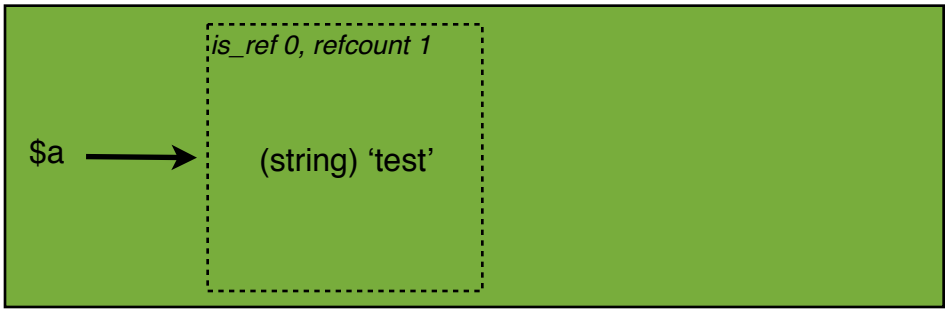
PHP5 References Explained Visually

by Monte Ohrt

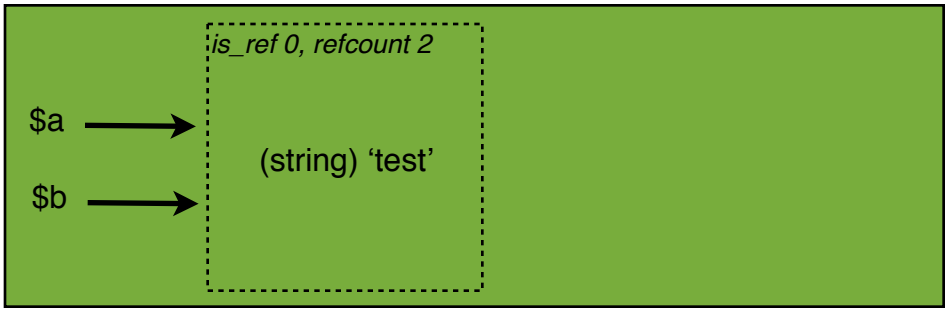
Variables

$a = 'test';



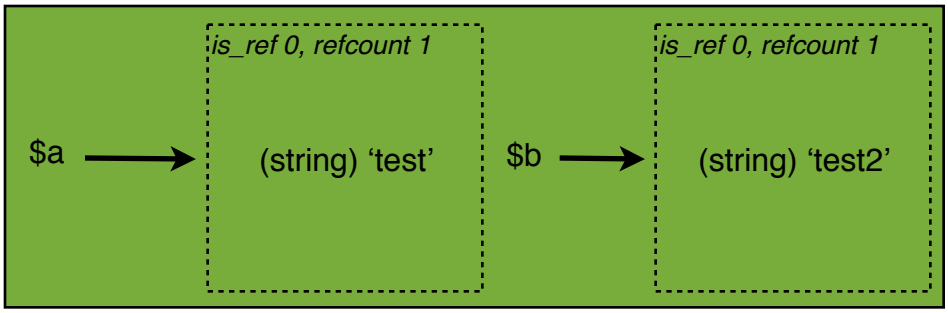
*The dotted square represents a memory container, with the string value of "test". The variable **$a** points to this container, thus the container sets a refcount of 1. (We'll get to is_ref later.)*

$b = $a;



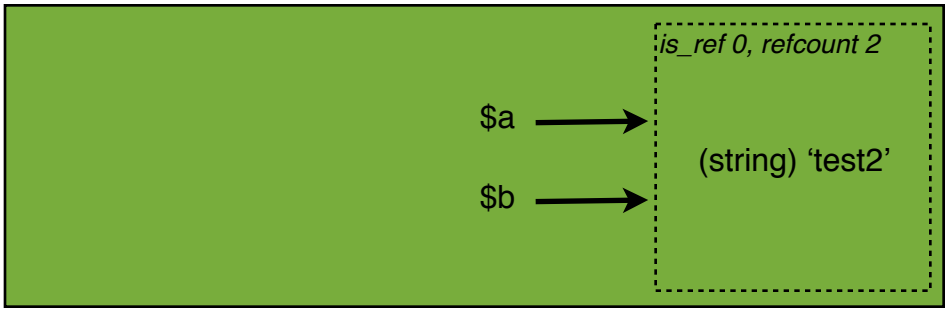
*Now **$b** also points to the same memory container, upping the refcount to 2. Conceptually **$a** and **$b** are separate variables, but as long as they are the same value, PHP uses to the same memory container. This drastically reduces the memory footprint for multiple variables of the same value.*

$b = 'test2';



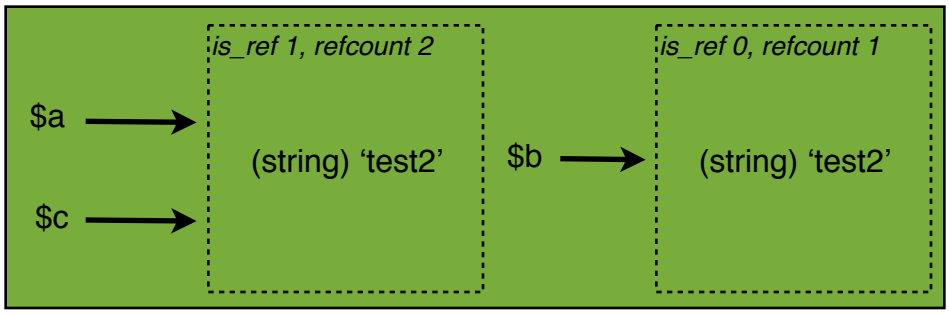
*When **$b** gets a new value set, a new memory container for that value is created, and now **$b** points to the new container. **$a** still points to the original container. The refcounts are reset to 1.*

$a = $b;



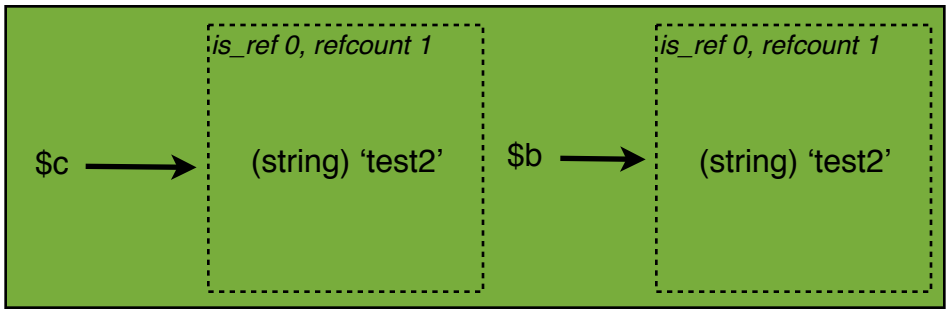
*Now **$a** points to the same memory container of **$b**. Since the original container no longer has any references to it, it gets destroyed. The refcount is upped to 2.*

$c =& $a;



*With a reference assignment (=&), is_ref is set to **1**. This tells PHP that **$a** and **$c** really are the same value conceptually, and must reflect that when one is changed. If we were to set $a = "foo", then $c would also equal "foo" and the pointer doesn't change.*

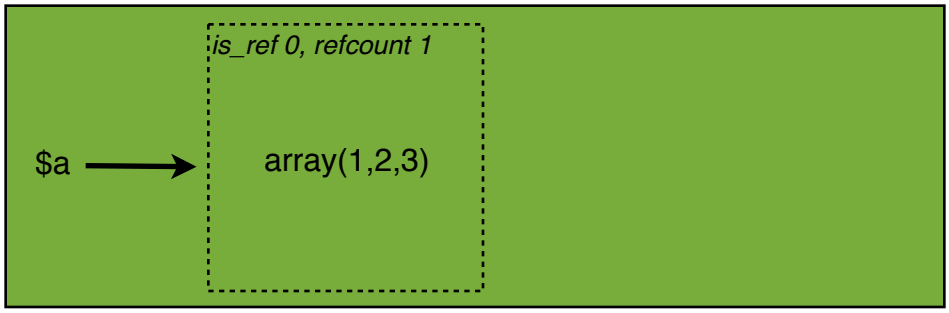
unset($a);



*If a variable is unset, it is merely unlinked from its container. Since "**test2**" still has **$c** pointing to it, it is not destroyed.*

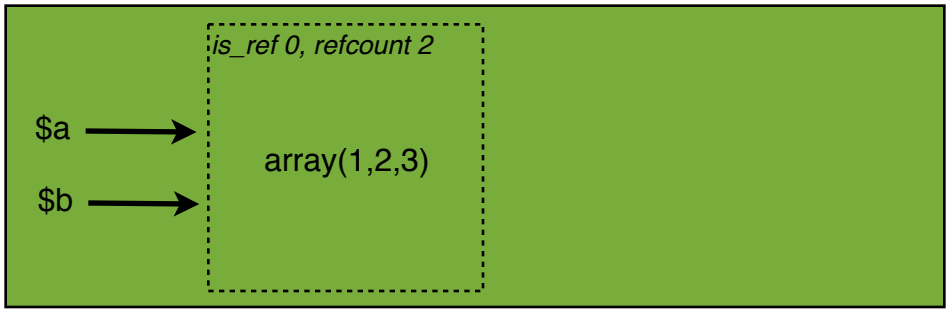
Arrays

$a = array(1,2,3);



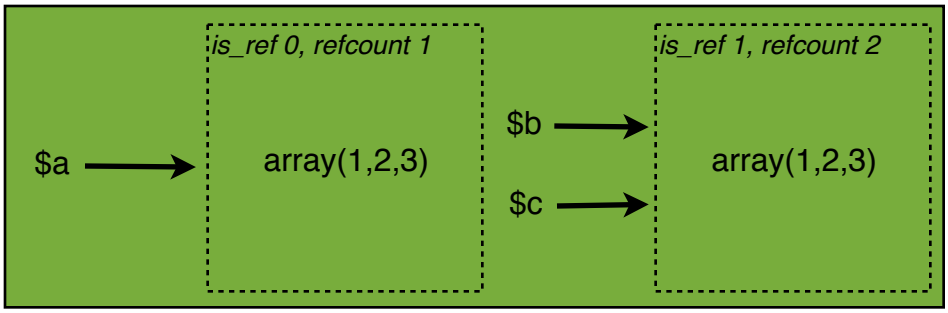
*With arrays the same basic principles as scalars still apply. Here **$a** points to the memory container of the array, and the refcount is set to **1**.*

$b = $a;



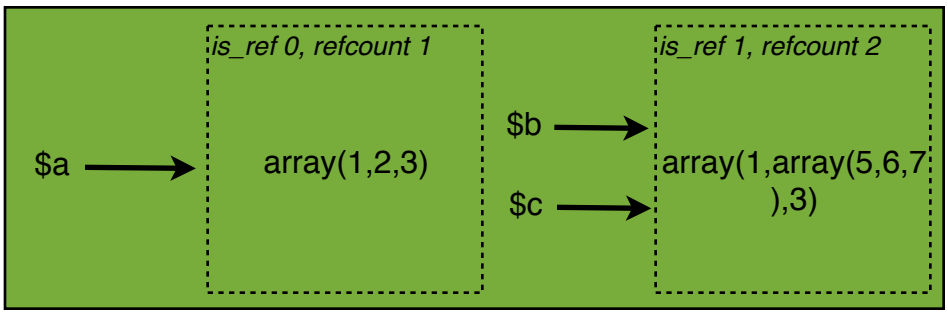
*Again, conceptually **$a** and **$b** are separate values, but PHP points to the same memory container so long as they are identical. Any change to **$a** or **$b** will result in a new memory container, same as the previous examples.*

$c =& $b;



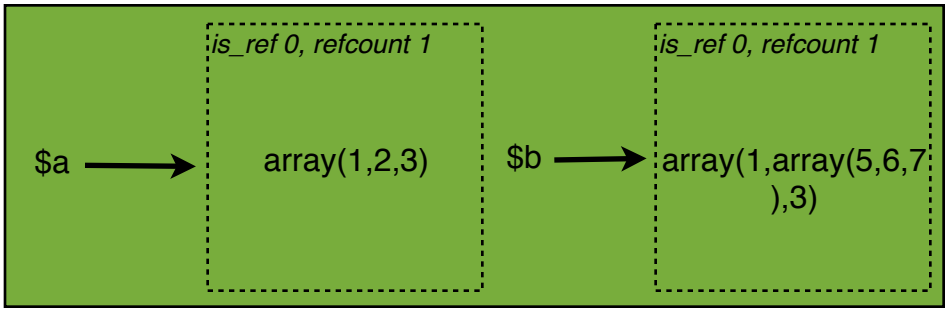
*Here we assign **$c** as a reference, so PHP must create a separate memory container for **$c** and **$b**, and is_ref is set to **1** so any change to **$b** will also happen to **$c**. **$a** remains pointing to the original container.*

$c[1] = array(5,6,7);



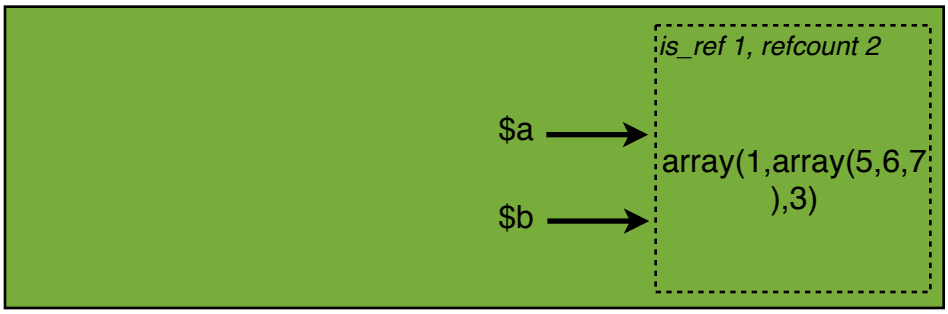
*Since is_ref=**1**, any change to **$c** is also reflected with **$b**. The pointers do not change.*

unset($c);



*again, unset causes **$c** to be unlinked from its container, and the refcount is reduced to **1**.*

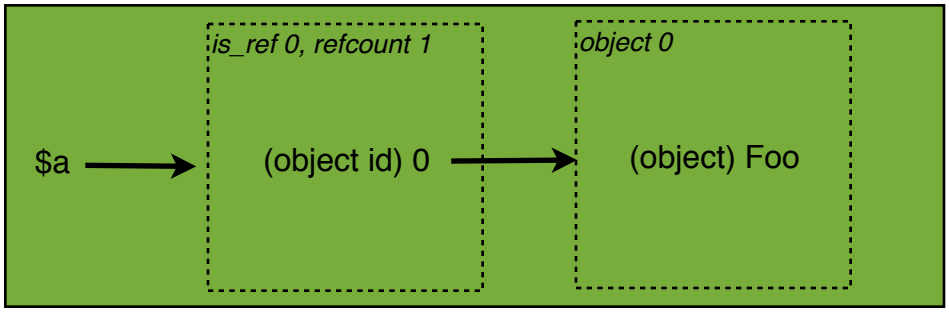
$a =& $b;



***$a** is now pointing to **$b**'s container, and is_ref is set to **1** so they will be treated as the same value. The original container no longer has a reference, so it is destroyed.*

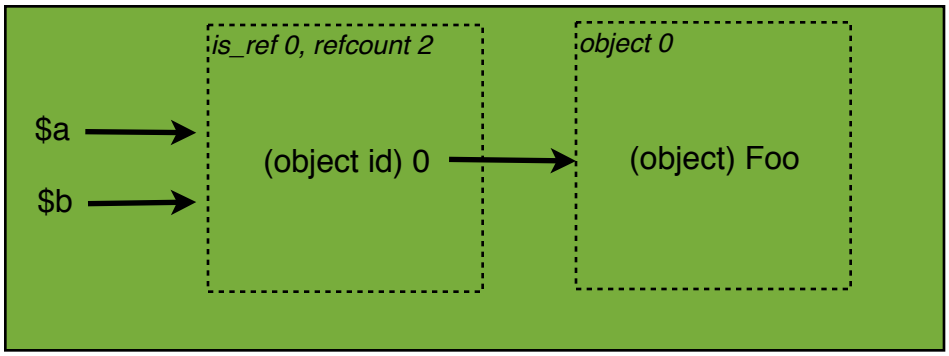
Objects

$a = new Foo;



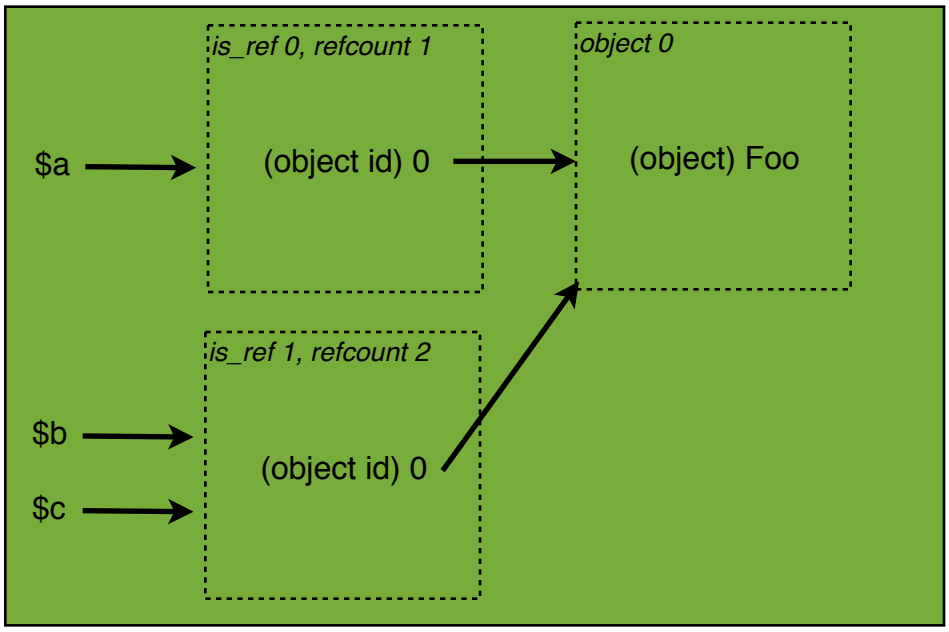
Objects behave slightly different than scalars and arrays. Instead of the variable pointing to the memory container of the object, it instead points to the memory container of an object id, which in turn points to the memory container of the object. (light bulb here.)

$b = $a;



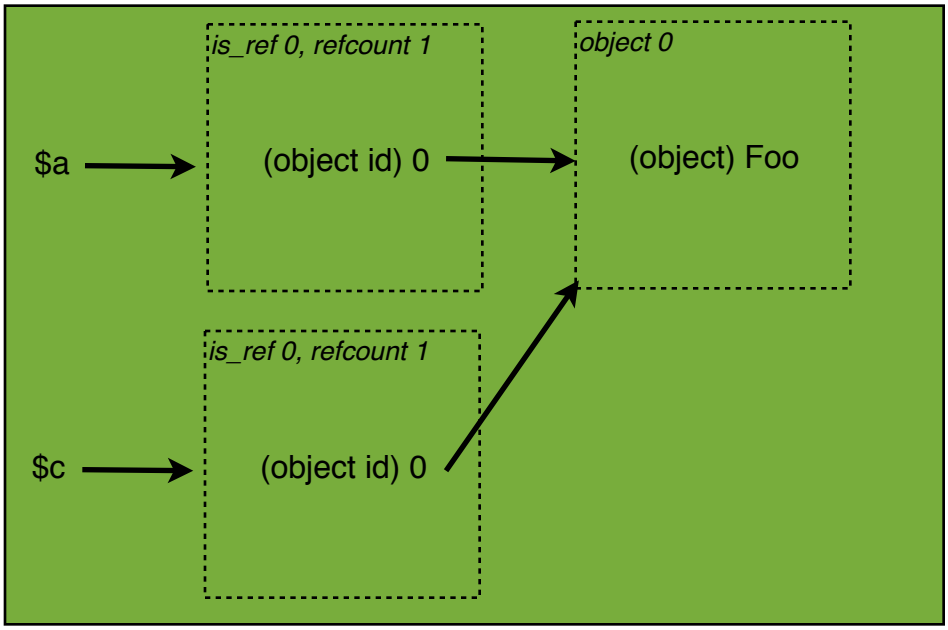
*Now you can apply the same concepts of arrays and scalars to the object id. When **$b=$a**, **$b** points to the same object id as **$a**, but they both still point to the same object container.*

$c =& $b;



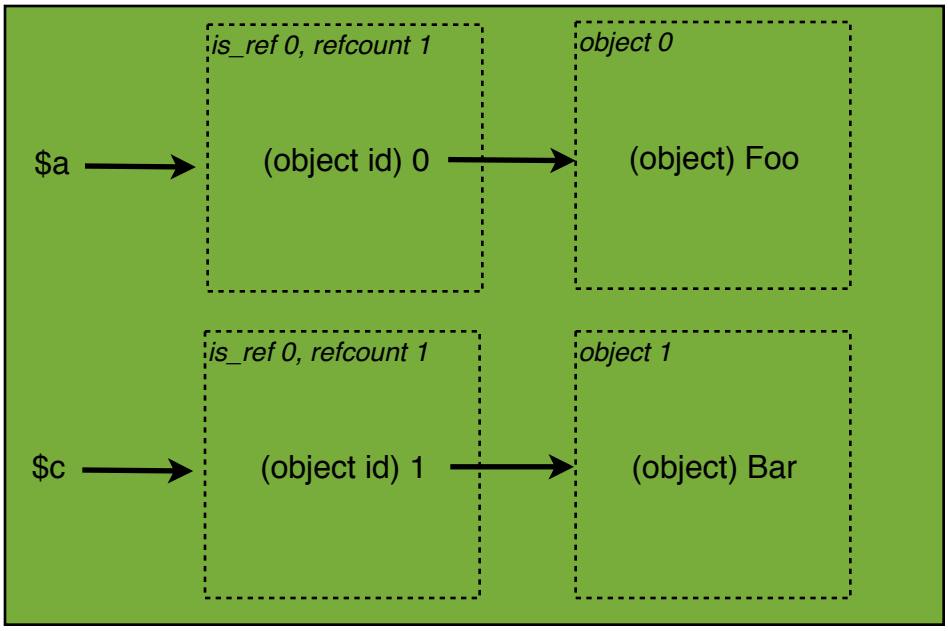
*When we set **$c=&$b**, a new object id is created for them, however both object id's still point to the same object container. This is important, because now if we do something like **$a->foo = "bar"**, this will affect **$a**, **$b**, and **$c** since they all ultimately point to the same object, even though the object id's differ!*

unset($b);



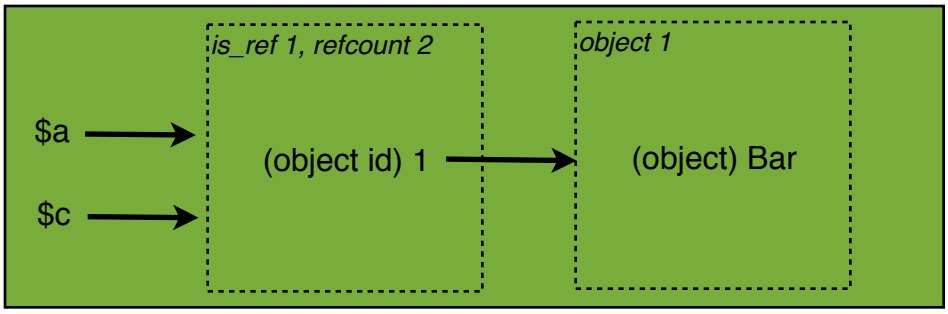
When a variable is unset, it is unlinked from its container.

$c = new Bar;



***$c** is now a new object, so a new object id and object are created for it.*

$a =& $c;



***$a** now points to the same object id as **$c**. Since the original object id has no reference, it is destroyed. Since the original object no longer has any reference, it is also destroyed.*