第3章 HTML5画布

HTML5 画布即 HTML5 canvas，是现代浏览器都支持的 HTML5 非插件绘图的功能。HTML5 canvas 是 HTML5 新增的专门用于绘制图形的元素。在页面上放置一个 canvas 元素就相当于放置了一块画布，可以在其中进行图形的绘制。在 canvas 元素内绘制图形需要结合 JavaScript 脚本。利用 canvas 可以进行跨平台的动画和游戏的开发，能够实现对图像进行像素级别的操作。

3.1 HTML5 的 canvas 元素


观看视频

canvas 元素的外观与 img 元素相似，但是没有 img 元素的 src 属性和 alt 属性。canvas 元素的 height 属性和 width 属性分别用来设置画布的高度和宽度，单位是像素。默认的画布高度是 150 像素，宽度是 300 像素。id 属性为 canvas 元素的标识，在 JavaScript 脚本中需要根据 id 值来寻找 canvas 元素。< canvas >标签必须成对使用。默认的 canvas 元素在页面上会显示一块空白的没有边框的矩形。可以通过 CSS 来设置 canvas 元素的外观，例如 canvas. html，为 canvas 元素添加一个实心的边框，宽度为 3px。

canvas. html 的代码如下：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>canvas 元素示例</title>
    </head>
    <body>
         <canvas id="myCanvas" width="200" height="50" style="border:3px solid">
</canvas>
    </body>
</html>
```

canvas. html 的显示结果如图 3-1 所示。

图 3-1　canvas 元素在 Chrome 浏览器中的显示结果

观看视频

3.2　绘制简单的图形

canvas 元素本身并不能实现图形绘制，需要和 JavaScript 脚本结合起来。首先，给 canvas 元素添加一个 id 属性，在 JavaScript 脚本中通过 id 属性寻找对应的 canvas 元素。然后通过 canvas 元素的 getContext()方法获取其上下文，即创建 Context 对象，以获取允许进行绘制的 2D 环境。最后通过 Context 对象的相关方法完成绘制，例如 fillStyle()方法、fillRect()方法等。

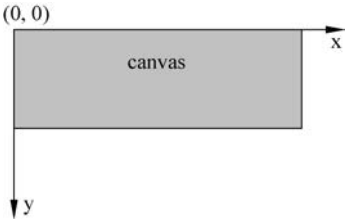

图 3-2　canvas 元素的坐标

进行绘制时，需要指定确定的坐标位置，坐标原点(0，0)位于 canvas 的左上角，x 轴水平方向向右延伸，y 轴垂直向下延伸，如图 3-2 所示。

3.2.1　绘制直线

Context 对象的 moveTo(x,y)方法是将画笔移动到指定的坐标点(x,y)，lineTo(x,y)方法是从落笔点绘制路径到坐标点(x,y)。只使用以上两个方法是无法在画布上看到直线的，lineTo(x,y)方法用于绘制路径，要使路径在画布上显示出来，还需要进行描边。可以连续地绘制多条路径，然后使用 stroke()方法一次性描边。可以使用 CSS 设置绘制直线的样式，例如 line.html。

line.html 的代码如下：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>绘制直线</title>
    </head>
    <body>
        <canvas id="myCanvas" width="150" height="150" style="border:1px solid"></canvas>
    </body>
    <script type="text/javascript">
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.moveTo(0, 0);
        context.lineTo(150, 150);
        context.moveTo(0, 150);
```

```
        context.lineTo(150, 0);
        context.stroke();
    </script>
</html>
```

line.html 在浏览器中的显示结果如图 3-3 所示。

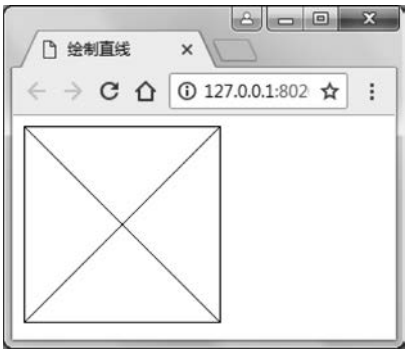

图 3-3 line.html 的显示结果

3.2.2 绘制矩形

canvas 元素可以绘制两种矩形：一种是填充矩形，另一种是矩形轮廓。Context 对象的 fillRect()方法用来绘制填充矩形，strokeRect()方法用来绘制矩形轮廓。

fillRect()方法的前两个参数为矩形的左上角的坐标，后两个参数为矩形的宽度和高度。strokeRect()方法的参数与 fillRect()方法的参数含义相同。设置矩形的外观可以使用 fillStyle 属性和 strokeStyle 属性。fillStyle 属性用来设置矩形区域的填充颜色，strokeStyle 属性用来设置矩形轮廓的颜色。例如 rect.html，绘制一个填充矩形和一个矩形轮廓。

rect.html 的代码如下：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>绘制矩形</title>
    </head>
    <body>
        <canvas id="myCanvas" width="200" height="150" style="border:1px solid">
</canvas>
    </body>
    <script type="text/javascript">
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.fillStyle="green";
        context.fillRect(20,20,100,80);
        context.strokeStyle="red";
        context.strokeRect(50,50,100,40);
    </script>
</html>
```

rect.html 在浏览器中的显示结果如图 3-4 所示。

3.2.3 绘制圆或圆弧

canvas 元素可以用来绘制圆或圆弧，使用的方法有 beginPath()、arc()、closePath()、fill()。

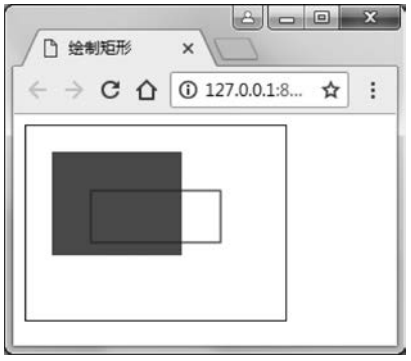

图 3-4　rect.html 的显示结果

1. beginPath()

开始一条路径或重置路径。

2. arc(x,y,r,sAngle,eAngle,counterclockwise)

x、y 为圆心的坐标；r 为圆的半径；sAngle 为以弧度计的起始角；eAngle 为以弧度计的结束角；counterclockwise 参数可选，规定逆时针或顺时针绘图，true 为逆时针，false 为顺时针。

3. closePath()

闭合路径，如果图形本来就是闭合的，则此方法不起作用。

4. fill()

填充当前的路径或图像，默认的颜色是黑色。

例如 arc.html，绘制一个圆和若干条圆弧。

arc.html 的代码如下：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>绘制圆或圆弧</title>
    </head>
    <body>
        <canvas id="myCanvas" width="300" height="200" style="border:1px solid">
</canvas>
    </body>
    <script type="text/javascript">
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.fillStyle="yellow";
        context.beginPath();
        context.arc(150,100,50,0,Math.PI*2);
        context.closePath();
        context.fill();
        for(var i=0;i<20;i++){
            context.strokeStyle="green";
            context.beginPath();
            context.arc(0,200,i*10,0,Math.PI*3/2,true);
            context.stroke();
        }
    </script>
</html>
```

arc.html 在浏览器中的显示结果如图 3-5 所示。

3.2.4　绘制三角形

使用绘制路径的方法可以自由绘制出三角形等多边形。例如 triangle.html，绘制一个填充色为绿色的三角形。

triangle.html 的代码如下：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>绘制三角形</title>
    </head>
    <body>
        <canvas id="myCanvas" width="200" height="200" style="border:1px solid"></canvas>
    </body>
    <script type="text/javascript">
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.beginPath();
        context.moveTo(20, 20);
        context.lineTo(40, 100);
        context.lineTo(180, 100);
        context.closePath();
        context.stroke();
        context.fillStyle = "green";
        context.fill();
    </script>
</html>
```

triangle.html 在浏览器中的显示结果如图 3-6 所示。

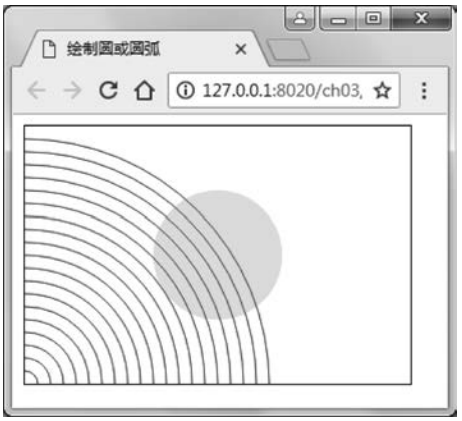

图 3-5 arc.html 的显示结果

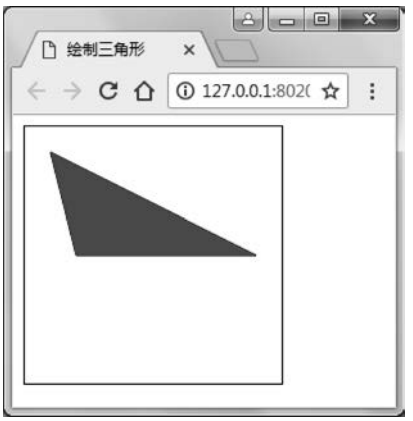

图 3-6 triangle.html 的显示结果

3.3 绘制文字


观看视频

可以使用 fillText()方法和 strokeText()方法来绘制文字，其中 fillText()方法用来绘制填充文字，strokeText()方法用来绘制轮廓文字。

3.3.1 绘制填充文字

绘制填充文字时，fillText()方法的用法如下：

```
context.fillText(text, x, y, [max Width]);
```

其中，text 为要绘制的文字；x、y 为绘制文字的坐标；maxWidth 为可选参数，表示显示文字时的最大宽度。如果要绘制的文字大于最大宽度，则字体会被调整成更小的字号或者更瘦的字体。例如 filledText.html。

filledText.html 的代码如下：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>绘制填充文字</title>
    </head>
    <body>
        <canvas id="myCanvas" width="250" height="150" style="border:1px solid">
</canvas>
    </body>
    <script type="text/javascript">
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.font="italic 35px 黑体";
        context.fillStyle="red";
        context.fillText("红色填充文字",10,40,200);
        context.font="bold 40px 宋体";
        context.fillStyle="green";
        context.fillText("绿色填充文字",10,100,200);
    </script>
</html>
```

图 3-7　filledText.html 的显示结果

filledText.html 在浏览器中的显示结果如图 3-7 所示。

3.3.2　绘制轮廓文字

绘制轮廓文字时，strokeText() 方法的用法如下：

```
context.strokeText(text, x, y, [max Width]);
```

其中，text 为要绘制的文字；x、y 为绘制文字的坐标；maxWidth 为可选参数，表示显示文字时的最大宽度。例如 hollowText.html。

hollowText.html 的代码如下：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>绘制轮廓文字</title>
    </head>
    <body>
```

```
        <canvas id="myCanvas" width="250" height="150" style="border:1px solid"></canvas>
    </body>
    <script type="text/javascript">
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.font="italic 35px 黑体";
        context.strokeStyle="red";
        context.strokeText("红色轮廓文字",10,40,200);
        context.font="bold 40px 宋体";
        context.strokeStyle="green";
        context.strokeText("绿色轮廓文字",10,100,200);
    </script>
</html>
```

hollowText.html 在浏览器中的显示结果如图 3-8 所示。

图 3-8 hollowText.html 的显示结果

3.4 图形变换


观看视频

利用图形的变换可以绘制出大量复杂多变的图形。图形的变换主要包括移动、缩放、旋转、变形等。

3.4.1 保存与恢复

在绘画时，本来使用绿色笔画，突然需要用红色笔画，但画完之后又要使用绿色笔画。如果是现实中的作画，可以使用不同的笔蘸上不同的墨水，根据颜色选择画笔。但是在 canvas 中画笔只有一支。如果要更换画笔的颜色，就需要保存和恢复状态，状态其实就是画布当前属性的一个快照，包括图形的属性、当前裁切路径、当前应用的变换。canvas 中使用 save()方法来保存状态，使用 restore()方法来恢复状态。canvas 状态是用栈来保存的，每次调用 save()方法，就把当前状态入栈保存，当前状态成为栈顶；每次调用 restore()方法，就把栈顶的状态取出来，并将画布恢复到这个状态。例如 saveAndRestore.html，利用状态的保存与恢复画颜色不同的填充矩形。

saveAndRestore.html 的代码如下：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>状态的保存和恢复</title>
    </head>
    <body>
        <canvas id="myCanvas" width="250" height="250" style="border:1px solid">
</canvas>
    </body>
    <script type="text/javascript">
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.fillStyle = "yellow";
        context.fillRect(10, 10, 230, 230);
        context.fill();

        context.save();
        context.fillStyle = "aquamarine";
        context.fillRect(30, 30, 190, 190);

        context.save();
        context.fillStyle = "cornflowerblue";
        context.fillRect(50, 50, 150, 150);

        context.restore();
        context.beginPath();
        context.fillRect(70, 70, 110, 110);
        context.restore();
        context.fillRect(90, 90, 80, 80);
        context.fill();
    </script>
</html>
```

saveAndRestore.html 在浏览器中的显示结果如图 3-9 所示。

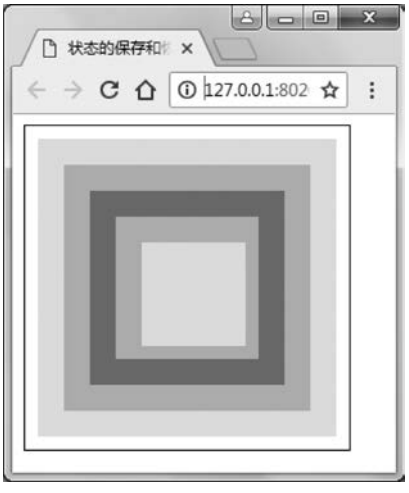

图 3-9　saveAndRestore.html 的显示结果

3.4.2 移动

在绘制图形时，可以使用 Context 对象的 translate()方法移动坐标空间，使画布的坐标空间发生水平和垂直方向的偏移，translate(dx,dy)中 dx 为水平方向的偏移量，dy 为垂直方向的偏移量。例如 translate.html，利用移动绘制一排圆。

translate.html 的代码如下：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>移动</title>
    </head>
    <body>
         <canvas id="myCanvas" width="400" height="80" style="border:1px solid">
</canvas>
    </body>
    <script type="text/javascript">
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.fillStyle = "orange";
        for(var i = 1; i<10; i++) {
            context.save();
            context.beginPath();
            context.arc(40, 40, 20, 0, Math.PI * 2);
            context.closePath();
            context.fill();
            context.translate(40, 0);
        }
    </script>
</html>
```

translate.html 在浏览器中的显示结果如图 3-10 所示。

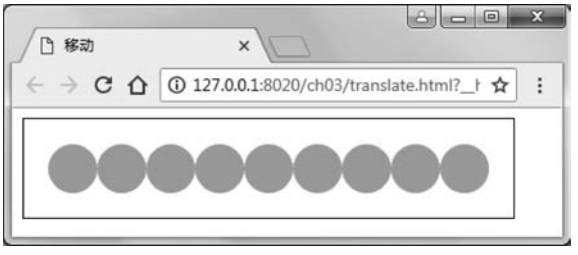

图 3-10 translate.html 的显示结果

3.4.3 缩放

Context 对象的 scale()方法用于增减 canvas 上下文对象中的像素数目，从而实现图形或图像的放大或缩小，context.scale(sx,sy)中的 sx 为 x 轴的缩放因子，sy 为 y 轴的缩放因子，它们的值必须是正数，如果值大于 1，则为放大图形，如果值小于 1，则为缩小图形。例如 scale.html，绘制逐渐缩小的圆。

scale.html 的代码如下：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>缩放</title>
    </head>
    <body>
         <canvas id="myCanvas" width="400" height="80" style="border:1px solid">
</canvas>
    </body>
    <script type="text/javascript">
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.fillStyle = "orange";
        for(var i = 1; i<15; i++) {
            context.save();
            context.beginPath();
            context.arc(40, 40, 20, 0, Math.PI * 2);
            context.closePath();
            context.fill();
            context.translate(40, 0);
            context.scale(0.9,0.9);
        }
    </script>
</html>
```

scale.html 在浏览器中的显示结果如图 3-11 所示。

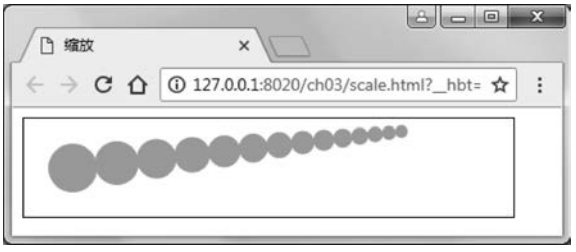

图 3-11　scale.html 在浏览器中的显示结果

3.4.4　旋转

Context 对象的 rotate()方法用于以原点为中心旋转上下文对象的坐标空间，context.rotate(angle)方法中的 angle 参数指以弧度计的顺时针方向旋转的角度。例如 rotate.html 利用旋转绘制环状排列的圆。

rotate.html 的代码如下：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>旋转</title>
```

```
    </head>
    <body>
        <canvas id="myCanvas" width="300" height="300" style="border:1px solid"></canvas>
    </body>
    <script type="text/javascript">
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.translate(150,150);
        context.fillStyle = "orange";
        for(var i = 0; i < 8; i++) {
            context.save();
            context.rotate(Math.PI * (2/4 + i/4));
            context.translate(0, -100);
            context.beginPath();
            context.arc(100, 40,20, 0, Math.PI * 2);
            context.closePath();
            context.fill();
            context.restore();
        }
    </script>
</html>
```

rotate.html 在浏览器中的显示结果如图 3-12 所示。

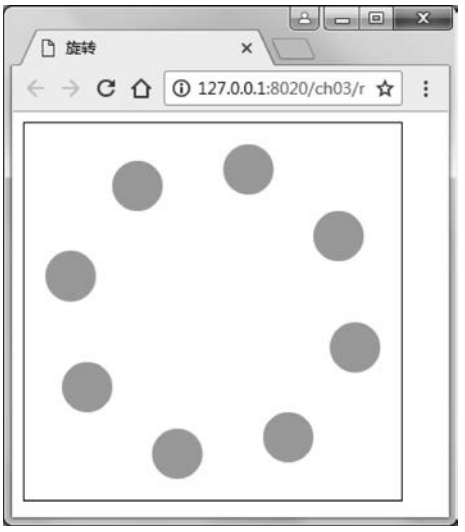

图 3-12 rotate.html 在浏览器中的显示结果

3.4.5 变形

Context 对象的 transform()方法用于直接修改变换矩阵。矩阵变换常用于坐标变换不能达到预期效果的情况，能够实现比普通的坐标变换更加复杂的效果。transform()方法的用法如下：

```
context.transform(a, b, c, d, e, f);
```

各参数的含义如下。

- a：水平缩放绘图。
- b：水平倾斜绘图。
- c：垂直倾斜绘图。
- d：垂直缩放绘图。
- e：水平移动绘图。
- f：垂直移动绘图。

可见，可以在 transform()方法中同时实现平移、缩放、旋转，也可以使用 transform()方法实现以上 3 种变换中的一种。

画布上的每一个对象都拥有一个当前的变换矩阵，Context 对象的 setTransform()方法用于将当前的变换矩阵重置为最初的矩阵，即单位矩阵，然后以相同的参数运行 transform()方法，也就是说 setTransform()方法允许缩放、旋转、移动并倾斜当前的环境。该变换只会影响 setTransform()方法之后的绘图。例如 transform.html，利用变形和旋转

在画布上显示呈螺旋状排列的半透明的半圆。

transform.html 的代码如下：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>矩阵变换</title>
    </head>
    <body>
        <canvas id="myCanvas" width="330" height="300" style="border:1px solid">
</canvas>
    </body>
    <script type="text/javascript">
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.translate(200, 20);
        for(var i = 1; i < 100; i++) {
            context.save();
            context.transform(0.95, 0, 0, 0.95, 30, 30);
            context.rotate(Math.PI / 12);
            context.beginPath();
            context.fillStyle = "orange";
            context.globalAlpha = "0.7";
            context.arc(0, 0, 55, 0, Math.PI, true);
            context.closePath();
            context.fill();
        }
    </script>
</html>
```

transform.html 在浏览器中的显示结果如图 3-13 所示。

图 3-13　transform.html 在浏览器中的显示结果

3.5 操作图像

利用 canvas 元素不仅可以绘制各种各样的图形，而且可以引入外部图像，并对图像进行各种操作，例如改变图像大小、图像切片、图像合成等。canvas 支持多种常见的图像格式。向 canvas 元素引入图像分以下 3 步。

1. 创建 image 对象

```
var image = new Image();
```


观看视频

2. 设定 image 对象的 onload 属性

```
image.onload = function(){
}
```

3. 在 function()中绘制图像

可以使用以下方法：

- context.drawImage(image,x,y)：在画布的指定位置绘制图像。
- context.drawImage(image,x,y,width,height)：按参数指定的位置和大小绘制图像。
- context.drawImage(image,x,y,sWidth,sHeight,dx,dy,dWidth,dHeight)：裁剪图像，并在画布上绘制图像。

例如 image.html，在画布上绘制图像，并绘制轮廓文字。

image.html 的代码如下：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>引入图像</title>
    </head>
    <body>
        <canvas id="myCanvas" width="300" height="200"></canvas>
    </body>
    <script type="text/javascript">
        function show() {
            var canvas = document.getElementById("myCanvas");
            var context = canvas.getContext("2d");
            var img = new Image();
            img.onload = function() {
                context.drawImage(img,0,0,300,200);
                context.font="bold 70px Arial Black";
                context.strokeStyle="orangered";
                context.strokeText("flowers",120,180,150);
            }
            img.src = "img/flower.jpg";
        }
        window.onload=function(){
            show();
        }
```

```
    </script>
</html>
```

image.html 在浏览器中的显示结果如图 3-14 所示。

图 3-14　image.html 的显示结果

观看视频

3.6　其他颜色和样式

前面使用 canvas 元素绘制图形时，使用的是 Context 对象的 fillStyle()方法和 strokeStyle()方法，除此之外，canvas 元素还支持更多的颜色和样式，包括线型、渐变、图案、透明度和阴影。合理利用这些颜色和样式，可以绘制出更加复杂多变、丰富多彩的图形。

3.6.1　线型

Context 对象的 lineWidth、lineCap、lineJoin、miterLimit 属性分别用于设置线条的粗细、端点样式、两条线段连接处的样式以及绘制交点的方式。

1. lineWidth

lineWidth 属性的值必须为正数，单位为像素，默认值为 1.0。

2. lineCap

lineCap 属性可取值 butt、round、square，默认值为 butt。butt 表示向线条的每个末端添加平直的边缘；round 表示向线条的每个末端添加圆形线帽；square 表示向线条的每个末端添加正方形线帽。例如 widthAndCap.html 分别绘制圆形线帽但粗细不同的线型。

widthAndCap.html 的代码如下：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>线型示例</title>
    </head>
    <body>
        <canvas id="myCanvas" width="300" height="200" style="border:1px solid"></canvas>
```

```
    </body>
    <script type="text/javascript">
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        for(var i=1;i<=14;i++){
            context.strokeStyle="green";
            context.lineWidth=i;
            context.lineCap="round";
            context.beginPath();
            context.moveTo(20*i,50);
            context.lineTo(20*i,200);
            context.stroke();
        }
    </script>
</html>
```

widthAndCap.html 在浏览器中的显示结果如图 3-15 所示。

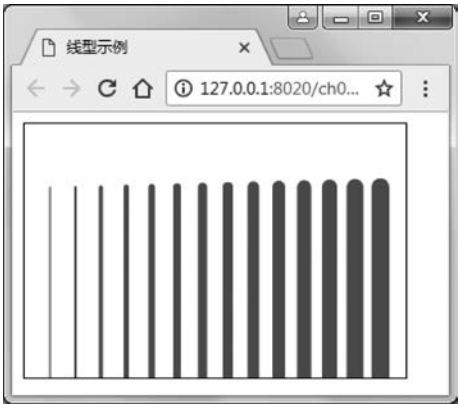

图 3-15 widthAndCap.html 在浏览器中的显示结果

3. lineJoin

lineJoin 属性用于设置两条线段连接处的样式，可取值 round、bevel、miter，默认值为 miter。round 为圆角，bevel 为边角，miter 为尖角。

4. miterLimit

miterLimit 属性用来设置或返回最大斜接长度，斜接长度指的是在两条线交汇处内角和外角之间的距离。只有当 lineJoin 属性为 miter 时，miterLimit 属性才有效。边角的角度越小，斜接长度就会越大，为了避免斜接长度太大，可以使用 miterLimit 属性进行限制。如果斜接长度超过了 miterLimit 属性设置的值，则边角会以 bevel(即边角)的形式显示。

3.6.2 渐变

canvas 元素在绘制图形时除了可以指定固定的颜色之外，还可以指定渐变色。渐变分为线性渐变和径向渐变。

1. 绘制线性渐变

要绘制线性渐变，首先需要使用 Context 对象的 createLinearGradient()方法创建 canvasGradient 对象，然后使用 addColorStop()方法上色。createLinearGradient()方法定义如下：

```
context.createLinearGradient(x1, y1, x2, y2);
```

其中，x1 和 y1 为渐变的起点，x2 和 y2 为渐变的终点。

addColorStop()方法定义如下：

```
context.addColorStop(position, color);
```

其中，position 为渐变中色标的相对位置，为 0～1 的浮点值，渐变起点的相对位置为 0，终点的相对位置为 1。例如 linearGradient.html，在矩形中添加 3 种颜色，起点为红色，中间点为黄色，终点为白色。

linearGradient.html 的代码如下：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>线性渐变</title>
    </head>
    <body>
        <canvas id="myCanvas" width="300" height="200" style="border:1px solid"></canvas>
    </body>
    <script type="text/javascript">
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        var gradient = context.createLinearGradient(0, 0, 0, 200);
        gradient.addColorStop(0, "red");
        gradient.addColorStop(0.5, "yellow");
        gradient.addColorStop(1, "white");
        context.fillStyle = gradient;
        context.fillRect(0, 0, 300, 200);
    </script>
</html>
```

图 3-16　linearGradient.html 的显示结果

linearGradient.html 在浏览器中的显示结果如图 3-16 所示。

2. 绘制径向渐变

径向渐变也称为扩散性渐变，首先需要使用 Context 对象的 createRadialGradient()方法创建 canvasGradient 对象，然后使用 addColors()方法上色。createRadialGradient()的使用方法如下：

```
context.createRadialGradient(x1, y1, r1, x2, y2, r2);
```

其中，x1 和 y1 定义一个以(x1,y1)为圆心、以 r1 为半径的开始圆，x2 和 y2 定义一个以(x2,y2)为圆心、以 r2 为半径的结束圆。这两个圆描述了渐变的方向及起止位置，也描述了渐变的形状。例如 radialGradient.html。

radialGradient.html 的代码如下：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>径向渐变</title>
    </head>
```

```
    <body>
        <canvas id="myCanvas" width="300" height="200"></canvas>
    </body>
    <script type="text/javascript">
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        var gradient = context.createRadialGradient(50,50,20,100,100,50);
        gradient.addColorStop(0, "red");
        gradient.addColorStop(0.8, "yellow");
        gradient.addColorStop(1, "blue");
        context.fillStyle = gradient;
        context.fillRect(10, 10, 200, 200);
    </script>
</html>
```

radialGradient.html 在浏览器中的显示结果如图 3-17 所示。

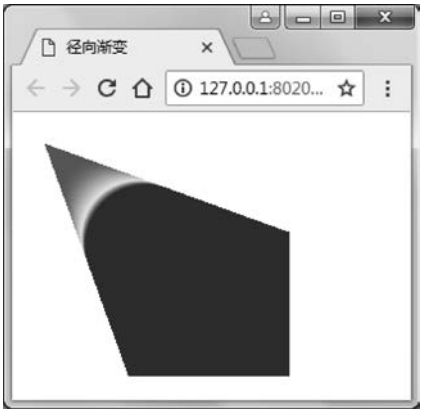

图 3-17 radialGradient.html 的显示结果

3.6.3 绘制图案

Context 对象的 createPattern()方法可以用来填充图像,其用法如下:

```
context.createPattern(image, type);
```

其中,image 为要引用的 image 对象或者另一个 Canvas 对象; type 是填充图像的方式,可取值如下。

- repeat: 同时沿 x 轴方向和 y 轴方向平铺。
- repeat-x: 只沿 x 轴方向平铺。
- repeat-y: 只沿 y 轴方向平铺。
- no-repeat: 不平铺。

例如 pattern.html,在 canvas 元素内沿 x 轴方向和 y 轴方向平铺图像。

pattern.html 的代码如下:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>平铺图像</title>
    </head>
    <body>
        <canvas id="myCanvas" width="200" height="200" style="border:1px solid"></canvas>
    </body>
    <script type="text/javascript">
        function show() {
            var canvas = document.getElementById("myCanvas");
            var context = canvas.getContext("2d");
            var img = new Image();
```

```
            img.onload = function() {
                var pattern = context.createPattern(img, 'repeat');
                context.fillStyle = pattern;
                context.fillRect(0, 0, 200, 200);
            }
            img.src = "img/timg.jpg";
            alert(img.src);
        }
        window.onload = function() {
            show();
        }
    </script>
</html>
```

图 3-18　pattern.html 在浏览器中的显示结果

pattern.html 在浏览器中的显示结果如图 3-18 所示。

3.6.4　透明度

除了可以使用 Context 对象的 globalAlpha 属性设置图形的透明度外，还可以利用 rgba()方法设置色彩的透明度。rgba()的使用方法如下：

```
rgba(R, G, B, A)
```

其中，R、G、B 分别是颜色的红、绿、蓝分量，A 为不透明值，为 0～1 的浮点数，0 为完全透明，1 为完全不透明。例如"rgba(0,255,0,0.5)"表示半透明的绿色。例如 alpha.html，在画布上画一排透明度不同的圆。

alpha.html 的代码如下：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>透明度</title>
    </head>
    <body>
        <canvas id="myCanvas" width="560" height="80" style="border:1px solid">
</canvas>
    </body>
    <script type="text/javascript">
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        for(var i = 1; i<10; i++) {
            context.fillStyle = "rgba(0,255,0," + ((i-1)/9) + ")";
            context.save();
            context.beginPath();
            context.arc(40, 40, 30, 0, Math.PI * 2);
            context.closePath();
```

```
            context.fill();
            context.translate(60, 0);
        }
    </script>
</html>
```

alpha.html 在浏览器中的显示结果如图 3-19 所示。

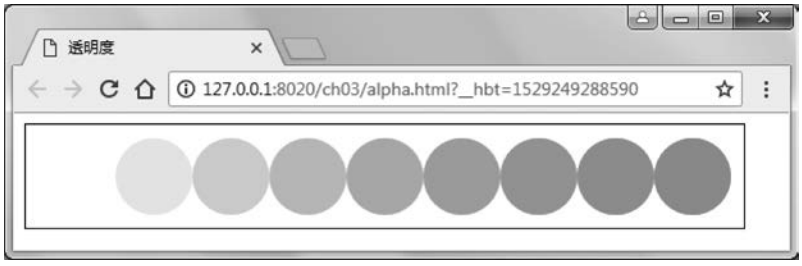

图 3-19 alpha.html 在浏览器中的显示结果

3.6.5 阴影

可以使用 Context 对象的属性对绘制的图形或文字添加阴影效果，设置阴影的主要属性如下。

- shadowOffsetX：设置或返回形状与阴影的水平距离，默认值为 0。
- shadowOffsetY：设置或返回形状与阴影的垂直距离，默认值为 0。
- shadowBlur：设置或返回阴影的模糊级数，取值为整数值。
- shadowColor：设置或返回阴影的颜色。

例如 shadow.html，为画布中的文字添加阴影。

shadow.html 的代码如下：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>阴影</title>
    </head>
    <body>
        <canvas id="myCanvas" width="250" height="150" style="border:1px solid"></canvas>
    </body>
    <script type="text/javascript">
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.shadowOffsetX=4;
        context.shadowOffsetY=4;
        context.shadowBlur=2;
        context.shadowColor="rgba(0,0,0,0.5)";
        context.font="italic 35px 黑体";
        context.fillStyle="red";
        context.fillText("红色填充文字",10,40,200);
        context.font="bold 40px 宋体";
```

```
        context.strokeStyle = "green";
        context.strokeText("绿色轮廓文字",10,100,200);
    </script>
</html>
```

shadow.html 在浏览器中的显示结果如图 3-20 所示。

图 3-20　shadow.html 在浏览器中的显示结果

小结

- 利用 HTML5 中的 canvas 元素可以实现图形的绘制，能够对图像进行像素级别的操作。
- 利用 canvas 元素可以绘制直线、矩形、圆或圆弧、多边形等图形。
- 利用 canvas 元素可以绘制填充文字或轮廓文字。
- 利用 canvas 元素可以实现状态的保存与恢复、图形的平移、图形的缩放、图形的旋转、图形的变形等。
- 利用 canvas 元素可以操作外部图像，将其引入网页中。
- canvas 元素还支持更多的颜色和样式，包括线型、渐变、图案、透明度和阴影。

习题

1. Context 对象的(　　)方法将画笔移动到指定的坐标点。
   A. moveTo()　　B. lineTo()　　C. fill()　　D. stroke()
2. canvas 元素使用(　　)方法获取上下文对象。
   A. stroke()　　B. getContext()　　C. fill()　　D. controller()
3. 以下关于线条属性的说法中，(　　)是正确的。
   A. lineCap 属性设置或返回线条末端线帽的样式
   B. butt 值为向线条末端添加平直的边缘
   C. round 值为向线条末端添加圆形线帽
   D. 以上都正确
4. Context 对象的(　　)方法可以绘制矩阵轮廓。

A. fillRect()　　B. strokeRect()　C. fillStyle()　　D. strokeStyle()

5. Context 对象的(　　)方法用于绘制填充文字。

A. fillText()　　B. strokeText()　C. fillStyle()　　D. strokeStyle()

6. Context 对象的(　　)方法可以实现坐标空间的移动。

A. scale()　　B. translate()　C. transform()　　D. rotate()

7. Context 对象的(　　)方法可以实现旋转坐标空间。

A. scale()　　B. translate()　C. transform()　　D. rotate()

8. 使用 canvas 绘制一个填充矩形,要求矩形的填充色为红色,再绘制一个矩形轮廓,要求矩形为圆角矩形,边框的颜色为蓝色,边框的宽度为 10px。