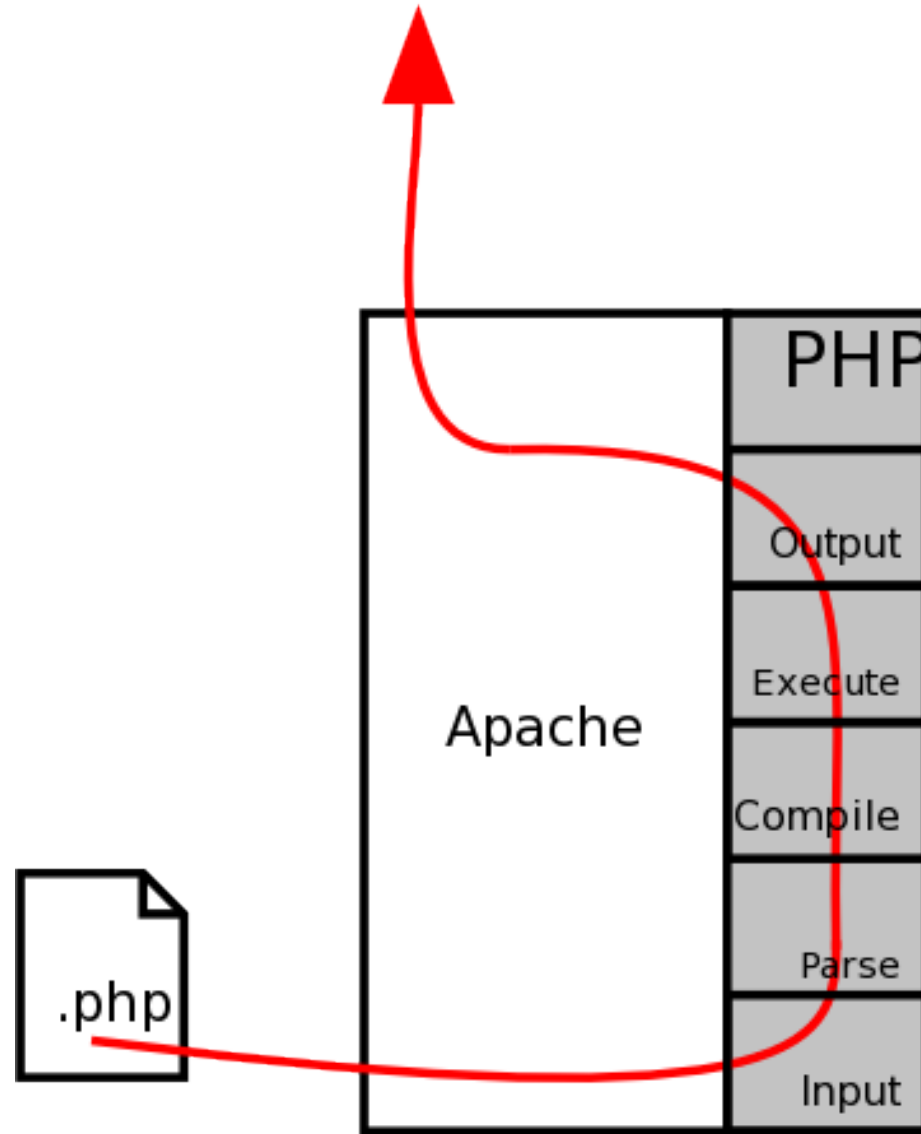




php|tek - Chicago, US  
Derick Rethans - dr@ez.no  
<http://derickrethans.nl/talks.php>

- Dutchman living in Norway
- eZ Systems A.S.
- eZ Components project lead
- PHP development
- mcrypt, input\_filter, date/time support, unicode
- QA

# Introduction



- Lexical analyze script source
- Divide into logical blocks of characters
- Give special blocks a meaning
- flex (but only 2.5.4!)

## The Parse Error:

```
Parse error: parse error,  
unexpected T_CLASS, expecting ',' or ';' in - on line 2
```

# Compiling Diagram

```
<?php
function normalizeColorArray($array)
{
    foreach (array_keys($array) as $key)
    {
        $array[$key] = (float) $array[$key]/255;
    }

    return $array;
}

function rgbToCMYK($rgbArray)
{
    $cya = 1 - min(1, max((float) $rgbArray['r'], 0));
    $mag = 1 - min(1, max((float) $rgbArray['g'], 0));
    $yel = 1 - min(1, max((float) $rgbArray['b'], 0));

    $min = min($cya, $mag, $yel);
    if (1 - $min == 0)
    {
        return array('c' => 1, 'm' => 1, 'y' => 1, 'k' => 0);
    }

    return array('c' => ($cya - $min) / (1 - $min),
                'm' => ($mag - $min) / (1 - $min),
                'y' => ($yel - $min) / (1 - $min),
                'k' => $min);
}

function rgbToCMYK2($r, $g, $b)
{
    return e2Nacht::rgbToCMYK(array('r' => $r, 'g' => $g, 'b' => $b));
}
?>
```

Parse

Compile  
zend\_compile



class symbol table

function symbol table
normalizeColorArray
rgbToCMYK
rgbToCMYK2



## fibonacci.php:

```
<?php
    $cache = array();

    function fibonacci($nr) {
        global $cache;

        if (isset($cache[$nr])) {
            return $cache[$nr];
        }
        switch ($nr) {
            case 0:
                die("Invalid Nr\n");
            case 1:
                return 1;
            case 2:
                return 1;
            default:
                $r = fibonacci($nr - 2) + fibonacci($nr - 1);
                $cache[$nr] = $r;
                return $r;
        }
    }

    echo fibonacci($argv[1])."\n";
?>
```



## Disassembler: vld

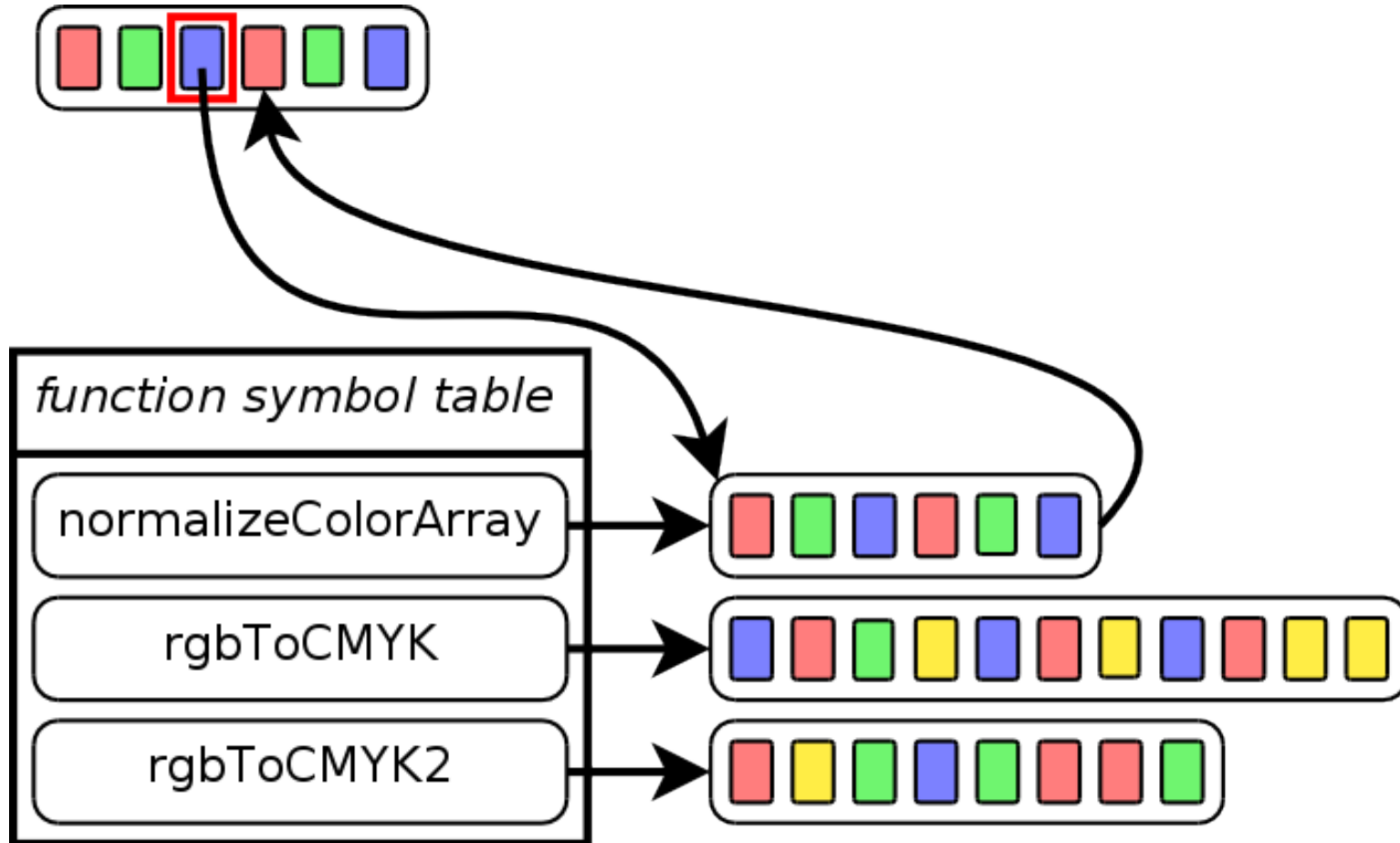
Dumps oparray per element:

- main script
- function
- class method

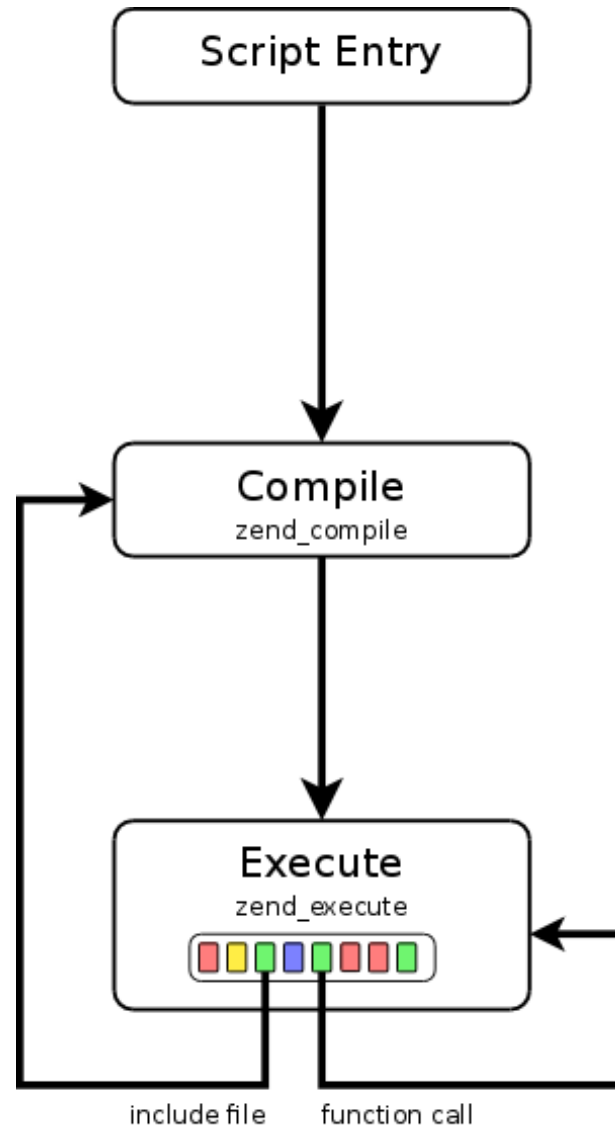
## Usage:

```
cvs -d :pserver:cvsread@cvs.xdebug.org:/repository login
# passwd = srmread
cvs -d :pserver:cvsread@cvs.xdebug.org:/repository co -d vld vle
cd vld
phpize && make && make install
php -dextension=vld.so -dvld.active=1 script.php
```

# Executing: Diagram



# Executing In a diagram



print vs. echo

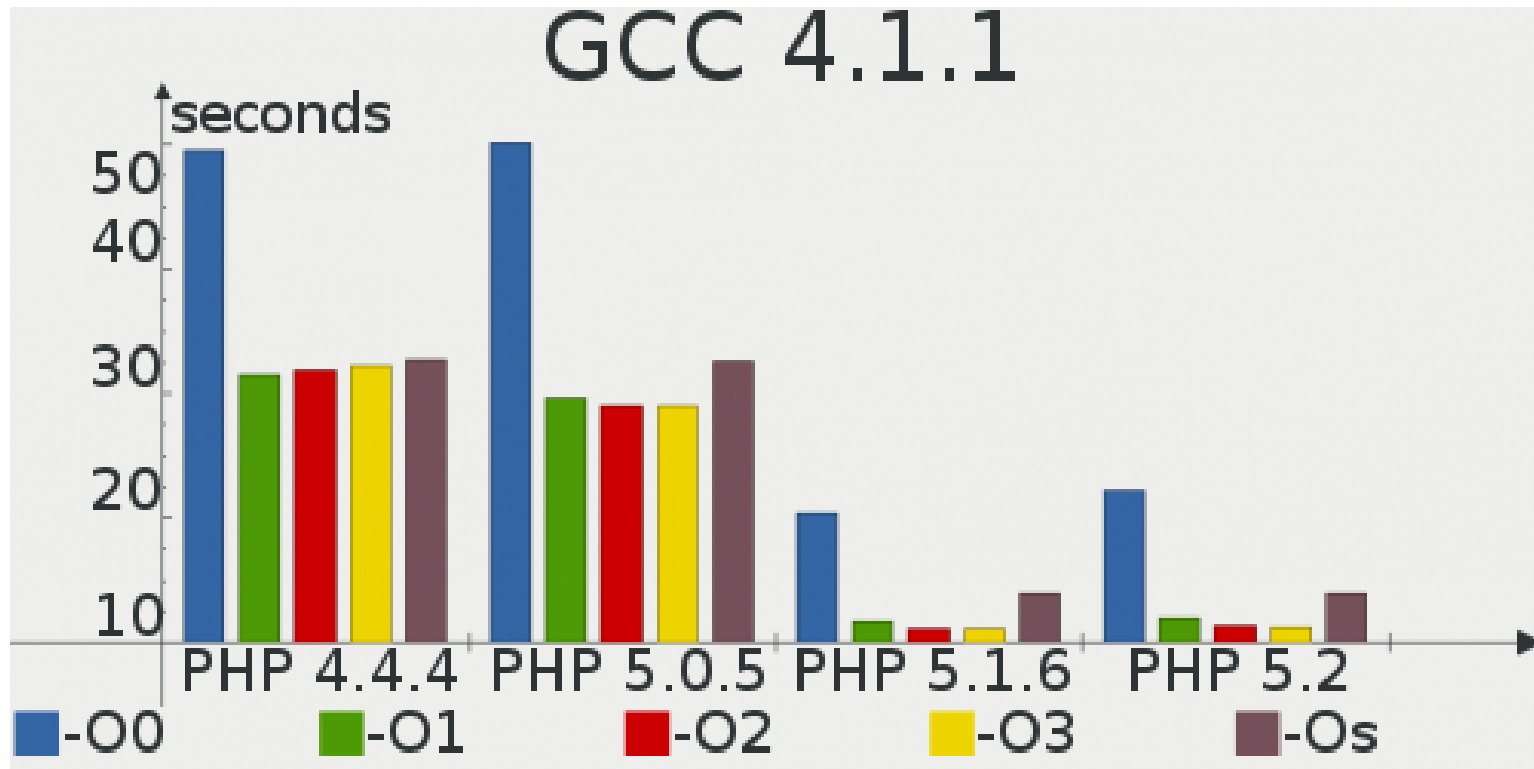
`$i++` vs. `++$i`

single quotes vs. double quotes

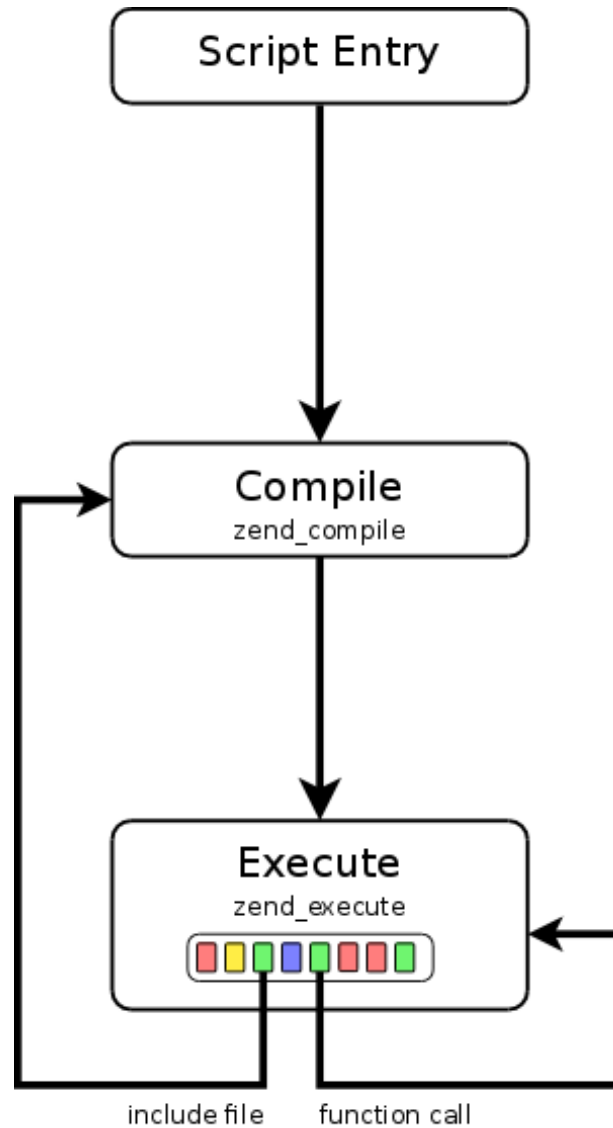
`${o}`



2003

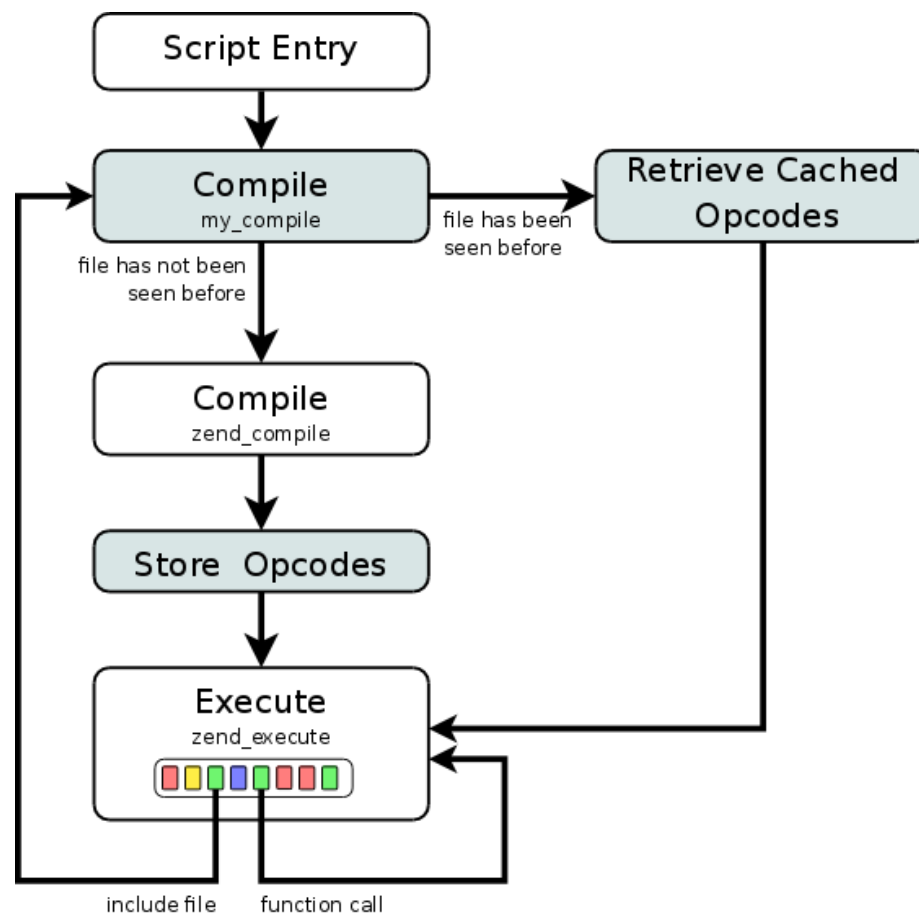


# Executing In a diagram

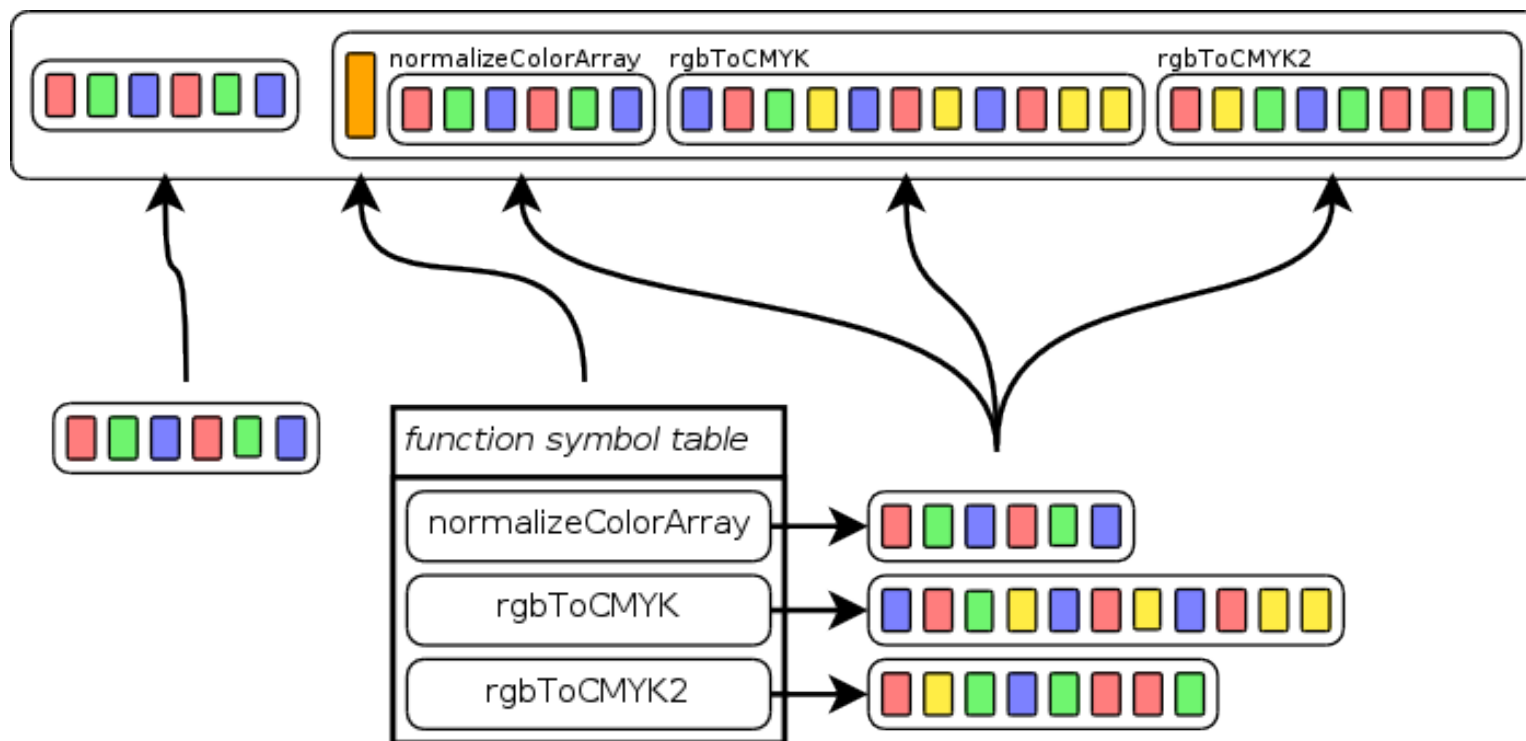


# Compiler Caches

## How it works



- In general, each source file is compiled once
- Compilation overhead becomes inconsequential
- Cache introduces its own overhead due to dynamic nature of includes



- Serialization into SHM
- Direct execution from SHM (mostly)

- APC: active development, PHP license
- eAccelerator: GPL
- PHP Accelerator: updated, but not improved
- Turck MM Cache: abandoned
- XCache: new
- Zend Platform: commercial

# Compiling

## Conditional functions

```
filename:      /home/httpd/html/test/conditional_func.php
function name: (null)
number of ops: 10
compiled vars: none
line   # op                                     operands
-----
if (true) {
  2     0 EXT_STMT
      1 JMPZ                                     true, ->5
      function foo() {
  3     2 EXT_STMT
      3 ZEND_DECLARE_FUNCTION                 '%00foo%2Ftmp%2Fconditional_func.php0xfe455f', 'foo'
  6     4 JMP                                     ->7
      function foo() {
  7     5 EXT_STMT
      6 ZEND_DECLARE_FUNCTION                 '%00foo%2Ftmp%2Fconditional_func.php0xfe458b', 'foo'
  }
  12    7 EXT_STMT
      8 RETURN                                  1
      9* ZEND_HANDLE_EXCEPTION
}
```

```
Function foo:
filename:      /home/httpd/html/test/conditional_func.php
function name: foo
number of ops: 6
compiled vars: none
line   # op                                     operands
-----
function foo() {
  3     0 EXT_NOP
      echo "1\n";
  4     1 EXT_STMT
      2 ECHO                                     '1%0A'
}
  5     3 EXT_STMT
      4 RETURN                                  null
      5* ZEND_HANDLE_EXCEPTION
}
```

End of function foo.

```
Function foo:
filename:      /home/httpd/html/test/conditional_func.php
function name: foo
number of ops: 6
compiled vars: none
line   # op                                     operands
-----
function foo() {
  7     0 EXT_NOP
      echo "2\n";
  8     1 EXT_STMT
      2 ECHO                                     '2%0A'
}
  9     3 EXT_STMT
      4 RETURN                                  null
      5* ZEND_HANDLE_EXCEPTION
}
```

End of function foo.

demo

Dumps includes/classes hierarchies

Download from <http://t3.dotgnu.info/blog/tags/included/>

**Install:**

```
tar -xvzf included-0.3.tgz
cd included
phpize && make && make install
```

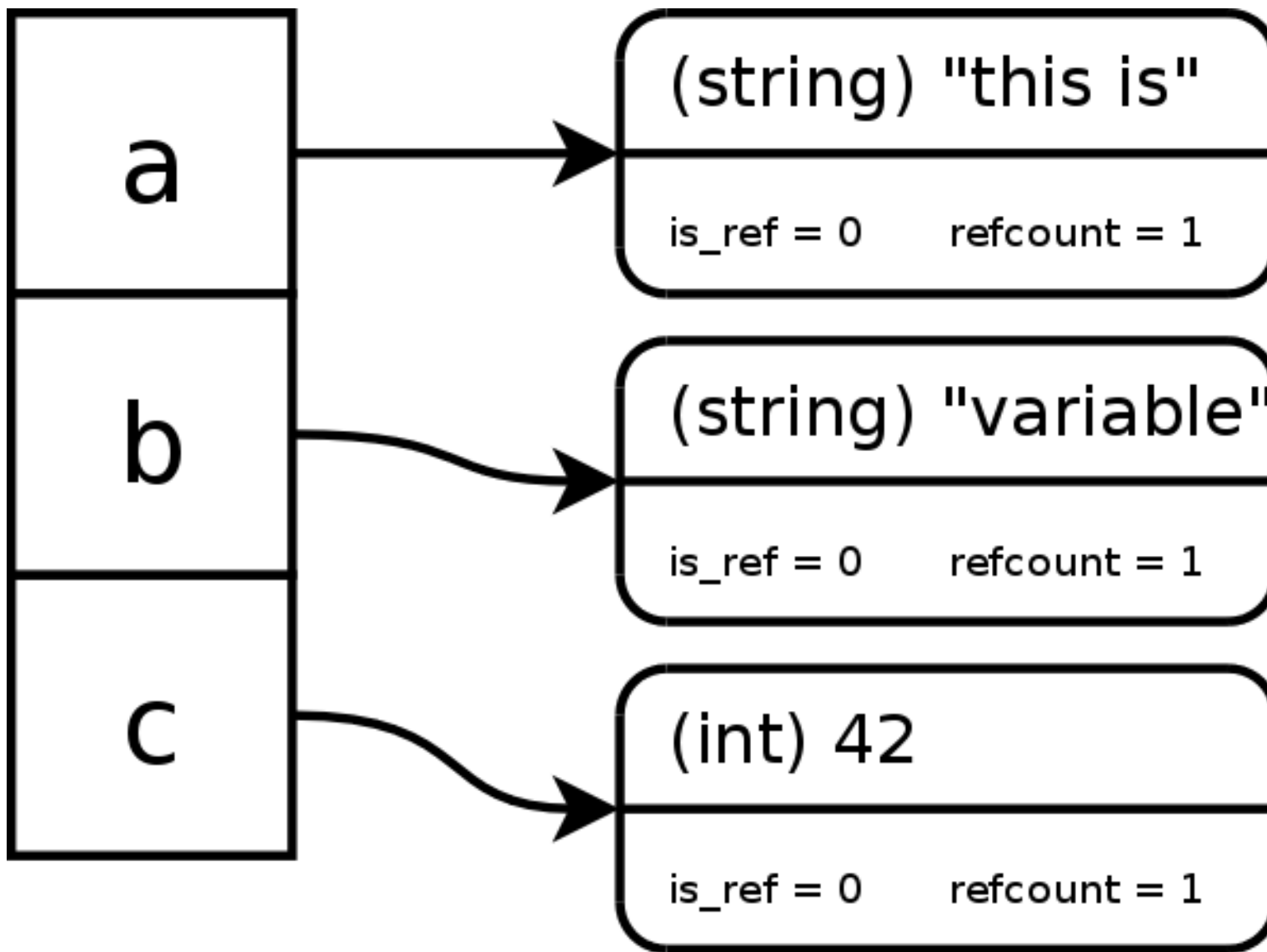
**Add to php.ini:**

```
extension=included.so
included.enabled=1
included.dumpdir=/tmp
```

demo

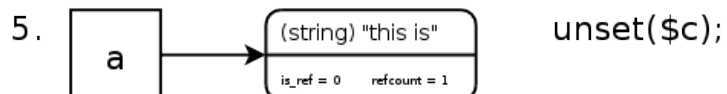
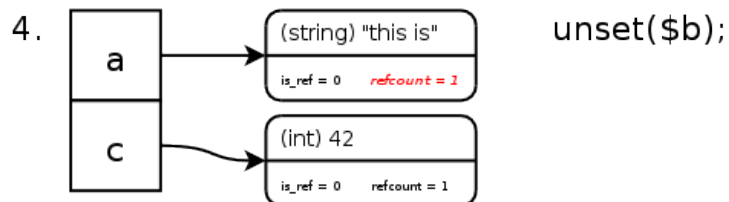
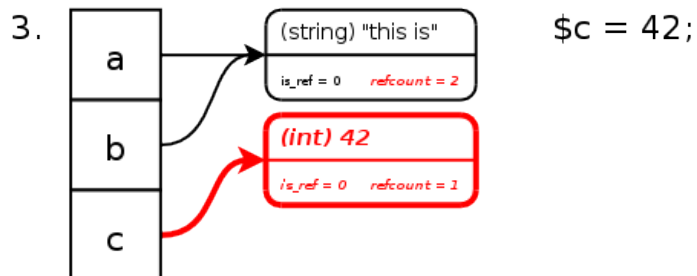
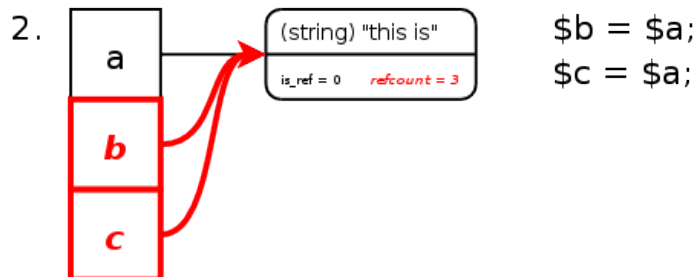
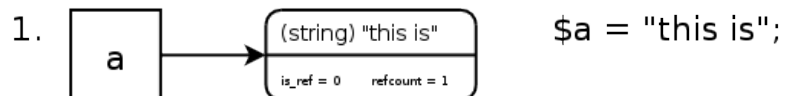
# Variables

## The Symbol Table



# Variables

## Introducing Refcounting

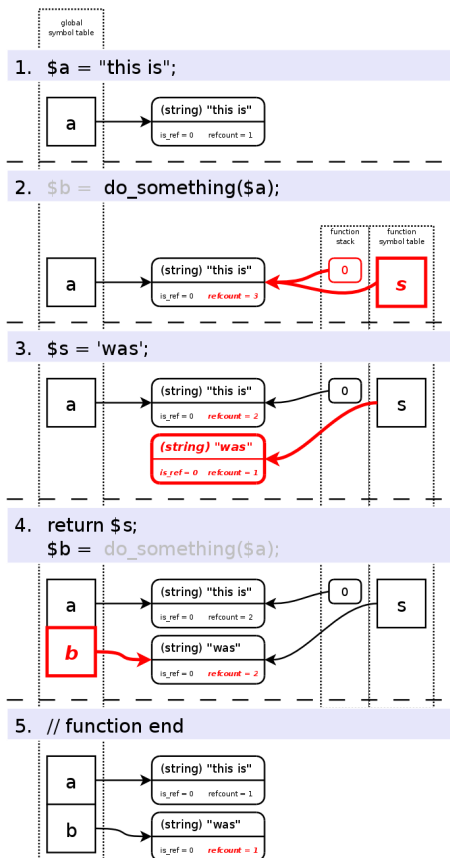


# Variables

## RefCounting and Passing Variables to Functions

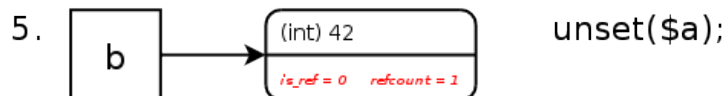
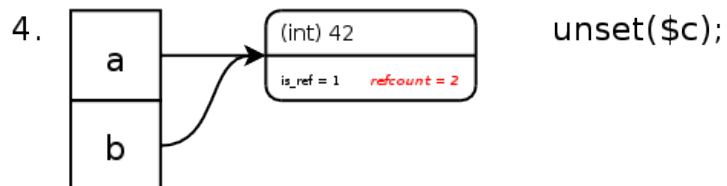
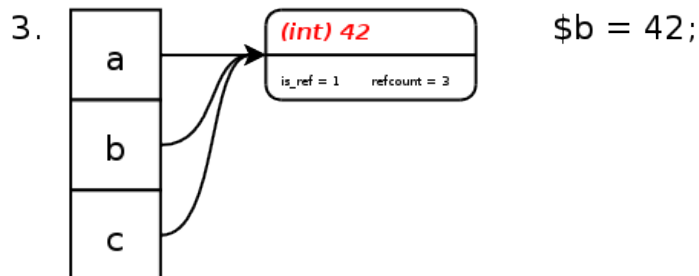
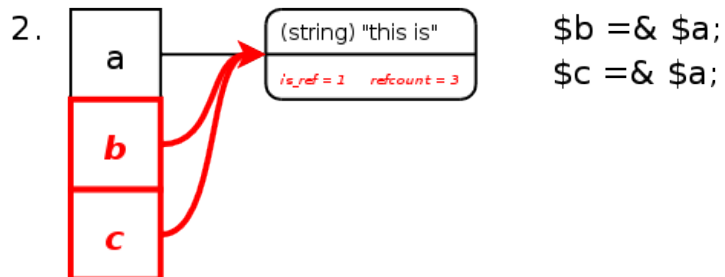
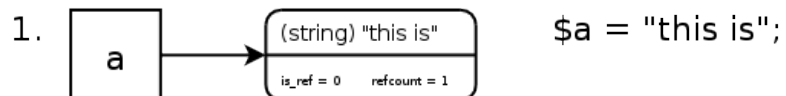
```
<?php
function do_something($s)
{
    $s = 'was';
    return $s;
}

$a = 'this is';
$b = do_something($a);
?>
```



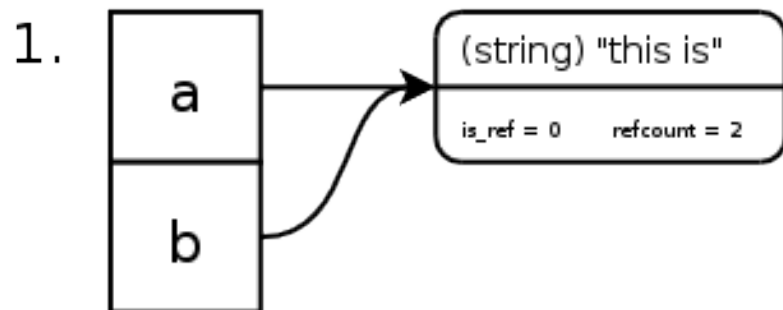
# Variables

## Introducing References

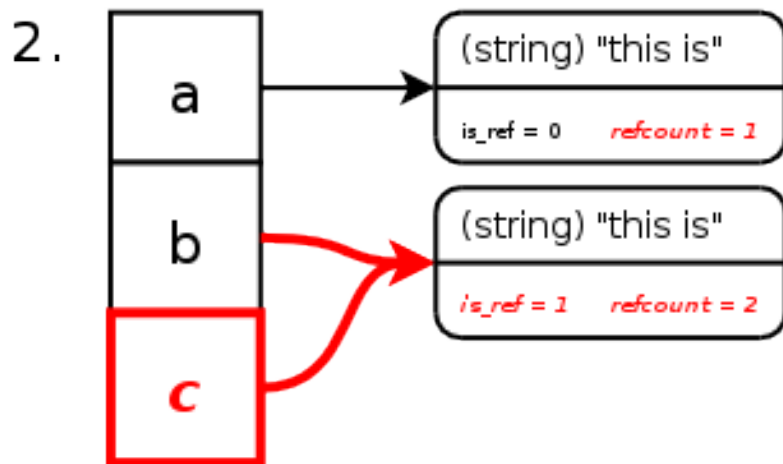


# Variables

## Mixing Assign-By Value and Assign-by Reference



```
$a = "this is";  
$b = $a;
```

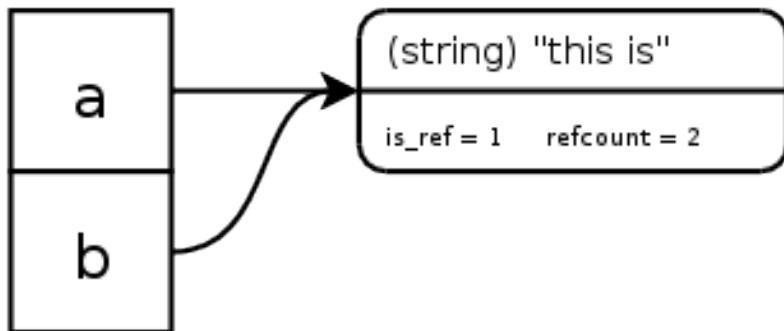


```
$c =& $b;
```

# Variables

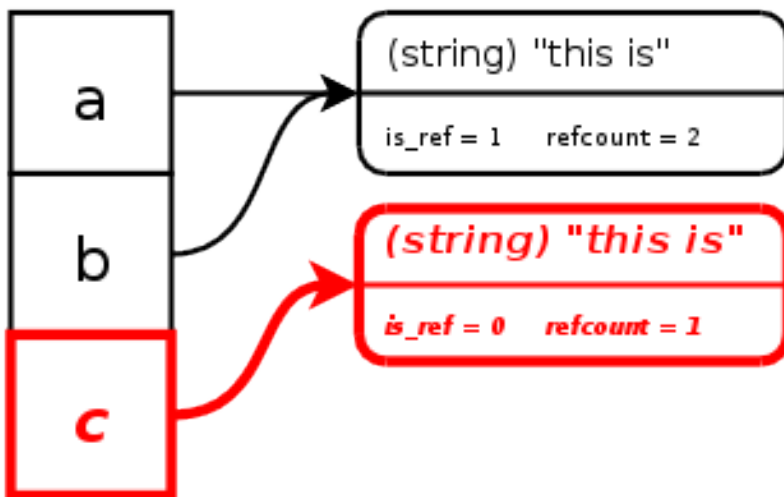
## Mixing Assign-by Reference and Assign-by Value

1.



```
$a = "this is";  
$b =& $a;
```

2.



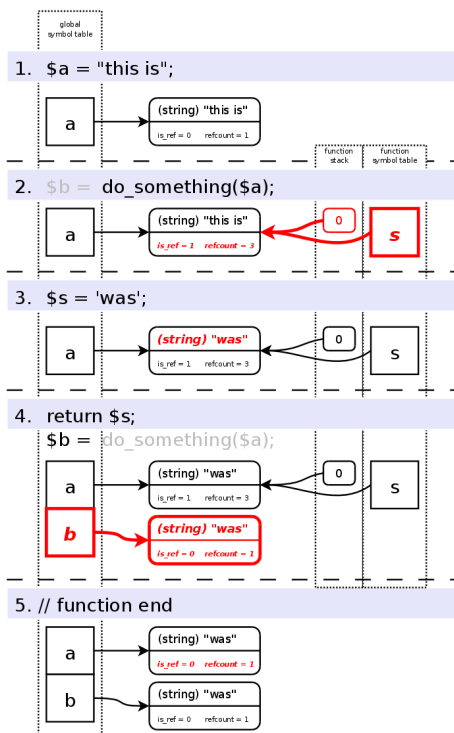
```
$c = $a;
```

# Variables

## Passing by Reference

```
<?php
function do_something(&$s)
{
    $s = 'was';
    return $s;
}

$a = 'this is';
$b = do_something($a);
?>
```

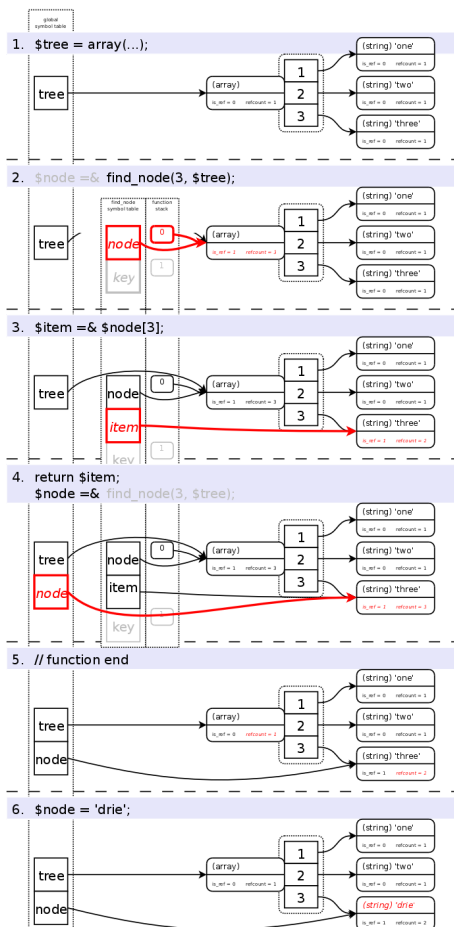


# Variables

## Return by Reference

```
<<?php
function &find_node($key, &$node)
{
    $item =& $node[$key];
    return $item;
}

$node =& find_node(3, $tree);
$node = 'drie';
?>
```

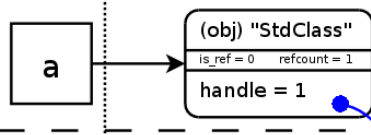


# Variables

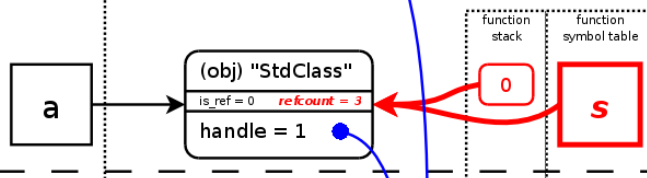
## Objects in PHP 5

```
<?php
function definition( $s )
{
    $s->prop = "baz";
}
$s = new StdClass;
$s->prop = "bar";
definition( $s );
?>
```

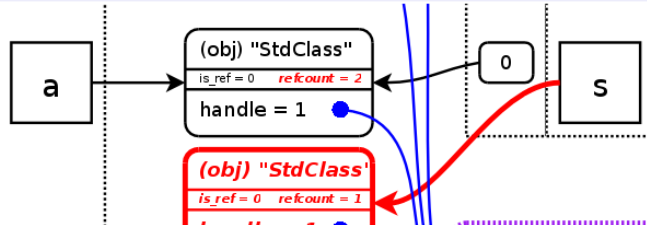
1. `$s = new StdClass; $s->prop = "bar";`



2. `definition( $s );`



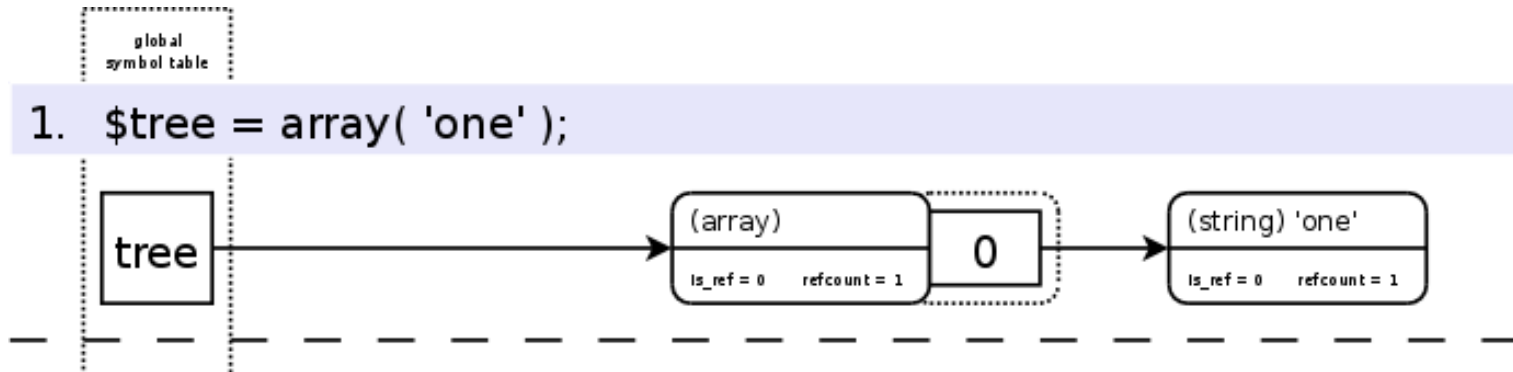
3. `$s->prop = "baz";`



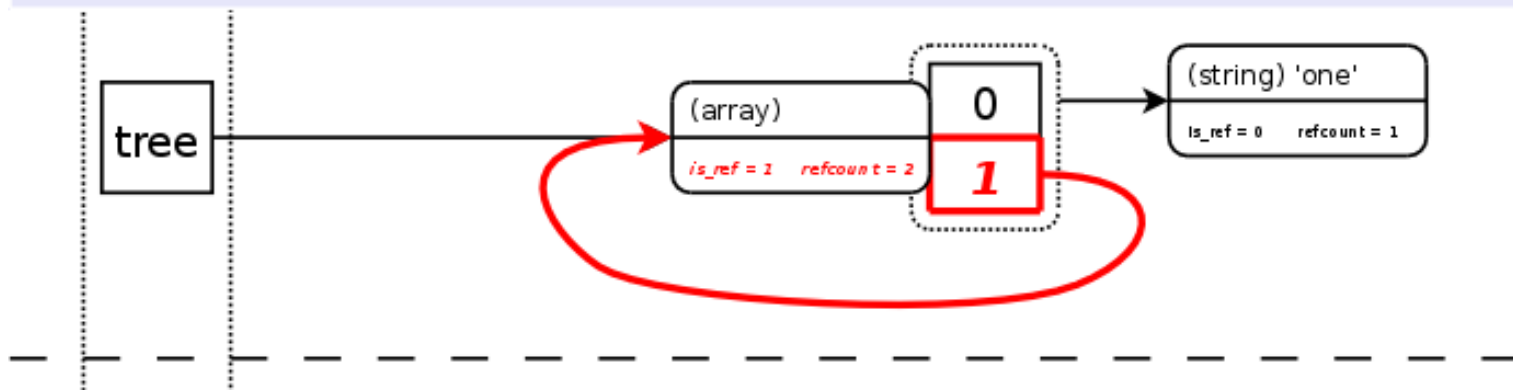
# Variables

## Circular References

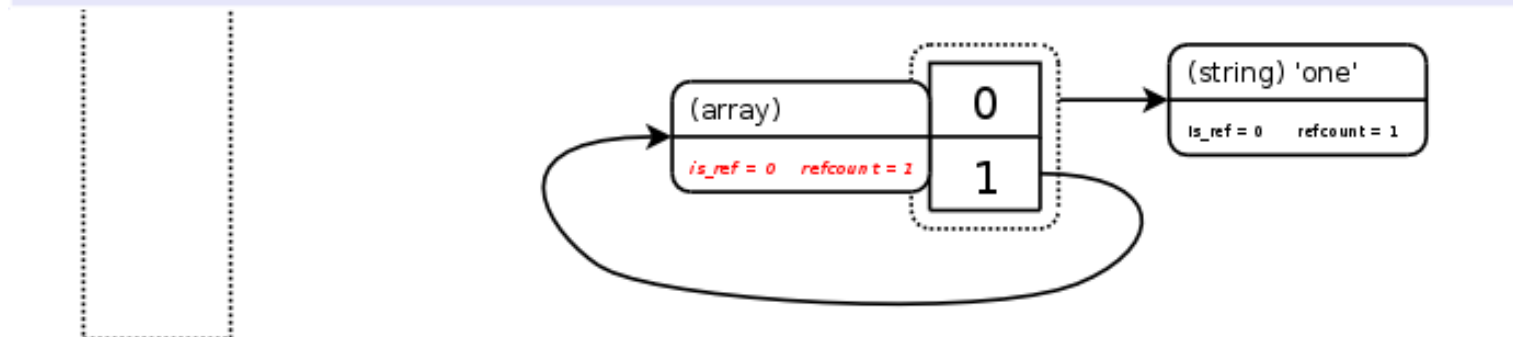
1. `$tree = array( 'one' );`



2. `$tree[] = &$tree;`



3. `unset( $tree );`





These Slides: <http://derickrethans.nl/talks.php>

VLD: <http://www.derickrethans.nl/vld.php>

Xdebug: <http://xdebug.org>

Included: <http://t3.dotgnu.info/blog/tags/included/>

PHP: <http://www.php.net>