PHP Functions

A function is a block of code written in a program to perform some specific task.

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.

Any name ending with an open and closed parenthesis is a function.

A function will not execute automatically when a page loads.

A function will be executed by a call to the function.

In PHP, we can define **Conditional function**, **Function within Function** and **Recursive function** also.

Advantage of PHP Functions

- **Code Reusability**
- **Less Code**
- **Easy to understand**
- **Easier error detection**

PHP provides us with two major types of functions:

User Defined Functions and **Built-in functions**

PHP User-defined Functions

PHP allows us to create our own customised functions called the user-defined functions.
Using this we can create our own packages of code and use it wherever necessary by simply calling it.

Syntax

```
1. function functionname(){
2. //code to be executed
```

3. }

Note: Function name must be start with letter and underscore only like other labels in PHP. It can't be start with numbers or special symbols. Function names are NOT case-sensitive.

PHP Functions Example

File: function1.php

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

Output:

```
Hello world!
```

In the example above, we create a function named "writeMsg()". The opening curly brace ({) indicates the beginning of the function code, and the closing curly brace (}) indicates the end of the function. The function outputs "Hello world!". To call the function, just write its name followed by brackets ():

PHP Function Parameters or Arguments

Information/value can be passed to functions through parameters.

Parameters are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma. These parameters are used to accept inputs during runtime. While passing the values like during a function call, they are called arguments. An argument is a value passed to a function and a parameter is used to hold those arguments. In common term, both parameter and argument mean the same. We need to keep in mind that for every parameter, we need to pass its corresponding argument.

Let's see the example to pass single argument in PHP function.

File: functionarg.php

```php
<?php
```

```
function sayHello($name){
echo "Hello $name<br/>";
}
sayHello("Sonoo");
sayHello("Vimal");
sayHello("John");
?>
```

Output:

```
Hello Sonoo
Hello Vimal
Hello John
```

Let's see the example to pass two argument in PHP function.

File: functionarg2.php

```
<?php
function sayHello($name,$age){
echo "Hello $name, you are $age years old<br/>";
}
sayHello("Sonoo",27);
sayHello("Vimal",29);
sayHello("John",23);
?>
```

Output:

```
Hello Sonoo, you are 27 years old
Hello Vimal, you are 29 years old
Hello John, you are 23 years old
```

PHP is a Loosely Typed Language

In php, we did not have to specify data type the 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. This gives us an option to specify the expected data type when declaring a function, and by adding the `strict` declaration, it will throw a "Fatal Error" if the data type mismatches.

In the following example we try to send both a number and a string to the function without using `strict`:

Example

```php
<?php
function addNumbers(int $a, int $b) {
  return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is NOT enabled "5 days" is changed to int(5), and it will
return 10
?>
```

Output:

10

To specify `strict` we need to set `declare(strict_types=1);`. This must be on the very first line of the PHP file.

In the following example we try to send both a number and a string to the function, but here we have added the `strict` declaration:

Example

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

function addNumbers(int $a, int $b) {
  return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is enabled and "5 days" is not an integer, an error will be
thrown
?>
```

Output:

PHP Fatal error: Uncaught TypeError: Argument 2 passed to addNumbers() must be of the type integer, string given, called in /home/0NwDXa/prog.php on line 7 and defined in /home/0NwDXa/prog.php:4 Stack trace: #0 /home/0NwDXa/prog.php(7): addNumbers(5, '5 days') #1 {main} thrown in /home/0NwDXa/prog.php on line 4

The `strict` declaration forces things to be used in the intended way.

PHP Function: Default Argument Value

PHP allows us to set default argument values for function parameters. If we do not pass any argument for a parameter with default value

then PHP will use the default set value for this parameter in the function call.

Example:

File: functiondefaultarg.php

```php
<?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

Functions can also return values to the part of program from where it is called. The *return* keyword is used to return value back to the part of program, from where it was called. The returning value may be of any type including the arrays and objects. The return statement also marks the end of the function and stops the execution after that and returns the value.

Example:

File: functiondefaultarg.php

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

Output:

```
Cube of 3 is: 27
```

Parameter passing to Functions

PHP allows us two ways in which an argument can be passed into a function:

- **Pass by Value:** By default, value passed to the function is call by value. On passing arguments using pass by value, the value of the argument gets changed within a function, but the original value outside the function remains unchanged. That means a duplicate of the original value is passed as an argument.

- **Pass by Reference:** When a function argument is passed by reference, changes to the argument also change the variable that was passed in. In pass by reference, we actually pass the address of the value, where it is stored using ampersand sign (&).

Example:

```php
<?php

// pass by value
function valGeek($num) {
    $num += 10;
    return $num;
}

// pass by reference
function refGeek(&$num) {
    $num += 10;
    return $num;
}

$n = 10;

valGeek($n);
echo "The original value is still $n \n";

refGeek($n);
echo "The original value changes to $n";

?>
```

Output:

```
The original value is still 10
The original value changes to 20
```