Function and String

Prepared By Deepali Sonawane, DoCSA, MIT-WPU

Function

- A **function** is a block of statements that can be used repeatedly in a program.
- There are two types of function
  - Built in function / Predefined function
  - User defined function
- PHP has 1000+ built-in functions that can be called directly, from within a script, to perform a specific task.
- Examples :

  ord(character) : To get ASCII value of specified character

  chr(value) : To get character for specified value

  array() : To define array

  date() : To get current date

  time() : To get current time

User defined function

- Besides the built-in PHP functions, it is possible to create our own functions.
- A function will be executed by a call to the function.
- Two steps to write user defined function:
  - Function definition
  - Function call
- A user-defined function declaration starts with the word **function**

  Syntax: function functionName([parameters])

  ```
              {
                    Statements;
              }
  ```
- A function name must start with a letter or an underscore.
- Function names are **NOT** case-sensitive.
- **Syntax** for function call is functionName([parameters]);

Types of functions

- **Non parameterized function** : Function without parameters are called as Non parameterized function.
- Example:

```
<?php
        function display()
        {
                echo “Hello World!!!”;
        }
        display();
?>
```

Types of functions

- **Parameterized function**: Function with the parameters are called as parameterized function.
- Example:

```
<?php
        function display($s)
        {
                echo $s;
        }
        display("PHP Programming");
        display("JAVA Programming");
?>
```

Types of functions

- **Parameterized function with two parameters:**
- Example:

```
<?php
        function display($nm,$rn)
        {
                echo “Roll Number of $nm is $rn”;
        }
        display(“Akash”,20);
?>
```

Types of Parameters

- **There are two types of parameters in PHP**
  1. **Actual Parameters :** These parameters are used in **function call**, that contains actual value of variables.

     Example : addition($a,$b)
  2. **Formal or Dummy Parameters :** These parameters are used in **function definition**. Example : function addition($c,$d)
- Example:

```
<?php
        function addition($c,$d)
        {
                echo "Addition is ", $c+$d;
        }
        $a=10, $b=20;
        addition($a,$b);
?>
```

Default parameters

- In the parameterized function we can set default value to the parameter.
- Specify default values **from right to left.**
- Example:

```
<?php
        function addition($a=1,$b=1)
        {
                $c = $a+$b;
                echo “Addition is ”, $c;
        }
        addition(10,20);//30
        addition(10);//11
        addition();//2
?>
```

Function with returning value

- **return** statement is used to **return value** from the function.
- Example:

```
<?php
        function addition($a,$b)
        {
                $c = $a+$b;
                return $c;
        }
        $d = addition(10,20);
        echo "<br>Addition is " , $d;
        echo "<br>Addition is ", addition(37,89);
?>
```

Methods of passing parameters to function

- There are two methods to pass parameters to function :

1. **Call by Value** : In call by value method, **actual values** are passed to the function and that are modified inside the function but not outside the function.
2. **Call by reference** :In call by reference method, actual values are modified if they are modified inside the function. In such case, we use & (ampersand) symbol with formal arguments. The & represents **reference of the variable**.

   Example : function swap(&$a , &$b)

```
              {
                      $t = $a;
                      $a = $b;
                      $b = $t;
              }
```

Variable Function

- If name of a variable has **parentheses** (with or without parameters in it) in front of it, PHP parser tries to find a function whose name corresponds to **value** of the variable and executes it. Such a function is called **variable function**.
- This allows for **dynamic function** calls at runtime.

Example:
```
<?php
function display(){
   echo “I am in display function";
}
function show($s){
  echo “<br>$s”;
}
$var=“display";
$var();
$var=“show”;
$var(“I am in show function”);
?>
```

Anonymous Function

- An **anonymous function** is a function that doesn't have **any name** specified at the time of definition.
- Syntax : $var=function ($arg1, $arg2) { return $val; };
- Note that there is no **function name** between the **function** keyword and the **opening parenthesis**, and the fact that there is a **semicolon** after the function definition.
- This implies that anonymous function definitions are **expressions**.
- When assigned to a variable, the anonymous function can be called later using the **variables name**.

Example for Anonymous Function

- Example 1:

```
<?php
$add = function ($a, $b) { return “Addition is: " . $a+$b; };
echo $add(5,10);
?>
```

- Example 2:

```
<?php
$add = function ($a, $b) {
        return “Addition is: " . $a+$b;
        };
        echo $add(5,10);
?>
```

Recursive Function

- **Recursive Function** : It is a function which calls itself.
- **Recursion** : It is a process in which function calls itself from its body.
- **Advantages of recursive function**:
  1. Reduce unnecessary calling of function.
  2. Through Recursion one can Solve problems in easy way while its iterative solution is very big and complex.
  3. Recursion uses stack to store data so that we get previous value of a variable.
- **Disadvantages of recursive function**:
  1. Recursive solution is always logical and it is very difficult to trace.(debug and understand).
  2. In recursive function we must have an if statement somewhere to force the function to return value.
  3. Recursion uses more processor time.
  4. Recursion takes a lot of stack space, usually not considerable when the program is small.

Recursive Function

- Example:

```
<?php
        function factorial($n)
        {
                if ($n>=1)
                return $n*factorial($n-1);
                else
                return 1;
        }
        echo "<br>Factorial is ", factorial(5);
?>
```