Lesson: 07 (Array, Functions, Get & Post Variable)

Arrays

An array is a special variable, which can hold more than one value, at a time. If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1="Saab";
$cars2="Volvo";
$cars3="BMW";
```

An array can hold all your variable values under a single name. And you can access the values by referring to the array name. Each element in the array has its own index so that it can be easily accessed. In PHP, there are three kinds of arrays:

- Numeric array - An array with a numeric index
- Associative array - An array where each ID key is associated with a value
- Multidimensional array - An array containing one or more arrays

Numeric Arrays: A numeric array stores each array element with a numeric index. There are two methods to create a numeric array.

Note: Example One

Associative Arrays: An associative array, each ID key is associated with a value.

Note: Example Two

Multidimensional Arrays: In a multidimensional array, each element in the main array can also be an array. And each element in the sub-array can be an array, and so on.

Note: Example Three

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 Four

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

Note: Example Five

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

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-->	**<!--Example Two-->**
<html>	<html>
<head>	<head>
</head>	</head>
<body>	<body>
<?php	<?php
//In the following example I assign the index	$ages['Peter'] = "32";
//manually:	$ages['Quagmire'] = "30";
$cars[0]="Saab";	$ages['Joe'] = "34";
$cars[1]="Volvo";	$ages1 = array("Peter"=>32,
$cars[2]="BMW";	"Quagmire"=>30, "Joe"=>34);
$cars[3]="Toyota";	echo "Peter is " . $ages['Peter'] . " years old.
//In the following example the index are	 ";

```
//automatically assigned (the index starts at 0):
$cars1=array("Saab","Volvo","BMW","Toyota");
echo $cars[0] . " and " . $cars[1] . " are Swedish
cars.<br/>";
echo $cars1[0] . " and " . $cars1[1] . " are
Swedish cars.";
?>
</body>
</html>
```

```
echo "Peter is " . $ages1['Peter'] . " years
old.";
?>
</body>
</html>
```

<!--Example Three-->

```
<html>
<head>
</head>
<body>
<?php
$families = array
  (
  "Shafiul"=>array
  (
          "Tania",
          "Saahil",
          "Miskat"
  ),
  "Talha"=>array
  (
          "Ammuni & Dada"
  ),
  "Tania"=>array
  (
          "Shafiul",
          "Talha"
  )
  );
 echo "Is " . $families['Shafiul'][1] . " a part of the
Shafiul family?";
 ?>
</body>
</html>
```

<!--Example Four-->

```
<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 Five-->

```
<html>
<head>
</head>
<body>
<?php
function add($x,$y)
{
```

<!--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" />
```

```
$total=$x+$y;
return $total;
}
echo "1 + 16 = " . add(1,16);
?>
</body>
</html>
```

```
</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>
```