

INTRODUCTION TO PHP (CONTINUED)

Chapter 9 (16.pdf)

Note

- Examples for this chapter are at

<https://swe.umbc.edu/~zzaidi1/is448/chap9-examples/>

- PHP programs, like CGI programs cannot be seen in the browser by doing a 'View Source' in the browser. All PHP examples used in this chapter are zipped up as **php2.zip** in the above examples folder

11.7 Arrays

- Arrays in PHP combine the characteristics of regular arrays and hashes
 - ▣ An array can have elements **indexed numerically**. These are maintained in order
 - ▣ An array, even the same array, can have elements **indexed by a string**. These are not maintained in any particular order
- The elements of an array are, conceptually, key/value pairs

11.7 Array Creation

- Two ways of creating an array
 - ▣ By assigning a value to an element of an array
 - ▣ By using the **array** function

- Create a numerically indexed array

```
$mixed = array(23, 'xiv', "bob", 777);
```

- Create an array with string indexes

```
$fruits = array("a" => "apple", "b" =>  
"banana", "k" => "kiwi", "r" => "orange");
```

Numerically Indexed Arrays

- The following statement, creates an array as shown below

```
$mixed = array(23, 'xiv', "bob", 777);
```

- Array variable name is **\$mixed**

Index:	0	1	2	3
Array contents:	23	xiv	bob	777

String Indexed Arrays

- The following statement, creates an array as shown below

```
$fruits = array("a" => "apple", "b" => "banana",  
               "k" => "kiwi", "r"=>"orange")
```

- Array variable name is **\$fruits**

Index:	a	b	k	r
Array contents:	apple	banana	kiwi	orange

- In string indexed arrays, the *index* is also called **key**, and the *array content* is also called **value**

11.7 Accessing Array Elements

- Array elements are accessed by using an index subscript in square brackets

```
//accessing numeric-indexed array elements
$element = $mixed[3];
echo $element;
//accessing string-indexed array elements
$fruitname = $fruits['r'];
```

- An array's content can be assigned to a list of variables, by using the **list** function

```
$trees = array("oak","pine","binary");
list($hardwood, $softwood, $data_structure) = $trees;
```

11.7 Functions for Dealing with Arrays

- The **unset** function can be used to delete an array or an element of an array
- The **array_keys** function returns an array of the keys of an array
- The **array_values** returns an array of values in an array

```
$numbers = array(2, 4, 6, 8);  
unset($numbers[2]);  
$highs = array("Mon" => 74, "Tue" => 70);  
$days = array_keys($highs);  
$temps = array_values($highs);
```


11.7 Functions for Dealing with Arrays

- The **array_key_exists** function returns true if a given key is actually present in a given array
- **is_array** determines if its argument is an array
- **sizeof** takes an array as argument and returns length of array

```
$highs = array("Mon" => 74, "Tue" => 70);  
if(array_key_exists("Mon",$highs)){  
    $highs["Mon"] = 90;  
}  
$len = sizeof($highs);  
print("Length of array is $len");
```

11.7 Functions for Dealing with Arrays

- **implode** converts an **array** of strings to a **single string**, separating the parts with a specified string
- **explode** converts a **string** in to an **array of strings** by separating the string at specified characters

```
$related = array("Chandler","and","Monica");  
$str2 = implode(" ", $related);
```

```
$str = "Chandler and Joey are Friends";  
$words = explode(" ", $str);
```

1 1.7 Iterating Through an Array

- Two ways of using foreach to iterate through array
- First way of using foreach

```
foreach (array-name as scalar_variable)
    loop-body
```
- Assigns each value in the array to the *scalar_variable*
- In this example, each element of *\$numbers* array is assigned to scalar variable *\$temp*

```
$numbers = array(1,2,45,32);
foreach ($numbers as $temp)
    print("$temp <br />");
```

- Example: See `access.php`

11.7 Iterating Through an Array

- Second way of using **foreach**

```
foreach (array-name as key-in-array => value-in-array)
    loop-body
```

- The second version assigns each key to **key-in-array** variable and the associated value to **value-in-array** variable
- In this example, each day is stored in **\$day** and temperature is stored in **\$temp** and printed

```
$lows = array("Mon" => 23, "Tue" => 18, "Wed" => 27);
foreach ($lows as $day => $temp)
    print("Low temperature on $day was $temp <br />");
```

Lab

- In the file from the previous lab, create an array `$bgcolors` to store all the colors that you want as the background of your page for the different days of the week
 - ▣ Use `changebackground.php` from `changebackground.zip` if you don't have the solution to the previous lab
- Then, change the background of your page, by accessing the corresponding element from the `$bgcolors` array

Solution: `changebackground3.php`

1 1.7 Sorting Arrays

- The **sort** function sorts the values in an array and makes a numerically subscripted array from the sorted list
- The function **asort** sorts the values in an array but keeps the original key/value association
- The function **ksort** is similar to asort but sorts by keys
- The functions **rsort**, **arsort** and **krsort** are similar but sort in reverse order
- See the example **sorting.php**
 - ▣ illustrates the various sort functions

1 1.8 Functions

- Function definition syntax

```
function name(parameters) {  
    ... //statements  
}
```

- The parameters are optional, but not the parentheses
- Function names are not case sensitive
- A **return** statement causes the function to immediately terminate and return a value, if any value is provided in the return
- A function that reaches the end of the body without executing a return, returns no value

1 1.8 Formal and Actual parameters

- Formal parameters:
 - ▣ Name given to parameters that are in function definition
- Actual parameters:
 - ▣ Name given to parameters that are in the function call
- See functions.php

1 1.8 Formal and Actual Parameters

- A formal parameter, specified in a function declaration, is simply a variable name (i.e., no need to specify the data type of the variable)
- If more actual parameters are supplied in a function-call than there are formal parameters, the extra values are ignored
- If more formal parameters are specified than there are actual parameters in a call then the extra formal parameters receive no value
- PHP defaults to pass by value
 - ▣ Putting an ampersand in front of a formal parameter specifies that pass-by-reference
 - ▣ An ampersand can also be appended to the actual parameter (which must be a variable name)
- See [functions.php](#)

1 1.8 The Scope of Variables

- A variable defined in a function is, by default, local to the function. Called a *local variable*.
- A variable defined outside the function is called a *global variable*
- A global variable with the same name as a local variable is not visible in the function
- See [scope.php](#)

Two keywords that matter to scope of variables

- global
- static

Keyword **global**

- Declaring a variable in a function with the **global** declaration means that the functions uses the global variable of that name
- See **global_scope.php**

```
<?php
$a = 1;
$b = 2;
function Sum()
{
    global $a; global $b;
    $b = $a + $b;
}
Sum();
echo $b;
?>
```

Keyword **static**

- The usual lifetime of a local variable is from the time the function begins to execute to the time the function returns
- Declaring a variable with the **static** keyword means that the lifetime is from the first use of the variable to the end of the execution of the entire PHP script
- In other words, a static variable exists only in a local function scope, but it does not lose its value when program execution leaves this scope.
- In this way a function can retain some 'history'

```
<?php
/*every time function is called, it will
print the current value of $a and
increment it*/
function test()
{
    //$a is initialized only first time this
function is called
    static $a = 0;
    echo $a;
    $a++;
}
?>
```

Example: See [static_variables.php](#)

11.10 Form Handling

- Users expect an HTML response page when they submit forms
- Embedded PHP allows you run some server-side code and also send back an HTML response page
- The user-entered values from forms can be accessed in PHP using the `$_POST` and `$_GET` arrays
- Example:
 - See `form_proc.html`, `form_proc.css`, `display_info.php`
 - See `ret_images.html`, `ret_image.php`
 - See `multi_checkbox.html`, `multi_checkbox.php`

Checklist for proper form handling implementation

- In your HTML page
 - ▣ Do your form elements have a *name* attribute and a value set for the *name* attribute?
 - ▣ Have you set the *action* attribute of the form tag to 'POST' or "GET"? Use one or the other. POST is more secure than GET.
 - ▣ Set your form's *action* attribute to point to the PHP program that is supposed to process the form data
- In your PHP program,
 - ▣ The data entered in the form is stored in the `$_POST` or the `$_GET` array (depends on which *method* you used in the HTML form)
 - ▣ You can access the elements of the array just like you access an element from a PHP array. Name of the array is `$_POST` or `$_GET` and the index is the **name of the form element**.

A bad way to produce HTML in PHP

- printing HTML code with print statements is ugly and error-prone:
 - ▣ must quote the HTML and escape special characters, e.g. \"
- Don't print HTML; it's bad style!
- best PHP style is to use as few print/echo statements as possible in embedded PHP code

```
<?php
print "<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.1//EN\"";
print " \http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd\>";
print "<html xmlns=\"http://www.w3.org/1999/xhtml\">";
print " <head>\n"; print " <title>My web page</title>"; ...
?>
```


A good way to produce HTML in PHP

- any contents of a .php file that are not between `<?php` and `?>` are output as pure HTML
- can switch back and forth between HTML and PHP "modes"

```
html content
<?php
    PHP code
?>
html content <?php php code ?> html content
```

Lab

- Download the HTML file: ratings.html
- Modify this file by setting the form's **action** attribute to point to a new PHP file
- Create a new PHP file, **ratings.php**, that will take the user entered movie name and rating and print these two values to the screen
- Solution: ratings_sol.html, ratings_sol.php