Marwadi University

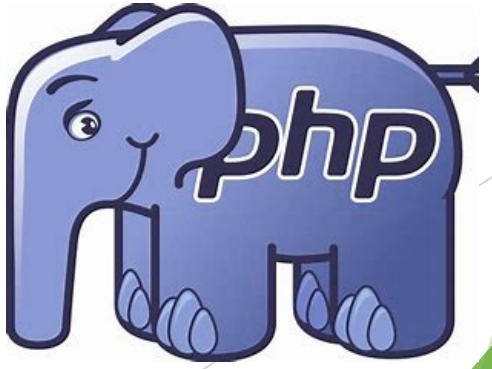
Web Application Development (PHP)

Unit – 2

Working with Functions, Arrays

Prof. Hardik Chavda

FOCA,MU



- **Working with Functions, Arrays**
  - What is a function, Calling function, Defining a function,
  - Returning values from User-defined functions,
  - Variable scope, function arguments,
  - Testing for the existence of a function,
  - What are Arrays,
  - Creating arrays,
  - Some array related functions.
- **Working with Strings, Date and Time**
  - Formatting Strings with PHP,
  - Investigating Strings in PHP,
  - Manipulating Strings with PHP,
  - Using Date and Time functions in PHP,
  - Other String,
  - Date and Time Functions.

What is a function

- PHP function is a piece of code that can be reused many times. It can take input as argument list and return value. There are thousands of built-in functions in PHP.
- Advantage of PHP Functions
  - **Code Reusability**: PHP functions are defined only once and can be invoked many times, like in other programming languages.
  - **Less Code**: It saves a lot of code because you don't need to write the logic many times. By the use of function, you can write the logic only once and reuse it.
  - **Easy to understand**: PHP functions separate the programming logic. So it is easier to understand the flow of the application because every logic is divided in the form of functions.

- Syntax

```
function functionname(){
//code to be executed
}
```

- Example

```
<?php
function sayHello(){
echo "Hello PHP Function";
}
sayHello(); //calling function
?>
```

- Output:

```
Hello PHP Function
```

PHP Function Arguments

- We can pass the information in PHP function through arguments which is separated by comma.
- PHP supports **Call by Value** (default), **Call by Reference**, **Default argument values** and **Variable-length argument list**.
- Example passing single argument

```
<?php
function sayHello($name){
echo "Hello $name<br/>";
}
sayHello(“Hardik");
sayHello(“Superman”);
sayHello(“Neo");
?>
```

Output:

Hello Hardik
Hello Superman
Hello Neo

- Example passing 2 arguments

```
<?php
function sayHello($name,$age){
echo "Hello $name, you are $age years old<br/>";
}
sayHello(“Hardik",27);
sayHello(“Superman”,29);
sayHello(“Neo",23);
?>
```

- Output:

Hello Hardik, you are 27 years old

Hello Superman, you are 29 years old

Hello Neo, you are 23 years old

PHP Call By Reference

- Value passed to the function doesn't modify the actual value by default (call by value). But we can do so by passing value as a reference.
- By default, value passed to the function is call by value. To pass value as a reference, you need to use ampersand (&) symbol before the argument name.

```
<?php
function adder(&$str2)
{
    $str2 .= 'Call By Reference';
}
$str = 'Hello ';
adder($str);
echo $str;
?>
```

Output:

```
Hello Call By Reference
```

PHP Function: Default Argument Value

- We can specify a default argument value in function. While calling PHP function if you don't specify any argument, it will take the default argument. Let's see a simple example of using default argument value in PHP function.
- Example

```
<?php
function sayHello($name="Sonoo"){
echo "Hello $name<br/>";
}
sayHello("Rajesh");
sayHello();//passing no value
sayHello("John");
?>
```

Output:

Hello Rajesh
Hello Sonoo
Hello John

PHP Function: Returning Value

Example

```
<?php
function cube($n){
return $n*$n*$n;
}
echo "Cube of 3 is: ".cube(3);
?>
```

- Output:
  - Cube of 3 is: 27

PHP Variable Length Argument Function

- PHP supports variable length argument function. It means you can pass 0, 1 or n number of arguments in function. To do so, you need to use 3 ellipses (dots) before the argument name.
- The 3 dot concept is implemented for variable length argument since PHP 5.6.
- **Example**

```
<?php
function add(...$numbers) {
    $sum = 0;
    foreach ($numbers as $n) {
        $sum += $n;
    }
    return $sum;
}
echo add(1, 2, 3, 4);
?>
```

Output:

10

PHP Arrays

- PHP array is an ordered map (contains value on the basis of key). It is used to hold multiple values of similar type in a single variable.
- Advantage of PHP Array
  - **Less Code**: We don't need to define multiple variables.
  - **Easy to traverse**: By the help of single loop, we can traverse all the elements of an array.
  - **Sorting**: We can sort the elements of array.
- PHP Array Types
  - Indexed Array
  - Associative Array
  - Multidimensional Array

PHP Indexed Array

- PHP index is represented by number which starts from 0. We can store number, string and object in the PHP array. All PHP array elements are assigned to an index number by default.
- There are two ways to define indexed array:
- 1st way:
  - $season=**array**("summer","winter","spring","autumn");
- 2nd way:
  - $season[0]="summer";
  - $season[1]="winter";
  - $season[2]="spring";
  - $season[3]="autumn";

- Example

```
<?php
$season=array("summer","winter","spring","autumn");
echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
?>
```

- Output:
  - Season are: summer, winter, spring and autumn
- Example 2

```
<?php
$season[0]="summer";
$season[1]="winter";
$season[2]="spring";
$season[3]="autumn";
echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
?>
```

- Output:
  - Season are: summer, winter, spring and autumn

Traversing PHP Index Arrays

- <?php
- $size=array("Big","Medium","Short");
- foreach($size as $s)
- {
- echo "Size is: $s
";
- }
- ?>
- Output:

- Size is: Big
- Size is: Medium
- Size is: Short

PHP Associative Array

- We can associate name with each array elements in PHP using => symbol.
- There are two ways to define associative array:
- 1st way:
  - $salary=**array**("Sonoo"=>"350000","John"=>"450000","Kartik"=>"200000");
- 2nd way:
  - $salary["Sonoo"]="350000";
  - $salary["John"]="450000";
  - $salary["Kartik"]="200000";

- Example

```
<?php
$salary=array("Sonoo"=>"350000","John"=>"450000","Kartik"=>"200000");
echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
echo "John salary: ".$salary["John"]."<br/>";
echo "Kartik salary: ".$salary["Kartik"]."<br/>";
?>
```

Output:

Sonoo salary: 350000

John salary: 450000

Kartik salary: 200000

- Example 2

```
<?php
$salary["Sonoo"]="350000";
$salary["John"]="450000";
$salary["Kartik"]="200000";
echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
echo "John salary: ".$salary["John"]."<br/>";
echo "Kartik salary: ".$salary["Kartik"]."<br/>";
?>
```

- Output:
  - Sonoo salary: 350000
  - John salary: 450000
  - Kartik salary: 200000

Traversing PHP Associative Array

- By the help of PHP for each loop, we can easily traverse the elements of PHP associative array.

```
<?php
$salary=array(“Hardik"=>"550000",“Mukesh"=>"250000","Ratan"=>"200000");
foreach($salary as $k => $v) {
echo "Key: ".$k." Value: ".$v."<br/>";
}
?>
```

- Output:

```
Key: Hardik Value: 550000
Key: Mukesh Value: 250000
Key: Ratan Value: 200000
```

PHP Multidimensional Array

- PHP multidimensional array is also known as array of arrays. It allows you to store tabular data in an array. PHP multidimensional array can be represented in the form of matrix which is represented by row * column.

```
$emp = array
  (
  array(1,"sonoo",400000),
  array(2,"john",500000),
  array(3,"rahul",300000)
  );
```

PHP Multidimensional Array Example

```
<?php
$emp = array
  (
  array(1,"sonoo",400000),
  array(2,"john",500000),
  array(3,"rahul",300000)
  );

for ($row = 0; $row < 3; $row++) {
  for ($col = 0; $col < 3; $col++) {
    echo $emp[$row][$col]."  ";
  }
  echo "<br/>";
}
?>
```

Id	Name	Salary
1	sonoo	400000
2	john	500000
3	rahul	300000

Output:

1 sonoo 400000
2 john 500000
3 rahul 300000

Arrays functions

array()	Creates an array
array_change_key_case()	Changes all keys in an array to lowercase or uppercase
array_chunk()	Splits an array into chunks of arrays
array_combine()	Creates an array by using the elements from one "keys" array and one "values" array
array_count_values()	Counts all the values of an array
array_merge()	Merges one or more arrays into one array
array_pop()	Deletes the last element of an array
array_push()	Inserts one or more elements to the end of an array
array_replace()	Replaces the values of the first array with the values from following arrays

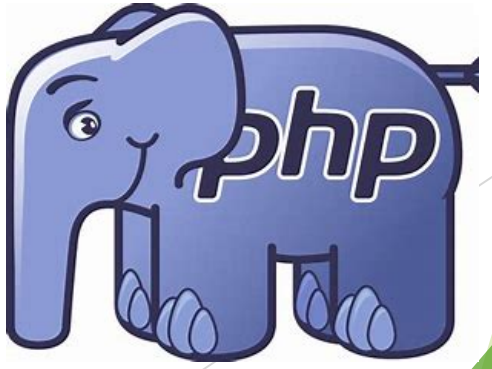
array_reverse()	Returns an array in the reverse order
array_search()	Searches an array for a given value and returns the key
array_shift()	Removes the first element from an array, and returns the value of the removed element
array_unique()	Removes duplicate values from an array
array_values()	Returns all the values of an array
count()	Returns the number of elements in an array
sort()	Sorts an indexed array in ascending order
in_array()	Checks if a specified value exists in an array

Marwadi University

Web Application Development (PHP)

Unit – 2

Working with Strings, Date and Time



Strings

- PHP string is a sequence of characters i.e., used to store and manipulate text. PHP supports only 256-character set and so that it does not offer native Unicode support.
- There are 4 ways to specify a string literal in PHP.
  - single quoted
  - double quoted
  - heredoc syntax
  - newdoc syntax (since PHP 5.3)

- **Single Quoted**
- We can create a string in PHP by enclosing the text in a single-quote. It is the easiest way to specify string in PHP.
- For specifying a literal single quote, escape it with a backslash (\) and to specify a literal backslash (\) use double backslash (\\). All the other instances with backslash such as \r or \n, will be output same as they specified instead of having any special meaning.

- Example

```
<?php
    $str='Hello text within single quote';
    echo $str;
?>
```

- Output:
  - Hello text within single quote

- **Double Quoted**
- In PHP, we can specify string through enclosing text within double quote also. But escape sequences and variables will be interpreted using double quote PHP strings.
- Example

```
<?php
$str="Hello text within double quote";
echo $str;
?>
```

- Output:
  - Hello text within double quote

- **Heredoc**
- Heredoc syntax (<<<) is the third way to delimit strings. In Heredoc syntax, an identifier is provided after this heredoc <<< operator, and immediately a new line is started to write any text. To close the quotation, the string follows itself and then again that same identifier is provided. That closing identifier must begin from the new line without any whitespace or tab.
- Naming Rules
- The identifier should follow the naming rule that it must contain only alphanumeric characters and underscores, and must start with an underscore or a non-digit character.
- For Example
- <?php

  $str = <<<Demo

  It is a valid example

  Demo; //Valid code as whitespace or tab is not valid before closing identifier

  echo $str;
- ?>
- Output:
  - It is a valid example

- **Newdoc**
- Newdoc is similar to the heredoc, but in newdoc parsing is not done. It is also identified with three less than symbols <<< followed by an identifier. But here identifier is enclosed in single-quote,
  - **e.g. <<<'EXP'**. Newdoc follows the same rule as heredocs.
- The difference between newdoc and heredoc is that - Newdoc is a **single-quoted string** whereas heredoc is a **double-quoted string**.
- **Note:** Newdoc works as single quotes.

```
<?php
    $str = <<<'DEMO'
    Welcome to Marwadi University.
        Learn with newdoc example.
DEMO;
echo $str;
echo '</br>';

echo <<< 'Demo'     // Here we are not storing string content in variable str.
    Welcome to  Marwadi University.
        Learn with newdoc example.
Demo;
?>
```

String Functions

echo()	It is used for output one or more strings.
print()	It is used for output one or more strings.
strpos()	It is used to return the position of the first occurrence of a string inside another string.
ltrim()	It is used to remove whitespace from the left side of a string.
trim()	Remove whitespace or other characters from the beginning and end of the string.
rtrim()	It is used to remove whitespace from the right side of a string.
strrev()	It is used to reverse a string.
strlen()	It is used to return the length of a string.
strtolower()	Convert the string in lowercase
strtoupper()	Convert the strings in uppercase
str_replace()	It replaces all occurrences of the search string with the replacement string.

Date & Time

- The PHP **date()** function formats a timestamp to a more readable date and time.
- Syntax
  - **date**(format,timestamp)
- Parameter Description
  - **format**: Required. Specifies the format of the timestamp
  - **timestamp** : Optional. Specifies a timestamp. ***Default is the current date and time***.
- The required format parameter of the date() function specifies how to format the date (or time).
- Here are some characters that are commonly used for dates:
  - d - Represents the day of the month (01 to 31)
  - m - Represents a month (01 to 12)
  - Y - Represents a year (in four digits)
  - l (lowercase 'L') - Represents the day of the week
- Other characters, like"/", ".", or "-" can also be inserted between the characters to add additional formatting.

- **Example**

```
<?php
echo "Today is " . date("Y/m/d") . "<br>";
echo "Today is " . date("Y.m.d") . "<br>";
echo "Today is " . date("Y-m-d") . "<br>";
echo "Today is " . date("l");
?>
```

- **Output**

Today is 2020/11/03
Today is 2020.11.03
Today is 2020-11-03
Today is Tuesday

Get a Time

- Here are some characters that are commonly used for times:
  - H - 24-hour format of an hour (00 to 23)
  - h - 12-hour format of an hour with leading zeros (01 to 12)
  - i - Minutes with leading zeros (00 to 59)
  - s - Seconds with leading zeros (00 to 59)
  - a - Lowercase Ante meridiem and Post meridiem (am or pm)
- Example
  - <?php
    echo "The time is " . date("h:i:sa");
    ?>
- Output
  - The time is 03:26:36pm

Get Your Time Zone

- If the time you got back from the code is not correct, it's probably because your server is in another country or set up for a different timezone.
- So, if you need the time to be correct according to a specific location, you can set the timezone you want to use.
- The example below sets the timezone to "America/New_York", then outputs the current time in the specified format:
- **Example**
  - ```
    <?php
    date_default_timezone_set("America/New_York");
    echo "The time is " . date("h:i:sa");
    ?>
    ```
- **Output**
  - The time is 11:28:12am
    ```

PHP Time Function

- The time() function is used to get the current time as a Unix timestamp (the number of seconds since the beginning of the Unix epoch: January 1, 1970, 00:00:00 GMT).
- The following characters can be used to format the time string:
  - h: Represents hour in 12-hour format with leading zeros (01 to 12).
  - H: Represents hour in 24-hour format with leading zeros (00 to 23).
  - i: Represents minutes with leading zeros (00 to 59).
  - s: Represents seconds with leading zeros (00 to 59).
  - a: Represents lowercase antemeridian and post meridian (am or pm).
  - A: Represents uppercase antemeridian and post meridian (AM or PM).

- Example

```
<?php
  $timestamp = time();
  echo($timestamp);
  echo "\n";
  echo(date("F d, Y h:i:s A", $timestamp));
?>
```

- Output
  - 1512486297
  - December 05, 2017 03:04:57 PM

Create a Date With mktime()

- The optional timestamp parameter in the date() function specifies a timestamp. If omitted, the current date and time will be used (as in the examples above).
- The PHP mktime() function returns the Unix timestamp for a date. The Unix timestamp contains the number of seconds between the Unix Epoch (January 1 1970 00:00:00 GMT) and the time specified.
- Syntax
  - mktime(*hour, minute, second, month, day, year*)
- Example

```
<?php
$d=mktime(11, 14, 54, 8, 12, 2022);
echo "Created date is " . date("Y-m-d h:i:sa", $d);
?>
```

- OP
  - Created date is 2022-08-12 11:14:54am