HTML5

画布

目录

一、 canvas简介 5

1.1 什么是canvas？（了解） 5

1.2 canvas主要应用的领域（了解） 5

1.3我们课程的目标 6

1.4我们课程的案例和项目演示 6

二、 canvas绘图基础 6

2.1 Canvas标签 6

2.1.1 canvas标签语法和属性 （重点） 6

2.1.2 浏览器不兼容处理（重点） 7

2.2 canvas绘图上下文context 7

2.2.1 Context：Canvas的上下文、绘制环境。（重点掌握） 7

2.3 基本的绘制路径（重点） 7

2.3.1 canvas坐标系 7

2.3.2 设置绘制起点(moveTo) 8

2.3.3 绘制直线(lineTo) 8

2.3.4 路径开始和闭合 8

2.3.5 描边(stroke) 9

2.3.6 canvas绘制的基本步骤： 9

2.3.7 填充(fill) 10

2.3.8 快速创建矩形rect()方法 11

2.3.9 快速创建描边矩形和填充矩形 12

2.3.10 清除矩形(clearRect) 12

2.4 绘制圆形（arc) 12

2.5 绘制文字（会使用就可以了） 13

2.5.2 上下文绘制文字方法 15

2.5.3 案例07文字绘制.html 16

2.6 绘制图片（drawImage） （重点） 16

2.6.1 基本绘制图片的方式 16

2.6.2 在画布上绘制图像，并规定图像的宽度和高度 16

2.6.3 图片裁剪，并在画布上定位被剪切的部分 16

2.6.4 用JavaScript创建img对象 16

2.6.5 面向对象基础复习补充： 17

三、 canvas进阶 19

3.1 Canvas颜色样式和阴影 19

3.1.1 设置填充和描边的颜色（掌握） 19

3.1.2 设置阴影（了解，少用，性能差） 19

3.2 复杂样式（了解） 20

3.2.1 创建线性渐变的样式（了解） 20

3.2.2 设置圆形渐变（径向渐变） 了解 20

3.2.3 绘制背景图（了解） 21

3.3 变换（重点） 22

3.3.1 缩放（重点） 22

3.3.2 位移画布（重点） 22

3.3.3 旋转（重点） 22

3.4 绘制环境保存和还原（重要） 23

3.5 设置绘制环境的透明度（了解） 23

3.6 画布限定区域绘制（了解） 23

3.7 画布保存base64编码内容（重要） 23

3.8 画布渲染画布（重要） 24

3.9 了解：线条样式（了解） 24

3.10 了解贝塞尔曲线（知道有） 26

3.10.1 绘制一条二次方曲线。 26

3.10.2 绘制贝塞尔曲线（知道有） 27

3.11了解创建两条切线的弧（知道有） 28

3.12了解判断点是否在路径中（知道有） 29

3.13了解文本宽度计算（知道有） 29

3.14 如果以后做canvas游戏方向开发深入学习可以扩展内以下容 29

# 一、 canvas简介

## 1.1 什么是canvas？（了解）

canvas是HTML5提供的一种新标签 <canvas></canvas>

英 ['kænvəs] 美 ['kænvəs] 帆布 画布

Canvas是一个矩形区域的画布，可以用JavaScript在上面绘画。控制其每一个像素。

canvas 标签使用 JavaScript 在网页上绘制图像，本身不具备绘图功能。

canvas 拥有多种绘制路径、矩形、圆形、字符以及添加图像的方法。

HTML5之前的web页面只能用一些固定样式的标签：比如p、div、h1等

## 1.2 canvas主要应用的领域（了解）

1.游戏：canvas在基于Web的图像显示方面比Flash更加立体、更加精巧，canvas游戏在流畅度和跨平台方面更牛。

25 超棒的 HTML5 Canvas 游戏

2.可视化数据.数据图表话，比如:百度的echart

3.banner广告：Flash曾经辉煌的时代，智能手机还未曾出现。现在以及未来的智能机时代，HTML5技术能够在banner广告上发挥巨大作用，用Canvas实现动态的广告效果再合适不过。

4.未来=> 模拟器：无论从视觉效果还是核心功能方面来说，模拟器产品可以完全由JavaScript来实现。

5.未来=> 远程计算机控制：Canvas可以让开发者更好地实现基于Web的数据传输，构建一个完美的可视化控制界面。

6.未来=> 图形编辑器：Photoshop图形编辑器将能够100%基于Web实现。

7.其他可嵌入网站的内容(多用于活动页面、特效)：类似图表、音频、视频，还有许多元素能够更好地与Web融合，并且不需要任何插件。

8.完整的canvas移动化应用

## 1.3我们课程的目标

我们不是主要做游戏开发的

要求必须会做基本的用canvas绘制的特效页面

会用canvas做一些简单的广告、活动页面

## 1.4我们课程的案例和项目演示

canvas的标准：

最新标准：http://www.w3.org/TR/2dcontext/

稳定版本的标准：http://www.w3.org/TR/2013/CR-2dcontext-20130806/

目前来说，标准还在完善中。先用早期的api足够完成所有的应用

# 二、 canvas绘图基础

## 2.1 Canvas标签

### 2.1.1 canvas标签语法和属性 （重点）

canvas：画布油布的意思 ==英 ['kænvəs] 美 ['kænvəs] ==

标签名canvas，需要进行闭合。就是一普通的html标签。

可以设置width和height属性，但是属性值单位必须是px，否则忽略。

width和hegiht：默认300\*150像素

注意：

不要用CSS控制它的宽和高,会走出图片拉伸，

重新设置canvas标签的宽高属性会让画布擦除所有的内容。

可以给canvas画布设置背景色

### 2.1.2 浏览器不兼容处理（重点）

ie9以上才支持canvas, 其他chrome、ff、苹果浏览器等都支持

只要浏览器兼容canvas，那么就会支持绝大部分api(个别最新api除外)

移动端的兼容情况非常理想，基本上随便使用

2d的支持的都非常好，3d（webgl）ie11才支持，其他都支持

如果浏览器不兼容，最好进行友好提示

例如： <canvas id="cavsElem"> 你的浏览器不支持canvas，请升级浏览器.浏览器不支持，显示此行文本 </canvas>

浏览器不兼容，可以使用flash等手段进行优雅降级

## 2.2 canvas绘图上下文context

### 2.2.1 Context：Canvas的上下文、绘制环境。（重点掌握）

上下文：上知天文，下知地理。是所有的绘制操作api的入口或者集合。

Canvas自身无法绘制任何内容。Canvas的绘图是使用JavaScript操作的。

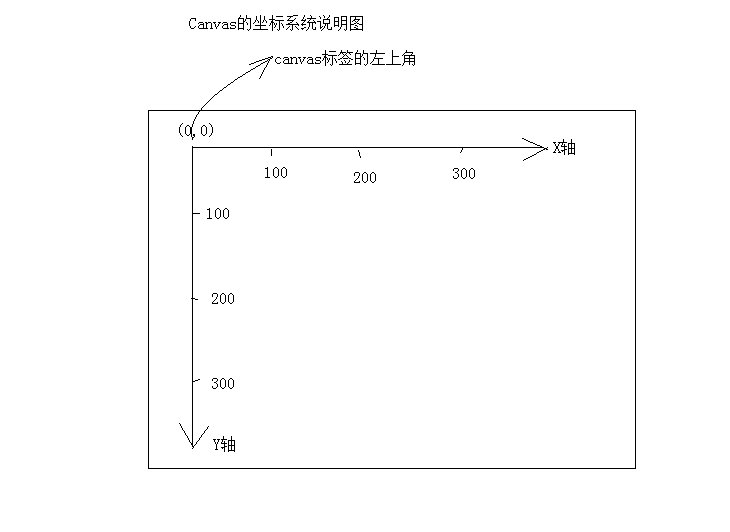
Context对象就是JavaScript操作Canvas的接口。 \*使用[CanvasElement].getContext(‘2d’)来获取2D绘图上下文。

var canvas = document.getElementById( 'cavsElem' ); //获得画布 var ctx = canvas.getContext( '2d' );//注意：2d小写， 3d：webgl

## 2.3 基本的绘制路径（重点）

### 2.3.1 canvas坐标系

canvas坐标系，从最左上角0,0开始。x向右增大， y向下增大



### 2.3.2 设置绘制起点(moveTo)

\* 语法：ctx.moveTo(x, y);

\* 解释：设置上下文绘制路径的起点。相当于移动画笔到某个位置

\* 参数：x,y 都是相对于 canvas盒子的最左上角。

\* 注意：\*\*绘制线段前必须先设置起点。\*\*

### 2.3.3 绘制直线(lineTo)

\* 语法：ctx.lineTo(x, y);

\* 解释：从x,y的位置绘制一条直线到起点或者上一个线头点。

\* 参数：x,y 线头点坐标。

### 2.3.4 路径开始和闭合

\* 开始路径：ctx.beginPath();

\* 闭合路径：ctx.closePath();

\* 解释：如果是绘制不同状态的线段或者形状，必须使用开始新路径的方法把不同的绘制操作隔开。闭合路径会自动把最后的线头和开始的线头连在一起。

\* beginPath: 核心的作用是将 不同绘制的形状进行隔离，

每次执行此方法，表示重新绘制一个路径,跟之前的绘制的墨迹可以进行分开样式设置和管理。

### 2.3.5 描边(stroke)

\* 语法：ctx.stroke();

\* 解释：根据路径绘制线。路径只是草稿，真正绘制线必须执行stroke

\* stroke: （用笔等）画；轻抚；轻挪；敲击；划尾桨；划掉；（打字时）击打键盘

英 [strəʊk] 美 [strok]

### 2.3.6 canvas绘制的基本步骤：

第一步：获得上下文 =>canvasElem.getContext('2d');

第二步：开始路径规划 =>ctx.beginPath()

第三步：移动起始点 =>ctx.moveTo(x, y)

第四步：绘制线(矩形、圆形、图片...) =>ctx.lineTo(x, y)

第五步：闭合路径 =>ctx.closePath();

第六步：绘制描边 =>ctx.stroke();

html部分：

<canvas id="cavsElem">

你的浏览器不支持canvas，请升级浏览器

</canvas>

javascript部分：

//===============基本绘制api====================

//获得画布

var canvas = document.querySelector('#cavsElem');

var ctx = canvas.getContext('2d'); //获得上下文

canvas.width = 900; //设置标签的属性宽高

canvas.height = 600; //千万不要用 canvas.style.height

canvas.style.border = "1px solid #000";

//绘制三角形

ctx.beginPath(); //开始路径

ctx.moveTo(100,100); //三角形，左顶点

ctx.lineTo(300, 100); //右顶点

ctx.lineTo(300, 300); //底部的点

ctx.closePath(); //结束路径

ctx.stroke(); //描边路径

综合案例：02绘制定位表格.html

综合案例：03画画板.html

### 2.3.7 填充(fill)

\* 语法：ctx.fill();

\* 解释：填充，是将闭合的路径的内容填充具体的颜色。默认黑色。

\* 注意：交叉路径的填充问题，“非零环绕原则”，顺逆时针穿插次数决定是否填充。

以下是非0环绕原则的原理：（了解即可，非常少会用到复杂的路径）

非零环绕规则：对于路径中指定范围区域，从该区域内部画一条足够长的线段，使此线段的完全落在路径范围之外。

非零环绕规则计数器：

然后，将计数器初始化为0，每当这个线段与路径上的直线或曲线相交时，就改变计数器的值，如果是与路径顺时针相交时，那么计数器就加1， 如果是与路径逆时针相交时，那么计数器就减1.

如果计数器始终不为0，那么此区域就在路径范围里面，在调用fill()方法时，浏览器就会对其进行填充。如果最终值是0，那么此区域就不在路径范围内，浏览器就不会对其进行填充。

从上图中看

第一条线段：根据非零环绕规则，这条直线只经过路径一次且路径是逆时针方向，那么计数器为-1;根据浏览器对计数器的计算原理得出，当调用fill()方法时浏览器会填充此区域。

第二条线段：根据非零环绕规则，这条直线经过路径二次且路径都是逆时针方向，那么计数器为-2;根据浏览器对计数器的计算原理得出，当调用fill()方法时浏览器会填充此区域。

第三条线段：根据非零环绕规则，这条直线经过路径二次;第一次经过的路径是逆时针方向，计数器则为-1; 第二次经过的路径是顺时针方向，计数器为1;原因计数器的最终值为0-1+1 = 0;所以根据浏览器对计数器的计算原理得出，当调用fill()方法时浏览器不会填充此区域。

\* 案例： 04填充矩形.html



### 2.3.8 快速创建矩形rect()方法

\* 语法：ctx.rect(x, y, width, height);

\* 解释：x, y是矩形左上角坐标， width和height都是以像素计

\* rect方法只是规划了矩形的路径，并没有填充和描边。

\* 改造案例：04填充矩形.html

\*rect: abbr. 矩形（rectangular）；收据（receipt）

### 2.3.9 快速创建描边矩形和填充矩形

\* 语法： ctx.strokeRect(x, y, width, height);

- 参数跟2.3.8相同，注意此方法绘制完路径后立即进行stroke绘制

\* 语法：ctx.fillRect(x, y, width, height);

- 参数跟2.3.8相同， 此方法执行完成后。立即对当前矩形进行fill填充。

### 2.3.10 清除矩形(clearRect)

\* 语法：ctx.clearRect(x, y, width, hegiht);

\* 解释：清除某个矩形内的绘制的内容，相当于橡皮擦。

## 2.4 绘制圆形（arc)

概述：arc() 方法创建弧/曲线（用于创建圆或部分圆）。

语法：ctx.arc(x,y,r,sAngle,eAngle,counterclockwise);

arc: 弧（度）弧形物；天穹 英 [ɑːk] 美 [ɑrk]

counter 反击，还击；反向移动，对着干；反驳，回答 ['kaʊntə] 美 ['kaʊntɚ]

解释：

x,y：圆心坐标。

r：半径大小。

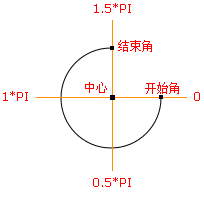
sAngle:绘制开始的角度。 圆心到最右边点是0度，顺时针方向弧度增大。

eAngel:结束的角度，注意是弧度。π

counterclockwise：是否是逆时针。true是逆时针，false：顺时针

弧度和角度的转换公式： rad = deg\*Math.PI/180;

在Math提供的方法中sin、cos等都使用的弧度



案例：05绘制圆形.html

案例：06绘制饼状图.html

## 2.5 绘制文字（会使用就可以了）

#### 2.5.1 绘制上下文的文字属性 （有印象就行了）

font 设置或返回文本内容的当前字体属性

font 属性使用的语法与 CSS font 属性相同。

例如：ctx.font = "18px '微软雅黑'";

textAlign 设置或返回文本内容的当前对齐方式

start : 默认。文本在指定的位置开始。

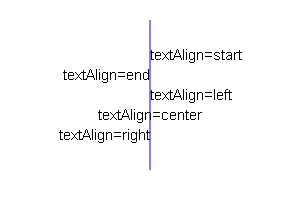
end : 文本在指定的位置结束。

center: 文本的中心被放置在指定的位置。

left : 文本左对齐。

right : 文本右对齐。

例如：ctx.textAlign = 'left';



textBaseline 设置或返回在绘制文本时使用的当前文本基线

alphabetic ： 默认。文本基线是普通的字母基线。

top ： 文本基线是 em 方框的顶端。。

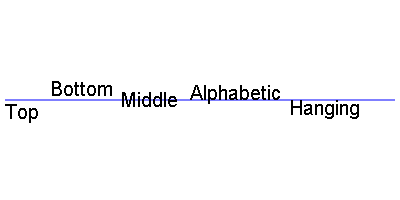
hanging ： 文本基线是悬挂基线。

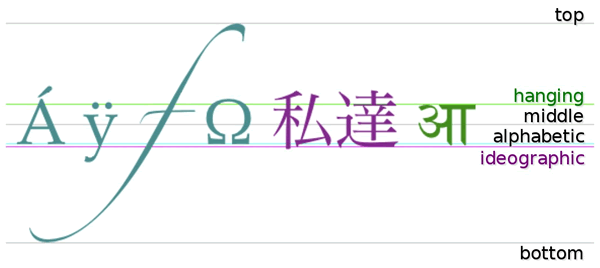
middle ： 文本基线是 em 方框的正中。

ideographic： 文本基线是em基线。

bottom ： 文本基线是 em 方框的底端。

例如： ctx.textBaseline = 'top'; 单词: alphabetic: 字母的；照字母次序的 [,ælfə'bɛtɪk] ideographic：表意的；表意字构成的 英 [,ɪdɪəʊ'ɡræfɪk] 美 [,ɪdɪə'græfɪk] 设置文字为主





### 2.5.2 上下文绘制文字方法

\* ctx.fillText() 在画布上绘制“被填充的”文本

\* ctx.strokeText() 在画布上绘制文本（无填充）

\* ctx.measureText() 返回包含指定文本宽度的对象

\* 单词：measure 测量；估量；权衡 英 ['meʒə] 美 ['mɛʒɚ]

//综合案例代码：

ctx.moveTo( 300, 300 );

ctx.fillStyle = "purple"; //设置填充颜色为紫色

ctx.font = '20px "微软雅黑"'; //设置字体

ctx.textBaseline = "bottom"; //设置字体底线对齐绘制基线

ctx.textAlign = "left"; //设置字体对齐的方式

//ctx.strokeText( "left", 450, 400 );

ctx.fillText( "Top-g", 100, 300 ); //填充文字

### 2.5.3 案例07文字绘制.html

## 2.6 绘制图片（drawImage） （重点）

### 2.6.1 基本绘制图片的方式

context.drawImage(img,x,y);

参数说明： x,y 绘制图片左上角的坐标， img是绘制图片的dom对象。

### 2.6.2 在画布上绘制图像，并规定图像的宽度和高度

context.drawImage(img,x,y,width,height);

参数说明：width 绘制图片的宽度， height：绘制图片的高度

如果指定宽高，最好成比例，不然图片会被拉伸</em>

等比公式： toH = Height \* toW / Width; //等比

设置高 = 原高度 \* 设置宽/ 原宽度;

### 2.6.3 图片裁剪，并在画布上定位被剪切的部分

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

参数说明：

sx,sy 裁剪的左上角坐标，

swidth：裁剪图片的高度。 sheight:裁剪的高度

其他同上

### 2.6.4 用JavaScript创建img对象

第一种方式：

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

第二种方式：

var img = new Image();//这个就是 img标签的dom对象

img.src = "imgs/arc.gif";

img.alt = "文本信息";

img.onload = function() {

//图片加载完成后，执行此方法

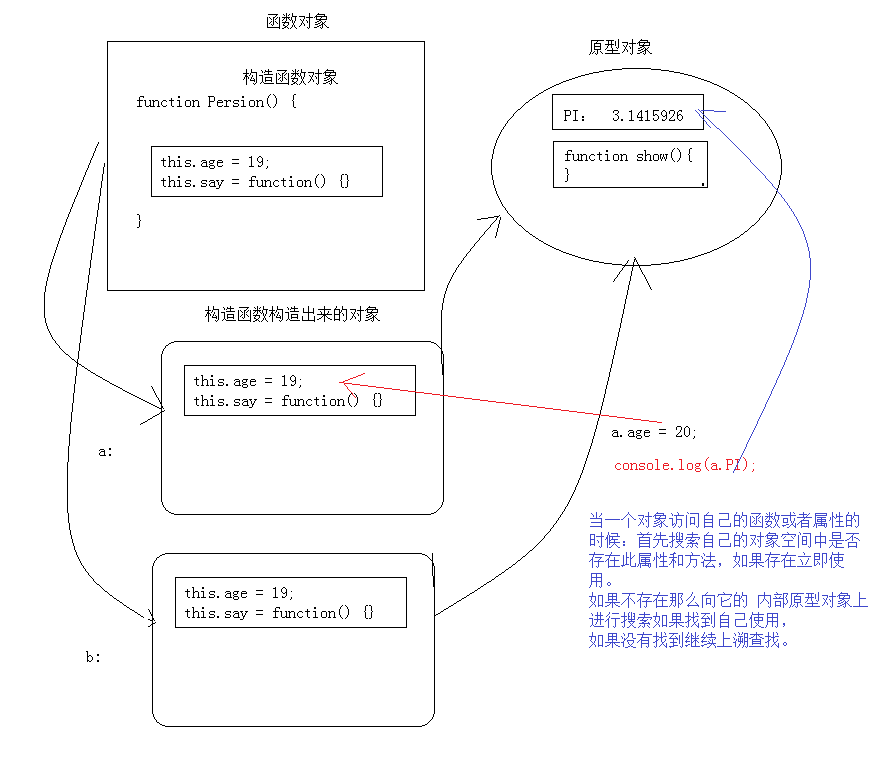
}

### 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 }; //json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。



案例： 08绘制图片.html

案例： 09绘制图片裁剪.html

案例： 10序列帧动画.html

案例： 11面向对象版本的动画帧.html 重点

# 三、 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);

例如：

案例： 12设置box盒子阴影.html

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 案例13设置线性渐变.html

### 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);

案例14圆形渐变.html

### 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();

案例15背景图填充.html

## 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。

案例16缩放案例.html

### 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。

案例： 17位移画布.html

### 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。

案例：18旋转画布.html

## 3.4 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。

案例： 19矩形旋转案例.html

## 3.5 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。

## 3.6 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。

## 3.7 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

## 3.8 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

## 3.9 了解：线条样式（了解）

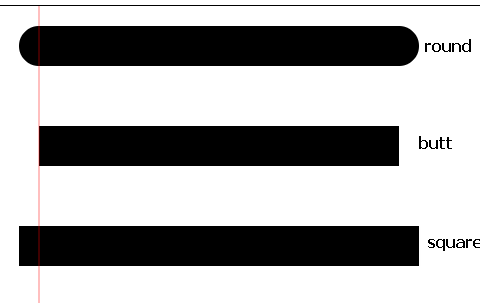
+ lineCap 设置或返回线条的结束端点(线头、线冒)样式

\* 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。



参考：23线的样式.html

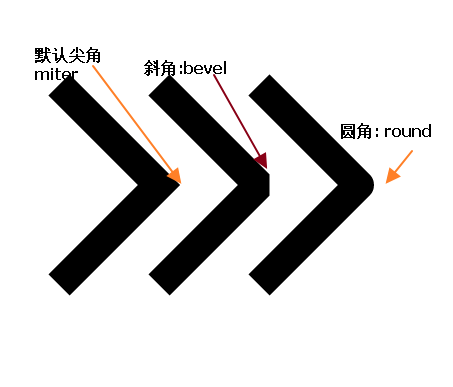
+ lineJoin 设置或返回两条线相交时，所创建的拐角类型

\* 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



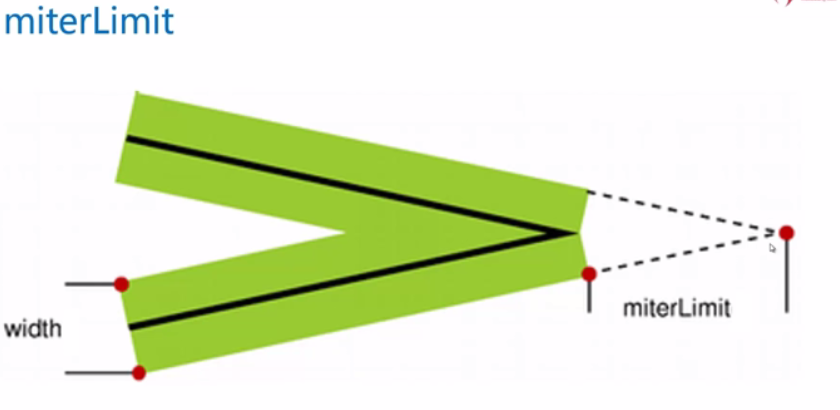
+ lineWidth 设置或返回当前的线条宽度

+ 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。



## 3.10 了解贝塞尔曲线（知道有）

### 3.10.1 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);

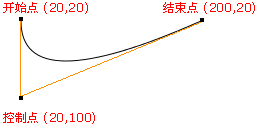
+ 参数：

\* 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();

### 3.10.2 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

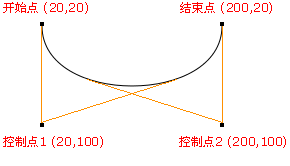
// cp2y： 第二个贝塞尔控制点的 y 坐标

// 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();

+ 案例：25绘制贝塞尔曲线.html



## 3.11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();

## 3.12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。

## 3.13了解文本宽度计算（知道有）

context.measureText(text).width;

## 3.14 如果以后做canvas游戏方向开发深入学习可以扩展内以下容

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