**Iterating Through Arrays**

Hopefully, you've seen PHP array variables by now in your PHP programming experience. An *array variable* is a single variable that can contain multiple values. The array references each individual value using a *key*, sort of like a database. Each key references a single data value contained in the array. PHP provides two different types of keys you can use to reference values:

* Numeric keys
* Associative keys

A *numeric array variable* is what you're probably used to if you've used arrays in other programming languages. The numeric array variable uses numbers for keys to reference values in the array. The key numbers always start at zero and increase as you store new values in the array.

There are two ways to create a numeric array variable. First, you can create the array using the PHP array() function:

$myarray = array(10, 20, 30, 40, 50);

This code creates a numeric array variable called *$myarray* and assigns five values to the array. You can access the individual array values by using their numeric location in the array (starting at location zero). Just specify the numeric location of the desired array element in square brackets next to the variable name.

$test = $myarray[0] + $myarray[1];

This code adds the values stored in the first and second *$myarray* array elements (which would be the values 10 and 20).

The second way to create a numeric array variable is to assign values to individual array elements, like this:  
  
$myarray = array();  
$myarray[0] = 10;  
$myarray[1] = 20;  
$myarray[2] = 30;  
$myarray[3] = 40;  
$myarray[4] = 50;

Before you can assign the individual elements, you need to tell PHP that the $myarray variable is going to be an array variable. You do this by using the empty array() function.

The array values don't have to be numbers. You can also store string values in an array:  
  
$myarray = array();  
$myarray[0] = "Rich";  
$myarray[1] = "Barbara";  
$myarray[2] = "Katie";  
$myarray[3] = "Jessica";  
echo "The first person in the array is {$myarray[0]}";

Notice the echo statement in the code. You must be careful when using arrays in echo statements. If you want to use an array variable in an echo statement, you must enclose it in braces to separate it from the rest of the text. Otherwise PHP gets confused.

**Associative Array Variables**

*Associative array variables* use strings as keys. This can sometimes get confusing, but it provides a powerful tool for storing information. Similar to the numeric arrays, there are two ways you can create a new associative array variable. The first way uses the array() function:

$myarray = array("state"=>"Illinois", "city"=>"Chicago", "zip"=>"60633");

This statement creates three associative keys for the $myarray array variable and assigns values to each key. The funny looking => symbol assigns the value to the key. In this example, I associate each key with a string value, but you can use numerical values as well.

With the associative array, you must use the string key to recall the stored value:

echo "The state is {$myarray['state']}";

You also create the individual array elements as you would with numeric array variables.  
  
$myarray = array();  
$myarray['state'] = "Illinois";  
$myarray['city'] = "Chicago";  
$myarray['zip'] = "60633";  
echo "The state is {$myarray['state']}";

Just as with the numeric array variables, if you want to use an associative array variable in an echo statement, you must include braces around it.

**Listing Values Stored in the Array**

The tricky part of working with arrays is extracting all of the data they contain. If you use associative array variables, you may have no way of knowing what key values the array contains. Even if you use numeric array variables, there's no guarantee that the key numbers are sequential (you can assign a value to $myarray[6] without assigning a value to $myarray[3], $myarray[4], or $myarray[5]).

Fortunately for us, those thoughtful programmers who developed PHP came up with a simple solution. The *foreach*statement iterates through an array, extracting every key/value pair defined within the array, one at a time.

There are a couple of ways to use the foreach statement with an array. The first method only extracts the value stored in the array, without regard to the key:  
  
$myarray = array("state"=>"Illinois", "city"=>"Chicago", "zip"=>"60633");  
foreach ($myarray as $value)  
{  
echo "The value is $value<br>\n";  
}

In this code snippet, the foreach statement loops through the $myarray array variable. Each value stored in the array is extracted as the variable $value and can be used anywhere within the foreach loop. Notice that the $value variable is a normal variable and can be used without requiring the braces.

The second way to extract array values pulls the entire key/value pair.  
  
$myarray = array("state"=>"Illinois", "city"=>"Chicago", "zip"=>"60633");  
foreach($myarray as $key=>$value)  
{  
echo "The $key value is $value<br>\n";  
}

Now you have access to both the key and the data value within the array. Again, the $key and $value variables are normal variables you can use anywhere within the foreach loop to represent the key and data value. We'll be using this feature in our shopping cart to extract items our customers purchase.

In the next chapter, we'll take a look at a PHP feature that will help reduce the amount of code typing we need to do in our applications!