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

Example of Recursive Function

- Example:

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

Creating and Accessing Strings

- String is a **sequence of characters or array of Characters.**
- Example: $str = “PHP”;
- Code to display string :

```
<?php
        $str = “PHP Programming”;
        for($i =0; $i<strlen($str);$i++)
        {
                echo “<br>”, $str[$i];
        }
?>
```

Formatting Strings

- Like many other languages, PHP features the printf() function that we can use to format strings in many different ways. These functions are handy when we need convert data between different formats.
- Example :

```
<?php
        $name = “Ashish”;
        $rno=10;
        printf(“Roll number of %s is %d” , $name , $rno);
?>
```

List of format specifier

- %b: Format the argument as a binary integer. (e.g. 10010110)
- %c: Format the argument as a character with the argument's ASCII value
- %d: Format the argument as a signed decimal integer
- %e : Format the argument in scientific notation (e.g. 1.234e+3)
- %f : Format the argument as a floating-point number
- %o : Format the argument as an octal integer
- %s : Format the argument as a string
- %u : Format the argument as an unsigned decimal integer
- %x : Format the argument as a lowercase hexadecimal integer (e.g. 4fdf87)
- %X : Format the argument as an uppercase hexadecimal integer (e.g. 4FDF87)

Regular Expression and Pattern matching

- A regular expression is a sequence of characters that **forms a search pattern**. When you search for data in a text, you can use this search pattern to describe what you are searching for.
- Regular expressions can be used to perform all types of **text search** and **text replace** operations.
- In PHP, regular expressions are strings composed of delimiters, a pattern and optional modifiers.
- Example : $exp = "/Kothrud/i";

  In the example above, / is the **delimiter**, Kothrud is the **pattern** that is being searched for, and i is a **modifier** that makes the search case-insensitive.
- The delimiter can be any character that is not a letter, number, backslash or space. The most common delimiter is the **forward slash (/)**, but when your pattern contains forward slashes it is convenient to choose other delimiters such as **# or ~**.

Regular Expression Functions

- preg_match($pattern,$str) : Returns 1 if the pattern was found in the string and 0 if not
- preg_match_all ($pattern,$str) : Returns the number of times the pattern was found in the string, which may also be 0
- preg_replace($pattern,$replace,$str) : Returns a new string where matched patterns have been replaced with another string.
- preg_split($pattern, $str[,$limit,$flags]): This function breaks a string into an array using matches of a regular expression as separators.

  where limit is the number of elements that the returned array can have and flags is PREG_SPLIT_NO_EMPTY : Empty strings will be **removed** from the returned array.

  PREG_SPLIT_DELIM_CAPTURE: If the regular expression contains a group wrapped in parentheses, matches of this group will be included in the returned array.

  PREG_SPLIT_OFFSET_CAPTURE :Each element in the returned array will be an array with two element, where the first element is the substring and the second element is the **position of the first character** of the substring in the input string.

Regular Expression Functions

- Example for preg_match():

```
<?php
   $str = “Good morning";
   $pattern = "/Morning/i";
   echo preg_match($pattern, $str); // Output: 1
?>
```

- Example for preg_match_all():

```
<?php
   $str = "The rain in SPAIN falls mainly on the plains.";
   $pattern = "/ain/i";
   echo preg_match_all($pattern, $str); // Output: 4
?>
```

Regular Expression Functions

- Example for preg_replace():

```
<?php
   $str = “Good morning!";
   $pattern = “/morning /i";
   echo preg_replace($pattern, “afternoon", $str); // Output: "Good afternoon!“
?>
```

- Example for preg_split():

```
<?php
$date = “10-08-2025";
$pattern = "/-/";
$components = preg_split($pattern, $date, 3, PREG_SPLIT_NO_EMPTY);
print_r($components);
?>
```

Regular Expression Functions

- Example for preg_split():

```
<?php
$date =“10-08-2025";
$pattern = "/(-)/";
$components = preg_split($pattern, $date, 5, PREG_SPLIT_DELIM_CAPTURE);
print_r($components);
?>
```

- Example for preg_split():

```
<?php
    $date = "10-08-2025";
    $pattern = "/-/";
    $components = preg_split($pattern, $date, 3, PREG_SPLIT_OFFSET_CAPTURE  );
    print_r($components);
?>
```

Regular Expression Patterns

- Brackets are used to find a range of characters:
  - [abc] : Find one character from the options between the brackets
  - [^abc] : Find any character NOT between the brackets
  - [0-9] : Find one character from the range 0 to 9
- Example:

```
<?php
   $str = "The rain in SPAIN falls mainly on the plains.";
   $pattern = "/[ah]/i";
   echo preg_match_all($pattern, $str); //Output is 7
?>
```

String Functions

1. strlen($str) : It returns the string length.
2. str_word_count($str) : It returns the number of words in a string.
3. strrev($str) : This function reverses a string.
4. strpos($str1,$str2): This function searches for a specific text within a string. If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.
5. str_replace($search,$replace,$str): This function replaces some characters with some other characters in a string.
6. ucfirst($str) : converts the first character of a string to uppercase
7. lcfirst($str) : converts the first character of a string to lowercase
8. ucwords($str) : converts the first character of each word in a string to uppercase

String Functions

9. trim($str) : Removes whitespace from both sides of a string.
10. ltrim($str): Removes whitespace from the left side of a string.
11. rtrim($str): Removes whitespace from the right side of a string.
12. strcmp($str1, $str2): Compares two strings and return 0 if same, -1 if $str1<$str2, 1 if $str1>$str2.
13. strtolower($str): converts a string to lowercase
14. strtoupper($str): converts a string to uppercase
15. substr($str,$start[,$length]): Returns a substring of a string.
16. str_repeat($str,$number) : Repeats a string to specified number of times.
17. strstr($str,$search[,$before_search]): Searches for the first occurrence of a string inside another string and returns rest of the part. Default value for before_search is false(before_search is true or false)

String Functions

18. str_split($str[,$length]) : Splits string into an array.
19. join(separator,array) : It returns a string from the elements of an array.
20. ctype_lower($str): Returns true if string is in lowercase else returns false.
21. ctype_upper($str): Returns true if string is in uppercase else returns false.
22. ctype_alpha($str): Returns true if $str is alphabet else returns false.
23. str_pad(string, length, pad_string[,pad_type]) : Attach a string to a new length. where pad_type = STR_PAD_BOTH, STR_PAD_LEFT, STR_PAD_RIGHT
    Default value for pad_type is STR_PAD_RIGHT.

Thank you