Arrays, Functions, Strings in PHP

Mahesh Kumar
Assistant Professor (Adhoc)

Department of Computer Science
Acharya Narendra Dev College
University of Delhi

Course webpage
[http://www.mkbhandari.com/mkwiki]

Outline

1. Arrays
2. Strings
3. Functions

Arrays

- An array is a special variable, which can hold more than one value at a time. [it stores list of items in a single variable]
- For example: if you want to store 100 numbers then instead of defining 100 variables its easy to define an array of 100 length.
- An array can hold many values under a single name, and you can access the values by referring to an index number.
- How to Create an Array?
  - the **array( )** function is used to create an array.
  - $colleges = array("ANDC","ARSD","Aryabhatta");
  - echo $colleges[0]; // will print ANDC
- To get the Length of an Array – Use the count( )/sizeof( ) function
  - echo count($colleges); // will print 3

Types of Arrays

1. Indexed arrays /Numeric arrays
   - An array with a numeric index. Values are stored and accessed in linear fashion.
   - These arrays can store numbers, strings and any object but their index will be represented by numbers. By default array index starts from zero.
   - There are two ways to create indexed arrays:

```
// First method to create array
$numbers = array( 1, 2, 3, 4, 5);
foreach( $numbers as $value ) {
    echo "Value is $value <br />";
}
```

```
// Second method to create array
$numbers[0]  = 10;
$numbers[0]  = 20;
$numbers[0]  = 30;
$numbers[0]  = 40;
$numbers[0]  = 50;
foreach( $numbers as $value ) {
    echo "Value is $value <br />";
}
```

Types of Arrays

2 Associative Arrays

- Associative array will have their index as string(named keys) so that you can establish a strong association between key and values.
- There are two ways to create indexed arrays:

```
// First method to create array
$rollNumbers = array("Raj" => 101, "Simran" => 102, "Kiran" => 103);
echo "Roll Number of Raj is ". $rollNumbers['Raj'] . "<br />";
echo "Roll Number of Simran is ". $rollNumbers['Simran'] . "<br />";
echo "Roll Number of Kiran is ". $rollNumbers['Kiran'] . "<br />";
}
```

```
// Second method to create array
$rollNums['Raj']  = 101;
$rollNums['Simran']  = 102;
$rollNums['Kiran']  = 103;
foreach($rollNums as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
```

Types of Arrays

3 Multidimensional Arrays

- A multidimensional array is an array containing one or more arrays.
- PHP supports multidimensional arrays that are two, three, four, five, or more levels deep.
- However, arrays more than three levels deep are hard to manage for most people.
- The dimension of an array indicates the number of indices you need to select an element.
- A two-dimensional array is an array of arrays (a three-dimensional array is an array of arrays of arrays).

Name	Stock	Sold
C++	20	30
Java	30	20
PHP	15	35
Python	25	25

Types of Arrays

3 Multidimensional Arrays

```
// To create a 2-D array:
$books = array
(
        array(“C++”, 20, 30);
        array(“Java”, 30, 20);
        array(“PHP”, 15, 35);
        array(“Python”, 25, 25);
);
```

- To get access to the elements of the $books array, point to the two indices (row and column):

```
echo $books[0][0] . ": In stock: " . $books[0][1] ., sold: " . $books[0][2] ." .<br>";
echo $books[1][0].": In stock: ".$books[1][1].", sold: ".$books[1][2].".<br>";
echo $books[2][0].": In stock: ".$books[2][1].", sold: ".$books[2][2].".<br>";
echo $books[3][0].": In stock: ".$books[3][1].", sold: ".$books[3][2].".<br>";
```

Types of Arrays

3 Multidimensional Arrays

- We can also put a for loop inside another for loop to get the elements of the $books array

```
// To  access a 2-D array in PHP using nested for loops
for ($row = 0; $row < 4; $row++) {
        echo "<p><b>Row number $row</b></p>";
        echo "<ul>";
         for ($col = 0; $col < 3; $col++) {
                echo "<li>". $books[$row][$col] ."</li>";
         }
        echo "</ul>";
}
```

Sort Functions for Arrays

- The elements in an array can be sorted in alphabetical or numerical order, descending or ascending.
  - sort($numbers) - sort arrays in ascending order
  - rsort($numbers) - sort arrays in descending order
  - asort($rollNumbers) - sort associative arrays in ascending order, according to the value
  - ksort($rollNumbers) - sort associative arrays in ascending order, according to the key
  - arsort($rollNumbers) - sort associative arrays in descending order, according to the value
  - krsort($rollNumbers) - sort associative arrays in descending order, according to the key

PHP Strings

- A string is a sequence of characters, like "Hello world!" or 'Hello world!'
- strlen() - Returns the Length of a String

```
echo strlen("Hello world!");                    // outputs 12
```

- str_word_count() - Count Words in a String

```
echo str_word_count("Hello world!");           // outputs 2
```

- strrev() - Reverse a String

```
echo strrev("Hello world!");                   // outputs !dlrow olleH
```

- strpos() - Search For a Text Within a String

```
echo strpos("Hello world!", "world");          // outputs 6 , If no match is found, it will return FALSE.
```

- str_replace() - Replace Text Within a String

```
echo str_replace("world", "India", "Hello world!");          // outputs Hello India!
```

- Please see the complete list of string functions in your reference for more string manipulation operations

PHP Functions

- PHP has more than 1000 built-in functions, and in addition you can create your own custom functions.
- Built-in functions can be called directly, from within a script, to perform a specific task.
- User Defined Functions
  - A function is a block of statements that can be used repeatedly in a program.
  - A function will not execute automatically when a page loads.
  - A function will be executed by a call to the function.
- How to create a function?

```
function functionName( ) {
  code to be executed;
}
```

1. A function name must start with a letter or an underscore.
2. Function names are NOT case-sensitive.

PHP Functions

- Example

```
<?php
function writeMsg() {
   echo "Hello world!";
}
writeMsg();       // call the function
?>
```

- Function Arguments
  - Information can be passed to functions through arguments. An argument is just like a variable.
  - Arguments are specified after the function name, inside the parentheses.
  - You can add as many arguments as you want, just separate them with a comma.

```
// function argument in PHP
<?php
function studentName($sname) {
   echo "$sname .<br>";
}

studentName(“Purushottam”);
studentName(“Kapil”);
studentName(“Himani”);
studentName(“Ishaan”);
studentName(“Furquan”);

?>
```

PHP Functions

- Example

```
<?php
function writeMsg() {
    echo "Hello world!";
}
writeMsg();       // call the function
?>
```

- Function Arguments
  - Information can be passed to functions through arguments. An argument is just like a variable.
  - Arguments are specified after the function name, inside the parentheses.
  - You can add as many arguments as you want, just separate them with a comma.

// Single Agrument Function

```
<?php
function studentName($sname) {
        echo "$sname .<br>";
}
studentName(“Purushottam”);
studentName(“Kapil”);
studentName(“Himani”);
studentName(“Ishaan”);
studentName(“Furquan”);
?>
```

// Two Agruments Function

```
<?php
function stuData($sname,$sbranch){
 echo "$sname is from $sbranch . <br>";
}
stuData("Naved","Computer Science");
?>
```

PHP is a Loosely Typed Language

- We do not need to tell PHP the data type of a variable.
- PHP automatically associates a data type to the variable, depending on its value.
- Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.
- In PHP 7, type declarations were added. Which gives us an option to specify the expected data type when declaring a function.
- By adding the strict declaration, it will throw a "Fatal Error" if the data type mismatch.

```
//Without using strict
<?php
function addNumbers(int $a, int $b) {
   return $a + $b;
}
echo addNumbers(5, "5 days");  // it will return 10
?>
```

```
declare(strict_types=1);      // strict requirement
<?php
function addNumbers(int $a, int $b) {
   return $a + $b;
}
echo addNumbers(5, "5 days");      // fatal Error
?>
```

PHP Default Argument Value

```
<?php declare(strict_types=1);          // strict requirement   ?>
<!DOCTYPE html>
<html>
<body>
<?php
function setMarks(int $avgMarks = 60) {
        echo "Marks is : $avgMarks <br>";
}
setMarks(65);
setMarks( );                // will use the default value of 60
setMarks(50);
setMarks(70);
?>
</body>
</html>
```

//Output

Marks is : 65

Marks is : 60

Marks is : 50

Marks is : 70

PHP Functions - Returning values

```
<?php declare(strict_types=1);        // strict requirement   ?>
<!DOCTYPE html>
<html>
<body>
<?php
function sum(int $x, int $y) {
   $z = $x + $y;
   return $z;
}
echo "5 + 10 = " . sum(5,10) . "<br>";
echo "7 + 13 = " . sum(7,13) . "<br>";
echo "2 + 4 = " . sum(2,4);
?>
</body>
</html>
```

//Output

5 + 10 = 15

7 + 13 = 20

2 + 4 = 6

PHP Return Type Declarations

- PHP 7 also supports Type Declarations for the return statement.
- Like with the type declaration for function arguments, by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch.
- To declare a type for the function return, add a colon **(:)** and the type right before the opening curly **({)** bracket when declaring the function.

```php
<?php declare(strict_types=1); // strict requirement

function addNumbers(float $a, float $b) : float {
    return $a + $b;
}
echo addNumbers(1.2, 5.2);
?>
```

```php
<?php declare(strict_types=1); // strict requirement

function addNumbers(float $a, float $b) : int {
    return (int)($a + $b);
}
echo addNumbers(1.2, 5.2);
?>
```

- You can specify a different return type, than the argument types, but make sure the return is the correct type

Program No. 5

```
<html>
   <head>
      <title>Lab Exercise No. 5</title>
   </head>
   <body>
      <?php
         $color = array("white", "red", "green");

         foreach($color as $clr )
               echo  "$clr, ";

         sort($color);

         echo "<ul>";
         foreach($color as $clr )
            echo "<li> $clr </li>";
         echo "</ul>";
      ?>
</body>
</html>
```

References

Reference for this topic

- [Introduction to PHP]
  https://www.w3schools.com/php/default.asp

- [Install and Run PHP]
  https://www.techomoro.com/how-to-run-a-php-application-on-ubuntu-18-04-2-lts/

- [GeeksforGeeks]
  https://www.geeksforgeeks.org/php/