第 6 章

使用图形

您经常需要在您网站的页面上添加图形，例如位图图像或 2D 形状。此外，您还可以向网页上的图形添加文本和应用不同的颜色和渐变。HTML 中的画布元素提供一个绘制图面来允许您向网站动态添加文本、形状和图像。此外，您可以向图形添加转换和动画效果，以使网页在视觉上具有吸引力。

本章讨论了画布和可在画布上创建的各种图形对象。此外，它还讨论了如何在画布的图形对象上应用动画和变换。

目标

在本课中，您将学习：

- 介绍画布
- 变换和动画显示画布元素


NIIT
training
china
.com

介绍画布

LearnGraphs Ltd. 提供关于基本绘图技能的在线教程。这些教程包括需要在网页上绘制的各种图形，例如条形图、直方图和饼图。但是，要在网页上绘制图形，需要合并诸如线、弧和圆之类的图形，并在形状上应用颜色和渐进等效果。在 HTML 中引入画布元素后，您可以很轻松地在网页上创建图形和应用动画。画布元素具有大量 2D 应用编程接口 (API) 形式的函数，用于在画布上绘制图形。

创建画布

画布提供一种简单有效的方式来在网页上创建图形。画布自身没有内容。画布仅是网页上的一个区域，充当嵌入图形对象的容器。它允许使用 JavaScript 动态呈现位图图像和 2D 形状。您需要执行以下任务来创建画布并将其用于绘制各种图形：

1. 定义 画布
2. 访问 画布

定义画布

画布是使用 `<CANVAS>` 标记定义的。该标记在 HTML 文档的正文部分中定义。`<CANVAS>` 标记提供使您能够为画布指定大小、边框和 ID 的各种属性。下表中列出了 `<CANVAS>` 标记提供的属性。

属性	***描述***
`ID`	*用于指定画布的唯一 ID，该 ID 用于标识 JavaScript 代码中的画布。*
`width`	*用于指定画布宽度（以像素为单位）。`width` 属性的默认值是 300 像素。*
`height`	*用于指定画布的高度（以像素为单位）。`height` 属性的默认值是 150 像素。*
`Style`	*用于指定要应用到画布的样式。*

<CANVAS> 标记的属性

您可以通过在 `<BODY>` 标记中使用以下代码来定义画布：

```
<CANVAS ID="myCanvas" width="300" height="300" style="border:1px solid
black">
</CANVAS>
```

上述代码将创建 ID 为 `myCanvas`、大小为 300 x 300 且具有厚度为 `1px` 的实线黑色边框的画布。

访问画布元素

定义画布元素仅创建一个空白的绘制图面。但是，要在画布上实际绘制图形对象，需要在 JavaScript 代码中访问画布。您可以在 <BODY> 标记中编写以下代码来访问画布元素：

```
<SCRIPT>
var c=document.getElementByIDgetElementById("myCanvas");
var ctx=c.getContext("2d");
</SCRIPT>
```

在上述代码中，getElementByIDgetElementById() 方法用于返回 ID 为 myCanvas 的元素引用，并将该引用存储在名为 c 的变量中。进而，getContext() 方法用于返回一个绘制上下文对象，它提供在画布上绘制图形所需的方法和属性。该对象存储在名为 ctx 的变量中。getContext() 方法接受指定要创建的画布类型的字符串参数。在上述代码中，2d 作为参数传递到 getContext() 方法，因为大部分浏览器支持在画布内创建 2D 图形。

使用颜色、形状和样式

思考针对学龄前儿童的网站的场景，该网站向这些儿童提供基本的学习解决方案。它提供简单的在线活动来帮助儿童识别基本的颜色和形状。例如，其中一个活动向学生显示颜色选取器和各种形状。作为该活动的一部分，要求学生识别颜色并使用指定颜色填充特定形状。要创建此类应用程序，需要在网页上嵌入颜色、形状和样式。在画布中，这通过使用 JavaScript 预定义的颜色和样式属性和方法来实现。

画布的引入简化了在网页上绘制对象（例如矩形）的任务。您可以使用 JavaScript 方法轻松地绘制这些对象。而且，您还可以指定颜色、渐变或模式样式来装饰画布上的图形对象。

使用形状

创建画布后，您可以在其上绘制形状，例如矩形和正方形。矩形和正方形是可使用 JavaScript 函数在画布元素上绘制的最简单的形状。使用这些函数，您可以创建形状、清除形状的某个部分以及将轮廓线应用到形状。为此，您可以使用以下方法：

- rect()
- fillRect()
- strokeRect()
- clearRect()

rect()

rect() 方法用于在画布上创建矩形。但是，它选取画布的默认颜色来绘制矩形的轮廓线。因此，该矩形在画布上不可见。要使矩形在画布上可见，需要使用 stroke() 方法提供其轮廓线或线条颜色。此方法使用默认的黑色线条来绘制矩形。

使用 rect() 方法在画布上创建矩形的语法为：

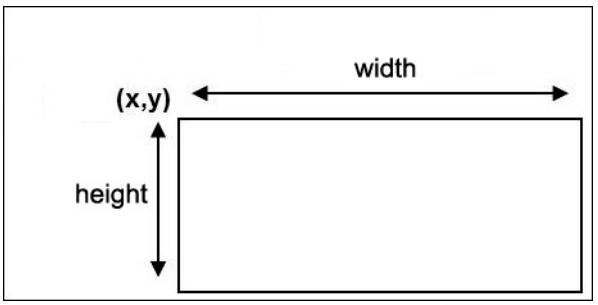
```
rect(x,y,width,height);
```

在上述语法中：

- x：指定矩形左上角的 X 坐标。

- `y`：指定矩形左上角的 Y 坐标。
- `width`：以像素为单位指定矩形宽度。
- `height`：以像素为单位指定矩形的高度。

下图解释了矩形的维度。

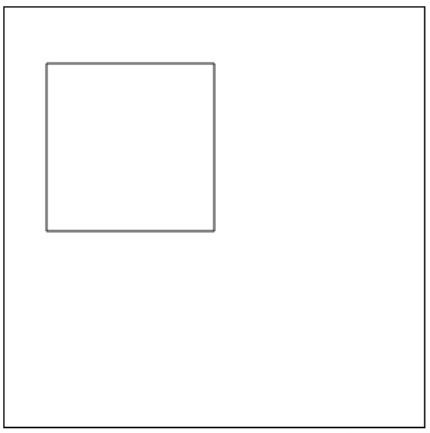


矩形的维度

思考以下代码，它用于在 ID 为 `myCanvas` 的画布上创建矩形：

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas" width="300" height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementByIDgetElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.rect(30,40,120,120);
ctx.stroke();
</SCRIPT>
</BODY>
</HTML>
```

上述代码在画布上创建了开始于坐标 (30, 40) 大小为 120 x 120 的矩形，如下图所示。



在画布创建的矩形

fillRect()

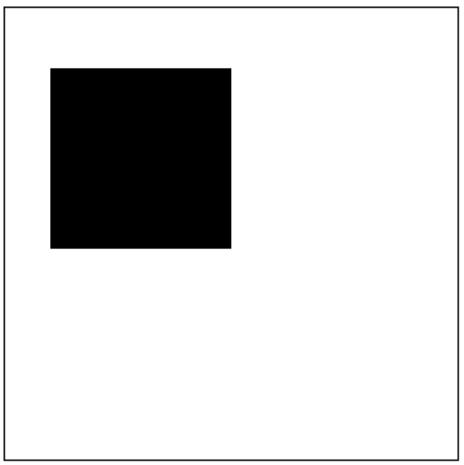
`fillRect()` 方法用于创建使用指定颜色填充的矩形。默认填充颜色为黑色。以下语法用于在画布上创建填色矩形：

```
fillRect(x,y,width,height);
```

思考用于在画布上创建填色矩形的以下代码：

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas" width="300" height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementByIDgetElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.fillRect(30,40,120,120);
</SCRIPT>
</BODY>
</HTML>
```

上述代码将创建填充黑色的矩形，如下图所示。



填充黑色的矩形

strokeRect()

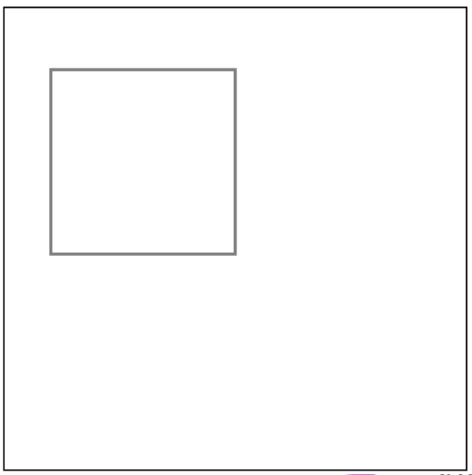
使用 `rect()` 方法创建矩形时，还需要使用 `stroke()` 方法在画布上定义其轮廓线。如果不使用 `rect()` 和 `stroke()` 这两种方法在画布上绘制矩形，您可以使用一个方法 `strokeRect()` 来用指定的线条颜色绘制矩形。默认情况下，`strokeRect()` 方法使用黑色创建矩形的轮廓线。使用 `strokeRect()` 方法在画布上绘制矩形的语法为：

```
strokeRect(x,y,width,height);
```

思考用于在画布上绘制矩形的以下代码：

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas" width="300" height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementByIDgetElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.strokeRect(30,40,120,120);
</SCRIPT>
</BODY>
</HTML>
```

上述代码将在画布上创建位置为 (30, 40) 大小为 120 x 120 的矩形，如下图所示。



使用 strokeRect() 方法创建的矩形

clearRect()

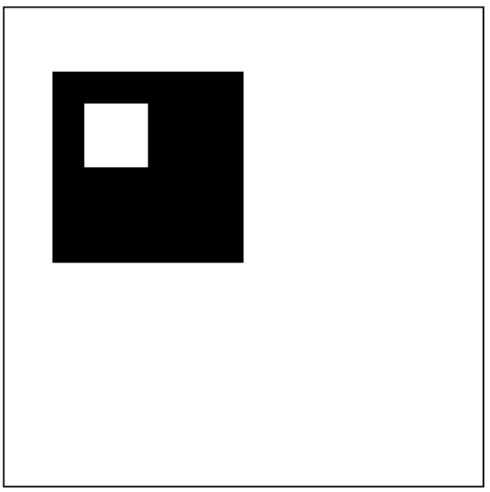
`clearRect()` 方法用于清除矩形的一部分。它清除给定矩形内的指定像素。可使用以下语法清除画布上的矩形：

```
clearRect(x,y,width,height);
```

思考以下代码，它用于清除在 ID 为 `myCanvas` 的画布上创建的矩形的一部分：

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas" width="300" height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementByIDgetElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.fillRect(30,40,120,120);
ctx.clearRect(50,60,40,40);
</SCRIPT>
</BODY>
</HTML>
```

上述代码清除了开始于坐标 (500, 60) 大小为 40 x 40 的矩形的一部分，如下图所示。



使用 clearRect() 方法后的派生输出

现在，思考以下代码，它用于通过使用各种方法在画布上绘制矩形形状：

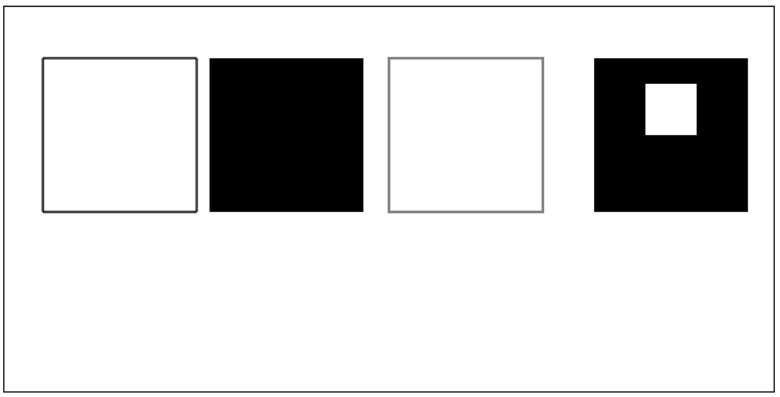
```
<!DOCTYPE HTML>
<HTML>

<BODY>
<CANVAS ID="myCanvas"  width="600" height="300"
style="border:1px solid black">
</CANVAS>
<SCRIPT>
var c=document.getElementByIDgetElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.rect(30,40,120,120);
ctx.stroke();

ctx.fillRect(160,40,120,120);
ctx.stroke();

ctx.strokeRect(300,40,120,120);
ctx.fillRect(460,40,120,120);
ctx.clearRect(500,60,40,40);
</SCRIPT>
</BODY>
</HTML>
```

上述代码在画布上创建了矩形，如下图所示。



在画布创建的矩形

使用颜色

画布上的图形对象是使用默认线条和填充颜色创建的。但是，可以使用除默认颜色以外的颜色来创建图形对象。可使用以下属性将颜色应用到画布对象：

- `fillStyle`
- `strokeStyle`
- `shadowColor`

fillStyle

`fillStyle` 属性用于定义将用于填充在画布上绘制的任何闭合形状的颜色。`fillStyle` 属性的默认值是纯黑色。以下语法用于在图形对象上应用填色样式：

```
fillStyle="color";
```

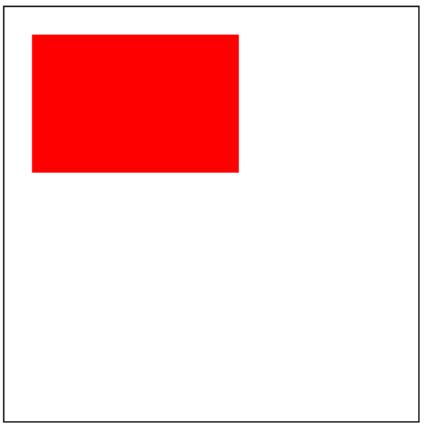
在上述语法中，可以将颜色指定为红色、绿色或蓝色。此外，还可以指定范围介于 000000 到 FFFFFF 的颜色的十六进制值。

思考以下代码，它用于将填色样式应用到在 ID 为 `myCanvas` 的画布上所绘制的矩形：

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas"  width="300" height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementByIDgetElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.fillStyle="red";
ctx.fillRect(20,20,150,100);
</SCRIPT>
</BODY>
```

```
</HTML>
```

上述代码段在画布上创建了位置为 (20, 20) 大小为 150 x 100 的红色矩形，如下图所示。



填充红色的矩形

strokeStyle

`strokeStyle` 属性用于设置在画布上绘制的形状的轮廓线颜色。`strokeStyle` 属性的默认值是纯黑色。以下语法可用于在图形对象上应用线条样式：

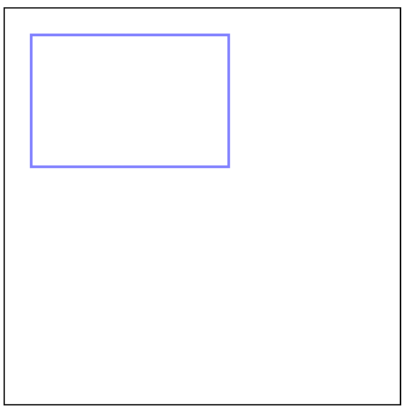
```
strokeStyle="color";
```

在上述语法中，`color` 指定颜色的名称或十六进制值。

思考用于在矩形上应用线条样式的以下代码：

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas" width="300" height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementByIDgetElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.strokeStyle="blue";
ctx.strokeRect(20,20,150,100);
</SCRIPT>
</BODY>
</HTML>
```

上述代码将在画布上创建位置为 (20, 20) 大小为 150 x 100 且轮廓线颜色为蓝色的矩形，如下图所示。



线条颜色为蓝色的矩形

shadowColor

一旦在画布上绘制了形状，您可能想要通过在其上转换阴影来使其样式更为时尚。要转换画布上图形对象的阴影，您需要指定阴影的颜色。此外，您还需要指定您希望阴影模糊的级别。`shadowColor` 属性用于设置图形对象上显示的阴影的颜色，`shadowBlur` 属性用于设置阴影的模糊级别。

可使用以下语法来使用 `shadowColor` 属性：

```
shadowColor="color";
```

在上述语法中，`color` 指定将应用到阴影上的颜色。`shadowColor` 属性的默认值是纯黑色。

可使用以下语法来定义 `shadowBlur` 属性：

```
shadowBlur=number;
```

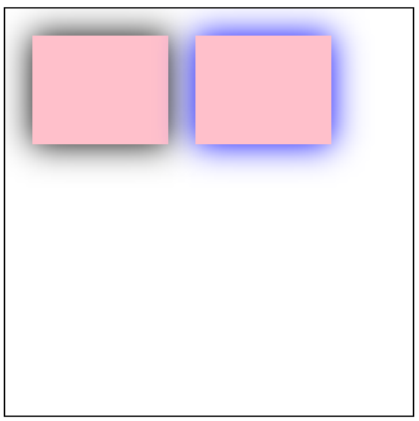
在上述语法中，`number` 指定阴影的模糊级别。它可以接受整数值，例如 `1`、`2` 和 `20`。默认值是 `0`。

思考用于在矩形上应用阴影的以下代码：

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas"   width="300" height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementByIDgetElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.shadowBlur=40;
ctx.fillStyle="pink";
ctx.shadowColor="black";
```

```
ctx.fillRect(20,20,100,80);
ctx.shadowColor="blue";
ctx.fillRect(140,20,100,80);
</SCRIPT>
</BODY>
</HTML>
```

在上述代码中，图形对象的阴影的模糊级别设置为 40。此外，第一个矩形的阴影颜色设置为黑色，第二个矩形的阴影颜色设置为蓝色。将在下图中显示使用 `shadowBlur` 和 `shadowColor` 属性后的派生输出。



使用 shadowBlur 和 shadowColor 属性后的派生输出

使用样式

除了在画布上创建简单的形状，还可以在这些形状上应用样式，例如渐变。渐变是提供在两个或多个颜色之间进行平滑转换的对象。要使用渐变样式，可以使用以下方法：

- `addColorStop()`
- `createLinearGradient()`
- `createRadialGradient()`
- `createPattern()`

addColorStop()

要创建渐变，需要先指定颜色及其在渐变对象中的位置。这是因为在将颜色添加到对象之前，渐变不可见。因此，要在实际上使渐变效果在图形对象上可见，需要添加颜色。您可以使用 `addColorStop()` 方法在渐变对象上添加一个或多个颜色。`addColorStop()` 方法用于指定颜色及其在渐变对象中的相应位置。可使用以下语法来定义 `addColorStop()` 方法：

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

在上述语法中：

- `position`：指定介于 0.0 到 1.0 的值来表示开始和结束渐变颜色的位置。

- color：指定需要应用到各个位置的颜色。

将 addColorStop() 方法与 createLinearGradient() 或 createRadialGradient() 方法一起使用来显示渐变。

createLinearGradient()

createLinearGradient() 方法用于返回表示沿线绘上特定颜色的线性渐变的渐变对象。可使用以下语法应用线性渐变：

```
createLinearGradient(x0,y0,x1,y1);
```

在上述语法中：

- x0：指定渐变开始点的 X 坐标。
- y0：指定渐变开始点的 Y 坐标。
- x1：指定渐变终结点的 X 坐标。
- y1：指定渐变终结点的 Y 坐标。

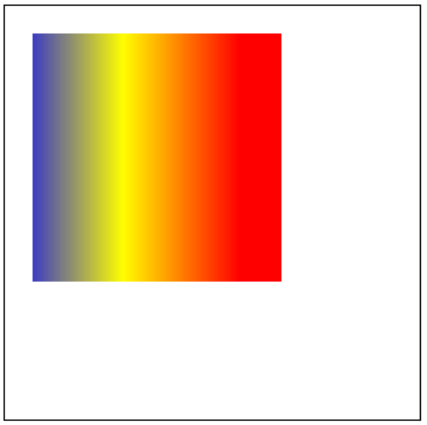
创建线性渐变对象后，您需要使用 addColorStop() 方法创建渐变。一旦创建了线性渐变，就需要使用以下方法将其应用到图形对象：

- 使用 fillStyle 属性用线性渐变填充图形对象。
- 使用 strokeStyle 属性将线性渐变应用到图形对象的轮廓线。

思考用于在矩形上应用线性渐变的以下代码：

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas"   width="300" height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementByIDgetElementById("myCanvas");
var ctx=c.getContext("2d");
var grad=ctx.createLinearGradient(0,0,170,0);
grad.addColorStop(0,"blue");
grad.addColorStop("0.5","yellow");
grad.addColorStop(1,"red");
ctx.fillStyle=grad;
ctx.fillRect(20,20,180,180);
</SCRIPT>
</BODY>
</HTML>
```

在上述代码中，使用 createLinearGradient() 方法创建了渐变对象。进而，使用 addColorStop() 方法向渐变对象指定不同颜色，然后将渐变对象传递到 fillStyle 属性以使用三种不同的颜色从左到右渐变矩形的颜色。将显示使用 createLinearGradient() 方法后的派生输出，如下图所示。



使用 createLinearGradient() 方法后的派生输出

createRadialGradient()

`createRadialGradient()` 方法用于返回表示要应用到图形对象上的径向或圆形渐变的渐变对象。圆形渐变沿由内外两个圆指定的锥形填充颜色。可使用以下语法应用径向渐变：

```
createRadialGradient(x0,y0,r0,x1,y1,r1);
```

在上述语法中：

- `x0`：指定渐变开始点的 X 坐标。
- `y0`：指定渐变开始点的 Y 坐标。(x0,y0) 指定锥形的第一个圈的中心坐标。
- `r0`：指定开始圆的半径。
- `x1`：指定渐变终结点的 X 坐标。
- `y1`：指定渐变终结点的 Y 坐标。(x1,y1) 指定锥形的第二个圈的中心坐标。
- `r1`：指定结束圆的半径。

创建径向渐变对象后，您需要使用 `addColorStop()` 方法创建渐变。一旦创建了径向渐变，就需要使用以下方法将其应用到图形对象：

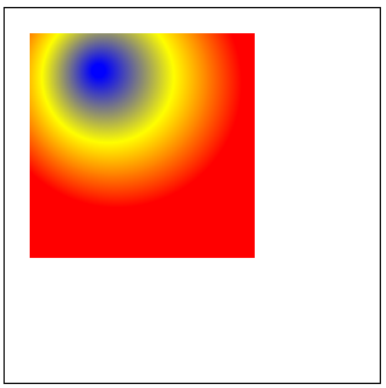
- 使用 `fillStyle` 属性用径向渐变填充图形对象。
- 使用 `strokeStyle` 属性将径向渐变应用到图形对象的轮廓线。

思考用于在矩形上应用径向渐变的以下代码：

```
   <!DOCTYPE HTML>
   <HTML>
   <BODY>
<CANVAS ID="myCanvas"   width="300" height="300"style="border:1px solid black">
   </CANVAS>
   <SCRIPT>
   var c=document.getElementByIDgetElementById("myCanvas");
   var ctx=c.getContext("2d");
   var grad=ctx.createRadialGradient(75,50,5,90,60,100);
   grad.addColorStop(0,"blue");
```

```
grad.addColorStop("0.5","yellow");
grad.addColorStop(1,"red");
ctx.fillStyle=grad;
ctx.fillRect(20,20,180,180);
</SCRIPT>
</BODY>
</HTML>
```

在上述代码中，使用 `createRadialGradient()` 方法创建了渐变对象。进而，使用 `addColorStop()` 方法向渐变对象指定不同颜色，然后将渐变对象传递到 `fillStyle` 属性以使用三种不同的颜色沿圆的给定半径渐变矩形的颜色。将在下图中显示使用 `createRadialGradient()` 方法后的派生输出。



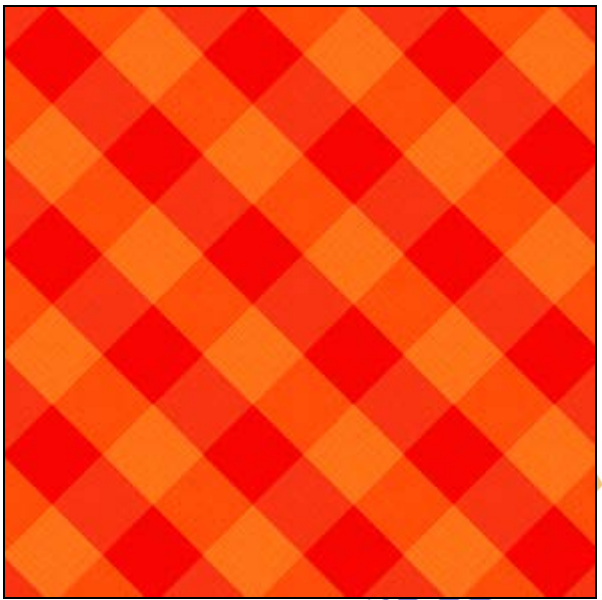
使用 createRadialGradient() 方法后的派生输出

createPattern()

`createPattern()` 方法用于通过按指定方向在画布上重复显示图像来创建模式。例如，思考以下图像。

图像

如果按垂直和水平方向重复上述图像，那么您可以创建一个模式，如下图所示。



模式

可使用以下语法来创建模式：

```
createPattern(img, "repeat|repeat-x|repeat-y|no-repeat");
```

在上述语法中：

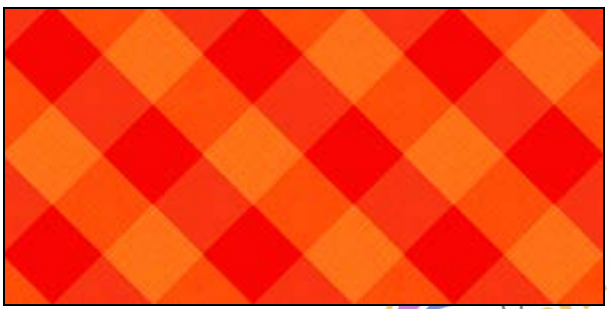
- `img`：指定将用于创建模式的图像或视频。
- `repeat`：指定模式应按水平方向和垂直方向重复。
- `repeat-x`：指定模式应按水平方向重复。
- `repeat-y`：指定模式应按垂直方向重复。
- `no-repeat`：指定模式应仅显示一次。

思考用于按水平方向和垂直方向重复图像的以下代码段：

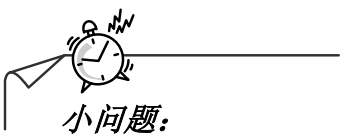
```
<P>Image to use:</P>
<IMG src="pattern.png" ID="pattern">
<P>Canvas:</P>
<BUTTON onclick="draw('repeat')">Repeat</BUTTON>
<CANVAS ID="myCanvas" width="300" height="150" style="border:1px solid
#d3d3d3;">
Your browser does not support the HTML5 canvas tag.</CANVAS>
<SCRIPT>
function draw(direction)
{
var c=document.getElementByIDgetElementById("myCanvas");
var ctx=c.getContext("2d");
var img=document.getElementByIDgetElementById("pattern")
```

```
var pat=ctx.createPattern(img,direction);
ctx.rect(0,0,300,150);
ctx.fillStyle=pat;
ctx.fill();
}
</SCRIPT>
```

当用户单击 **Repeat** 按钮时，上述代码段将按水平方向和垂直方向在画布上的矩形区域中重复 `pattern.png` 图像，如下图所示。



画布上的重复图像



小问题：

以下哪个方法可用于移除画布上矩形的某个部分？

1. *rect()*
2. *createRect()*
3. *strokeRect()*
4. *clearRect()*

答案：

4.clearRect()

使用路径、文本和图像

LearnGraphs Ltd. 提供有关创建饼图的教程。为此，需要绘制饼图及其标题。但是，饼图是划分为各种扇区的圆形图表。要创建饼图，需要绘制一个圆。此外，要将圆划分为几个扇区，需要绘制线。在画布中，您可以使用路径方法和属性创建形状，例如圆、线、弧和三角形。

进而，要插入饼图的标题，需要插入文本。可以使用文本属性和方法在画布中插入文本。此外，还可以向画布添加图像。

使用路径

路径是一系列连接在一起以创建线或形状的点。在画布中，可以使用线或路径绘制除矩形或正方形之外的形状。使用路径或线，可以创建诸如圆、多边形或三角形之类的形状。但是，要创建路径，需要先开始路径。接下来，需要调用方法（例如 `moveTo()` 和 `lineTo()`）以实际绘制路径。最后，需要结束或闭合路径以便创建出形状。要使用路径创建形状，可以使用各种方法。下表列出了这些方法。

方法	*描述*
fill()	*用于填充画布上的路径。默认颜色为黑色。您可以使用 `fillStyle` 属性更改填充颜色。*
stroke()	*用于绘制路径的轮廓线。默认颜色为黑色。您可以使用 `strokeStyle` 属性更改轮廓线颜色。*
beginPath()	*用于开始路径或重置当前路径。*
moveTo(x,y)	*用于将路径移动到画布上的 (x,y) 坐标处。*
closePath()	*用于创建从当前位置返回到起始位置的路径。*
lineTo(x,y)	*用于创建从起始位置到由 (x,y) 坐标指定的结束位置的线。起始位置由 `moveTo()` 方法中指定的 (x,y) 坐标定义。*
clip()	*用于从画布裁剪指定区域。*
arc(x,y,r,sAngle, eAngle,counterclockwise)	*用于在画布上创建一个弧或曲线，其坐标点为 (x,y)、半径为 r、起始角度为 sAngle、结束角度为 eAngle。最后一个参数指定该图形是应逆时针创建还是顺时针创建。false 值指定顺时针，true 值指定逆时针。*
arcTo(x1,y1,x2,y2,r)	*用于在画布上创建位于两个坐标点 (x1,y1) 和 (x2,y2) 之间半径为 r 的弧。*

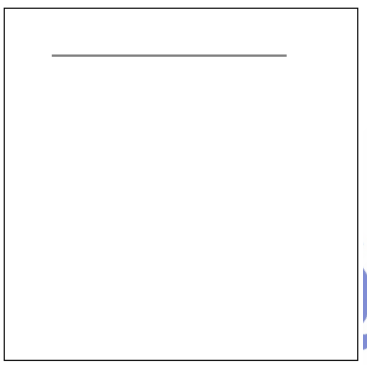
路径方法

思考用于在画布上创建线的以下代码：

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas" width="300" height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
```

```
var c=document.getElementByIDgetElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.beginPath();
ctx.moveTo(40, 40);
ctx.lineTo(240, 40);
ctx.stroke(); </SCRIPT>
</BODY>
</HTML>
```

在上述代码中，`beginPath()` 方法将在画布上开始路径。接着，`lineTo()` 方法将绘制一条始于 `moveTo()` 方法中指定的画布坐标 (40,40) 结束于坐标 (2340, 40) 的直线，如下图所示。

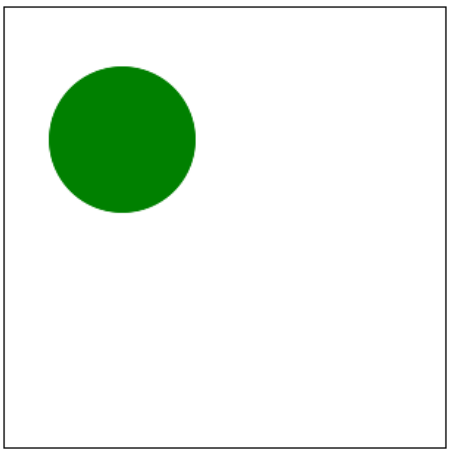


在画布上绘制的线

思考用于在画布上创建圆的以下代码：

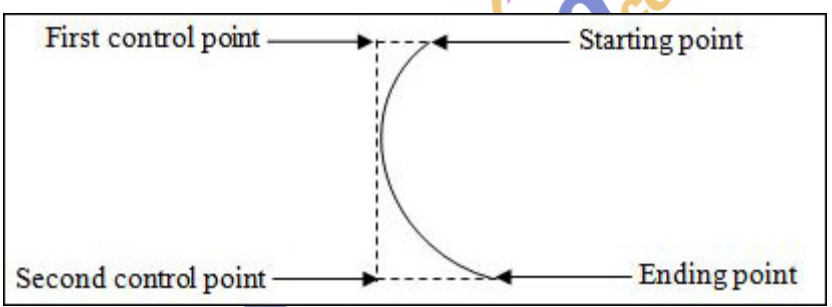
```
<!DOCTYPE HTML>
<HTML>   <BODY>
<CANVAS ID="myCanvas"   width="300" height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementByIDgetElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.beginPath();
ctx.arc(80, 90, 50, 0, Math.PI*2, false);
ctx.closePath();
ctx.fillStyle="green";
ctx.fill();
</SCRIPT>
</BODY>
</HTML>
```

在上述代码中，在画布上坐标 (80, 90) 处绘制了半径为 50 填充了绿色的圆，如下图所示。



使用 arc() 方法后的派生输出

除了绘制线、弧或圆，还可以绘制更复杂的曲度，例如贝塞尔曲线。贝塞尔曲线是使用一个上下文或起点、两个控制点和一个结束点定义的，如下图所示。



贝塞尔曲线

可以使用路径方法 `bezierCurveTo()` 创建贝塞尔曲线。可使用以下语法创建贝塞尔曲线：

```
bezierCurveTo(cp1x,cp1y,cp2x,cp2y,x,y);
```

在上述语法中：

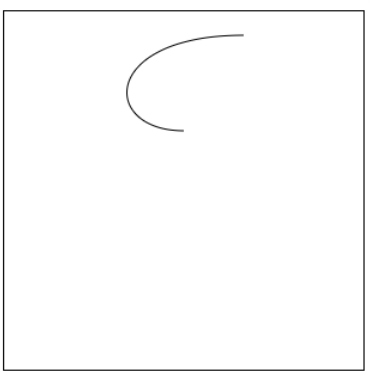
- `cp1x`：指定第一个控制点的 X 坐标。
- `cp1y`：指定第一个控制点的 Y 坐标。
- `cp2x`：指定第二个贝塞尔控制点的 X 坐标。
- `cp2y`：指定第二个贝塞尔控制点的 Y 坐标。
- `x`：指定结束点的 X 坐标。
- `y`：指定结束点的 Y 坐标。

思考以下代码：

```
<!DOCTYPE HTML>
<HTML>
```

```
<BODY>
<CANVAS ID="myCanvas"   width="300" height="300"
style="border:1px solid black">
</CANVAS>
<SCRIPT>
var c=document.getElementByIDgetElementById('myCanvas');
var ctx=c.getContext('2d');
ctx.beginPath();
ctx.moveTo(200,20);
ctx.bezierCurveTo(80,20,80,100,150,100);
ctx.stroke();
</SCRIPT>
</BODY>
</HTML>
```

在上述代码中，绘制了开始于点 (200, 20) 结束于点 (150, 100) 的贝塞尔曲线，如下图所示。



贝塞尔曲线

使用文本

除了在画布上绘制形状或线，还可以在其上绘制文本。您还可以在文本上应用不同的样式。为此，可以使用各种文本属性。下表列出了文本属性。

属性	***值***	***描述***
font	*font-style|font-variant|font-weight|font-size|font-family*	*用于设置画布上文本的字体。*
textAlign	*start|end|center|left|right*	*用于设置文本的对齐。*
textBaseLine	*top|hanging|middle|bottom|alphabetic*	*用于设置相对于文本的起点将在该处绘制文本的基线。*

文本属性

可以使用上述属性来装饰文本。但是，您需要使用以下步骤才能实际在画布上绘制文本：

- `fillText()`

- strokeText()

fillText()

fillText() 方法用于在画布上绘制填充了纯色的文本。fillText() 的默认值为黑色。可使用以下语法绘制填色文本：

```
fillText(text,x,y,width);
```

在上述语法中：

- text：指定要在画布上编写的文本。
- x：指定文本起点的 X 坐标。
- y：指定文本起点的 Y 坐标。
- width：指定文本宽度。

思考用于在画布上绘制填色文本的以下代码：

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas"   width="200" height="200" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementByIDgetElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.font="15px Georgia";
ctx.fillText("Fill Text",10,60,120);
ctx.font="15x Verdana";
var gradient=ctx.createLinearGradient(0,0,170,0);
gradient.addColorStop("0","magenta");
gradient.addColorStop("0.5","blue");
gradient.addColorStop("1.0","red");
ctx.fillStyle=gradient;
ctx.fillText("Fill Text with Gradient",10,90,120);
</SCRIPT>
</BODY>
</HTML>
```

上述代码使用 font 属性将文本的字体样式设置为 15px Georgia，并在画布上的画布坐标 (10, 60) 处绘制文本 Fill Text。同样，它在画布坐标 (10, 90) 处绘制了文本 Fill Text with Gradient，并使用 createLinearGradient() 方法和 fillStyle 属性在其上应用渐变，如下图所示。

使用 fillText() 属性后的派生输出

strokeText()

`strokeText()` 方法用于通过使用当前字体和颜色在画布上的指定位置绘制文本。该方法用于在画布上绘制文本的默认轮廓线颜色是黑色。使用 `strokeText()` 方法的语法是：

```
strokeText(text,x,y,width);
```

在上述语法中：

- `text`：指定要在画布上编写的文本。
- `x`：指定文本起点的 X 坐标。
- `y`：指定文本起点的 Y 坐标。
- `width`：指定文本宽度。

思考以下代码：

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas"   width="300" height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementByIDgetElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.font="25px Georgia";
ctx.strokeText("Stroke Text",10,60,120);
ctx.font="25x Verdana";
var gradient=ctx.createLinearGradient(0,0,170,0);
gradient.addColorStop("0","magenta");
gradient.addColorStop("0.5","blue");
gradient.addColorStop("1.0","red");
ctx.strokeStyle=gradient;
ctx.strokeText("Stroke Text with Gradient",10,90,200);
</SCRIPT>
</BODY>
```

```
</HTML>
```

上述代码使用 font 属性将文本的字体样式设置为 25px Georgia，并在画布上的画布坐标 (10,60) 处绘制文本 Stroke text。同样，它在画布坐标 (10,90) 处绘制了文本 Stroke Text with Gradient，并使用 createLinearGradient() 方法和 strokeStyle 属性在其上应用渐变，如下图所示。

使用 strokeText() 方法后的派生输出

使用图像

在画布中，还可以绘制图像、图像剪辑或渲染视频。为此可使用 drawImage() 方法。drawImage() 方法用于在画布上绘制图像或视频。此外，它还使您能够在画布上绘制图像的一部分。要定义 drawImage() 方法，可使用以下语法：

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

在上述语法中：

- img：指定需要在画布上绘制的图像。
- x：指定图像起点的 X 坐标。
- y：指定图像起点的 Y 坐标。
- width：指定图像宽度。这是一个可选参数。如果未指定，将默认采用图像的实际宽度。
- height：指定图像高度。如果未指定，将默认采用图像的实际高度。
- sx：指定图像的裁剪起点的 X 坐标。
- sy：指定图像的裁剪起点的 Y 坐标。
- swidth：指定裁剪的图像的宽度。
- sheight：指定裁剪的图像的高度。

思考用于在画布上绘制图像的以下代码：

```
<!DOCTYPE HTML>
<HTML>
```

```
<BODY onload=setImage()>
<CANVAS ID="myCanvas" width="250" height="300" style="border:1px solid
#d3d3d3;">
Your browser does not support the HTML5 canvas tag.</CANVAS>
<SCRIPT>
function setImage(){
var c=document.getElementByIDgetElementById("myCanvas");
var ctx=c.getContext("2d");
var img=new Image();
img.src="orchid.jpg";
img.onload = function() {
ctx.drawImage(img,10,10,150,150);
}
}
</SCRIPT>
</BODY>
</HTML>
```

在上述代码中，在画布上绘制了位于点 (10, 10) 且大小为 150 x 150 的图像，如下图所示。

在画布上绘制的图像

使用图形

图形是表示一组项之间关系的方式。图形包括一组称为节点的对象，通过称为边缘的链接连接。在画布中，您可以创建图形（例如条形图或饼图）来表示关系。可以使用为绘制形状而提供的方法来创建这些图形。但是，通过使用这些方法绘制图形是一项繁琐的任务。要简化该任务，可以使用各种可免费下载的 JavaScript 库。其中一个此类库就是 RGraph。要使用 RGraph 库创建图形，必须下载此轻量级的 JavaScript 库并将其保存在您的系统中。RGraph 允许您在网页上创建不同类型的图形。

一旦下载了 JavaScript 库，网页就可以通过使用 Web 文档标头部分中的 `<SCRIPT>` 标记引用它。

以下语法用于指定 jQuery 库：

```
<SCRIPT type="text/javascript" src="<RGraph_file_name>" ></SCRIPT>
```

在上述语法中：

- `<script SCRIPT>` 标记告知浏览器 HTML 文档使用脚本。

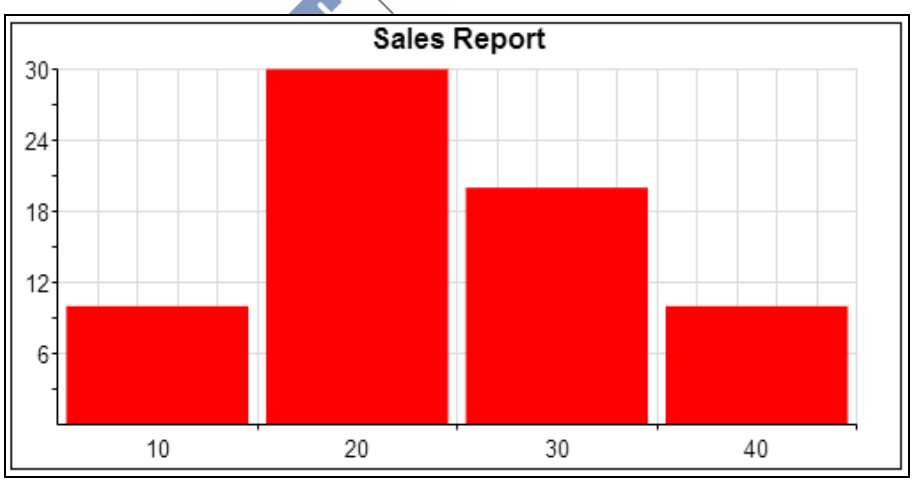
- `type` 标记属性指定所使用的脚本的类型。
- `src` 标记属性指定所使用的 JavaScript 库的名称。如果 JavaScript 库和 HTML 文档存储在同一位置，那么可以仅提供库的名称。但是，如果 JavaScript 库和 HTML 文档存储在不同位置，那么必须指定 JavaScript 库的完整路径。

例如，您想在网页上创建一个条形图。为此，您需要下载 **RGraph.common.core.js** 和 **RGraph.bar.js** 文件。思考以下代码段，它用于在网页上创建条形图：

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<SCRIPT type="text/javascript" src="RGraph.common.core.js" ></SCRIPT>
<SCRIPT type="text/javascript" src="RGraph.bar.js" ></SCRIPT>
<TITLE>Bar chart</TITLE>
</HEAD>
<BODY>

<CANVAS width="500" height="250" ID="test" style="border:1px solid
black"></CANVAS>
<SCRIPT type="text/javascript" charset="utf-8">
var bar = new RGraph.Bar('test', [10,30,20,10]);
bar.Set('chart.gutter', 5);
bar.Set('chart.colors', ['red']);
bar.Set('chart.title', "Sales Report" );
bar.Set('chart.labels', ["10" , "20" , "30" , "40" ]);
bar.Draw();
</SCRIPT>
</BODY>
</HTML>
```

上述代码在画布上创建了一个图形，如下图所示。



条形图

注释

要在网页上创建不同的图形，需要下载不同的 JavaScript 文件。为此，您可以参考 http://www.rgraph.net/demos 链接。

小问题：

以下哪个方法用于在画布上绘制路径的轮廓线？

1. `fill()`
2. `stroke()`
3. `lineTo()`
4. `moveTo()`

答案：

2.`stroke()`

活动 6.1：介绍画布

变换和动画显示画布元素

LearnGraphs Ltd. 想要使图形对于用户在视觉上具有吸引力。为此，他们需要动画制作或变换在画布上绘制的形状，例如缩放和旋转线。要应用此类动画，需要对形状应用动画效果。使用画布，可以变换、旋转或缩放图形对象。

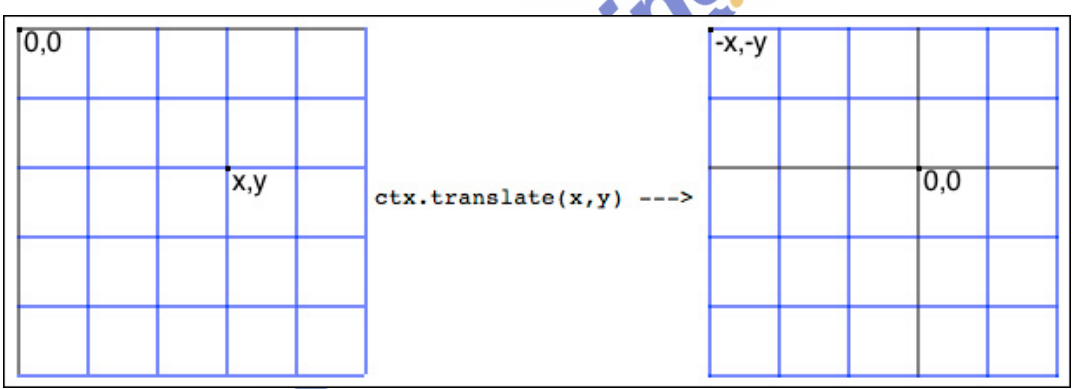
变换画布元素

使用 CSS，可以很轻松地将 HTML 元素从一个位置移动到其他位置。但是，在使用画布时，您可能想要将图形对象从一个位置移动到其他位置并增加或减少其大小。这可通过变换画布元素来实现。要变换画布元素，可以使用以下方法：

- `translate()`
- `scale()`
- `rotate()`

转换

`translate()` 方法用于将画布的原点重置到指定位置，如下图所示。



画布原点的重置

在上图中，画布的原点移动到 (x, y) 坐标。`translate()` 方法使您能够只使用一种方法就可移动画布上的所有图形对象。例如，您在画布上绘制了一个复杂的图形，您需要将该图形在画布上到处移动。要执行此类任务，您需要调整该图形中包括的所有图形对象的 x 和 y 位置。这是一项耗时且容易出错的任务。您可以通过以下方法简化此任务：将画布的上下文或原点转换到新位置，然后使用所有图形对象的 x 和 y 位置在画布上重新绘制它们。通过这种方法，将根据新原点的位置重新绘制图形对象。因此，通过只使用一种方法调用，就可将图形对象移动到画布上的所需位置。

`translate()` 方法的语法是：

```
translate(x,y);
```

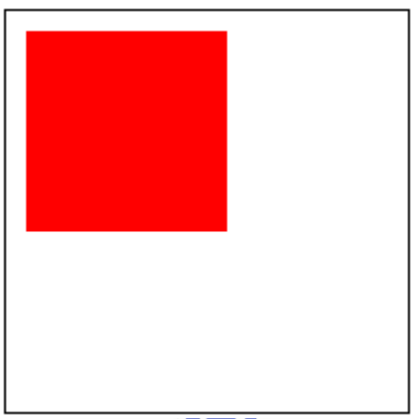
在上述语法中：

- x：指定要添加到 X 坐标的现有值的值。
- y：指定要添加到 Y 坐标的现有值的值。

例如，您使用以下代码段创建了一个矩形：

```
var c=document.getElementByIDgetElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.fillStyle="red";
ctx.fillRect(10,10,100,100);
```

上述代码将在坐标 (10, 10) 处绘制一个矩形，如下图所示。



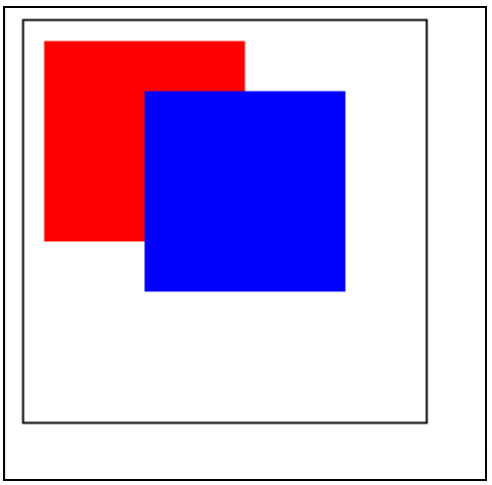
矩形

现在，您希望应从坐标 (60, 35) 开始重新绘制该矩形。为此可使用 `translate()` 方法，如以下代码段所示：

```
var c=document.getElementByIDgetElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.fillStyle="red";
ctx.fillRect(10,10,100,100);
ctx.translate(50,25);
ctx.fillStyle="blue";
ctx.fillRect(10,10,100,100);
```

在上述代码段中，从画布的原点开始在点 (10, 10) 处以红色绘制了第一个矩形。然后 `translate()` 方法在点 (50, 25) 处重置了画布的原点。接着从新原点 (50, 25) 开始在点 (10, 10) 处以蓝色绘制了第二个矩形。

因此，第二个矩形将从画布的点 (0, 0) 开始在点 (60, 35) 处绘制，如下图所示。



使用 translate() 方法后的派生输出

缩放

`scale()` 方法用于增加或减少画布网格中的单元。这将允许您绘制缩小的或放大的图形对象。可使用以下语法缩放图形对象：

```
scale(scalewidth,scaleheight);
```

在上述语法中：

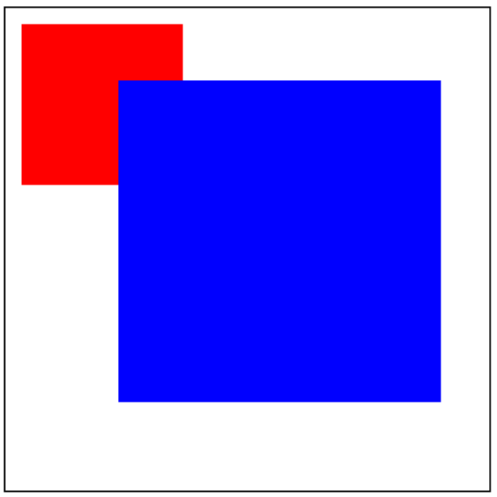
- `scalewidth`：指定图形对象应缩放至的宽度（以百分比计）。
- `scaleheight`：指定图形对象应缩放至的长度（以百分比计）。

思考用于在画布上缩放矩形的以下代码：

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas"  width="300" height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementByIDgetElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.fillStyle="red";
ctx.fillRect(10,10,100,100);
ctx.translate(50,25);
ctx.scale(2,2);
ctx.fillStyle="blue";
ctx.fillRect(10,10,100,100);
</SCRIPT>
</BODY>
</HTML>
```

在上述代码中，从画布坐标 (10, 10) 处开始创建了大小为 100 x 100 的红色矩形。然后通过使用 `translate()` 方法将画布的原点重置到点 (50, 25) 处。接着，使用 `scale()` 方法将矩形的宽度和高

度缩放 2 倍。因此，蓝色矩形的宽度和高度增大到两倍，并从画布坐标点 (60, 375) 处重新绘制，如下图所示。



使用 scale() 方法后的派生输出

旋转

`rotate()` 方法用于将图形对象按顺时针方向旋转到指定角度。可使用以下语法应用旋转：

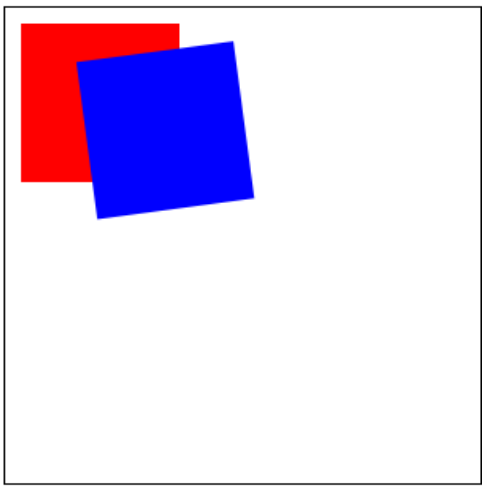
```
rotate(angle);
```

在上述语法中，`angle` 指定应将图形对象旋转的角度。

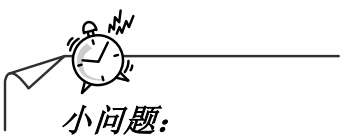
思考用于在画布上应用旋转的以下代码：

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas"  width="300" height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementByIDgetElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.fillStyle="red";
ctx.fillRect(10,10,100,100);
ctx.rotate(25);
ctx.fillStyle="blue";
ctx.fillRect(40,40,100,100);
</SCRIPT>
</BODY>
</HTML>
```

在上述代码中，在坐标 (10, 10) 处绘制了红色的矩形。接着，将画布元素的旋转角度指定为 25。因此，将以蓝色重新绘制开始于坐标 (40, 40) 并旋转到 25 度角度的矩形，如下图所示。



使用 rotate() 方法后的派生输出



小问题：

以下哪个方法用于增加图形元素的大小？

1. `translate()`
2. `scale()`
3. `rotate()`
4. `transform()`

答案：

2.scale()

动画显示画布元素

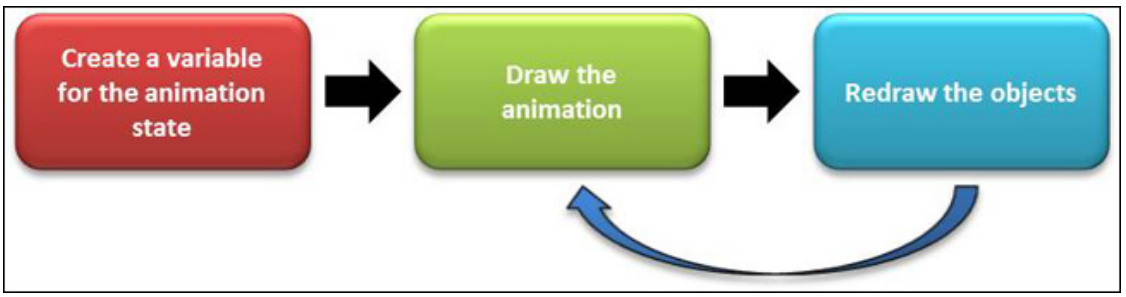
动画是一项可视技术，通过以快速的序列显示图形对象来提供对象正在移动的影像。在动画中，首先绘制对象，接着清除画布，然后通过稍微更改其属性重新绘制框架。该过程非常快速地重复，以创建一种影像。例如，您想要在画布上表示一个正在滚动的球。您已在球上应用了一个显示变换的旋转。但是，如果您在该球上应用多个旋转，则将创建动画。

要创建此类动画，需要执行以下步骤：

1. 为动画状态创建变量。
2. 绘制动画。

3. 重新绘制对象。

下图描述了在画布上创建动画的过程。



在画布上创建动画的过程

为动画状态创建变量

变量的创建目的是存储图形对象的初始属性，例如宽度或高度。而且，他们可用于存储图形对象的更新值。因此，只要为图形对象计算了新位置，该新值将存储在变量中。

思考创建在画布图面上滚动的球的示例。要创建此类动画，需要先定义变量：思考以下用于在 JavaScript 中定义变量的代码段：

```
var canvas;
var ctx;
var x = 400;
var y = 300;
var dx = 2;
var dy = 4;
var WIDTH = 400;
var HEIGHT = 300;
```

在上述代码段中，创建了可用于在画布上绘制动画的变量。

绘制动画

创建变量后，需要在屏幕上绘制图形对象。思考用于在画布上绘制球的以下代码段：

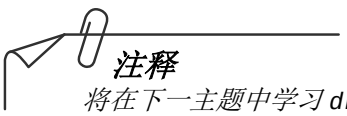
```
function circle() {
ctx.beginPath();
ctx.fillStyle="red";
ctx.arc(x, y, 10, 0, Math.PI*2, true);
ctx.fill();
ctx.closePath();
}

function clear() {
ctx.clearRect(0, 0, WIDTH, HEIGHT);
}
function init() {
canvas = document.getElementByIDgetElementById("canvas");
ctx = canvas.getContext("2d");
```

```
return setInterval(draw, 10);
}
```

上述代码段定义了以下函数：

- `init()` 函数：在此函数中，画布元素的元素 ID 存储在变量 `canvas` 中。接着，在变量 `ctx` 中检索画布元素的上下文。现在上下文 `ctx` 可用于在画布上绘制图形对象。它是将在代码中调用的第一个函数。在下一行代码中，`setInterval()` 函数用于每 10 毫秒调用一次 `draw()` 函数。`draw()` 函数是用户定义函数，其创建目的是用于在画布上重新绘制球。



注释

将在下一主题中学习 `draw()` 函数的代码。

- `circle()` 函数：在此函数中，调用了 `beginpath()` 方法来开始新路径。`arc()` 方法用于定义圆的大小和形状。接着，`fill()` 方法用于使用指定颜色填充整个圆。
- `clear()` 函数：在此函数中，调用了 `clearRect()` 方法来擦除画布上的图形对象。

重新绘制对象

现在，要创建动画效果，需要使用更新的属性不断重复绘制图形对象。

思考用于在画布上重新绘制球的以下代码段：

```
function draw() {
clear();
circle();
if (x> WIDTH || x< 0)
dx = -dx;
if (y> HEIGHT || y< 0)
dy = -dy;

x += dx;
y += dy;
}
```

在上述代码段中，首先调用了 `clear()` 函数来清除画布。接着，调用了 `circle()` 函数来在画布上创建一个圆。

该圆的半径为 10，原点位于 (x,y)。要移动该圆，需要更改变量 x 和 y 的值。只要执行了 `draw()` 函数，变量 dx 和 dy 就将确定变量 x 和 y 的值。如果 x 的值不大于画布的宽度或者该值小于零，那么 dx 将更改 x 的值。但是，如果该值超过了宽度大小，那么变量 dx 将设置为 -dx，过程继续。相同情况适用于变量 y。

以下行显示了旋转球的全部代码：

```
<!DOCTYPE HTML>
<HTML>
```

```
<HEAD>
<TITLE>Rotating Ball</TITLE>
</HEAD>
<BODY>
<DIV>
<CANVAS ID="canvas" width="400" height="300" style="border:2px solid black">
</CANVAS>
</DIV>

<SCRIPT type="text/javascript">
var canvas;
var ctx;
var x = 400;
var y = 300;
var dx = 2;
var dy = 4;
var WIDTH = 400;
var HEIGHT = 300;

function circle() {
ctx.beginPath();
ctx.fillStyle="red";
ctx.arc(x, y, 10, 0, Math.PI*2, true);
ctx.fill();
ctx.closePath();
}

function clear() {
ctx.clearRect(0, 0, WIDTH, HEIGHT);
}

function init() {
canvas = document.getElementByIDgetElementById("canvas");
ctx = canvas.getContext("2d");
return setInterval(draw, 10);
}

function draw() {
clear();
circle();
if (x > WIDTH || x < 0)
dx = -dx;
if (y > HEIGHT || y < 0)
dy = -dy;


x += dx;
y += dy;
}

init();
</SCRIPT>
</BODY>
</HTML>
```

活动 6.2：创建游戏

练习问题

1. 以下哪个路径属性用于创建从当前位置返回到起始位置的路径？

   a. `beginPath()`
   b. `closePath()`
   c. `lineTo()`
   d. `moveTo()`

2. 指出在画布上应用径向渐变的正确语法。

   a. `createRadialGradient(x0,y0,r0);`
   b. `createRadialGradient(x0,y0,x1,y1);`
   c. `createRadialGradient(r0,r1);`
   d. `createRadialGradient(x0,y0,r0,x1,y1,r1);`

3. 以下哪个方法用于将画布的原点重置到指定位置？

   a. `rotate()`
   b. `translate()`
   c. `scale()`
   d. `fill()`

4. 以下哪个方法用于返回画布上的绘制上下文对象？

   a. `getContext()`
   b. `getElementByIDgetElementById()`
   c. `write()`
   d. `writeln()`

5. 以下哪个方法用于指定渐变对象中的颜色？

   a. `colorStop()`
   b. `addColorStop()`
   c. `addColor()`
   d. `color()`

小结

在本章中，您学习了：

- 画布提供一种简单有效的方式来在网页上创建图形。
- 画布是使用 <CANVAS> 标记定义的。该标记在 HTML 文档的正文部分中定义。
- 定义画布元素仅创建一个空白的绘制图面。但是，要在画布上实际绘制图形对象，需要在 JavaScript 代码中访问画布。
- 可以使用以下方法在画布上绘制形状：
  - `rect()`
  - `fillRect()`
  - `strokeRect()`
  - `clearRect()`
- 可使用以下属性将颜色应用到画布对象：
  - `fillStyle`
  - `strokeStyle`
  - `shadowColor`
- 除了在画布上创建简单的形状，还可以在这些形状上应用样式，例如渐变。
- 路径是一系列连接在一起以创建线或形状的点。在画布中，可以使用线或路径绘制除矩形或正方形之外的形状。
- 除了在画布上绘制形状或线，还可以在其上绘制文本。您还可以在文本上应用不同的样式。
- `drawImage()` 方法用于在画布上绘制图像或视频。
- 在画布中，您可以创建图形（例如条形图或饼图）来表示关系。
- 要变换画布元素，可以使用以下方法：
  - `translate()`
  - `scale()`
  - `rotate()`
- 要创建此类动画，需要执行以下步骤：
  - 为动画状态创建变量。
  - 绘制动画。
  - 重新绘制对象。