创建。这个方法的缺点是，脚本会因为等待图片加载而暂停。可以使用onload事件，等图片加载完再执行相应的代码。

例如：

```
1. var img = new Image();   // Create new Image object
2. img.onload = function(){
3.   // 执行drawImage语句
4. }
5. img.src = 'myImage.png'; // Set source path
```

第二种：缩放图片

```
1. drawImage(img,x,y,width,height);
```

img,x,y三个参数和上面一致，新增的两个参数分别表示：width，在canvas中图片要显示的宽度；height，在canvas中图片要显示的高度。

第三种：剪切图片

```
1. drawImage(img,sx,sy,swidth,sheight,x,y,width,height);
```

它一共有九个参数，分别为：img，源图片对象；sx，开始剪切的x坐标位置；sy，开始剪切的y坐标位置；swidth，要剪切的宽度；sheight，要剪切的高度；x，剪切后在画布显示的x坐标位置；y，剪切后，在画布上显示的y坐标的位置；width，剪切后要显示的图片的宽度；height，剪切后要显示的图片的高度。

变形 transform

translate(x,y)
rotate(angle)
scale(x,y)

状态的保存与恢复

save()与restore()
save 和 restore 方法是用来保存和恢复 canvas 状态的，都没有参数。Canvas 的状态就是当前画面应用的所有样式和变形的一个快照。
执行的步骤：
Canvas状态存储在栈中，每当 `save()` 方法被调用后，当前的状态就被推送到栈中保存。一个绘画状态包括：

1. 当前应用的变形（即移动，旋转和缩放，见下）
2. strokeStyle, fillStyle, globalAlpha, lineWidth, lineCap, lineJoin, miterLimit, shadowOffsetX, shadowOffsetY, shadowBlur, shadowColor, globalCompositeOperation 的值
3. 当前的裁切路径（clipping path），会在下一节介绍

每一次调用 restore 方法，上一个保存的状态就从栈中弹出，所有设定都恢复。
典型的例子就是canvas绘制时钟

使用clip()裁切路径

除了之前的stroke与fill，还有一个是clip()，用法语其他两个一样，只是这个是用来裁切的。 注意：裁剪区域必须放在绘制区域上面 。

```
1.      c.arc(0,0,60,0,Math.PI*2,true);
2.      c.fill();
3.      c.clip();
4.      c.fillStyle = "red";
5.      c.fillRect(0,0,150,150);
```

canvas动画

动画的基本步骤

1. 清空canvas
2. 保存canvas状态
3. 重绘动画
4. 恢复canvas

操控动画

使用
window.setTnterval(function,delay)
window.setTimeote(function,delay)
一般是使用下面这个
window.requestAnimationFrame(callback)