PHP - Call by Reference

PHP uses the "call by value" mechanism, by default, for passing arguments to a function. If the arguments within the function are changed, the changes do not reflect outside of the function. To allow a function to modify its arguments, the "call by reference" mechanism must be used.

In PHP, a reference variable acts as an "alias" to the original or host variable so that both of them can read and write a single value. In other words, variables of two different names can access to the same value and they behave as if they are the same variable.

The following PHP script will help in understanding what references are. Here, **$var** is a normal string variable. We declare **$var1** as a reference to **$var**, append "&" symbol to the latter.

```
$var = "Hello";
$var1 = &$var;
```

When we say that **$var1** is an alias or reference of **$var**, it means any change in its value will also change the value of **$var**, and vice versa.

Example

The following example demonstrates how "call by reference" works in PHP –

</> Open Compiler

```
<?php
   $var = "Hello";
   $var1 = &$var;

   $var1 = "Hello World";
   echo "var=$var var1=$var1" . PHP_EOL;

   $var = "How are you?";
   echo "var=$var var1=$var1" . PHP_EOL;
?>
```

It will produce the following **output** –

```
var=Hello World var1=Hello World
var=How are you? var1=How are you?
```

Calling a PHP Function by Reference

To call a function by reference, you need to declare the formal arguments with name prefixed by "&" symbol.

```
function callref(&$arg1, &$arg2) {
   Statements;
}
```

The call to the function is just as in "call by value" method.

```
callref($x, $y);
```

When the function is invoked, **$arg1** becomes a reference to **$x** and **$arg2** becomes a reference to **$y**.

If, inside the function body, the value of **$arg1** or **$arg2** (or both) changes, it also causes the values of **$x** and **$y** to change.

Example

Let us have a look at the following example –

</> Open Compiler

```
<?php
   function  change_name(&$nm) {
      echo "Initially the name is $nm" . PHP_EOL;
      $nm = $nm."_new";
      echo "This function changes the name to $nm" . PHP_EOL;
   }

   $name = "John";
   echo "My name is $name" . PHP_EOL;
   change_name($name);
   echo "My name now is $name" . PHP_EOL;
?>
```

The variable **$name** is passed to the function **change_name()**. A reference variable **&$nm** becomes its reference variable. Any change in **$nm** is reflected in **$name** outside the function.

It will produce the following **output** –

```
My name is John
Initially the name is John
This function changes the name to John_new
My name now is John_new
```


Swapping Two Variables

In the following PHP code, we call a function by passing the arguments by value. The function attempts to swap their values.

Inside the function, their values are changed, but this swap doesn't reflect in the actual arguments after the execution of the function.

When the same function is called by passing the arguments by reference, the swap effect is reflected in the actual arguments as well.

</> Open Compiler

```php
<?php
   function  swap_value($a, $b) {
      echo "Initial values a = $a b = $b \n";
      $c = $a; $a = $b; $b = $c;
      echo "Swapped values a = $a b = $b \n";
   }

   $x = 10; $y =20;
   echo "Actual arguments x = $x y = $y \n\n";

   swap_value($x, $y);
   echo "Actual arguments do not change after the function: \n";
   echo "x = $x y = $y \n\n";

   function  swap_ref(&$a, &$b) {
```

```
      echo "Initial values a = $a b = $b \n";
      $c = $a; $a = $b; $b = $c;
      echo "Swapped values a = $a b = $b \n";
   }

   swap_ref($x, $y);
   echo "Actual arguments get changed after the function: \n";
   echo "x = $x y = $y";
?>
```

It will produce the following **output** –

```
Actual arguments x = 10 y = 20

Initial values a = 10 b = 20
Swapped values a = 20 b = 10
Actual arguments do not change after the function:
x = 10 y = 20

Initial values a = 10 b = 20
Swapped values a = 20 b = 10
Actual arguments get changed after the function:
x = 20 y = 10
```

Return by Reference

Just as a function in PHP can accept arguments by reference, it can also return a reference. To define a function that returns a reference, prefix the name of the function by "&" symbol.

Example

The following code shows the example of a function returning a reference. It returns **$x**, which is a local static variable inside **myfunction()**. Since "&" symbol is prepended to it, **$a** (the variable that stores the return value) becomes a reference to **&x**. As a result, any change in **$a** will also change the value of **$x**.

</> Open Compiler

```
<?php
   function &myfunction(){
```

```
      static $x=10;
      echo "x Inside function: $x \n";
      return $x;
   }

   $a=&myfunction();
   echo "Returned by Reference: $a \n";
   $a=$a+10;
   $a=&myfunction();
?>
```

It will produce the following **output** –

```
x Inside function: 10
Returned by Reference: 10
x Inside function: 20
```