

后盾网 人人做后盾

www.houdunwang.com

Canvas

后盾网 2011-2016

什么是canvas:

- `<canvas>` 标签定义图形, 比如图表和其他图像。
- `<canvas>` 标签只是图形容器, 您必须使用脚本来绘制图形。
- canvas 其实对于HTML来说很简单, 只是一个标签元素而已, 自己并没有行为, 但却把一个绘图 API 展现给客户端 JavaScript 以使脚本能够把想绘制的东西都绘制到一块画布上, 拥有绘制路径, 矩形, 圆, 字符以及图像等功能。所有的标签只是图形的容器, 必须使用JavaScript的 API 操作绘图。

标签:

- `<canvas id="canvas" width="500" height="500"></canvas>`

什么是canvas

getContext

- 返回一个用于在画布上绘图的环境

```
<script type="text/javascript">  
  c = document.getElementById("canvas");  
  obj = c.getContext('2d');  
</script>
```

- getContext当前唯一的合法值是 "2d"，它指定了二维绘图，并且导致这个方法返回一个环境对象，该对象导出一个二维绘图 API。
- 提示：在未来，如果 <canvas> 标签扩展到支持 3D 绘图，getContext() 方法可能允许传递一个 "3d" 字符串参数。

getContext

矩形

- `context.fillRect(x,y,width,height)` 绘制“被填充”的矩形
- `context.strokeRect(x,y,width,height)` 绘制矩形（无填充）
- `context.clearRect(x,y,width,height)` 在给定的矩形内清除指定的像素

canvas方法或属性

```
<canvas id="c" width="300" height="300" style="border: 1px solid red;"></  
canvas>
```

```
<script type="text/javascript">  
    var c = document.getElementById("c");  
    var obj = c.getContext('2d');  
    //默认填充色和线条颜色全是黑色  
    //实体矩形  
    obj.fillRect(20,20,200,200);  
    //空心矩形  
    obj.strokeRect(0,0,250,250);  
    //清除矩形里面的像素  
    obj.clearRect(30,30,150,150);  
</script>
```

绘制矩形

颜色、样式

- `context.fillStyle='#f00f00'` 设置或返回填充绘画的颜色、渐变或模式
- `context.strokeStyle='green'` 设置或返回笔触的颜色、渐变或模式
- `context.lineWidth=10` 设置或返回当前的线条宽度
- `context.lineJoin="边界类型"` bevel:斜角,round:圆角,miter:尖角

canvas方法或属性

路径

- `beginPath()` 开始一条路径，或重置当前路径
- `closePath()` 创建从当前点回到起始点的路径(闭合路径)
- `moveTo(x,y)` 把路径移动到画布中的指定点，不创建线条
- `lineTo(x,y)` 添加一个新点，创建从该点到最后指定点的线条
- `fill()` 填充当前绘图（填充路径） 不加fill也可以画出线来
- `stroke()` 绘制已定义的路径（连线路径）

canvas方法或属性

```
<canvas id="canvas" width="300" height="300"></canvas>
  <script type="text/javascript">
    c = document.getElementById("canvas");
    obj = c.getContext('2d');
    obj.lineWidth = 10;
    //线颜色
    obj.strokeStyle = "red";
    //开始绘制路径
    obj.beginPath();
    //光标移动到0,0
    obj.moveTo(0, 0);
    //绘制到300,300
    obj.lineTo(300, 300);
    //绘制定义好的路径
    obj.stroke();
  </script>
```

绘制线

```
//获得canvas原生对象
var canvas = document.getElementById("canvas");
//获得画布对象
var ctx = canvas.getContext('2d');
//重置路径
ctx.beginPath();
ctx.moveTo(40,20);
ctx.lineTo(180,20);
ctx.lineTo(180,80);
ctx.strokeStyle = 'aqua';
//路径闭合，必须是在stroke的上面
ctx.closePath();
ctx.stroke();
```

绘制三角形

思路

- 默认初始化线条颜色宽度等
- 鼠标按下触发开始路径，设置线条宽度，线条颜色，设置起点
- 在鼠标按下时同时移动的时候，设置终点位置，并且绘制线条
- 文档对象鼠标抬起，解除画布的mousemove事件
- 扩展：点击取色器画不同颜色，橡皮擦，一键擦黑板功能

html5黑板实例思路

设置字体属性

- `context.font="40px Arial"`

设置对齐方式

- `context.textAlign="left | right | center"`

在画布上绘制“被填充的”文本

- `context.fillText(text,x,y,maxWidth);`

在画布上绘制文本（无填充）

- `context.strokeText(text,x,y,maxWidth)`

设置文字基线

- `context.textBaseline="top | middle | bottom";`

获取文本宽度

- `context.measureText(text);`

文本控制

画圆

- `context.arc(x,y,r,sAngle,eAngle,counterclockwise)` 创建弧/曲线（用于创建圆形或部分圆）

参数说明：

x	圆的中心的 x 坐标。
y	圆的中心的 y 坐标。
r	圆的半径。
sAngle	起始角，以弧度计。（弧的圆形的三点钟位置是 0 度）。
eAngle	结束角，以弧度计。
counterclockwise	可选。False = 顺时针，true = 逆时针。

弧度计算公式： 角度*Math.PI/180

canvas方法或属性

```
//获得canvas原生对象
var canvas = document.getElementById("canvas");
//获得画布对象
var ctx = canvas.getContext('2d');
//圆*****
//arc(x,y,radius,startAngle, endAngle,bAntiClockwise)
//x,y: 是arc的中心点
//radius: 是半径长度
//startAngle: 是以starAngle开始 (弧度)
//endAngle: 是以endAngle结束 (弧度)
//bAntiClockwise: 是否是逆时针, 设置为true意味着弧形的绘制是逆时针方向的, 设置为false意味着顺时针进行

ctx.beginPath();
ctx.strokeStyle = 'blueviolet';
ctx.lineWidth = 3;
//2*Math.PI就是整个圆的弧度, 是固定的数学公式
ctx.arc(150,150,150,0,2*Math.PI,true);
ctx.stroke();//fill就是实心圆
```



封闭状态

- save()
- restore()

保存当前环境的状态

返回之前保存过的路径状态和属性

canvas方法或属性

```
<canvas id="canvas" width="300" height="300" style="border: 1px solid blueviolet;"></canvas>
<script type="text/javascript">
    //先画红色的圆，再画黑色的圆
    var c = document.getElementById("canvas");
    var ctx = c.getContext('2d');
    //保存当前行上面的canvas的状态，也就是刚刚获得canvas干净的对象，没有平移属性，也没有线条颜色
    ctx.save();
    ctx.beginPath();
    //平移50px
    ctx.translate(50,50);
    ctx.strokeStyle = 'red';
    //圆
    ctx.arc(20,20,20,0,2*Math.PI);
    ctx.stroke();
    //恢复刚才保存的Canvas的干净的状态，因为要重新来
    ctx.restore();
    ctx.beginPath();
    ctx.arc(20,20,20,0,2*Math.PI);
    ctx.stroke();
</script>
```

save和restore

原生对象获取图片信息方法

- toDataURL() 获得图片base64加密的数据

`document.getElementById("canvas").toDataURL('image/png')`

图像控制
