PHP Functions - Part 2 - Dairy Town

Objectives

- Create a function that displays a string.
- Create functions that use the return statement.
- Creating a function that receives an array.
- Returning more than one value from a function.
- Use an external function file.
- Process an associative array

Introduction

This is the second part of chapter 5 on functions. In the first part you learned how to break up HTML code so you can re-use it. In this part you will create a small program that applies what you learned in part one and in addition learn more about code modularity using functions.

Read pages 140 - 150 from chapter 5 for this set of exercises

Watch Chapter 8. User Defined Functions from Lynda.com titled PHP with MySQL Essential Training with Kevin Skoglund. This information will help reinforce the topics from chapter 5.

Note: All students at A-B Tech have Lynda.com accounts. If you haven't used Lynda.com from home, here is a link with instructions for setting up your account and using it from home: A-B Tech students and Lynda.com

Start by working locally before uploading these files to your webhost.

Make sure XAMPP (or similar application) and Apache web server is already running so you can test your work as you continue.

File Structure

Continue working in your **ch05** folder that you created for the previous assignment.

Make a copy of the file named **html-template-function.php** and paste it as a new file named **dairy-town.php**. Note that I've removed a couple of lines (commented below). We will write functions to replace them. Also notice that I've added a new file in the includes section called

diary-functions.php

Here is the code.

```
<?php

include 'functions/html-builder.php';

include 'functions/dairy-functions.php';

makeHeader();

?>

<body>

<div class="container-fluid">

   <!-- removed the original h1 - function coming here →

   <?php makeNavBar(); ?>

   <div class="container">

       <div class="row">

           <div class="col-sm-12">

               <!-- removed the unordered list - function coming here -->

           </div> <!--col-sm-12-->

       </div> <!--row-->

   </div> <!--container-->

<?php makeFooter(); ?>
```

Update index.php

Next update your **ch05/index.php** file so there is a link to DairyTown.php. Do this by adding the line.

```
<li><a href="dairyTown.php">Dairy Town</a></li>
```

to the links section of your file.

Run the **my-code/ch05/index.php** file to see that the link works as expected. It is pretty plain,just a navbar so let’s improve it.

Assignment

Organize your function library

In the previous assignment we built some functions that were specific to creating a web page. In the section we are going to

1. Write some functions that are specific to our ice cream ordering application
2. Modify the existing html-builder.php functions so they are a little more useful.

Create a new function file for our functions library

Create a new php file named **dairy-functions.php** and store it in your **ch05/functions** folder.

Welcome

Create a new function called **`welcome($message, $headingNum)`** in **functions\dairy-functions.php** that accepts one argument and displays “Welcome to Dairy Town.” Surround it with **<h1>** tags. Call the function from the **ch05\dairy-town.php** file. Here is some code to help get you started.

dairy-functions.php

```
/*

*  Name:       welcome($message, $headingNum)

*  Desc:       Displays a welcome message

*  Args:       Accepts two parameters: 1 a string that displays a message
and

*              2 an integer value to place inside the <hx> tag

*  Returns:    None

*/



function welcome($message, $headingNum) {

   $openTag = "<h$headingNum>";

   $closeTag = "</h$headingNum>";

   // the curly braces are optional but make it easier

   // to read what you are inserting a variable inside quotes

   echo "{$openTag} {$message} {$closeTag}";

}
```

Dairy-town.php

Now you need to call the function from the main page for the application. Do so by replacing the

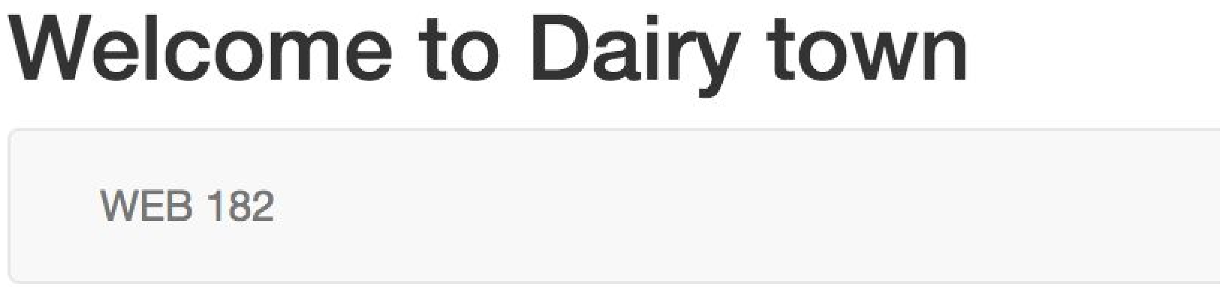
```
<!-- removed the original h1 - function coming here -->
```

With

```
<?php welcome("Welcome to Dairy Town", 2); ?>
```

The opening php tag is needed to surround the code because we are introducing a line of PHP code inside an existing HTML code block. Also notice that there are two arguments in the parameter list.

Go ahead and run the code. It should look like


Welcome to Dairy town

WEB 182

Names

Now we are at a point where we need to create a user input form to gather some information. Create a new file named **dairy-town-input.php**. To start, this file should have an HTML form that accepts a first and last name. Here's the nifty thing, you have already created a library to help you build a web page so you can use the content from a previous page to do so. For example here is one way to do it based on the previous files. Note: feel free to improve upon this (there is actually a lot we can do to streamline this even more). But for now the following will do.

In order to make our program easier to use, let's also update our index.php file so it shows the **dairy-town-input.php** file. Here is what it should look like at this point. Note, I've used tables (I know I shouldn't but it is a quick fix to make the form look okay). I hope to cover more on making the form look decent in a later exercise.

Create a new function named `checkNames($first, $last)` that accepts a first name and a last name. This function should make sure that there is at least one character for the first name and the last name. You can use any of the PHP functions you have learned so far such as `strlen(), empty() or isset()` . You can even use a combination of them to be more precise. The function should work so it returns the value **true** to the function call if both first and last have a character and **false** if they don't. This is a typical way to write a function for validation. Here is the function code to get you started.

```
/*

*  Name:       checkNames($first, $last)

*  Desc:       checks to see if at least one character is entered by the

*              user for the first and last name

*  Args:       two strings, $first and $last for first name and last name

*  Returns:    boolean

*/



function checkNames($first, $last) {

   if (!empty($first) ) {

       return true;

   }   else {

       return false;

   }

}
```

I'll leave it to you to write the function call in the **dairy-town.php** file

Code the dairy-town.php file so it displays “Thank you for your order, firstName lastName” replacing the variables with the person's name.

Toppings

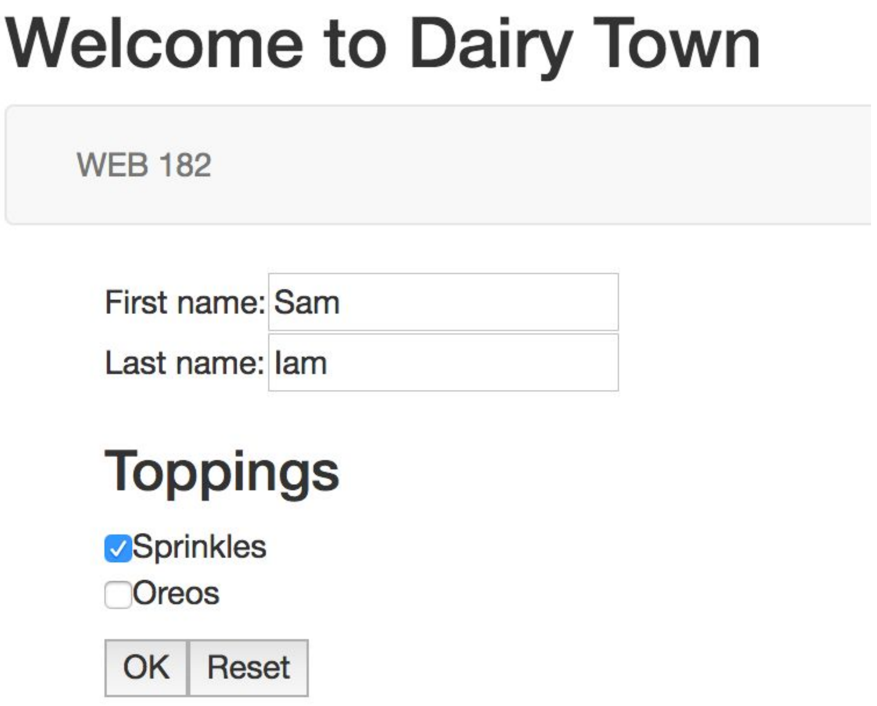
Create a new function named `checkToppings($toppings)` . The function will accept one argument, an array. Like the previous functions it will return either **true** or **false** depending if the user checked any boxes. You can create as many toppings as you like!

First let's update the **dairy-town-input.php** file.

Since an ice cream order can contain more than one topping we will need check boxes in our HTML form. Here is an example of the code to get you started. Notice that there are open and closing square brackets in the checkbox name. This is a method to create an array so you can send more than one item to your PHP file.

```
<input type="checkbox" name="toppings[]" value="sprinkles">Sprinkles<br>
```

Once you have the HTML code running it should look similar to



Welcome to Dairy Town

WEB 182

First name: Sam

Last name: Iam

Toppings

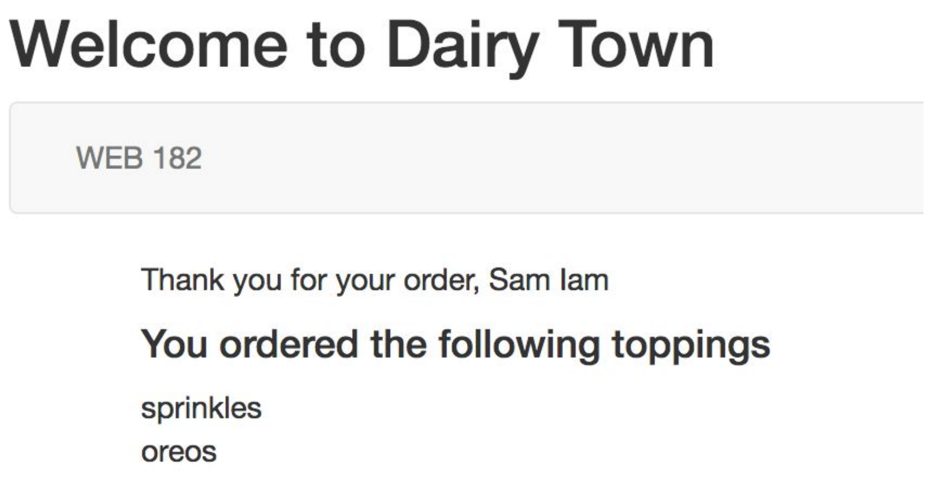
✓ Sprinkles

☐ Oreos

Now we need to write another function to display the toppings. This could be considered overdoing it, but since we are working on writing functions it seems like a good idea.

Write a function called `displayToppings($toppings)` . For a hint here is the code block describing the function. I've made it a bit more generic than the previous functions so you can start to see how generic functions can be used again for different projects.

Code the function so it displays the toppings in an unordered list. Your output should look similar to the following depending on the topics you have selected.



Welcome to Dairy Town

WEB 182

Thank you for your order, Sam Iam

You ordered the following toppings

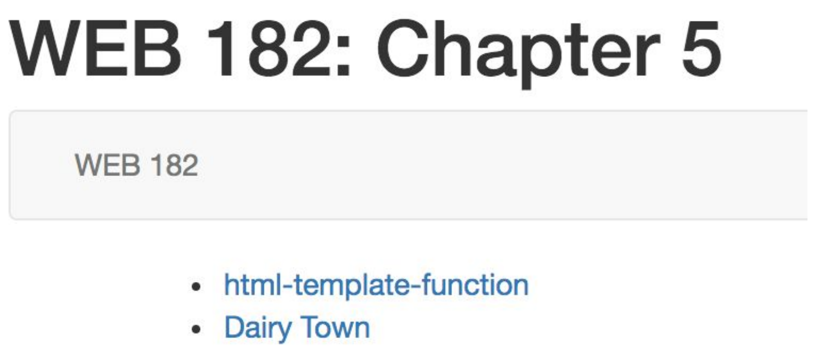
sprinkles
oreos

Tidy up the ch05\index.php file

There are really only two links to files that I would like to see in the **ch05\index.php** file

1. html-template-function
2. Dairy Town

Your **ch05\index.php** file should look like (or similar to)



WEB 182: Chapter 5

WEB 182

- html-template-function
- Dairy Town

To finish up this assignment (by the way -- there is a lot more we can do to clean up our code but that will come with time) we can see an unordered list and a chance to write another function.

We have two options for the function, modify our existing function so it takes an optional argument or write a new function that displays an unordered list. Let's just create a new one for this assignment.

First let's write the function in our **dairy-functions.php** file. Name the function `displayOrderedLinks($array)` . The function will accept an array and display its contents in an unordered list. In our case we will be working with an associative array because we have a hyperlink and the name we want displayed on the webpage. Here is how your associative array could look

```
$links = array(

   "html-template-function.php" => "html-template-function",

   "dairy-town-input.php" => "Dairy Town"

);
```

Also you will need to **include** the **dairy-functions.php file** in your **ch05\index.php** file otherwise you will receive an error message that the function is missing.

Next, here is a snippet of code for your `displayOrderedLinks($array)` function. It shows how to handle an associative array

```
foreach ($array as $key => $value) {

   echo '<li><a href="'. $key . '">' . $value . '</a></li>';

}
```

Remember that the key is the link and the value is the name we want displayed.

After running the program to make sure it works as expected, take a moment to view the source code in the browser. It is just a straight forward HTML bulleted list.

Conclusion

There is a lot you can do with functions and hopefully this primer will get you started and provide you with ideas for future programming projects.

Submitting your work

Zip up all of your chapter 5 work. It will contain all of the files from both of the function phases. Name the zipped file ch05-2-yourLastName.zip.

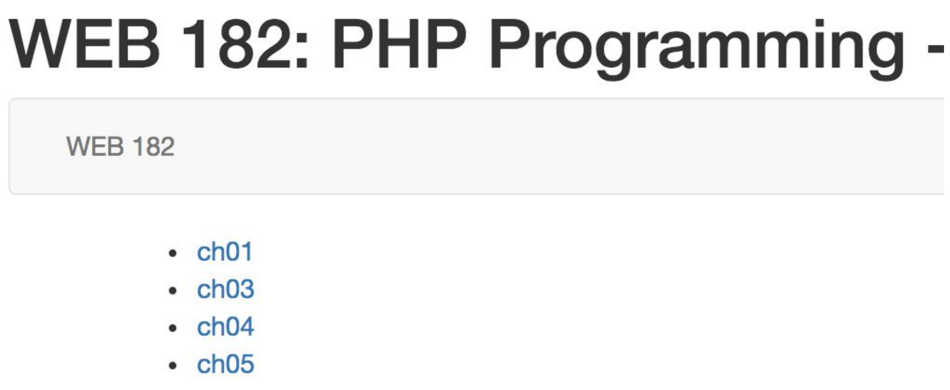
Upload all of your files for chapter 5 to your web host.

Put a URL that links directly to your chapter 5 material.

Here is an image of what your files should look like in your my-code folder. Again, if you have something similar that is easy to navigate, that is fine. Image below.

Charle Wallin

2/26/17



WEB 182: PHP Programming -

WEB 182

- ch01
- ch03
- ch04
- ch05