Basic PHP

Objective

In this chapter learner able to understand:

- PHP Array
- PHP Strings
- String functions
- String “explode” and “implode”

PHP Array

Objective

In this chapter learner able to understand:

- PHP Arrays
- Multidimensional Arrays
- Array Operators
- Array Sorting
- Array Functions

Array

An array in PHP is actually an ordered map. A map is a type that associates values to keys.

Type of Array

There are three kind of arrays

- Numeric array - An array with a numeric index .
- Associative array - An array where each ID key is associated with a value .
- Multidimensional array - An array containing one or more arrays .

Numeric Arrays

A numeric array stores each array element with a numeric index.

There are two methods to create a numeric array.

1. In the following example the index are automatically assigned.

```
$cars=array("Saab","Volvo","BMW","Toyota");
```

2. In the following example we assign the index manually:

```
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
```

Example of Numeric array

```
<?php
    $cars[0]="Saab";
    $cars[1]="Volvo";
    $cars[2]="BMW";
    $cars[3]="Toyota";
    echo $cars[0] . " and " . $cars[1] . " are Swedish cars.";
?>
```

Associative Arrays

- An associative array, each ID key is associated with a value.
- When storing data about specific named values, a numerical array is not always the best way to do it.
- With associative arrays we can use the values as keys and assign values to them.

Example of Associative array

There are two methods to create a Associative array.

```
$ages = array(“Rahul"=>32, “Amit"=>30, “Anoop"=>34);
```

```
$ages[‘Rahul'] = "32";
$ages[‘Amit'] = "30";
$ages[‘Anoop'] = "34";
```

Creating/modifying with square-bracket

```
<?php
	$arr = array(5 => 1, 12 => 2);
	$arr[] = 56;    // This is the same as $arr[13] = 56;
	$arr["x"] = 42; // This adds a new element to the array with key "x“
	unset($arr[5]); // This removes the element from the array
	unset($arr);    // This deletes the whole array
?>
```

Multidimensional Arrays

In a multidimensional array, each element in the main array can also be an array. And each element in the sub-array can be an array, and so on.

Example of Multidimensional Arrays

```
<?php
        $marks = array( "mohammad" => array ( "physics" => 35, "maths" => 30,
        "chemistry" => 39 ), "qadir" => array ( "physics" => 30, "maths" => 32,
        "chemistry" => 29 ), "zara"  => array ( "physics" => 31, "maths" => 22,
        "chemistry" => 39 ) );
        echo "Marks for mohammad in physics : " ;
        echo $marks['mohammad']['physics'] . "<br />";
        echo "Marks for qadir in maths : ";
        echo $marks['qadir']['maths'] . "<br />";
        echo "Marks for zara in chemistry : " ;
        echo $marks['zara']['chemistry'] . "<br />";
?>
```

foreach multi-dimensional arrays

```
<?php
        $a = array();
        $a[0][0] = "a";
        $a[0][1] = "b";
        $a[1][0] = "y";
        $a[1][1] = "z";
        foreach ($a as $v1)
        {
                foreach ($v1 as $v2)
                {
                        echo "$v2\n";
                }
        }
?>
```

Array Functions

Count all elements in an array, or something in an object.

int count(array , mode)

Parameter	Description
array	Required. Specifies the array or object to count.
mode	If the optional mode parameter is set to COUNT_RECURSIVE (or 1), count() will recursively count the array. This is particularly useful for counting all the elements of a multidimensional array. count() does not detect infinite recursion.

Array Functions

sizeof ($array,mode)- Alias of count().

```
<?php
$arr =array("apple","banana","mango","avocado");
echo sizeof($arr);
?>
```

Arrays Sorting Function

sort($array): Sorts an array in ascending value order .

rsort($array) : Sorts an array in descending value order.

asort($array) : Sorts an array in ascending value order while maintaining the key/value relationship

arsort($array) : Sorts an array in descending value order while maintaining the key/value relationship

Arrays Sorting Function

The **uksort()** function sorts an array by the element keys using user defined comparison function.

This function returns TRUE on success, or FALSE on failure.

uksort(array,sorttype)

Parameter	Description
array	Required. Specifies the array to sort
function	Required. A user specified function. The function must return -1, 0, or 1 for this method to work correctly. It should be written to accept two parameters to compare, and it should work something like this: If a = b, return 0 If a > b, return 1 If a < b, return -1

uksort() example

```
<?php
function cmp_function($a, $b)
{
  if ($a == $b) return 0;
  return ($a > $b) ? -1 : 1;
}
$friends = array("a"=>"Amit", "b"=>"Anoop", "c"=>"Raj" );
uksort($friends, "cmp_function");
print_r($friends);
?>
```

Array Functions

array_combine — Creates an array by using one array for keys and another for its values .

Syntax: **array array_combine (array $keys, array $values)**

Example:

```
<?php
        $a = array('green', 'red', 'yellow');
        $b = array('avocado', 'apple', 'banana');
        $c = array_combine($a, $b);
        print_r($c);
?>
```

Output -:

```
Array (
[green] => avocado
[red] => apple
[yellow] => banana )
```

Array Functions

array_count_values — Counts all the values of an array

Syntax: array array_count_values (array $input)

Example:

```
<?php
      $array = array(1, "hello", 1, "world", "hello");
      print_r(array_count_values($array));
?>
```

Output;

```
Array
(
   [1] => 2
   [hello] => 2
   [world] => 1
)
```

Array Functions

array_fill : Fill an array with values.
array array_fill (int $start_index , int $num , mixed $value)

Example:

```
<?php
    $a = array_fill(3,6,"MY PHP");
    print_r($a);
 ?>
```

Array Functions

array_flip : Exchanges all keys with their associated values in an array.

Syntax: **array array_flip(array)**

```
<?php
        $data = array("a" => "apple", "b" => "ball");
        print_r(array_flip($data));
?>
```

Array Functions

array_rand : Pick one or more random entries out of an array.

Syntax: **key array_rand(array)**

```
<?php
      $data = array("white", "black", "red");
      echo "Today's color is " . $data[array_rand($data)];
?>
```

Array Functions

**array_keys — ** Return all the keys of an array

array array_keys (array $input [, mixed $search_value [, bool $strict]])

array_keys() returns the keys, numeric and string, from the *input* array.

```
<?php
        $array = array(0 => 100, "color" => "red");
        print_r(array_keys($array));

        $array = array("blue", "red", "green", "blue", "blue");
        print_r(array_keys($array, "blue"));

        $array = array("color" => array("blue", "red", "green"),
        "size"  => array("small", "medium", "large"));
        print_r(array_keys($array));
?>
```

Array Functions

array_merge() merges the elements of one or more arrays together so that the value one are appended to the end of the previous one. It returns the resulting array.

array array_merge (array $array1 [, array $array2 [, array $...]])

```
<?php
    $array1 = array("color" => "red", 2, 4);
    $array2 = array("a", "b", "color" => "green", "shape" => "trapezoid", 4);
    $result = array_merge($array1, $array2);
    print_r($result);
?>
```

output

```
Array
(
    [color] => green
    [0] => 2
    [1] => 4
    [2] => a
    [3] => b
    [shape] => trapezoid
    [4] => 4
)
```

Array Functions

array_search — Searches the array for a given value and returns the corresponding key if successful

Syntax : **key array_search(Search_value , array)**

```
<?php
        $array = array(0 => 'blue', 1 => 'red', 2 => 'green', 3 => 'red');
        $key = array_search('green', $array); // $key = 2;
        $key = array_search('red', $array);   // $key = 1;
?>
```

Array Functions

array_shift : Shift an element off the beginning of array

Synatx : Shifted_value array_shift(array);

```
<?php
    $array = array("aa","bb","cc","dd",5,6);
    $a=array_shift($array);
    print_r($a);
    print_r($array);
?>
```

Output:

```
aa
Array
(
  [0] => bb
  [1] => cc
  [2] => dd
  [3] => 5
  [4] => 6 )
```

String in PHP

String

Strings in PHP are a sequence of characters that are always internally null terminated.

```
<?php
     $my_string = “Welcome HPES Students";
     echo " Welcome HPES Students ";
     echo $my_string
?>
```

String Creation Type

We can create two type string:

- Single Quotes
- Double-Quotes
- Heredoc

Single Quotes String

```
$my_string = ' Welcome HPES Students ';
 echo ' Welcome HPES Students ';
 echo $my_string;
```

If you want to use a single-quote within the string you have to *escape* the single-quote with a backslash \ . Like this: \' !

String Creation Double-Quotes

Double-quotes allow for many special escaped characters to be used that you cannot do with a single-quote string.

❖$tab = "A tab is \t";

❖$dollar = "A dollar sign is \$";

❖$doublequote = "A double-quote is \"";

String Creation Heredoc

- The two methods above are the traditional way to create strings in most programming languages.
- PHP introduces a more robust string creation tool called heredoc that lets the programmer create multi-line strings without using quotations.

String Creation Heredoc

```
<?php
 $str = <<<TEST
            hello This is heredoc string demo.
           TEST;
echo $str
   ?>
```

Rule for heredoc

There are a few very important things to remember when using heredoc.

Use <<< and some identifier that you choose to begin the heredoc. In this example we chose TEST as our identifier.

Repeat the identifier followed by a semicolon to end the heredoc string creation. In this example that was TEST;

The closing sequence TEST; must occur on a line by itself and cannot be indented!

STRING Functions

PHP gives you a huge variety of functions for the munching and crunching of strings.

strlen — Get string length

“Returns the length of the given *string*. “

Syntax: int strlen (string $string)

```
<?php
    $str = 'abcdef';
    echo strlen($str);
?>
```

strcmp — Binary safe string comparison

Note that this comparison is case sensitive.

int strcmp (string $str1, string $str2)

```
<?php
$var1 = "Hello";
$var2 = "hello";
if (strcmp($var1, $var2) == 0) {
   echo ”$var1 is equal to $var2 ”;}
      else    {  echo “$var1 $var2 is not sane”;}
?>
```

Return Values

Returns < 0 if str1 is less than str2; > 0 if str1 is greater than str2, and 0 if they are equal

strcasecmp — Binary safe string comparison

Note that this comparison is case insensitive.

strcasecmp (string $str1, string $str2)

```
<?php
        $var1 = "Hello";
        $var2 = "hello";
        if (strcasecmp($var1, $var2) == 0)
        {
                echo '$var1 is equal to $var2 in a case-
                insensitive string comparison';
        }
?>
```

strstr - Find first occurrence of a string

Find first occurrence of a string.Returns part of haystack string from the first occurrence of needle to the end of haystack

String strstr (string $haystack,string $needle)

Example:

```
<?php
      $str1="Indian Computer Education";
      $str2="Computer";
      If(strstr($str1,$str2))
      {
            echo “$str1 contains $str2”;
      }
      else
      {
            echo “$str1 Doesnot contains $str2”;
      }
?>
```

strpos()

Find position of first occurrence of a string .

The strpos() function is used to search for character within a string.

int strpos (string $haystack, mixed $needle [, int $offset])

```
<?php
    echo strpos("Hello world!","world");
    $newstring = 'abcdef abcdef';
    $pos = strpos($newstring, 'a', 1);
    echo $pos;
?>
```

str_replace()

str_replace : Replace all occurrences of the search string with the replacement string

mixed str_replace (mixed $search, mixed $replace, mixed $subject [, int &$count])

Example:

```
<?php
        $vowels = array("a", "e", "i", "o", "u", "A", "E", "I", "O", "U");
        $onlyconsonants = str_replace($vowels, “@", "Hello World of PHP");
        echo $onlyconsonants;
        $str = str_replace("ll", "", "good golly miss molly!", $count);
        echo $count;
?>
```

ucfirst()

Make a string's first character uppercase

string ucfirst (string $str)

```
<?php
        $str ="this is string Demo in PHP";
        $str1=ucfirst($str);
        echo $str1."<br>";
?>
```

ucwords ()

Uppercase the first character of each word in a string

string ucwords (string $str)

```
<?php
     $str = 'hello world!';
     $str1 = ucwords($str);
     echo $str1;
?>
```

strtolower — Make a string lowercase

Syntax: strtolower (string $str)

Returns string with all alphabetic characters converted to lowercase.

```
<?php
            $str = “THIS IS STRING ";
            $str = strtolower($str);
            echo $str;
?>
```

strtoupper -Make a string uppercase

strtoupper (string $string)

Returns string with all alphabetic characters converted to uppercase.

```
<?php
      $str = “This is string demo for upper case";
      $str = strtoupper($str);
      echo $str;
?>
```

str_shuffle()

str_shuffle — Randomly shuffles a string.

string str_shuffle (string $str)

```
<?php
    $str = 'abcdef1234';
    $shuffled = str_shuffle($str);
    echo $shuffled;
    ?>
```

str_split()

str_split — Convert a string to an array.

array str_split (string $string [, int $split_length])

```
<?php
      $str = "Hello Friend";
      $arr1 = str_split($str);
      $arr2 = str_split($str, 3);
      print_r($arr1);
      print_r($arr2);
?>
```

explode()

- **array explode (string $delimiter, string $string)**
- Returns an array of strings, each of which is a substring of string formed by splitting it on boundaries formed by the string delimiter.

```
<?php
$str1 = "This is Example expload";
$substr = explode(" ",$str1);
echo print_r($substr);
?>
```

Implode()

implode — Join array elements with a string

string implode (string $glue, array $pieces)

Join array elements with a glue string.

```
<?php
	$array = array('Rahul','rahulraj@gmail.com', '12345');
	$comma_separated = implode(",", $array);
	echo $comma_separated;
?>
```

ord()

ord — Return ASCII value of character

Returns the ASCII value of the first character of string .

Syntax: int ord (string $string)

```
<?php
  echo ord(“H”);
?>
```

chr()

chr — Return a specific character

Returns a one-character string containing the character specified by ascii .

Syntax: string chr (int $ascii)

```
 <?php
     echo chr(65);
?>
```

md5()

The md5 function in PHP is used to calculate the md5 hash of a string. It is commonly used to encrypt a string.

md5 ('string', [raw_output])

Example:

```
<?php
    print md5(“helllo”)
?>
```

htmlspecialchars()

We can display some special chars especially html tags on the screen by using htmlspecialchars function of PHP.

Example:

```
<?php
print htmlspecialchars('<br> Show line break tag');
?>
```

Chapter Summary

In this chapter, you have learned about:

- Iterations like for loop, while, do..while and foreach loop.

In this chapter, you have learned:

- An array is a collection of data-elements grouped together as a single unit, organized as an ordered collection of key-value pairs.
- Array with numeric indexes called as Indexed Array.
- Associative arrays are used for accessing elements of an array by name as a key instead of numeric index.
- An array that can contain another array as value, called as multidimensional-array.
- If an array contains several elements and we want to pick one element at a time and perform some action on it, just use PHP's special loop foreach().
- The sort() function is used for sorting an indexed array in ascending order.
- String is a series of characters, where a character is the same as a byte.
- explode() function returns an array of strings, each of which is a substring of string formed by splitting it on boundaries formed by the string delimiter.
- The functionality of implode() function is Join array elements with a string

Thank you