

第11章 图形图像处理

- + PHP不仅限于处理文本数据，PHP还可以处理多种格式的图像。虽然在PHP中有一些简答的图形图像处理函数是可以直接使用的。但是大多数要处理的图像都要通过GD库来处理，它不仅可以创建新图像，而且可以处理已有的图像。本章我们的重点就是学习GD库。

11.1 加载GD库

- + 在PHP 5之后的版本中默认就安装了GD库，默认情况下GD库是没有被加载的。我们需要通过配置php.ini来加载GD库，如图所示。
- + 在将图中的语句改为图中所示的形式后重启服务器即可成功加载。当然我们使用的集成环境默认就加载了GD库。当然读者最好打开php.ini文件确认一下。在修改配置后我们可以通过两种方法来验证。

```
956 ;extension=php_fileinfo.dll  
957 extension=php_gd2.dll
```

默认是被“;”注释的，
删除注释后GD库即加载

11.1 加载GD库

- + (1)通过输出PHP信息确认GD库已成功加载。
- + (2)通过gd_info()取得GD库信息。

11.2 创建图像

+ 在PHP中GD库处理图像的操作都是先在内存中处理，操作完成后再以数据流的方式输出到浏览器或者保存在服务器磁盘中。

创建一个图像通常要经过四个步骤：

+ (1) 创建画布

+ (2) 绘制图像

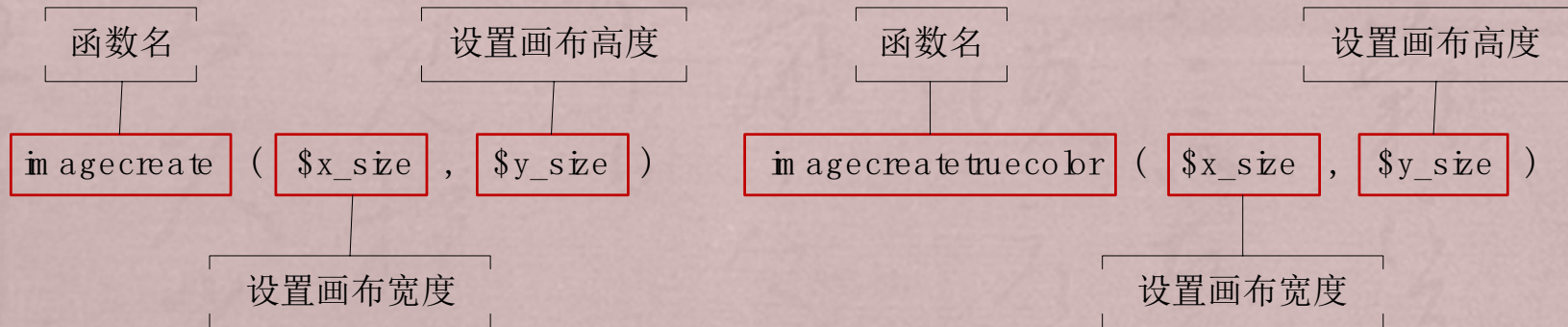
+ (3) 输出图像

+ (4) 释放资源

+ 下面我们就来分别学习这些步骤。

11.2.1 创建画布

- + 在使用GD库处理图像时，首先要创建一张画布。创建画布就是在内存中开辟一块存储区域，以后GD库的所有操作都是基于这个画布处理的。
- + 我们通常使用`imagecreate()`和`imagecreatetruecolor()`来创建指定的画布，它们的语法如图所示。
- + 以上两个函数都可以创建一张画布，成功都会返回一个资源句柄，失败则返回`FALSE`。不同的是它们可以容纳的色彩范围不同，`imagecreate()`创建一个基于普通调色板的图像，通常支持256色。`imagecreatetruecolor()`可以创建一个真色彩图像，但是该函数不可以用于GIF格式图像。



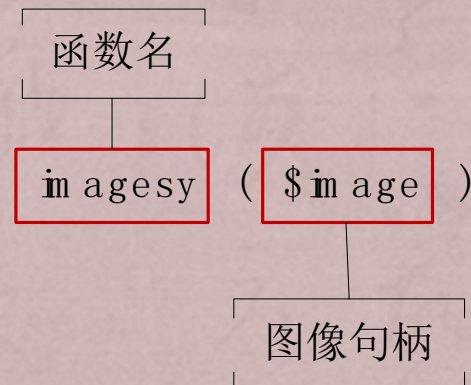
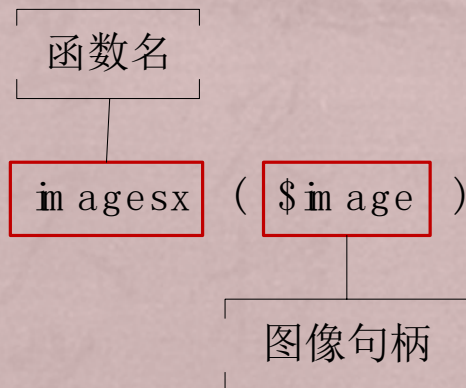
11.2.1 创建画布

- + (1)演示使用创建画布函数创建画布。
- + 由于我们没有在画布上执行任何操作，因此浏览器不会显示出画布。除了使用上面两个函数可以创建一个画布外。我们还可以通过表中的函数打开服务器或者网络文件中已经存在的图像。
- + 表中的函数都只接受一个文件路径或者URL，在执行成功后返回文件句柄，失败则返回FALSE。

函数名	描述
imagecreatefromgif()	通过GIF文件或者URL新建一个图像
imagecreatefromjpeg()	通过JPEG文件或者URL新建一个图像
imagecreatefrompng()	通过PNG文件或者URL新建一个图像
imagecreatefromwbmp()	通过WBMP文件或者URL新建一个图像

11.2.1 创建画布

- + (1) 演示通过常用新建图像函数新建图像。
- + 以上代码就通过两种方式创建了图像。但是由于我们并没有输出图像，因此浏览器不会有任何显示。但是我们可以通过 `imagesx()` 和 `imagesy()` 来获取创建图像的尺寸（以像素为单位），它们的语法如图所示。
- + (2) 通过 `imagesx()` 和 `imagesy()` 获得新建图像的宽和高。



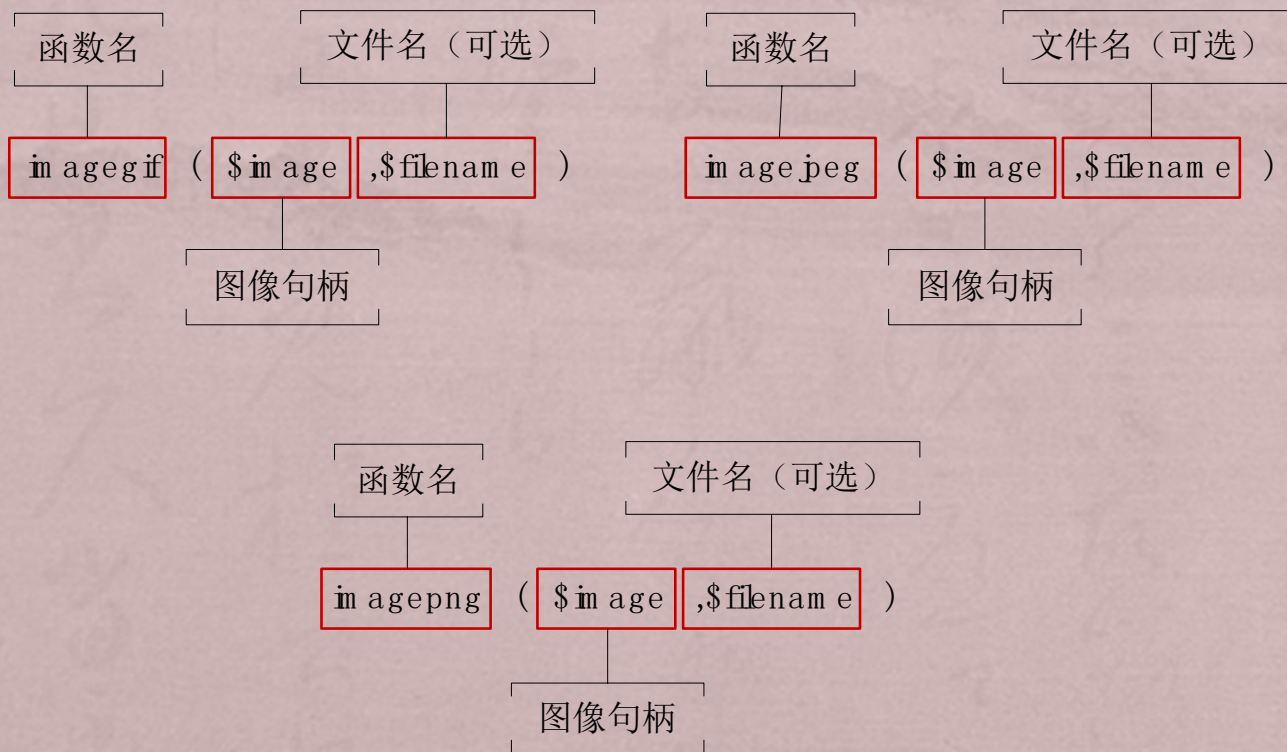
11.2.2 输出图像

+ 在进过上面这些函数的学习，相比读者已经很像看看这些图像到底是什么样子的。在PHP中可以使用不同的函数输出不同格式的图像，如表是常用的输出图像函数。

函数名	描述
imagegif()	输出一个GIF格式图像到浏览器或文件
imagejpeg()	输出一个JPEG格式图像到浏览器或文件
imagepng()	输出一个PNG格式图像到浏览器或文件

11.2.2 输出图像

- + 表中函数的语法如图所示。
- + 在所示的语法中，如果指定了第二个参数则图像会以文件的形式输出。

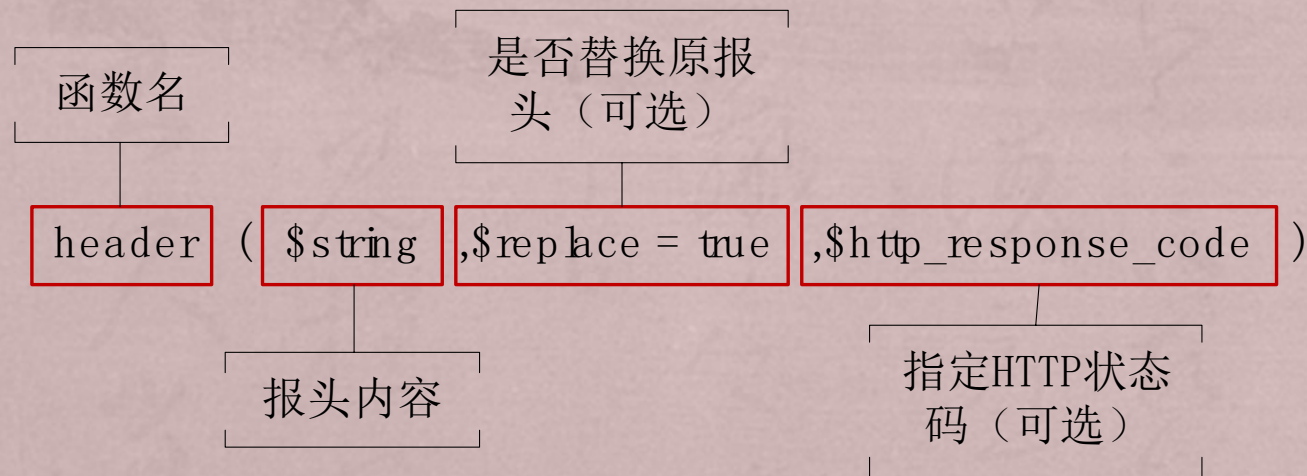


11.2.2 输出图像

- + (1) 使用图像输出函数输出创建的图像。
- + (2) 演示代码会出现的错误。
- + (3) 演示使用 `ob_clean()` 清除输出缓冲后正确输出图像。
- + (4) 代码运行在IE内核之外浏览器的情况。

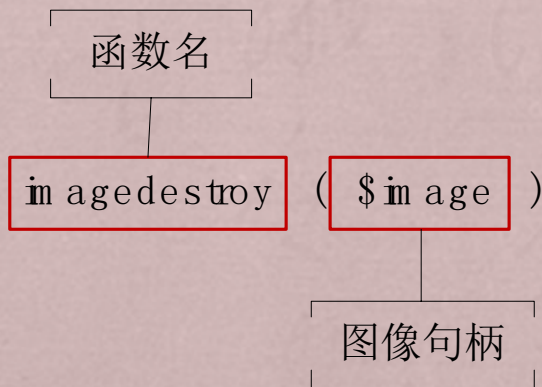
11.2.2 输出图像

- + 为了保持兼容，我们需要在程序中使用header()来指浏览器以什么形式输出类型，它的语法如图所示。
- + (1) 演示使用header()函数指定浏览器输出类型，从而正确输出图像。



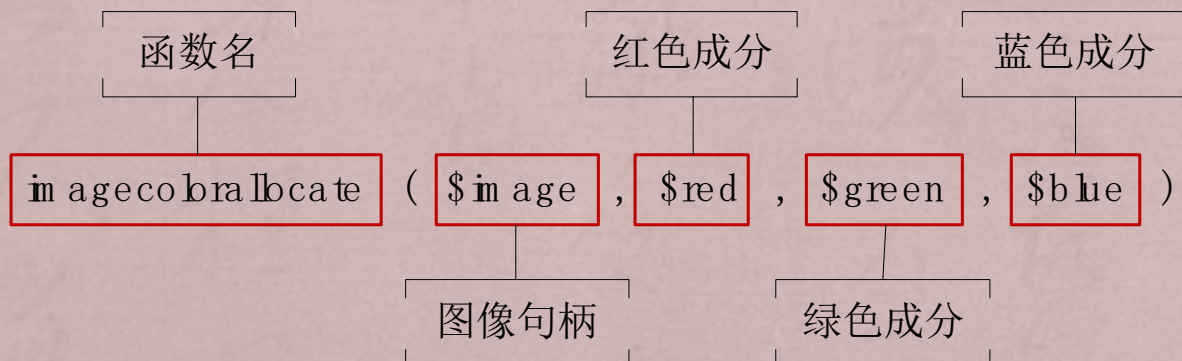
11.2.3 释放资源

- + 在图像的所有资源使用完毕后，我们通常就需要释放图像处理所占用的内存了。在PHP中通过`imagedestroy()`来释放资源，它的语法如图所示。
- + (1) 演示图像处理程序执行完毕后使用`imagedestroy()`释放内存资源。
- + (2) 证明释放资源是有意义的。



11.2.4 设置颜色

- + 前面我们已经学习了如何建立一张画布，但是要是现在就开始绘画，是不会成功的，因为我们现在还没有“彩笔”。这样就是开始绘画结果就像是拿一支没有笔芯的笔在画布上画一样，都是徒劳的。下面我们就来创建我们的“彩笔”——设置颜色。
- + 在PHP中通过imagecolorallocate()来设置颜色，它的语法如图所示。
- + imagecolorallocate()会返回一个标识符，代表了由给定的RGB成分组成的颜色。图中所示语法中\$red、\$green、\$blue的取值可以是0到255的整数或者十六进制的0x00到0xFF。



11.2.4 设置颜色

- + (1) 演示使用 `imagecolorallocate()` 设置颜色。
- + (2) 演示为背景设置不同的颜色。

11.2.5 绘制图像

- + 在前面小节的学习中我们学习了通过使用相关函数来打开服务器存储的图像或者通过URL打开互联网上的图像。在本节，我们要学习的是自己通过使用一些函数来绘制图像。这开始之前我们需要明白的一个知识点就是PHP的坐标系统，

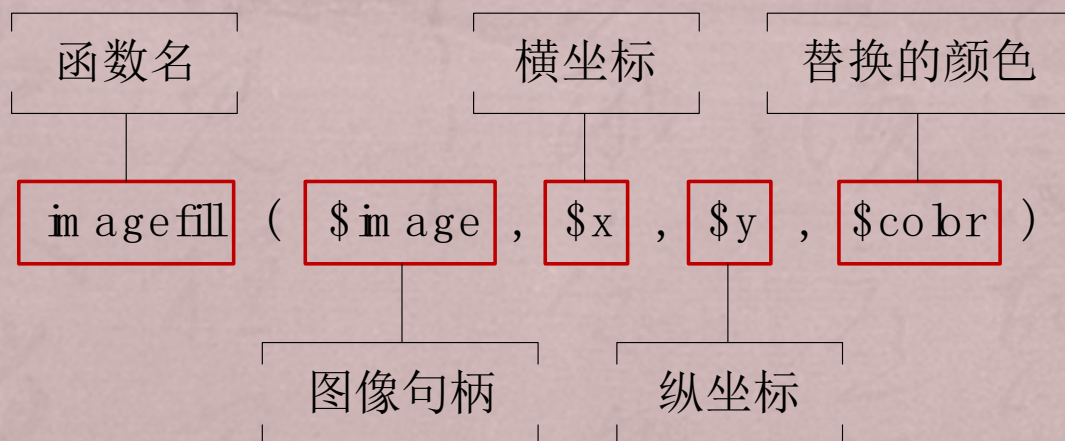
11.2.5 绘制图像

- + 如图所示。
- + 通过坐标系，我们就可以比较准确地定位到一个点，然后通过相关函数来绘制图形。下面我们来学习PHP中绘制各种图形的函数。



1. 区域填充

- + 区域填充不可以用来绘制图像。它可以将一个已存在图像中大面积的颜色很方便地替换为别的颜色。在PHP中通过imagefill()来执行区域填充，它的语法如图所示。
- + imagefill()会将与 (x, y) 点处颜色相同并且相邻的颜色替换为\$color设置的颜色。

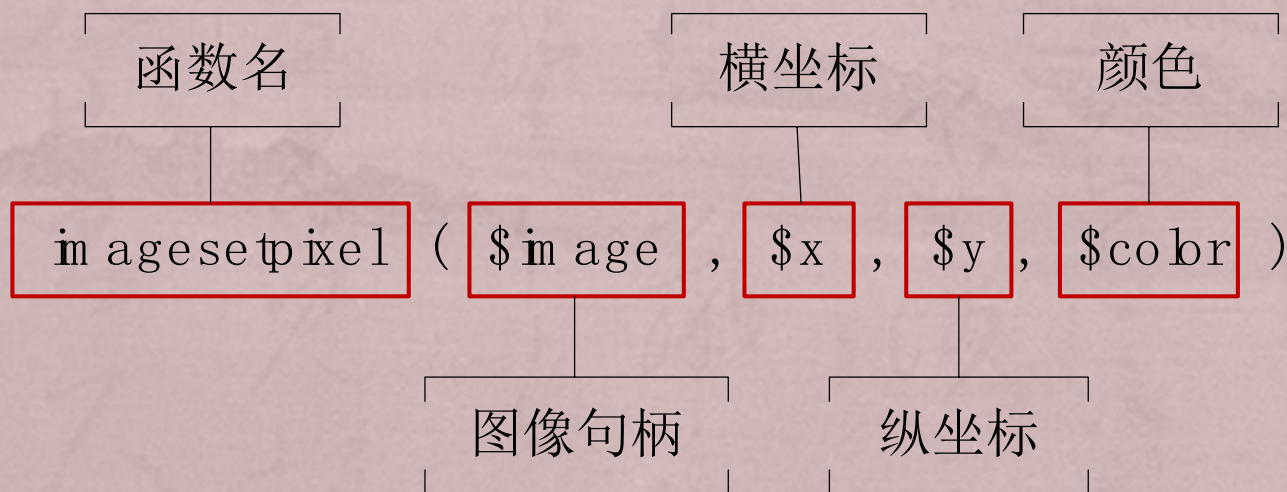


1. 区域填充

- + (1) 演示使用区域填充将一张图片连续的白色部分替换为蓝色。
- + (2) 演示使用随机数作为参数设置颜色并使用 `imagefill()` 填充图像。

2. 绘制点和线

+ 在PHP中，我们可以通过 `imagejpeg()` 来绘制一个像素点，它的语法如图所示。

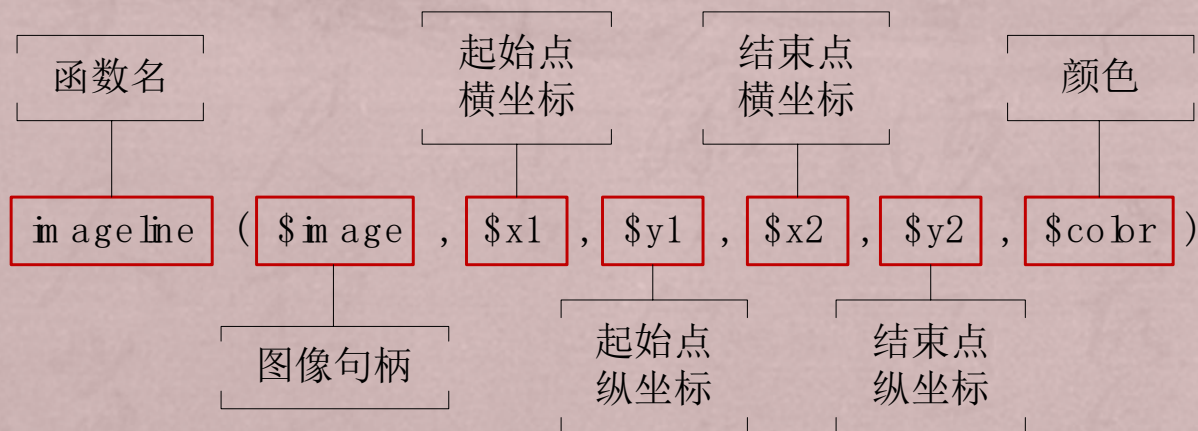


2.绘制点和线

- + (1) 演示在一张画布上绘制像素点。
- + (2) 演示通过循环和随机数配合在画布绘制多个像素点。

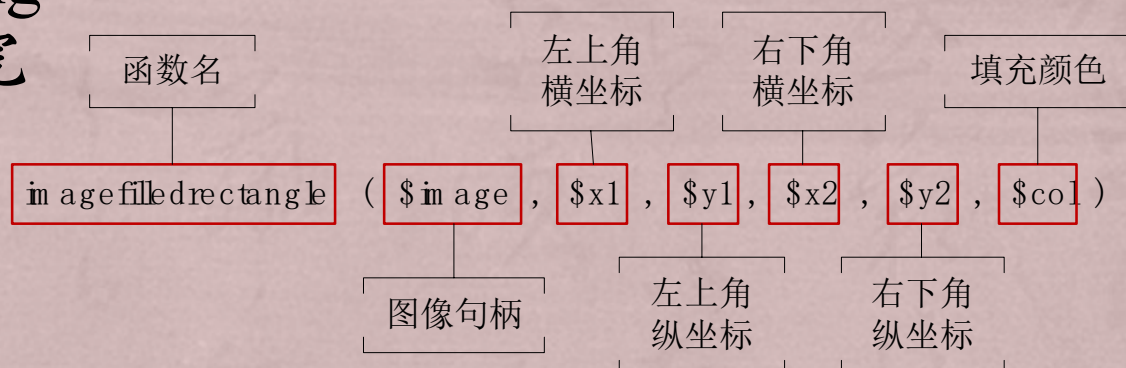
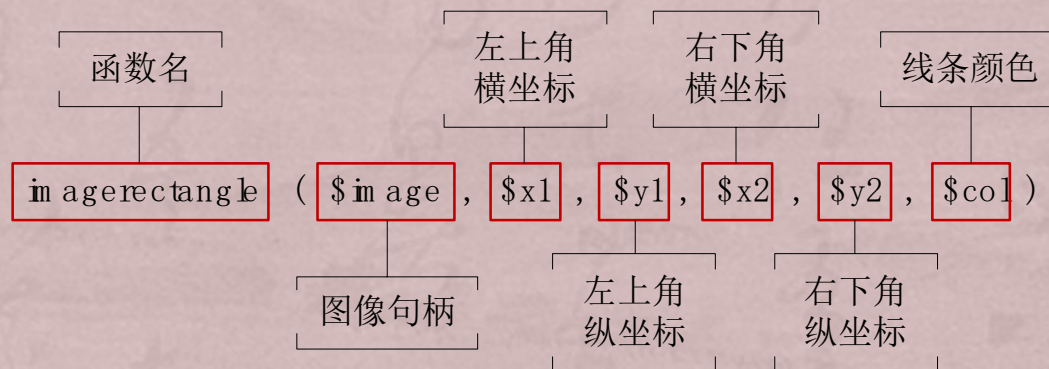
2. 绘制点和线

- + 在PHP中我们使用`imageline()`来绘制一条直线，它的语法如图所示。
- + `imageline()`函数看起来参数比较多，其实也是很容易理解的。因为线就是连接两个点而组成的，所以坐标就应该有两对也就是四个了。
- + (1) 演示在画布上绘制两条白色对角线。
- + (2) 演示使用循环联合随机数在一张画布内绘制线条。



3. 绘制矩形

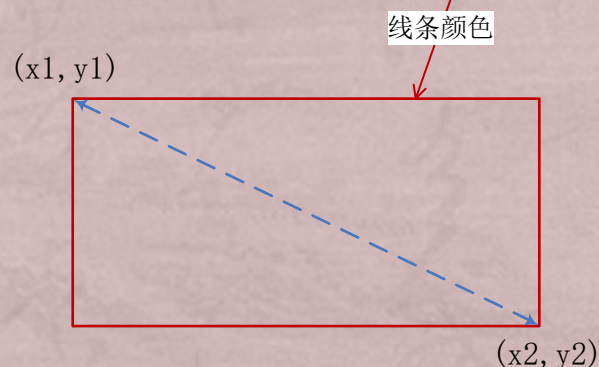
+ 在PHP中我们使用 `imagerectangle()` 来绘制一个矩形。和 `imagerectangle()` 类似的函数是 `imagefilledrectangle()` 它会在绘制完成矩形后并填充矩形，它们的语法如图所示。



3. 绘制矩形

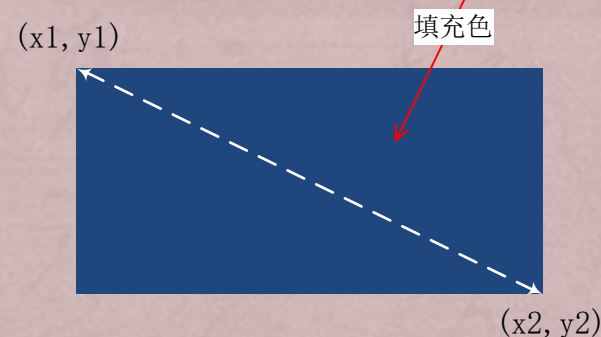
+ 为了使表达更加清楚，我们再以图的形式表示出矩形与函数的对应关系如图所示。

```
imagerectangle ( $image , $x1 , $y1 , $x2 , $y2 , $col )
```



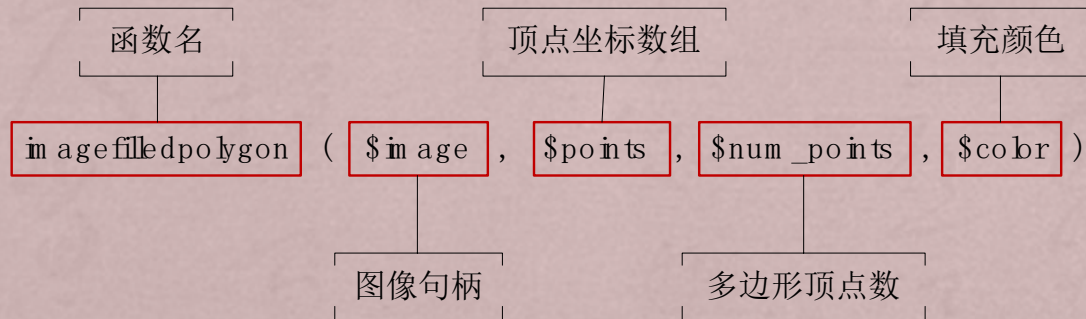
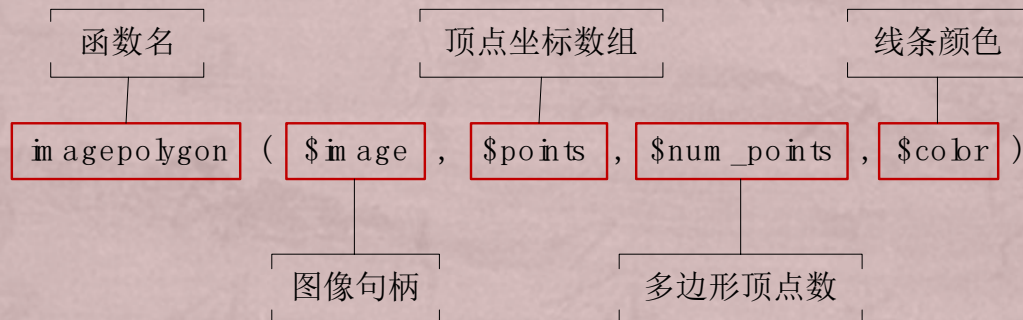
+ (1) 演示使用 imagerectangle() 和 imagefilledrectangle() 绘制矩形。

```
imagefilledrectangle ( $image , $x1 , $y1 , $x2 , $y2 , $col )
```



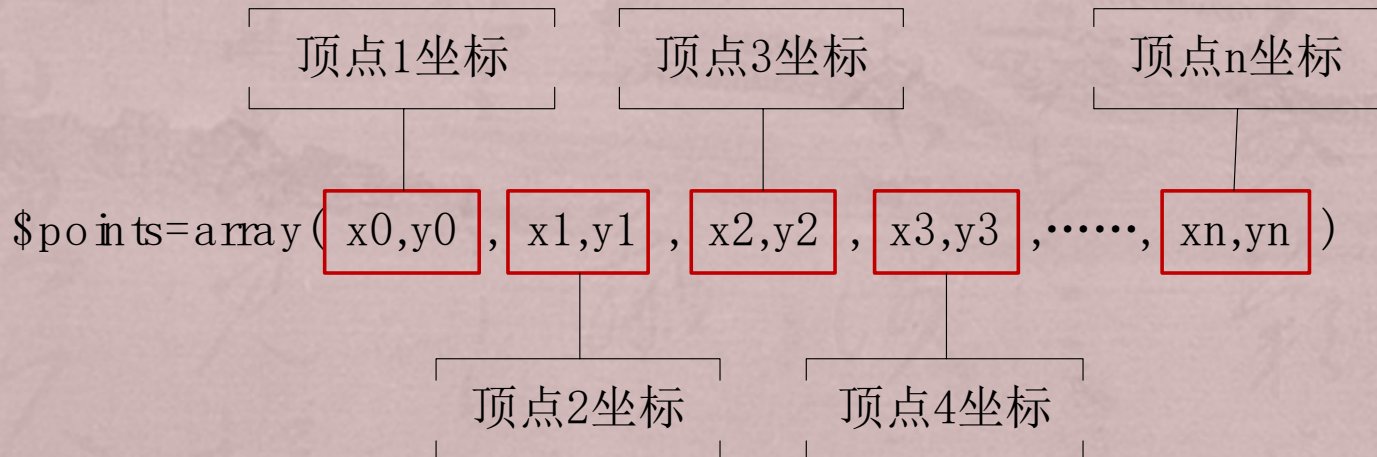
4. 绘制多边形

- + 前面我们以及学习了绘制矩形，但是通常这是不能满足我们对图像的要求。在PHP中我们可以使用`imagepolygon()`来绘制一个多边形，例如三角形，五边形，八边形等图形。也可以使用`imagefilledpolygon()`来绘制并填充多边形，它们的语法如图所示。



4.绘制多边形

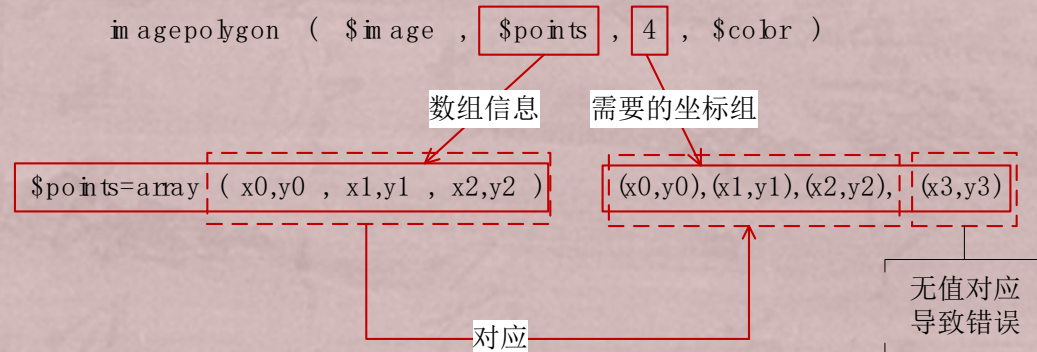
- + 在所示的语法中，\$points是保存多边形顶点坐标的数组，它的保存的详细信息如图所示。



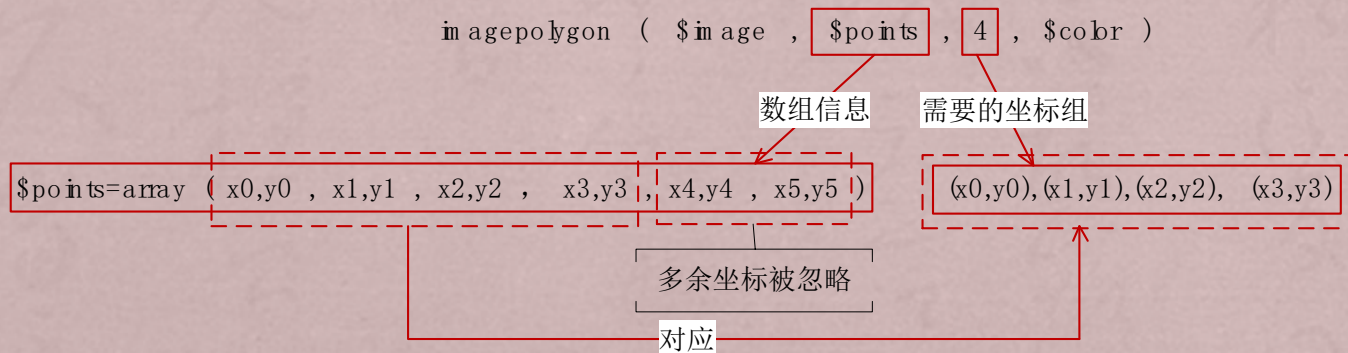
4. 绘制多边形

- + 这还要注意的一个知识点是数组的顶点坐标数不得小于多边形顶点数 \$num_points\$，我们来看图。
- + 图中形象地表示了顶点坐标组与顶点数的关系，那么我们就可以利用这个特点来做出一个随机绘制多边形的程序。
- + (1) 演示使用 `imagepolygon()` 和 `imagefilledpolygon()` 联合随机数绘制随机图像。

```
imagepolygon ( $image , $points , 4 , $color )
```

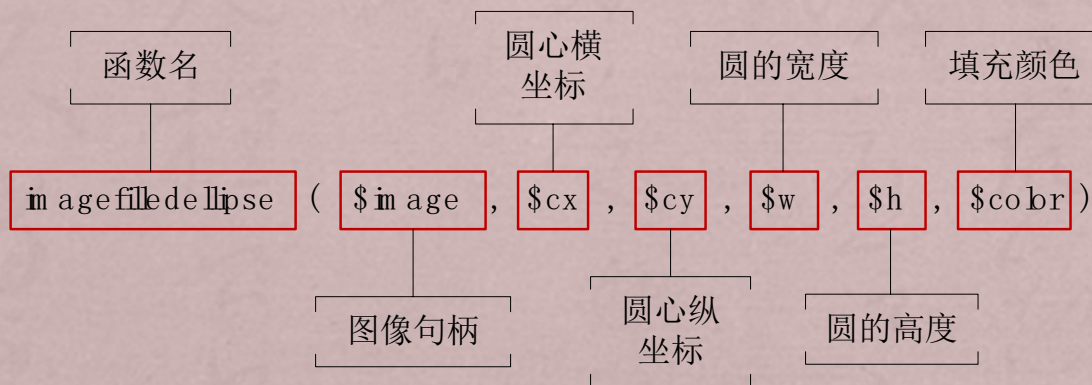
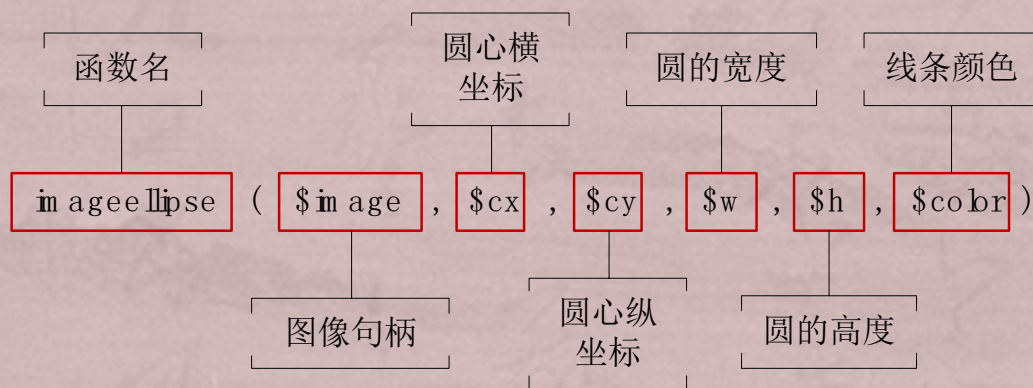


```
imagepolygon ( $image , $points , 4 , $color )
```



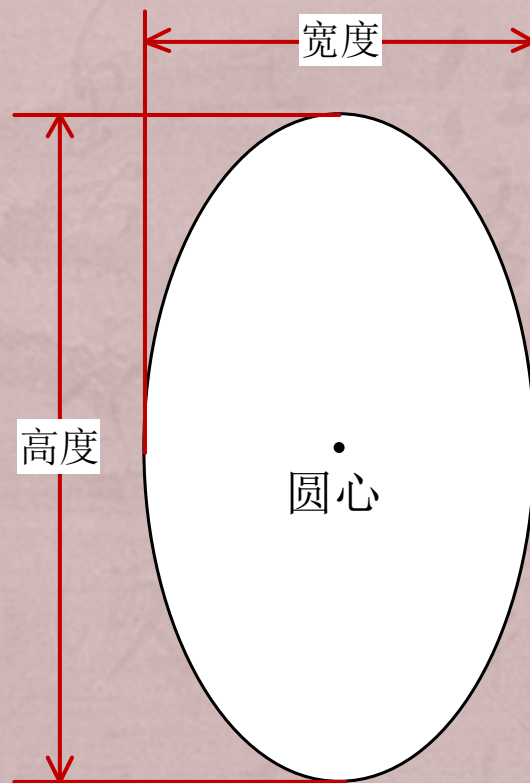
5. 绘制椭圆

- + 在PHP中可以使用imageellipse()来绘制一个椭圆，使用imagefilledellipse()绘制并填充椭圆，它们的语法如图所示。






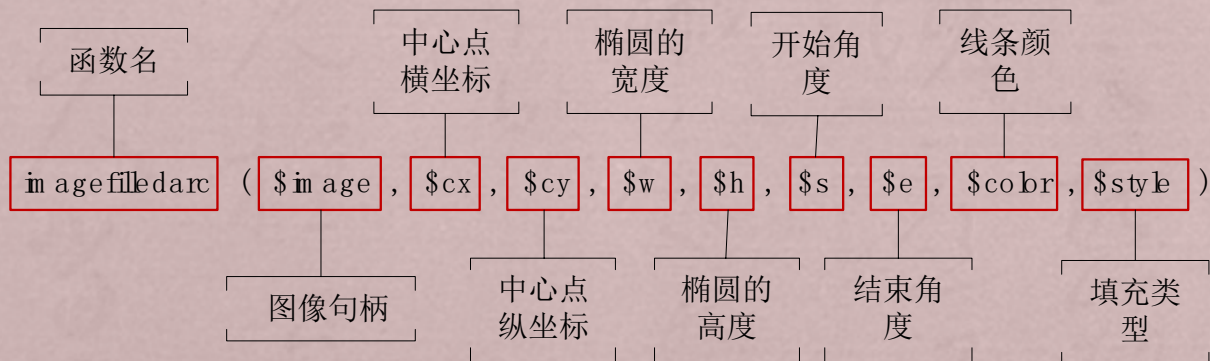
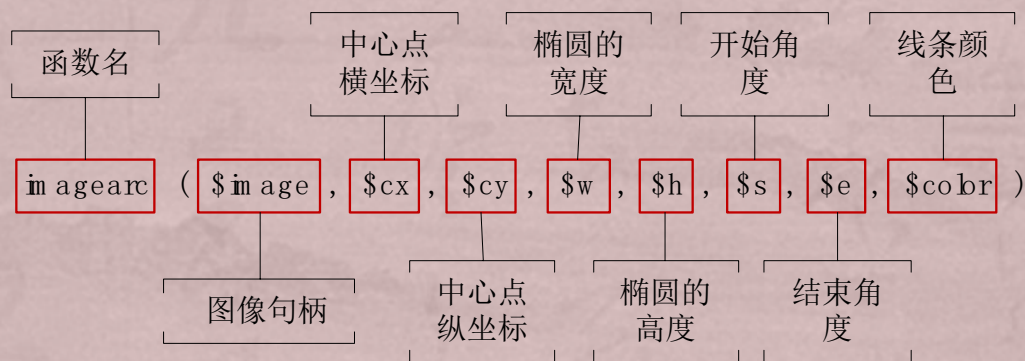
5. 绘制椭圆

- + 在学习这些函数的使用之前，我们首先来了解一下绘制出的圆形的坐标系，如图所示。
- + (1) 演示使用绘制椭圆函数的使用方法以及输出结果。



6. 绘制弧线

- + 在PHP中我们可以使用来画出一条弧线或者圆形，也可以使用来绘制弧线或者圆形并填充，它的语法如图所示。
- + 在语法中，\$style可以有如下值：
- + IMG_ARC_PIE：普通填充，产生圆形边界。
- + IMG_ARC_CHORD：只使用直线连接起始和结束点，与IMG_ARC_PIE方式互斥。
- + IMG_ARC_NOFILL：指明弧或弦只有轮廓，不填充。
- + IMG_ARC_EDGED：用直线将起始和结束点与中心点相连。

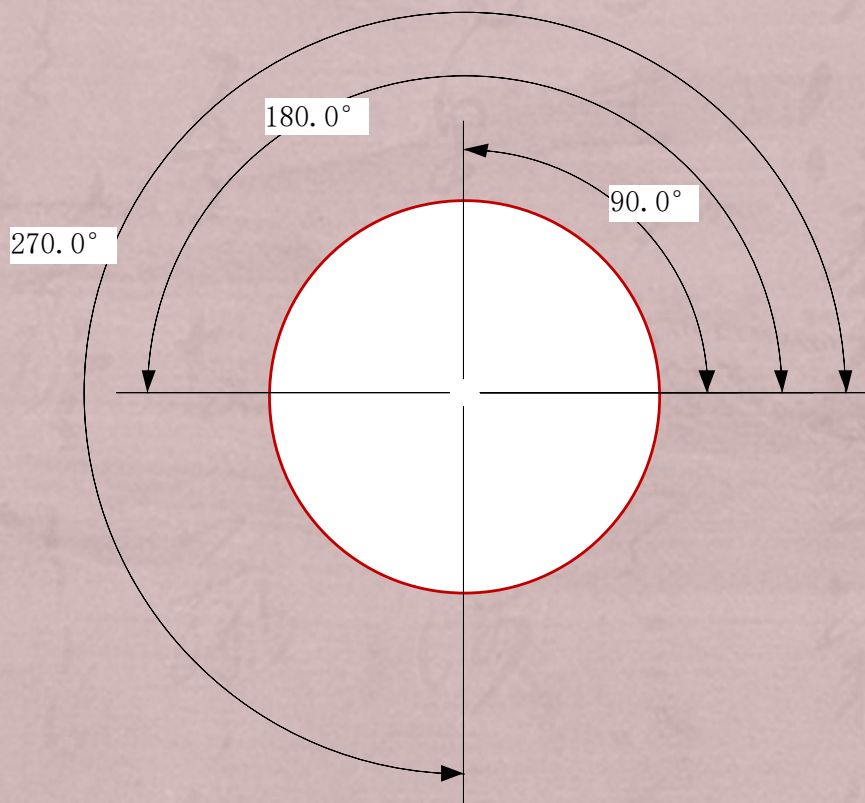


6. 绘制弧线

+ 在绘制弧线之前，还需要读者了解的就是绘制弧线的函数的角度系统，如图所示。

+ (1) 演示 `imagearc()` 和 `imagefilledarc()` 的使用方法以及输出的效果。

+ (2) 演示利用 `imagefilledarc()` 绘制有3D效果的饼状图。



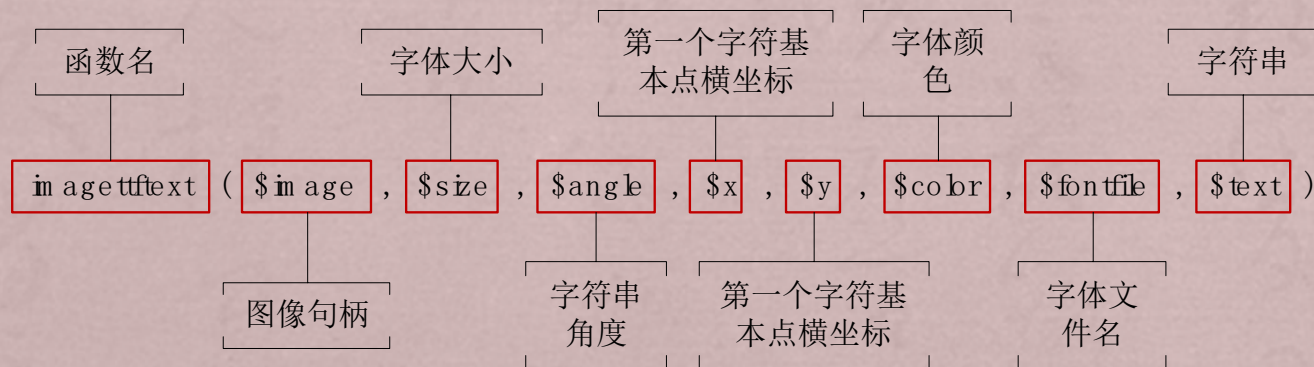
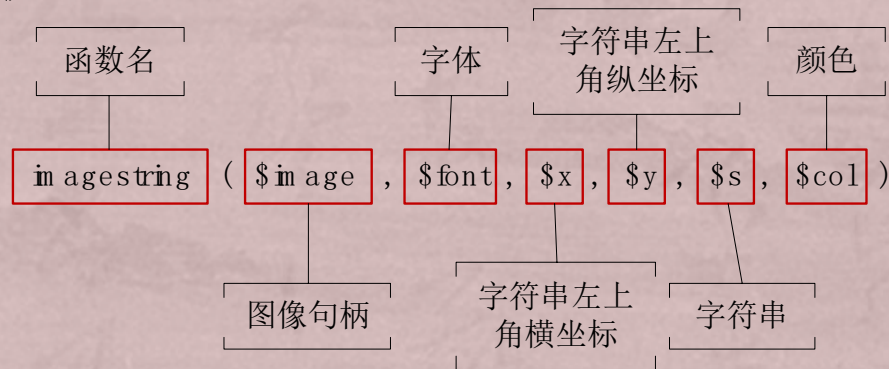
11.2.6 在图像上绘制文字

- + 在PHP中除了可以在画布上绘制图像以外，还可以将文字绘制在画布上。我们通常会使用如表中所示的函数将文字绘制在图像上。

函数名	描述
<code>imagestring()</code>	水平绘制一行字符串
<code>imagestringup()</code>	垂直绘制一行字符串
<code>imagechar()</code>	水平绘制一个字符
<code>imagecharup()</code>	垂直绘制一个字符
<code>imagettftext()</code>	用TrueType字体向图像写入文本

11.2.6 在图像上绘制文字

- + 以上函数的语法如图所示。
- + 由于imagestringup()、imagechar()和imagecharup()与imagestring()的参数都相同，因此就不费较大篇幅来讲解语法。在图11.21中的\$font可以为1、2、3、4和5，代表使用内置字体。如果想使用其他字体则可使用imagedloadfont()来载入新字体。imagettftext()返回一个保存了输出的文本四个角的坐标值的数组。

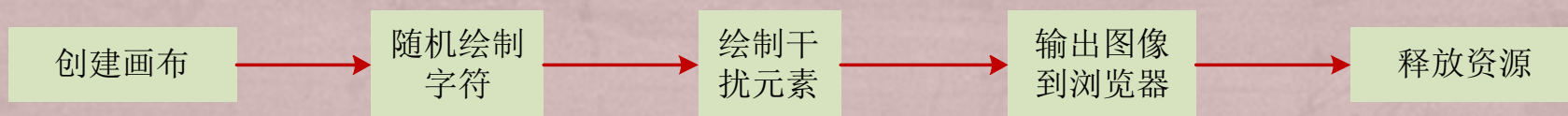


11.2.6 在图像上绘制文字

- + (1) 演示使用 `imagestring()`、`imagestringup()`、`imagechar()` 和 `imagecharup()` 在图像上绘制文字。
- + (2) 演示使用 `imagefttext()` 在图像上绘制字符。

11.3 通过GD库生成验证码

+ 我们已经将PHP常用的图形图像处理函数学习完毕了。我们经常在网上注册一些账号的时候遇到一些注册码的输入。在本小节中，我们就利用前面所学的知识综合起来实现生产验证码。我们先来看一下实现的步骤，如图所示。



11.3 通过GD库生成验证码

- + (1) 通过使用代码实现图11.22所示的步骤生成验证码。
- + 上面的运行结果我们只取了随机的三个验证码显示，这也表明了我们成功地实现了生成验证码，但是生成的方式远远不止这种方式，只要满足验证码条件即可。读者可以在理解以上示例的逻辑以后自行设计一段验证码输出程序。

11.3 小结

+ 本章中我们学习的是PHP的图形图像处理技术，在学习的过程中有不少的需要注意的地方。例如坐标系统和角度系统。如果这两个知识点不能透彻掌握，那么学习过程中一定是非常艰难的。并且本章中的函数参数比较多，稍有不对应就会出现错误，也可能为学习带来麻烦。因此还是需要多理解，在理解的基础上多实践就必定会轻松掌握这些知识。