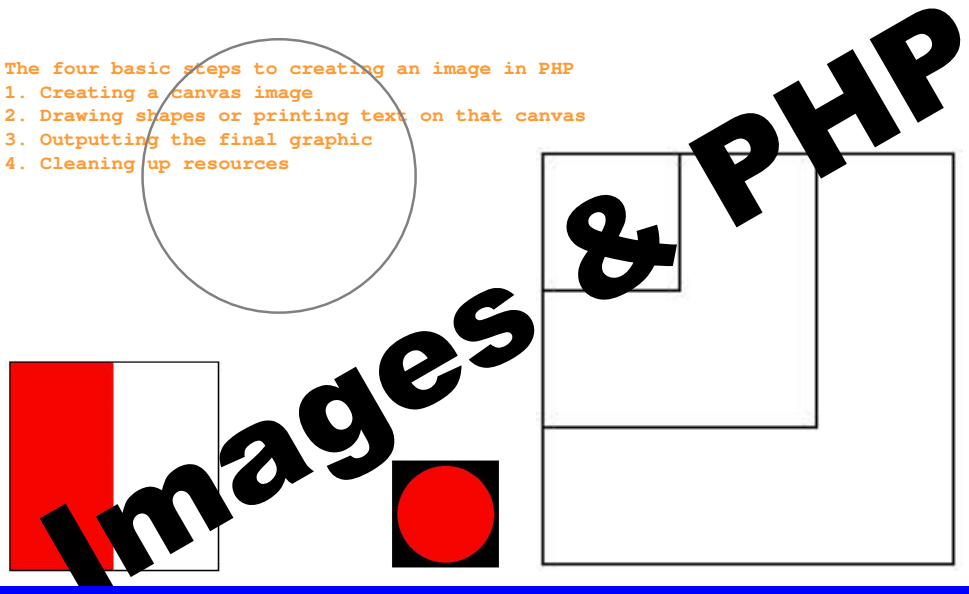
```
The four basic steps to creating an image in PHP
1. Creating a canvas image
2. Drawing shapes or printing text on that canvas
3. Outputting the final graphic
4. Cleaning up resources
```



08 Images & PHP

Images & PHP

- Understanding image formats
- Creating images
- Using text and fonts to create images
- Drawing figures and graphing data
- Workshop

Images & PHP

Understanding image formats

- The gd library supports JPEG, PNG, and WBMP formats. It no longer supports the GIF (GD v.1.6 - Before v.2.0.28)
  - **JPEG** (pronounced "jay-peg") actually stands for *Joint Photographic Experts Group*
  - **PNG** (pronounced "ping") stands for *Portable Network Graphics*.
    - This file format is the replacement for *GIF (Graphics Interchange Format)*
    - The PNG Web site describes it as "a turbo-studly image format with lossless compression."
  - **WBMP** stands for *Wireless Bitmap*. It is a file format designed specifically for wireless devices.
  - **GIF** stands for Graphics Interchange Format.
    - Standard GIFs use a form of compression known as *LZW (Lempel Ziv Welch)*, which is subject to a patent owned by UNISYS.
    - Providers of programs that read and write GIFs must pay licensing fees to UNISYS.

Images & PHP

Creating images

- The four basic steps to creating an image in PHP are as follows:
  - Creating a canvas image on which to work.
    - resource **imagecreate** (int x_size, int y_size)
      - » **imagecreate()** returns an image identifier representing a blank image of size *x_size* by *y_size*
    - int **imagecolorallocate** (resource image, int red, int green, int blue)
      - » **imagecolorallocate()** returns a color identifier representing the color composed of the given RGB components.
  - Drawing shapes or printing text on that canvas.
    - bool **imagefill** (resource image, int x, int y, int color)
      - » **imagefill()** performs a flood fill starting at coordinate *x, y* (top left is 0, 0) with color *color* in the image *image*.
    - bool **imageline** (resource image, int x1, int y1, int x2, int y2, int color)
      - » **imageline()** draws a line from *x1, y1* to *x2, y2* (top left is 0, 0) in image *image* of color *color*.
    - bool **imagestring** (resource image, int font, int x, int y, string s, int color)
      - » **imagestring()** draws the string *s* in the image identified by *image* with the upper-left corner at coordinates *x, y* (top left is 0, 0) in color *color*. If font is 1, 2, 3, 4 or 5, a built-in font is used.

Creating images (cont.)

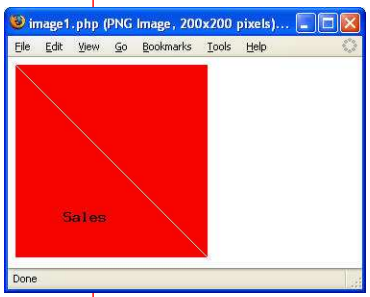
- Outputting the final graphic.
  - void **header** (string string [, bool replace [, int http_response_code]])
    - » **header()** is used to send raw HTTP headers.
  - bool **imagepng** (resource image [, string filename])
    - » The **imagepng()** outputs a GD image stream (*image*) in PNG format to standard output (usually the browser) or, if a filename is given by the *filename* it outputs the image to the file.
- Cleaning up resources.
  - bool **imagedestroy** (resource image)
    - » **imagedestroy()** frees any memory associated with image *image*.

Creating images (cont.)

- Creating a canvas image
  - Create a blank canvas
    - resource **imagecreate** (int x_size, int y_size)
  - Read in an existing image file that you can then filter, resize, or add to.
    - imagecreatefromPNG() or imagecreatefromJPEG()
    - Each of these takes the filename as a parameter, as in, for example, $im = imagecreatefromPNG('baseimage.png');
- Drawing or printing text on to the image
  - There are really two stages to drawing or printing text on the image.
    - First, you must select the colors in which you want to draw.
    - Second, to actually draw into the image, a number of different functions are available, depending on what you want to draw–lines, arcs, polygons, or text.
    - The drawing functions generally require the following as parameters:
      - » The image identifier
      - » The start and sometimes the end coordinates of what you want to draw
      - » The color you want to draw in
      - » For text, the font information

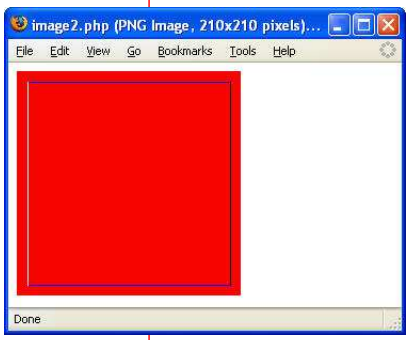
Creating images (cont.)

```
<?php
   // 1. set up image
   $width = 200;
   $height = 200;
   $img = imagecreate($width, $height)
         or die("Cannot Initialize new GD image stream");
   $white = imagecolorallocate ($img, 255, 255, 255);
   $black = imagecolorallocate ($img, 0, 0, 0);
   $red = imagecolorallocate ($img, 255, 0, 0);

   // 2. draw on image
   imagefill($img, 0, 0, $red);
   imageline($img, 0, 0, $width, $height, $white);
   imagestring($img, 5, 50, 150, 'Sales', $black);

   // 3. output image
   header ('Content-type: image/png');
   imagepng ($img);

   // 4. clean up
   imagedestroy($img);
?>
```



Creating images (cont.)

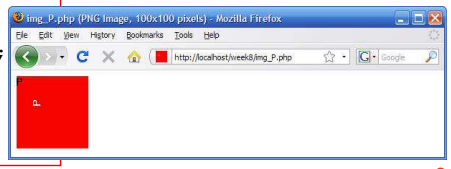
```
<?php
   // 1. set up image
   $img = ImageCreate(210, 210)
         or die("Cannot Initialize new GD image stream");
   $white = ImageColorAllocate ($img, 255, 255, 255);
   $black = ImageColorAllocate ($img, 0, 0, 0);
   $red = ImageColorAllocate ($img, 255, 0, 0);
   $blue = ImageColorAllocate ($img, 0, 0, 255);

   // 2. draw on image
   ImageFill($img, 0, 0, $red);
   ImageLine($img, 10, 10, 200, 10, $blue);
   ImageLine($img, 200, 10, 200, 200, $black);
   ImageLine($img, 200, 200, 10, 200, $blue);
   ImageLine($img, 10, 200, 10, 10, $white);

   // 3. output image
   Header ('Content-type: image/png');
   ImagePng ($img);

   // 4. clean up
   ImageDestroy($img);
?>
```



Creating images (cont.)

- bool **imagechar** (resource image, int font, int x, int y, string c, int color)
  - **imagechar()** draws the first character of *c* in the image identified by *image* with its upper-left at *x,y* (top left is 0, 0) with the color *color*. If *font* is 1, 2, 3, 4 or 5, a built-in font is used (with higher numbers corresponding to larger fonts)
- bool **imagecharup** (resource image, int font, int x, int y, string c, int color)
  - **imagecharup()** same as imgechar(), but draws a first character in string *c* vertically

```
<?php
$img = imagecreate(100, 100);
$string = 'PHP';
$white = imagecolorallocate($img, 255, 255, 255);
$red = imagecolorallocate($img, 255, 0, 0);
$black = imagecolorallocate($img, 0, 0, 0);
// prints a black "P" in the top left corner
imagefill($img, 0, 0, $red);
imagechar($img, 5, 0, 0, $string, $black);
imagecharup($img, 5, 20, 40, $string, $white);
header('Content-type: image/png');
imagepng($img);
imagedestroy($img);
?>
```

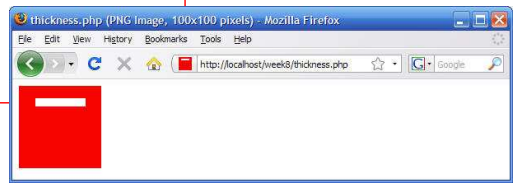


Images & PHP
523313 Web Applications

Creating images (cont.)

- bool **imagestring** (resource image, int font, int x, int y, string s, int color)
  - **imagestring()** draws the string *s* in the image identified by *image* at coordinates *x*, *y* (top left is 0, 0) in color *col*. If font is 1, 2, 3, 4 or 5, a built-in font is used.
- bool **imagestringup** (resource image, int font, int x, int y, string s, int color)
  - **imagestringup()** same as imagestring(), but draws a string vertically
- bool **imagesetthickness** (resource image, int thickness)
  - **Imagesetthickness()** sets the thickness for line drawing

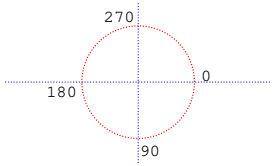
```
<?php
$img = imagecreate(100, 100);
$red = imagecolorallocate($img, 255, 0, 0);
$white = imagecolorallocate($img, 255, 255, 255);
imagesetthickness($img, 10);
imageline($img, 20, 20, 80, 20, $white);
header('Content-type: image/png');
imagepng($img);
imagedestroy($img);
?>
```



Images & PHP
523313 Web Applications

Creating images (cont.)

- Drawing a circle with imagearc()
  - bool **imagearc** (resource image, int cx, int cy, int w, int h, int s, int e, int color)
    - **imagearc()** draws a partial ellipse
      - Centered at ***cx***, ***cy*** (top left is 0, 0) in the image represented by ***image***.
      - ***w*** and ***h*** specifies the ellipse's width and height respectively
      - The start and end points are specified in degrees indicated by the ***s*** and ***e*** arguments.
      - 0° is located at the **three-o'clock** position, and the arc is drawn clockwise.



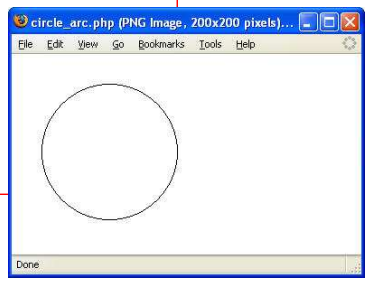
  - bool **imagefilledellipse** (resource image, int cx, int cy, int w, int h, int color)
    - **imagefilledellipse()** draws an ellipse
      - Centered at ***cx***, ***cy*** (top left is 0, 0) in the image represented by *image*.
      - ***w*** and ***h*** specifies the ellipse's width and height respectively.
      - The ellipse is filled using ***color***.
      - Returns **TRUE** on success or **FALSE** on failure

Images & PHP
523313 Web Applications

Creating images (cont.)

- Example of using imagearc()

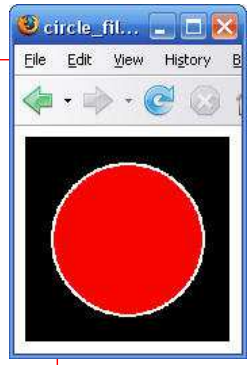
```
<?php
   // create a 200*200 image
   $img = imagecreate(200, 200);

   // allocate some color
   $white = imagecolorallocate($img, 255, 255, 255);
   $black = imagecolorallocate($img, 0, 0, 0);

   // draw a black circle
   imagearc($img, 100, 100, 150, 150, 0, 360, $black);

   // output image in the browser
   header("Content-type: image/png");
   imagepng($img);

   // free memory
   imagedestroy($img);
?>
```



Creating images (cont.)

- Example of using imagefilledellipse ()

```
<?php
  // create a blank image
  $height = 200;
  $width = 200;
  $img = imagecreate($width, $height)
         or die("Cannot Initialize new GD image stream");
  $white = imagecolorallocate ($img, 255, 255, 255);
  $black = imagecolorallocate ($img, 0, 0, 0);
  $red = imagecolorallocate ($img, 255, 0, 0);

  // fill the background in black color
  imagefill($img, 0, 0, $black);

  // draw the White and Red ellipses
  imagefilledellipse($img, 100, 100, 150, 150, $white);
  imagefilledellipse($img, 100, 100, 145, 145, $red);

  // output the picture
  header("Content-type: image/png");
  imagepng($img);
  imagedestroy($img);
?>
```



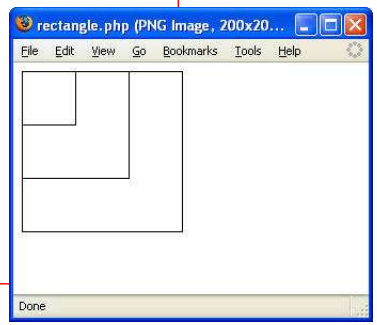
Creating images (cont.)

- Drawing a rectangle with imagerectangle() and imagefilledrectangle()
  - bool **imagerectangle** (resource image, int x1, int y1, int x2, int y2, int col)
    - **imagerectangle()**
      - » creates a rectangle of color ***col*** in image ***image***
      - » starting at upper left coordinate ***x1***, ***y1*** and ending at bottom right coordinate ***x2***, ***y2***.
      - » 0, 0 is the top left corner of the image.
  - bool **imagefilledrectangle** (resource image, int x1, int y1, int x2, int y2, int color)
    - **imagefilledrectangle()**
      - » creates a filled rectangle of color ***color*** in image ***image***
      - » starting at upper left coordinates ***x1***, ***y1*** and ending at bottom right coordinates ***x2***, ***y2***.
      - » 0, 0 is the top left corner of the image.

Creating images (cont.)

- Example of using imagerectangle()

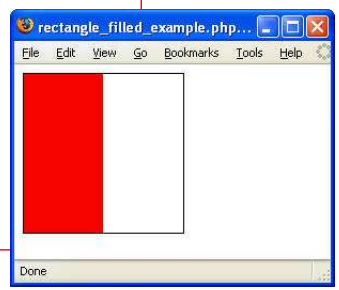
```
<?php
   // create an 200*200 image
   $img = imagecreate(200, 200);

   // allocate some colors
   $white = imagecolorallocate($img, 255, 255, 255);
   $black = imagecolorallocate($img, 0, 0, 0);

   // draw a black rectangle
   imagerectangle($img, 0, 0, 150, 150, $black);
   imagerectangle($img, 0, 0, 100, 100, $black);
   imagerectangle($img, 0, 0, 50, 50, $black);

   // output image in the browser
   header("Content-type: image/png");
   imagepng($img);

   // free memory
   imagedestroy($img);
?>
```



Creating images (cont.)

- Example of using imagrectangle() and imagefilledrectangle()

```
<?php
   // create an 160*160 image
   $img = imagecreate(160, 160);

   // allocate some colors
   $white = imagecolorallocate($img, 255, 255, 255);
   $black = imagecolorallocate($img, 0, 0, 0);
   $red = imagecolorallocate($img, 255, 0, 0);

   // draw a black rectangle
   imagerectangle($img, 0, 0, 150, 150, $black);
   imagefilledrectangle($img, 1, 1, 74, 149, $red);

   // output image in the browser
   header("Content-type: image/png");
   imagepng($img);

   // free memory
   imagedestroy($img);
?>
```



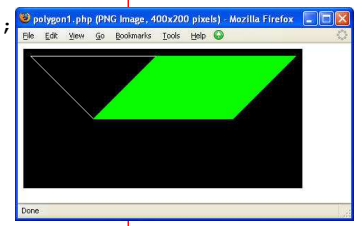
Creating images (cont.)

- Drawing a polygon with imagepolygon() and imagefilledpolygon()
  - bool **imagepolygon** (resource image, array points, int num_points, int color)
    - **imagepolygon()**
      - » creates a polygon in image *image*.
      - » *points* is a PHP array containing the polygon's vertices, i.e. points[0] = x0, points[1] = y0, points[2] = x1, points[3] = y1, etc.
      - » *num_points* is the total number of points (vertices).
  - bool **imagefilledpolygon** (resource image, array points, int num_points, int color)
    - **imagefilledpolygon()**
      - » creates a filled polygon in image *image*.
      - » *points* is an array containing the *x* and *y* co-ordinates of the polygons vertices consecutively.
      - » *num_points* is the total number of vertice

Creating images (cont.)

- Example of using imagepolygon() and imagefilledpolygon()

```
<?php
   // create a blank image
   $img = imagecreate(400,200);
   // fill the background color
   $bg = imagecolorallocate($img, 0, 0, 0);

   // choose a color for the polygon
   $col_poly = imagecolorallocate($img, 255, 255, 255);
   $green = imagecolorallocate($img, 0, 255, 0);

   // draw the polygon
   $draw4 = array (10,10,100,100,300,100,390,10);
   $draw4_in = array (100,100,300,100,390,10,190,10);
   imagepolygon($img, $draw4, 4, $col_poly);
   imagepolygon($img, $draw4_in , 4, $col_poly);
   imagefilledpolygon($img, $draw4_in, 4, $green);

   // output the picture
   header("Content-type: image/png");
   imagepng($img);
   imagedestroy($img);
?>
```



Creating images (cont.)

- Using automatically generated images in other pages
  - Because a header can only be sent once, and this is the only way to tell the browser that we are sending image data, it is slightly tricky to embed any images we create on-the-fly in a regular page. Three ways you can do it are as follows:
    - 1. You can have an entire page consist of the image output, as we did in the previous example.
    - 2. You can write the image out to a file as previously mentioned, and then refer to it with a normal <img> tag.
    - 3. You can put the image production script in an image tag. We have covered methods 1 and 2 already. Let' s briefly look at method 3. To use this method, you include the image inline in HTML by having an image tag along the lines of the following:
      <img src="simplegraph.php" height="200" width="200" alt="Sales going down">

Using text and fonts to create images

- resource **imagecreatefrompng** (string filename)
  - **imagecreatefrompng()** returns an image identifier representing the image obtained from the given filename.
- int **imagesx** (resource image)
  - **imagesx()** returns the width of the image identified by *image*.
- int **imagesy** (resource image)
  - **imagesy()** returns the height of the image identified by *image*.
- bool **putenv** (string setting)
  - **putenv()** sets the value of an environment variable

Using text and fonts to create images

- array **imagettfbbox**(float size, float angle, string fontfile, string text)
  - This function calculates and returns the bounding box in pixels for a TrueType text.
    - *size*
      - » The font size in pixels.
    - *angle*
      - » Angle in degrees in which *text* will be measured
    - *fontfile*
      - » The name of the TrueType font file (can be a URL). Depending on which version of the GD library that PHP is using, it may attempt to search for files that do not begin with a leading '/' by appending '.ttf' to the filename and searching along a library-defined font path.
    - *text*
      - » The string to be measured.

Using text and fonts to create images

- array **imagettfbbox**(float size, float angle, string fontfile, string text)
  - **imagettfbbox()** returns an array with 8 elements representing four points making the bounding box of the text:
    - 0 lower left corner, X position
    - 1 lower left corner, Y position
    - 2 lower right corner, X position
    - 3 lower right corner, Y position
    - 4 upper right corner, X position
    - 5 upper right corner, Y position
    - 6 upper left corner, X position
    - 7 upper left corner, Y position

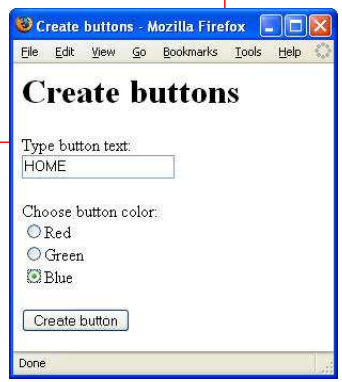
Using text and fonts to create images

- array **imagettftext** (resource image, float size, float angle, int x, int y, int color, string fontfile, string text)
  - *Image:* The image resource.
  - *Size:* The font size.
  - *Angle:* The angle in degrees, with 0 degrees being left-to-right reading text.
  - *X:* The coordinates given by *x* and *y* will define the basepoint of the first character (roughly the lower-left corner of the character)
  - *Y:* The y-ordinate. This sets the position of the fonts baseline, not the very bottom of the character.
  - *Color:* The color index.
  - *Fontfile:* The path to the TrueType font you wish to use.
  - *Text:* The text string.

Using text and fonts to create images

design_button.html

```
<html>
<head><title>Create buttons</title><head>
<body>
 <h1>Create buttons</h1>
  <form method="post" action="make_button.php"> Type button text:<br>
   <input type=text name="button_text"><br /><br>
     Choose button color:<br />
     <input type="radio" name="color" value="red">Red<br>
     <input type="radio" name="color" value="green">Green<br>
     <input type="radio" name="color" value="blue">Blue<p>
     <input type="submit" value="Create button">
   </form>
</body>
</html>
```



Using text and fonts to create images (cont.)

make_button.php

```
<?php
   // check we have the appropriate variable data
   // variables are button-text and color

   $button_text = $_POST['button_text'];
   $color = $_POST['color'];

   if (empty($button_text) || empty($color)){
      echo 'Could not create image - form not filled out correctly';
      exit;
   }

   // create an image of the right background and check size
   $img = imagecreatefrompng ($color.'-button.png');

   $width_image = ImageSX($img);
   $height_image = ImageSY($img);

   // Our images need an 18 pixel margin in from the edge image
   $width_image_wo_margins = $width_image - (2 * 18);
   $height_image_wo_margins = $height_image - (2 * 18);
```

1 of 3

Using text and fonts to create images (cont.)

make_button.php

```
   // Work out if the font size will fit and make it smaller until it does
   // Start out with the biggest size that will reasonably fit on our buttons
   $font_size = 33;
   // you need to tell GD2 where your fonts reside
   putenv('GDFONTPATH=C:\WINDOWS\Fonts');
   $fontname = 'arial';

   do{
      $font_size--;

      // find out the size of the text at that font size
      $bbox=imagettfbbox ($font_size, 0, $fontname, $button_text);
      $right_text = $bbox[2];    // right co-ordinate
      $left_text = $bbox[0];     // left co-ordinate
      $width_text = $right_text - $left_text;  // how wide is it?
      $height_text = abs($bbox[7] - $bbox[1]);  // how tall is it
   }
   while($font_size>8 && ($height_text > $height_image_wo_margins ||
                          $width_text>$width_image_wo_margins));
```

2 of 3

Using text and fonts to create images (cont.)

make_button.php

```
   if($height_text > $height_image_wo_margins ||
      $width_text>$width_image_wo_margins ){
        // no readable font size will fit on button
        echo 'Text given will not fit on button.<br />';
   }else{
       // We have found a font size that will fit
       // Now work out where to put it
       $text_x = $width_image/2.0 - $width_text/2.0;
       $text_y = $height_image/2.0 - $height_text/2.0 ;
       if ($left_text < 0)
          $text_x += abs($left_text);      // add factor for left overhang
       $above_line_text = abs($bbox[7]);   // how far above the baseline?
       $text_y += $above_line_text;        // add baseline factor
       $text_y -= 2;  // adjustment factor for shape of our template
       $white = ImageColorAllocate ($img, 255, 255, 255);
       ImageTTFText($img,$font_size,0,$text_x,$text_y,$white,$fontname,
                $button_text);
       Header ('Content-type: image/png');
       ImagePng ($img);
   }
   ImageDestroy ($img);
?>
```

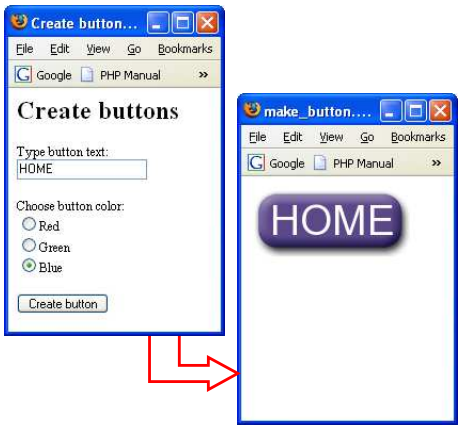
3 of 3

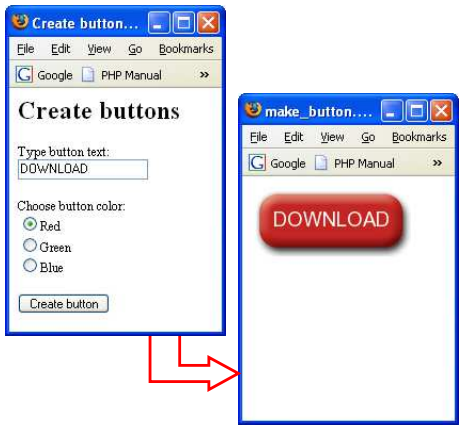
Using text and fonts to create images (cont.)


green-button.png


red-button.png


blue-button.png





Images & PHP

- Workshop
  - Laboratory 8
    - Page 205 (1-5)