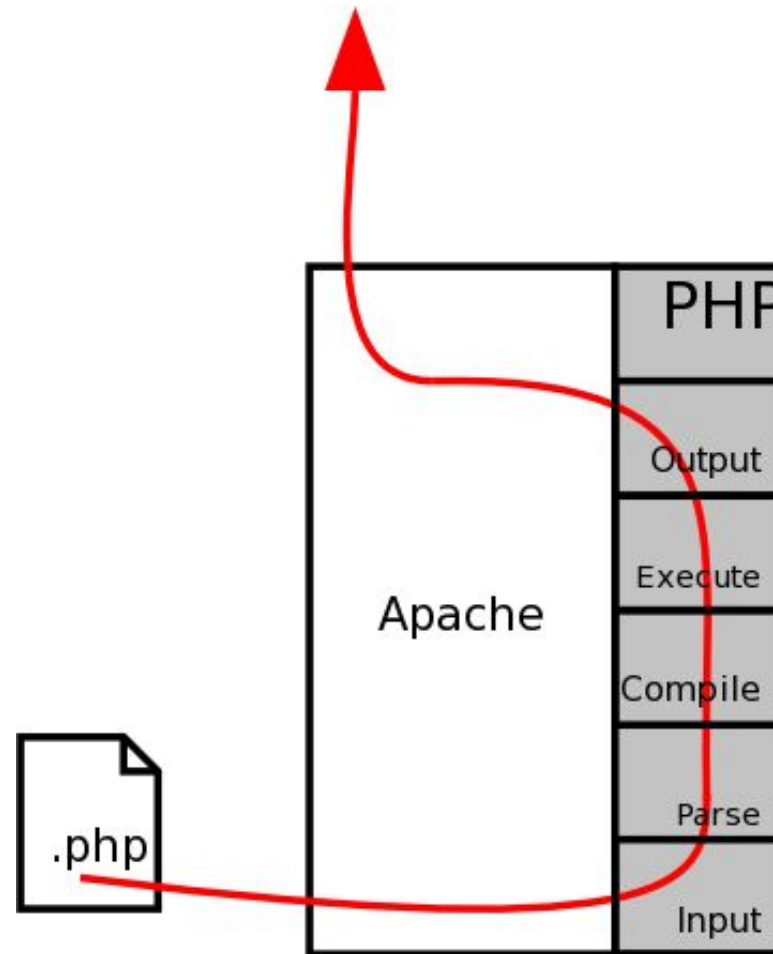


Welcome!

# 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

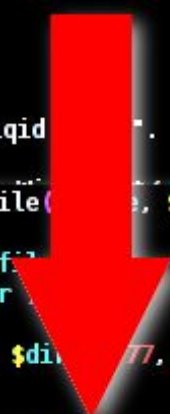
- Lexical analyze script source
- Divide into logical blocks of characters
- Give special blocks a meaning
- flex (but only 2.5.4!)

## The Parse Error:

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

# Parsing: Tokenize

Syntax highlighting = Tokenizing



```
function storeCachedFile( $file, $content )
{
    $dir = dirname( $file );
    if ( !is_dir( $dir ) )
    {
        eZDir::mkdir( $dir, 0777, true );
    }

    $oldumask = umask( 0 );

    $tmpFileName = $file . '.' . md5( $file, uniqid( "ezp". getmypid(), true ) );

    /* Remove files, this might be necessary for Windows */
    @unlink( $tmpFileName );
    @unlink( $file );

    /* Write the new cache file with the data attached */
    $fp = fopen( $tmpFileName, 'w' );
    if ( $fp )
    {
        fwrite( $fp, $content . '<!-- Generated: ' . date( 'Y-m-d H:i:s'
```

# 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;  
}
```

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_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
```

# Parsing: Tokens in PHP

# 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)

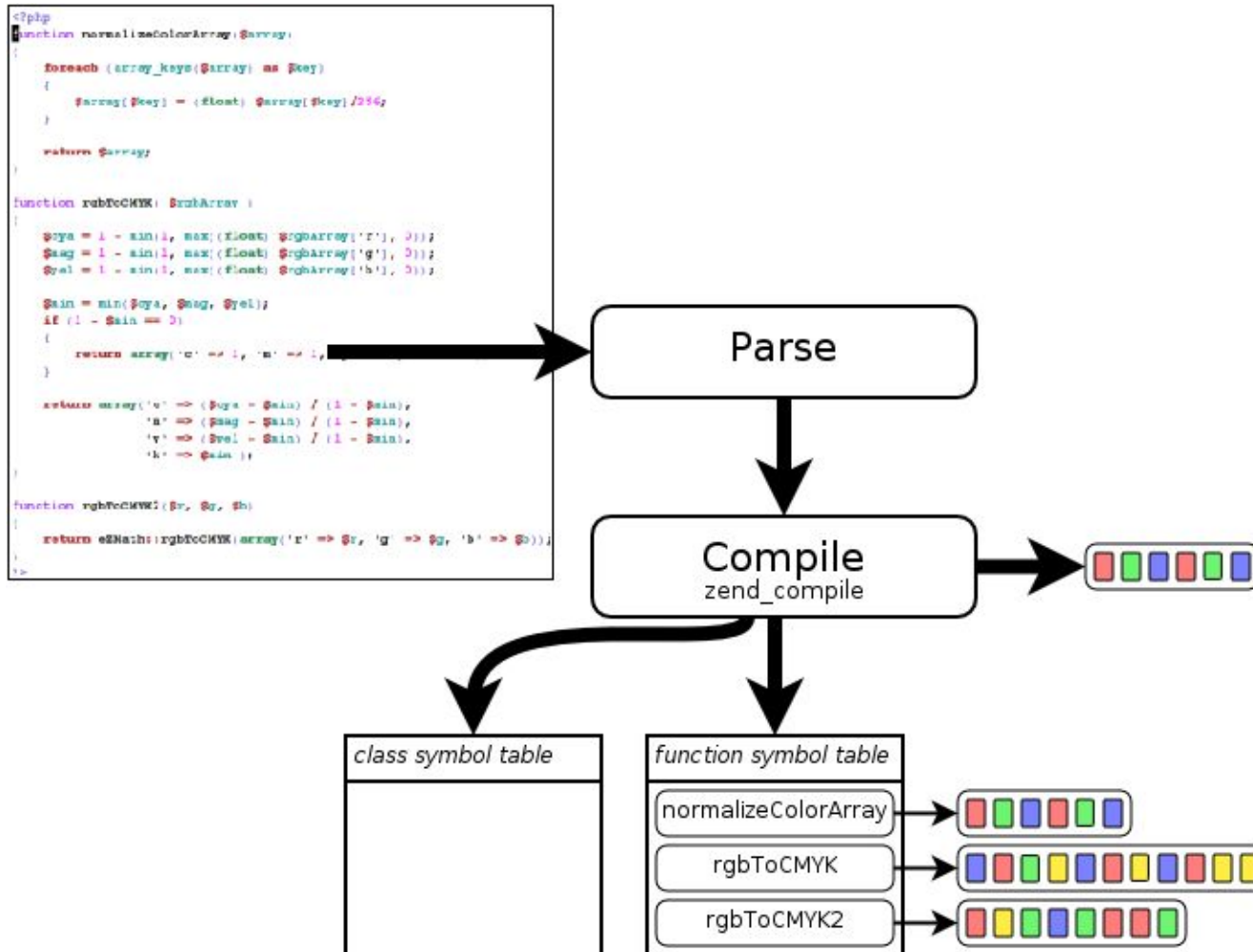
```
<?php
function apply_magic(&$a) {
$a = $a + rand(0,3);
}
?>
```

```
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]
```

# Compiling

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

# Compiling: Diagram



# Compiling: Oparrays

## Oparrays

- Compiled code
- One for every script element

## Opcode

- Basic execution unit
- Two operands
- One result

# 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

```
filename:      compile.php
function name: (null)
number of ops: 11
-----
line # op      operands      code
-----
2 0 INIT_ARRAY  -1,          $cache = array();
1 FETCH_W    #0, 'cache'
2 ASSIGN    #0, -1
4 3 NOP
4 4 FETCH_R   #3, 'argv'   function fibonacci($nr) {
24 5 FETCH_DIM_R #4, $1      echo fibonacci($argv[1])."\n";
6 SEND_VAR  #4
7 DO_FCALL  #5, 'fibonacci', 0
8 CONCAT   #6, $5, '0a'
9 ECHO     #6
26 10 RETURN  1

Function fibonacci:
filename:      compile.php
function name: fibonacci
number of ops: 58
-----
line # op      operands      code
-----
4 0 FETCH_W    #0, 'nr'     function fibonacci($nr) {
1 RCV      #0, 1
5 2 FETCH_W    #1, 'cache'  global $cache;
3 FETCH_W    #2, 'cache'
4 ASSIGN_REF #4, $1
7 5 FETCH_R   #4, 'nr'     if (isset($cache[$nr])) {
6 FETCH_IS  #3, 'cache',
7 FETCH_DIM_IS #5, $3, $4
8 ISSET_ISMPTY #6, $5,
9 JMPZ     #6, >=15
9 10 FETCH_R  #8, 'nr'   return $cache[$nr];
11 FETCH_R  #7, 'cache'
12 FETCH_DIM_R #9, $7, $8
13 RETURN  #9
9 14 JMP      #>15
10 15 FETCH_R  #10, 'nr'   switch ($nr) {
16 BOOL    #11,
11 17 CASE     #11, $10, 0
18 JMPZ    #11, >=21
19 EXIT    #12, 'Invalid $nr\n',
13 20 JMP      #>23
21 CASE   #11, $10, 1
14 22 JMPZ    #11, >=26
23 SWITCH_FREE #10,
24 RETURN  1
15 25 JMP      #>28
26 CASE   #11, $10, 2
27 JMPZ    #11, >=31
28 SWITCH_FREE #10,
29 RETURN  1
17 30 JMP      #>33
31 JMP     #>35
32 BOOL    #11,
18 33 INIT_FCALL_BY_NAME 'fibonacci' $r = fibonacci($nr - 2);
34 FETCH_R  #13, 'nr'
35 SUB     #14, $13, 2
36 SEND_VAL #14
37 DO_FCALL_BY_NAME 'fibonacci' $r = fibonacci($nr - 1);
38 INIT_FCALL_BY_NAME 'fibonacci'
39 FETCH_R  #16, 'nr'
40 SUB     #17, $16, 1
41 SEND_VAL #17
42 DO_FCALL_BY_NAME 'fibonacci' $r = fibonacci($nr);
43 ADD     #19, $15, $18
44 FETCH_W  #12, 'r'
45 ASSIGN  #12, #19
19 46 FETCH_R  #22, 'nr' $cache[$nr] = $r;
47 FETCH_R  #24, 'r'
48 FETCH_W  #21, 'cache'
49 FETCH_DIM_W #23, $21, $22
50 ASSIGN  #23, $24
20 51 FETCH_R  #26, 'r' return $r;
52 SWITCH_FREE #10,
53 RETURN  #26
21 54 JMP      #>35
55 JMPZ    #11, >=32
56 SWITCH_FREE #10,
22 57 RETURN  null
}

End of function fibonacci.
```

# Compiling: Disassembler

Disassembler: vld

Dumps oparray per element:

- main script
- function
- class method

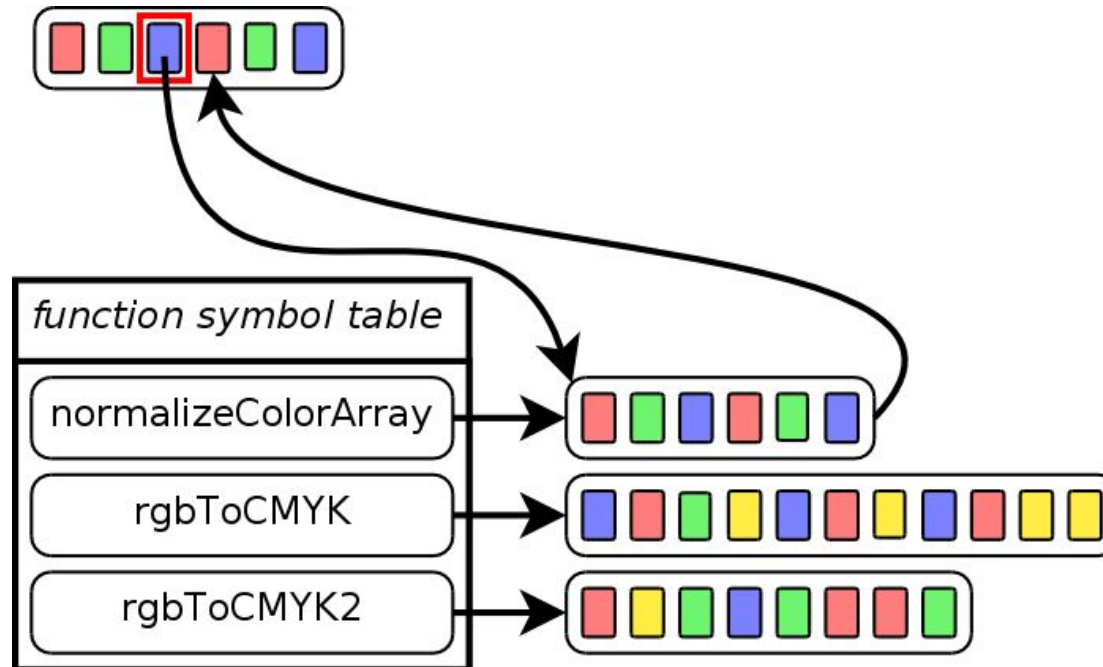
Usage:

```
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 (1)

**test.inc:**

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

**test.php:**

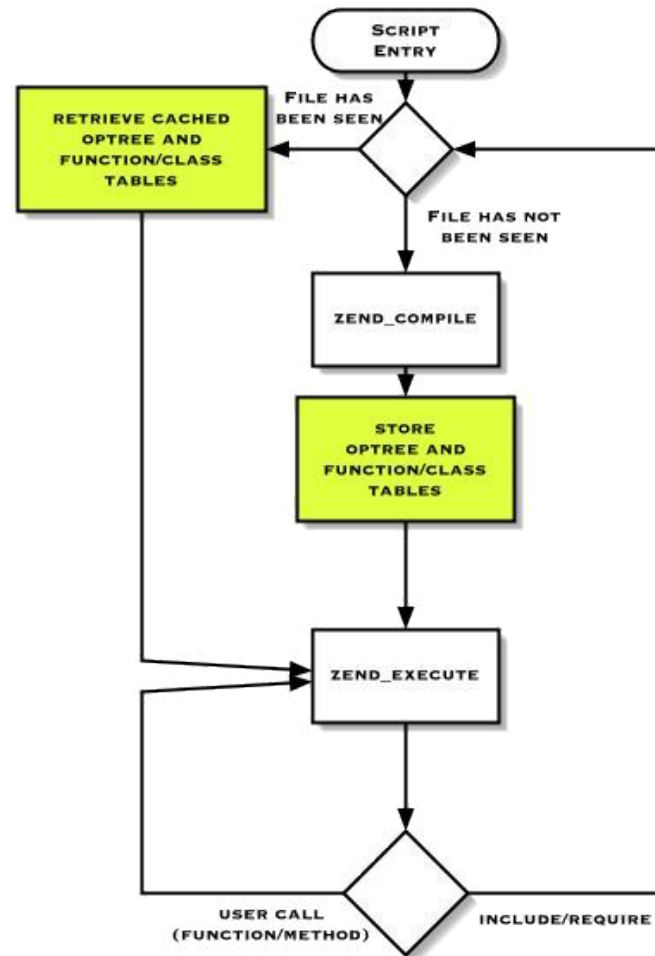
```
<?php
echo "foo\n";
include "test.inc";
foo();
echo "more bar\n";
? >
```

# Executing: Example (2)

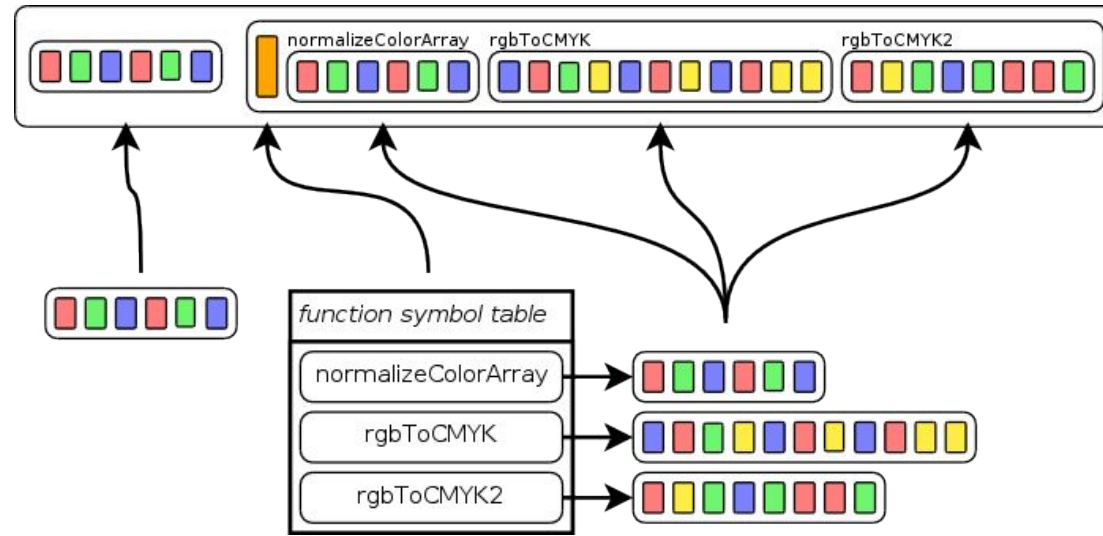
The engine:

- 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

# Compiler Caches



# Compiler Caches

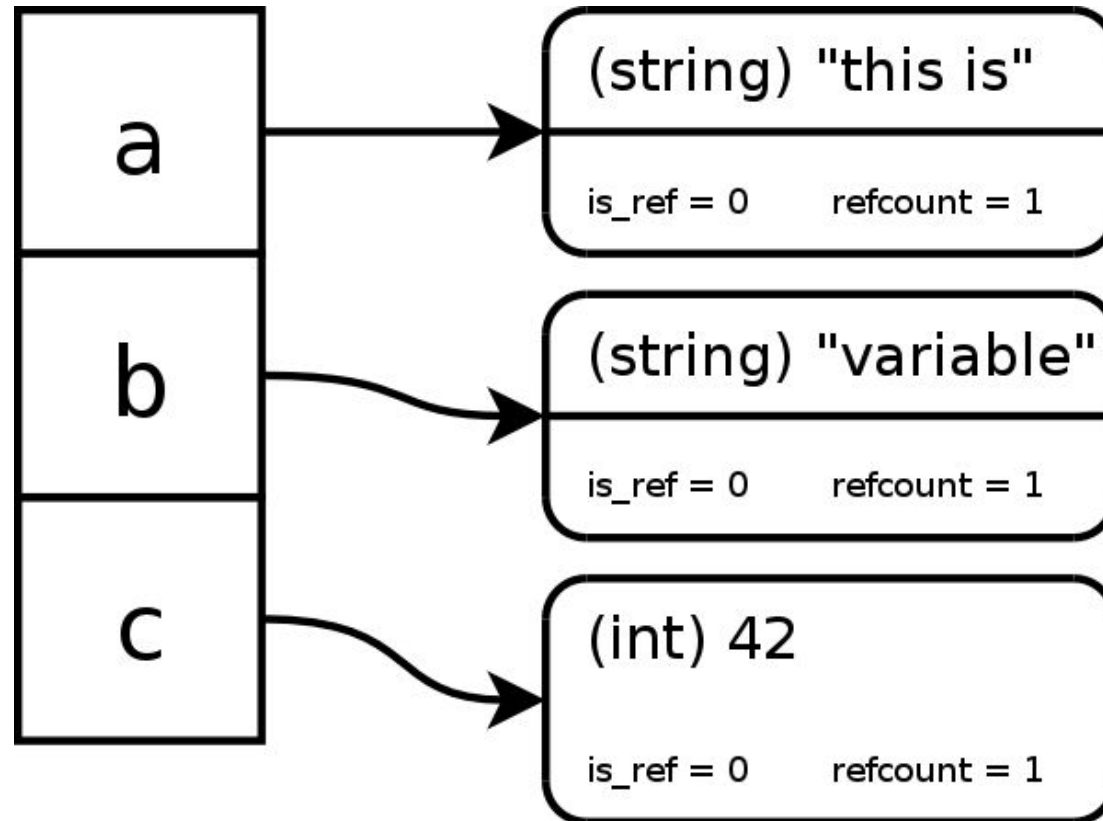


- Serialization into SHM
- Direct execution from SHM (mostly)

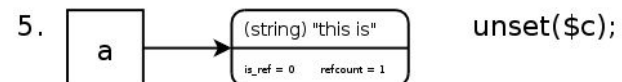
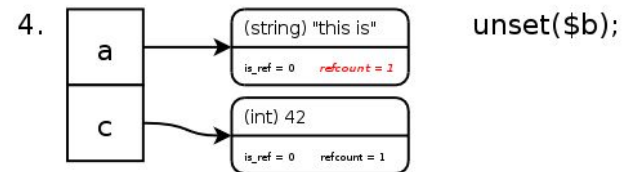
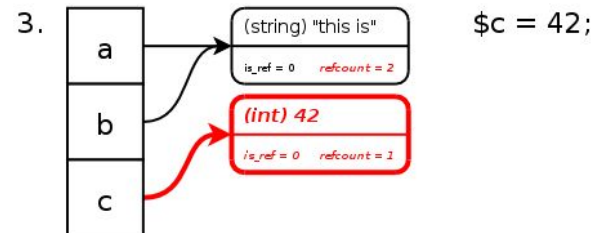
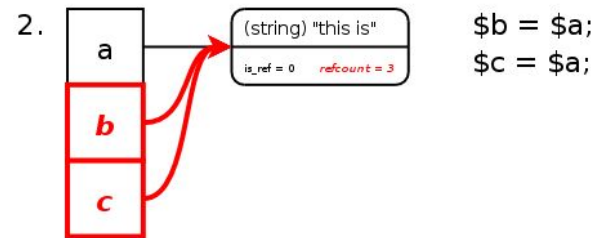
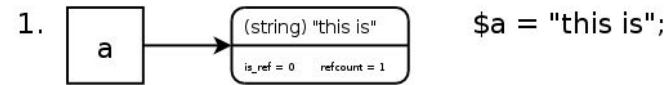
# PHP Accelerators

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

# Variables



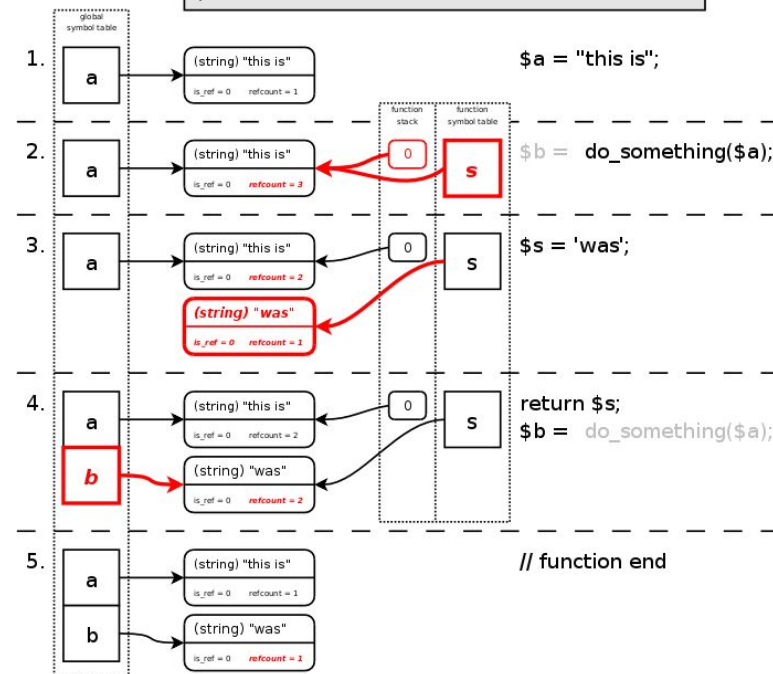
# Variables



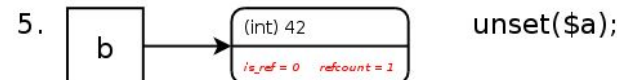
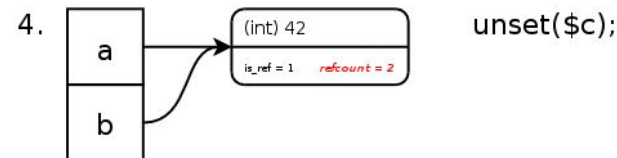
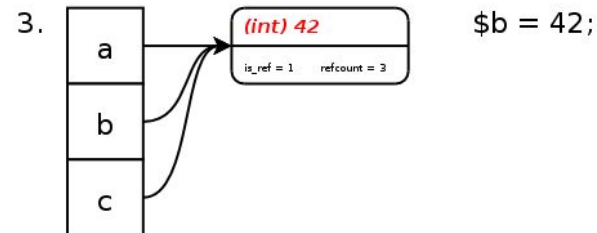
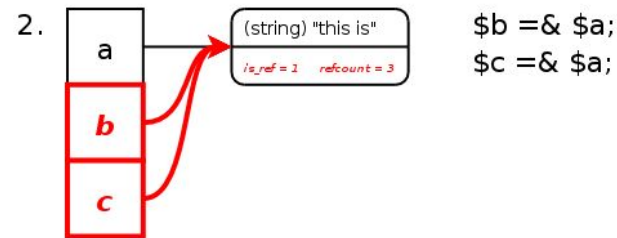
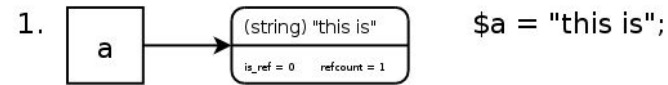
# Variables

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

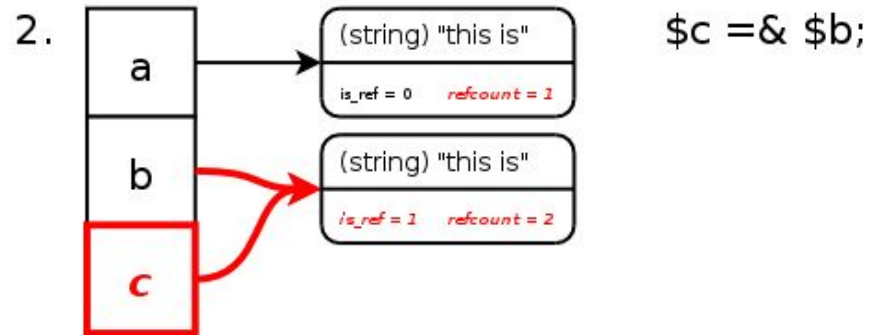
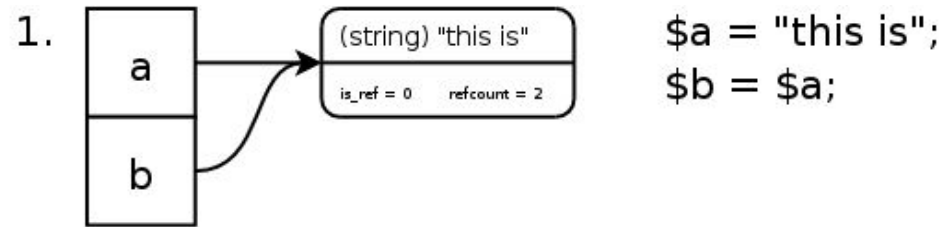
$a = 'this is';
$b = do_something($a);
?>
```



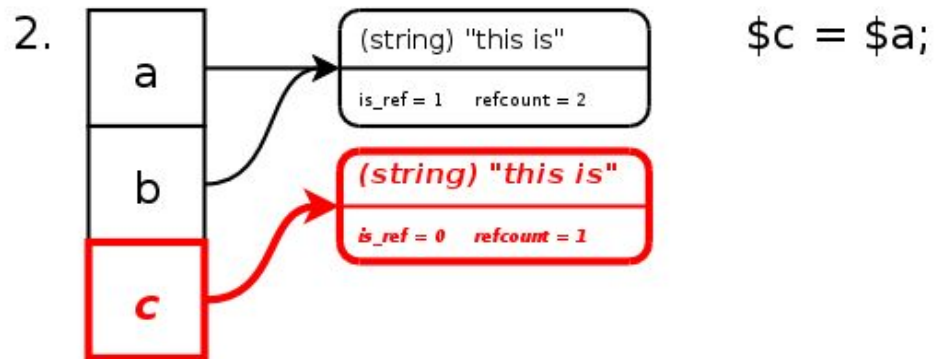
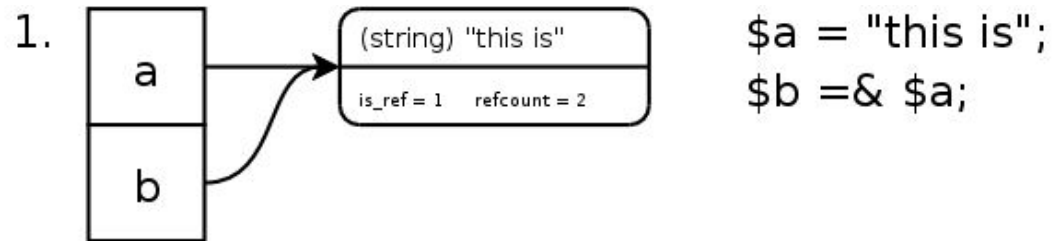
# Variables



# Variables



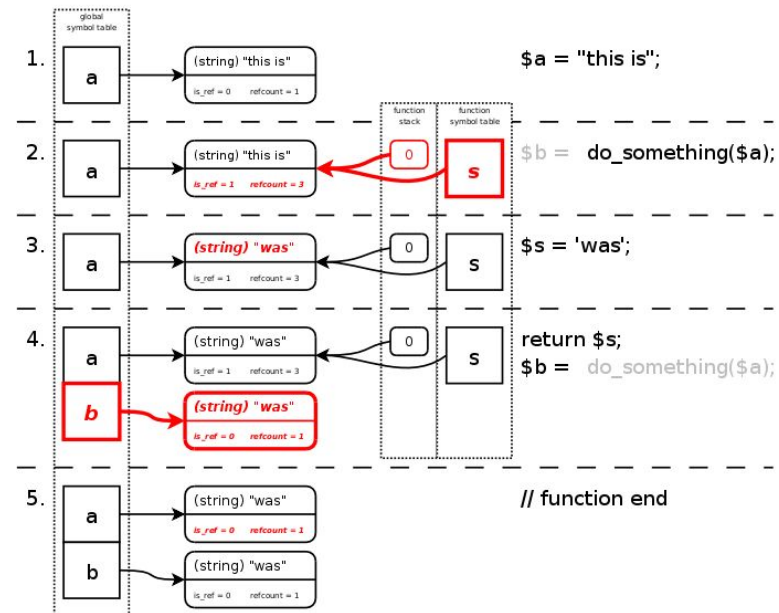
# Variables



# Variables

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

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



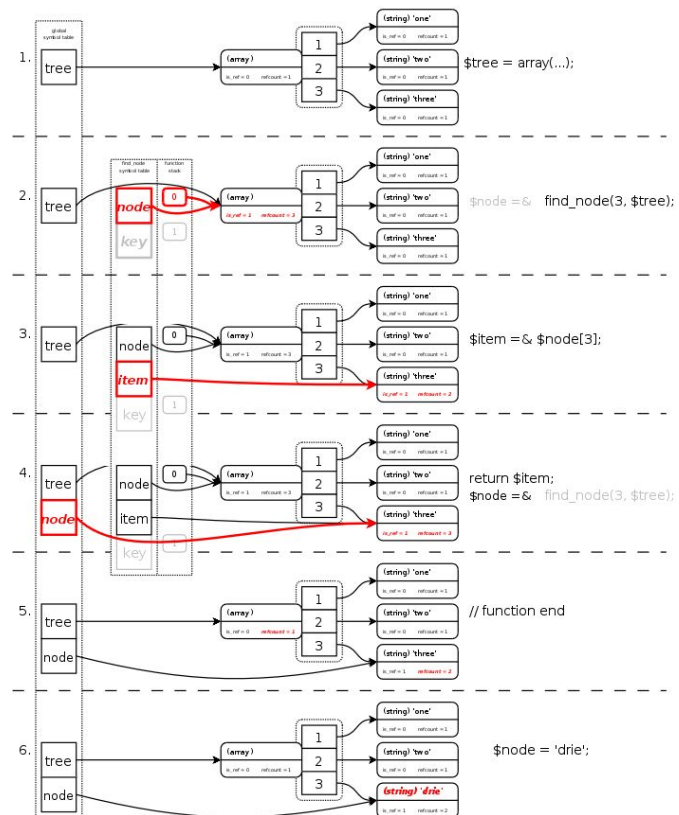
# Variables

```

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

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

```

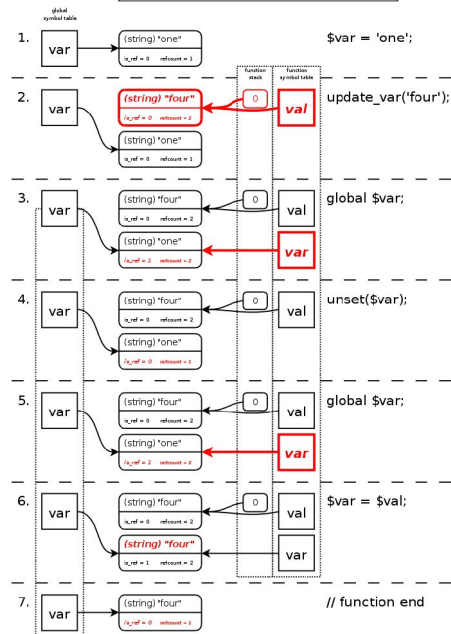


# Variables

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

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

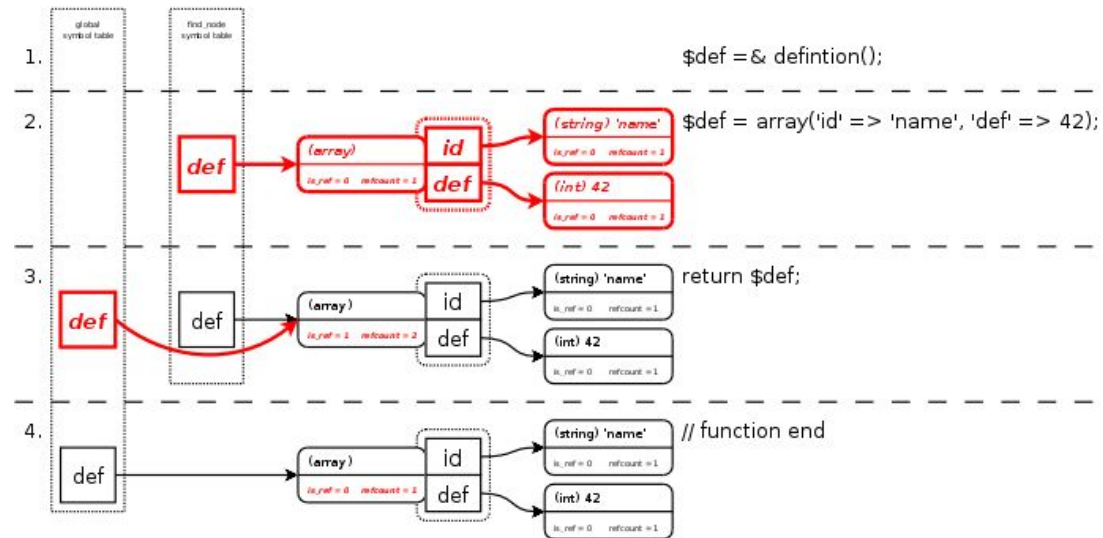
update_tree('four');
```



# Variables

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

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



# By-Reference Example

```
<?php
$var="pippo";

function printme(&$var) {
    echo "{$var}\n";
}

printme(serialize($var));
?>
```

# By-Reference Example

```
<?php
function get_filename_parts($file)
{
    $extension = end(explode(".", $file));
    $filename = substr($file, 0, -(strlen($extension)+1) );
    return array($filename, $extension);
}

var_dump(get_filename_parts(__FILE__));

?>
```

# By-Reference Example

```
<?php
function &dosomething($a)
{
    $b = false;
    return empty($a) ? $b: $a;
}

$a = 42;
$foo =& dosomething($a);
?>
```

# By-Reference Example

```
<?php
$x = current(explode(' ', 'a b'));
echo $x;
?>
```

# Resources

These Slides: <http://pres.derickrethans.nl>

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

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