PHP function

Functions

- A function is a block of code that has a name and it has a property that it is reusable.
- Function groups a number of script statements into a unit and gives it a name.
- To keep the script from being executed when the page loads, you can put it into a function.
- A function will be executed by a call to the function.
- function can be called from anywhere within a page.

Functions

- PHP has a wide api for built-in functions.
- **PHP User Defined Functions**
    - Besides the built-in PHP functions, we can create our own functions.
    - 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.

Syntax for user defined function

-
  ```
  function functionName($param1, $param2,…)
  {
  code to be executed;
  return $value;
   }
  ```
- Example :-

```
 <?php
   function writeName()
   {
        echo “ABC";
   }
   echo "My name is ";
   writeName();
?>
```

```
Example : with parameter
<?php
function writeName($fname){
    echo $fname . "Paul.<br />";
}
 echo "My name is ";
  writeName("Jim");
  echo "My sister's name is ";
   writeName("Agnes");
  echo "My brother's name is ";
     writeName("Steve");

?>
```

A function name can start with a letter or underscore (not a number). Function names are NOT case-sensitive.

Function Arguments

- Information can be passed to functions through arguments. An argument is just like a variable.
- Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.
- **Example**

```
<?php
  function familyName($fname){
    echo "$fname Sharma.<br>";
  }
  familyName("Raj");
  familyName("Kavita");
  familyName("Pooja");
  familyName("Suyash");
  familyName("Vivaan");
?>
```

Default Argument Value

Example

```
<?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);
?>
```

Functions - Returning values

Example

```
<?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);
?>
```

https://wiki.php.net/rfc/skipparams

Note: As of PHP 7.1.0, omitting a parameter which does not specify a default throws an ArgumentCountError; in previous versions it raised a Warning.

PHP 8.0 - Functions

- PHP supports
  - passing arguments by value (the default),
  - passing by reference, and
  - default argument values.
  - Variable-length argument lists and
  - Named Arguments

Example # Function Argument List with trailing Comma

```php
<?php
  function takes_many_args(
      $first_arg,
      $second_arg,
      $a_very_long_argument_name,
      $arg_with_default = 5,
     $again = 'a default string',
  )
?>
```

```php
<?php
    class DCMaker {
        public function brew() {
            return 'Making coffee.';
        }
    }
    function makecoffee($coffeeMaker = new DCMaker)
    {
        return $coffeeMaker->brew();
    }
    echo makecoffee();
?>
```

Variable-length argument lists

- PHP has support for variable-length argument lists in user-defined functions by using the ... token.
- Note: It is also possible to achieve variable-length arguments by using *func_num_args()*, *func_get_arg()*, and *func_get_args()* functions. This technique is not recommended as it was used prior to the introduction of the ... token.
- Argument lists may include the ... token to denote that the function accepts a variable number of arguments. The arguments will be passed into the given variable as an array:

```
<?php
function foo()
{
          echo   "Number   of   arguments:   ",
func_num_args(), PHP_EOL;
}

foo(1, 2, 3);
?>
```

```php
<?php
function helloWorld($ArgA, $ArgB="HelloWorld!") {
  return func_num_args();
}

$Returns = helloWorld("HelloWorld!");

$Returns = helloWorld("Hello", "World!");
?>
```

```php
<?php
function foo()
{
      $numargs = func_num_args();
      echo "Number of arguments: $numargs\n";
      if ($numargs >= 2) {
         echo "Second argument is: " . func_get_arg(1)
. "\n";
      }
}
foo(1, 2, 3);
?>
```

```php
<?php
function byRef(&$arg) {
    echo 'As passed      : ',
var_export(func_get_arg(0)), PHP_EOL;
    $arg = 'baz';
    echo 'After change   : ',
var_export(func_get_arg(0)), PHP_EOL;
}
$arg = 'bar';
byRef($arg);
?>
```

```
<?php
function foo()
{
    $numargs = func_num_args();
    echo "Number of arguments: $numargs \n";
    if ($numargs >= 2) {
        echo "Second argument is: " . func_get_arg(1) . "\n";
    }
    $arg_list = func_get_args();
    for ($i = 0; $i < $numargs; $i++) {
        echo "Argument $i is: " . $arg_list[$i] . "\n";
    }
}
foo(1, 2, 3);
?>
```

```php
<?php
#Using ... to access variable arguments
function sum(...$numbers) {
    $acc = 0;
    foreach ($numbers as $n) {
        $acc += $n;
    }
    return $acc;
}

echo sum(1, 2, 3, 4);
?>
```

```php
<?php
#Using ... to provide arguments
function add($a, $b) {
    return $a + $b;
}

echo add(...[1, 2])."\n";

$a = [1, 2];
echo add(...$a);
?>
```

Named Arguments

- PHP 8.0.0 introduced named arguments as an extension of the existing positional parameters.
- Named arguments allow passing arguments to a function based on the parameter name, rather than the parameter position.
- This makes the meaning of the argument self-documenting, makes the arguments order-independent and allows skipping default values arbitrarily.
- Named arguments are passed by prefixing the value with the parameter name followed by a colon.
- Using reserved keywords as parameter names is allowed. The parameter name must be an identifier, specifying dynamically is not allowed.

```
#Positional arguments versus named arguments
<?php
// Using positional arguments:
array_fill(0, 100, 50);

// Using named arguments:
array_fill(start_index: 0, count: 100, value:
50);
?>
```

```
#Error thrown when passing the same parameter
#multiple times
<?php
function foo($param) { ... }
foo(param: 1, param: 2);
// Error: Named parameter $param overwrites
previous argument
foo(1, param: 2);
// Error: Named parameter $param overwrites
previous argument
?>
```

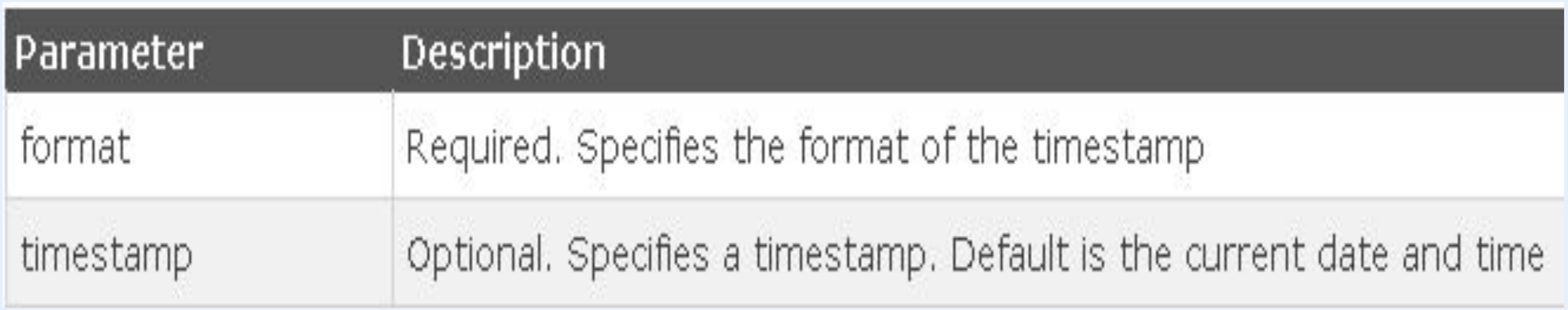
```php
<?php
myFunction(paramName: $value);
array_foobar(array: $value);

// NOT supported.
function_name($variableStoringParamName
: $value);
?>
```

```php
<?php
function foo($a, $b, $c = 3, $d = 4) {
  return $a + $b + $c + $d;
}

var_dump(foo(...[1, 2], d: 40));

var_dump(foo(...['b' => 2, 'a' => 1], d: 40));

?>
```

https://www.php.net/manual/en/functions.arguments.php

The Date() Function

- 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

Cont...

- **Get a Simple Date**
  - 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.

Cont…

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");
?>
```

Get a Simple Time

- Here are some characters that are commonly used for times:
    - 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)
- The example below outputs the current time in the specified format:
- **Example**

```
<?php
    echo "The time is " . date("h:i:s a");
?>
```

Get Your Time Zone

- So, if you need the time to be correct according to a specific location, you can set a time-zone to use.
- The example below sets the time-zone to "Asia/Calcutta", then outputs the current time in the specified format:
- **Example**

```
<?php
    date_default_timezone_set("Asia/Calcutta");
    echo "The time is " . date("h:i:sa");
 ?>
```

String Functions

substr() :-This function returns the part of the string as an output.

Syntax : -

```
    string substr(<string s>,<int start>,[<int length>]);
```

- s: mandatory parameter. The string from which the part is to be extracted is mentioned here.
- Start : The start in the string from which the characters are to be extracted
  - Positive number – Start at a specified position in the string
  - Negative number – Start at a specified position from the end of the string
  - 0 – Start at the first character in string
- Length : It is an optional parameter. It specifies the length of the string which is to be extracted.
  - Positive number – The length to be returned from the start parameter
  - Negative number – The length to be returned from the end of the string

String Functions

- Examples:-

```
<?php echo substr("Hello world",6); ?> //Returns world

<?php echo substr("Hello world",6,4); ?> // Returns worl

<?php echo substr("Hello world", -1); ?> // Returns  d

<?php echo substr("Hello world", -3, -1);?> // Returns rl
```

String Functions

strlen() :-This function returns the length of the string.

Syntax :-

```
int strlen(<string s>);
```

- string s: It is mandatory field. The string whose length is to be found out is mentioned here.

Example :-

```
<?php echo strlen("Hello world");  ?> // Returns 11
```

String Functions

trim() **:-**This function removes the whitespaces from both start and the end of the string.

Syntax :-

```
string trim(<string s>);
```

ltrim() **:-** This function removes the whitespaces from the left part of the string.

Syntax:-

```
string ltrim(<string s>);
```

- string s: It is mandatory field. The string of which the white spaces are to be removed is passed as parameter.

Example :-

```
<?php echo trim( "       Hello World    "); ?>//Hello World
<?php echo ltrim( "       Hello World    "); ?>  // returns Hello World
```

String functions

rtrim():- This function removes the whitespaces from the right part of the string.

Syntax :-

```
string rtrim(<string s>);
```

- string s: It is mandatory field. The string of which the whitespaces are to be removed from right side is passed as parameter.

Example :-

```
<?php echo rtrim(" Hello World  "); ?>// Hello World
```

strtolower():- This function converts the string to lower case

Syntax :

```
string strtolower(<string s>);
```

String s : It is mandatory field. The string which is to be converted to lower case is passed here.

Example :-

```
<?php echo strtolower("HELLO WORLD"); ?> // Returns hello world
```

String functions

strtoupper():- This function converts the string to upper case

Syntax :

```
strtoupper(<string s>);
```

- string s: It is mandatory field. The string which is to be converted to upper case is passed here.

Example :-

```
<?php echo strtoupper("hello world"); ?>
// Returns HELLO WORLD
```

String functions

strcmp():- The strcmp() function compares two strings.
This function returns:

- 0 – if the two strings are equal
- <0 – if string1 is less than string2
- >0 – if string1 is greater than string2

Syntax :

```
strcmp(<string1>,<string2>);
```

- String1 and String 2 are mandatory.

Example :-

```
<?php echo strcmp("Hello world!","Hello world!"); ?>
// 0
```

String functions

Reverse a String

- **strrev()** function reverses a string. E.g.

```
<?php
echo strrev("Hello world!"); // outputs !dlrow olleH
?>
```

String functions

Search For a Specific Text Within a String

- **strpos()** 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.

Syntax :

```
strpos(  string, find, start)
```

- String : specifies the string to search
- Find : Specifies the string to find
- Start : Specifies where to begin the search. If *start* is a negative number, it counts from the end of the string.

Cont…

E.g.

```
<?php
echo strpos("Hello world!", "world"); // outputs 6
?>
```

```
<?php
$str = “This is string of the string into the string.”
echo strpos( $str, “string“ , 10 ); // outputs 22
?>
```

Cont...

Replace Text Within a String

- **str_replace()** function replaces some characters with some other characters in a string.

E.g.

```
<?php
   echo str_replace("world", "Dolly", "Hello world!");
   // outputs Hello Dolly!
?>
```

Use of Heredoc in PHP

- Heredoc is one of the ways to store or print a block of text in PHP.
- The data stored in the heredoc variable is more readable and error-free than other variables for using indentation and newline.
- The following steps need to follow to store or print the heredoc document.
    - ‘<<<’ is used to start the heredoc document.
    - A delimiter is required to use after ‘<<<‘ to define the starting of the document and the same delimiter name with a semicolon(;) is used at the end of the heredoc document to define the end of the document.

```
<?php
    //Printing heredoc content
    print <<< HERE
    PHP is a general-purpose scripting language especially suited to web
    development.
    It was created by Danish-Canadian programmer Rasmus Lerdorf in 1994.
    The PHP reference implementation is now produced by The PHP Group.
    HERE;
    //Print the second heredoc document
    print <<< DOC
    <pre>
    www.google.com
    www.bing.com
    www.ddu.ac.in
    www.yahoo.com
    </pre>
    DOC;
?>
```

```
<?php   #Using heredoc content in a variable
    //Define a string variable
    $name = 'Himanshu Purohit';
    //Define a heredoc variable
    $address = <<< addr
    <pre>
        MCA Department, D.D.University,
        Nadiad 387001.
    </pre>
    addr;
    //Define another string variable
    $phone = '9999999999';
    //Print the variables
    echo "Name : <br/> <pre>    $name </pre>". "Address : $address".
    "Phone : <pre>
    $phone</pre>";
?>
```

```
<?php    #Displaying HTML form using heredoc variable
//Define the login form
$form = <<< HTML
    <form action="#" method="post">
        <input type="text" name="username" /><br/> <br/>
        <input type="password" name="password" /><br/> <br/>
        <input type="submit" name="submit" value="Submit" />
    </form>
html;

echo "<h3>Login Form</h3>";
//Display the login form
echo $form;
//Check the submit button is clicked or not
if(isset($_POST['submit']))
{
        //Check the validity of the user
        if($_POST['username'] == 'admin' && $_POST['password'] == 'secret'){
            echo "Authenticated user";
        }
        else{
            echo "Username or password is wrong.";
        }
}
?>
```

```
<?php   #Using the variable inside the heredoc content
    //Declare a variable with string value
    $website = 'DDU MCA';
    //Use variable in the heredoc content
    $var = <<<here
    $website is a popular blog site.
    here;
    //Print the heredoc variable
    echo "<h2 style='color:blue'>". $var ."</h2>";
?>
```

```
<?php #Using heredoc variable inside the function
   //Define a user-defined function
   function display($book,$author)
   {
       //Use the argument values inside the heredoc content
       print <<<book
   <pre>
       Book Name: $book <br/>
       Author Name: $author <br/>
       Publisher: O'Reilly
   </pre>
   book;
   }
   //Call the function
   display("Head  First  PHP  &  MySQL","Lynn  Beighley  and
   Michael Morrison");
?>
```