It's All About the Goto

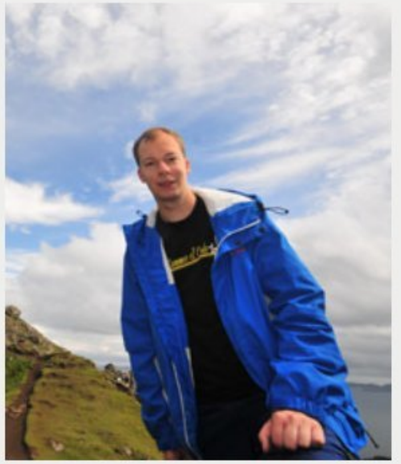
Derick Rethans

derick@php.net—@derickr

https://joind.in/20439

About Me

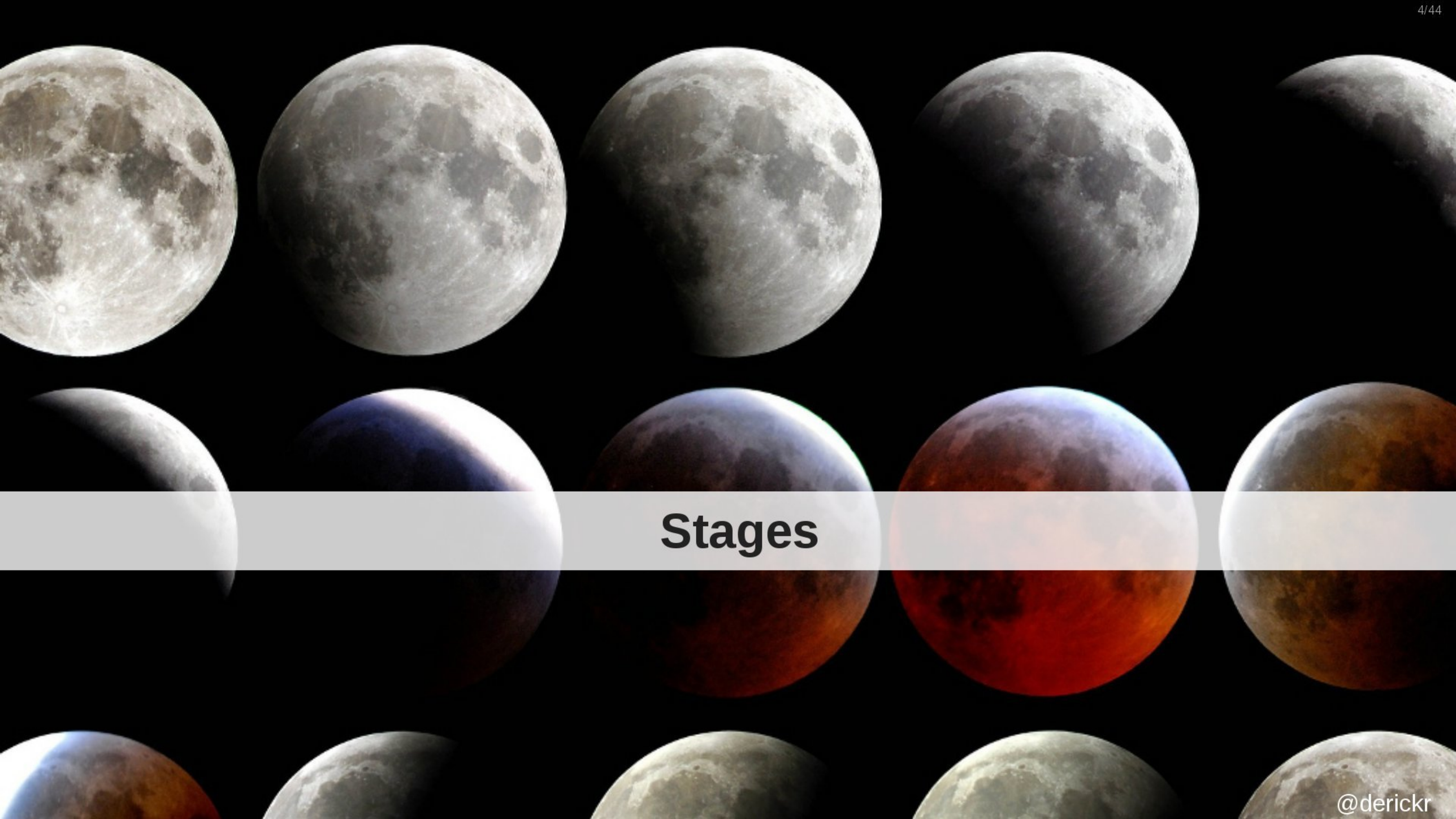
Derick Rethans



- I'm ~~Dutch/British~~ European, living in London
- One of the PHP/HHVM **MongoDB** driver maintainers
- Author of the PHP debugger tool **Xdebug**, PHP's Date/Time/Timezone support, and a bunch of other PHP extensions
- I ♥🌍 maps, I ♥🍺 beer, I ♥🥃 whisky
- twitter: @derickr

Agenda

- Stages
- Conversion Stages
- Conclusion



Stages

Stages

Parse Script

Create Logical Representation

Create Executable Code

Execute Bytecode

Run the Opcodes with the Zend Engine

Parsing

$1LENCE D00600D at English Wikipedia

@derickr

Tokenization

- It's a big state machine starting with `INITIAL`
- States: `INITIAL`, `ST_IN_SCRIPTING`, `ST_DOUBLE_QUOTES`, `ST_NOWDOC`...
- Tokens can change the state:

```
<INITIAL>"<?php"([ \t]|{NEWLINE}) {
    HANDLE_NEWLINE(yytext[yyleng-1]);
    BEGIN(ST_IN_SCRIPTING);
    RETURN_TOKEN(T_OPEN_TAG);
}
```

- No **meaning** is given to these tokens
- PHP comes with a **tokenizer** extension

Tokenization

Example script:

```
<?php
namespace DramIO;
class Whisky
{
    private $name;

    public function __construct($name)
    {
        $this->name = $name;
```

In tokens:

```
T_OPEN_TAG <?php
T_NAMESPACE T_WS T_STRING DramIO ; T_WS
T_CLASS T_WS T_STRING Whisky T_WS
{ T_WS
    T_PRIVATE T_WS T_VARIABLE $name ; T_WS

    T_PUBLIC T_WS T_FUNCTION T_WS T_STRING __construct ( T_VARIABLE $name ) T_WS
    { T_WS
       T_VARIABLE $this T_OBJECT_OPERATOR -> T_STRING name T_WS = T_WS T_VARIABLE $name ;
```

Scanner Rules

```
top_statement:
        statement                               { $$ = $1; }
    |   function_declaration_statement          { $$ = $1; }
    |   class_declaration_statement             { $$ = $1; }
    |   trait_declaration_statement             { $$ = $1; }
…
;

class_declaration_statement:
        class_modifiers T_CLASS { $<num>$ = CG(zend_lineno); }
        T_STRING extends_from implements_list backup_doc_comment '{' class_statement_list
            { $$ = zend_ast_create_decl(ZEND_AST_CLASS, $1, $<num>3, $7, zend_ast_get_str
    |   T_CLASS { $<num>$ = CG(zend_lineno); }
        T_STRING extends_from implements_list backup_doc_comment '{' class_statement_list
            { $$ = zend_ast_create_decl(ZEND_AST_CLASS, 0, $<num>2, $6, zend_ast_get_str(
;

class_modifiers:
        class_modifier                      { $$ = $1; }
    |   class_modifiers class_modifier      { $$ = zend_add_class_modifier($1, $2); }
;

class_modifier:
        T_ABSTRACT      { $$ = ZEND_ACC_EXPLICIT_ABSTRACT_CLASS; }
    |   T_FINAL         { $$ = ZEND_ACC_FINAL; }
;
```

Scanner Rules

- Gives Meaning to Tokens
- Constructs AST through Rules:

```
class_declaration_statement:
        class_modifiers T_CLASS { $<num>$ = CG(zend_lineno); }
        T_STRING extends_from implements_list backup_doc_comment '{' class_statement_
            { $$ = zend_ast_create_decl(ZEND_AST_CLASS, $1, $<num>3, $7, zend_ast_gei
    |
…
;
```

```
  case 167:
#line 493 "/home/derick/dev/php/php-src.git/Zend/zend_language_parser.y"
    { (yyval.num) = CG(zend_lineno); }
    break;

  case 168:
#line 495 "/home/derick/dev/php/php-src.git/Zend/zend_language_parser.y"
    { (yyval.ast) = zend_ast_create_decl(ZEND_AST_CLASS, (yyvsp[(1) - (10)].num), (yy
    break;
```

Abstract Syntax Tree

Abstract Syntax Tree

- An AST describes the structure of the parser script
- Each node is a language construct
- Nested structures are represented through a tree
- Not all the original information from the original source code is kept
- The **php-ast** extension on Nikita Popov's (nikic) GitHub account can visualize them
- Optimisations are possible by modifications of the tree

Abstract Syntax Tree

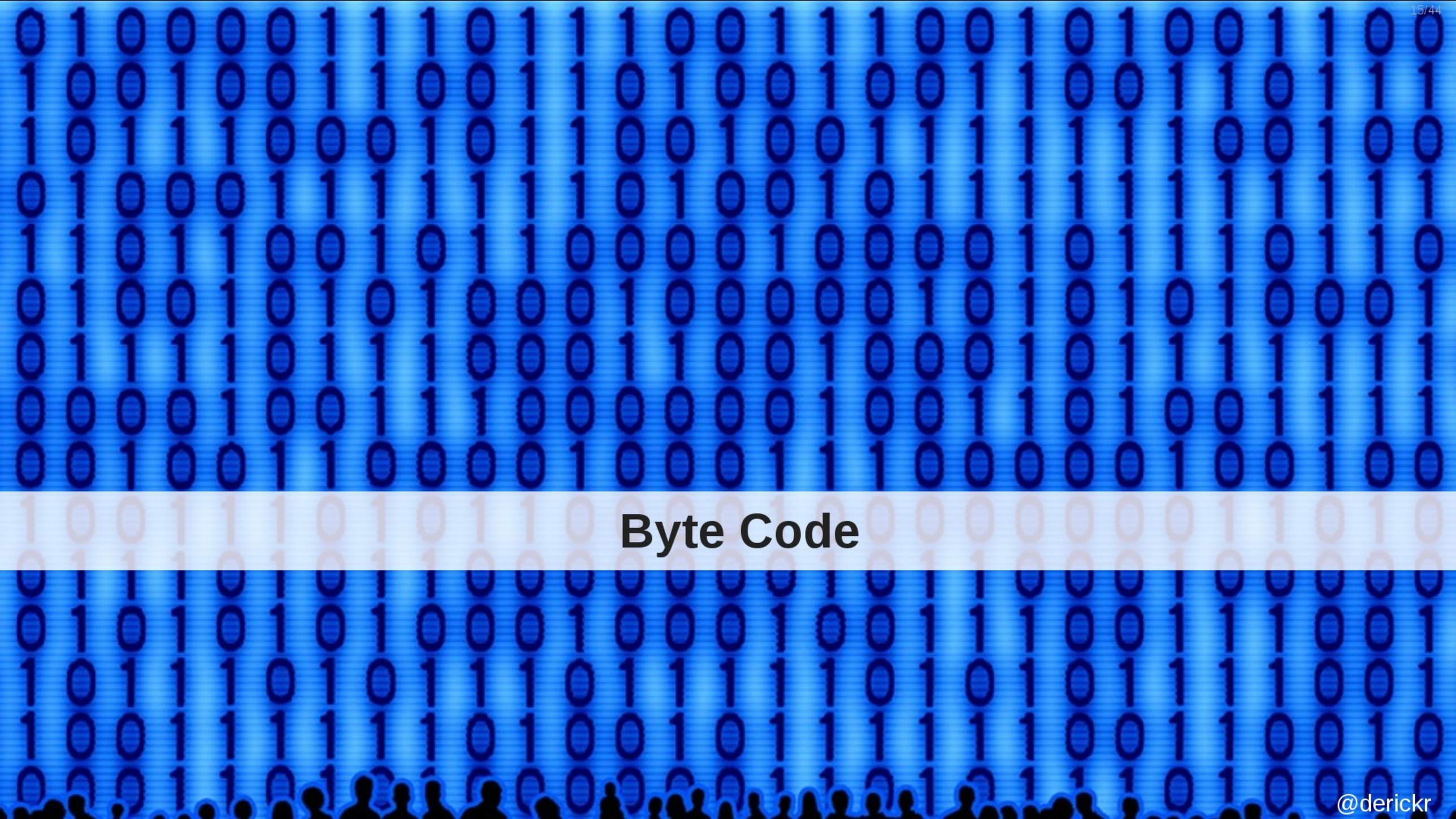
Raw output from `ast\parse_code` function:

```
object(ast\Node)#1 (4) {
  ["kind"]=>
  int(133)
  ["flags"]=>
  int(0)
  ["lineno"]=>
  int(1)
  ["children"]=>
  array(2) {
    [0]=>
    object(ast\Node)#2 (4) {
      ["kind"]=>
      int(257)
      ["flags"]=>
      int(0)
      ["lineno"]=>
      int(2)
      ["children"]=>
      array(1) {
        ["name"]=>
        object(ast\Node)#3 (4) {
          ["kind"]=>
          int(2048)
          ["flags"]=>
          int(2)
          ["lineno"]=>
          int(2)
          ["children"]=>
```

Abstract Syntax Tree (formatted)

```php
public function __construct( $name )
{
    $this->name = $name
}
```

```
1: AST_METHOD
    flags: MODIFIER_PUBLIC (256)
    name: __construct
    params: AST_PARAM_LIST
        0: AST_PARAM
            flags: 0
            type: null
            name: "name"
            default: null
    uses: null
    stmts: AST_STMT_LIST
        0: AST_ASSIGN
            var: AST_PROP
                expr: AST_VAR
                    name: "this"
                prop: "name"
            expr: AST_VAR
                name: "name"
    returnType: null
```



Byte Code

Byte Code

- In PHP also called Opcodes
- Each function, method, or main body is represented by an Oparray
- Each Oparray contains Opcodes with instructions for the Zend Engine
- The Zend Engine executes each opcode in turn
- Very similar to Assembler instructions
- You can visualize them with the PECL **vld** extension

AST is converted to bytecode

```
1: AST_METHOD
    flags: MODIFIER_PUBLIC (256)
    name: __construct
    params: AST_PARAM_LIST
        0: AST_PARAM
            name: "name"
    stmts: AST_STMT_LIST
        0: AST_ASSIGN
            var: AST_PROP
                expr: AST_VAR
                    name: "this"
                prop: "name"
            expr: AST_VAR
                name: "name"
```

```
compiled vars:  !0 = $name
line     #* E I O op                                   fetch          ext  return  operands
-------------------------------------------------------------------------------------------
   8     0  E >   EXT_NOP
         1        RECV                                                    !0
  10     2        EXT_STMT
         3        ASSIGN_OBJ                                                   'name'
         4        OP_DATA                                                      !0
  11     5        EXT_STMT
         6      > RETURN                                                       null
```

Jumps

Thor-Wiggo Skille (http://www.nfk.no/Artikkel.aspx?AId=25071&back=1&MId1=4557&MId2=&MId3=&)

IF

```
if ( $a == 42 )
{
    echo "To life, the universe, and everything.\n";
}
```

```
AST_STMT_LIST
    0: AST_IF
        0: AST_IF_ELEM
            cond: AST_BINARY_OP
                flags: BINARY_IS_EQUAL (17)
                left: AST_VAR
                    name: "a"
                right: 42
            stmts: AST_STMT_LIST
                0: AST_ECHO
                    expr: "To life, the universe, and everything.
```

```
compiled vars:  !0 = $a
line     #* E I O op            return  operands
-------------------------------------------------------------------------------
   2     0  E >   EXT_STMT
         1        IS_EQUAL      ~1      !0, 42
         2      > JMPZ                  ~1, ->5
   4     3    >   EXT_STMT
         4        ECHO                  'To+life2C+the+universe2C+and+everything.%0A'
         5    > > RETURN                1
```

IF with ELSE

```
if ( $a == 3.1415926 ) {
    echo "Circles";
} else {
    echo "Squares";
}
```

```
AST_STMT_LIST
    0: AST_IF
        0: AST_IF_ELEM
            cond: AST_BINARY_OP
                flags: BINARY_IS_EQUAL (17)
                left: AST_VAR
                    name: "a"
                right: 3.1415926
            stmts: AST_STMT_LIST
                0: AST_ECHO
                    expr: "Circles"
        1: AST_IF_ELEM
            cond: null
            stmts: AST_STMT_LIST
                0: AST_ECHO
                    expr: "Squares"
```

```
compiled vars:  !0 = $a
line     #* E I O op              return  operands
-------------------------------------------------------------------------------
   2     0  E >   EXT_STMT
         1        IS_EQUAL        ~1      !0, 3.14159
         2      > JMPZ                    ~1, ->6
   3     3    >   EXT_STMT
         4        ECHO                    'Circles'
         5      > JMP                     ->8
   5     6    >   EXT_STMT
         7        ECHO                    'Squares'
         8    > > RETURN                  1
```


Rings
Les Molunes
D 991
D 1005
Gex
Vesancy
Chavannes-de-Bogis
Coppet
10
Founex
A1
Grilly
Cessy
Sauverny
Chavannes-des-Bois
Commugny
Coppet
Tannay
Mies
Échenevex
Lélex
D 984c
Versonnex
Ségny
Maconnex
1
D 991
Chevry
Flies
Ornex
Versoix
Pregnin
Prévessin-Moëns
Ferney-Voltaire
Genthod
Échangeur du Vengeron
9
Bellevue
Versoix
Le Grand-Saconnex / Ferney
7
Sergy
Saint-Genis-Pouilly
Chambésy
Genève Aéroport
© OpenStreetMap contributors
@derickr

FOR

```php
<?php
for ( $i = 0; $i < 42; ++$i ) {
    echo $i;
}
```

```
        init: AST_EXPR_LIST
            0: AST_ASSIGN
                var: AST_VAR
                    name: "i"
                expr: 0
        cond: AST_EXPR_LIST
            0: AST_BINARY_OP
                flags: BINARY_IS_SMALLER (19)
                left: AST_VAR
                    name: "i"
                right: 42
        loop: AST_EXPR_LIST
            0: AST_PRE_INC
                var: AST_VAR
                    name: "i"
        stmts: …
```

```
compiled vars:  !0 = $i
line     #* E I O op                           fetch          ext  return  operands
   2     0  E >   EXT_STMT
         1        ASSIGN                                                 !0, 0
         2      > JMP                                                    ->6
   4     3    >   EXT_STMT
         4        ECHO                                                   !0
   2     5        PRE_INC                                                !0
         6    >   IS_SMALLER                                    ~3       !0, 42
         7        EXT_STMT
         8      > JMPNZ                                                  ~3, ->3
         9    > > RETURN                                                 1
```

FOR (rewritten)

```php
for ( $i = 0; $i < 42; ++$i ) {
    echo $i;
}
```

```php
        $i = 0;
        goto COND;
STMT:
        echo $i;
        ++$i;
COND:
        $_ = $i < 42;
        if ( ! $_ ) {
                goto STMT;
        }
```

```
compiled vars:  !0 = $i
line     #* E I O op                                fetch          ext  return  operands
   2     0  E >   EXT_STMT
         1        ASSIGN                                                      !0, 0
         2      > JMP                                                         ->6
   4     3    >   EXT_STMT
         4        ECHO                                                        !0
   2     5        PRE_INC                                                     !0
         6    >   IS_SMALLER                                        ~3       !0, 42
         7        EXT_STMT
         8      > JMPNZ                                                       ~3, ->3
         9    > > RETURN                                                      1
```

WHILE

```php
<?php
$i = 10;
while ( --$i ) {
    echo $i;
}
```

```
compiled vars:  !0 = $i
line     #* E I O op                           fetch          ext  return  operands
   2     0  E >   EXT_STMT
         1        ASSIGN                                                   !0, 10
   3     2        EXT_STMT
         3      > JMP                                                      ->6
   4     4    >   EXT_STMT
         5        ECHO                                                     !0
   3     6    >   PRE_DEC                                        $2       !0
         7      > JMPNZ                                                    $2, ->4
         8    > > RETURN                                                   1
```

DO .. WHILE

```php
<?php
$i = 10;
do {
    echo $i;
} while ( --$i );
```

```
compiled vars:  !0 = $i
line     #* E I O op                           fetch          ext  return  operands
   2     0  E >   EXT_STMT
         1        ASSIGN                                                   !0, 10
   3     2        NOP
   4     3    >   EXT_STMT
         4        ECHO                                                     !0
   5     5        PRE_DEC                                        $2        !0
         6      > JMPNZ                                                    $2, ->3
         7    > > RETURN                                                   1
```

FOREACH

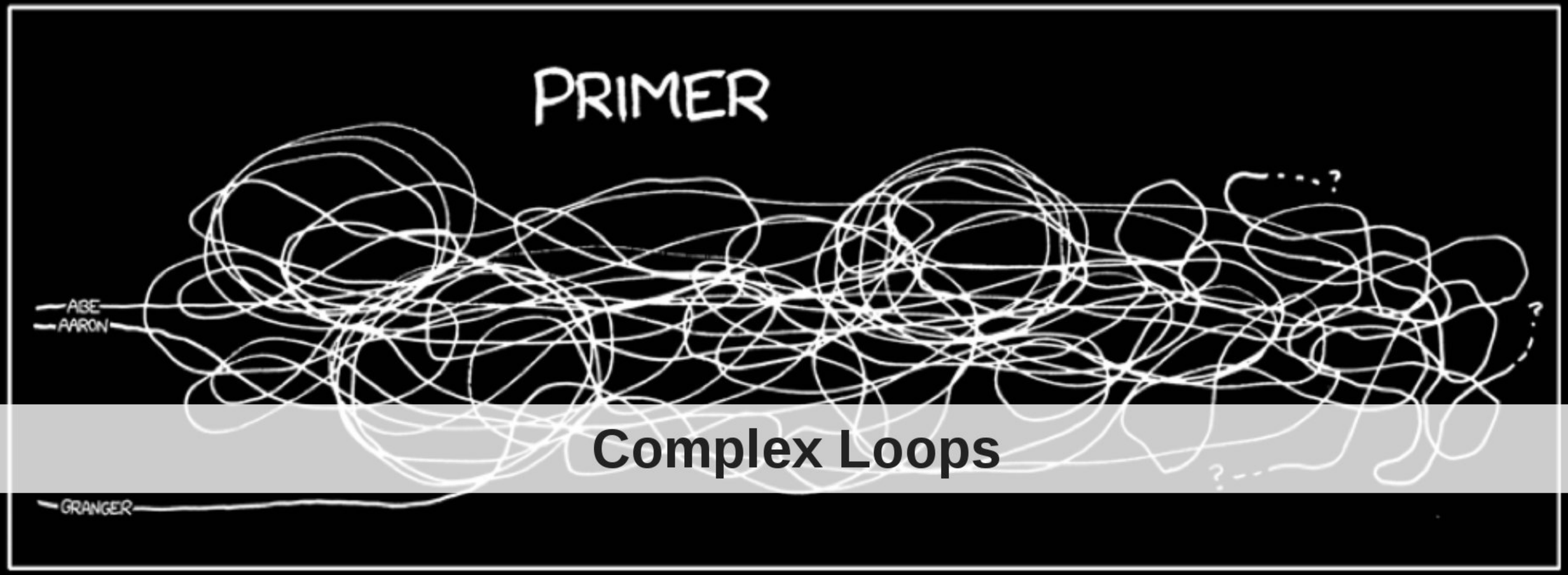
```php
<?php
$a = [ 2.7, 4.9 ];
foreach ( $a as $key => $value ) {
    echo $key, $value;
}
```

```
compiled vars:  !0 = $a, !1 = $value, !2 = $key
line     #* E I O op                            fetch          ext  return  operands
   2      0  E >   EXT_STMT
          1        ASSIGN                                                  !0, <array>
   3      2        EXT_STMT
          3      > FE_RESET_R                                      $4      !0, ->11
          4    > > FE_FETCH_R                                      ~5      $4, !1, ->11
          5    >   ASSIGN                                                  !2, ~5
   4      6        EXT_STMT
          7        ECHO                                                    !2
          8        EXT_STMT
          9        ECHO                                                    !1
         10      > JMP                                                     ->4
         11    >   FE_FREE                                                $4
         12      > RETURN                                                  1
```

Complex Loops

```
<?php
$a = [ 2.7, 4.9 ];
foreach ( $a as $key => $value ) {
        if ( $value < 3 ) {
                echo $key;
        } else {
                if ( $key > 0 ) {
                        echo $value;
                }
        }
}
```

```
line     #* E I O op                           fetch          ext  return  operands
   2      0  E >   ASSIGN                                               !0, <array>
   3      1      > FE_RESET_R                                    $4      !0, ->12
          2    > > FE_FETCH_R                                    ~5      $4, !1, ->12
          3    >   ASSIGN                                               !2, ~5
   4      4        IS_SMALLER                                    ~7      !1, 3
          5      > JMPZ                                                 ~7, ->8
   5      6    >   ECHO                                                 !2
          7      > JMP                                                  ->11
   7      8    >   IS_SMALLER                                    ~8      0, !2
          9      > JMPZ                                                 ~8, ->11
   8     10    >   ECHO                                                 !1
         11    > > JMP                                                  ->2
         12    >   FE_FREE                                             $4
         13      > RETURN                                               1
```

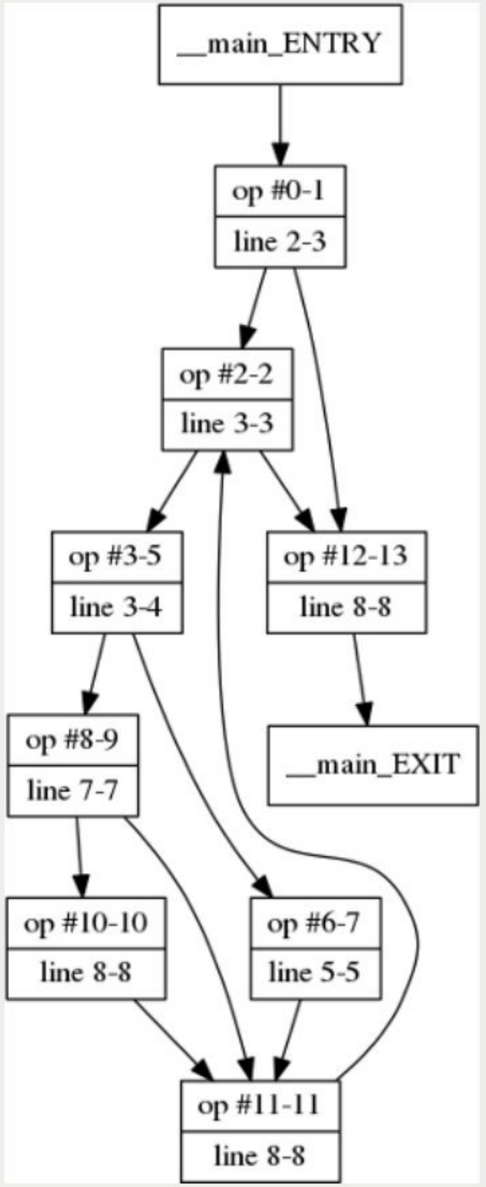


Complex Loops

Complex Loops

```php
<?php
$a = [ 2.7, 4.9 ];
foreach ( $a as $key => $value ) {
        if ( $value < 3 ) {
                echo $key;
        } else {
                if ( $key > 0 ) {
                        echo $value;
                }
        }
}
```

```
line     #* E I O op              return   operands
   2      0  E >   ASSIGN                   !0, <array>
   3      1      > FE_RESET_R     $4       !0, ->12
          2    > > FE_FETCH_R     ~5       $4, !1, ->12
          3    >   ASSIGN                   !2, ~5
   4      4        IS_SMALLER     ~7       !1, 3
          5      > JMPZ                     ~7, ->8
   5      6    >   ECHO                     !2
          7      > JMP                      ->11
   7      8    >   IS_SMALLER     ~8       0, !2
          9      > JMPZ                     ~8, ->11
   8     10    >   ECHO                     !1
         11    > > JMP                      ->2
         12    >   FE_FREE                  $4
         13      > RETURN                   1
```

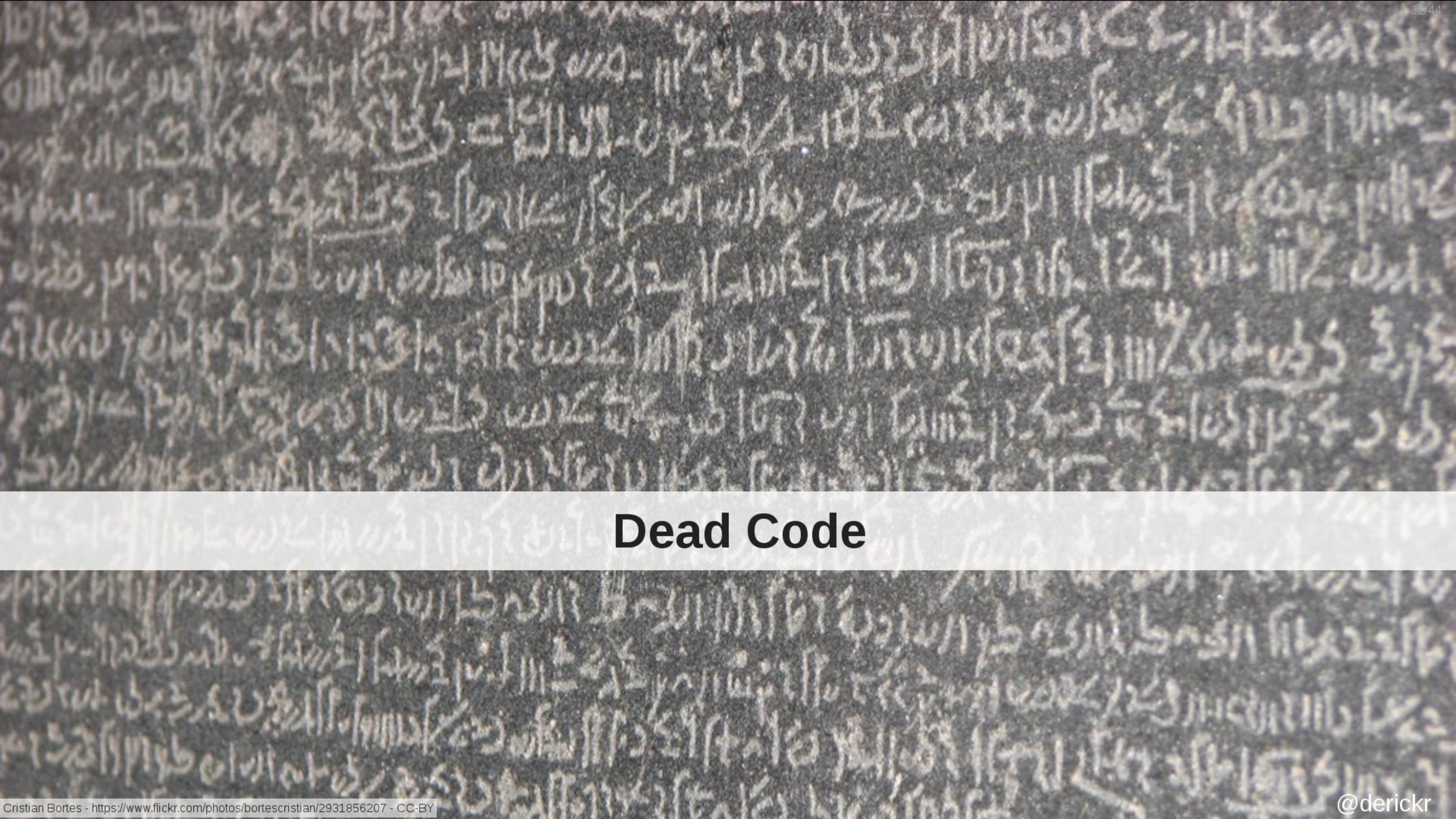


Exceptions (Try/Catch)

TRY ... CATCH ... FINALLY

```php
<?php
try {
    echo "in try";
} catch ( Exception1 $e ) {
    echo "exception 1 thrown";
} catch ( Exception2 $e ) {
    echo "exception 2 thrown";
} finally {
    echo "in finally";
}
echo "after";
```

```
line     #* E I O op            return  operands
   3      0  E >   ECHO                  'in+try'
          1      > JMP                   ->7
   4      2  E > > CATCH                 'Exception1', !0, ->5
   5      3    >   ECHO                  'exception+1+thrown'
          4      > JMP                   ->7
   6      5  E > > CATCH                 'Exception2', !0
   7      6    >   ECHO                  'exception+2+thrown'
   8      7    > > FAST_CALL             ->9
          8    > > JMP                   ->11
   9      9    >   ECHO                  'in+finally'
         10      > FAST_RET
  11     11    >   ECHO                  'after'
         12      > RETURN                1
```



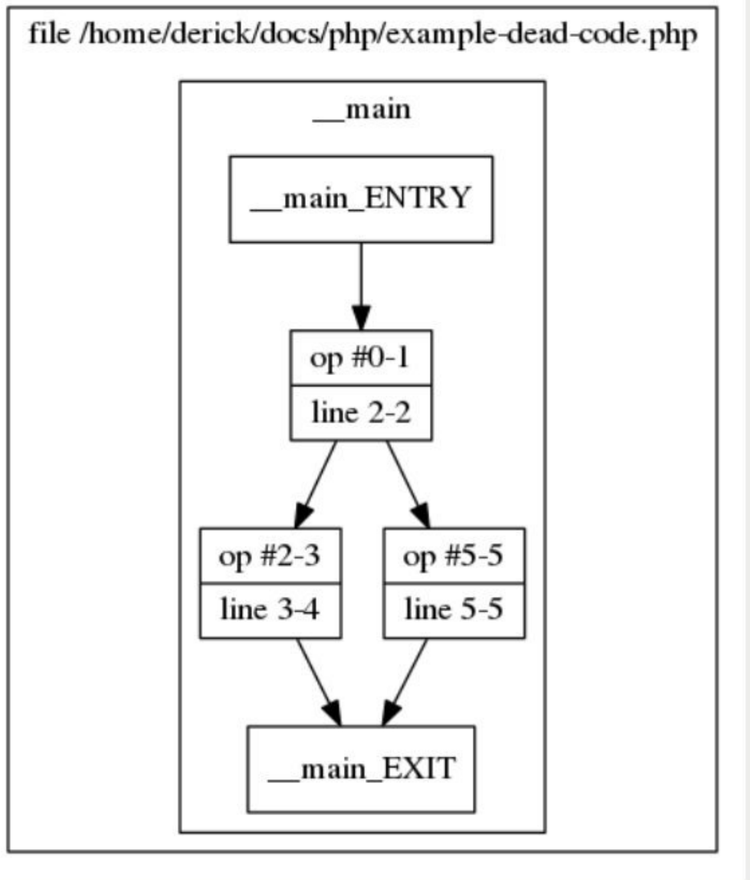
Dead Code

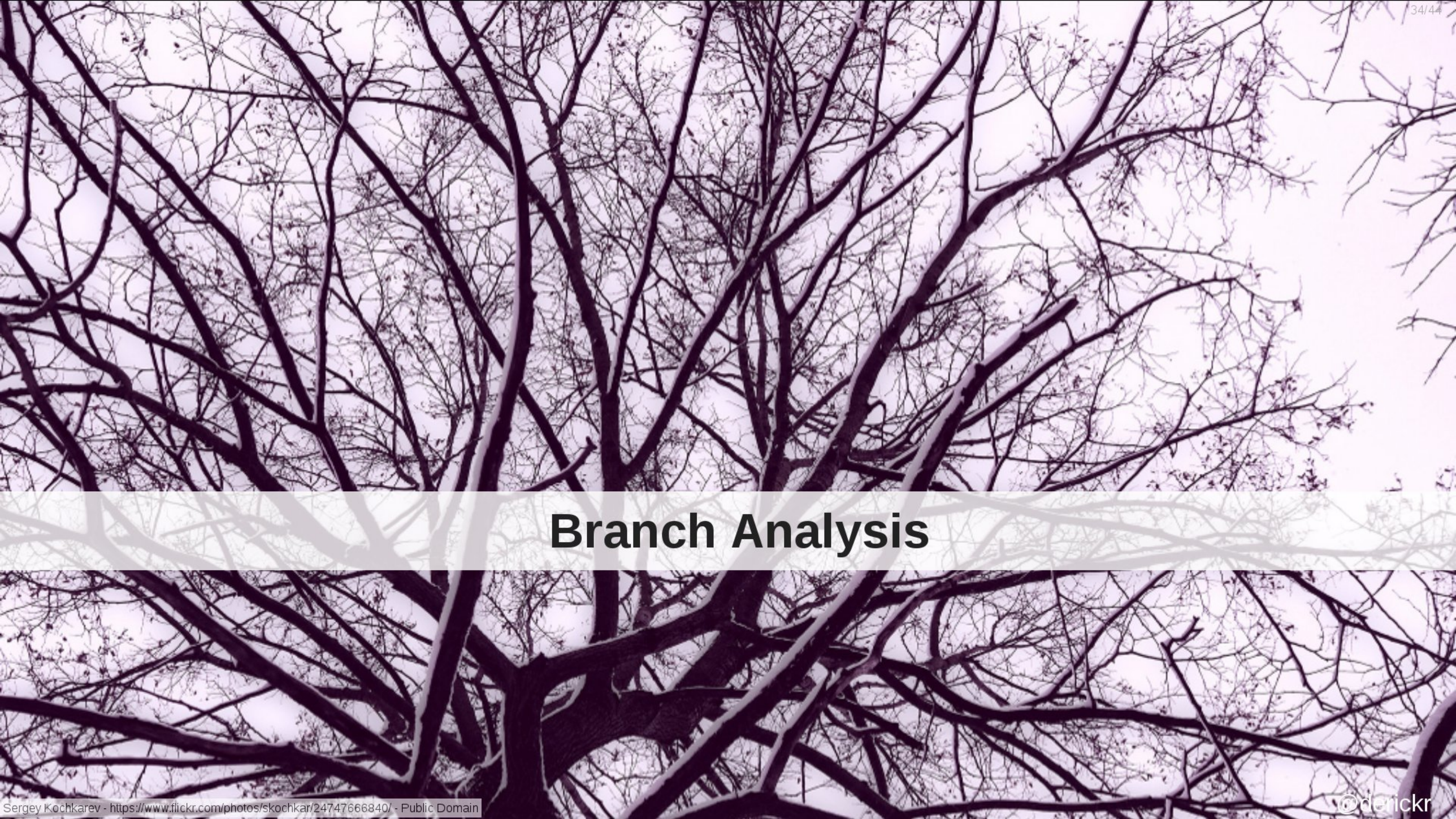
Dead Code

```php
<?php
if ( $a < 42 ) {
        echo "fourty";
        return;
        echo "two";
}
```

```
line   #* E I O op              return  operands
   2   0  E >   IS_SMALLER     ~1      !0, 42
       1      > JMPZ                    ~1, ->5
   3   2    >   ECHO                    'fourty
   4   3      > RETURN                  null
   5   4*       ECHO                    'two'
       5    > > RETURN                  1
```

- Follow all branches
- Find unreached opcodes





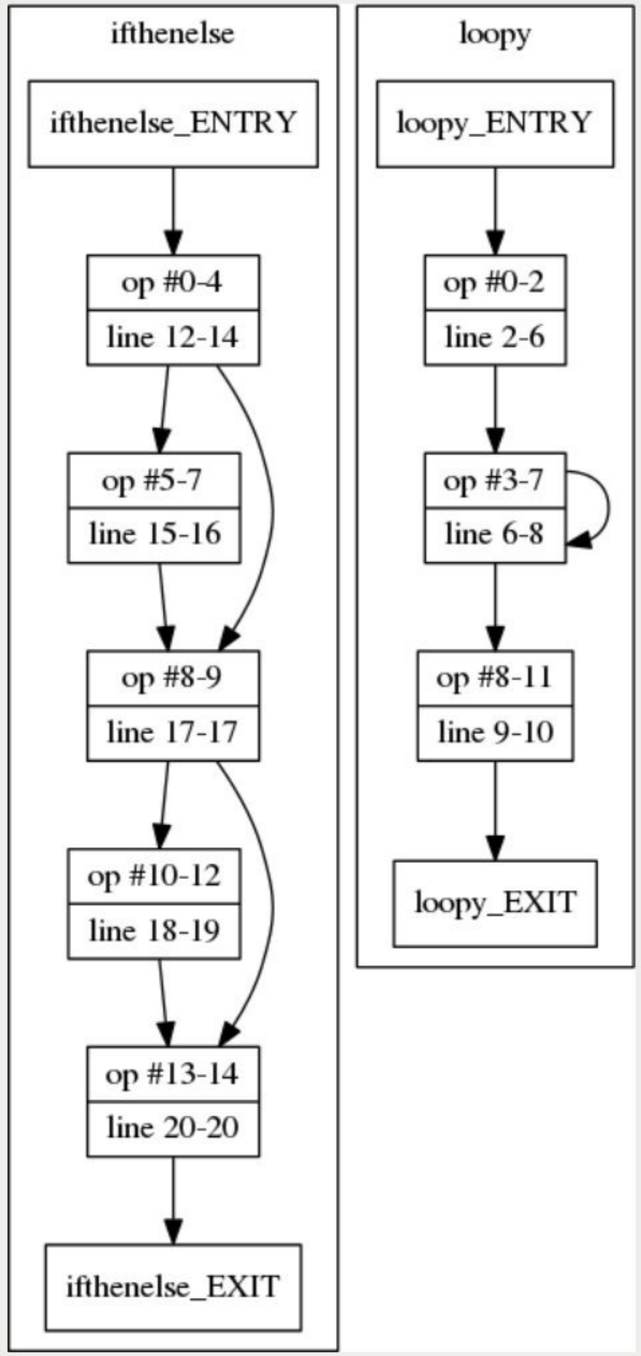
Branch Analysis

```php
<?php
function loopy($i)
{
    do
    {
        echo $i, ' ';

    } while( $i-- );
    echo "\n";
}

function ifthenelse( $a, $b )
{
    if ($a) {
        echo "A HIT\n";
    }
    if ($b) {
        echo "B HIT\n";
    }
}

loopy(0);
ifthenelse( true, false );
ifthenelse( false, true );
?>
```



Xdebug code coverage

```php
<?php
function ifthenelse( $a, $b )
{
    if ($a) {
        echo "A HIT\n";
    }
    if ($b) {
        echo "B HIT\n";
    }
}
```

```php
<?php
require 'vendor/autoload.php';
include 'test.php';

$coverage = new PHP_CodeCoverage;

$coverage->start('coverage1');
ifthenelse( true, false );
$coverage->stop();

$coverage->start('coverage2');
ifthenelse( false, true );
$coverage->stop();

$writer = new PHP_CodeCoverage_Report_HTML;
$writer->process($coverage, '/tmp/code-coverage-report');
```

Xdebug code coverage output

/home/httpd/html/test/xdebug/code-coverage / test.php

	Code Coverage							
	Classes and Traits			Functions and Methods			Lines		
Total						CRAP		100.00%	7 / 7
ifthenelse($a, $b)					100.00%	1 / 1	0		100.00%	7 / 7

```
 1 <?php
 2 function ifthenelse( $a, $b )
 3 {
 4     if ($a) {
 5         echo "A HIT\n";
 6     }
 7     if ($b) {
 8         echo "B HIT\n";
 9     }
10 }
11 ?>
```

Legend

Executed **Not Executed** **Dead Code**

Generated by PHP_CodeCoverage 3.0-dev using PHP 5.4.33-dev at Wed Oct 15 18:27:29 BST 2014.

Xdebug branch coverage

```
<?php
include 'dump-branch-coverage.inc';
include 'test.php';

xdebug_start_code_coverage(
        XDEBUG_CC_UNUSED |
        XDEBUG_CC_DEAD_CODE |
        XDEBUG_CC_BRANCH_CHECK // ← new magic sauce
);

ifthenelse( true, false );
ifthenelse( false, true );

xdebug_stop_code_coverage(false);

$c = xdebug_get_code_coverage();
dump_branch_coverage($c);
?>
```

Xdebug branch coverage output

```
A HIT
ifthenelse
- branches
  - 00; OP: 00-04; line: 02-04 HIT; out1: 05 HIT; out2: 08 HIT
  - 05; OP: 05-07; line: 05-06 HIT; out1: 08 HIT
  - 08; OP: 08-09; line: 07-07 HIT; out1: 10 HIT; out2: 13 HIT
  - 10; OP: 10-12; line: 08-09 HIT; out1: 13 HIT
  - 13; OP: 13-14; line: 10-10 HIT; out1: EX  X
- paths
  - 0 5 8 10 13:  X
  - 0 5 8 13: HIT
  - 0 8 10 13: HIT
  - 0 8 13:  X
```

Xdebug branch coverage display

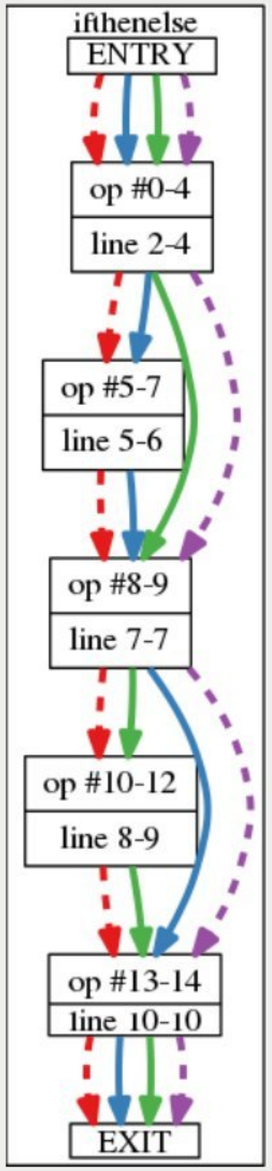
test.php

```
 1 <?php
 2 function ifthenelse( $a, $b )
 3 {
 4     if ($a) {
 5         echo "A HIT\n";
 6     }
 7     if ($b) {
 8         echo "B HIT\n";
 9     }
10 }
11 ?>
```

branch.php

```
…
xdebug_start_code_coverage(
        XDEBUG_CC_UNUSED | XDEBUG_CC_DEAD_CODE | XDEBUG_CC_BRANCH_CHECK
);

ifthenelse( true, false );
ifthenelse( false, true );
…
```



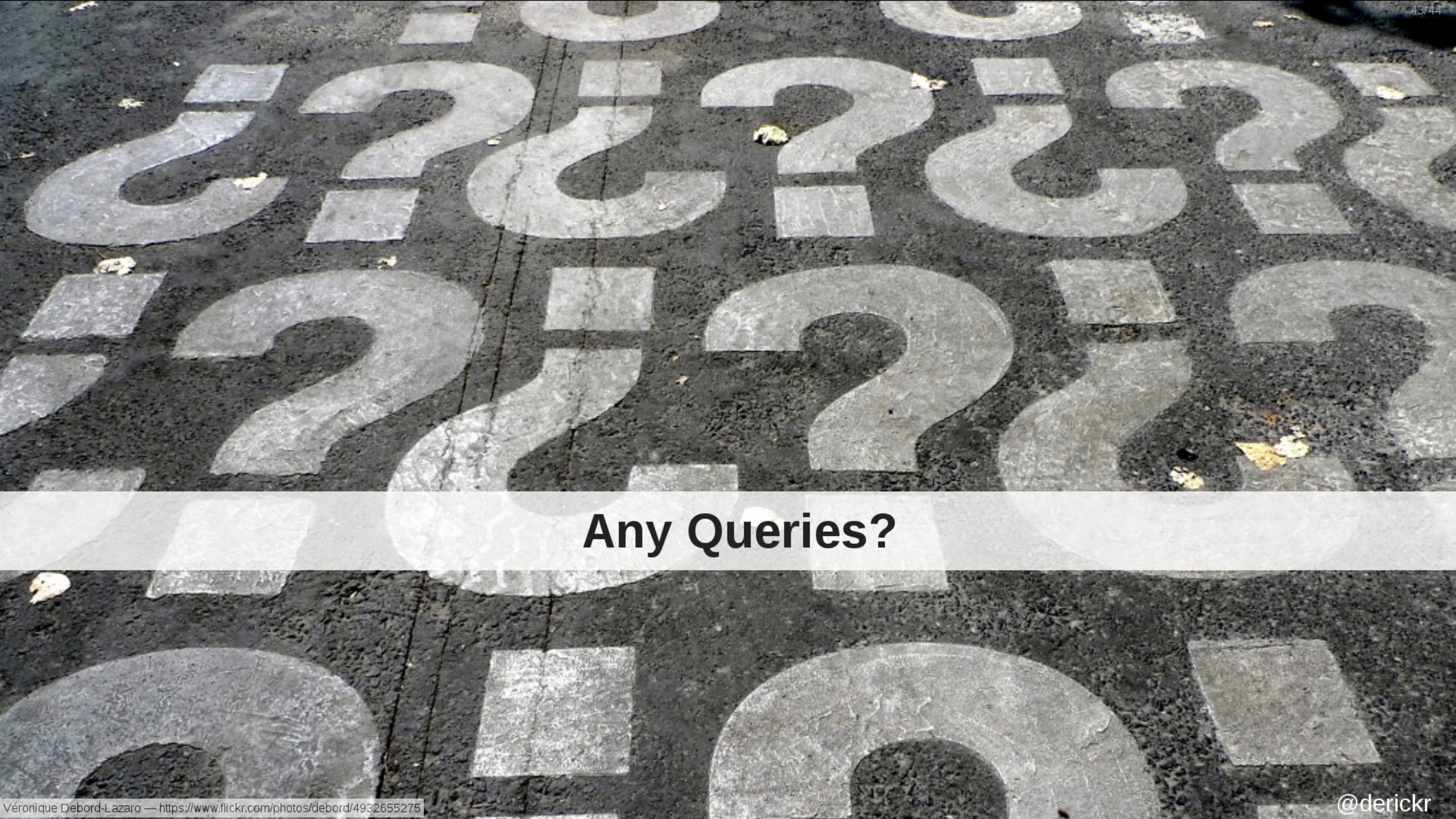
Recap

Recap

- Stages:
  Code → Tokens → Parsing → AST → Byte Code
- All looping structures are jumps
- Code analysis for Fun and Profit

Tools

- PHP's tokenizer: http://php.net/tokenizer
- Nikita's AST extension: https://github.com/nikic/php-ast
- VLD: https://pecl.php.net/vld



Any Queries?

https://joind.in/20439

derick@php.net—@derickr