

Introduction to PHP (Hypertext PreProcessor, HyperText Processor)

Part 2

PHP Arrays and Superglobals

References

- Chapter 8 Introduction to Server-Side Development with PHP of the Text Book Entitled Fundamentals of Web Development, by Randy Connolly and Ricardo Hoar
- PHP 5.6.26, Sept. 16, 2016; <http://php.net/>
- PHP 5.6.14 Released Oct. 1, 2015
- PHP 5.6.1 Released Oct. 2, 2014, <http://php.net/>
- PPH Documentation, <http://php.net/urlhowto.php>
- PHP Language Reference, <http://php.net/manual/en/langref.php>

PHP Data Types

- Boolean // true, false
- Integer // whole numbers
- Float // Decimal numbers
- String // Letters, characters
- Array // A collection of data of various data types
- Object // Instances of classes

Functions

- PHP Built-in
 - echo() //Output to HTML
 - define() // Define constants
 - printf() // Formatted output
- User Defined
 - Syntax: function, return
 - Calling a function
 - Parameters
 - Passing by values
 - Passing by reference
 - Variable scopes

PHP Arrays, <http://php.net/manual/en/language.types.array.php>

An [array](#) in PHP is actually an **ordered map**. A map is a type that associates *values* to *keys*. This type is optimized for several different uses; it can be **treated as an array, list (vector), hash table (an implementation of a map), dictionary, collection, stack, queue, and probably more**. As [array](#) values can be other [arrays](#), trees and multidimensional [arrays](#) are also possible.

Array Applications

- Using an array as a List
 - A group of images
- Using an array as a Sortable Table
 - Data Table (row, column): Catalog of Pet (pet_name, owner_name, weight, animal, etc)
 - Database Tables (row, column)
- Using an array as a Lookup Table
 - Cryptogram generator
- Web database applications

Array, <http://php.net/manual/en/function.array.php>

Definition of Array

- A data structure that allows the programmer to collect a number of related elements together under a single variable.

PHP Arrays

- Called “Associated Arrays”
- An ordered map which associates each **value** in the array with a **key**.
- PHP arrays are like vector, hash table, dictionary, and list collection.
- Can be used like collection classes in other languages

Arrays Key (index)

- Must be either integers or strings, and need not be in sequential
- The default array index starts at 0, 1, 2,... n

Array Values

- Not restricted to integers and strings
- They can be any object, type, or primitive data supported in PHP

Array Manipulations and Functions (Array Functions, <http://php.net/manual/en/ref.array.php>)

- Declaration of an array (just name of array)
- Define an array (with values)
- Accessing and editing the array (read values)
- Adding and Deleting Elements
- Swapping keys and values
- Merging array elements
- Sorting
- Determining whether keys and values exist
- Searching the array
- Other Array Functions
 - Returning an indexed array
 - Reverse ordering
 - Search array for a value

Defining and Accessing an Array

```
$days = array();           // Declare an array
$days = array("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"); // Define the contents
$days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]; // Alternate syntax
```

Or

```
$day = array();
$days[0] = "Mon",
$days[1] = "Tue",
$days[2] = "Wed",
```

Or assigning keys explicitly to array elements

```
$days = array (1 => "Mon", 2 => "Tue", 3 => "Wed", 4 => "Thu", 5 => "Fri", 6 => "Sat", 7 => "Sun");
# keys are 1, 2, ..., 7
# values are "Mon", "Tue", ...
```

Array with strings as keys and integer numbers as values

```
$salesForecast = array("Mon" => 100, "Tue" => 200, "Wed" => 40, "Thu" => 100, "Fri" => 200,
"Sat" => 250, "Sun" => 350);
# Keys are "Mon", "Tue", etc
# Values are 100, 200, etc
```

```
echo $salesForecast["Sun"]; // Output 350
```

Multidimensional Arrays

```
$month = array
(
array("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"),
array("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"),
array("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"),
array("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun")
)
```

```
echo $month[0][3]; // Output Thu
echo $month[3][3]; // Output Thu
echo $month[3][6]; // Output Sun
```

Shopping Cart Example

```
$cart = array();  
$cart[] = array("id" => 37, "title" => "Burial at Orans", quantity => 1);  
$cart[] = array("id" => 345, "title" => "The Death of Marat", quantity => 1);  
$cart[] = array("id" => 63, "title" => "Starry Night", quantity => 1);  
  
echo $cart[2]["title"]; // Outputs Starry night
```

Iteration through an Array

- for, while, do while – only good for those arrays with sequential integers as index
- For arrays without sequential index (associative arrays)
 - Need to use count() function to know the number of elements (length) in an array, then use for, while, etc
 - foreach loop

Examples

```
foreach ($salesForecast as $value){  
    echo $value . "<br>";  
}
```

```
foreach ($salesForecast as key => $value){  
    echo "day" . $key . "=" . $value . "<br>";  
}
```

Array Functions, <http://php.net/manual/en/ref.array.php>

- array_keys(\$someArray); // Return all the keys or a subset of the keys of the given array
- array_values(\$someArray); // Return all the values of an array
- array_rand(\$someArray, \$num =1); // Pick or select a random element in an array
- array_reverse(\$someArray); // Return an array with elements in reverse order
- array_walk(\$someArray, \$callback, \$optionalParam); // Apply a user supplied function to every member (element) of an array
- in_array(\$needle, \$haystack); // Check if a value exist in an array; \$needle – value, \$haystack - array
- shuffle(\$someArray); // Shuffle an array

Predefined Superglobals, <http://php.net/manual/en/language.variables.superglobals.php>

Superglobals // Bulit-in variables that are always available in all scopes

- `$GLOBALS` // All variables available in global scope (array containing info about the request and the server)
- `$_SERVER` // Server and execution environment info
- `$_GET` // HTTP GET variables (array of query string data passed to the server via the URL)
- `$_POST` // HTTP POST variables (array of query string data passed to the server via the HTTP header)
- `$_FILES` // HTTP file upload variables (array of file items uploaded to the server)
- `$_REQUEST` // HTTP request variables (array containing the contents of `$_GET`, `$_POST`, and `$_COOKIE`)
- `$_SESSION` // Session variables (array contains session data)
- `$_ENV` // Environment variables (array) of a server
- `$_COOKIE` // Array of Cookie data passed to page via HTTP request

PHP Functions Reference

- `echo()`
- `unset()` // Destroy variables
- `func_get_args()` // get function arguments
- `func_num_args()` // get function argument count

String Manipulation Functions

- `strcmp()` // string comparison, case sensitive
- `strcasecmp()` // string comparison, non case sensitive
- `substr()` // the sub string
- `strlen()` // The number of chars in the string
- `strpos()` // The character position
- `chop()` // Remove all white spaces from its ends
- `trim()` // remove all white spaces from both ends
- `strtolower()` // covert to lower case characters
- `strtoupper()` // convert to upper case characters

Arithmetic Functions

- `floor()`, `ceil()`, `round()`, `srand()`, `rand()`, `abs()`, `min()`, `max()`

Output Functions

- `print()`
- `printf()` // formatted output

PHP Functions

- `func_get_args()` // <http://us3.php.net/manual/en/function.func-get-args.php>
- `func_num_args()`

<!DOCTYPE html>

<!--funcnt_arguments.php - A trivial example to illustrate a php document -->

```
<html lang = "en">
  <head>
    <title> funct_arguments.php </title>
  <meta charset = "utf-8" />
</head>
<body>
<?php
  function func_dynamic() {
    echo "ITC 250/CPET 499 Web Systems: ".func_num_args(). "Number of arguments.<br>";
    $args = func_get_args();
    for($i = 0; $i < count($args); $i++)
    {
      echo "Passed arguments: {args[$i]} <br>";
    }
  }
  func_dynamic(5, 4, 3, 2, 1 );
?>
</body>
</html>
```