logo.png

canvas

H5 web前端

[www.bufanui.com](http://www.bufanui.com)

# 一、 canvas

## 1.1 Canvas颜色样式和阴影

### 1.1.1 设置填充和描边的颜色

* fillStyle : 设置或返回用于填充绘画的颜色
* strokeStyle: 设置或返回用于笔触的颜色

### 1.1.2 设置阴影

* shadowColor ： 设置或返回用于阴影的颜色
* shadowBlur ： 设置或返回用于阴影的模糊级别,大于1的正整数，数值越高，模糊程度越大
* shadowOffsetX： 设置或返回阴影距形状的水平距离
* shadowOffsetY： 设置或返回阴影距形状的垂直距离

|  |
| --- |
| ctx.fillStyle = "red"  ctx.shadowColor = "rgba(0,0,0,.5)";  ctx.shadowBlur = 10;  ctx.shadowOffsetX = 10;  ctx.shadowOffsetY = 10;  ctx.fillRect(100, 100, 100, 100); |

**设置png图片的阴影，图片透明部分不会被投影。**

## 1.2 颜色渐变和背景图

### 1.2.1 线性渐变

* 线性渐变可以用于 矩形、圆形、文字等颜色样式
* 线性渐变是一个对象
* 语法：ctx.createLinearGradient(x0,y0,x1,y1); //参数：x0,y0起始坐标，x1,y1结束坐标

|  |
| --- |
| var cvs = document.querySelector('canvas')  var ctx = cvs.getContext("2d")  cvs.style.border = '1px solid red'  cvs.width = 600  cvs.height = 600  var clg = ctx.createLinearGradient(50, 50, 500, 50)  clg.addColorStop(0, 'blue')  // clg.addColorStop(.3, 'blue')  clg.addColorStop(1, 'yellow')  ctx.fillStyle = clg;  ctx.fillRect(50, 50, 100, 100)  ctx.fillRect(160, 50, 200, 100)  ctx.fillText('不凡学院', 360, 50) |

### 1.2.2径向渐变

* context.createRadialGradient(x0,y0,r0,x1,y1,r1);
* 参数详解：
  + x0: 渐变的开始圆的 x 坐标
  + y0: 渐变的开始圆的 y 坐标
  + r0: 开始圆的半径
  + x1: 渐变的结束圆的 x 坐标
  + y1: 渐变的结束圆的 y 坐标
  + r1: 结束圆的半径

|  |
| --- |
| var clg = ctx.createLinearGradient(50, 50, 500, 50)  clg.addColorStop(0, 'blue')  // clg.addColorStop(.3, 'blue')  clg.addColorStop(1, 'yellow')  var clg2 = ctx.createRadialGradient(100,200,50,200,200,150)  clg2.addColorStop(0,'red')  clg2.addColorStop(1,'green')  ctx.fillStyle = clg2;  ctx.fillRect(0, 0, 400, 400) |

### 3.2.3 绘制背景图

* ctx.createPattern(img,repeat) 方法在指定的方向内重复指定的元素了解
* 第一参数：设置平铺背景的图片，第二个背景平铺的方式。
  + image ： 规定要使用的图片、画布或视频元素。
  + repeat ： 默认。该模式在水平和垂直方向重复。
  + repeat-x ： 该模式只在水平方向重复。
  + repeat-y ： 该模式只在垂直方向重复。
  + no-repeat： 该模式只显示一次（不重复）。

|  |
| --- |
| var img = new Image();  img.src = 'img/hulk.jpg'  var pat=ctx.createPattern(img,"repeat");  ctx.rect(0,0,350,300);  ctx.fillStyle=pat;// 把背景图设置给填充的样式  ctx.fill(); |

## 3.3 变换

### 3.3.1 缩放

* scale() 方法缩放当前绘图，更大或更小
* 语法：context.scale(scalewidth,scaleheight)
  + scalewidth : 缩放当前绘图的宽度 (1=100%, 0.5=50%, 2=200%)
  + scaleheight : 缩放当前绘图的高度 (1=100%, 0.5=50%, 2=200%)

### 3.3.2 位移画布

* ctx.translate(x,y) 方法重新映射画布上的 (0,0) 位置
* 参数说明：
  + x： 添加到水平坐标（x）上的值
  + y： 添加到垂直坐标（y）上的值
* 发生位移后，相当于把画布的0,0坐标 更换到新的x,y的位置，所有绘制的新元素都被影响。

### 3.3.3 旋转

* context.rotate(angle); 方法旋转当前的绘图
* 注意参数是弧度（PI）
* 如需将角度转换为弧度，请使用 degrees\*Math.PI/180 公式进行计算。

## 3.3 绘制环境保存和还原

* ctx.save() 保存当前环境的状态
  + 可以把当前绘制环境进行保存到缓存中。
* ctx.restore() 返回之前保存过的路径状态和属性
  + 获取最近缓存的ctx
* 一般配合位移画布使用。

## 3.4 设置绘制环境的透明度

* context.globalAlpha=number;
* number:透明值。必须介于 0.0（完全透明） 与 1.0（不透明） 之间。
* 设置透明度是全局的透明度的样式。注意是全局的。

## 3.5 画布限定区域绘制

* ctx.clip(); 方法从原始画布中剪切任意形状和尺寸
* 一旦剪切了某个区域，则所有之后的绘图都会被限制在被剪切的区域内（不能访问画布上的其他区域）
* 一般配合绘制环境的保存和还原。

|  |
| --- |
| ctx.rect(0, 0, 200, 200)  ctx.save();  ctx.clip()  ctx.beginPath()  ctx.arc(200, 200, 100, 0, 2\*Math.PI, false)  ctx.fill()  ctx.restore();  ctx.beginPath();  ctx.arc(220, 220, 100, 0, 2\*Math.PI, false)  ctx.fill() |

## 3.6 画布保存base64编码内容

* 把canvas绘制的内容输出成base64内容。
* 语法：canvas.toDataURL(type, encoderOptions);
* 例如：canvas.toDataURL("image/jpg",1);
* 参数说明：
  + type，设置输出的类型，比如 image/png image/jpeg等
  + encoderOptions： 0-1之间的数字，用于标识输出图片的质量，1表示无损压缩，类型为： image/jpeg 或者image/webp才起作用。

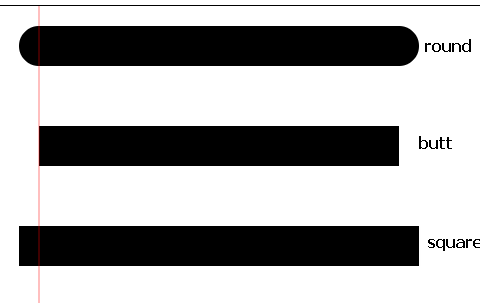
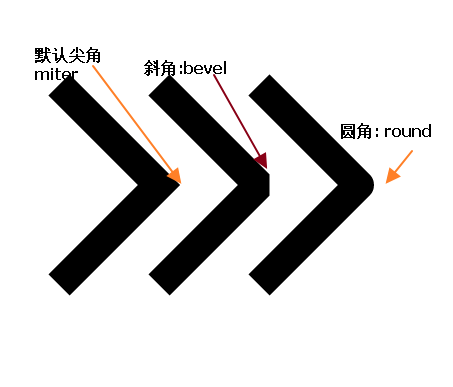
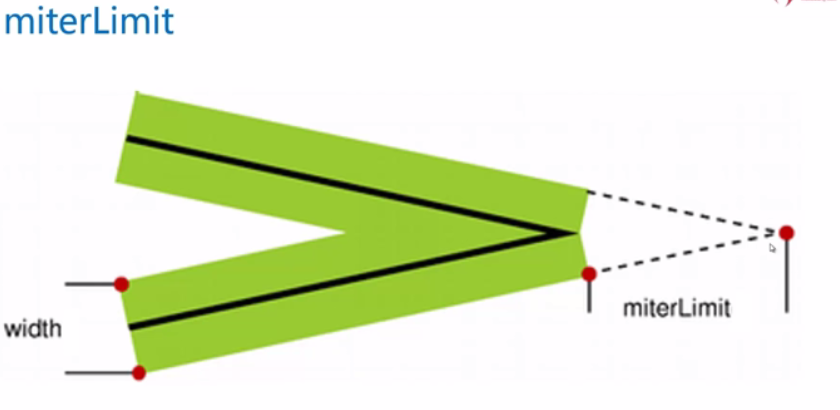
作用：document按行加载内容，把图片转换成base64写到css中，能使css加载完毕的时候，图片内容已经加载，避免dom元素加载完了，图片还没出来的情况。而且能提高访问速度。

## 3.7 画布渲染画布

* context.drawImage(img,x,y);
* img参数也可以是画布，也就是把一个画布整体的渲染到另外一个画布上。

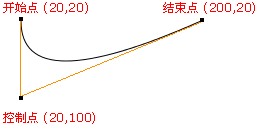
|  |
| --- |
| var cvs = document.querySelectorAll('canvas')  var cvs1 = cvs[0]  var cvs2 = cvs[1]  var ctx1 = cvs1.getContext("2d")  var ctx2 = cvs2.getContext("2d")  cvs1.width = 300  cvs1.height = 300  cvs2.width = 600  cvs2.height = 600  ctx1.fillRect(0, 0, 100, 100)  ctx2.drawImage(cvs1, 100, 100) |

## 3.8线条样式

* lineCap 设置或返回线条的结束端点(线头、线冒)样式
* butt ： 默认。向线条的每个末端添加平直的边缘。
* round ： 向线条的每个末端添加圆形线帽。
* square： 向线条的每个末端添加正方形线帽。
  + 
* lineJoin 设置或返回两条线相交时，所创建的拐角类型
  + bevel: 创建斜角。
  + round: 创建圆角。
  + miter: 默认。创建尖角
  + 
* lineWidth 设置或返回当前的线条宽度
* miterLimit 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。
  + 一般用默认值：10就可以了。除非需要特别长的尖角时，使用此属性。 

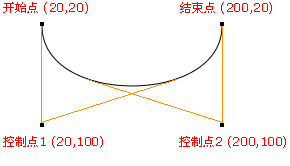
## 3.9 贝塞尔曲线

### 3.9.1 绘制一条二次方曲线。

* 语法： context.quadraticCurveTo(cpx,cpy,x,y);
* 参数：
  + cpx： 贝塞尔控制点的 x 坐标
  + cpy： 贝塞尔控制点的 y 坐标
  + x ： 结束点的 x 坐标
  + y ： 结束点的 y 坐标   
    

### 3.9.2绘制一条三次贝塞尔曲线

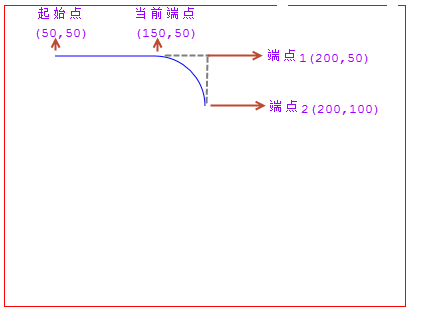
* 语法：context.bezierCurveTo(cp1x,cp1y,cp2x,cp2y,x,y);
* 提示：三次贝塞尔曲线需要三个点。前两个点是用于三次贝塞尔计算中的控制点，第三个点是曲线的结束点。曲线的开始点是当前路径中最后一个点。如果路径不存在，那么请使用 beginPath() 和 moveTo() 方法来定义开始点。
* 参数说明：
  + cp1x： 第一个贝塞尔控制点的 x 坐标
  + cp1y： 第一个贝塞尔控制点的 y 坐标
  + cp2x： 第二个贝塞尔控制点的 x 坐标
  + cp2y： 第二个贝塞尔控制点的 y 坐标
  + x: 结束点的 x 坐标
  + y: 结束点的 y 坐标

* 

## 3.10创建两条切线的弧（知道有）

* 在画布上创建介于当前起点和两个点形成的夹角的切线之间的弧
* 语法： context.arcTo(x1,y1,x2,y2,r); //类比：css3中的圆角。
* 例如： ctx.arcTo(240, 100, 240, 110, 40);
* 参数：
  + x1: 弧的端点1的 x 坐标
  + y1: 弧的端点1的 y 坐标
  + x2: 弧的端点2(终点)的 x 坐标
  + y2: 弧的端点2(终点)的 y 坐标
  + r : 弧的半径

|  |
| --- |
| ctx.moveTo(100,100);  ctx.lineTo(200,100);  //context.arcTo(x1,y1,x2,y2,r); //类比：css3中的圆角。  ctx.arcTo(240, 100, 240, 104, 40);  ctx.lineTo(240, 300);  ctx.stroke(); |



## 3.11了解判断点是否在路径中

|  |
| --- |
| ctx.strokeRect(50, 50, 200, 200)  alert(ctx.isPointInPath(110,100))  // ctx.stroke(); |

思考：封装一个用canvas画矩形的对象，用面向对象封装，要求能方便的画矩形的长宽位置，颜色，而且能够发生变换。