**Canvas**

[1、 Canvas的大小和绘图表面大小 2](#_Toc28571)

[1． 通过style和css修改区别 2](#_Toc10247)

[2、 Canvas的元素属性 2](#_Toc6028)

[Width(非负整数) 2](#_Toc5201)

[Height(非负整数) 2](#_Toc15095)

[3、 Canvas的元素方法 2](#_Toc13137)

[GetContext() 2](#_Toc31698)

[toDataURL(type,quality) 3](#_Toc27219)

[toBlob(callback,type,args...) 3](#_Toc28858)

[4、 Canvas的CanvasRenderingContext2D对象所含的属性 3](#_Toc10739)

# Canvas的大小和绘图表面大小

Canvas元素实际上有两套尺寸。一个是元素本身的大小，还有一个是元素绘图表面的大小

默认情况下，浏览器所创建的canva元素的大小300\*150，可以通过指定width和height属性值而修改canvas元素的大小。

## 通过style和css修改区别

设置元素的width与height属性时，实际上是同时修改了该元素本身的大小和元素绘图表面的大小。如果通过css来设定canvas元素的大小，那么只会改变元素本身的大小，而不会影响到绘图表面！

# Canvas的元素属性

## Width(非负整数)

设置元素width与height，如果是在css 中写，那么就会将绘图表面进行缩放，使之符合元素尺寸，不能给数值加px

## Height(非负整数)

置元素的width与he

# Canvas的元素方法

## GetContext()

返回与该canvas元素相关的绘图环境对镜对象。每个canvas元素都有这样一个环境对象，而且每个环境对象均与一个canvas元素相关联

## toDataURL(type,quality)

返回一个数据地址(data URL)

## toBlob(callback,type,args...)

# Canvas的CanvasRenderingContext2D对象所含的属性

# context.beginPath

开始一条路径，或重置当前的路径(使用这些方法来创建路径：moveTo()、lineTo()、quadricCurveTo()、bezierCurveTo()、arcTo() 以及 arc())

# Context.arc()

context.arc(x,y,r,sAngle,eAngle,counterclockwise);

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。规定应该逆时针还是顺时针绘图。False = 顺时针，true = 逆时针。

# Context.stroke()

stroke() 方法会实际地绘制出通过 moveTo() 和 lineTo() 方法定义的路径。默认颜色是黑色。

# Context.measureText

measureText() 方法返回包含一个对象，该对象包含以像素计的指定字体宽度。*context*.measureText(*text*).width;

# clearRect

clearRect() 方法清空给定矩形内的指定像素。

context.clearRect(x,y,width,heigh

# context.fillText

fillText() 方法在画布上绘制填色的文本。文本的默认颜色是黑色。

context.fillText(*text*,*x*,*y*,*maxWidth*);

# context.fill

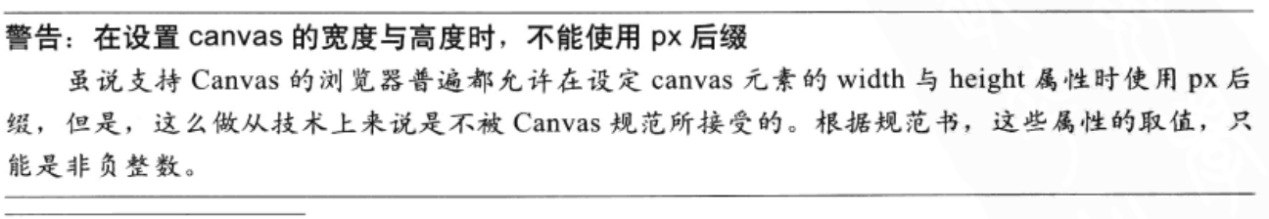
fill() 方法填充当前的图像（路径）。默认颜色是黑色。

# context.drawImage

在画布上定位图像：*context*.drawImage(*img*,*x*,*y*);

在画布上定位图像，并规定图像的宽度和高度：*context*.drawImage(*img*,*x*,*y*,*width*,*height*);

剪切图像，并在画布上定位被剪切的部分：*context*.drawImage(*img*,*sx*,*sy*,*swidth*,*sheight*,*x*,*y*,*width*,*height*);



返回一个数据地址(data URL)

16

