Variable Scope

H S Rana

CIT,UPES

February 19, 2014

Passing Arguments by Reference

It is possible to pass arguments to functions by reference. This means that a reference to the variable is manipulated by the function rather than a copy of the variable's value.

```
<?php
function addFive($num)
{
   $num += 5;
}
function addSix(&$num)
{
   $num += 6;
}
$orignum = 10;
addFive( &$orignum );
echo "Original Value is $orignum<br />";
addSix( $orignum );
echo "Original Value is $orignum<br />";
```

Dynamic Function Calls

It is possible to assign function names as strings to variables and then treat these variables exactly as you would the function name itself.

Dynamic Function Calls

It is possible to assign function names as strings to variables and then treat these variables exactly as you would the function name itself.

```
<?php
function sayHello()
{
   echo "Hello<br />";
}
$function_holder = "sayHello";
$function_holder();
?>
```

Variable Scope

Scope can be defined as the range of availability a variable has to the program in which it is declared. PHP variables can be one of four scope types:

- Local variables
- Function parameters
- Global variables
- Static variables

Local Variables

A variable declared in a function is considered local; that is, it can be referenced solely in that function. Any assignment outside of that function will be considered to be an entirely different variable from the one contained in the function.

Local Variables

A variable declared in a function is considered local; that is, it can be referenced solely in that function. Any assignment outside of that function will be considered to be an entirely different variable from the one contained in the function.

```
<?php
$x = 4;
function assignx () {
$x = 0;
print "\$x inside function is $x.
";
}
assignx();
print "\$x outside of function is $x.
";
?>
```

Function Parameters

Function parameters are declared after the function name and inside parentheses.

```
<?php
// multiply a value by 10 and return it to the caller
function multiply ($value) {
    $value = $value * 10;
    return $value;
}

$retval = multiply (10);
Print "Return value is $retval\n";
?>
```

Global Variables

In contrast to local variables, a global variable can be accessed in any part of the program. However, in order to be modified, a global variable must be explicitly declared to be global in the function in which it is to be modified. This is accomplished, conveniently enough, by placing the keyword GLOBAL in front of the variable that should be recognized as global. Placing this keyword in front of an already existing variable tells PHP to use the variable having that name.

```
<?php
$somevar = 15;
function addit() {
GLOBAL $somevar;
$somevar++;
print "Somevar is $somevar";
}
addit();
?>
```

Static Variables

Value of the local variable is wiped out when the function ends.A static variable will not lose its value when the function exits and will still hold that value should the function be called again.
You can declare a variable to be static simply by placing the keyword STATIC in front of the variable name.

```
<?
function keep_track() {
   STATIC $count = 0;
   $count++;
   print $count;
   print "
";
}
keep_track();
keep_track();
keep_track();
?>
```