PHP Functions

1. Functions
    I. User-defined Functions
    II. Function Arguments
        a) Parameters by reference
        b) Returning values
        c) Default argument values
    III. Returning values
    IV. Variable Functions
    V. Anonymous Functions
2. Arrays
    I. Indexed Arrays
        a) Assigning Manually
        b) Assigning Automatically
    II. Associative Arrays
    III. Multidimensional Arrays
3. Array Sort Functions
    I. sort()
    II. rsort()
    III. asort()
    IV. ksort()
    V. arsort()
    VI. krsort()
4. Super-Global Variables
    I. $GLOBALS
    II. $_SERVER
    III. $_REQUEST
    IV. $_POST
    V. $_GET
    VI. $_FILES $_ENV
    VII. $_COOKIE
    VIII. $_SESSION
5. Get and Post method
6. Get vs Post

Functions

Like other programming languages PHP also supports functions. A function is just a name we give to a block of code that can be executed whenever we need it. In PHP we have multiple functions.

- User-defined functions
- Function arguments
- Returning values
- Variable functions
- Anonymous functions

User-defined functions

- Like other programming languages we can create our own functions.
- Function names also have the same rules as other labels in PHP.
- A valid function name starts with a letter or underscore, followed by any number of letters, numbers, or underscores.
- User defined functions will start with the keyword "*function*".
- Function names are not case-sensitive.
  Generally we use function names based on the purpose that are created.

Syntax

```
<?php
function show($arg_1, $arg_2,...., $arg_n)
{
code that we need to execute;
}
?>
```

Example

```
<?php
footer(); // call the function
function footer() {
   echo "Copyrights @ Sakshi Education";
}
?>
```

Output

Copyrights @ Sakshi Education

Function Arguments

Information can be passed to functions through the argument list, the arguments are evaluated from left to right. PHP supports passing arguments by value, passing by reference, and default argument values.

Example

```
<?php
function schoolName($sname)
{
 echo "$snameHigh School.<br>";
}

schoolName("YSR");
schoolName("DPS");
schoolName("Hyderabad");
?>
```

Parameters by reference

By default, function arguments are passed by value, if the value of the argument within the function is changed, it does not get changed outside of the function. To allow a function to modify its arguments, they must be passed by reference.

Example

```
<?php
function extend_text(&$extra)
{
   $extra .= 'text added now.';
}
$str = 'It is a Stirin, ';
extend_text($str);
echo $str;
?>
```

Outputs

it is a string, text added now.

Default argument values

Example

```
<?php
function driveBike($type = "gare")
{
echo "Drive bike with $type.\n";
}
driveBike();
driveBike(null);
driveBike("3 tyres");
?>
```

Output

Drive bike with gare.

Drive bike with .

Drive bike with 3 tyres.

Returning values

Values are returned by using the optional return statement. Any type may be returned, including arrays and objects.

Example

```
<?php
function mul($num)
{
   return $num * $num;
}
echo mul(4);
?>
```

Output

16

Returning an array to get multiple values

Example

```
<?php
function numbers()
```

```
{
   return array (0, 1, 2);
}
list ($zero, $one, $two) = numbers();
?>
```

Variable Functions

PHP supports the concept of variable functions. This means that if a variable name has parentheses appended to it, PHP will look for a function with the same name as whatever the variable evaluates to, and will attempt to execute it.

Example

```
<?php
function check() {
   echo "In check()<br />\n";
}
$task = check';
$task();       // This calls check()
?>
```

Output

In check()

Anonymous functions

Anonymous functions, also known as closures, allow the creation of functions which have no specified name. They are most useful as the value of call back parameters.

Example

```
<?php
$sakshi = function($name)
{
   echo “Learn with $name ”;
};

$sakshi('sakshi education');
$ sakshi (‘sakshi post');
?>
```

Output

Learn with sakshi education

Learn with sakshi post

Arrays

Like all other programming languages an array is a special variable, which can hold more than one value at a time.

Example

```
<?php
$array = array("Samsung", "Nokia", "Motorola");
echo “Mobiles in our shop are ”.$array[0].”, ”. $array[0].” and ”.
$array[0].”;
?>
```

Output

Mobiles in our shop are Samsung, Nokia and Motorola

In PHP there are three types of arrays

- Indexed arrays - Arrays with index
- Associative arrays - Arrays with keys
- Multidimensional arrays - Arrays inside arrays

Indexed Arrays

Index of an array will start with “0, Indexed arrays can be created in two ways. Assigning index manually or by automatically.

Assigning Manually

```
<?php
$mobile[0] = "Samsung";
$ mobile[1] = "Nokia";
$ mobile[2] = "Tata";
?>
```

Assigning Automatically

```
<?php
$mobile = array(“Samsung”,”Nokia”,”Tata”);
?>
```

Associative Arrays

If we want to create associative arrays, the key is always required.

Syntax

```
$name_of_array =
array("key"=>"value","key"=>"value","key"=>"value",.....);
```

Example

```
<?php
$mobile = array( “one”=>“Samsung” , ”two”=> ”Nokia” , “three”=>
”Tata” );
?>
```

Multidimensional Arrays

Create arrays of arrays, known as multidimensional arrays.

Example

```
<?php
$mobile = array
 (
 array("Samsung",s4,15),
 array("Nokia",1100,4),
 array("Tata",34,12)
 );
echo “Mobilename ”.$mobile[0][0]. “ Model ”.$mobile[0][1]. “Stock
”.$mobile[0][2];
?>
```

Array Sort Functions

Here in PHP we have a list of sort functions.

- sort()
- rsort()
- asort()
- ksort()
- arsort()
- krsort()

sort() function:

By default, PHP’s sorting functions sort in accordance with the rules as specified by the English language.

Example

```
<?php
$mobile = array("Samsung", "Nokia", "Tata");
sort($mobile);
echo $mobile;
?>
```

Example

```
<?php
$value = array("45", "34", "2");
sort($value);
echo $value;
?>
```

rsort()

Sort the elements in the array in descending order.

Example

```
<?php
$mobile = array("Samsung", "Nokia", "Tata");
rsort($mobile);
echo $mobile;
?>
```

Example

```
<?php
$value = array("45", "34", "2");
rsort($value);
echo $value;
?>
```

asort()

Sort array based on values inside the array in ascending order.

Example

```
<?php
$mobile = array(“one”=>“Samsung”, “two”=> "Nokia", “three”=>
"Tata");
asort($mobile);
echo $mobile;
?>
```

Example

```
<?php
$value = array(“one”=>"45", “two”=>"34", “three”=>"2");
asort($value);
echo $value;
?>
```

ksort()

Sort array based on keys inside the array in ascending order.

Example

```
<?php
$mobile = array(“one”=>“Samsung”, “two”=> "Nokia", “three”=>
"Tata");
ksort($mobile);
echo $mobile;
?>
```

Example

```
<?php
$value = array(“one”=>"45", “two”=>"34", “three”=>"2");
ksort($value);
echo $value;
?>
```

arsort()

Sort array based on values inside the array in descending order.

Example

```
<?php
$mobile = array(“one”=>“Samsung”, “two”=> "Nokia", “three”=>
"Tata");
arsort($mobile);
echo $mobile;
?>
```

Example

```
<?php
$value = array(“one”=>"45", “two”=>"34", “three”=>"2");
arsort($value);
```

```
echo $value;
?>
```

krsort()

Sort array based on keys inside the array in descending order.

Example

```
<?php
$mobile = array(“one”=>“Samsung”, “two”=> "Nokia", “three”=>
"Tata");
krsort($mobile);
echo $mobile;
?>
```

Example

```
<?php
$value = array(“one”=>"45", “two”=>"34", “three”=>"2");
krsort($value);
echo $value;
?>
```

Super-Global Variables

In PHP we have no of Super-Global Variables, are accessible from anywhere within theexecuting script and provide you with a substantial amount of environment-specific information. Have a look on the list

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES$_ENV
- $_COOKIE
- $_SESSION

$GLOBALS

To access global variables from entire PHP script then we prefers $GLOBALS, which is the super global variable.$GLOBALS can be used instead of 'global' keyword to access variables from global scope.

Example

```
<?php
$s = 100;
$t = 25;
function sub()
{
$GLOBALS['j'] = $GLOBALS['t'] - $GLOBALS['s'];
}
sub();
echo $j;
?>
```

Output

75

$_SERVER

$_SERVER stores the details aboutheaders, paths, and script locations etc...

Example

```
<?php
        echo $_SERVER['HTTP_HOST'];
        echo $_SERVER['HTTP_USER_AGENT'];
        echo $_SERVER['PHP_SELF'];
        echo $_SERVER['HTTP_REFERER'];
        echo $_SERVER['SCRIPT_NAME'];
        echo $_SERVER['SERVER_NAME'];
?>
```

$_REQUEST

To receive data after an html form was submitted, we prefer $_REQUEST.

Example

```
<html>
<body>
<form method= “post” action= “<?php echo
$_SERVER['PHP_SELF'];?>”>
 Name: <input type= “text” name= “firstname”>
 <input type= “submit” value= “submit”>
```

```
</form>
<?php
if ($_SERVER[“REQUEST_METHOD”] == “POST”) {
    $fname = $_REQUEST[‘firstname’];
   if (empty($fname)) {
      echo “Name is empty”;
   } else {
      echo $fname;
   }
}
?>
</body>
</html>
```

$_POST

Like $_REQUEST, $_POST also used to collect form data after submitting an HTML form, we have to use the method method="post" in the form. $_POST is also used to pass variables.

Example

```
<html>
<body>
<form method= “post” action= “<?php echo
$_SERVER['PHP_SELF'];?>”>
 Name: <input type= “text” name= “firstname”>
 <input type= “submit” value= “submit”>
</form>
<?php
if ($_SERVER[“REQUEST_METHOD”] == “POST”) {
    $fname = $_POST[‘firstname’];
   if (empty($fname)) {
      echo “Name is empty”;
   } else {
      echo $fname;
   }
}
?>
```

```
</body>
</html>
```

$_POST

Like $_REQUEST, $_GET also used to collect form data after submitting an HTML form, we have to use the method method="get" in the form. $_GET also collects data from url.

Example

```
<html>
<body>
<form method= “get” action= “<?php echo
$_SERVER['PHP_SELF'];?>”>
 Name: <input type= “text” name= “firstname”>
 <input type= “submit” value= “submit”>
</form>
<?php
if ($_SERVER[“REQUEST_METHOD”] == “GET”) {
    $fname = $_GET[‘firstname’];
   if (empty($fname)) {
      echo “Name is empty”;
   } else {
      echo $fname;
   }
}
?>
</body>
</html>
```

Example 2

```
<html>
<body>
<a
href="workwith_get.php?subject=sakshi&site=sakshieducation.com">
working with $GET</a>
</body>
</html>
```

In the above example, if you click on the <a> link then the perameters subject and site will be sent to workwith_get.php and by using $_GET method you can retrive data sent by the link/url.

$_COOKIE

By Introduced $_COOKIE, $HTTP_COOKIE_VARS was deprecated. A cookie is often used to identify a user.

Example

```
<?php
   echo 'This is ' . htmlspecialchars($_COOKIE["name"]) . '!';
?>
```

$_SESSION

After introducing $_SESSION, $HTTP_SESSION_VARS was deprecated. Sessions are used to carry data from one page to another page.

GET method

- If a form is submitted by GET method the information it is sending is visible to everyone.
- All the variable names and its values shown in the URL.
- The amount of data to be sent by Get method is limited, it couldn't exceeds 2000 characters.
- Because the variables can be displayed in the URL, it can be bookmarked.
- To send Passwords and some user related information we cannot use GET method.
- Get method can be used to send some simple data.

POST method

- If a form is submitted by POST method the information it is sending is invisible.
- All the variable names and its values are secure.
- No limit on information to be sent.
- Because the variables and the names are embedded within the body of the HTTP request.

- To send Passwords and some user related information we can use POST method.

Get vs Post

- Both GET and POST create an array and the arrays carry key and value pairs, here keys are the names of the form inputs and values are data passed by the user.
- Both GET and POST are super global variables, so we can use them anywhere of our program, in any class.
- In GET method, it uses URL to pass the values and keys.
- Where as in POST method it uses HTTP Post method to send the data.

Example

```
<html>
<body>
<form action="read.php" method="get">
Full Name: <input type="text" name="name"><br>
Mobile Number: <input type="text" name="number"><br>
<input type="submit">
</form>
</body>
</html>
```

read.php

```
<html>
<body>
Hi <?php echo $_GET["name"]; ?><br>
Your mobile number: <?php echo $_GET["number"]; ?>
</body>
</html>
```

Output:

Hi Sakshi Education
Your mobile number 9642000000.