References in PHP

References in PHP are a means to access the same variable content by different names.They are not like C pointers; for instance, you cannot perform pointer arithmetic using them, they are not actual memory addresses, and so on.
Instead, they are symbol table aliases. Note that in PHP, variable name and variable content are different, so the same content can have different names.

There are three basic operations performed using references:

1. Assigning by reference
2. Passing by reference
3. Returning by reference

1. Assigning by reference

- $a =& $b; - it means that $a and $b point to the same content
- **if you assign, pass, or return an undefined variable by reference, it will get created.**

```
function foo(&$var) { }

foo($a); // $a is “created” and assigned to null

$b = array();
foo($b[’b']);
var_dump(array_key_exists(’b', $b)); // bool(true)

$c = new StdClass;
foo($c->d);
var_dump(property_exists($c, ‘d’)); // bool(true)
```

- **if you assign a value to a variable with references in a foreach statement, the references are modified too.**

```
$ref = 0;
$row =& $ref;
foreach (array(1, 2, 3) as $row) {
// do something
}
echo $ref; // 3 - last element of the iterated array
```

- **while not being strictly an assignment by reference, expressions created with the language construct array() can also behave as such by prefixing & to the array element to add**

```
$a = 1;
$b = array(2, 3);
$arr = array(&$a, &$b[0], &$b[1]);
$arr[0]++; $arr[1]++; $arr[2]++;
/* $a == 2, $b == array(3, 4); */
```

- **note, however, that references inside arrays are potentially dangerous. Doing a normal (not by reference) assignment with a reference on the right side does not turn the left side into a reference, but references inside arrays are preserved in these normal assignments. This also applies to function calls where the array is passed by value**

```
/* Assignment of scalar variables */
$a = 1;
$b =& $a;
$c = $b;
$c = 7; //$c is not a reference; no change to $a or $b

/* Assignment of array variables */
$arr = array(1);
$a =& $arr[0]; //$a and $arr[0] are in the same reference set
$arr2 = $arr; //not an assignment-by-reference!
$arr2[0]++;
/* $a == 2, $arr == array(2) */
/* The contents of $arr are changed even though it’s not a reference! */
```

2. Passing By Reference

- **The second thing references do is to pass variables by reference. This is done by making a local variable in a function and a variable in the calling scope referencing the same content**

```
function foo(&$var)
{
$var++;
}

$a=5;
foo($a);
```

will make $a to be 6. This happens because in the function foo the variable $var refers to the same content as $a

- **no other expressions should be passed by reference, as the result is undefined. For example, the following examples of passing by reference are invalid**

```
function foo(&$var)
{
$var++;
}
function bar() // Note the missing &
{
$a = 5;
return $a;
}
foo(bar()); // Produces fatal error since PHP 5.0.5

foo($a = 5); // Expression, not variable
foo(5); // Produces fatal error
```

3. Returning by reference

- **Returning by reference is useful when you want to use a function to find to which variable a reference should be bound. Do not use return-by-reference to increase performance. The engine will automatically optimize this on its own. Only return references when you have a valid technical reason to do so. To return references, use this syntax**

```
class foo {
public $value = 42;
public function &getValue() {
return $this->value;
}
}

$obj = new foo;
$myValue = &$obj->getValue(); // $myValue is a reference to $obj->value, which is 42.
$obj->value = 2;
echo $myValue; // prints the new value of $obj->value, i.e. 2.
```

in this example, the property of the object returned by the getValue function would be set, not the copy, as it would be without using reference syntax

31 janvier 2011
by Enrico Besenyei