# GD图像处理

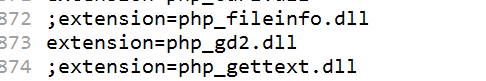
## GD图像处理基本技术

### GD库引入与介绍

API：外部提供的应用接口：已经准备好了一套处理某些功能的机制，用户只需要按照指定的数据要求，调用指定的函数或者方法（类）就可以实现某个功能。

1）GD库的概念：Graphic Device，图像处理扩展（外部提供的API），能够允许PHP在脚本中使用对应的函数来实现某些图像制作功能

2）GD库的引入：GD库是外部提供的API，已经被集成到PHP扩展库中（不需要下载），但是需要在PHP的配置文件中开启对应的扩展。GD扩展（GD2）。



注意：记得重启Apache服务器

## GD图像处理基本技术

画图的基本流程：画图本质是在内存开辟一块很大的内存区域用于图片制作

1、 准备画布

2、 开始作画

3、 保存内容

4、 销毁画布

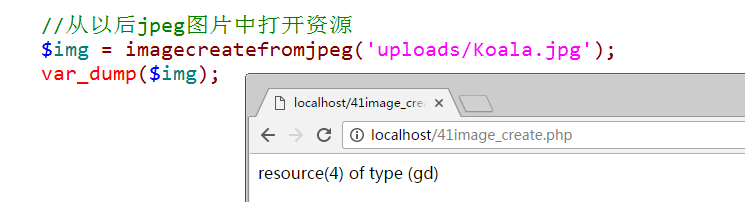
### 创建画布资源

1）ImageCreate(宽, 高)：创建一个空白画布（背景色是白色的）

2）ImageCreateTrueColor(宽, 高)：创建一个真彩画布（背景色是黑色的，需要填充）



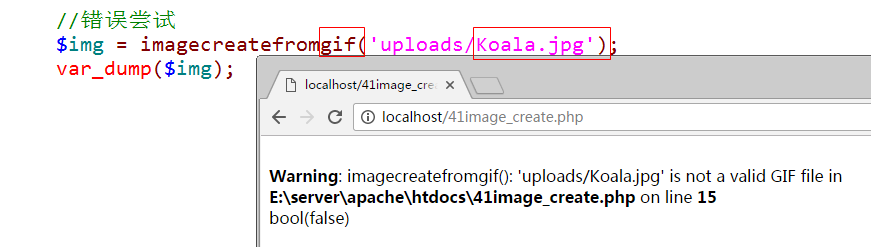
3）ImageCreatefromJpeg(图片文件路径)：打开一个jpeg格式的图片资源



4）ImageCreatefromGif(图片文件路径)：打开个gif格式图片资源（PHP中无法实现动态）

5）ImageCreatefromPng(图片文件路径)：打开png格式图片资源

如果从已知文件创建图片资源，那么一定要匹配对应的打开方式，否则会出错

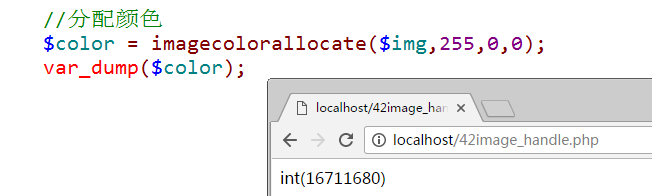


## GD图像处理基本技术

### 操作画布资源

说明：所有的画布资源操作都是需要指定画布资源，而且都是第一个参数。

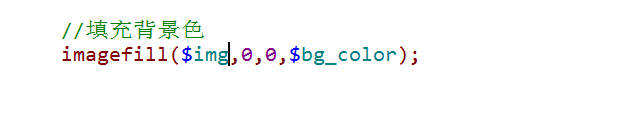
1）分配颜色：imageColorAllocate(画布资源,红色,绿色,蓝色)，根据RGB三色组给指定画布资源分配一组颜色，会返回一个颜色句柄（一组整数），每一个色组都是从0到255



在真彩图片资源中，所有分配的颜色都不会自动给图片资源上色，是用来后续操作图片资源的时候，指定着色的：但是如果当前使用的imagecreate创建的图片资源，那么第一个分配的颜色，会自动被着色为图片背景色。

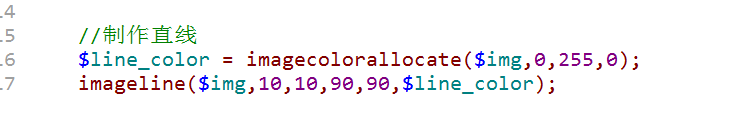
注意：凡是给图片上增加内容，基本都需要分配颜色（每一个操作图片的函数之前，都需要先调用分配颜色的函数得到一个颜色）

2）填充区域：imageFill(画布资源,起始X坐标,起始Y坐标,颜色句柄)：指定位置开始填充指定颜色

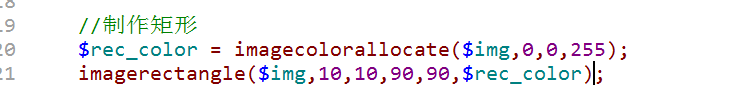


Imagefill的填充逻辑：从指定点开始，自动匹配相邻点，如果颜色一致，自动渲染，扩展到全图。

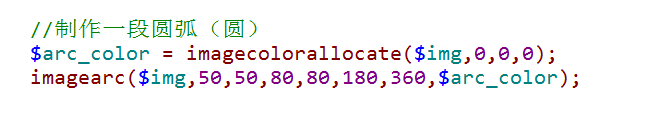
3）画直线：imageLine(图片资源,起始点X,起始点Y,终点X,终点Y,颜色)，制作一条直线



4）画矩形：imageRectangle(图片资源,左上角X,左上角Y,右下角X,右下角Y,颜色)：制作一个矩形

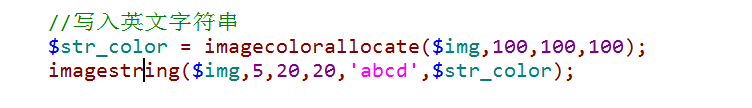


5）画圆弧：imageArc(图像资源,轴点X,轴点Y,宽度,高度,弧度起点,弧度终点,颜色)：制作弧度图像

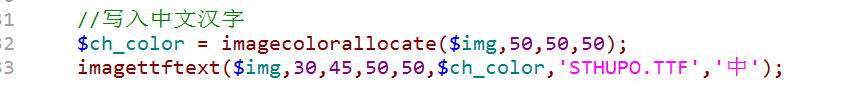


6）在画布上写字：imageString()，imageTtfText()

Imagestring(图片资源,文字大小,起始X,起始Y,内容,颜色)：用来书写ASCII对应的字符（英文）



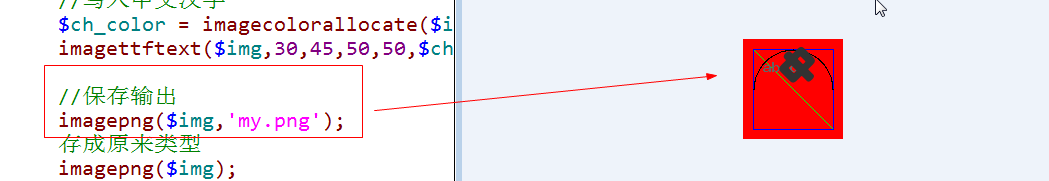
Imagettftext(图片资源,文字大小,旋转角度,起始X,起始Y,颜色,字体,内容)：可以书写任意文字（中文）：需要指定字体路径（ttf文件：默认是windows/fonts/）



## GD图像处理基本技术

### 输出画布资源

1）输出为图片文件：以图片文件形式保存到本地文件夹



2）输出为网页图片：将图片展示给HTML（用户）：服务器需要告知服务器当前内容是图片（修改响应头）



Image+图片格式

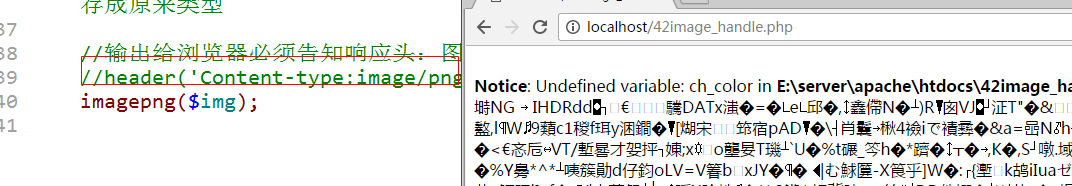
Imagejpeg()：保存成jpeg格式图片

Imagepng()：保存成png格式图片

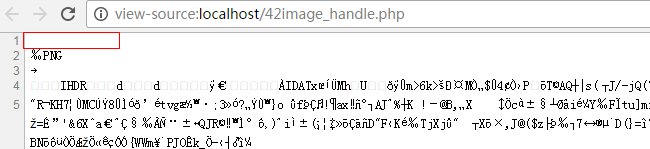
Imagegif()：保存gif格式图片

如果图片只是提供了图片资源，不指定保存的文件位置，那么系统认为是输出给浏览器；如果指定了位置，那么系统认为是保存到本地（第二个参数）；

细节1：如果图片输出或者保存出错，浏览器看到的永远是告诉你图片错了，但是绝对不会告知错误原因在哪，需要关闭header图片输出，再看问题



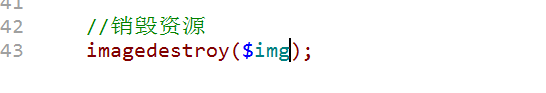
细节2：如果图片输出之后没有成功，但是关闭header也看不到错误：最大的可能是图片输出之前输出的别的额外的内容（喜欢输出pre），应该查看网页原码，看看图片输出之前是否有任何输出：尤其是空格空行



### 销毁画布资源

从内存中将画布资源清理掉，释放内存。

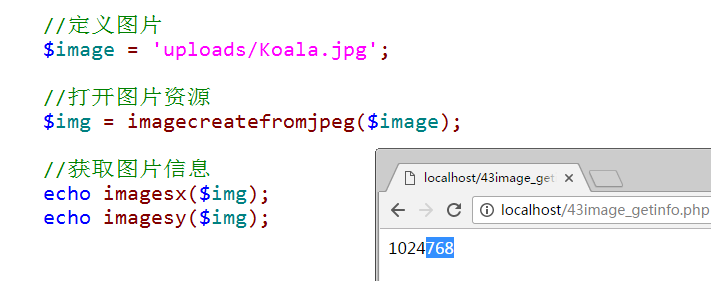
Imagedestroy(画布资源)；



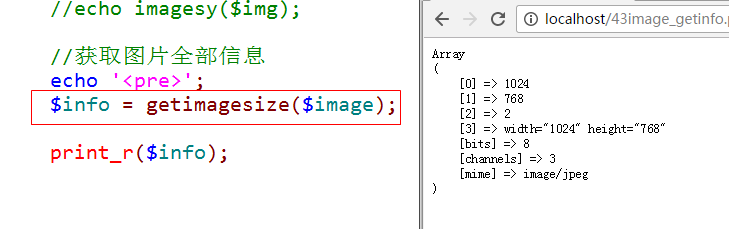
## GD图像处理基本技术

### 获取图片信息

1）获取画布尺寸：imagesx(), imagesy()



2）获取图片尺寸：getimagesize()



## GD图像处理应用案例

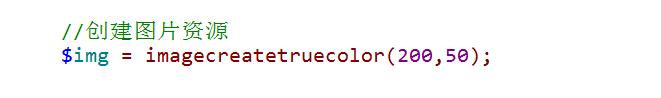
### 验证码的实现

验证码（CAPTCHA）是“Completely Automated Public Turing test to tell Computers and Humans Apart”（全自动区分计算机和人类的图灵测试）的缩写，是一种区分用户是计算机还是人的公共全自动程序。可以防止：恶意破解密码、刷票、论坛灌水，有效防止某个黑客对某一个特定注册用户用特定程序暴力破解方式进行不断的登陆尝试，实际上用验证码是现在很多网站通行的方式，我们利用比较简易的方式实现了这个功能。这个问题可以由计算机生成并评判，但是必须只有人类才能解答。由于计算机无法解答CAPTCHA的问题，所以回答出问题的用户就可以被认为是人类。

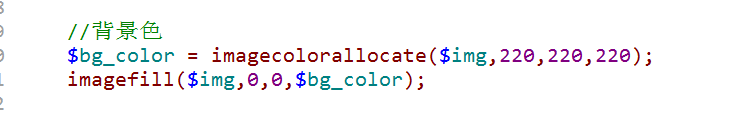
图片验证码：计算机将拿到的验证码存放到图片中，然后用户看到然后识别，然后提交给服务器，服务器再根据用户提交的和服务器之前生成的进行比较。

1）实现验证码图片的展示

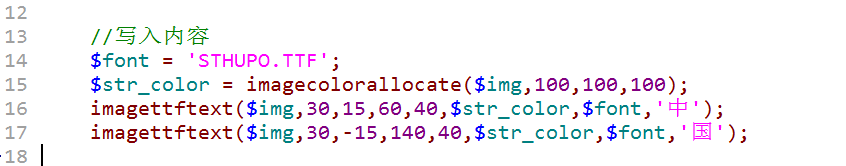
a. 生成图片资源：背景色设定



背景色处理



b. 写入文字：imagestring写的效果一般，所以一般可以使用imagettftext



c. 输出图片给浏览器



d. 关闭资源

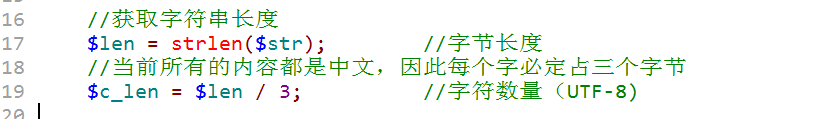


2）实现验证码文字的随机变化：有一串文字可以随机选择

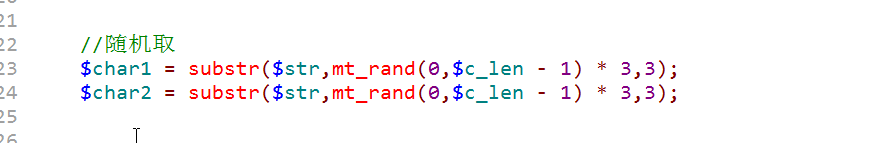
a. 制作目标字符串集：从哪里选内容



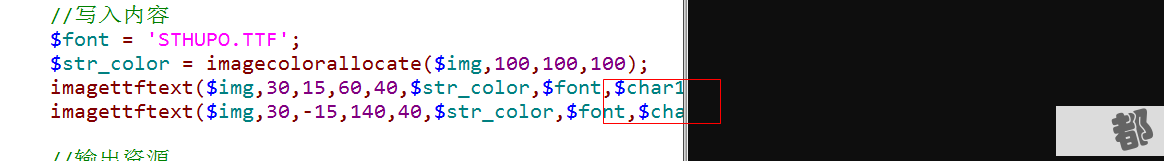
b. 如何随机从字符串中取出对应的汉字：汉字在utf-8字符集中一个字占用3个字节，英文字母只占一个字节：确定字符数



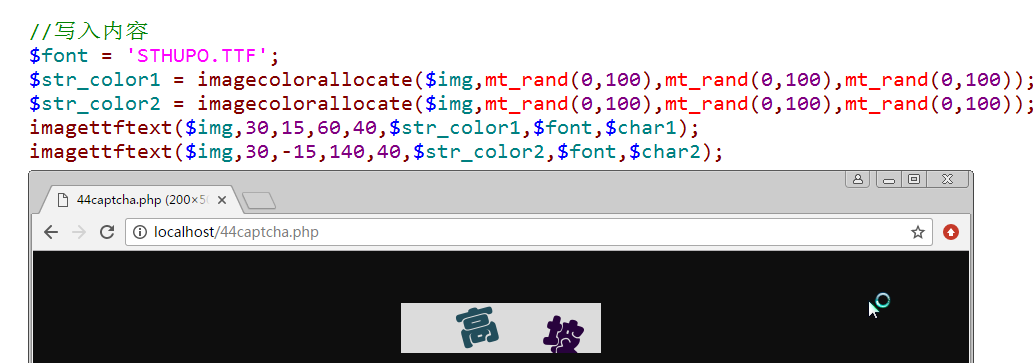
C. 随机取出对应的字符（中文汉字）



d. 将取到的字符放到指定的图片位置即可

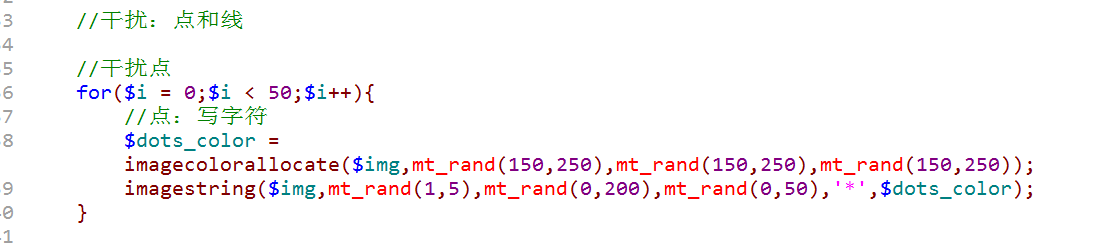


3）实现验证码文字颜色的随机变化：每次产生不同的文字颜色

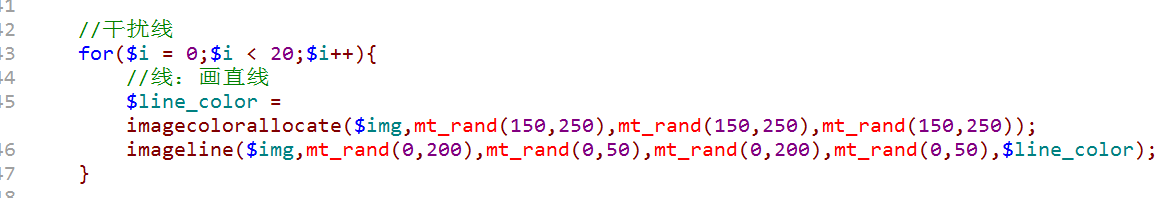


4）实现验证码背景或干扰噪点：增加一些额外的不影响用户看但是会产生模糊效果的内容（点或者直线）：imagestring/imageline

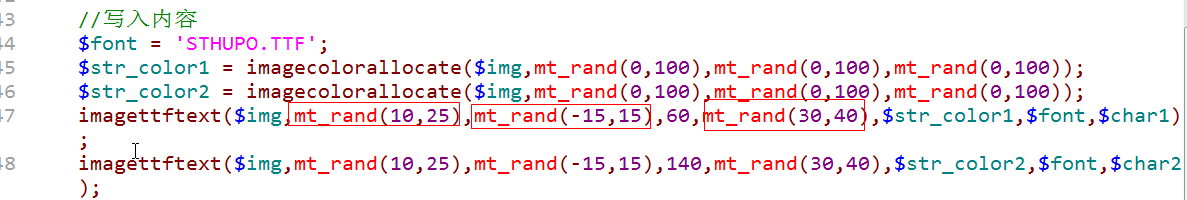
a. 增加干扰点



b. 增加干扰线



c. 改变文字的大小和位置以及其他可变的信息

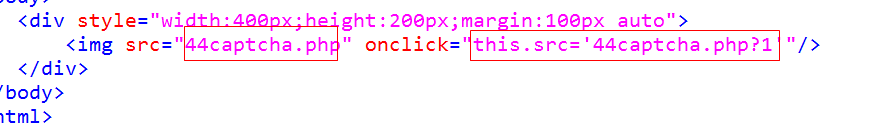


5）实现点击刷新验证码功能：实现验证码在浏览器显示的功能

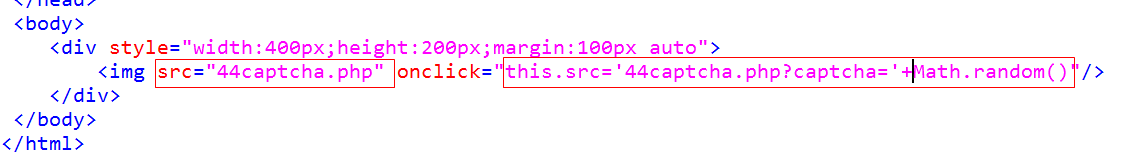
a. 创建一个表单文件，里面有一个img标签能够显示图片：宽200，高50



b. 实现点击更换验证码：让src重新请求PHP脚本，产生一张新的图片。因此需要给img标签增加一个点击事件：img的src是否重新发起请求，取决于浏览器；浏览器是否重新发起请求取决于src是否改变。



C. 如何让img标签的src每次点击都不一样？就可以解决问题



## GD图像处理应用案例

### 缩略图的实现

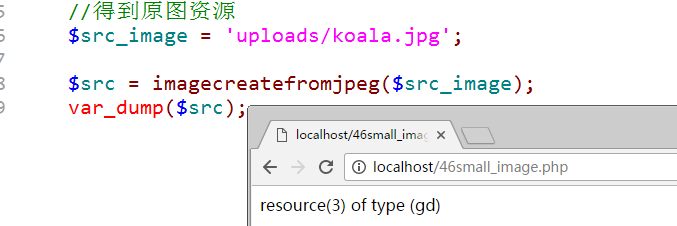
1）制作图片缩略图的原理

缩略图：将原图得到一个较小的图（尺寸上）

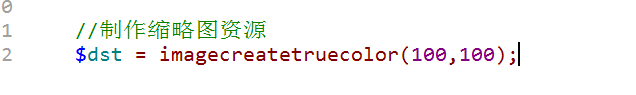
缩略图原理：将原图打开，然后放到另外一个较小的图片资源中，最后进行保存即可。

2）实现固定宽高的缩略图

a. 得到一张原图资源

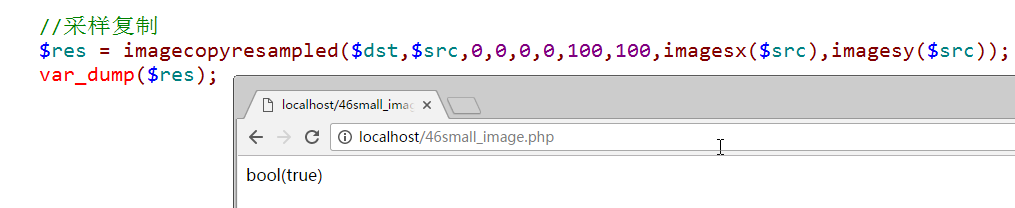


b. 得到一个缩略图资源（较小）

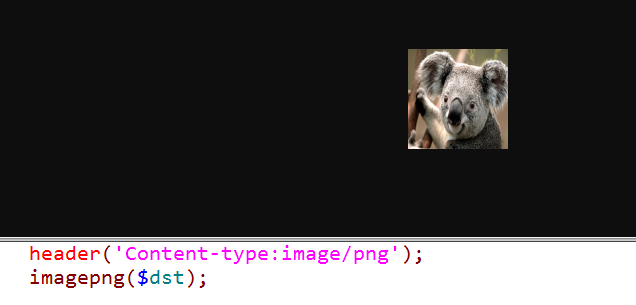


c. 图片采样复制：GD提供了一个函数

imagecopyresamepled(缩略图资源,原图资源,缩略图开始放X,放Y,原图采样其实X,起始Y,缩略图存放宽,存放高,原图采样宽,采样高)

;

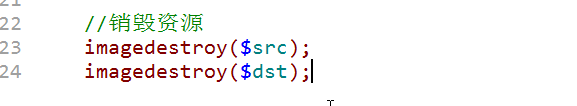
d. 保存缩略图



保存到本地



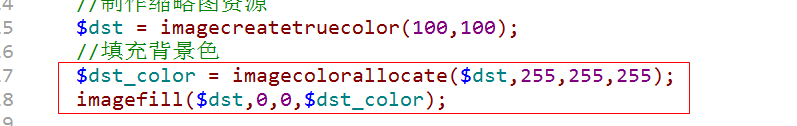
e. 销毁所有资源（原图和缩略图）



3）实现等比例缩放的固定宽或高的缩略图

优点：图片不会失真（变形）

缺点：缩略图有些部分需要进行额外填充（白色填充：补白）



等比例缩略图与固定缩略图的制作区别：在于需要通过计算来得出缩略图的宽和高

算法原理：

1、 计算缩略图宽高比

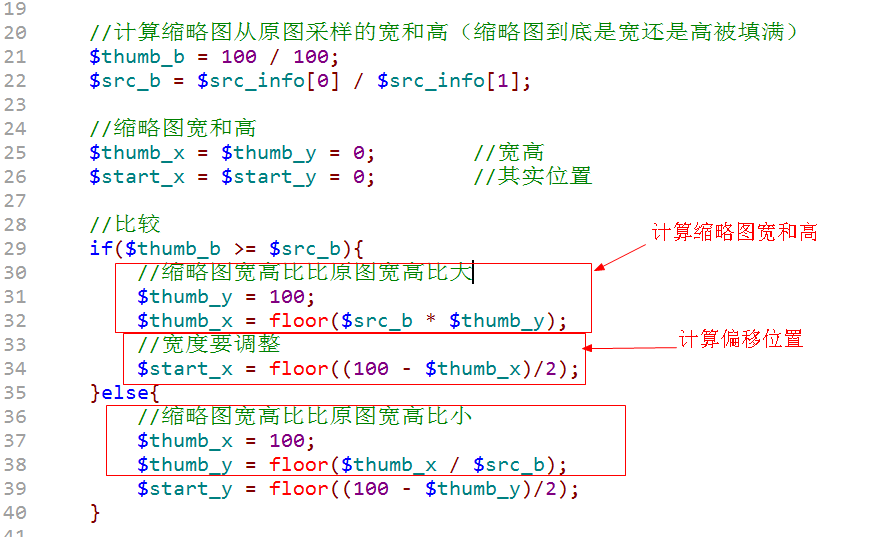
2、 计算原图宽高比

3、 比较：

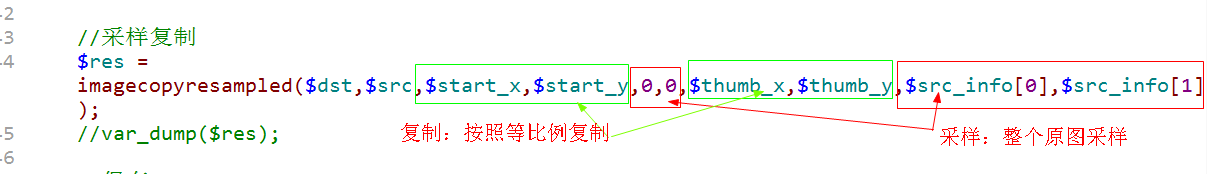
a) 如果缩略图宽高比大于原图宽高比，将缩略图中用原图的高尽可能填满：缩略图的高是完整的，宽度不够（补白）

b) 如果缩略图宽高比小于原图宽高比，将缩略图中用原图的宽尽可能填满：缩略图的宽是完整的，高度不够（补白）

4、 将图片放到缩略图中间



在采样过程中，将得到的数据进行采样复制



## GD图像处理应用案例

### 水印图的实现

水印图：watermark，在某个图片上增加一个透明的印记（马赛克）

水印图用途：版权操作

1）制作图片水印图的原理

水印图制作原理：将一个带有明显标志的图片放到另外一张需要处理的图片之上

a. 获取原图资源（需要放上水印图）

b. 获取水印图资源

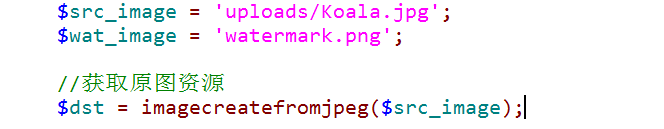
c. 合并图片（把水印图合到目标图上）

d. 保存输出

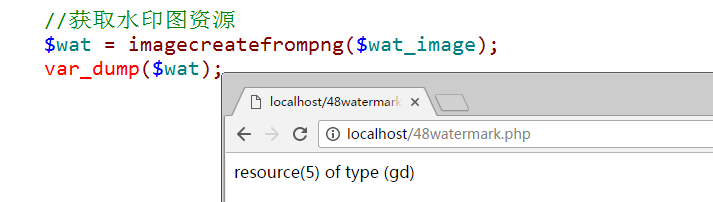
e. 清除资源

2）实现固定位置的水印图：左上角

a. 获取原图资源（需要放上水印图）

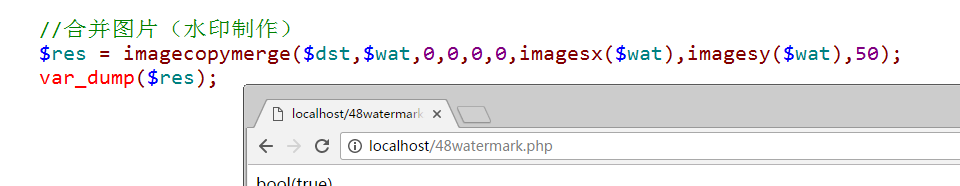


b. 获取水印图资源

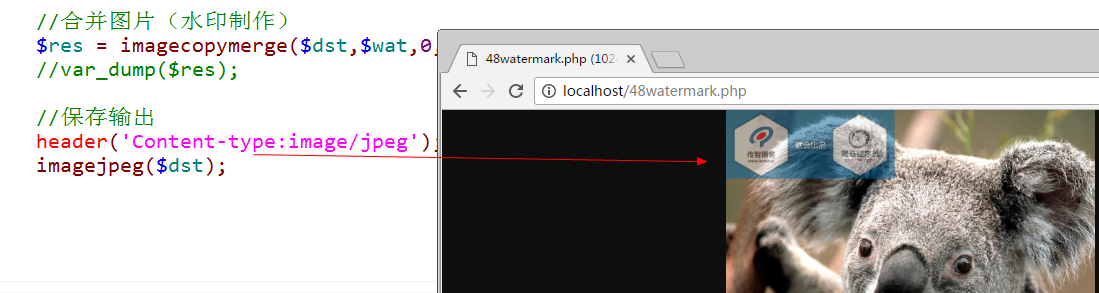


c. 合并图片（把水印图合到目标图上）

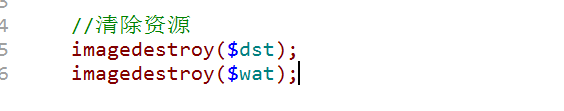
imagecopymerge(目标资源,水印资源,目标起始X,起始Y,水印起始X,起始Y,水印宽,水印高,透明度)



d. 保存输出

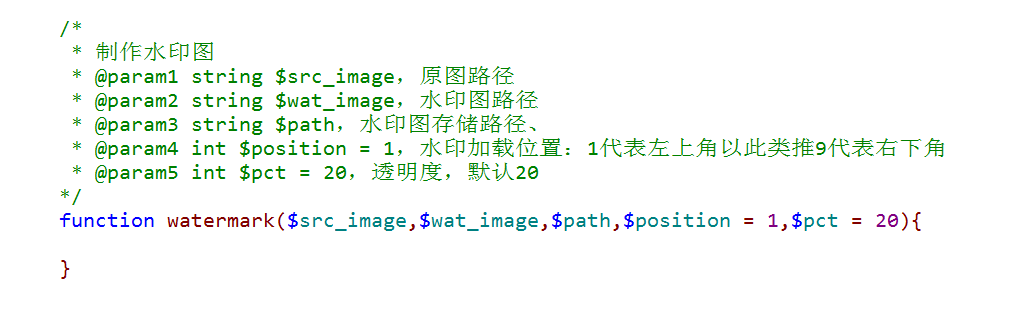


e. 清除资源

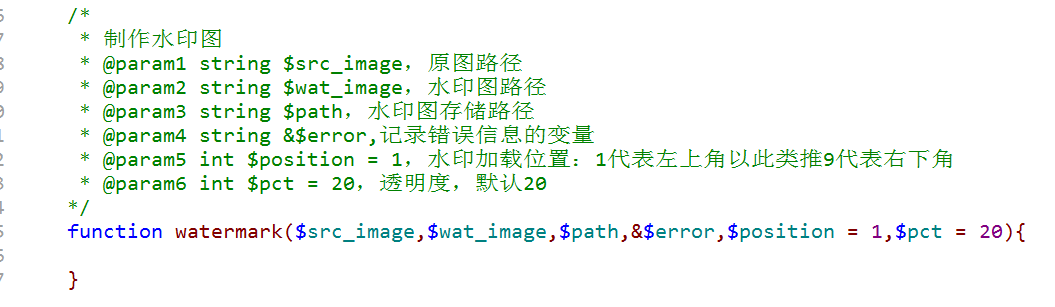


3）实现可选9个位置的水印图：封装制作水印图的函数

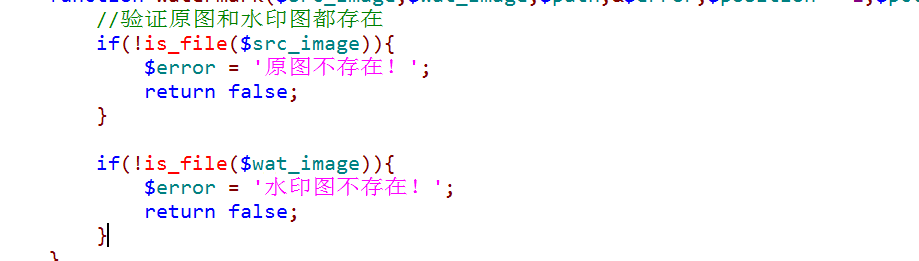
a. 创建一个制作水印图的函数结构：制作水印图需要提供多少条件？原图资源，水印图资源，位置选择？9个，透明度（保存位置）



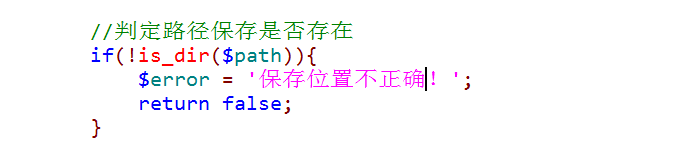
b. 结果是希望产生水印图：但是可能产生成功，返回文件保存名字；如果产生失败，应该返回false，但是还需要告知外界原因：通过引用传参解决



C. 水印图前提：原图和对应的水印图都存在



d. 判定保存路径是否存在

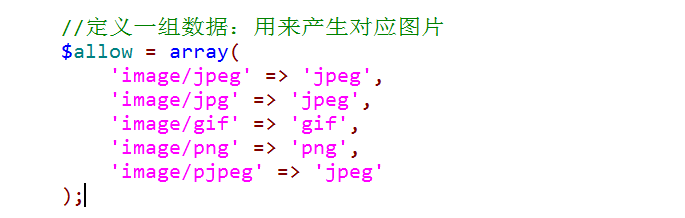


e. 打开原图和水印图资源

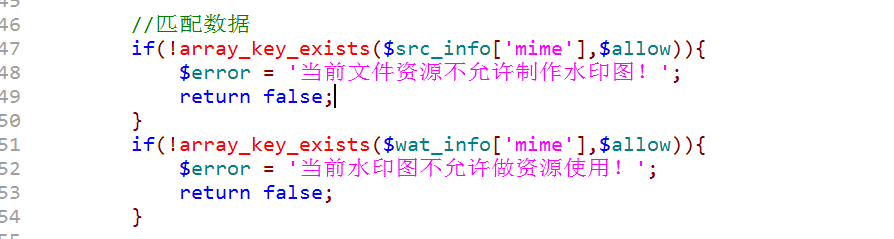
1） 想办法确定要什么函数来打开图片资源：需要通过图片MIME类型类确定，获取图片信息



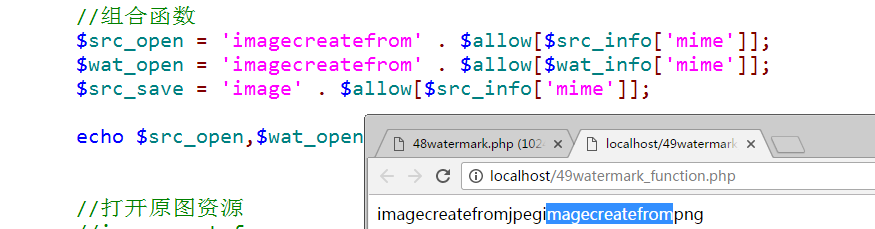
2）通过MIME类型得到要打开图片的函数：先设定一个数组进行匹配，匹配成功自动构造创建函数（保存函数），失败则提示错误



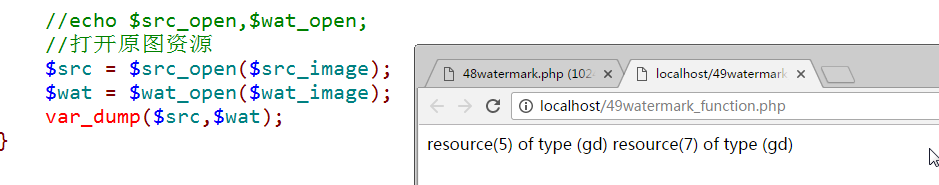
3） 匹配数据



4） 组合函数名字：打开原图资源函数，打开水印图函数，保存水印完成图函数



5） 打开图片资源

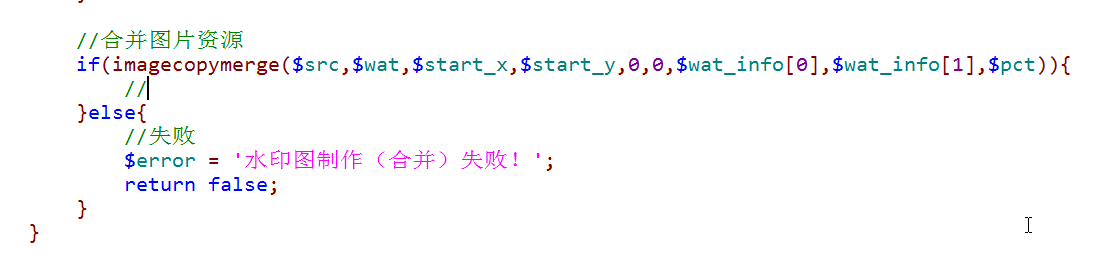


f. 合并图片资源：产生水印：位置需要计算

1） 计算水印图在原图中的位置



2） 合并图片资源



g. 保存水印图片和销毁资源



h. 测试：函数调用的时候，可以通过结果判定操作是否成功

