COMP 1920

Server Side Web Scripting with PHP

Lesson 08

Creating Images by Example

There are several useful functions which allow PHP authors to create images. First let's look at an example. The commented numbers will be discussed more thoroughly in the section below.

```
// 1
header("Content-Type: image/png");

// 2
$myFirstImage = imageCreateTrueColor(300, 200);

// 3
$purple = imageColorAllocate($myFirstImage, 255, 0, 255);

// 4
$white = imageColorAllocate($myFirstImage, 255, 255, 255);

// 5
imageFill($myFirstImage, 0, 0, $purple);

// 6
imageString($myFirstImage, 5, 20, 30, "I love PHP!", $white);

// 7
imagePNG($myFirstImage);

// 8
imageDestroy($myFirstImage);
```

This should create an image in your browser with the text “I love PHP!” in white, with a purple-ish background.

Example in Detail

header("Content-Type: image/png"); // 1

The first line is a `header()` function call, which tells the browser that a PNG ("portable network graphics") image is going to be output to the browser. Other options include `image/jpeg`. You must send this header always when creating images in PHP, or else your images will not be rendered correctly. Since we have specified a content type of `image/png`, we cannot output text (PHP or HTML), since that would require a header of `text/html`. You can only send one Content-Type header per page.

Optionally, you can ask the browser to download the image by changing the header to . Should the browser choose to respect the header (modern browsers do), the user will be prompted to save the file.

```
header("Content-Disposition: attachment;filename=mySquare.png");
```

$myFirstImage = imageCreateTrueColor(300, 200); // 2

The line is a call to the function `imageCreateTrueColor($widthInPixels, $heightInPixels)`, which creates an empty (black; no color) image. It returns an image handle/variable, which is used for the other function calls.

$purple = imageColorAllocate($myFirstImage, 255, 0, 255); // 3

This line uses the image handle created by `imageCreateTrueColor()`. It allocates a color that can be used with that image. The color is specified by the three numbers `255, 0, 255` in this example.

This uses the RBG (Red Blue Green) color system, where `0, 1, 2` means 0 Red (none at all), 1 Blue (a tiny bit of blue), and 2 Green (a tiny bit more green). Each color can be 255 maximum, so "0, 255, 0" means "all green", for example.

We are describing purple with "255, 0, 255", which we can use for the background of our image.

$white = imageColorAllocate($myFirstImage, 255, 255, 255); // 4

Similar to purple from line 3, we are describing white with "255, 255, 255", which we can use for the text on our image.

imageFill($myFirstImage, 0, 0, $purple); // 5

This line of code floods the specified image with the specified color, beginning at the specified coordinates (0, 0 means "left, top" of the image). Before calling this function, you must set your colors with a call to `imageColorAllocate()`.

imageString($myFirstImage, 5, 20, 30, "I love PHP!", $white); //6

This line writes the specified string on the specified image, beginning at horizontally (20 in the example) and vertically (30 in the example).Note that the y-value indicates the placement of the top of the text.

The "5" refers to which font is used...fonts range from 1 to 5; any value less than 1 will be rounded up to 1 and any value greater than 5 will be rounded down to 5.

Note that strings are not nicely wrapped automatically using the `imageString()` function. We can use the function `imagestringup()` to write text vertically.

imagePNG($myFirstImage); // 7

This important function outputs the PNG image to the browser. It takes in the image handler as its first parameter.

If you provide a string as the second parameter, it will treat the string as a file path and save the image to that location. For example,

```
imagepng($myFirstImage, "aSquare.png");
```

This will save the image to the local directory as `aSquare.png`.

imageDestroy($myFirstImage); // 8

When your image has been created and output, always free up the memory by calling this function.

Saving to Local File System

Using the same example, but omitting the header and providing a second parameter in the output function, we can

- create an image,
- save it to the file system,
- and provide a link for download

```
$myFirstImage = ImageCreateTrueColor(300, 200);

$purple = imageColorAllocate($myFirstImage, 255, 0, 255);

$white = imageColorAllocate($myFirstImage, 255, 255, 255);

imageFill($myFirstImage, 0, 0, $purple);

imageString($myFirstImage, 5, 20, 30, "I love PHP!", $white);

imageJPEG($myFirstImage, "a.jpg", 100);

echo '<img src="a.jpg"/>';
```

Note in this case we used the `imageJPEG()` functioninstead of `imagePNG()`.

More Drawing Functions

These functions permit you to easily draw shapes other than (and including) rectangles.

imagearc()

Draws an arc (or a circle) with the following list of parameters

```
imageArc(
    $image,
    $center_x_coordinate,
    $center_y_coordinate,
    $ellipse_width,
    $ellipse_height,
    $arc_start_point_degrees, //zero degrees is at three o'clock
    $arc_end_point_degrees,
    $color
);
```

In practice, the code will look like this

```
header("Content-Type: image/png");

$myArcImage = imageCreateTrueColor(400, 400);
$black = imageColorAllocate($myArcImage, 0, 0, 0);
$white = imageColorAllocate($myArcImage, 255, 255, 255);
imageFill($myArcImage, 0, 0, $black);
imageArc($myArcImage, 200, 200, 400, 400, 180, 0, $white);

imagePNG($myArcImage);
imageDestroy($myArcImage);
```

You can create a circle by setting the start to zero degrees and the end to 360 degrees.

```
imageArc($myArcImage, 200, 200, 400, 400, 0, 360, $white);
```

The following functions all work similarly to the `imageArc()` function.

- `imageellipse`
- `imagefilledarc`
- `imagefilledellipse`
- `imagefilledpolygon`
- `imagefilledrectangle`
- `imagerectangle`

imageCreateFromPng()

This function can be used to copy existing images and working on these copies, for instance to create resized versions of them (thumbnails, for instance).

```
$copy = imageCreateFromPng("image.png");
```

imageCopyResampled()

This function is used to copy and resize (part of) an image, with very good quality. It has the following parameters

```
imageCopyResampled(
    $newImageFile,
    $oldImageFile,
    $newImageStartCoordinateX,
    $newImageStartCoordinateY,
    $oldImageStartCoordinateX,
    $oldImageStartCoordinateY,
    $newImageWidth,
    $newImageHeight,
    $oldImageWidth,
    $oldImageHeight
);
```

Note that you will have to create `$newImage` yourself, for instance by calling `imageCreateTrueColor()`. See next section for an example.

imageRotate()

To rotate an existing image, `imageRotate()` has the following parameters

```
$rotatedImage = (
    $original_image,
    $angle_degrees,
    $background_color
);
```

To rotate an existing image, use the following steps

1. Create a new image from the original
2. Rotate the new image using the `imageRotate()` function
3. Output the image to the screen

```
header("Content-Type: image/jpeg");

$rotatedImage = imageCreateFromJpeg("smile.jpg");

$grey = imagecolorallocate($rotatedImage, 100, 100, 100);

$rotatedImage = imageRotate($rotatedImage, 45, $grey);

imageJpeg($rotatedImage);
imageDestroy($rotatedImage);
```

imagesx() and imagesy()

imagesx() will return the width of the image (in pixels) and imagesy() will return the height

```
$image = imageCreateFromJpeg("smile.jpg");
$width = imagesx($image);
$height = imagesy($image);
echo $width;
echo $height;
```

Example Thumbnail Generator

This creates a 125 x 125 square pixel thumbnail of the larger "smile.jpg" image.

```
$src_img = imagecreatefromjpeg("smile.jpg");
$dst_img = ImageCreateTrueColor(125, 125);

imagecopyresampled(
    $dst_img,
    $src_img,
    0,0,0,0,
    125,125,
    imagesx($src_img),
    imagesy($src_img)
);

// 100 means best quality (0 is worst)
imagejpeg($dst_img, "newsmile.jpg", 100);

echo "newsmile.jpg thumbnail generated...<br>";

echo '<a href="newsmile.jpg">Click here </a>';
```