LAB 12b

PHP 2: Arrays and Superglobals

What You Will Learn

- How to use PHP arrays
- Hot to use PHP classes
- How PHP makes HTTP variables easily accessible through superglobals
- How to modify the HTTP header

Approximate Time

The exercises in this lab should take approximately 90 minutes to complete.

Fundamentals of Web Development, 3rd Ed

Randy Connolly and Ricardo Hoar

Textbook by Pearson
http://www.funwebdev.com

Date Last Revised: Nov 2, 2020

Arrays

Preparing Directories

1. Like Lab 11, this set of labs requires a functioning webserver to interpret the PHP code. If you do not have a webserver running go back to the last Lab exercises and get one set up.
2. Copy the folder titled `lab12b` to your PHP development location on your development machine (if using XAMPP, this might be `c:/xampp/htdocs`). This `lab12b` folder could be provided by your instructor, or you could clone it from GitHub repo for this lab.

Arrays are important data structures in programming because they allow programmers to manage sets of objects as a single entity. In PHP, arrays take on additional importance in the form of super global arrays that hold data sent from the client to the server.

Exercise 12.1 — Using Arrays

1. Examine `lab12b-ex01.php` in a text editor and in your browser. You are going to replace the hard-coded markup with arrays and loops.
2. You are going to move the stock symbol, name, and price to arrays. Later, you will learn to use more efficient data structures. Define these arrays at the top of the page as follows:

```
<?php
$symbols =

array("ADBE","APPL","ATVI","AMZN","ADSK","FB","GOOGL","MSFT");
$names = ["Adobe Systems Inc.","Apple Inc.",
          "Activision Blizzard","Amazon.com Inc.",
          "Autodesk Inc.","Facebook Inc.",
          "Alphabet Inc.","Microsoft Corporation"];
$prices =
[240.36,218.32,75.42,1760.19,135.25,154.62,1111.39,108.82];
?>
```

Notice that there are two commonly-used ways of defining arrays in PHP.

3. Replace the eight `<div>` for the different stocks and replace it with the following loop:

```
<div class="grid-container">
    <?php for ($i=0; $i < count($symbols); $i++) { ?>
        <div class='box'>
            <h2><?=$symbols[$i]?></h2>
            <h3><?=$names[$i]?></h3>
            <p><?=$prices[$i]?></p>
        </div>
    <?php } ?>
</div>
```

4. Test. The result should look similar to that shown in Figure 12.1.

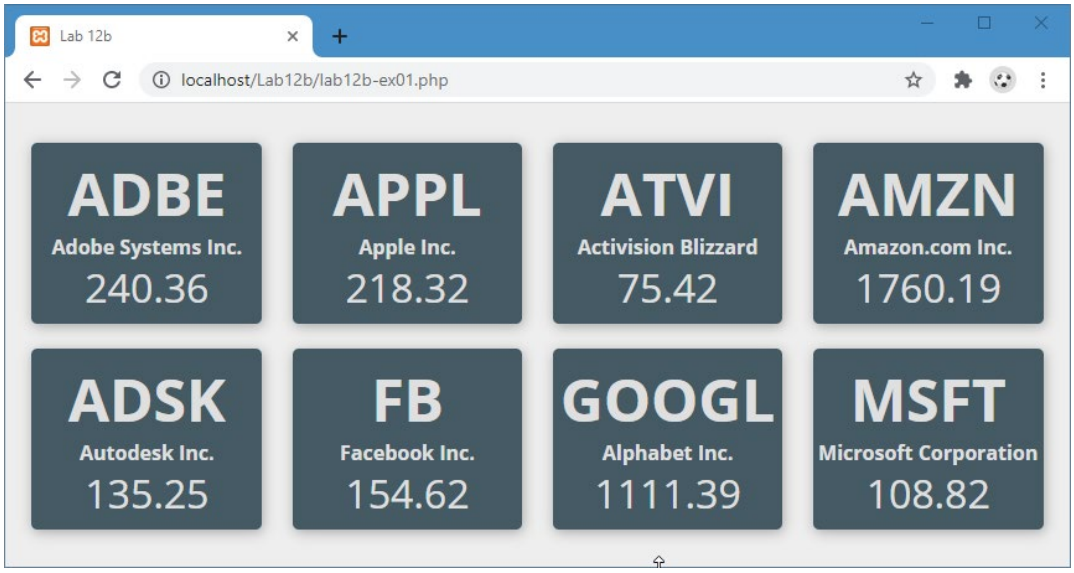

Figure 12.1 – Finished Exercise 12.1

5 Ideally, you want to minimize the amount of PHP code within your markup. Let's refactor your code by defining a function to perform this task as follows:

```
function outputStocks() {
    for ($i=0; $i < count($symbols); $i++) {
        echo "<div class='box'>";
        echo "<h2>$symbols[$i]</h2>";
        echo "<h3>$names[$i]</h3>";
        echo "<p>$prices[$i]</p>";
        echo "</div>";
     }
}
```

6 Replace the loop code in your markup with a call to this function:

```
<div class="grid-container">
    <?php outputStocks(); ?>
</div>
```

7 Test. It won't work. Can you figure out why?

In PHP, a function cannot access variables defined outside that function (i.e., it can only access parameters and variables defined within the function).

8 Revise your function definition as follows:

```
function outputStocks($symbols, $names, $prices) {
```

9 Revise your function invocation as follows and test.

```
<div class="grid-container">
    <?php outputStocks($symbols, $names, $prices); ?>
</div>
```

It should now work.

Exercise 12.2 — Alternate Looping Technique

1 Examine `lab12b-ex02.php` in a text editor. You are going to loop and display the `$stocks` array using an alternate looping technique: the foreach loop.

Add the following loop code and test.

```
<h1>Portfolio Overview</h1>
<div class="data">
    <?php
        foreach ($stocks as $s) {
            echo $s . '<br>';
        }
    ?>
</div>
```

This approach is often preferred as it simplifies your code.

2 PHP has many powerful array manipulation functions. Perhaps the most common is the sort function. Add the following code and test.

```
<h1>Portfolio Overview</h1>
<div class="data">
    <?php
        sort($stocks);
        foreach ($stocks as $s) {
            echo $s . '<br>';
        }
    ?>
</div>
```

Arrays can have multiple dimensions. In PHP, this is achieved by having arrays within arrays.

Exercise 12.3 — two-Dimensional Arrays

1 Examine `lab12b-ex03.php` in a text editor and in your browser. Notice the `points` attribute within the `<polyline>` element in the `<svg>` element? You are going to replace the hard-coded markup with a 2D array and a loop.

2 Add the following two-dimensional array at the top of `lab12b-ex03.php`.

```
<?php
$data = [
            [10,10],
            [60,60],
            [110,80],
            [160,20],
            [210,80],
            [260,120],
            [310,150],
            [390,110],
            [440,10],
            [490,190]
        ];
?>
```

3 Replace the hard-coded `points` attribute with the following and test.

```
<polyline
     points="
<?php
    foreach ($data as $point) {
        echo $point[0] . ',' . $point[1] . ' ';
    }
?>
       "/>
```

Array keys in PHP can be integer or strings. Using string keys can often make your array code more intuitive. Arrays with non-consecutive integers as keys are often referred to as associative arrays.

Exercise 12.4 — Associative Arrays

1 Examine `lab12b-ex04.php` in a text editor. Add the following array at the top of the page:

```
<?php
$portfolio = ["AMZN" => 25,"APPL" => 20,"MSFT" => 15,
              "GOOGL" => 40];
?>
```

2 You can examine any element in an associative array via its key. For instance, add this code and test.

```
<h1>Portfolio Overview</h1>
<div class="data">
    <?php
    echo $portfolio["MSFT"];
    ?>
</div>
```

3 PHP has a special `foreach` syntax for looping through an associate array. Modify your code as follows and test:

```
<h1>Portfolio Overview</h1>
<div class="data">
    <?php
    foreach ($portfolio as $key => $value) {
        echo $key . ' - ' . $value . '%<br>';
    }
    ?>
</div>
```

4 Modify the portfolio array as follows:

```
$portfolio = [ "AMZN"  => [25,1760.19,"Amazon.com Inc."],
               "APPL"  => [20,218.32,"Apple Inc."],
               "MSFT"  => [15,108.82,"Microsoft Corporation"],
               "GOOGL" => [40,1111.39,"Alphabet Inc."]
             ];
```

This essentially is a two-dimensional associative array (the value of each outer array element is another array).

5 Change the code from step 3 as follows:

```
<h1>Portfolio Overview</h1>
<div class="data">
    <?php
    foreach ($portfolio as $key => $value) {
        echo "<h2>$key</h2>";
        echo "<h3>$value[2]</h3>";
        echo "<p>$value[0]% at $" . $value[1] . "</p>";
    }
    ?>
</div>
```

Notice that the `$value` variable is another array. The result should look similar to the Figure 12.2.

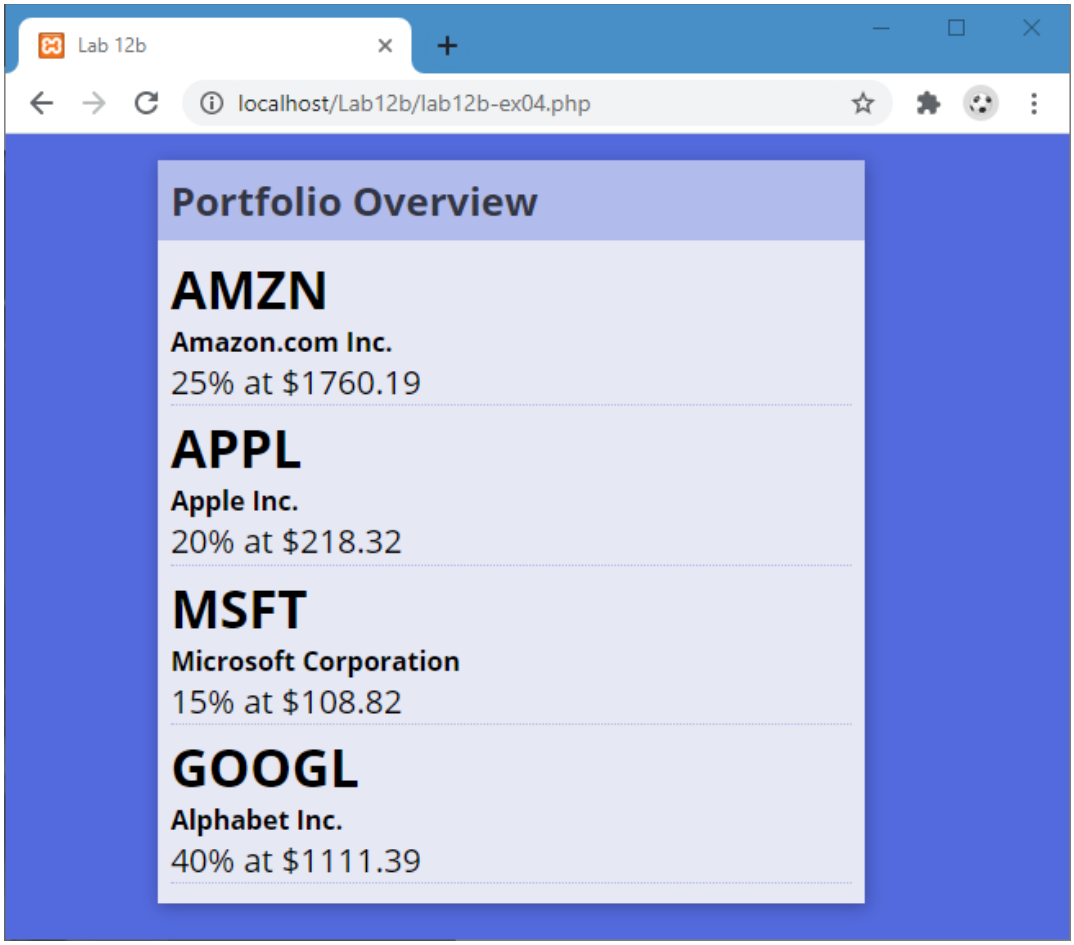

Figure 12.2 – Finished Exercise 12.4

Exercise 12.5 — Defining a Class

1 Examine `lab12b-ex05.php` in a text editor. Add the following class definition to the top of the page.

```
<?php
class Stock {
   public $symbol;
   public $name;
   public $percent;
   public $value;

   function __construct($s, $n, $p, $v) {
      $this->symbol = $s;
      $this->name = $n;
      $this->percent = $p;
      $this->value = $v;
   }
}
?>
```

2 Now that the class is defined, you can instantiate objects from it. Add the following after your class definition.

```
$foo = new Stock("AMZN","Amazon.com Inc.",25,1760.19);
```

3 You can now reference object properties. Add the following and test.

```
<div class="data">
   <?php
      echo "<h2>$foo->symbol</h2>";
      echo "<h3>$foo->name</h3>";
      echo "<p>$foo->percent% at $" . $foo->value . "</p>";
   ?>
</div>
```

Exercise 12.6 — Array of Objects

1 Continue with `lab12b-ex05.php`. Add the following after the class definition.

```
$portfolio = [
   new Stock("AMZN","Amazon.com Inc.",25,1760.19),
   new Stock("APPL", "Apple Inc.",20,218.32),
   new Stock("MSFT","Microsoft Corporation", 15, 108.82),
   new Stock("GOOGL","Alphabet Inc.",40,1111.39)
];
```

2 Comment out the code from Step 3 of Exercise 12.5.

3 In its place, add the following and test.

```
foreach ($portfolio as $p) {
   echo "<h2>$p->symbol</h2>";
   echo "<h3>$p->name</h3>";
   echo "<p>$p->percent% at $" . $p->value . "</p>";
}
```

Compare this code to the equivalent array code in Step 5 of Exercise 4.

5 Modify the `Stock` class by adding the following function.

```
class Stock {
   ...
   public function __toString() {
      $s = "<h2>$$this->symbol</h2>";
      $s .= "<h3>$this->name</h3>";
      $s .= "<p>$this->percent% at $" . $this->value . "</p>";
      return $s;
   }
}
```

The __toString() function provides a string representation of an object.

6 Comment out the code from Step 3 of Exercise 12.6. Replace it with the following and test.

```
foreach ($portfolio as $p) {
   echo $p;
}
```

Test Your Knowledge #1

1. Examine `lab12b-test01.php` and view in browser. You are going to replace the hard-coded markup using the provided array with a function and loops.
2. Examine the file `includes/lab12b-test01.inc.php`. This file contains the populated array that contains all the relevant data needed to generate the page shown in Figure 12.3.
3. Replace the `<article>` markup with a loop that iterates through the `$weatherData` array and outputs an individual city box for each element. To make this task more manageable, you should create some type of function for outputting a single city box and a function for outputting a single day forecast (e.g., Mon/Cloudy/44). Your function for outputting a single city box will need a loop also to loop through the 5 day forecast array.

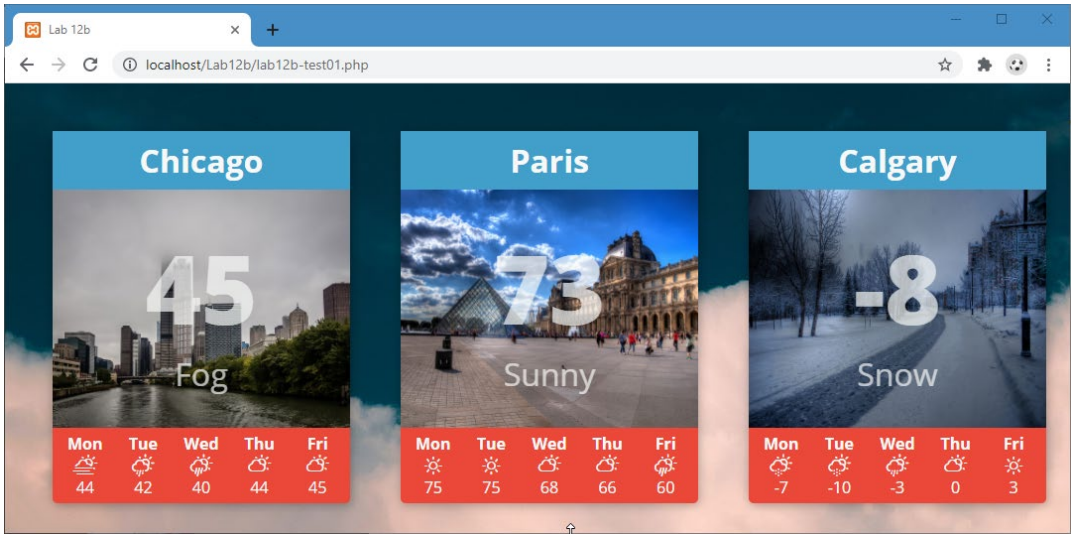

Figure 12.3 – Finished Test Your Knowledge #1

Superglobals

Exercise 12.7 — Working with Post Data

1. Test `lab12b-ex07.php` in a browser and notice that it contains a simple form with one input field that posts the results back to itself (that is, the `action` attribute of the form is the same as the file name).
2. At the top of the page, add the following code that checks if the page contained any post data in the request header.

```
function displayPostStatus() {
   if ($_SERVER["REQUEST_METHOD"] == "POST") {
      echo "Post detected";
   }
   else{
      echo "No Post Detected";
   }
}
```

3 Add a call to this function in the messages `<div>` element:

```
<div id="messages">
    <?php displayPostStatus(); ?>
</div>
```

4 Test in browser.

You should now see message about whether a post was detected or not.

5 Edit the function by adding the following and test:

```
function displayPostStatus() {
   if ($_SERVER["REQUEST_METHOD"] == "POST") {
      echo "<pre>";
      print_r($_POST);
      echo "</pre>";
   }
   else{
      echo "No Post Detected";
   }
}
```

This demonstrates that $_POST is an associative array, with the querystring name as the key field.

6 Edit the function as follows (some code omitted) and test:

```
function displayPostStatus() {
   if ($_SERVER["REQUEST_METHOD"] == "POST") {
      echo "Your content was: " . $_POST["something"];
   }
```

7 Whenever you access querystring data, you need to cover the possibility that it is missing. Change the condition to the following and test.

```
   if (isset($_POST['something']) && $_POST['something']!="") {
      echo "Your content was: " . $_POST["something"];
   }
   else{
      echo "No Post Detected";
   }
```

Query strings are commonly used as a mechanism for passing data from page to page. In the previous example, the form data was POST-ed as a query string within the HTTP header. It is also common to pass data from page to page via hyperlinks, as the next exercise will demonstrate.

Exercise 12.8 — Working with Get Data

1 Examine `lab12b-ex08.php` in your editor. Test in browser. Right now the book links on the left don't really do anything other than re-request the same page.

2 Notice that it includes the file `book-data.inc.php` at the top of the page. Examine the associative data array in this file.

3 Modify the loop outputting the links as follows then test. It doesn't yet change the behavior of the page, but do examine the new URL after you click one of the links.

```
<div class="card-content">
    <ul>
    <?php
      foreach ($books as $key => $value) {
          echo '<li>';
          echo '<a href="lab12b-ex08.php?isbn=' . $key . '">';
          echo $value['title'];
          echo '</a>';
          echo '</li>';
      }
    ?>
    </ul>
</div>
```

This code adds a query string to the link containing the book's ISBN. We can now change our page to use that ISBN value to display the appropriate book.

4 Add the following code to the top of `lab12b-ex08.php`.

```
<?php
include 'includes/book-data.inc.php';

// has the user selected a book to display?
if (isset($_GET['isbn'])) {
    $isbn = $_GET['isbn'];

    // ensure we have this isbn in our data
    if (! array_key_exists($isbn, $books)) {
        $isbn = $defaultISBN;
    }
}
else {
    // if non selected, display first in list
    $isbn = $defaultISBN;
}
?>
```

What is this code doing? It checks to see if the specific query string exists. If it does, then it checks it exists in our array. If it doesn't it uses the default ISBN value; otherwise it uses the passed value.

5 Test in browser. The result should look similar to that shown in Figure 12.4. Try modifying the query string so that the ISBN doesn't exist in your data array. It such a case, it will display the default ISBN. But what if we wanted different behavior?

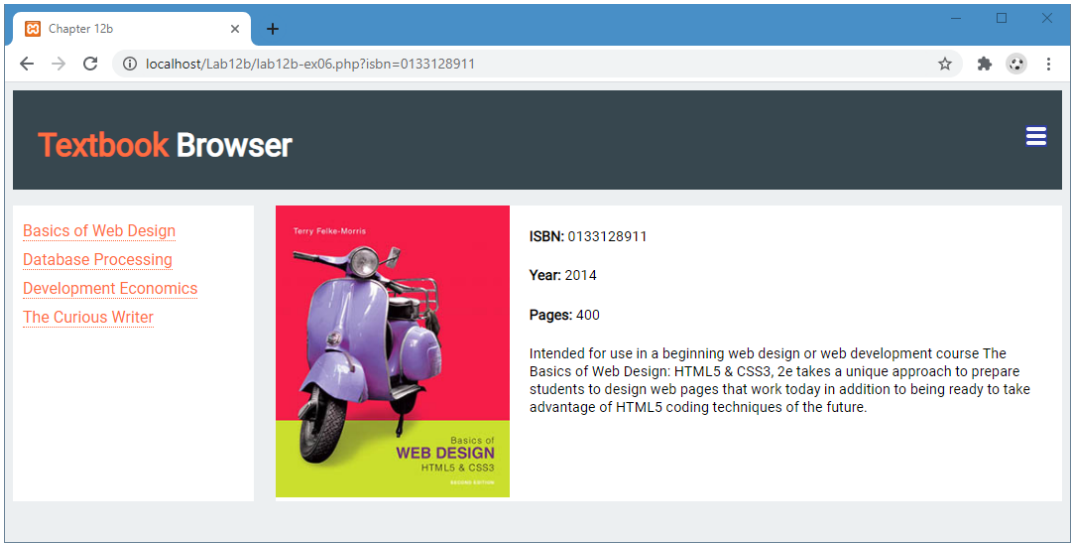

Figure 12.4 – Finished Exercise 12.6

Exercise 12.9 — Redirecting

1 In the previous exercise, if the ISBN in the query string didn't exist, then the page would display the default ISBN. An alternate approach would be to redirect to another page.

Continue working with the same `lab12b-ex08.php` page. Modify the code at the top of the page as follows:

```
// has the user selected a book to display?
if (isset($_GET['isbn'])) {
    $isbn = $_GET['isbn'];

    // ensure we have this isbn in our data
    if (! array_key_exists($isbn, $books)) {
        header("Location: lab12b-error.php");
    }
}
else {
    // if non selected, display first in list
    $isbn = $defaultISBN;
}
```

*The `header()` function is used in PHP for modifying the HTTP response header. The `Location` header tells the browser to request instead the specified URL (that is, it redirects to that other page). The `header()` function **must** be used before any markup is generated.*

2 Test by clicking on one of the book title links, and then modify the ISBN value in the query string and press Enter. It should redirect to error page.

The `header()` function has many uses. Another common use of it is to change the content type. By default, PHP sends HTML back to the browser. It can however, send other content types, for instance JSON or image data.

Exercise 12.10 — Returning JSON Data

1 Examine `lab12b-ex10.php` in your editor. Notice that it contains no HTML at all. This exercise is going to return JSON instead.

2 Add the following code:

```
<?php
include 'includes/simple-data.inc.php';

header('Content-Type: application/json');
echo json_encode($books,JSON_NUMERIC_CHECK+JSON_PRETTY_PRINT);
?>
```

The Content-Type http header is used to tell the browser what kind of content it is about to receive. The `json_encode()` function converts a PHP associative array into JSON.

3 Test in browser.

It should display JSON version of the PHP data array. You have implemented your first web api!

4 Query strings and redirection can be used with a web API as well. Add the following code and test.

```
include 'includes/simple-data.inc.php';

// see if query string exists ... if so return only data for single
book
$data = $books;
if (isset($_GET['isbn'])) {
    // does that ISBN exist in our data?
    $found = null;
    foreach ($books as $b) {
        if ($b['isbn'] == $_GET['isbn']) {
            $found = $b;
        }
    }
    if (isset($found)) {
        $data = $found;
    }
    else {
        header("Location: lab12-error.php");
    }
}
header('Content-Type: application/json');
echo json_encode($data, JSON_NUMERIC_CHECK+JSON_PRETTY_PRINT );
```

5 Test in browser by adding a query string, e.g. `lab12b-ex10.php?isbn=205235778`

Exercise 12.11 — Returning Image Data

1 The previous exercise demonstrated that PHP can return more than just HTML. In this example, your PHP will create and return a customized image. Open `lab12b-ex11.php` and add the following code:

```
<?php
// we need to tell browser that this is returning an image
header('Content-Type: image/jpeg');
// create a 500x500 image
$img = imagecreatetruecolor(500,500);
// fill this image with a RGB defined color
$color = imagecolorallocate($img,238,71,49);
// fill the entirety of this image with the color
imagefilledrectangle($img, 0, 0, 499, 499, $color);
// output the image to the response stream (back to the browser)
imagejpeg($img);
?>
```

2 Test in browser.

This should display a big red square.

3 Modify the code by adding the following:

```
$fontFile = realpath('font/Lato-Medium.ttf');
$fontSize = 24;
$textColor = imagecolorallocate($img,238,238,238);
imagettftext($img,$fontSize,0,50,230,$textColor,$fontFile,
              "Something Witty");

// output the image to the response stream (back to the browser)
imagejpeg($img);
```

This adds some text to the dynamic image. Notice that a font source file must exist on the web server.

4 Test in browser.

The quality is likely to be pretty poor. The JPG file format is not ideal for images containing large blocks of contiguous color. We should instead in such a case use PNG.

5 Modify the following lines (code in between is omitted) and test.

```
// we need to tell browser that this is returning an image
header('Content-Type: image/png');

...
// output the image to the response stream (back to the browser)
imagepng($img);
```

Exercise 12.12 — Displaying an Image File

1 The previous exercise created an image from scratch. This example customizes an existing image file. Open `lab12b-ex12.php` and add the following code:

```
<?php

// we need to tell browser that this is returning an image
header('Content-Type: image/jpeg');

// create the image from a file
$imgname = "images/art/0.jpg";
$img = imagecreatefromjpeg($imgname);

// and return it
imagejpeg($img);

?>
```

2 Edit `lab12b-ex12-tester.html`, add the following tag, and test.

```
<main class="container">
    <h1>Tester for Dynamic Image</h1>
    <img src="lab12b-ex12.php" alt="test image" />
</main>
```

This should display the image. Notice how the `<img>` tag references your php file rather than a .jpg file.

It might not be clear why this technique would be useful. You may wonder why you wouldn't just reference the `.jpg` file instead of having a PHP script load it.

One common use of this technique is to provide some type of customization to the actual image file (for instance, adding a watermark), as shown in the next exercise.

Another common use for this technique is to perform some type of application logic before supplying the image to the requestor. For instance, the image might not be in a .jpg file on the server, but be raw data contained within a database table. Alternately, our application might need to perform some type of user authentication before serving the file; this technique allows us to do so.

Finally, sites often want to hide filenames of images from users for security or authentication reasons; this approach completely hides the path and filename of the actual image file from the end-user.

Exercise 12.13 — Customizing an Image File

1 You will continue modifying the previous exercise file `lab12b-ex12.php`. Add the following code to it:

```
$img = imagecreatefromjpeg($imgname);

// resize the image to 600 x 600
$newimg = imagescale($img,600,600);

// add some text to it
$fontFile = realpath('font/Lato-Medium.ttf');
$fontSize = 16;
$textColor = imagecolorallocate($newimg,238,238,238);
imagettftext($newimg,$fontSize,0,250,160,$textColor,$fontFile,
                "Anyone else want a drink of this?");
// and return it
imagejpeg($newimg);
```

2 Test in browser.

Here the image filename, the additional text, and its position are all hard-coded. In the next exercise, these will be set via query string parameters.

Test Your Knowledge #2

1 Examine `lab12b-test02-form.php` and view in browser and then the editor. Notice that it contains a `<form>` using `method=get` and the `action=lab12b-test02.php`. You will implement `lab12b-test02.php` using code similar to exercise 10 and 11, except rather than hard-coded filenames, font sizes, and text labels, you will use the form data passed to it. Examining the form elements in `lab12b-test02-form.php` will indicate the values to use with the `$_GET` superglobal array.

Note: don't worry about trying to center the text in the image.

2 Implement as well a `width` query string which, if present, will use the `imagescale()` function to return an image of the specified width (the height will be the same as the width).

You can use this to generate the thumbnails at the top of `lab12b-test02-form.php` by adding a loop within the grid-container `<section>`. Simply loop 10 times: within the loop, generate an `<img>` element with the `src` attribute set to `lab12b-test02.php` that has a `file` querystring parameter set to the loop count (0-9) and a `width` querystring parameter set to 100.

3 When done, the form will look like that shown in Figure 12.5, while the generated image will look similar to that shown in Figure 12.6.

Within the browser, try saving the generated file to your computer and then examine it in your operating system. Unlike a JavaScript version of this functionality, this example actually generates an image file containing the text.

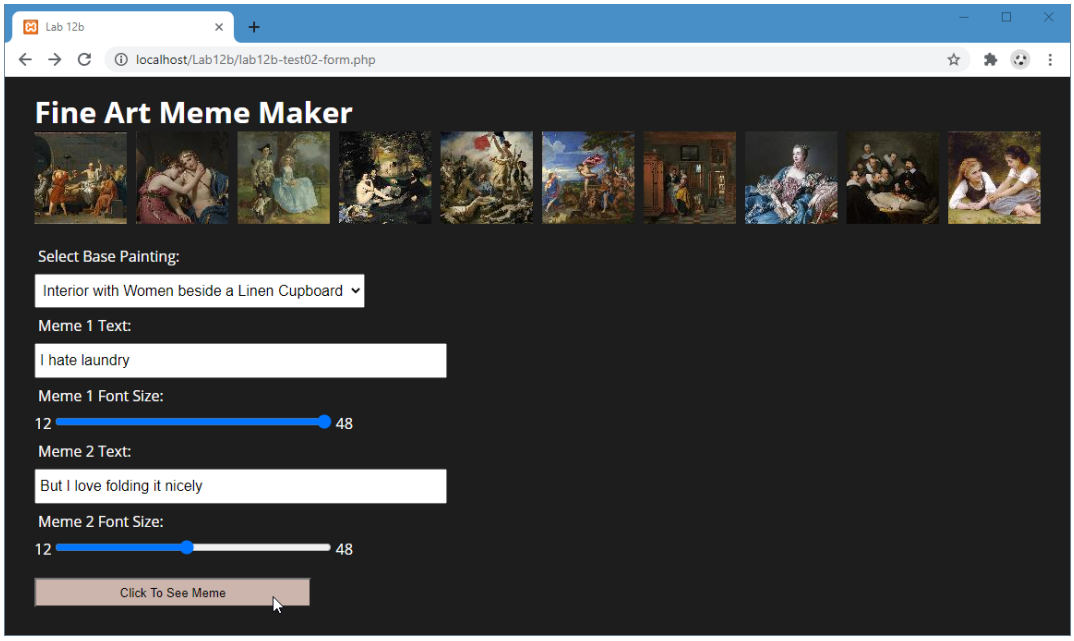

Figure 12.5 – Data Entry Form for Test Your Knowledge #2

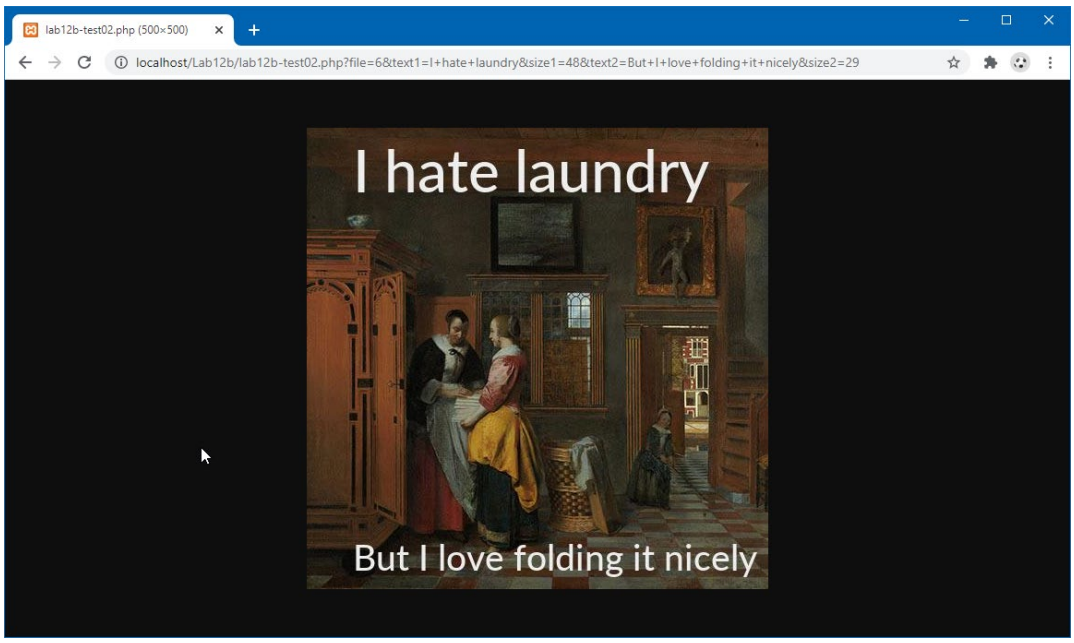

Figure 12.6 – Generated sample image (notice the query string in the URL)