## HTML5开发进阶指南学习笔记

第一章 canavs

1.1、canavs元素

原文Paths，表示由多条曲/直线段组成的闭合形状或不闭合路径，

根据语境译为“形状”或者“路径”。

1.1.1、2D 绘制上下文支持很多在画布上绘制路径的方法。通过路径可以创造出复

杂的形状和线条。要绘制路径,首先必须调用 beginPath() 方法,表示要开始

绘制新路径。然后,再通过调用下列方法来实际地绘制路径。

 arc(x, y, radius, startAngle, endAngle, counterclockwise) :以 (x,y) 为圆心绘

制一条弧线,弧线半径为 radius ,起始和结束角度(用弧度表示)分别为 startAngle 和

endAngle 。最后一个参数表示 startAngle 和 endAngle 是否按逆时针方向计算,值为 false

表示按顺时针方向计算。

 arcTo(x1, y1, x2, y2, radius) :从上一点开始绘制一条弧线,到 (x2,y2) 为止,并且以

给定的半径 radius 穿过 (x1,y1) 。

 bezierCurveTo(c1x, c1y, c2x, c2y, x, y) :从上一点开始绘制一条曲线,到 (x,y) 为

止,并且以 (c1x,c1y) 和 (c2x,c2y) 为控制点。

 lineTo(x, y) :从上一点开始绘制一条直线,到 (x,y) 为止。

 moveTo(x, y) :将绘图游标移动到 (x,y) ,不画线。

 quadraticCurveTo(cx, cy, x, y) :从上一点开始绘制一条二次曲线,到 (x,y) 为止,并

且以 (cx,cy) 作为控制点。

 rect(x, y, width, height) :从点 (x,y) 开始绘制一个矩形,宽度和高度分别由 width 和height 指定。这个方法绘制的是矩形路径,而不是 strokeRect() 和 fillRect() 所绘制的独立的形状。

创建了路径后,接下来有几种可能的选择。如果想绘制一条连接到路径起点的线条,可以调用

closePath() 。如果路径已经完成,你想用 fillStyle 填充它,可以调用 fill() 方法。另外,还可以调用 stroke() 方法对路径描边,描边使用的是 strokeStyle 。最后还可以调用 clip() ,这个方法可以在路径上创建一个剪切区域。

1.1.2、文本与图形总是如影随形。为此,2D 绘图上下文也提供了绘制文本的方法。绘制文本主要有两个

方法: fillText() 和 strokeText() 。其中fillText()方法是绘制实心文本，strokeText()是绘制空心文本。这两个方法都可以接收 4 个参数:要绘制的文本字符串、x 坐标、y 坐标和可选的最大像素宽度。而且,这两个方法都以下列 3 个属性为基础。

 font :表示文本样式、大小及字体,用 CSS 中指定字体的格式来指定,例如 "10px Arial" 。

 textAlign :表示文本对齐方式。可能的值有 "start" 、 "end" 、 "left" 、 "right" 和 "center" 。建议使用 "start" 和 "end" ,不要使用 "left" 和 "right" ,因为前两者的意思更稳妥,能同时适合从左到右和从右到左显示(阅读)的语言。

 textBaseline :表示文本的基线。可能的值有 "top" 、 "hanging" 、 "middle" 、 "alphabetic" 、"ideographic" 和 "bottom" 。

1.1.3、save()和restore()

无论是刚才执行的变换,还是 fillStyle 、 strokeStyle 等属性,都会在当前上下文中一直有效,除非再对上下文进行什么修改。虽然没有什么办法把上下文中的一切都重置回默认值,但有两个方法可以跟踪上下文的状态变化。如果你知道将来还要返回某组属性与变换的组合,可以调用 save() 方法。调用这个方法后,当时的所有设置都会进入一个栈结构,得以妥善保管。然后可以对上下文进行其他修改。等想要回到之前保存的设置时,可以调用 restore() 方法,在保存设置的栈结构中向前返回一级,恢复之前的状态。连续调用 save() 可以把更多设置保存到栈结构中,之后再连续调用 restore() 则可以一级一级返回。下面来看一个例子。

context.fillStyle = "#ff0000";//刚开始是红色

context.save();//进入一个栈结构,得以妥善保管，然后可以对上下文进行其他修改。

context.fillStyle = "#00ff00";//绿色

context.translate(100, 100);//变换原点

context.save();

context.fillStyle = "#0000ff";//蓝色

context.fillRect(0, 0, 100, 200); //从点(100,100)开始绘制蓝色矩形

context.restore();

context.fillRect(10, 10, 100, 200); //从点(110,110)开始绘制绿色矩形

context.restore();

context.fillRect(0, 0, 100, 200); //从点(0,0)开始绘制红色矩形

首先,将 fillStyle 设置为红色,并调用 save() 保存上下文状态。接下来,把 fillStyle 修改为绿色,把坐标原点变换到(100,100),再调用 save() 保存上下文状态。然后,把 fillStyle 修改为蓝色并绘制蓝色的矩形。因为此时的坐标原点已经变了,所以矩形的左上角坐标实际上是(100,100)。然后调用 restore() ,之后 fillStyle 变回了绿色,因而第二个矩形就是绿色。之所以第二个矩形的起点坐标是(110,110),是因为坐标位置的变换仍然起作用。再调用一次 restore() ,变换就被取消了,而fillStyle 也返回了红色。所以最后一个矩形是红色的,而且绘制的起点是(0,0)。需要注意的是, save() 方法保存的只是对绘图上下文的设置和变换,不会保存绘图上下文的内容。

1.1.4、 变换

通过上下文的变换,可以把处理后的图像绘制到画布上。2D 绘制上下文支持各种基本的绘制变换。

创建绘制上下文时,会以默认值初始化变换矩阵,在默认的变换矩阵下,所有处理都按描述直接绘制。

为绘制上下文应用变换,会导致使用不同的变换矩阵应用处理,从而产生不同的结果。

可以通过如下方法来修改变换矩阵。

 rotate(angle) :围绕原点旋转图像 angle 弧度。

 scale(scaleX, scaleY) :缩放图像,在 x 方向乘以 scaleX ,在 y 方向乘以 scaleY 。 scaleX

和 scaleY 的默认值都是 1.0。

 translate(x, y) :将坐标原点移动到 (x,y) 。执行这个变换之后,坐标(0,0)会变成之前由 (x,y)

表示的点。

 transform(m1\_1, m1\_2, m2\_1, m2\_2, dx, dy) :直接修改变换矩阵,方式是乘以如下

矩阵。

m1\_1

m1\_2

dx

m2\_1

m2\_2

dy

0

0

1

 setTransform(m1\_1, m1\_2, m2\_1, m2\_2, dx, dy) :将变换矩阵重置为默认状态,然后再调用 transform() 。

1. 创建基本的游戏世界