

H5 动画与游戏开发

---H5 Canvas基础

内容纲要

- **Canvas案例概览**
- **Canvas尺寸问题**
- **Canvas图形上下文**

Canvas 简单案例

- 获得Canvas元素节点

- var theCanvas = document.getElementById('canvas');
- theCanvas.width = 300; theCanvas.height = 150;

- 获得和设置Canvas的图形上下文

- var context = theCanvas.getContext('2d');//返回CanvasRenderingContext2D对象
- context.fillStyle = "#ff0000";

- 通过Canvas API绘制图形图像

- context.fillRect(0, 0, 100, 100); //context.strokeRect(0, 0, 100, 100); 注意顺序的影响
- context.clearRect(25, 25, 50, 50);



内容纲要

- Canvas案例概览
- Canvas尺寸问题
- Canvas图形上下文



Canvas 的尺寸大小（元素大小、绘画表面大小）

- Canvas元素样式的大小（若不设置，则大小同绘图表面大小）

```
<canvas id="CanvasOne"></canvas>
```

```
#CanvasOne{ /*通过css样式设置Canvas元素大小*/
```

```
    width: 200px; height: 200px;
```

```
}
```

- Canvas绘图表面的大小（默认宽为300，高为150）

```
<canvas id="CanvasOne"></canvas>
```

```
<canvas id="CanvasTwo" width="200" height="100"></canvas>
```

- Canvas元素大小和绘制表面大小不一致时（产生拉伸现象）

内容纲要

- Canvas案例概览
- Canvas尺寸问题
- Canvas图形上下文



Canvas 图形上下文 (context)

- Canvas区别与Flash、SVG，Canvas不是对绘制对象进行操作，而是基于状态的绘制（回顾：即使模式与保留模式）
- Canvas状态存储在图形上下文context中
 - 上下文属性（描边样式、填充样式、全局透明度、线宽、线连接方式等）
 - 变换矩阵信息（平移、旋转、缩放）
 - 剪贴区域：通过clip()方法创建的
- context原型链及继承关系
 - context的构造器（constructor）为CanvasRenderingContext2D
 - context对象的原型为CanvasRenderingContext2D.prototype

Canvas 图形上下文的状态存储

- context 保存与恢复绘图上下文状态 (Save、Restore)
 - 保存上下文状态到栈中，save之后，可调用平移、放缩、旋转、裁剪等操作
 - 恢复上下文之前保存的状态，防止save后对Canvas执行的操作对后续的绘制有影响

- context 状态保存及恢复案例，理解状态堆栈

```
context.fillStyle = "red" ;
```

```
context.save( );
```

```
context.fillStyle = "blue" ; context.fillRect(0,0,100,100);
```

```
context.restore( );
```

```
context.fillRect(100,100,100,100);
```


H5 动画与游戏开发

---H5 Canvas基础绘图

内容纲要

- **Canvas路径绘制**
- **Canvas基本形状**
- **Canvas描边及填充**



Canvas 图形及路径

- Canvas直线相关绘制

```
context.moveTo(100,100)
```

```
context.lineTo(200,200)
```

```
context.lineWidth = 10
```

```
context.strokeStyle = "#058"
```

} 状态设置

```
context.stroke()
```

} 绘制

- 也可以通过Path2D构造函数创建路径对象

Canvas 图形及路径

- 理解beginPath的作用（绘制多条直线时可能遇到的问题）
 - canvas中的绘制方法（如stroke,fill），都会以“上次beginPath”之后的所有路径为基础进行绘制，不管你用moveTo把画笔移动到哪里，只要不beginPath，都是在画一条路径，fillRect与strokeRect这种直接画出独立区域的函数，也不会打断当前的path
- 理解closePath的作用
 - closePath的意思不是结束路径，而是关闭路径，它会试图从（MoveTo点之后）当前路径的终点连一条路径到起点，让整个路径闭合起来。但是，这并不意味着它之后的路径就是新路径了
- beginPath和closePath的作用不同，并不是必须成对出现

Canvas 图形及路径

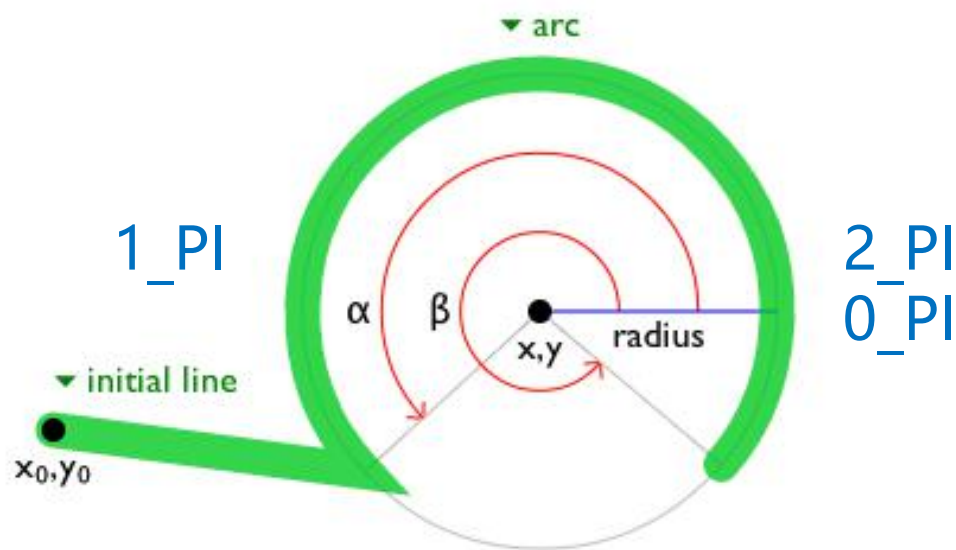


- 线宽及颜色样式
 - lineWidth、strokeStyle、fillStyle
 - stroke ()、fill ()
- 线端点样式 (lineCap)
 - butt (default)、round、square
- 连接点样式 (lineJoin)
 - miter (default)、bevel、round
- 样式与上下文状态 (context status)

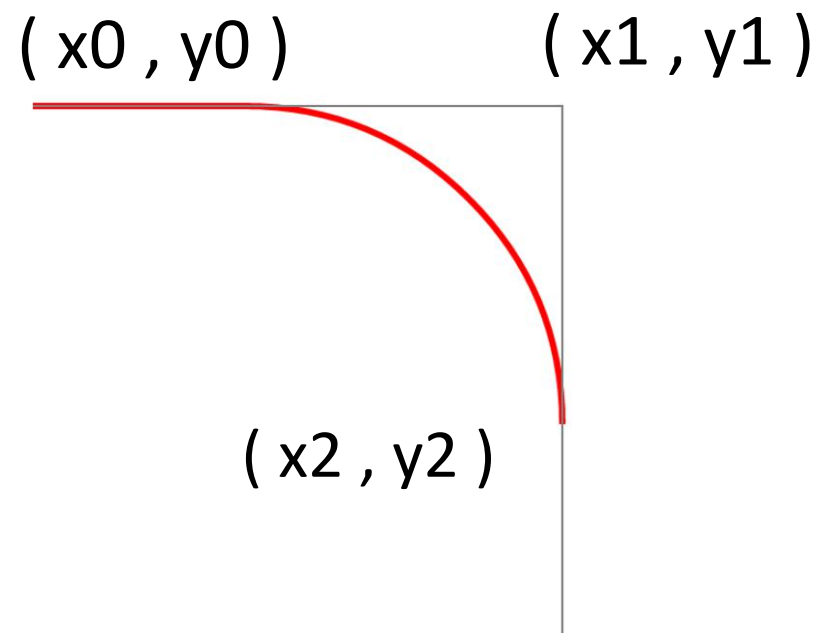
Canvas 图形及路径

Canvas曲线相关绘制（绘制弧）

- context.arc(centerx, centery, radius, startingAngle, endingAngle, anticlockwise = false);



```
// the thick line corresponds to:  
context.moveTo(x0, y0)  
context.arc(x, y, radius,  $\alpha$ ,  $\beta$ )  
context.stroke()
```



```
context.arcTo( x1 , y1 , x2 , y2 , radius );
```

Canvas 图形及路径

Canvas曲线相关绘制（绘制二次曲线）

```
context.moveTo( x0 , y0 );  
context.quadraticCurveTo(  
    x1, y1,    //控制点  
    x2, y2 ); //结束点
```

参考：<http://tinyurl.com/html5quadratic>

Canvas 图形及路径

Canvas曲线相关绘制（绘制贝塞尔曲线）

```
context.moveTo( x0 , y0 );  
context.bezierCurveTo(  
    x1, y1,           //控制点  
    x2, y2,           //控制点  
    x3, y3 );         //结束点
```

参考：<http://tinyurl.com/html5bezier>

内容纲要

- Canvas路径绘制
- Canvas基本形状
- Canvas描边及填充



Canvas 绘制基本形状

- Canvas绘制矩形相关

```
cxt.moveTo( x , y );  
cxt.lineTo( x + width , y );  
cxt.lineTo( x + width , y + height );  
cxt.lineTo( x , y + height );
```



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

- 手动绘制的区别在于能够控制绘图的方向（对填充的影响）
- 其他相关方法（fillRect（）、strokeRect（）、clearRect（））

Canvas 绘制基本形状

- Canvas绘制椭圆形相关

- <https://developer.mozilla.org/zh-CN/docs/Web/API/CanvasRenderingContext2D/ellipse>

```
var canvas = document.getElementById( 'canvas' );
```

```
var ctx = canvase.getContext( '2d' );
```

```
ctx.beginPath();
```

```
//x, y, radiusX, radiusY, rotation, startAngle, endAngle, anticlockwise(默认为false)
```

```
ctx.ellipse(100, 100, 50, 75, 45 * Math.PI/180, 0, 2 * Math.PI); //倾斜45°角
```

```
ctx.stroke();
```

- 思考如何扩充context，如增加context.triangle()方法来绘制三角形，如何开发自己的canvas库？

内容纲要

- Canvas路径绘制
- Canvas基本形状
- Canvas描边及填充



Canvas图案描边及填充



- 描边及填充的类型

- 纯色（不同设置方式）
- 渐变色（线性渐变、径向渐变）
- 图案样式（图片、视频、Canvas元素节点）

- 纯色描边及填充颜色设置（String类型）

- #ffffff、#642
- rgb(255 , 128 , 0)、rgba(100 , 100 , 100 , 0.8)
- hsl(20 , 62% , 28%)、 H: Hue 色相 S : Saturation 饱和度 L Lightness 明度
- red、yellow、blue
- 案例：context.strokeStyle= "red" ; context.fillStyle = "rgb(234,128,0)" ;

Canvas图案描边及填充（渐变色 - 线性渐变）

- 线性渐变色描边及填充案例

```
var grd = context.createLinearGradient(  
    xstart , ystart, xend , yend );  
grd.addColorStop( stop1 , color1 );  
grd.addColorStop( stop2 , color2);  
context.fillStyle = grd;
```

<https://developer.mozilla.org/zh-CN/docs/Web/API/CanvasRenderingContext2D/createLinearGradient>



Canvas图案描边及填充（渐变色 - 径向渐变）

- 径向渐变色描边及填充案例

```
var grd = context.createRadialGradient(  
    x0 , y0 , r0 , x1 , y1 , r1 );  
grd.addColorStop( stop1 , color1 );  
grd.addColorStop( stop2 , color2);  
context.fillStyle = grd;
```

<https://developer.mozilla.org/zh-CN/docs/Web/API/CanvasRenderingContext2D/createRadialGradient>



Canvas图案描边及填充（图案-图片、视频、画布）

- 图案描边及填充

- `var pattern = context.createPattern(image | video , repeat-style)`
- `var pattern = context.createPattern(canvas, repeat-style)`

- 图案描边及填充重复样式（repeat-style）

- `no-repeat`（不重复） `repeat`（重复）
- `repeat-x`（x轴重复） `repeat-y`（y轴重复）

- 设置填充样式（图案）

- `context.fillStyle = pattern ;`
- `context.strokeStyle = pattern ;`

Canvas图案描边及填充（图案-图片、视频、画布）

- 图案描边及填充（使用图片填充案例）

```
var canvas = document.getElementById("canvas");
```

```
var context= canvas.getContext("2d");
```

```
var img = new Image(); img.src = './images/fill_20x20.gif';
```

```
img.onload = function() {
```

```
    var pattern = context.createPattern(img, 'repeat' );
```

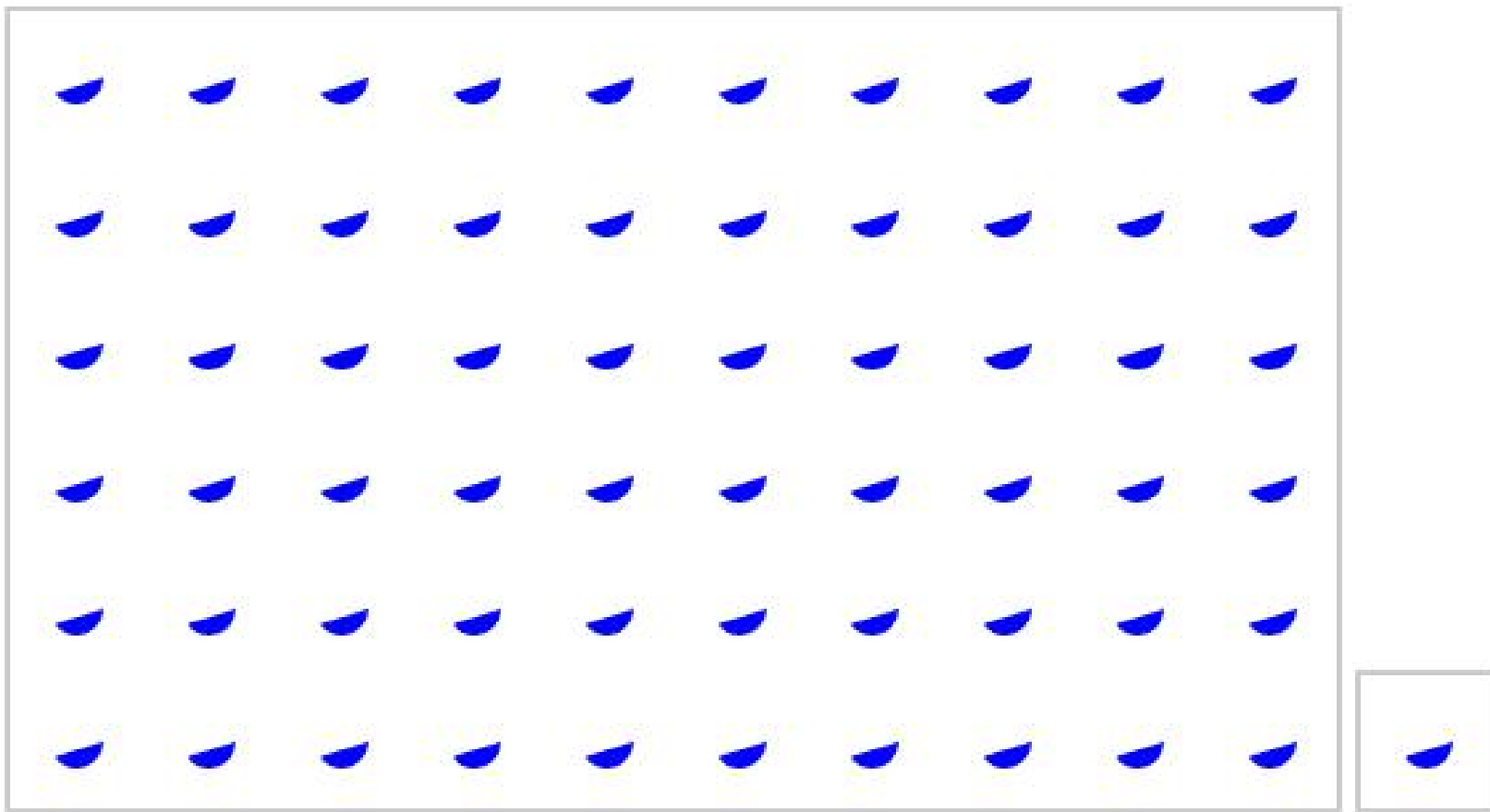
```
    context.fillStyle = pattern;
```

```
    context.fillRect(0,0,400,400);
```

```
};
```

Canvas图案描边及填充（图案-Canvas）

- 图案描边及填充（使用Canvas填充案例）



H5 动画与游戏开发

---H5 Canvas绘图补充一

内容纲要

- **Canvas阴影设置**
- **Canvas图像合成**
- **Canvas坐标系变换**

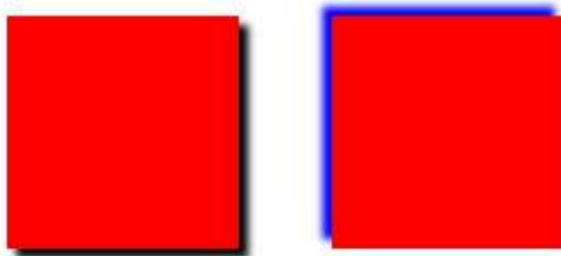


Canvas阴影设置

- 阴影设置参数

- context.shadowColor
- context.shadowOffsetX
- context.shadowOffsetY
- context.shadowBlur

- 阴影设置案例



内容纲要

- Canvas阴影设置
- Canvas图像合成
- Canvas坐标系变换

Canvas图像合成

- 全局透明度

- `context.globalAlpha = 1; //(default)`

- 全局图像混合设置

- `context.globeCompositeOperation = "source-over" ; //(default)`

- 图像混合类型

- source-over、source-atop、source-in、source-out
 - destination-over、destination-atop、destination-in、destination-out
 - lighter、copy、xor

内容纲要

- Canvas阴影设置
- Canvas图像合成
- Canvas坐标系变换



Canvas坐标系变换

- 平移 (translate)
 - context.translate(x , y) ;
- 旋转 (rotate)
 - context.rotate (deg) ;
- 缩放 (scale)
 - context.scale (sx,sy) ;
- 坐标系变换与状态存储

Canvas坐标系变换

- 矩阵变换 (transform)

- a 水平缩放 (1)
- b 水平倾斜 (0)
- c 垂直倾斜 (0)
- d 垂直缩放 (1)
- e 水平位移 (0)
- f 垂直位移 (0)

$$\begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

- 矩阵变换方法

- transform(a , b , c , d , e , f) 是对当前坐标系进行变换 ,
- setTransform(a , b , c , d , e , f) 是对默认坐标系进行变换 , 忽略之前的级联变换

H5 动画与游戏开发

---H5 Canvas绘图补充二

内容纲要

- **Canvas 文本**
- **Canvas 裁切**
- **Canvas 交互**

Canvas 文本

- 文字渲染基础

- context.font = "bold 40px Arial" ;
- context.fillText(string , x , y , [maxlen]);
- context.strokeText(string , x , y , [maxlen]);

- 字体 (font) 设置综述

- font-style (normal 默认、italic 斜体、oblique 倾斜)
- font-variant (normal、small-caps)
- font-weight (lighter、normal 默认、bold、bolder)
- font-size (20px、2em、150%)
- font-family (多种字体备选以逗号分割、Web安全字体)

Canvas 文本

- 文本水平对齐

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

- 文本垂直对齐

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

- `context.textBaseline = alphabetic (default) | ideographic (汉字等方块字体)`

- 文本度量

- `context.measureText(string).width`

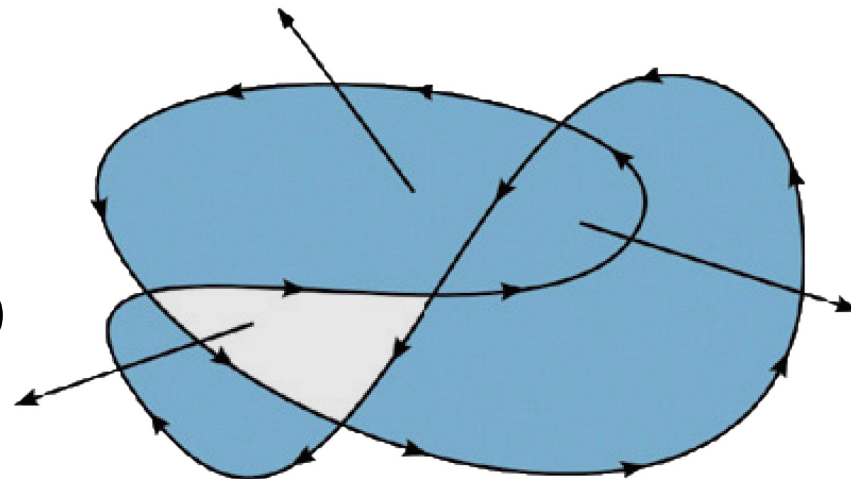
内容纲要

- Canvas 文本
- Canvas 裁切
- Canvas 交互



Canvas 剪切

- Canvas剪切方法与剪切路径
 - 绘制路径后使用`context.clip()`剪切
 - 剪切后，新绘制的图形将限定在此剪切区域
- 剪切与状态的关系
 - 有时可能需要取消或者新定义裁切区
 - 构建剪切区前可先保存状态（`save`）
 - 存储后，再绘制的图形将限定在裁剪区域内
 - 完成剪切区内的绘制后可进行状态恢复（`restore`）
- 非零环绕原则



参见实例：[LS02_16.html](#)

内容纲要

- Canvas 文本
- Canvas 裁切
- Canvas 交互



Canvas 交互

- Canvas 交互检查方法

- context.isPointInPath(x , y) ;
- context.isPointInPath(path, x, y);
- context.isPointInStroke(x , y) ;
- context.isPointInStroke(path, x, y);

- //鼠标点击的位置，注意坐标补偿
- var x = e.clientX - theCanvas.getBoundingClientRect().left;
- var y = e.clientY - theCanvas.getBoundingClientRect().top;

The background of the slide is decorated with numerous overlapping circles in various shades of green and yellow, scattered across the top and right sides.

Thank You !