code-sucks.com

# Use PHP To Resize an Images

So you like to have rich images in your website or blog. But hate having to resize your images. Most blog software can create thumbnails for you. But what if you want different size images then what the blog software provides.

Now you could could let your browser resize the image for you. But this makes the quality of the screen shot and makes it look like an ugly block of pixels. Plus the file size of the image is not reduced with it's new size like the image to the left at a whopping 319kb. So why not write your own script to resize the image dynamically and the file size.

Writing your our own php script to resize the image dynamically will control both the image size and the file size. Check out the third image to the right - which was resized with a php script. You should notice the screenshot looks cleaner and more professional - internet browsers simply are not designed to resize images.

This is how do we do this. You going to make use of the PHP GD library to manipulate the image, and a $_GET parameter to tell PHP which image to re-size. First, you need to open up your main screenshot file and get it’s size and type.

```
$src_img = imagecreatefrompng('image.png');
$srcsize = getimagesize('image.png');
```

imagecreatfrompng() creates an image object and stores it in $src_image. If the file is a different type you can replace “png” with “jpeg” or “gif.” getimagesize() stores an array in $srcsize that contains the width of the image ($srcsize[0]), the height of the image ($srcsize[1]), and the filetype of the image ($srcsize[2]). Now you need to create new dimensions for the new image and actually create an image. Let's resize the screenshot to 200 pixels wide - and then adjust the height to maintain the ratio.

```
$dest_x = 200;
$dest_y = (200 / $srcsize[0]) * $srcsize[1];
$dst_img = imagecreatetruecolor($dest_x, $dest_y);
```

imagecopyresampled() copies an image ($dst_img) into a new image ($src_img) with a set of given attributes (including our sizes). The “Header” line tells the browser that this is an image - so that it’s not displayed as a bunch of gibberish. Finally, the imagepng() function actually displays the image. Now, you need to clean up and get rid of the image objects. Otherwise, you could end up eatting a lot of memory from the server.

```
imagedestroy($src_img);
imagedestroy($dst_img);
```

If you load the script directly, you should simply see the image. PHP creates an image resource and displays it in the browser - as if “resize-image.php” was really “image.png.” Therefore you need to include this new image in an HTML page is use the php script as the src attribute of our <img> tag. Like this.

```
<img src='resize-image.php' />
```

The script will execute, create an image, and use that as the src. The last thing you need to do is modify your script so that we can pass it some information. It wouldn't any good if you had to write a new php script for every image you wanted to resize. Instead, we can use the $_GET array to send a variable (the image filename) to our image resizing script. Here's the entire image resizing script, with the $_GET variable used for the filename.

```
//  Create source image and dimensions
$src_img = imagecreatefrompng($_GET['image']);
$srcsize = getimagesize($_GET['image']);

$dest_x = 200;
$dest_y = (200 / $srcsize[0]) * $srcsize[1];

$dst_img = imagecreatetruecolor($dest_x, $dest_y);

//  Resize image
imagecopyresampled($dst_img, $src_img, 0, 0, 0, 0, $dest_x, $dest_y, $srcsize[0], $srcsize[1])


//  Output image
header("content-type: image/png");
imagepng($dst_img);

//  Deletes images
imagedestroy($src_img);
imagedestroy($dst_img);
```

To use this, you need to slightly modify your HTML and add an "image" parameter to the URL of your script. Like so…

<img src='resize-image.php?image=image.png' />

Now you can resize images on the fly the way you want, and when you want.