PHP - Call by Value

By default, PHP uses the "call by value" mechanism for passing arguments to a function. When a function is called, the values of actual arguments are copied to the formal arguments of the function's definition.

During the execution of the function body, if there is any change in the value of any of the **formal arguments**, it is not reflected in the **actual arguments**.

- **Actual Arguments** − The arguments that are passed in a function call.
- **Formal Arguments** − The arguments that are declared in a function definition.

Example

Let us consider the function used in the code below −

Open Compiler

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

   $name = "John";
   echo "My name is $name \n";
   change_name($name);
   echo "My name is still $name";
?>
```

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 is still John
```

In this example, the **change_name()** function appends **_new** to the string argument passed to it. However, the value of the variable that was passed to it remains unchanged after the function's execution.

Formal arguments, in fact, behave as local variables for the function. Such a variable is accessible only inside the scope in which it is initialized. For a function, its body marked by the curly brackets "{ }" is its scope. Any variable inside this scope is not available for the code outside it. Hence, manipulation of any local variable has no effect on the world outside.

The "call by value" method is suitable for a function that uses the values passed to it. It performs certain computation and returns the result without having to change the value of parameters passed to it.

Note – Any function that performs a formula-type computation is an example of call by value.

Example

Take a look at the following example –

```
<?php
   function addFunction($num1, $num2) {
      $sum = $num1 + $num2;
      return $sum;
   }
   $x = 10;
   $y = 20;
   $num = addFunction($x, $y);
   echo "Sum of the two numbers is : $num";
?>
```

It will produce the following **output** –

```
Sum of the two numbers is : 30
```

Example

Here is another example of calling a function by passing the argument by value. The function increments the received number by 1, but that doesn't affect the variable passed to it.

</> Open Compiler

```php
<?php
   function increment($num) {
      echo "The initial value: $num \n";
      $num++;
      echo "This function increments the number by 1 to $num \n";
   }
   $x = 10;
   increment($x);
   echo "Number has not changed: $x";
?>
```

It will produce the following **output** –

```
The initial value: 10
This function increments the number by 1 to 11
Number has not changed: 10
```

PHP also supports passing the reference of variables to the function while calling. We shall discuss it in the next chapter.