logo.png

canvas

H5 web前端

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

# 一、canvas

## 1.1 简介

是HTML5提供的一种新标签

* 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。

## canvas主要应用

## 1.3 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/

# 二、canvas基础

sublime配置canvas插件

|  |
| --- |
| 推荐：安装插件：AndyJS2github地址： https://github.com/malun666/AndyJS2 |

## 2.1 Canvas标签

### 2.1.1语法

* canvas：画布油布的意思 ==英 ['kænvəs] 美 ['kænvəs] ==
* 标签名canvas，需要进行闭合。
* 可以设置width和height属性，单位px。
* width和hegiht：默认300\*150像素
* 注意：
  + 不要用CSS控制它的宽和高,会出现图片拉伸，
  + 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 兼容处理

* ie9以上才支持canvas
* 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

|  |
| --- |
| <canvas id="c1 ">  你的浏览器不支持canvas，请升级浏览器.浏览器不支持  </canvas> |

如果不支持，但还需要怎么弄？

抢修时态图 ie6 flash 降级，使用flash

## 2.2 canvas上下文context

### 2.2.1 Context

* 上下文：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。
  + getContext(‘3d’) webgl 单独研究，因为需要的知识很多，可能看不懂。
  + 有一个3d的框架 three.js

|  |
| --- |
| var canvas = document.getElementById( 'cavsElem' );  var ctx = canvas.getContext( '2d' ); |

## 2.3 绘制路径

### 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，不然绘制无效。

### 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。

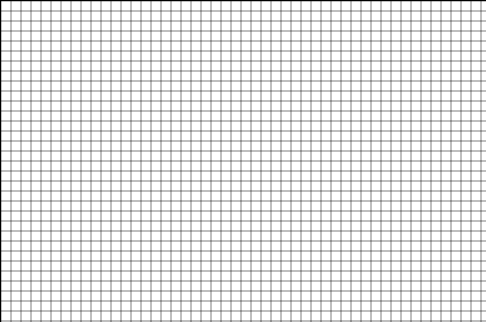
### 2.3.4 路径开始和闭合

* + - * 开始路径：ctx.beginPath();
      * 闭合路径：ctx.closePath();
      * 解释：如果复杂路径绘制，必须使用路径开始和结束。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
      * 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();

综合案例：绘制表格

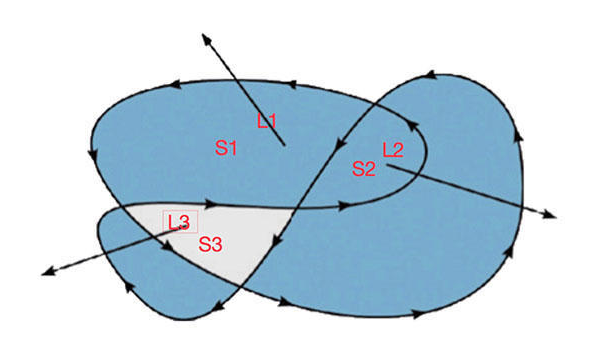


### 2.3.7 填充(fill)

* 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”是这么来判断有自我交叉情况的路径的：对于路径中的任意给定区域，从该区域内部画一条足够长的线段， 使此线段的终点完全落在路径范围之外。



们先假定路径的正方向为1（其实为-1啥的也都可以，正负方向互为相反数，不是0就行），那么反方向就是其相反数-1。

然后，我们在子路径切割的几块区域内的任意一点各取一条方向任意的射线，这里我只取了三个区域的射线为例，来判断这三块区域是“里面”还是“外面”。

接下来，我们就来判断了。S1中引出的射线L1，与S1的子路径的正方向相交，那么我们就给计数器+1，结果为+1，在外面。

S2中引出的射线L2，与两条子路径的正方向相交，计数器+2，结果为+2，在外面。

S3中引出的射线L3，与两条子路径相交，但是其中有一条的反方向，计数器+1-1，结果为0，在里面。没错，只要结果不为0，该射线所在的区域就在外面。

案例：填充矩形

### 2.3.8 快速创建矩形rect()方法

* 语法：ctx.rect(x, y, width, height);
* rect方法只是规划了矩形的路径，并没有填充和描边。

### 2.3.9 快速创建描边矩形和填充矩形

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

语法：ctx.fillRect(x, y, width, height);

### 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);
  + 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：绘制饼状图

## 2.5 绘制文字

### 2.5.1 绘制文字

context.fillText(*text*,*x*,*y*,*maxWidth*);

* 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。

### 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(txt) 返回包含指定文本宽度的对象

## 2.6 绘制图片（drawImage）

### 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 宽高

* 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

### 2.6.3 图片裁剪

* 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

案例： 图片裁剪

案例： 帧动画l