**Lesson: 10 (User Define & Building Functions, Get & Post Variable)**

**PHP Functions**

The real power of PHP comes from its functions. In PHP, there are more than 700 built-in functions.

***Syntax***

|  |
| --- |
| function functionName()  {  code to be executed;  } |

***PHP function guidelines:***

* Give the function a name that reflects what the function does
* The function name can start with a letter or underscore (not a number)

***Adding parameters:*** To add more functionality to a function, we can add parameters. A parameter is just like a variable. Parameters are specified after the function name, inside the parentheses.

***Note: Example One***

***Return values:*** To let a function return a value, use the return statement.

***Note: Example Two***

## PHP Array Functions

**PHP**: indicates the earliest version of PHP that supports the function.

|  |  |
| --- | --- |
| **Function** | **Description** |
| array() | Creates an array |
| array\_change\_key\_case() | Returns an array with all keys in lowercase or uppercase |
| array\_chunk() | Splits an array into chunks of arrays |
| array\_combine() | Creates an array by using one array for keys and another for its values |
| array\_count\_values() | Returns an array with the number of occurrences for each value |
| array\_diff() | Compares array values, and returns the differences |
| array\_diff\_assoc() | Compares array keys and values, and returns the differences |
| array\_diff\_key() | Compares array keys, and returns the differences |
| array\_diff\_uassoc() | Compares array keys and values, with an additional user-made function check, and returns the differences |
| array\_diff\_ukey() | Compares array keys, with an additional user-made function check, and returns the differences |
| array\_fill() | Fills an array with values |
| array\_filter() | Filters elements of an array using a user-made function |
| array\_flip() | Exchanges all keys with their associated values in an array |
| array\_intersect() | Compares array values, and returns the matches |
| array\_intersect\_assoc() | Compares array keys and values, and returns the matches |
| array\_intersect\_key() | Compares array keys, and returns the matches |
| array\_intersect\_uassoc() | Compares array keys and values, with an additional user-made function check, and returns the matches |
| array\_intersect\_ukey() | Compares array keys, with an additional user-made function check, and returns the matches |
| array\_key\_exists() | Checks if the specified key exists in the array |
| array\_keys() | Returns all the keys of an array |
| array\_map() | Sends each value of an array to a user-made function, which returns new values |
| array\_merge() | Merges one or more arrays into one array |
| array\_merge\_recursive() | Merges one or more arrays into one array |
| array\_multisort() | Sorts multiple or multi-dimensional arrays |
| array\_pad() | Inserts a specified number of items, with a specified value, to an array |
| array\_pop() | Deletes the last element of an array |
| array\_product() | Calculates the product of the values in an array |
| array\_push() | Inserts one or more elements to the end of an array |
| array\_rand() | Returns one or more random keys from an array |
| array\_reduce() | Returns an array as a string, using a user-defined function |
| array\_reverse() | Returns an array in the reverse order |
| array\_search() | Searches an array for a given value and returns the key |
| array\_shift() | Removes the first element from an array, and returns the value of the removed element |
| array\_slice() | Returns selected parts of an array |
| array\_splice() | Removes and replaces specified elements of an array |
| array\_sum() | Returns the sum of the values in an array |
| array\_udiff() | Compares array values in a user-made function and returns an array |
| array\_udiff\_assoc() | Compares array keys, and compares array values in a user-made function, and returns an array |
| array\_udiff\_uassoc() | Compares array keys and array values in user-made functions, and returns an array |
| array\_uintersect() | Compares array values in a user-made function and returns an array |
| array\_uintersect\_assoc() | Compares array keys, and compares array values in a user-made function, and returns an array |
| array\_uintersect\_uassoc() | Compares array keys and array values in user-made functions, and returns an array |
| array\_unique() | Removes duplicate values from an array |
| array\_unshift() | Adds one or more elements to the beginning of an array |
| array\_values() | Returns all the values of an array |
| array\_walk() | Applies a user function to every member of an array |
| array\_walk\_recursive() | Applies a user function recursively to every member of an array |
| arsort() | Sorts an array in reverse order and maintain index association |
| asort() | Sorts an array and maintain index association |
| compact() | Create array containing variables and their values |
| count() | Counts elements in an array, or properties in an object |
| current() | Returns the current element in an array |
| each() | Returns the current key and value pair from an array |
| end() | Sets the internal pointer of an array to its last element |
| extract() | Imports variables into the current symbol table from an array |
| in\_array() | Checks if a specified value exists in an array |
| key() | Fetches a key from an array |
| krsort() | Sorts an array by key in reverse order |
| ksort() | Sorts an array by key |
| list() | Assigns variables as if they were an array |
| natcasesort() | Sorts an array using a case insensitive "natural order" algorithm |
| natsort() | Sorts an array using a "natural order" algorithm |
| next() | Advance the internal array pointer of an array |
| pos() | Alias of current() |
| prev() | Rewinds the internal array pointer |
| range() | Creates an array containing a range of elements |
| reset() | Sets the internal pointer of an array to its first element |
| rsort() | Sorts an array in reverse order |
| shuffle() | Shuffles an array |
| sizeof() | Alias of count() |
| sort() | Sorts an array |
| uasort() | Sorts an array with a user-defined function and maintain index association |
| uksort() | Sorts an array by keys using a user-defined function |
| usort() | Sorts an array by values using a user-defined function |

**Date**

The PHP date() function is used to format a time and/or date.

***Syntax:***

date(format,timestamp)

***Format the Date:*** The required format parameter in the date() function specifies how to format the date/time.

* d - Represents the day of the month (01 to 31)
* m - Represents a month (01 to 12)
* Y - Represents a year (in four digits)

***Note: Example Three***

***Adding a Timestamp:*** The optional timestamp parameter in the date() function specifies a timestamp. If you do not specify a timestamp, the current date and time will be used.

***Syntax for mktime():***

mktime(hour,minute,second,month,day,year,is\_dst)

***Note: Example Four***

**Include**

***Server Side Includes (SSI):*** You can insert the content of one PHP file into another PHP file before the server executes it, with the include() or require() function.

The two functions are identical in every way, except how they handle errors:

* include() generates a warning, but the script will continue execution
* require() generates a fatal error, and the script will stop

These two functions are used to create functions, headers, footers, or elements that will be reused on multiple pages.

***include() Function:*** The include() function takes all the content in a specified file and includes it in the current file. If an error occurs, the include() function generates a warning, but the script will continue execution.

***require() Function:*** The require() function is identical to include(), except that it handles errors differently. If an error occurs, the include() function generates a warning, but the script will continue execution. The require() generates a fatal error, and the script will stop.

***Note: Example Five***

**PHP Forms**

The PHP $\_GET and $\_POST variables are used to retrieve information from forms, like user input.

***Note: Example Six***

**$\_GET Function**

The built-in $\_GET function is used to collect values in a form with method="get".Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and has limits on the amount of information to send (max. 100 characters).

***User of $\_GET Function:***

* When using method="get" in HTML forms, all variable names and values are displayed in the URL.
* This method should not be used when sending passwords or other sensitive information!
* Because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.
* The get method is not suitable for large variable values; the value cannot exceed 100 characters.

***Note: Example Seven***

**$\_POST Function**

The built-in $\_POST function is used to collect values from a form sent with method="post". Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.

***Note: Example Six***

**Difference between PHP 4 & PHP 5**

There are several differences between PHP4 and PHP5.

1. Unified constructor and Destructor.
2. Exception has been introduced.
3. New error level named E\_STRICT has been introduced.
4. Now we can define full method definitions for a abstract
5. Within a class we can define class constants.
6. we can use the final keyword to indicate that a method cannot be overridden by a child
7. Public, private and protected method introduced

|  |  |
| --- | --- |
| **<!--Example One-->**  <html>  <body>  <?php  function writeName1()  {  echo "Md. Saahil Alam Talha<br />";  }  function writeName($fname)  {  echo $fname . " Alam.<br />";  }  echo "My name is ";  writeName1();  echo "My name is ";  writeName("Md. Saahil");  echo "My Father's name is "; writeName("Md. Shafiul");  echo "My Mother's name is "; writeName("Tania");  ?>  </body>  </html> | **<!--Example Two-->**  <html>  <head>  </head>  <body>  <?php  function add($x,$y)  {  $total=$x+$y;  return $total;  }  echo "1 + 16 = " . add(1,16);  ?>  </body>  </html> |
| **<!--Example Three-->**  <html>  </head>  <body>  <?php  echo date("Y/m/d") . "<br />";  echo date("Y.m.d") . "<br />";  echo date("Y-m-d")  ?>  </body>  </html> | **<!--Example Four-->**  <html>  </head>  <body>  <?php  $tomorrow = mktime(0,0,0,date("m"),date("d")+1,date("Y"));  echo "Tomorrow is ".date("Y/m/d", $tomorrow);  ?>  </body>  </html> |
| **menu.php**  <html>  </head>  <body>  <a href="/default.php">Home</a>  <a href="/tutorials.php">Tutorials</a>  <a href="/references.php">References</a>  <a href="/examples.php">Examples</a>  <a href="/about.php">About Us</a>  <a href="/contact.php">Contact Us</a>  </body>  </html> | **<!--Example Three-->**  <html>  <body>  <div class="leftmenu">  <?php include("menu.php"); ?>  </div>  <h1>Welcome to my home page.</h1>  <p>Some text.</p>  </body>  </html> |
| **<!--Example Six-->**  <html>  <body>  <form action="welcome.php" method="post">  Name: <input type="text" name="fname" />  Age: <input type="text" name="age" />  <input type="submit" />  </form>  </body>  </html> | **<!--Example Seven-->**  <form action="welcome.php" method="get">  Name: <input type="text" name="fname" />  Age: <input type="text" name="age" />  <input type="submit" />  </form> |
| **welcome.php**  <html>  <body>  Welcome <?php echo $\_POST["fname"]; ?>!<br />  You are <?php echo $\_POST["age"]; ?> years old.  </body>  </html> |  |