# 【canvas】

## 《2.6.5 面向对象基础复习补充：》

创建对象的方式：

\* var o = { name: '123', age: 18 }; //字面量方式创建

\* var o = new Object(); //通过new的方式创建

\* var o = new Persion(); //通过类的构造函数创建

JS中对象的属性创建方式

\* json的方式： var o = { age: 19 };

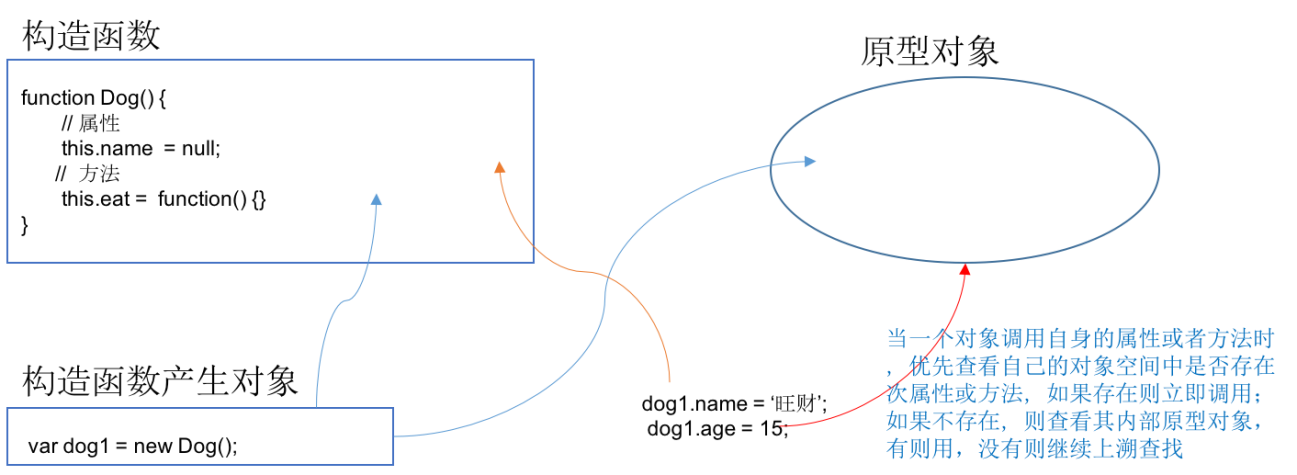
\* 直接添加属性：var o = {}; o.age = 19;//太分散了，不利于管理

\* 由于js动态语言的特性，如果属性不存在的时候，直接添加属性。

\* 构造函数添加属性

\* 原型添加公共的属性

JS的构造函数的原型 构造函数的原型就是：构造对象的模板，构造函数原型里面的所有的属性和方法都会共享给所有的 构造函数构造出来的所有实例。



# 【canvas进阶】

## 《3.1 Canvas颜色样式和阴影》

### 《3.1.1 设置填充和描边的颜色》（掌握）

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

以上两个值都可以接受颜色名,16进制数据，rgb值，甚至rgba. 一般先进行设置样式然后进行绘制。

例如：

ctx.strokeStyle = "red";

ctx.strokeStyle = "#ccc";

ctx.strokeStyle = "rgb(255,0,0)";

ctx.strokeStyle = "rgba(255,0,0,6)";

### 《3.1.2 设置阴影》（了解，少用，性能差）

类比于CSS3的阴影。

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 = "rgba(255,0,0, .9)"

ctx.shadowColor = "teal";

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。

## 《3.2 复杂样式》（了解）

### 《3.2.1 创建线性渐变的样式》（了解）

* 一般不用，都是用图片代替，canvas绘制图片效率更高。
* 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

例如：

//创建线性渐变的对象，

var grd=ctx.createLinearGradient(0,0,170,0);

grd.addColorStop(0,"black"); //添加一个渐变颜色，第一个参数介于 0.0 与 1.0 之间的值，表示渐变中开始与结束之间的位置。

grd.addColorStop(1,"white"); //添加一个渐变颜色

ctx.fillStyle =grd; //关键点，把渐变设置到 填充的样式

### 《3.2.2 设置圆形渐变（径向渐变）》 了解

创建放射状/圆形渐变对象。可以填充文本、形状等

context.createRadialGradient(x0,y0,r0,x1,y1,r1);

radial 半径的；放射状的；光线的；光线状的 英 ['reɪdɪəl] 美 ['redɪəl]

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 rlg = ctx.createRadialGradient(300,300,10,300,300,200);

rlg.addColorStop(0, 'teal'); //添加一个渐变颜色

rlg.addColorStop(.4, 'navy');

rlg.addColorStop(1, 'purple');

ctx.fillStyle = rlg;//设置 填充样式为延续渐变的样式

ctx.fillRect(100, 100, 500, 500);

### 《3.2.3 绘制背景图》（了解）

ctx.createPattern(img,repeat) 方法在指定的方向内重复指定的元素 了解

pattern：n. 模式；图案；样品 英 ['pæt(ə)n] 美 ['pætɚn]

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 ctx=c.getContext("2d");

var img=document.getElementById("lamp");

var pat=ctx.createPattern(img,"repeat");

ctx.rect(0,0,150,100);

ctx.fillStyle=pat;// 把背景图设置给填充的样式

ctx.fill();

## 《3.3 变换》※（重点）

### 《3.3.1 缩放》（重点）

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%, 依次类推)

scaleheight : 缩放当前绘图的高度 (1=100%, 0.5=50%, 2=200%, etc.) +注意：缩放的是整个画布，缩放后，继续绘制的图形会被放大或缩小。

### 《3.3.2 位移画布》（重点）

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。

位移画布一般配合缩放和旋转等。

### 《3.3.3 旋转》（重点）

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》（重要）

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》（了解）

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》（了解）

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。

## 《3.6 画布保存base64编码内容》（重要）

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。

案例1：

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

var dataURL = canvas.toDataURL();

console.log(dataURL);

// "data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAAUAAAAFCAYAAACNby

// blAAAADElEQVQImWNgoBMAAABpAAFEI8ARAAAAAElFTkSuQmCC"

var img = document.querySelector("#img-demo");//拿到图片的dom对象

img.src = canvas.toDataURL("image/png"); //将画布的内容给图片标签显示

## 《3.7 画布渲染画布》（重要）

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 canvas1 = document.querySelector('#cavsElem1');

var canvas2 = document.querySelector('#cavsElem2');

var ctx1 = canvas1.getContext('2d');

var ctx2 = canvas2.getContext('2d');

ctx1.fillRect(20, 20, 40, 40); //在第一个画布上绘制矩形

ctx2.drawImage(canvas1, 10, 10); //将第一个画布整体绘制到第二个画布上

## 《3.8 了解：线条样式》（了解）

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。

翻译.：屁股；烟头；笑柄；靶垛；粗大的一端 英 [bʌt] 美 [bʌt]

round： 向线条的每个末端添加圆形线帽。

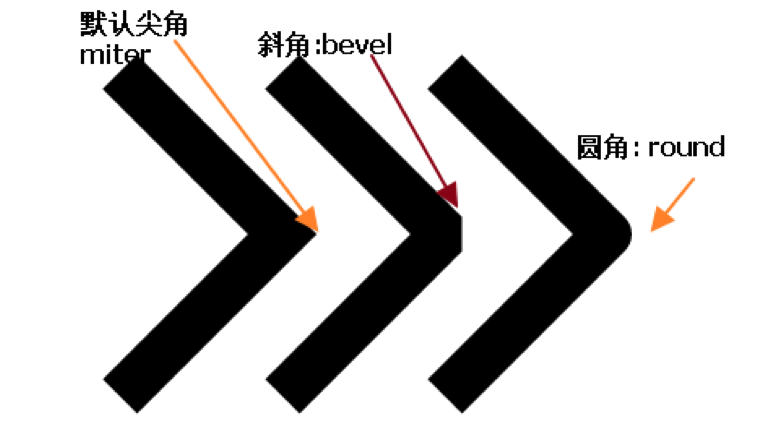
square： 向线条的每个末端添加正方形线帽。 

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。

翻译. 斜角；斜面；[测] 斜角规 英 ['bev(ə)l] 美 ['bɛvl]

round: 创建圆角。

miter: 默认。创建尖角 

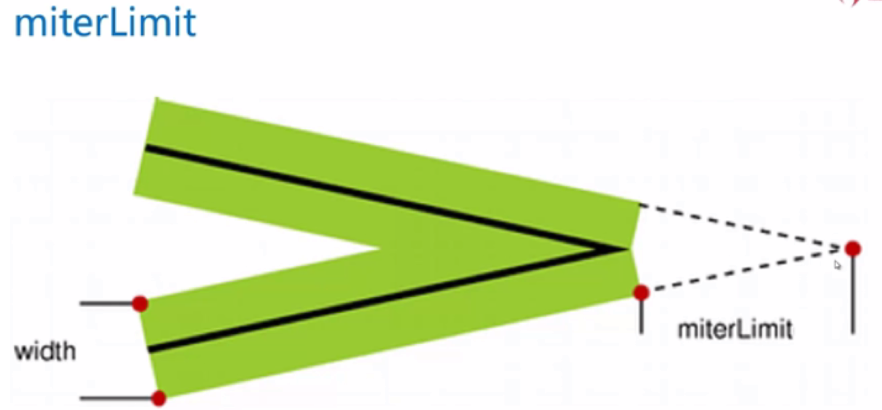
lineWidth 设置或返回当前的线条宽度

miterLimit 设置或返回最大斜接长度

意思: 斜接 英 ['maɪtə]

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 了解贝塞尔曲线》（知道有）

### 《3.9.1 绘制一条二次方曲线》

微软的画图板中的曲线的颜色。

quadratic：二次方的意思， 英 [kwɒ'drætɪk] 美 [kwɑ'drætɪk]

Curve：曲线的意思， 英 [kɜːv] 美 [kɝv]

语法：

context.quadraticCurveTo(cpx,cpy,x,y);

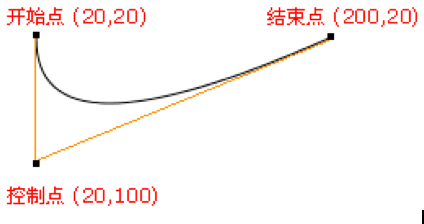
参数：

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



ctx.beginPath();

ctx.moveTo(20,20);

//绘制2次方曲线，贝赛尔曲线

ctx.quadraticCurveTo(20,100,200,20);

ctx.stroke();

### 《3.9.2 绘制贝塞尔曲线》（知道有）

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

//绘制复杂的贝塞尔曲线

ctx.beginPath();

ctx.moveTo(400,400);

//参数说明：context.bezierCurveTo(cp1x,cp1y,cp2x,cp2y,x,y);

// 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

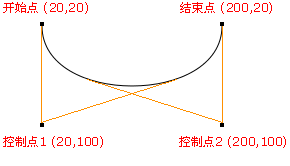
// cp2y： 第二个贝塞尔控制点的 y 坐标

// x: 结束点的 x 坐标

// y: 结束点的 y 坐标

ctx.bezierCurveTo(500, 200, 600, 600, 700, 300);

ctx.stroke();



## 《3.10了解创建两条切线的弧》（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

//代码demo：

ctx.beginPath();

ctx.moveTo(100,100);

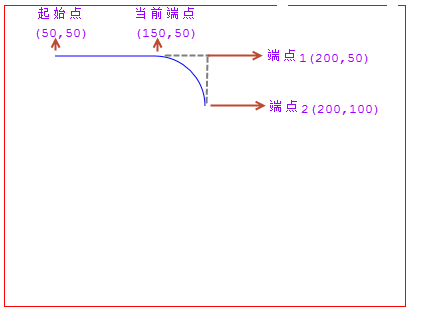
ctx.lineTo(200,100);

//context.arcTo(x1,y1,x2,y2,r); //类比：css3中的圆角。

ctx.arcTo(240, 100, 240, 110, 40);

ctx.lineTo(240, 300);

ctx.stroke();



## 《3.11了解判断点是否在路径中》（知道有）

context.isPointInPath(x,y);

//isPointInPath() 方法返回 true，如果指定的点位于当前路径中；否则返回 false。

//判断x,y坐标的点是否在当前的路径中。

## 《3.12了解文本宽度计算》（知道有）

context.measureText(text).width;

## 《3.13 其他》

如果以后做canvas游戏方向开发深入学习可以扩展以下内容：

* setTransform() 将当前转换重置为单位矩阵。然后运行 transform()
* transform() 替换绘图的当前转换矩阵
* globalCompositeOperation 设置或返回新图像如何绘制到已有的图像上
* 像素操作