<http://code.iamkate.com/php/references-tutorial/>

‘call-time pass by reference has been removed from PHP’, so Example 1 in [http://php.net/manual/en/function.debug-zval-dump.php produces an error i.e](http://php.net/manual/en/function.debug-zval-dump.php%20produces%20an%20error%20i.e). using &$var is no longer allowed… *in this context*. You *can* still use $b = &$a (see below).

See this: <https://stackoverflow.com/questions/34764119/confusion-about-php-7-refcount>. So the stuff about debug\_zval\_dump() in the ‘iamkate’ link is no longer accurate (and neither is the PHP manual), but it still explains references clearly.

When you assign by reference like this:

$var1 =& $var2

…you’re essentially giving two names to one variable. There will be two symbol table entries ‘var1’ and ‘var2’ but only one store of the variable’s value (this zval thing). If you delete either of $var1 or $var2, the zval will still exist, but with a decremented refcount (or not, depending!) NB it doesn’t matter that one was assigned first, $var1 and $var2 have equal status.

What does this output?

$a = '1';

$b = &$a;

$b = "2$b";

echo $a.", ".$b;

Answer: 21, 21

Because: the second line makes both var names reference the same zval, containing ‘1’; then the third line assigns a new value to $b, and what that means is it changes the zval referenced by $b and hence also changes the value of $a; the new value of both $a and $b is the string ‘21’.