eZ®
Share your information

Welcome!

php|tek - Chicago, US

Derick Rethans - dr@ez.no

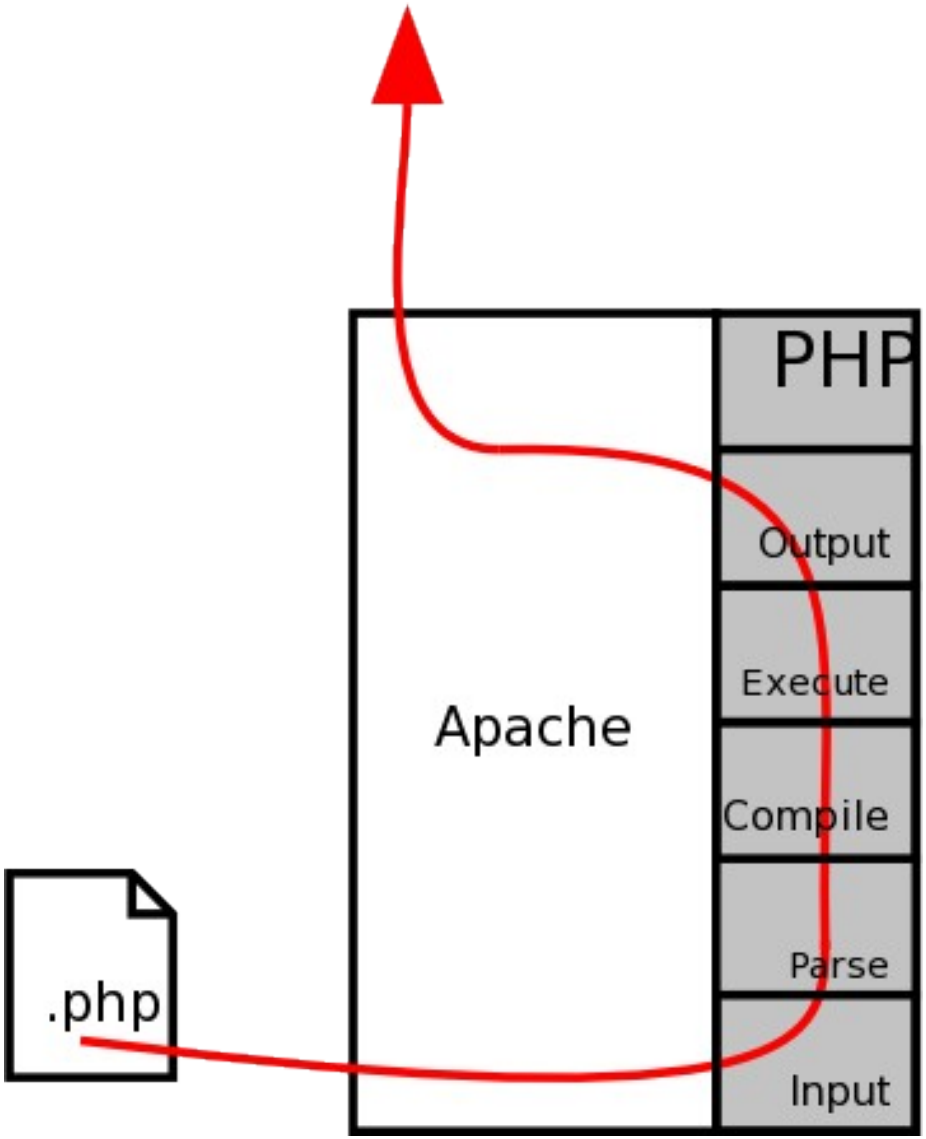
http://derickrethans.nl/talks.php

eZ
Share your information

About Me

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

Introduction



Input: Apache module

PHP registers handlers for:
Mime types:
- application/x-httpd-php
- application/x-httpd-php-source

Others:
- text/html (Apache xbithack)
- php-script (Apache 2)

On a request, Apache:
- checks for the MIME type handler
- opens the file
- fills a meta data record
- hands PHP a filepointer

Input: Apache/CGI

CGI binary is linked to mime-type:

```
ScriptAlias /php/ /usr/local/bin/php-cgi
AddType application/x-httpd-php .php
Action application/x-httpd-php "/php/php.exe"
```

On a request, Apache:

- checks if there is a handler for the mime type
- fills in needed environment variables
- executes the CGI processor

Parsing

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

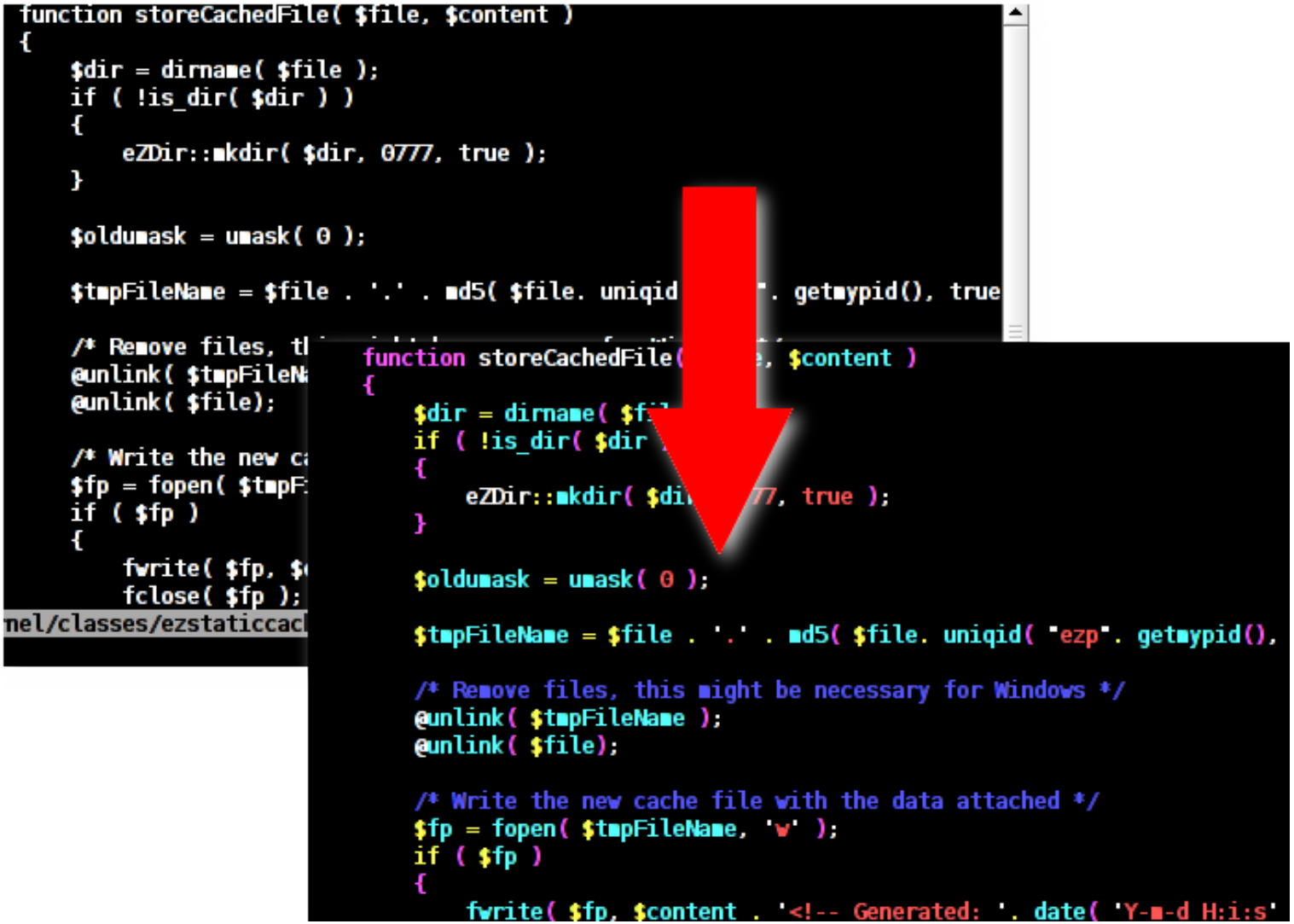
The Parse Error:

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

Parsing: Tokenize

Share your information

Syntax highlighting = Tokenizing



Parsing: Tokens

- The building blocks of a script
- Strings, variables, keywords
- Action rules:

```
<ST_IN_SCRIPTING>"::" {
    return T_PAAMAYIM_NEKUDOTAYIM;
}
<ST_IN_SCRIPTING>{DNUM}|{EXPONENT_DNUM} {
    zendlval->value.dval = strtod(yytext, NULL);
    zendlval->type = IS_DOUBLE;
    return T_DNUMBER;
}
```

Parsing: Tokens in PHP

From Zend/zend_language_parser.y:

```
%left T_INCLUDE T_INCLUDE_ONCE T_EVAL T_REQUIRE T_REQUIRE_ONCE
%left ','
%left T_LOGICAL_OR
%left T_LOGICAL_XOR
%left T_LOGICAL_AND
%right T_PRINT
%left '=' T_PLUS_EQUAL T_MINUS_EQUAL T_MUL_EQUAL T_DIV_EQUAL T_CONCAT_EQUAL
    T_MOD_EQUAL T_AND_EQUAL T_OR_EQUAL T_XOR_EQUAL T_SL_EQUAL T_SR_EQUAL
%left '?' ':'
%left T_BOOLEAN_OR
%left T_BOOLEAN_AND
%left '|'
%left '^'
%left '&'
%nonassoc T_IS_EQUAL T_IS_NOT_EQUAL T_IS_IDENTICAL T_IS_NOT_IDENTICAL
%nonassoc '<' T_IS_SMALLER_OR_EQUAL '>' T_IS_GREATER_OR_EQUAL
%left T_SL T_SR
%left '+' '-' '.'
%left '*' '/' '%'
%right '!' '~' T_INC T_DEC T_INT_CAST T_DOUBLE_CAST T_STRING_CAST T_ARRAY_CAST
    T_OBJECT_CAST T_BOOL_CAST T_UNSET_CAST '@'
%right '['
%nonassoc T_NEW T_INSTANCEOF
%token T_EXIT
%token T_IF
%left T_ELSEIF
%left T_ELSE
%left T_ENDIF
%token T_LNUMBER
%token T_DNUMBER
%token T_STRING
%token T_STRING_VARNAME
%token T_VARIABLE
%token T_NUM_STRING
%token T_INLINE_HTML
%token T_CHARACTER
%token T_BAD_CHARACTER
%token T_ENCAPSED_AND_WHITESPACE
%token T_CONSTANT_ENCAPSED_STRING
%token T_ECHO
```

Parsing: Example (1)

Tokenizer extension:

magic.php:

```
<?php
    function apply_magic(&$a) {
        $a = $a + rand(0,3);
    }
?>
```

tokenize.php:

```
<?php
    foreach (token_get_all($script) as $token) {
        if (count($token) == 2) {
            printf ("%-25s [%s]\n", token_name($token[0]), $token[1]);
        } else {
            printf ("%-25s [%s]\n", "", $token[0]);
        }
    }
?>
```

Parsing: Example (2)

```
<?php
    function apply_magic(&$a) {
        $a = $a + rand(0,3);
    }
?>
T_OPEN_TAG                [<?php\n]
T_WHITESPACE              [    ]
T_FUNCTION                [function]
T_WHITESPACE              [ ]
T_STRING                  [apply_magic]
                         [()]
                         [&]
T_VARIABLE                [$a]
                         [)]
T_WHITESPACE              [ ]
                         [{]
T_WHITESPACE              [\n        ]
T_VARIABLE                [$a]
T_WHITESPACE              [ ]
                         [=]
T_WHITESPACE              [ ]
T_VARIABLE                [$a]
T_WHITESPACE              [ ]
                         [+]
T_WHITESPACE              [ ]
T_STRING                  [rand]
                         [()]
T_LNUMBER                 [0]
                         [,]
T_LNUMBER                 [3]
                         [)]
                         [;]
T_WHITESPACE              [\n    ]
                         [}]
T_WHITESPACE              [\n]
T_CLOSE_TAG              [? >]
```

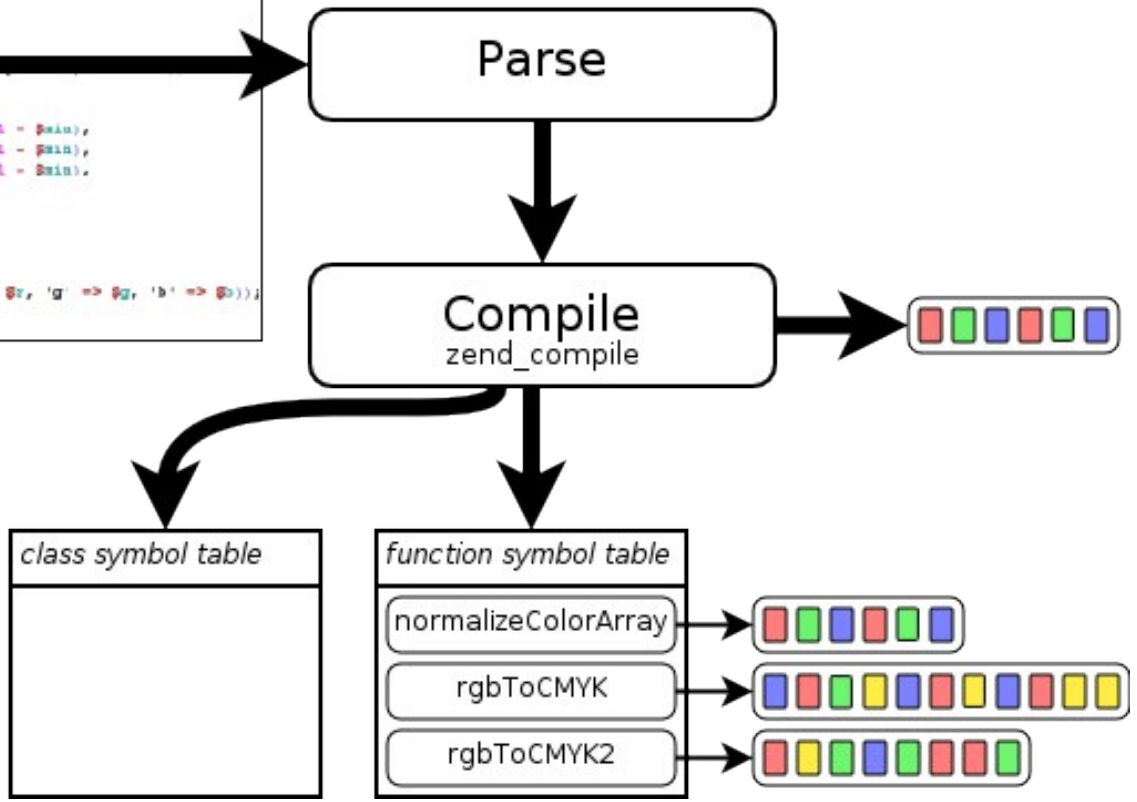
eZ®
Share your information

Compiling

- Interpreting tokens
- Generating execution units
- Generating class and function tables

Compiling
Diagram

```php
<?php
function normalizeColorArray($array)
{
    foreach (array_keys($array) as $key)
    {
        $array[$key] = (float) $array[$key]/256;
    }

    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,
    }

    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 eZMath::rgbToCMYK( array('r' => $r, 'g' => $g, 'b' => $b));
}
?>
```



Compiling: Example

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";
?>
```

Compiling
Break-down

```
Function fibonacci:
filename:       /home/httpd/html/test/pres/fibonacci.php
function name:  fibonacci
number of ops:  38
compiled vars:  !0 = $nr, !1 = $cache, !2 = $r
l#     #  op                                   return  operands
-------------------------------------------------------------------------
function fibonacci($nr) {
 4     0  RECV                                         1
    global $cache;
 5     1  FETCH_W                              $0      'cache'
       2  ASSIGN_REF                                    !1, $0
    if (isset($cache[$nr])) {
 7     3  ZEND_ISSET_ISEMPTY_DIM_OBJ           ~1      !1, !0
       4  JMPZ                                         ~1, ->8
        return $cache[$nr];
 8     5  FETCH_DIM_R                          $2      !1, !0
       6  RETURN                                       $2
    }
 9     7* JMP                                          ->8
    switch ($nr) {
        case 0:
11     8  CASE                                 ~3      !0, 0
       9  JMPZ                                         ~3, ->12
          die("Invalid Nr\n");
12    10  EXIT                                         'Invalid+Nr%0A'
13    11* JMP                                          ->14
        case 1:
      12  CASE                                 ~3      !0, 1
      13  JMPZ                                         ~3, ->16
          return 1;
14    14  RETURN                                       1
15    15* JMP                                          ->18
        case 2:
      16  CASE                                 ~3      !0, 2
      17  JMPZ                                         ~3, ->20
          return 1;
16    18  RETURN                                       1
17    19* JMP                                          ->21
        default:
      20  JMP                                          ->35
          $r = fibonacci($nr - 2) + fibonacci($nr - 1);
18    21  INIT_FCALL_BY_NAME                           'fibonacci'
      22  SUB                                  ~4      !0, 2
      23  SEND_VAL                                     ~4
      24  DO_FCALL_BY_NAME
```

Compiling
Disassembler

Disassembler: vld
Dumps oparray per element:

- main script
- function
- class method

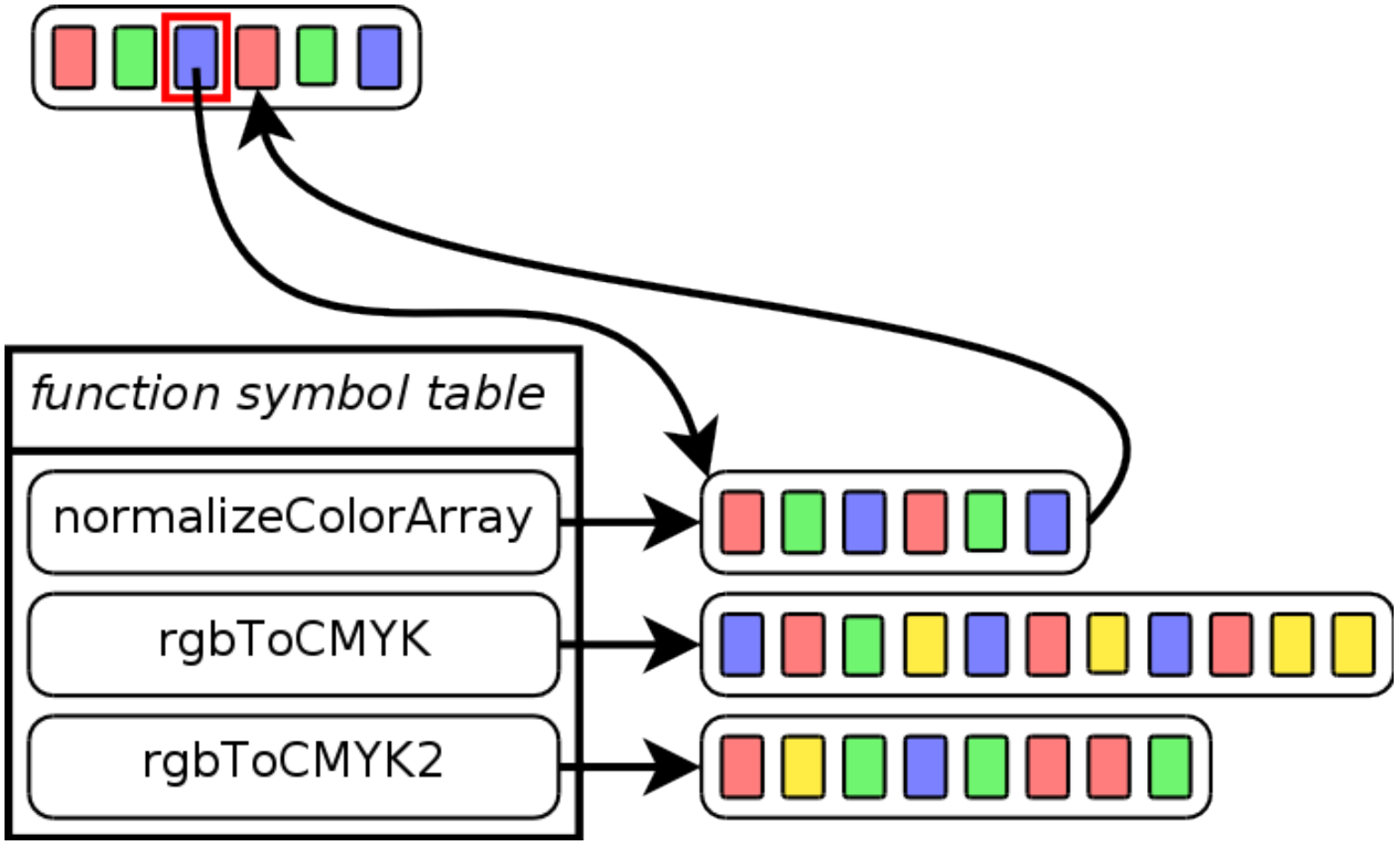
Usage:

```
wget wget http://pecl.php.net/get/vld-0.8.0.tgz
tar -xvzf vld-0.8.0.tgz
cd vld-0.8.0
phpize && make && make install
php -dextension=vld.so -dvld.active=1 script.php
```

Executing

- Executor executes opcodes
- 116 or 140 different opcodes
- Per file execution

Executing: Diagram



Executing Example
Scripts

test.inc:

```
<?php
    function foo () {
        echo "bar\n";
    }
    echo "inc\n";
?>
```

test.php:

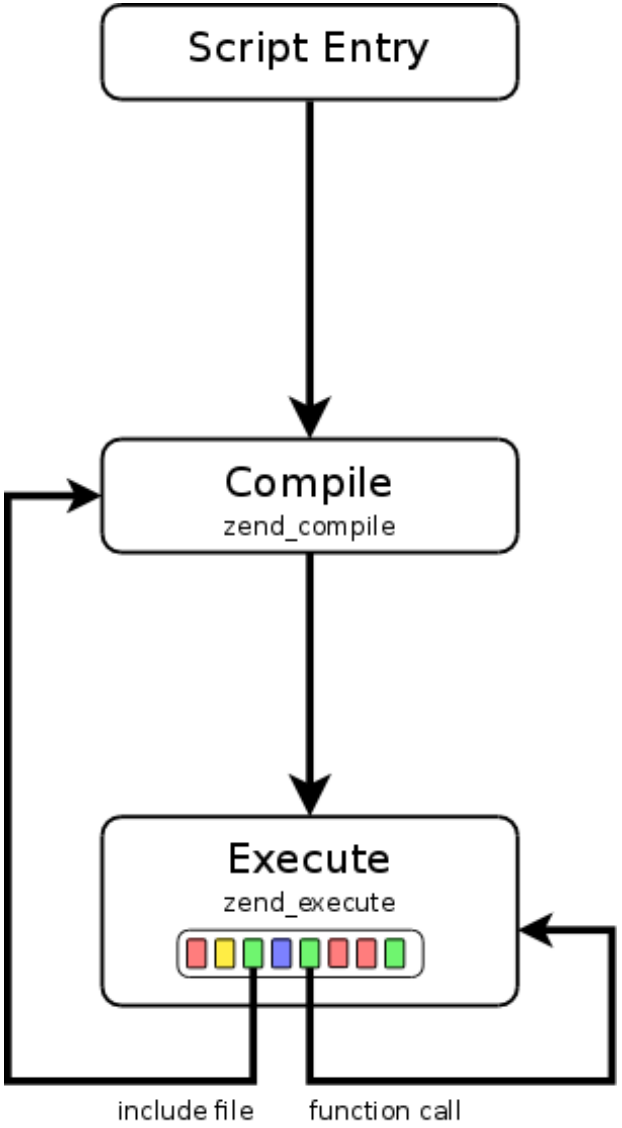
```
<?php
    echo "foo\n";
    include "test.inc";
    foo();
    echo "more bar\n";
? >
```

Executing Example
Workflow

The engine:

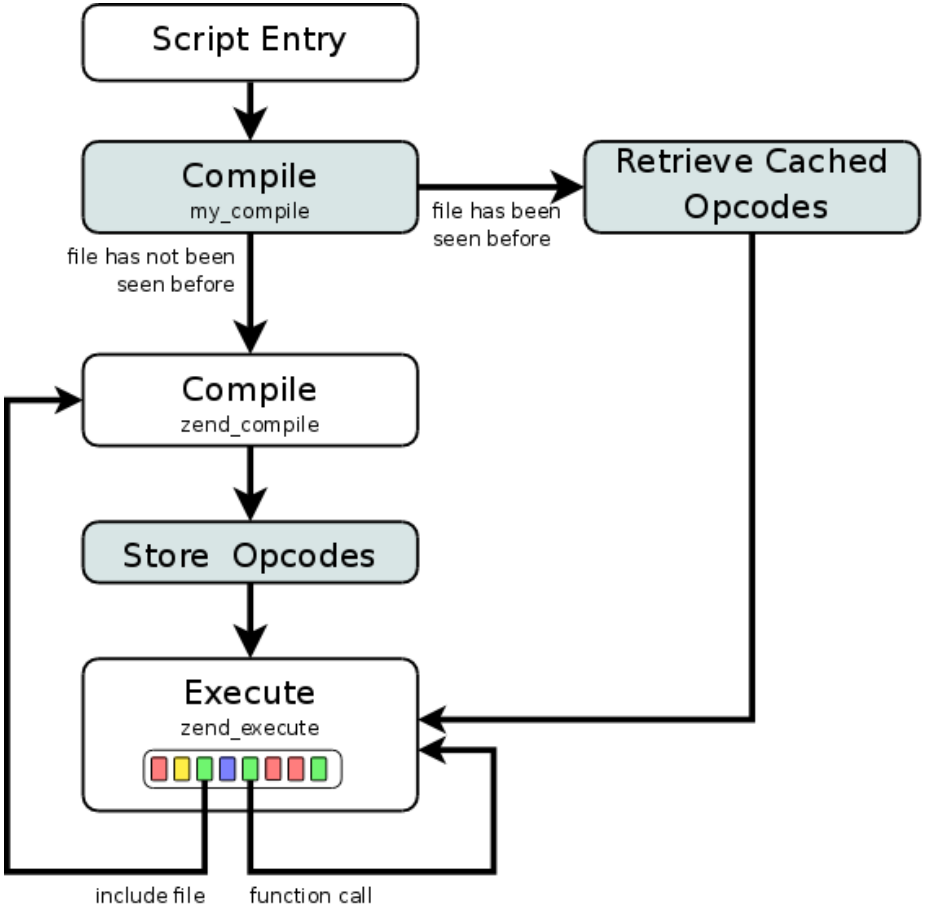
- parses and compiles test.php
- starts executing test.php
- parses and compiles test.inc when
- the include() statement is encountered
- starts executing test.inc
- resumes executing test.php

Executing
In a diagram



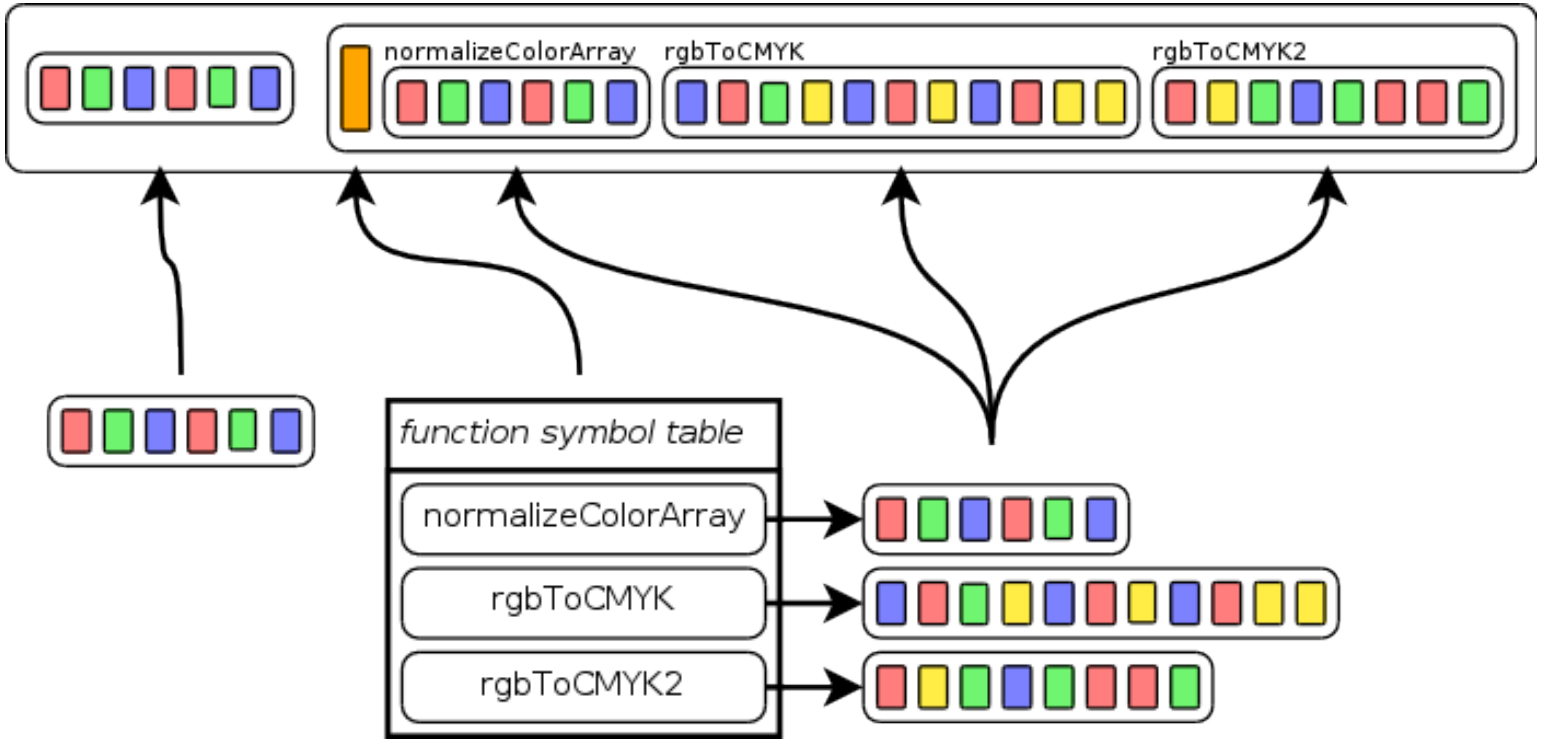
Compiler Caches

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

Compiler Caches
Serialization



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

PHP Accelerators

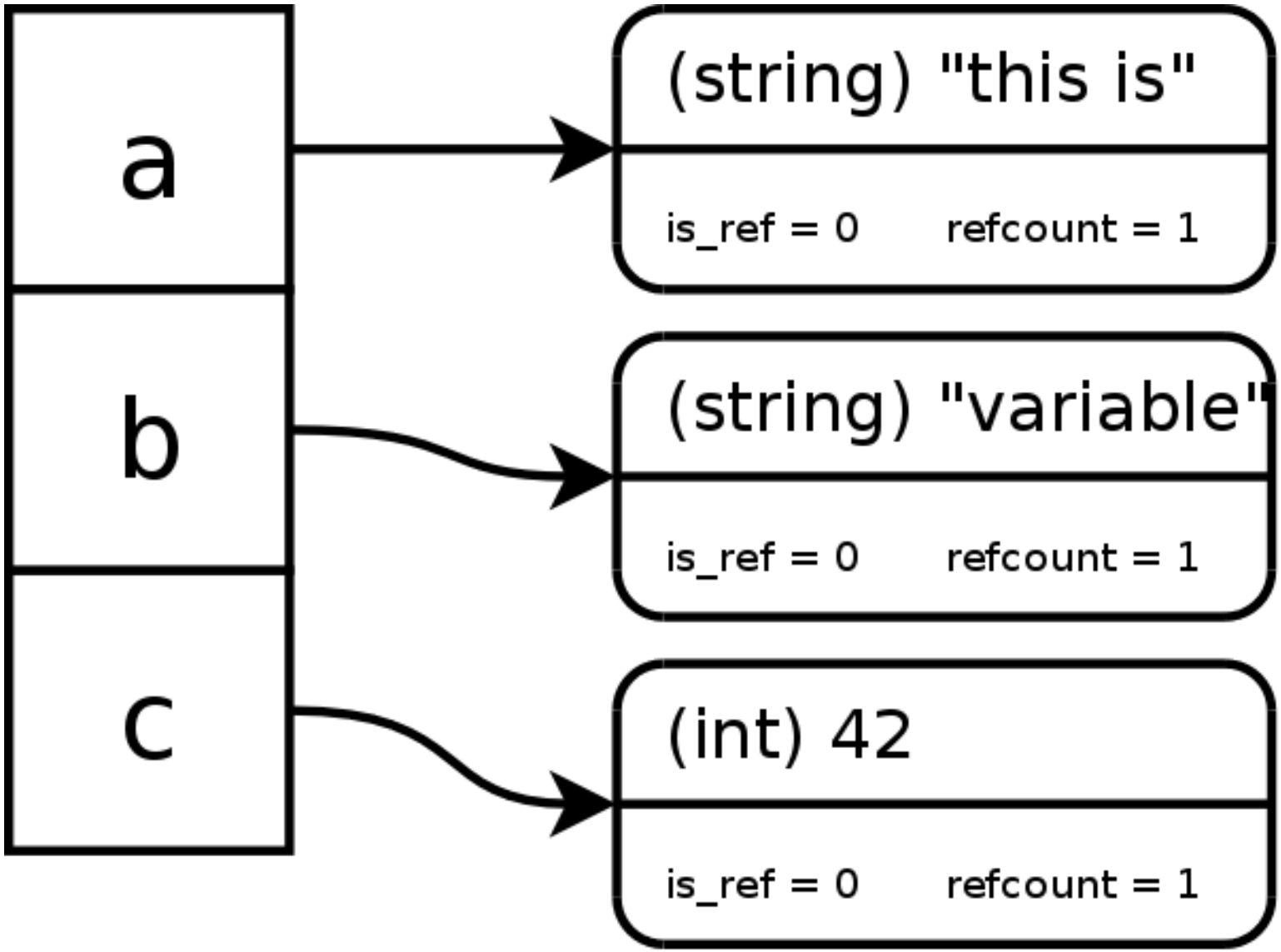
- APC: active development, PHP license
- eAcclerator: GPL
- PHP Accelerator: updated, but not improved
- Turck MM Cache: abandonned
- XCache: new
- Zend Platform: commercial

Questions

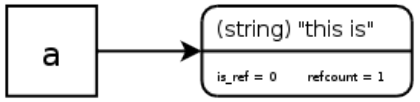
?

Variables

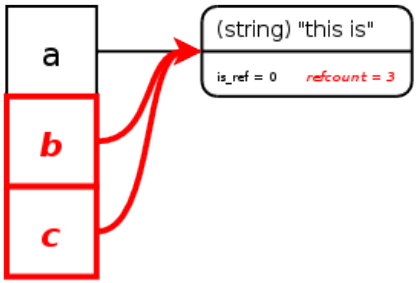
The Symbol Table



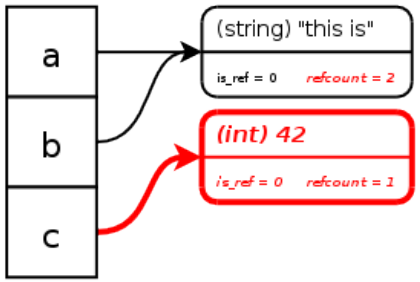
Variables
Introducing Refcounting

1. 

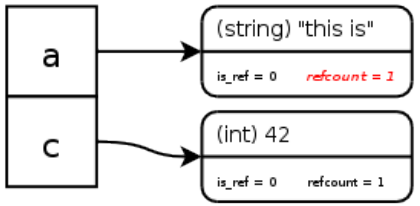

```
$a = "this is";
```

2. 

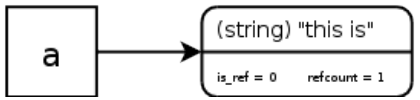

```
$b = $a;
$c = $a;
```

3. 


```
$c = 42;
```

4. 


```
unset($b);
```

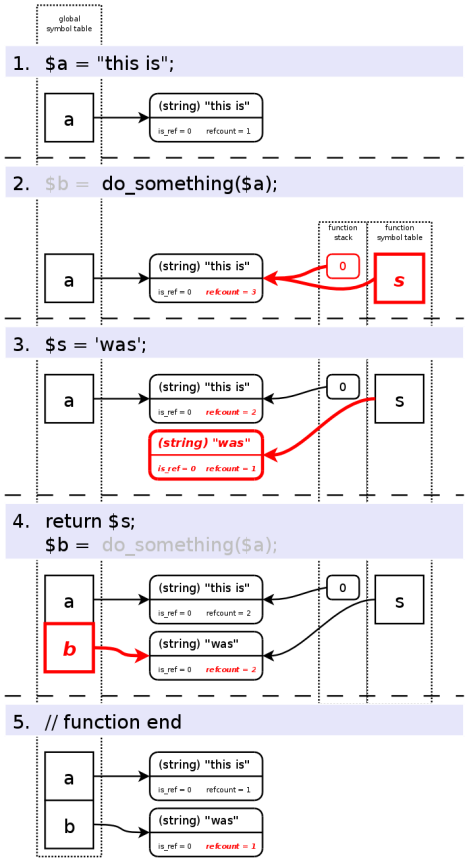
5. 


```
unset($c);
```

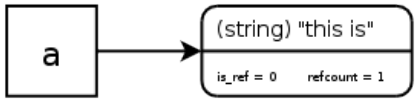
Variables

Refcounting and Passing Variables to Functions

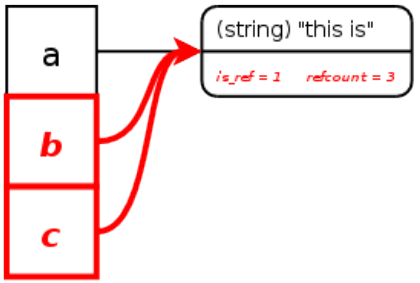
```
<?php
function do_something($s)
{
   $s = 'was';
   return $s;
}

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



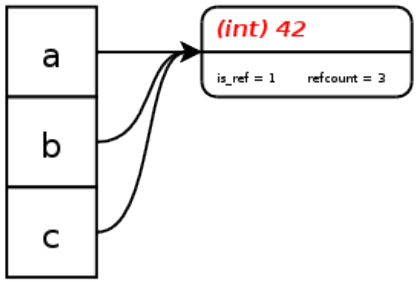
Variables
Introducing References

1. 

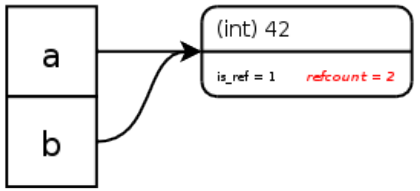

```
$a = "this is";
```

2. 

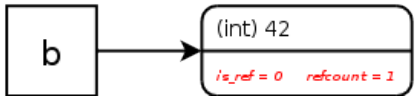

```
$b =& $a;
$c =& $a;
```

3. 


```
$b = 42;
```

4. 


```
unset($c);
```

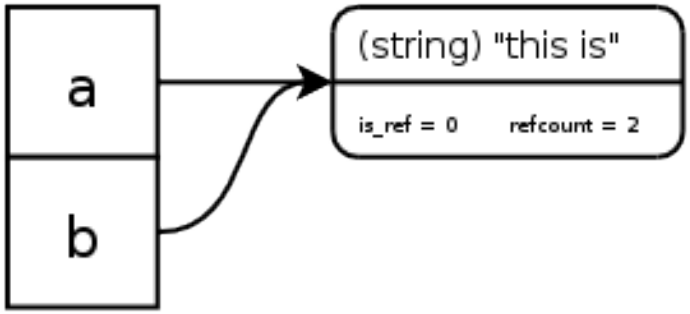
5. 


```
unset($a);
```

Variables

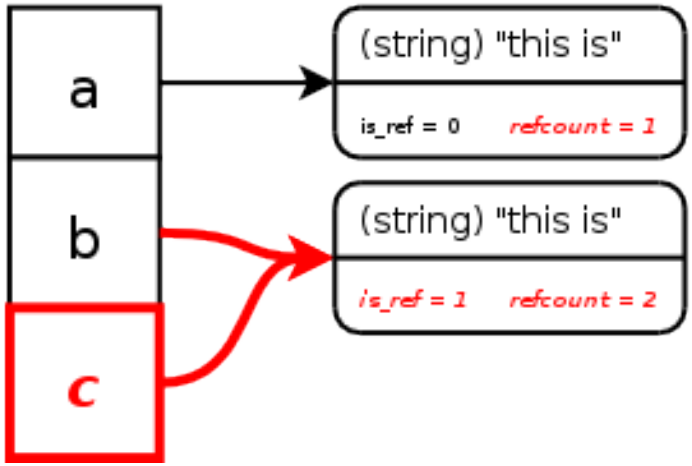
Mixing Assign-By Value and Assign-by Reference

1.




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

2.

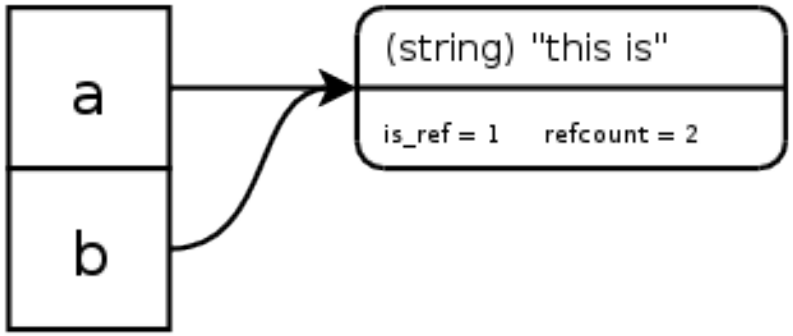



```
$c =& $b;
```

Variables

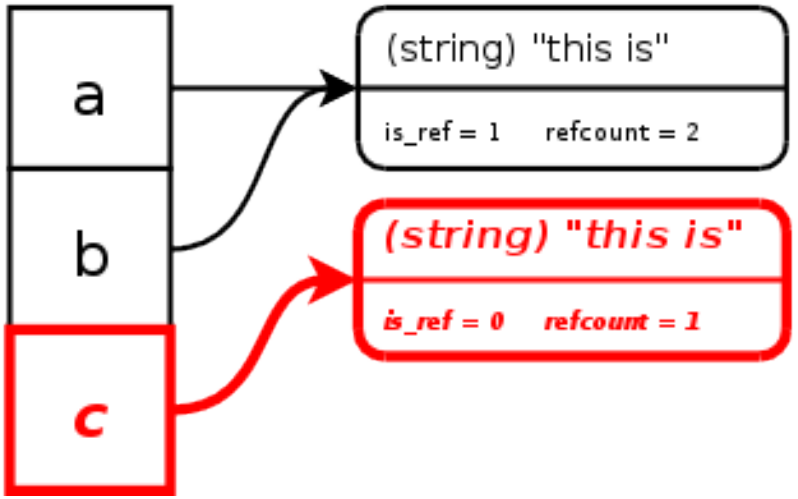
Mixing Assign-by Reference and Assign-by Value

1.




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

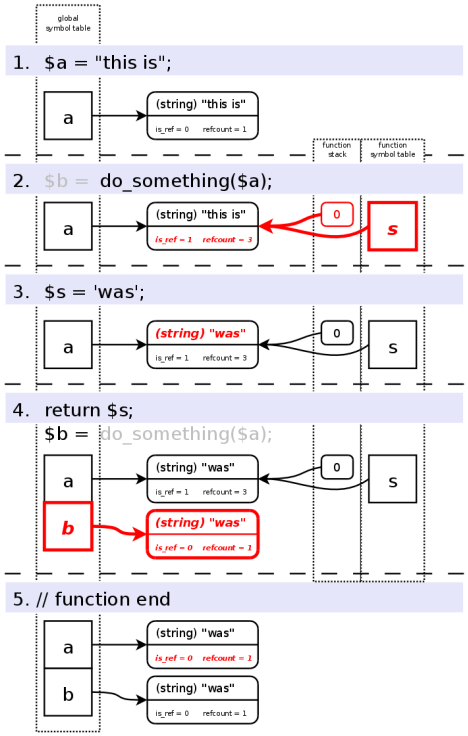
2.




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

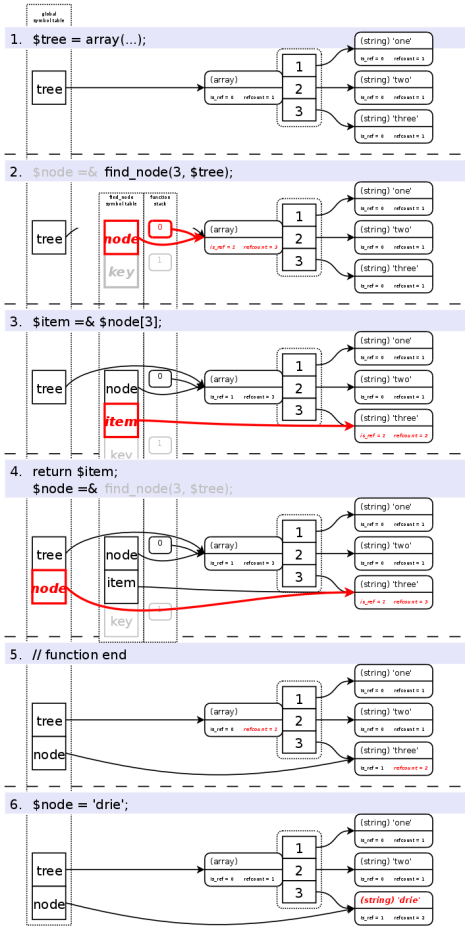
Variables
Passing by Reference

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

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



Variables
Return by Reference

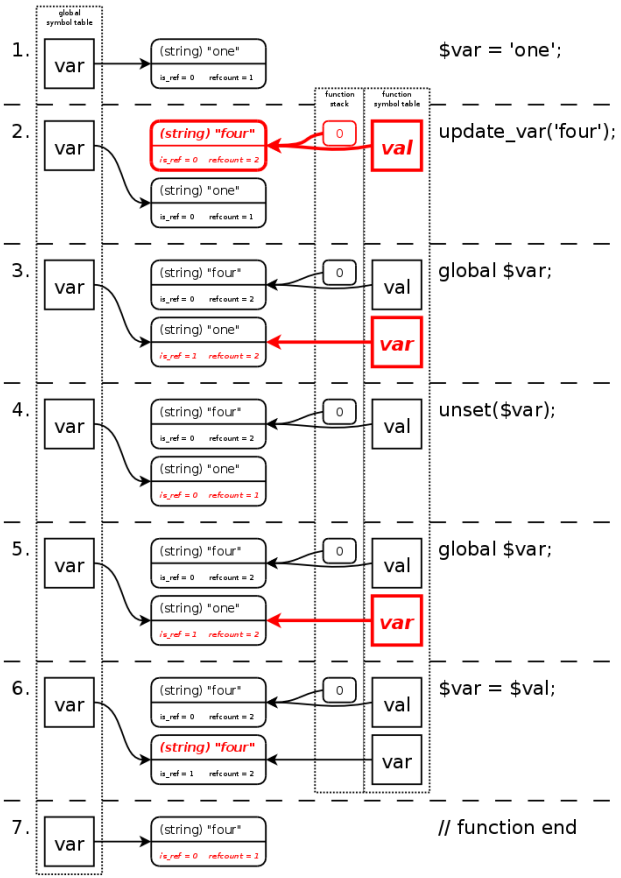
```
<?php
function &find_node($key, &$item)
{
    $item =& $node[$key];
    return $item;
}

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



Variables

The "global" Keyword

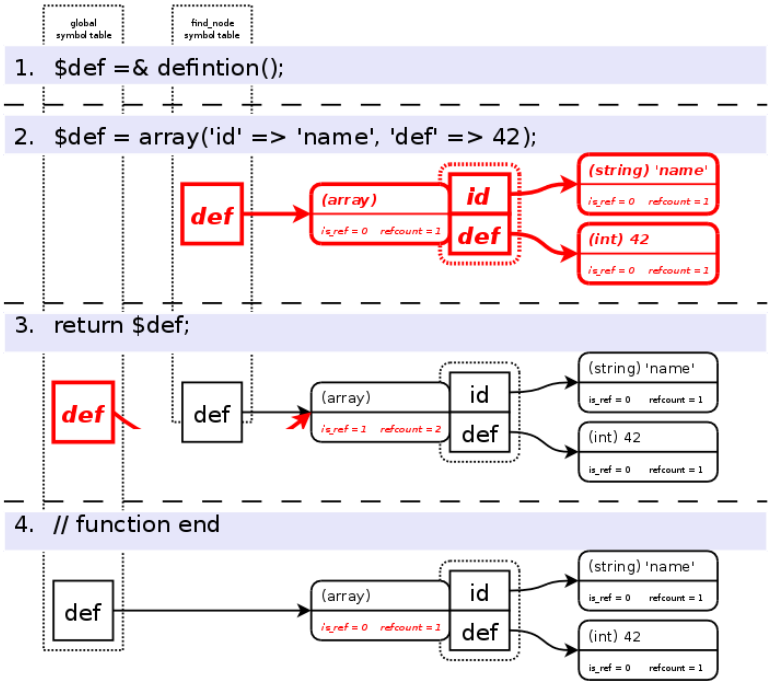
```
<?php
$var = 'one';

function update_var($val)
{
    global $var;
    unset($var);
    global $var;
    $var = $val;
}

update_tree('four');
```



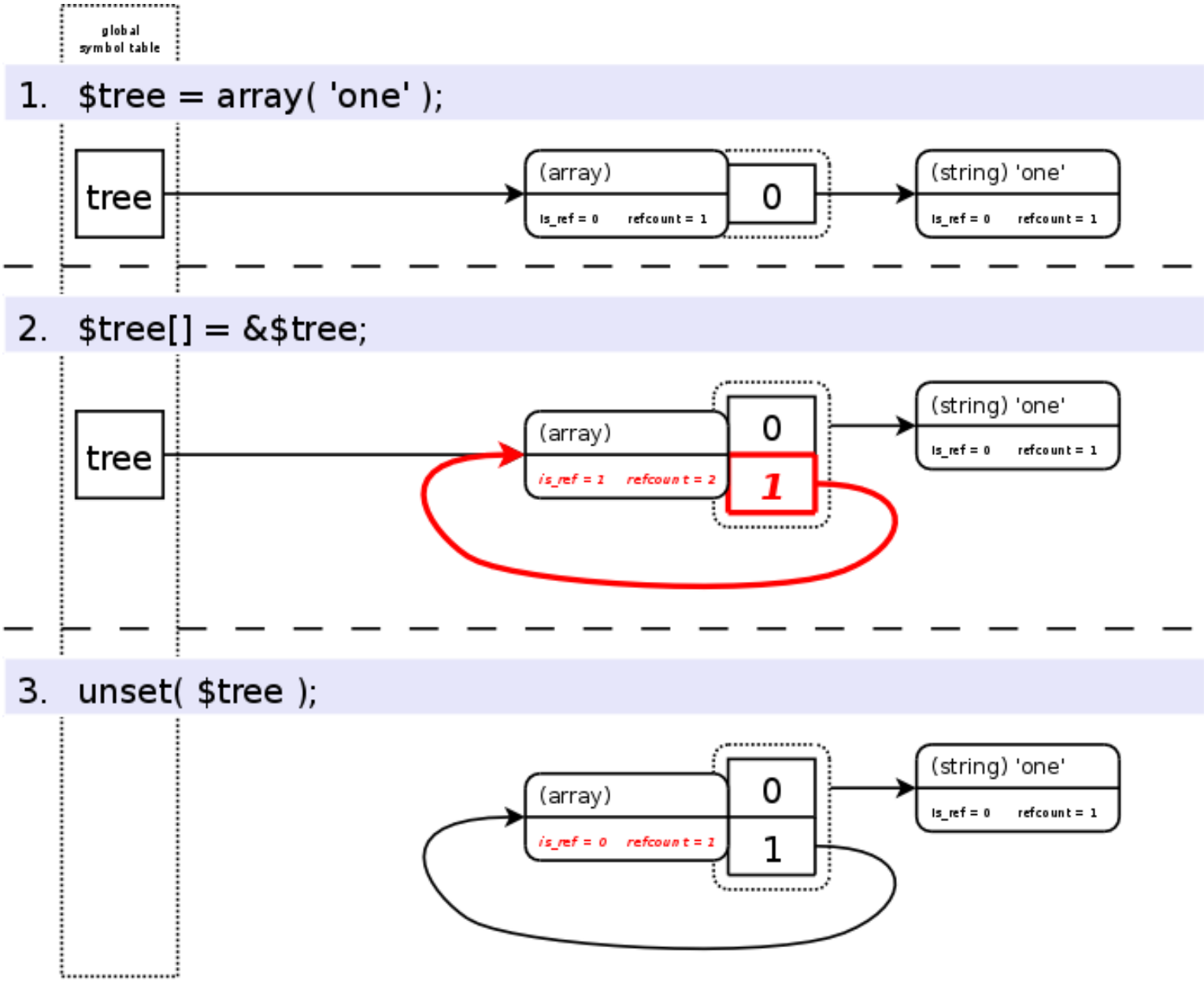
Variables

Bad Return by Reference

```
<?php
function &definition()
{
    $def = array('id' => 'name', 'def' => 42);
    return $def;
}

$def =& definition();
?>
```



Variables
Circular References



Questions

?

eZ
Share your information

Resources

These Slides: http://pres.derickrethans.nl
VLD site: http://www.derickrethans.nl/vld.php

PHP: http://www.php.net