



Lecture 4


PHP Functions

Topics

- User-defined functions
- Function arguments
- Returning values
- Variable functions
- Internal (built-in) functions
- Anonymous functions

Defining a function

- A function is a routine (or more accurately, a sub routine) that can be called to perform its duties from any point in the program that needs it.
- Allows to only write the code once and save a lot of time and save

- 
- A function is a block of statements that can be used repeatedly in a program.
 - A function will not execute immediately when a page loads.
 - A function will be executed by a call to the function.
 - *A code is probable be a mixture of existing functions combined with your own logic*

User Define Functions

- Functions that defined by users according to their requirements.
- User defined function declaration starts with the word “function”

User Define Functions

- Functions that defined by users according to their requirements.
- User defined function declaration starts with the word “function”

User Define Functions

- Syntax

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

Function declaration begins with the keyword function. So that PHP parser know that what follows is a user defined function

Guidelines for PHP functions

- Give a function name that reflects what the function does
- Function cannot have the same name as an existing function
- A valid function name starts with a letter or underscore, followed by any number of letters, numbers, or underscores
- Function name cannot begin with a digit
- Function names are case sensitive

Conditional Functions :

```
<?php
    $makefoo = true;
    /* We can't call foo() from here
       since it doesn't exist yet,
       but we can call bar() */
bar();
if ($makefoo) {
    function foo()
    {
        echo "I don't exist until program execution reaches me.\n";
    }
}

/* Now we can safely call foo()
   since $makefoo Something is wronguaded to true */
if ($makefoo) foo();

function bar()
{
    echo "I exist immediately upon program start.\n";
}
```

Functions need not be defined before they are referenced, *except* when a function is conditionally defined as shown in the example.

When a function is defined in a conditional manner as shown. Its definition must be processed *prior* to being called.

Functions within functions

```
<?php
function foo()
{
    function bar()
    {
        echo "I don't exist until foo() is called.\n";
    }
}
```

```
/* We can't call bar() yet
   since it doesn't exist. */
```

```
foo();
```

```
/* Now we can call bar(),
   foo()'s processing has
   made it accessible. */
```

```
bar();
```

Function Arguments

- Calling values that are sent to a function are called arguments
- information may be passed to functions via the argument list,
 - which is a comma-delimited list of expressions.
 - The arguments are evaluated from left to right.
- Syntax
function Myfunction(\$arg1, \$arg2)

Function arguments

- Function arguments can be passed by two ways,
 - Passed by value
 - Passed by reference

Passed by value

- Copy of the variable's value is manipulated by the function.

Passed by value

```
<?php
function addFive($num) {
    $num+= 5;

}
$orignum= 10;
addFive( $orignum);
echo "Original Value is $orignum<br/>";
?>
```

Output:

Original Value is 10 ??

Any value return from the function?

Passed by reference

- A reference to the variable is manipulated by the function rather than a copy of the variable's value.
- Any changes made to an argument in these cases will change the value of the original variable.
- Pass an argument by reference by adding an ampersand to the variable name in either the function call or the function definition.

Passed by reference

```
<?php
function addSix(&$num) {
    $num+= 6;
}
$orignum= 10;
addSix( $orignum);
echo "Original Value is $orignum<br/>";
?>
```

Output:
Original Value is 16

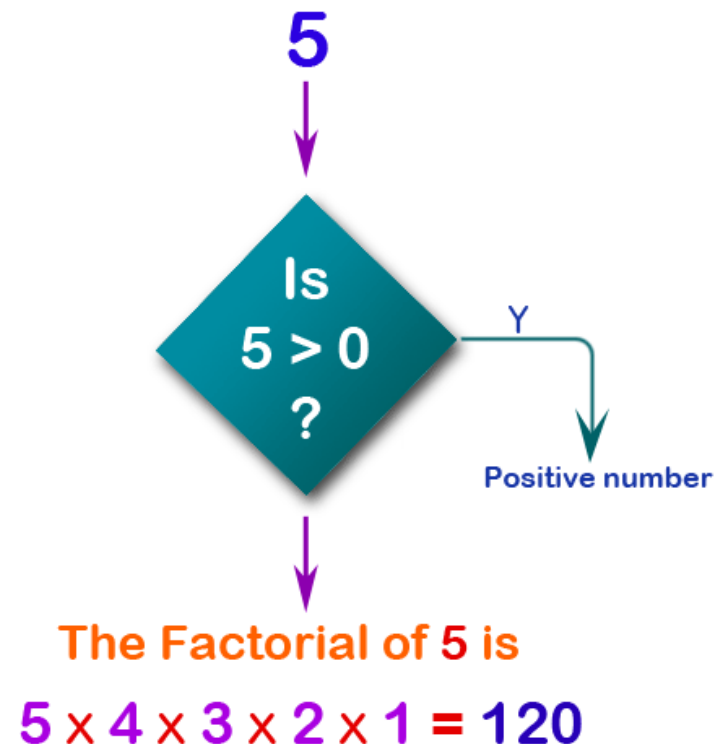
What will be the output of the following PHP code?

```
<?php
function addFive($num)
{
    $num += 5;
}
function addSix(&$num)
{
    $num += 6;
}
$orignum = 10;
addFive( &$orignum );
echo "Original Value is $orignum<br />";
addSix( $orignum );
echo "Original Value is $orignum<br />";
?>
```

- a) Original Value is 15
Original Value is 21
- b) Original Value is 15
Original Value is 21
- c) Original Value is 15
Original Value is 15
- d) None Of The mentioned

addSix() passes value of the variable by reference.

- Write a function to calculate the factorial of a number (a non-negative integer). The function accepts the number as an argument.



Default arguments in function

- When designing functions, it is often helpful to be able to assign default values for parameters that aren't passed
- PHP does this for most of its functions, and it saves you having to pass in parameters most of the time if they are usually the same.
- To define default parameters for a function, simply add the constant value you would like them set to after the variables,

Default Arguments in Functions

```
<?php
function setHeight($minheight=50)
{
    echo"The height is :". $minheight."<br>";
}
setHeight(350);
setHeight();// will use the default value of 50
setHeight(135);
setHeight(80);
?>
```

Returning values

- To let a function return a value,
- Ex:

```
<?php
function sum($x, $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);
?>
```

Recursive functions

: **Recursion occurs when a function calls itself**

```
<?php
//Our recursive function.
function recursive($num){
    //Print out the number.
    echo $num, '<br>';
    //If the number is less or equal to 50.
    if($num < 50){
        //Call the function again. Increment
        number by one.
        return recursive($num + 1);
    }
}
```

```
//Set our start number to 1.
$startNum = 1;
```

```
//Call our recursive function.
recursive($startNum);
```

```
?>
```

- **recursion** occurs when something contains, or uses, a similar version of itself. That similar version then contains or uses another similar version of itself, and so on.
- Sometimes this process can go on forever
- The number of repetitions, or "depth" of the recursion, is limited by some sort of end condition.

Variable-length argument list

- PHP also allows to access arguments provided in the function call operation without using argument variables,

1.func_get_arg(position) –

- It returns the value of the specified argument provided in the function call operation.
- "position" is the position index of the specified argument.
- The position index of the first argument is 0.

For example, if "func_get_arg(2)" is used in a function, it will return the value of the 3rd argument

Variable-length argument list

2. `func_num_args()` –

- It returns the total number of arguments provided in the function call operation.
- For example, if "`func_num_args()`" returns 1, you know that there is only 1 argument provided by calling code

Variable-length argument list

3. **func_get_args()** –

- It creates and returns an array which contains values of all arguments provided in the function call operation.
- For example, if "\$args= func_get_args()" is used in a function, \$args will be array containing all arguments

Variable-length argument list

- With all these 3 functions, PHP supports variable-length argument lists with some basic rules:
 - The function call can provide more arguments than the number of argument variables defined in the function statement.
 - You can pass arguments to a function that has no argument variable defined

Variable-length and argument lists of a user defined function

- **func_num_args()**
 - Returns the number of arguments passed to the function
- **func_get_arg()**
 - Return an item from the argument list
- **func_get_args()**
 - Returns an array comprising a function's argument list

Variable-length argument list

```
<?php
function dynamic_args() {
    for($i= 0 ; $i< func_num_args(); $i++) {
        echo "Argument $i=
".func_get_arg($i)."<br/>";
    }

    $args= func_get_args();
    foreach($args as $key => $value) {
        echo "Argument {$key}: {$value} <br/>";
    }
}

dynamic_args("a", "b", "c", "d", "e");
?>
```

Returning values

- A function returns a result to the calling code by using the "return: keyword.

```
<?php
```

```
function square($num) {  
    return $num * $num;  
}
```

```
echo square(4); // outputs '16'.
```

```
?>
```

- To get multiple values, should return an array.

Returning values as an array

```
<?php
    function small_numbers()
    {
        return array (0, 1, 2);
    }
    list ($zero, $one, $two) = small_numbers();
    echo "The array elements, 1 $zero, 2 $one
and 3 $two.";
?>
```

list() function

- The list() function is used to assign values to a list of variables in one operation.
- list() only works on numerical arrays and assumes the numerical indices start at 0.

Built-In functions


- We don't have to define those functions. We just need to use them, by calling their names and passing them values.
- A list of all PHP built-in functions is given at **<http://www.php.net/quickref.php>**

Categories of Built-In Function

- String handling and manipulating functions
- Array manipulating functions
- Date/Time functions
- MySQL functions to access MySQL database servers.
- Math functions
- Mail handling Functions

String Functions

- **echo()** : **Outputs one or more strings**
- **explode()**: **Break a string into an array**
 - `explode(separator,string,limit)`
- **strlen()**: **Returns the length of a string**
- **trim()**: **Strips whitespace from both sides of a string**
 - `ltrim()` and `rtrim()`
- **str_replace()**: **Replaces some characters in a string (case-sensitive)**
 - `substr_replace(string,replacement,start,length)`



```
<?php
```

```
//echo(strings)
```

```
$str1 = "Who's Kai Jim?";
```

```
echo $str1;
```

```
//explode(separator,string)
```

```
$str2 = "Hello world. It's a beautiful day.";
```

```
print_r(explode(" ", $str2)); //print_r(var_name, return_output)
```

```
Array ( [0] => Hello [1] => world. [2] => It's  
[3] => a [4] => beautiful [5] => day.)
```

```
//str_replace(find,replace,string,count)
```

```
echo str_replace("world", "Peter", "Helloworld!");
```

```
?>
```

```
<?php
//strlen(string)
echo strlen("Hello world!");
```

```
//trim(string,charlist)
$str = "Hello World!";
echo $str . "<br>";
echo trim($str,"Hed!");
```

```
//strrev(string)
$str5 = "Hello world";
echo strrev("Hello world!");
?>
```

Array functions

- Allow to manipulate arrays.
- Supports both simple and multi-dimensional arrays
- Few of the Mostly use functions
 - `array()` Creates an array
 - `current()` Returns the current element in an array
 - `next()` Advance the internal array pointer of an array
 - `reset()` Sets the internal pointer of an array to its first element
 - `sort()` Sorts an array

Date/Time functions

- Date/Time functions allow to extract and format the date and time on the server

Function	Description	Format
date()	Formats a local time/date	d- the day of the month (from 01 to 31)
		D-A textual representation of a day (3 letters)
		m-a numerical representation of a month (from 01-12)
		M - A short textual representation of a month (three letters)
		Y - A four digit representation of a year
		y - A two digit representation of a year

Math Functions

- Math functions can handle values within the range of integer and float types.
- Few of the mostly used functions:

Function	Description
<code>cos()</code>	Returns the cosine of a number
<code>exp()</code>	Returns the value of E^x
<code>log()</code>	Returns the natural logarithm (base E) of a number
<code>max()</code>	Returns the number with the highest value of two specified numbers
<code>pow()</code>	Returns the value of x to the power of y
<code>round()</code>	Rounds a number to the nearest integer

Mail Functions

- The mail() function allows you to send emails directly from a script.
- Syntax:
 - mail(to,subject,message,headers,parameters)

Parameter	Description
to	Required. Specifies the receiver / receivers of the email
subject	Required. Specifies the subject of the email.
message	Required. Defines the message to be sent.
headers	Optional. Specifies additional headers, like From, Cc, and Bcc.
parameters	Optional. Specifies an additional parameter to the sendmail program

- ```
<?php
$to = "somebody@example.com";
$subject = "My subject";
$txt = "Hello world!";
$headers = "From:
webmaster@example.com" . "\r\n" .
"CC: somebodyelse@example.com";

mail($to,$subject,$txt,$headers);
?>
```

# Summary

- The real power of PHP comes from its functions.
- Creating functions lets you reuse code that you've used before without rewrite the whole. This will save you time and make programming easier.