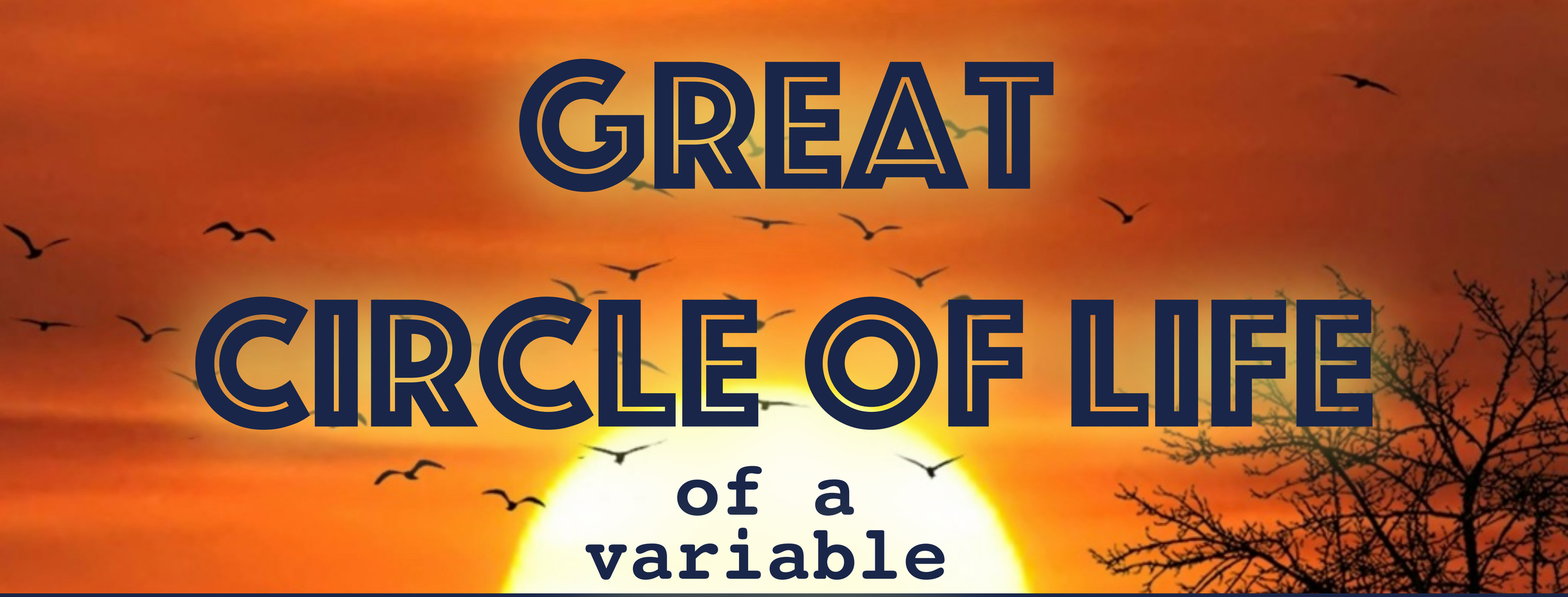
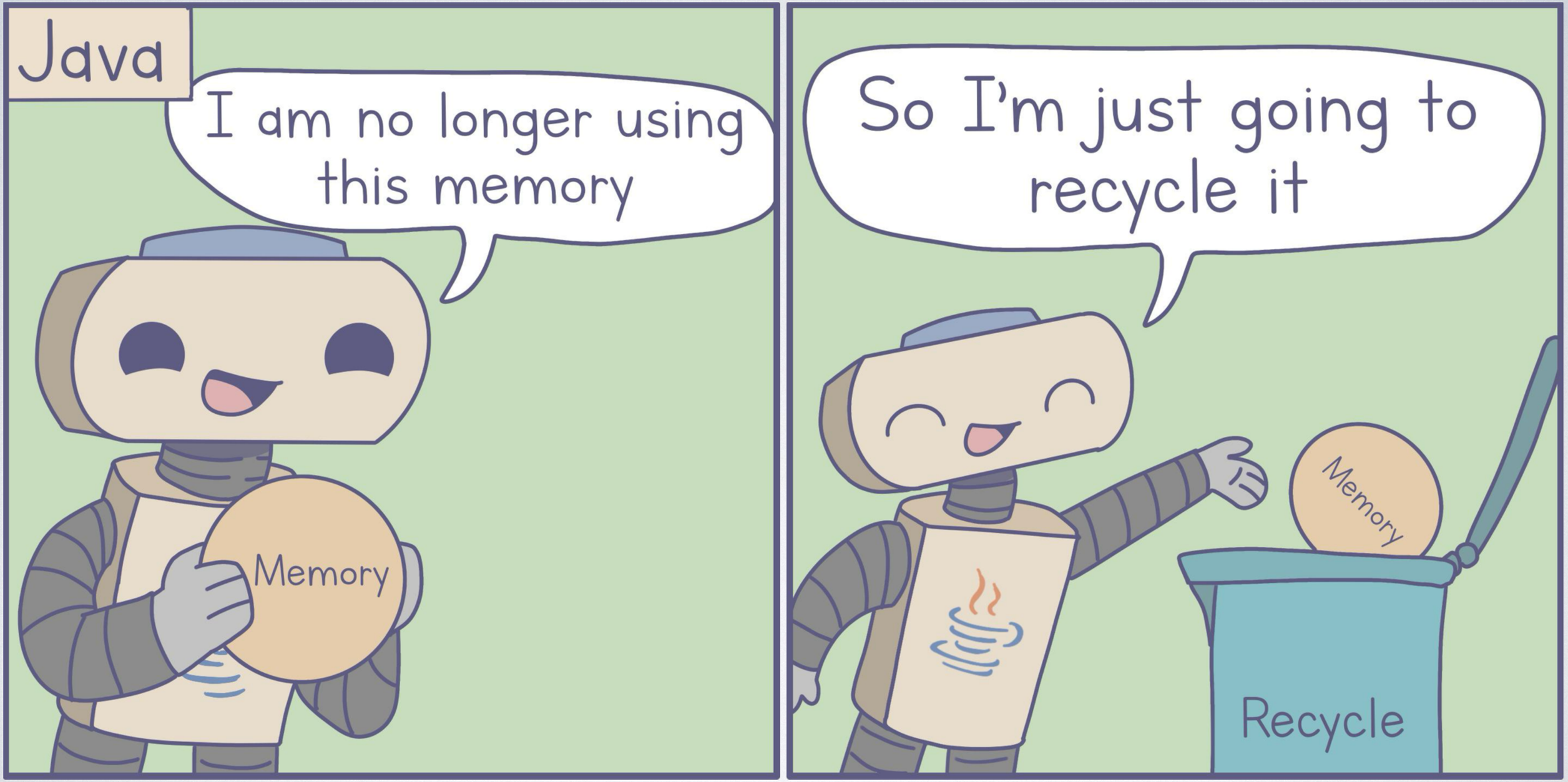


lendable

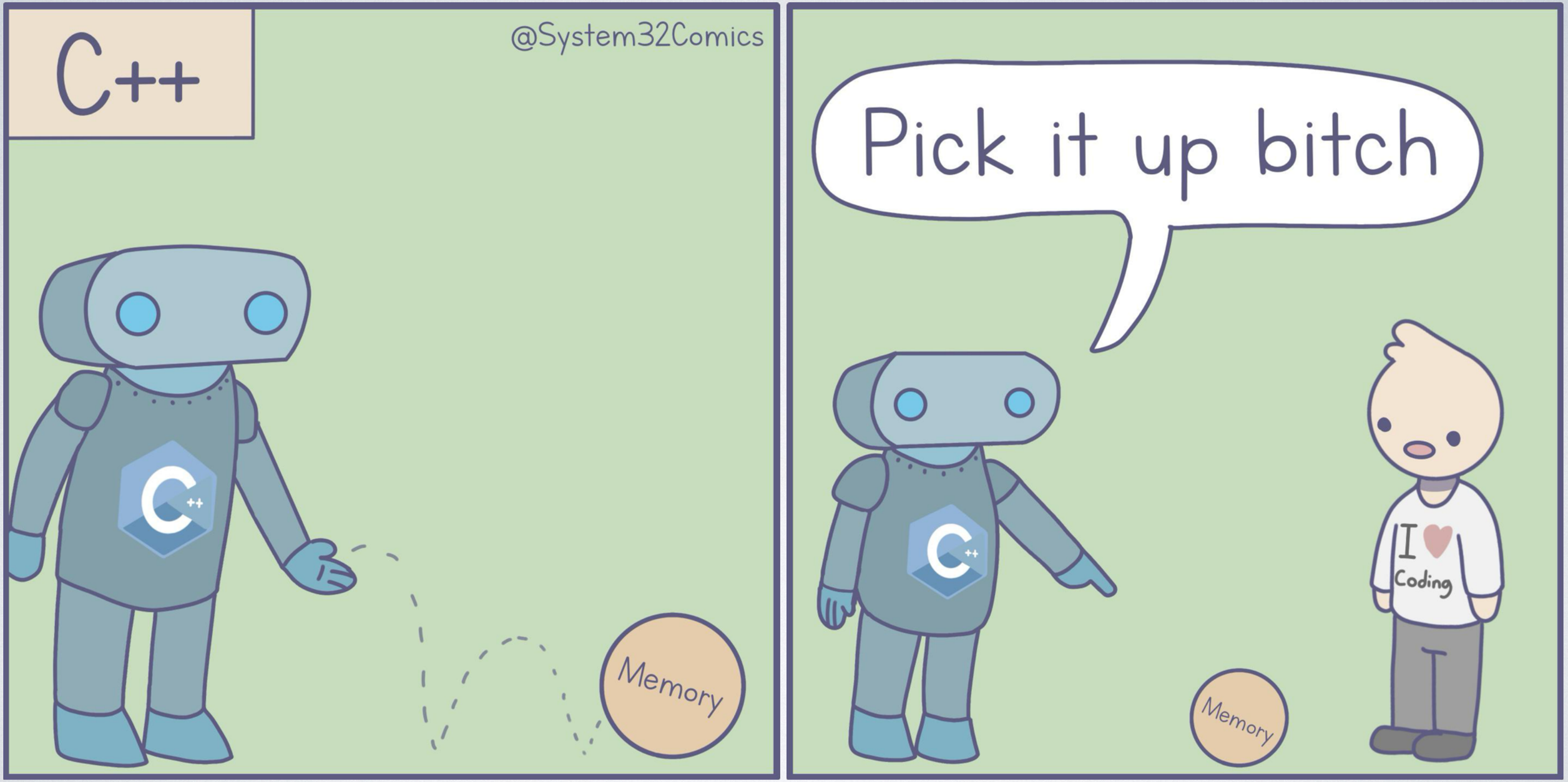
We're hiring!

Benoit VIGUIER
@b_viguier

FORUMPHP
PARIS2024



@System32Comics



@System32Comics

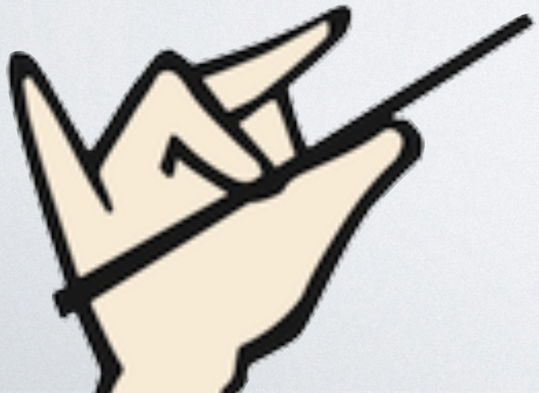
Why should we care?

« All non-trivial abstractions, to some degree, are leaky. »

Joel Spolsky - The Law of Leaky Abstractions (2002)

Composer

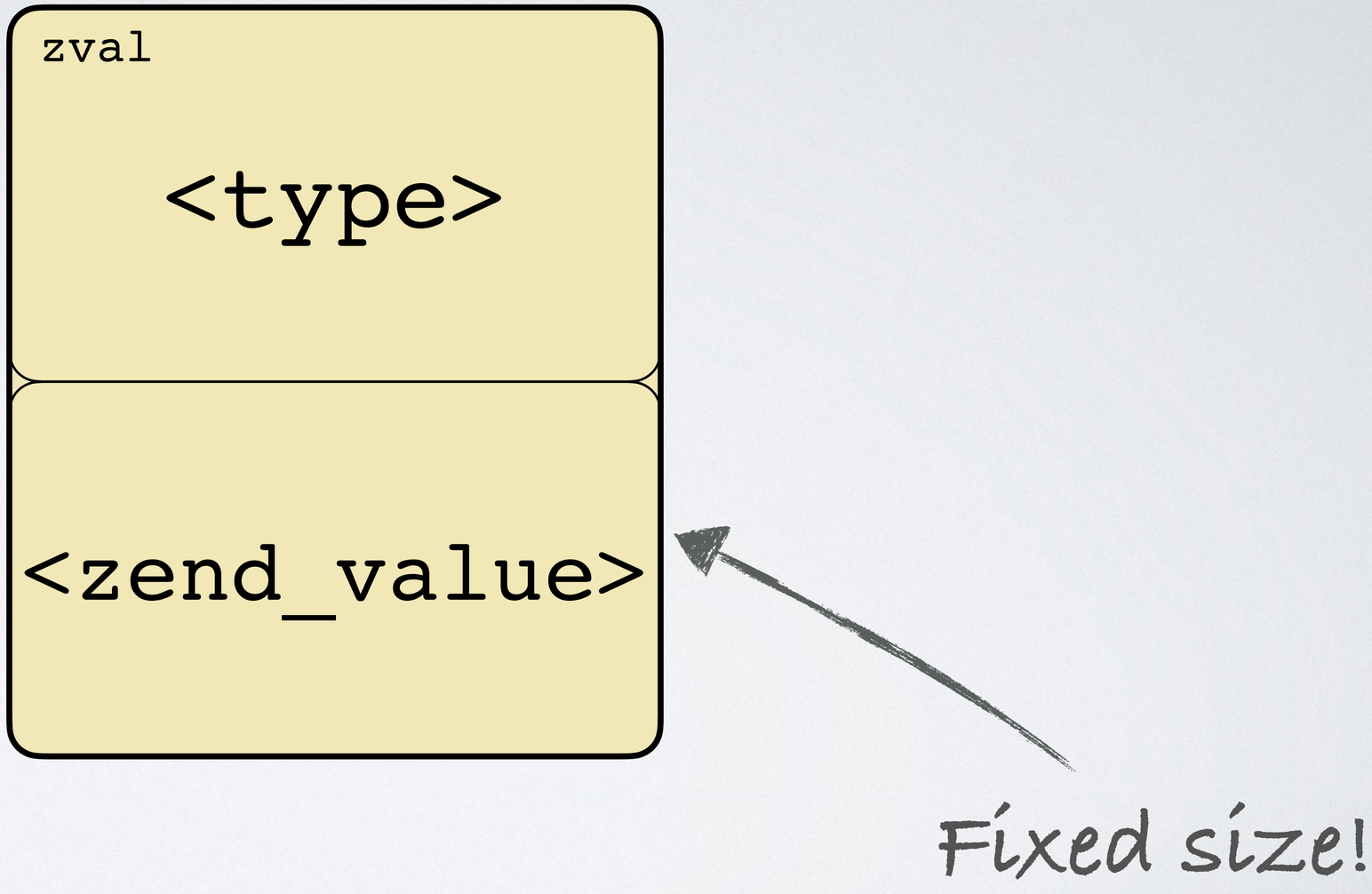
`gc_disable`



Short HTTP Queries

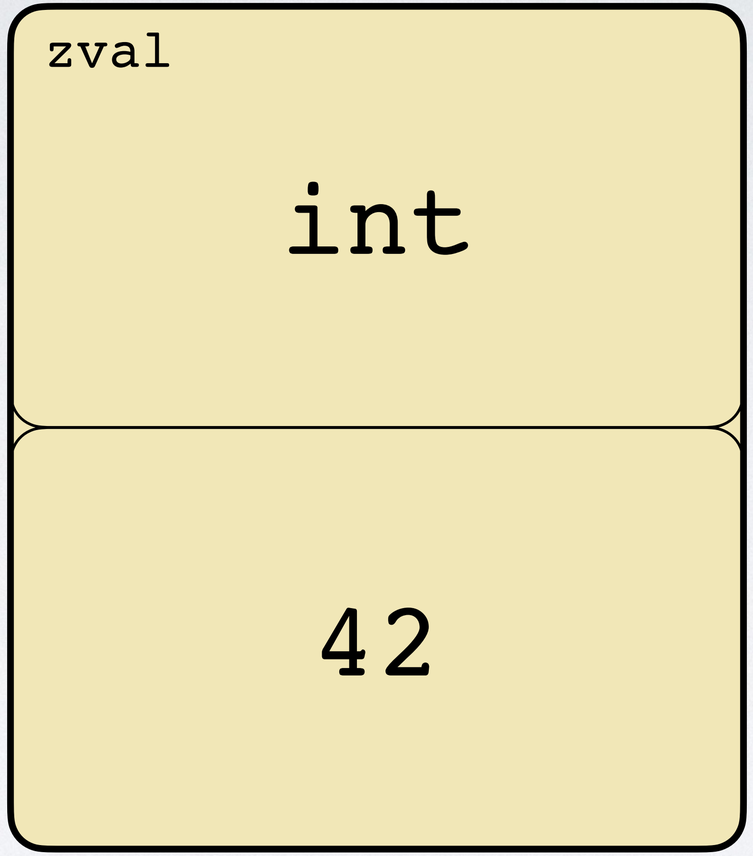
Long CLI Scripts

CREATION


zval
<type>
<zend_value>
Fixed size!

Numbers and Booleans

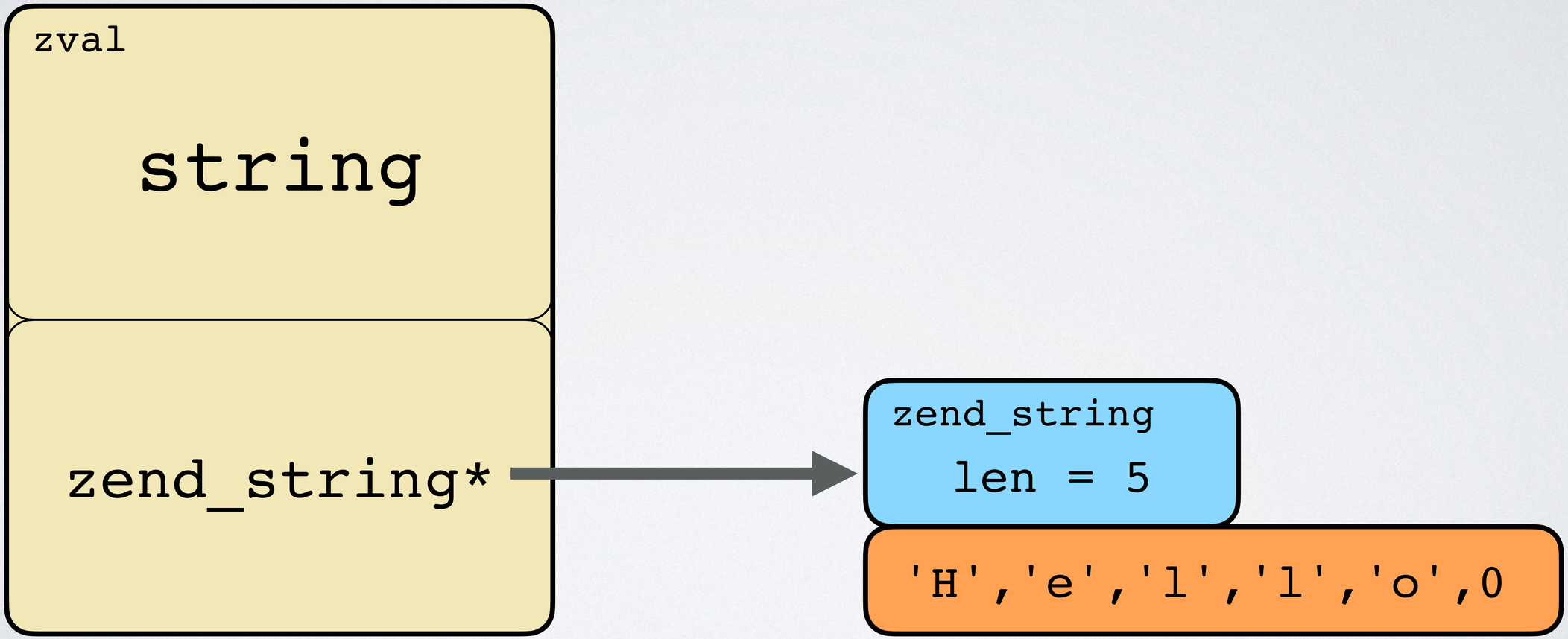
Fixed size!


zval
int
42

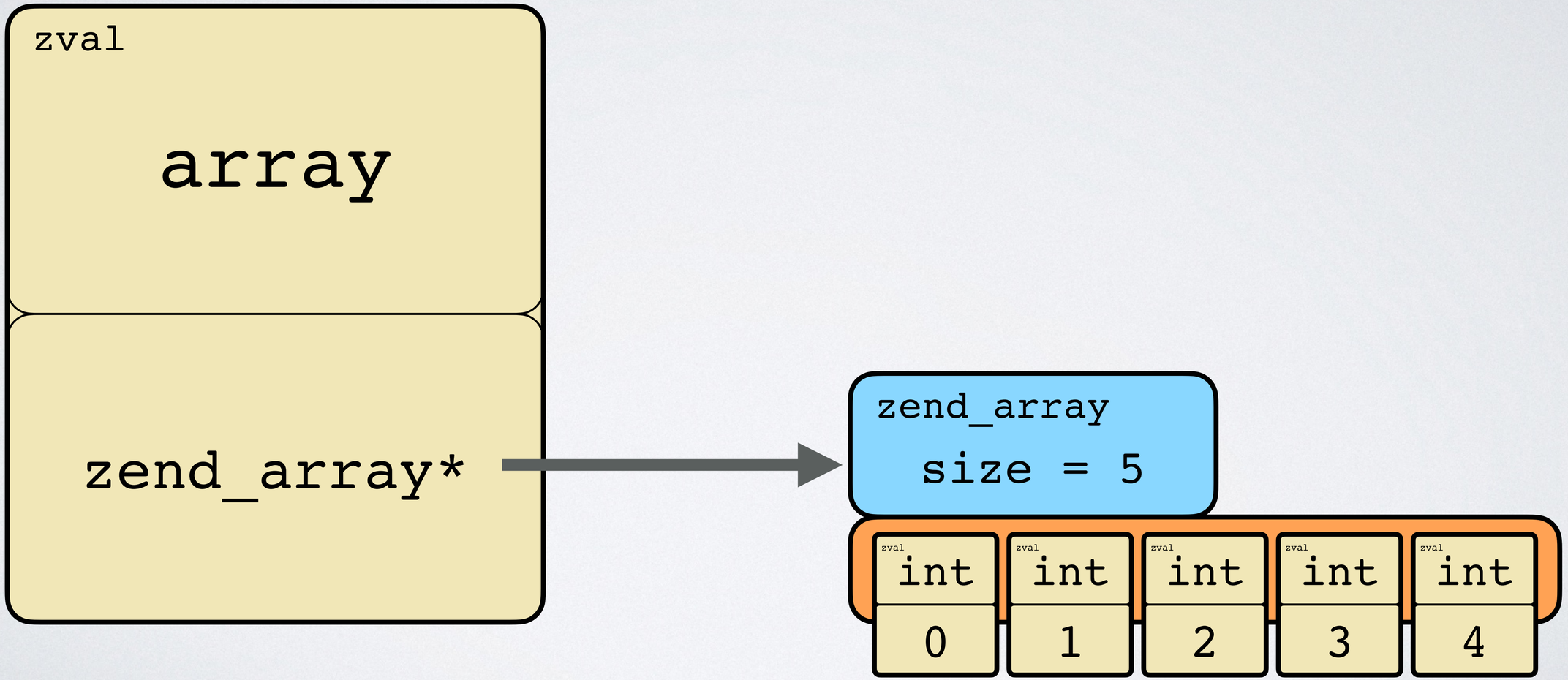
Strings and arrays

Variable size

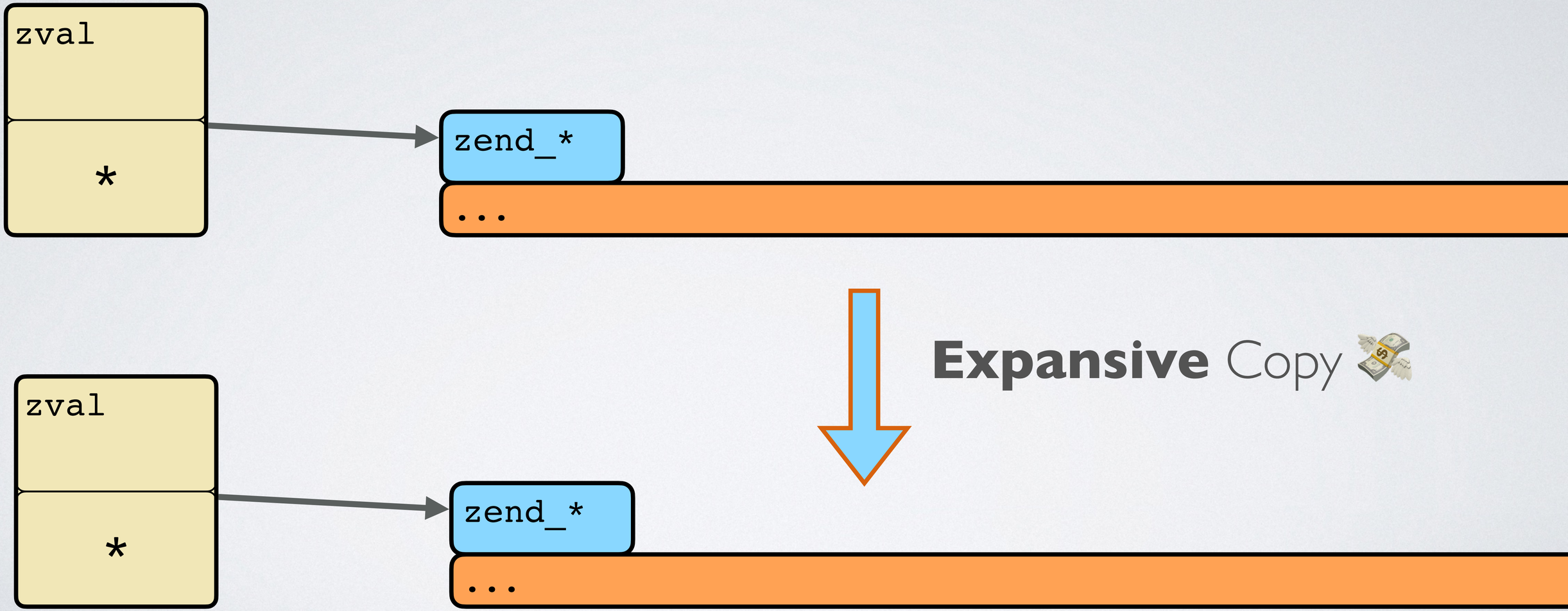
String



Array


zval
array
zend_array*
zend_array
size = 5
zval
int
0
zval
int
1
zval
int
2
zval
int
3
zval
int
4

Copy



Expansive Copy 💸

Implicit copy

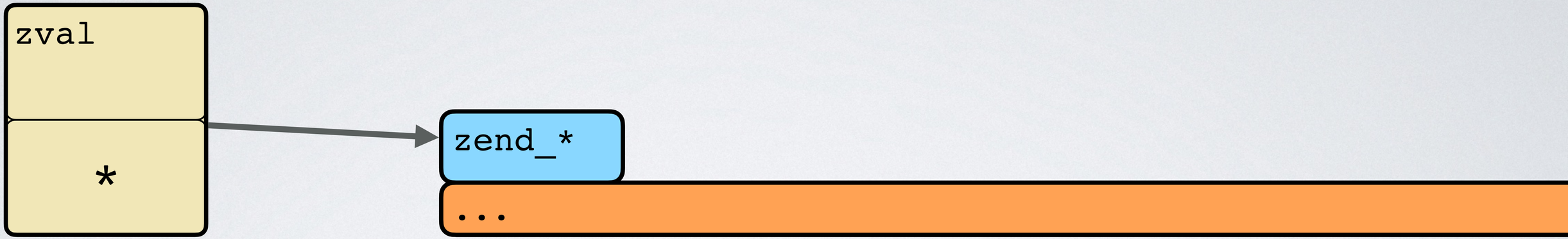
```php
function recursion(array $param, int $depth): void
{

    if ($depth > 0) {
        recursion($param, $depth - 1);
    }
}


$input = array_pad([], 1_000_000, 0);
recursion(param: $input, depth: 10);
```

Implicit copy

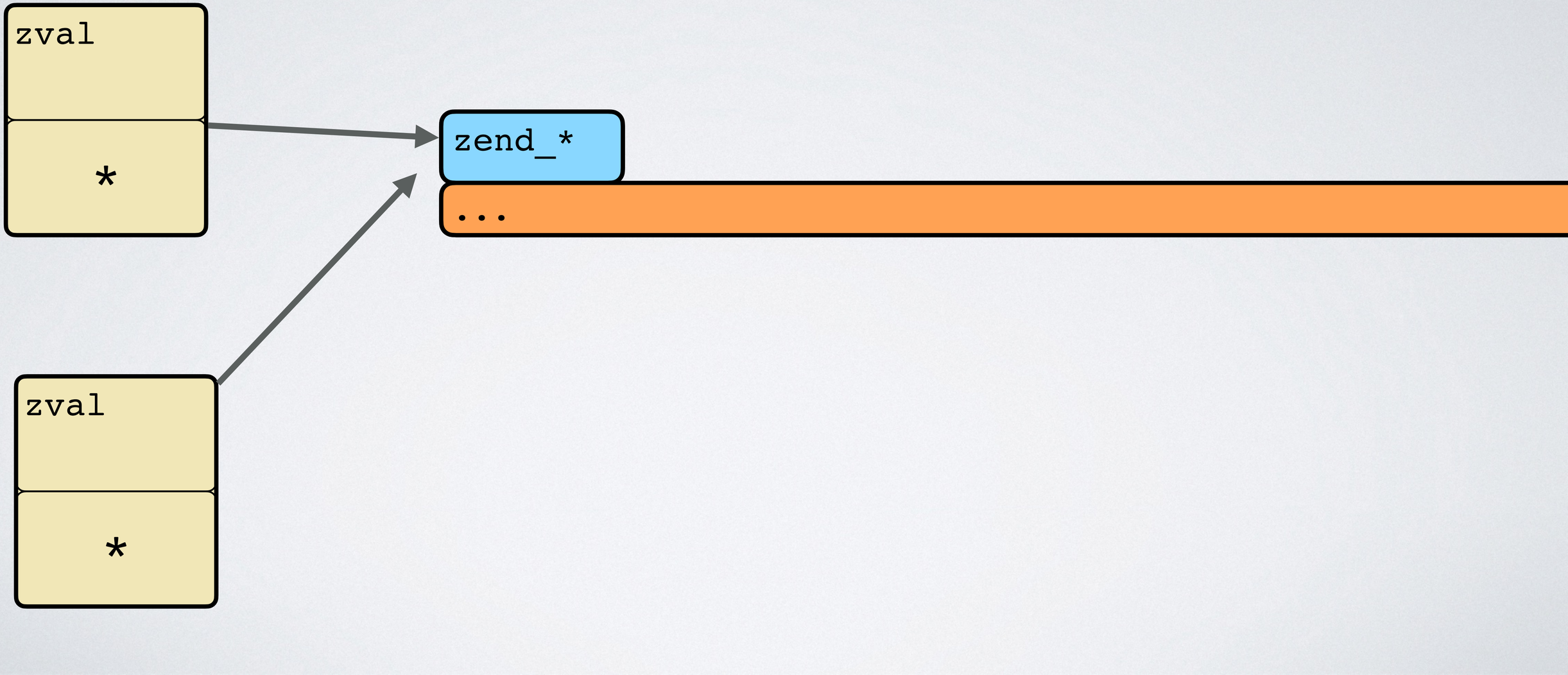
```php
function recursion(array $param, int $depth): void
{

    if ($depth > 0) {
        recursion($param, $depth - 1);
    }
}


$input = array_pad([], 1_000_000, 0);
recursion(param: $input, depth: 10);
```

Copy On Write

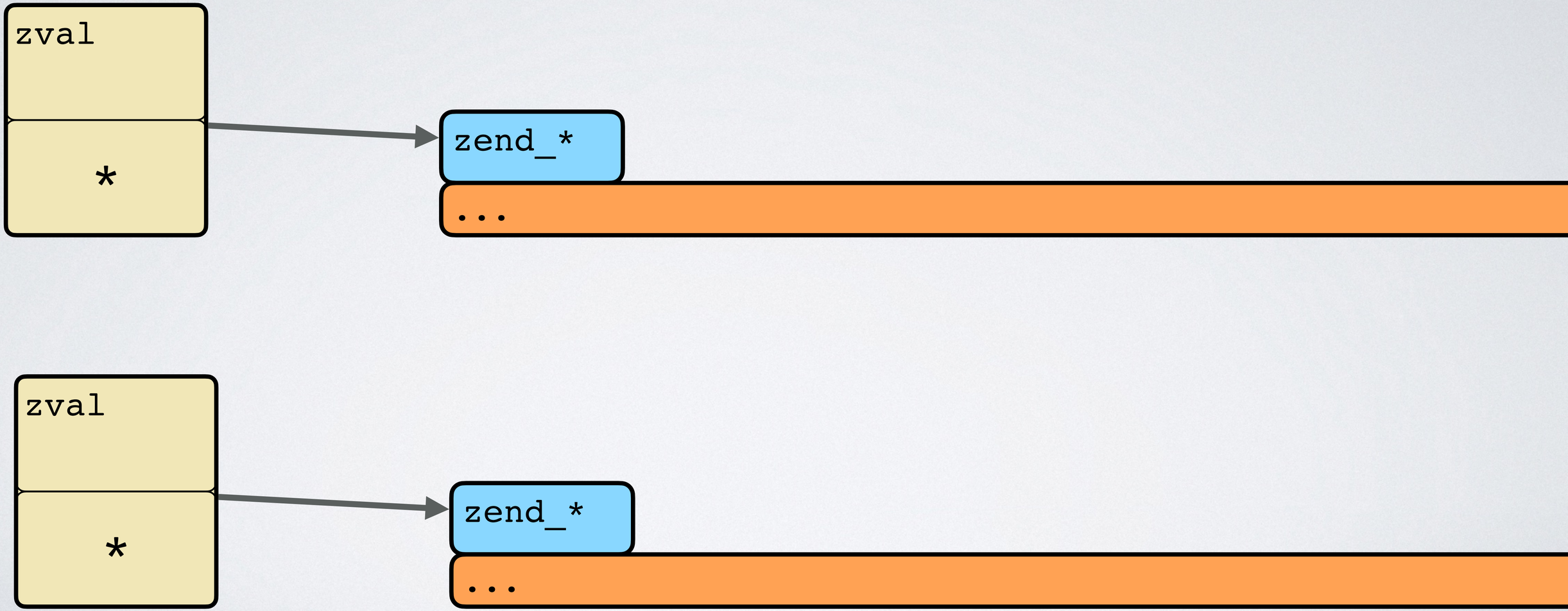
Creation



Copy



(Just before) Write



Side effect!

Memory bump…

Implicit copy

```php
function recursion(array $param, int $depth): void
{

    if ($depth > 0) {
        recursion($param, $depth - 1);
    }
}

$input = array_pad([], 1_000_000, 0);
recursion(param: $input, depth: 10);
```

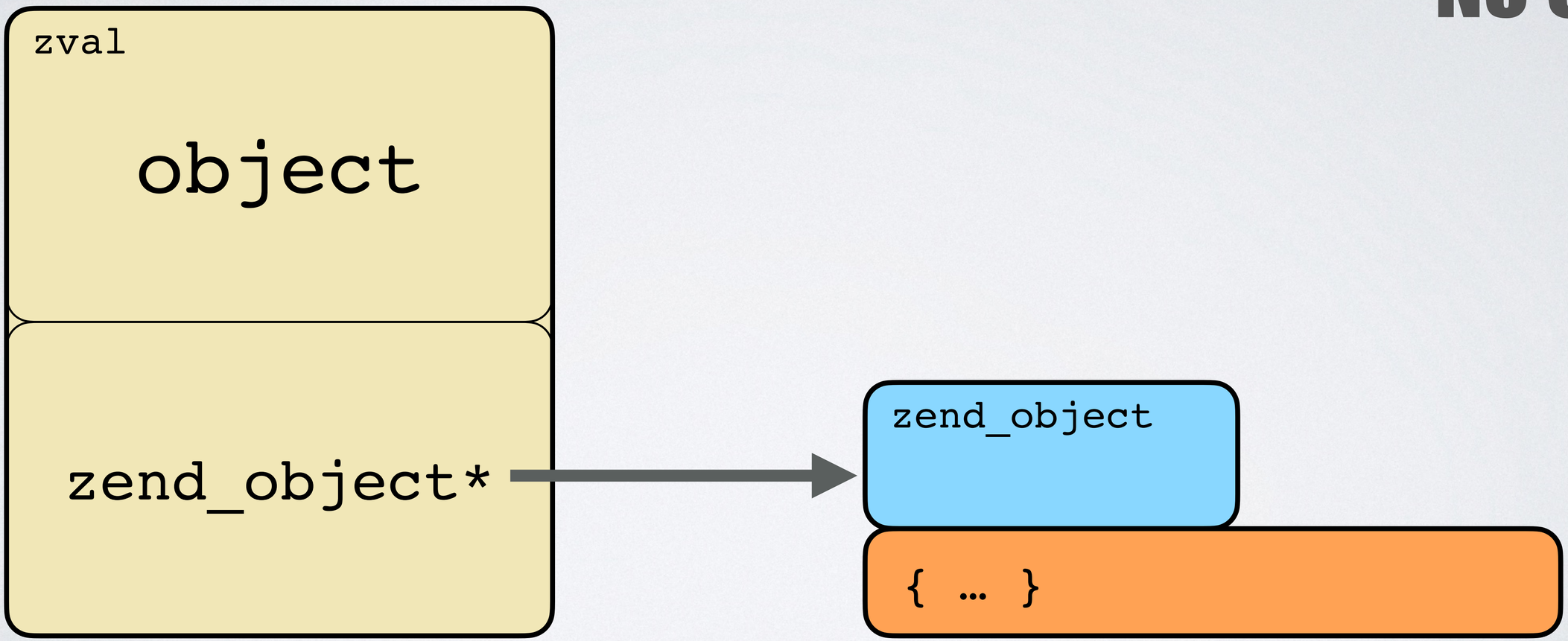
Implicit copy

```php
function recursion(array $param, int $depth): void
{
    $param[0] = $param[0]; // 💥
    if ($depth > 0) {
        recursion($param, $depth - 1);
    }
}

$input = array_pad([], 1_000_000, 0);
recursion(param: $input, depth: 10);
```

Objects

Explicit copy

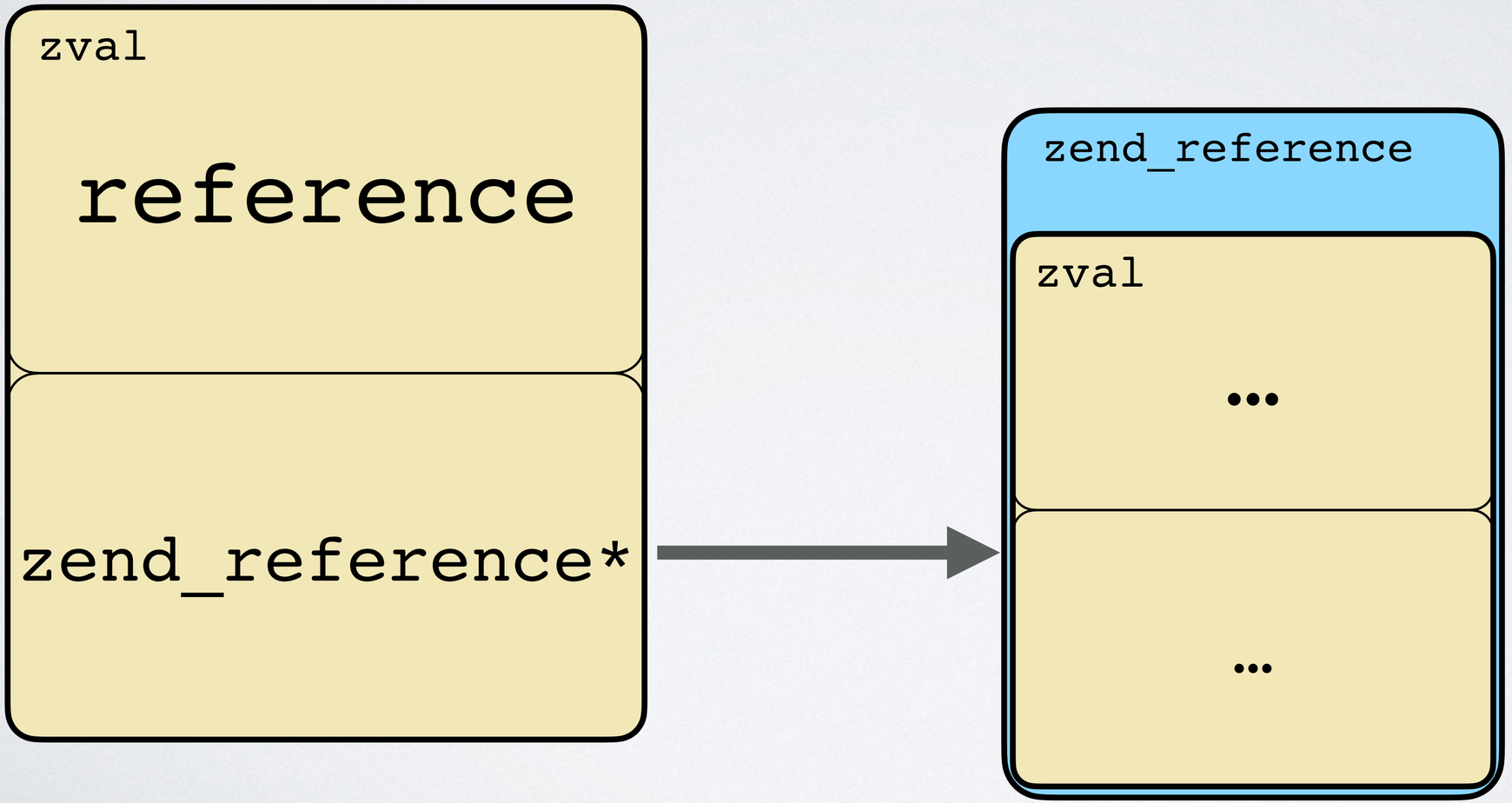
Object



References

Also a zval

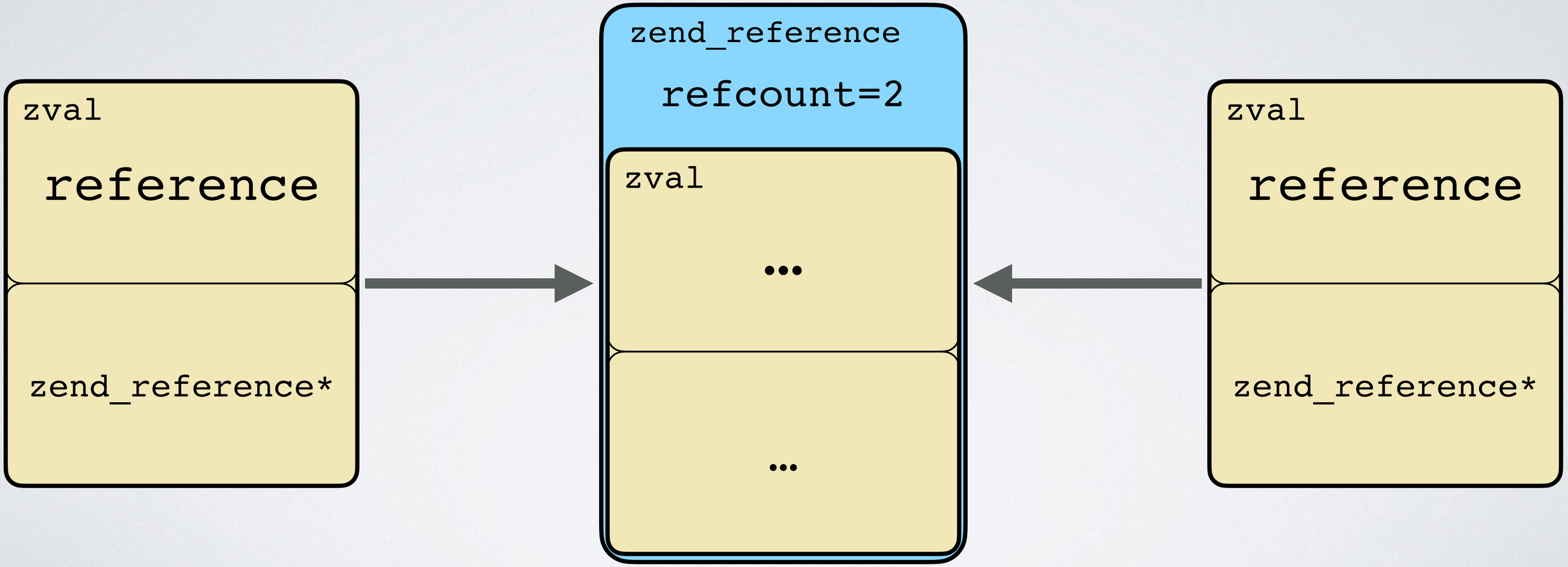
Reference



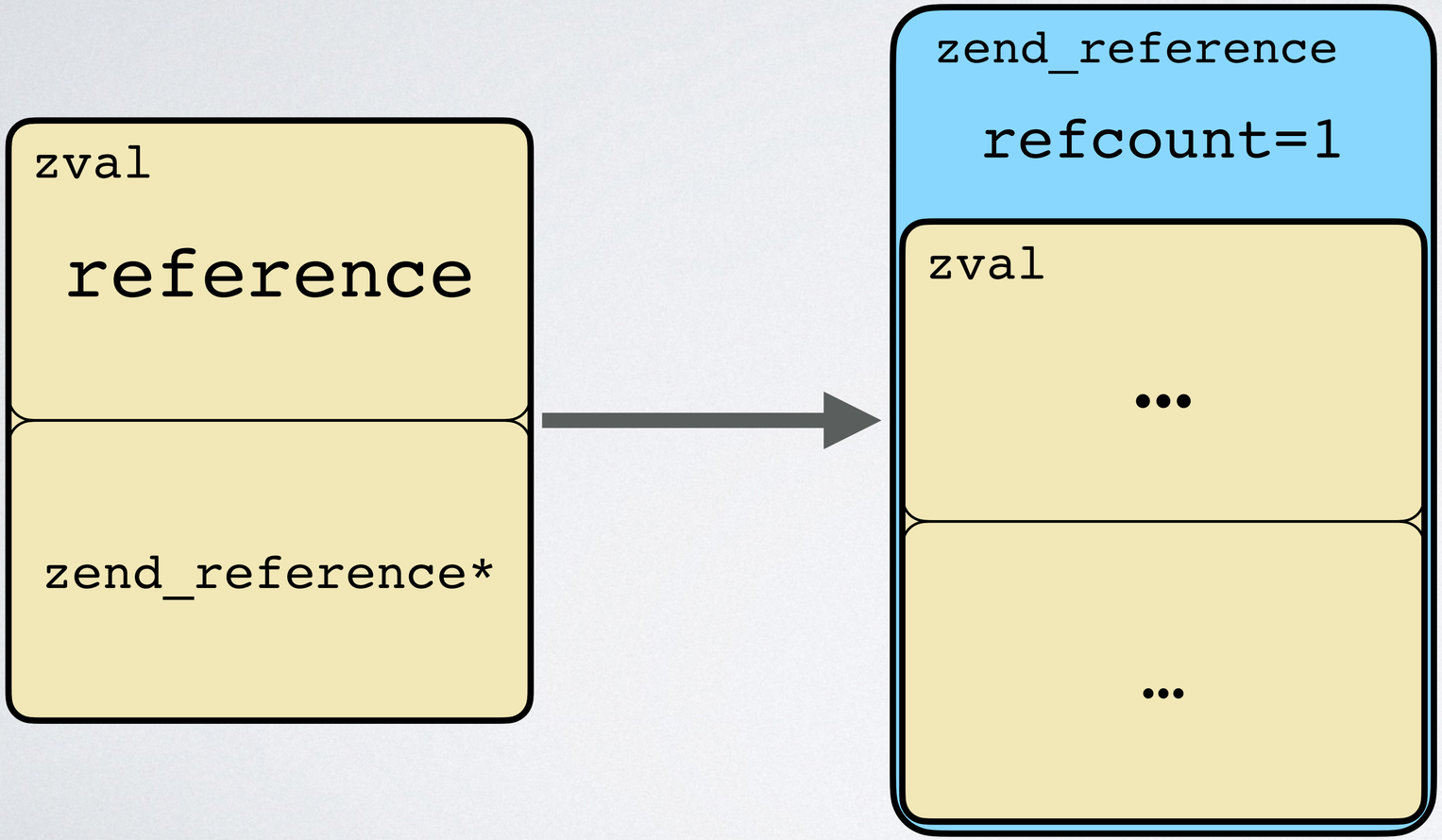
DESTRUCTION

How to be **SURE** that a *value* is not used anymore?

Reference Counting



Reference Counting



Reference Counting

Display refcount

```
$a = 'Hello';
$a .= ' World';
$b = &$a;
```

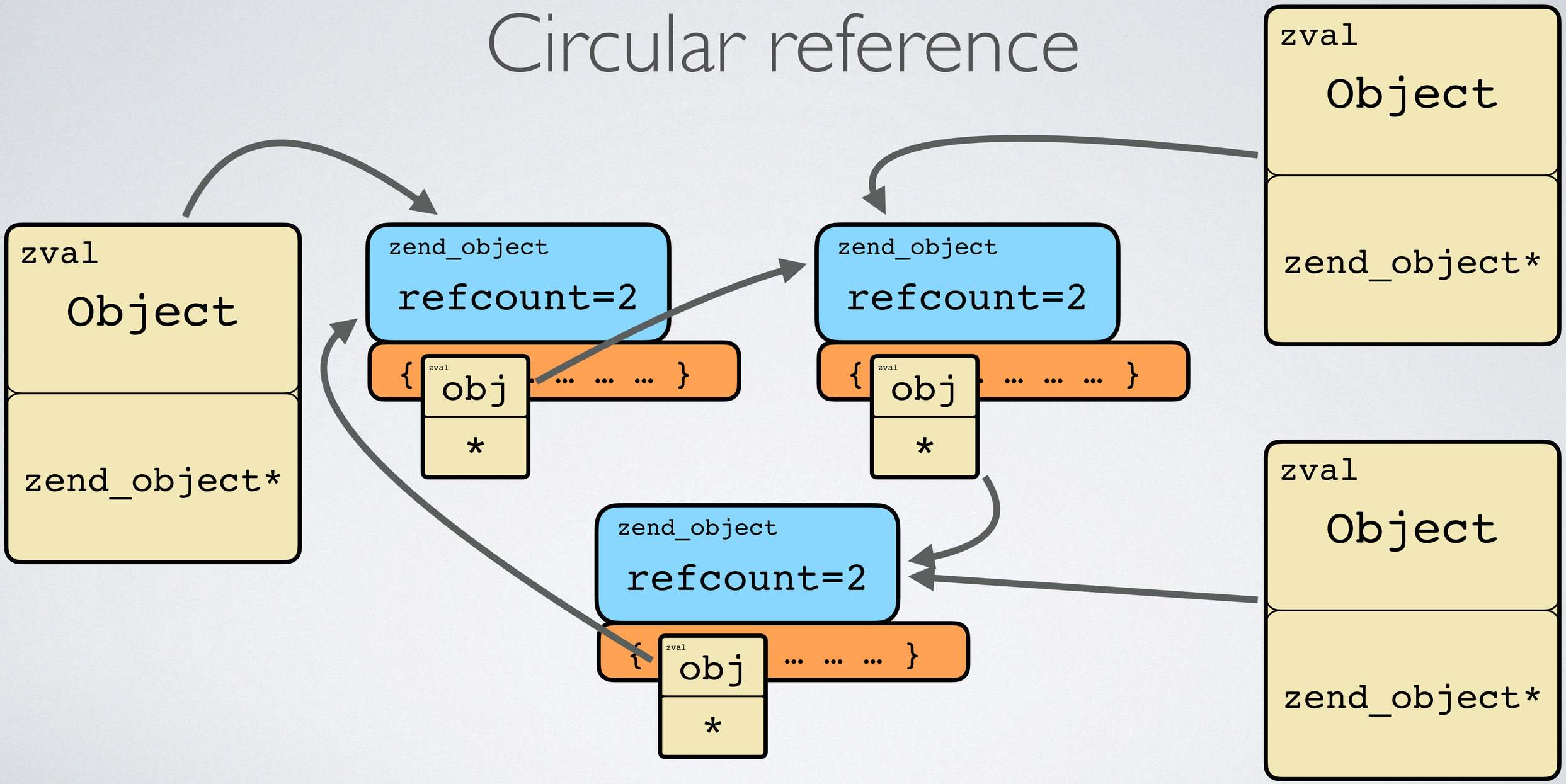
Display refcount

```
$a = 'Hello';
$a .= ' World';
$b = &$a;

debug_zval_dump($a, $b);
// string(11) "Hello World" refcount(3)
// string(11) "Hello World" refcount(3)
```

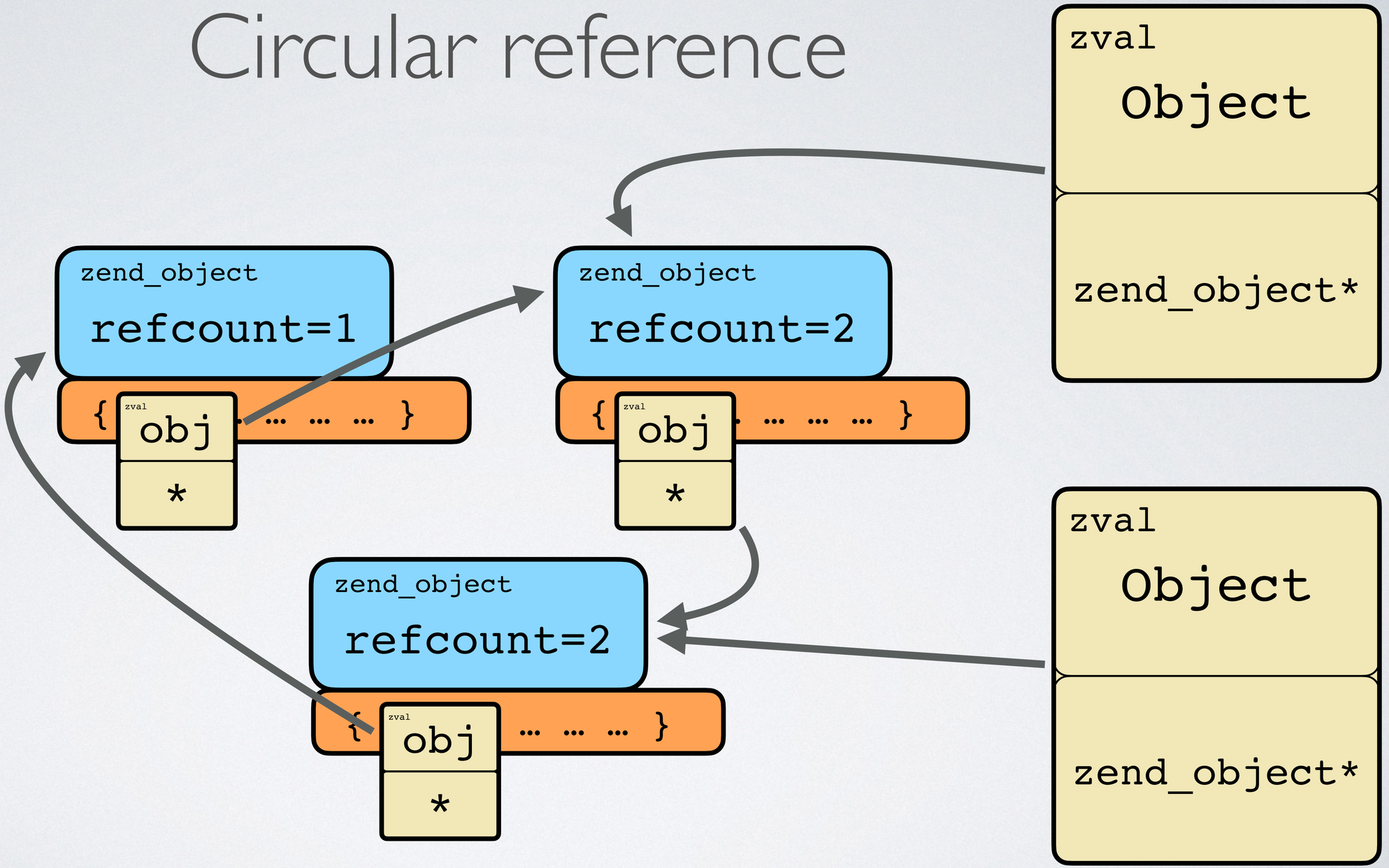
Display refcount

```
$a = 'Hello';
$a .= ' World';
$b = &$a;


debug_zval_dump($a, $b);
// string(11) "Hello World" refcount(3)
// string(11) "Hello World" refcount(3)
xdebug_debug_zval('a', 'b');
// a: (refcount=2, is_ref=1)='Hello World'
// b: (refcount=2, is_ref=1)='Hello World'
```

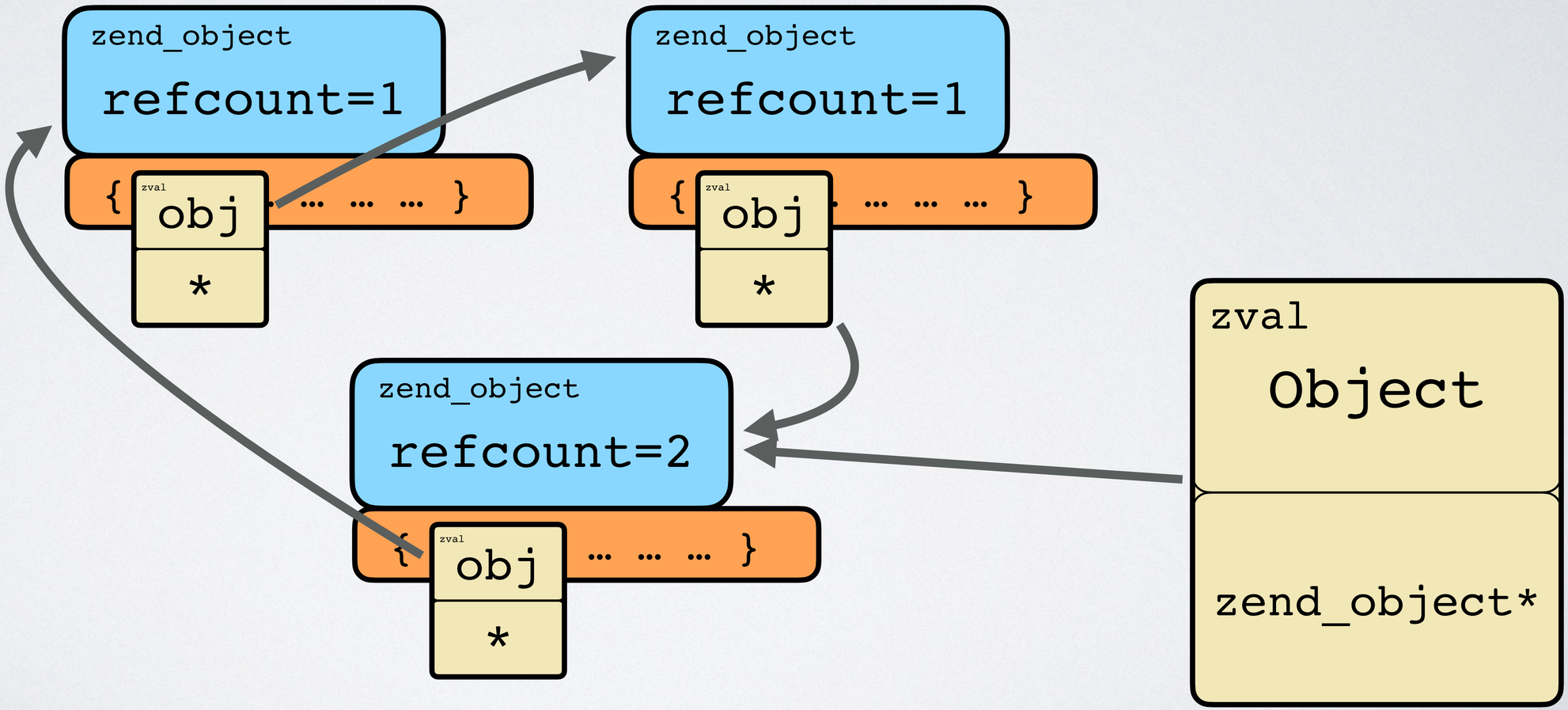
Circular reference



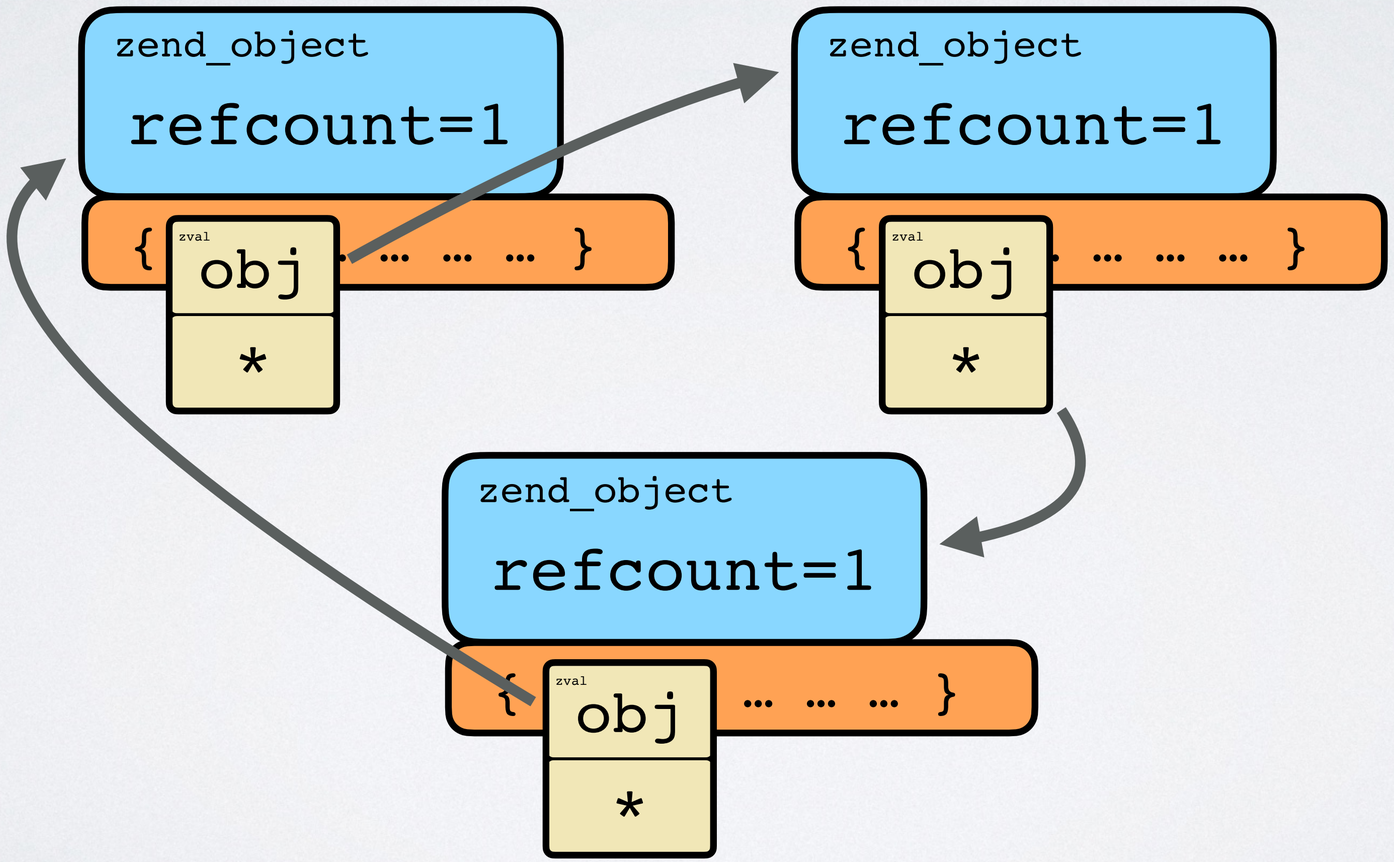
Circular reference



Circular reference



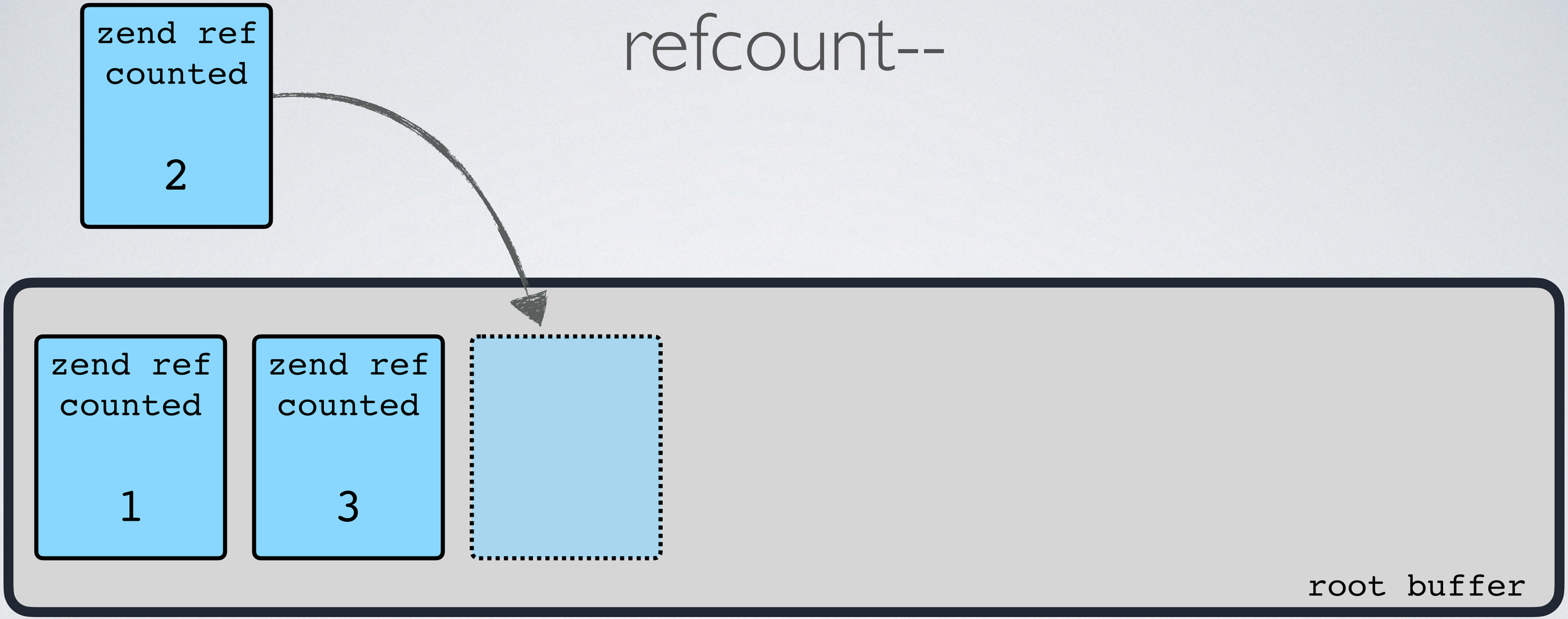
Circular reference



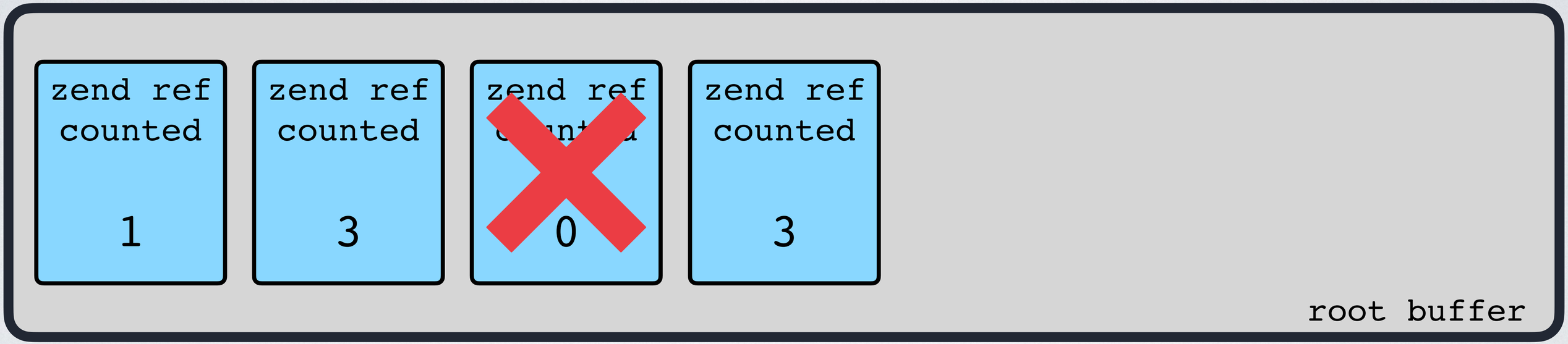
Cycles collection

Php >= 5.3

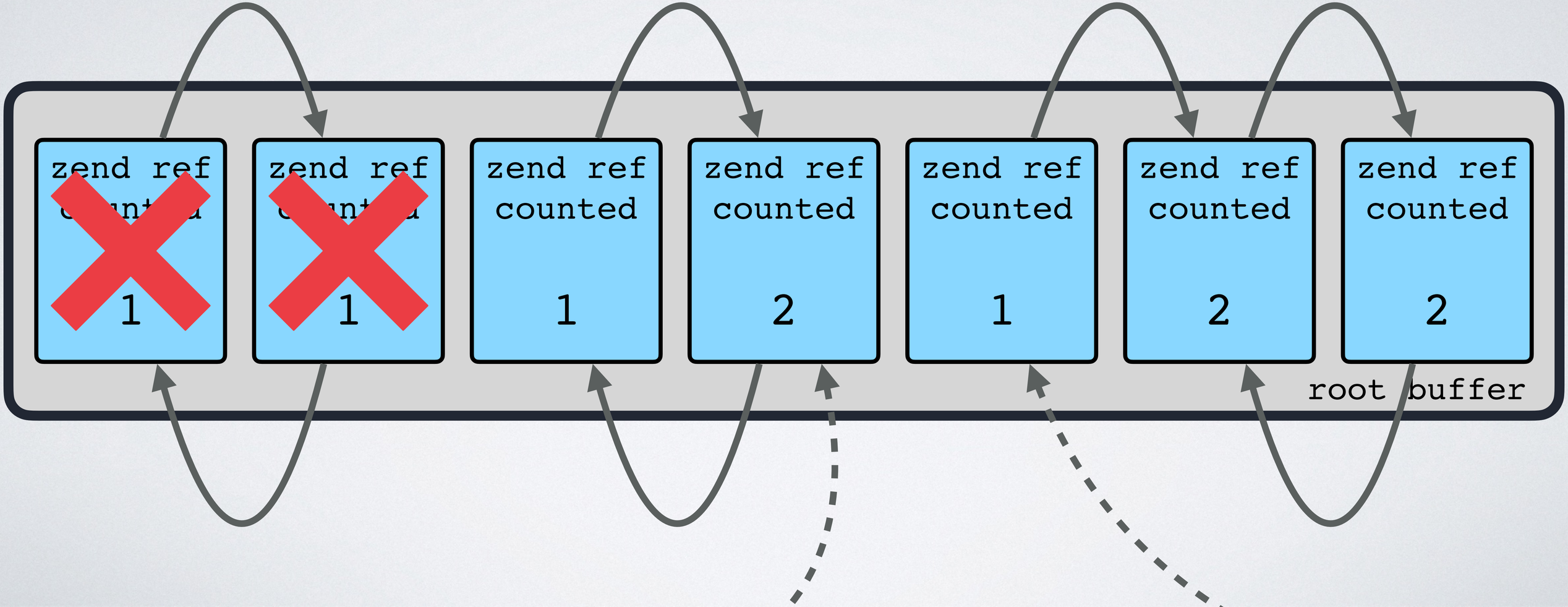
refcount--


zend ref
counted
2
zend ref
counted
1
zend ref
counted
3
root buffer

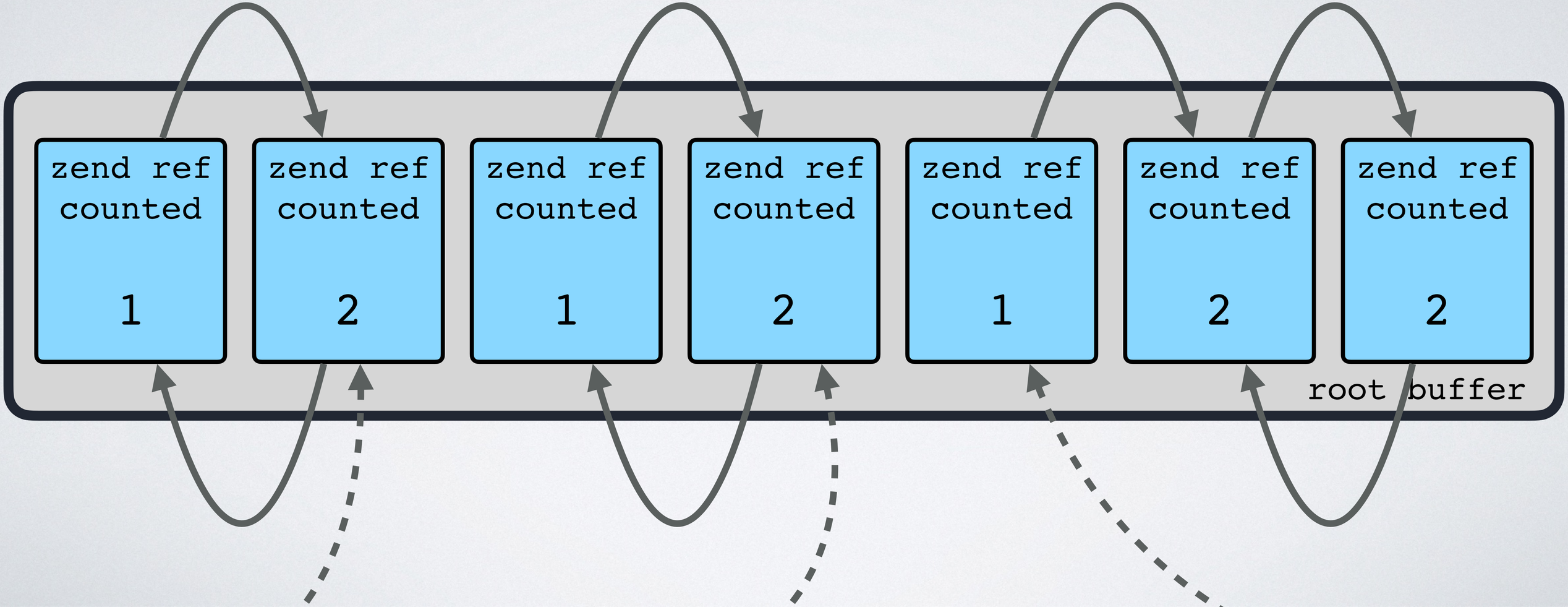
refcount === 0


zend ref counted
1
zend ref counted
3
zend ref counted
0
zend ref counted
3
root buffer

Size === 10000

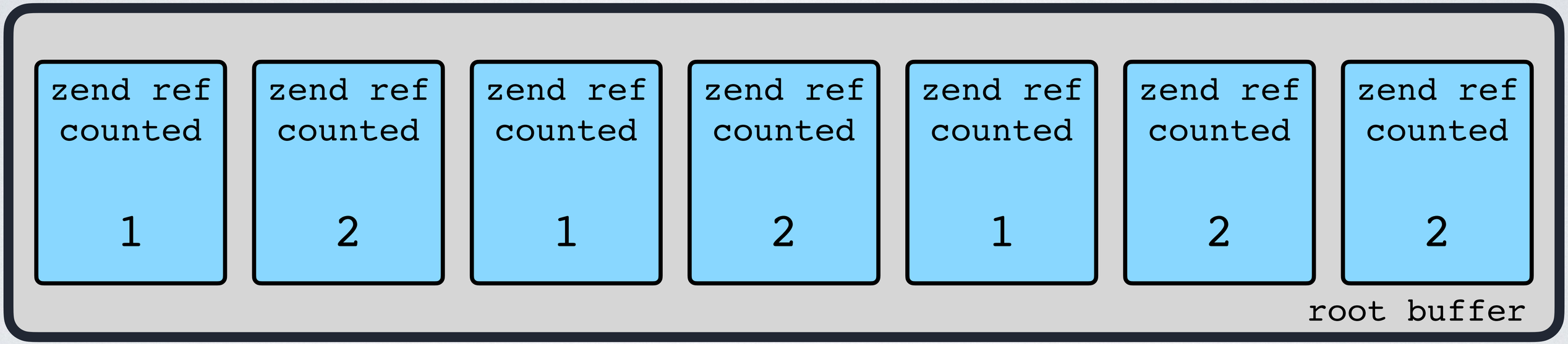

zend ref counted
1
zend ref counted
1
zend ref counted
1
zend ref counted
2
zend ref counted
1
zend ref counted
2
zend ref counted
2
root buffer

Worst case


zend ref counted
1
zend ref counted
2
zend ref counted
1
zend ref counted
2
zend ref counted
1
zend ref counted
2
zend ref counted
2
root buffer

Worst case


zend ref
counted
2
!?


zend ref
counted
1
zend ref
counted
2
zend ref
counted
1
zend ref
counted
2
zend ref
counted
1
zend ref
counted
2
zend ref
counted
2
root buffer

https://afup.org/talks/1765-php-meminfo-ou-la-chasse-aux-memory-leaks

HUNTING DOWN MEMORY LEAKS WITH PHP MEMINFO

Benoit Jacquemont / @bjacquemont

PHPTOUR2016
Clermont-Ferrand

Related functions

```
gc_status(): array

gc_enable(): void
gc_disable(): void

gc_collect_cycles(): int
```

Destructors

Don’t rely on a specific order of destruction

It's complicated…

```
class A
{
    public function __destruct()
    {
        throw new Exception('Error');
    }
}
```

It's complicated…

```
class A
{
    public static array $destroyed = [];

    public function __destruct()
    {
        self::$destroyed[] = $this;
    }
}
```

BREAKING CYCLES

WeakReference

```
final class WeakReference
{
  public function __construct();

  public static function create(
    object $object
  ): WeakReference;

  public function get(): ?object;
}
```

constructor

```
$weakRef = new WeakReference();
```

Do **not** use constructor

```
$weakRef = new WeakReference();
// PHP Fatal error:
// Uncaught Error: Direct
// instantiation of WeakReference
// is not allowed,
// use WeakReference::create
// instead
```

Example

```
$obj = new stdClass;
$weakref = WeakReference::create($obj);
```

Example

```
$obj = new stdClass;
$weakref = WeakReference::create($obj);

var_dump($weakref->get());
// object(stdClass)#1 (0) {}
```

Example

```
$obj = new stdClass;
$weakref = WeakReference::create($obj);

var_dump($weakref->get());
// object(stdClass)#1 (0) {}

unset($obj);
var_dump($weakref->get());
// NULL
```

Use case: graph

```
class Node {
    public ?Node $parent = null;
    public ?Node $child = null;
}
```

Use case: graph

```
$parent = new Node();
$child = new Node();
```

Use case: graph

```
$parent = new Node();
$child = new Node();

$parent->child = $child;
$child->parent = $parent;
```

Use case: graph

```
$parent = new Node();
$child = new Node();

$parent->child = $child;
$child->parent = $parent;

unset($parent, $child);
var_dump(gc_collect_cycles());//int(2)
```

Use case: graph

```
class Node {
    /**
     * @var ?WeakReference<Node>
     */
    public ?WeakReference $parent = null;
    public ?Node $child = null;
}
```

Use case: graph

```
$parent = new Node();
$child = new Node();
```

Use case: graph

```
$parent = new Node();
$child = new Node();

$parent->child = $child;
$child->parent =
      WeakReference::create($parent);
```

Use case: graph

```
$parent = new Node();
$child = new Node();

$parent->child = $child;
$child->parent =
      WeakReference::create($parent);

unset($parent, $child);
var_dump(gc_collect_cycles());//int(0)
```

One instance per object

```php
$obj = new stdClass();
$weakref1 = WeakReference::create($obj);
$weakref2 = WeakReference::create($obj);

var_dump($weakref1 === $weakref2);
// bool(true)
```

WeakMap

```
final class WeakMap
 implements ArrayAccess, Countable, IteratorAggregate
{
    public function count(): int;
    public function getIterator(): Iterator;
    public function offsetExists(object $object): bool;
    public function offsetGet(object $object): mixed;
    public function offsetSet(object $object, mixed $value): void;
    public function offsetUnset(object $object): void;
}
```

Example

```
$obj1 = new stdClass();
$obj2 = new stdClass();
$obj3 = new stdClass();

$weakMap = new WeakMap();
$weakMap[$obj1] = 'obj1';
$weakMap[$obj2] = 'obj2';
$weakMap[$obj3] = 'obj3';
```

Example

```
foreach ($weakMap as $value) {
    echo $value . PHP_EOL;
}
// obj1 obj2 obj3
```

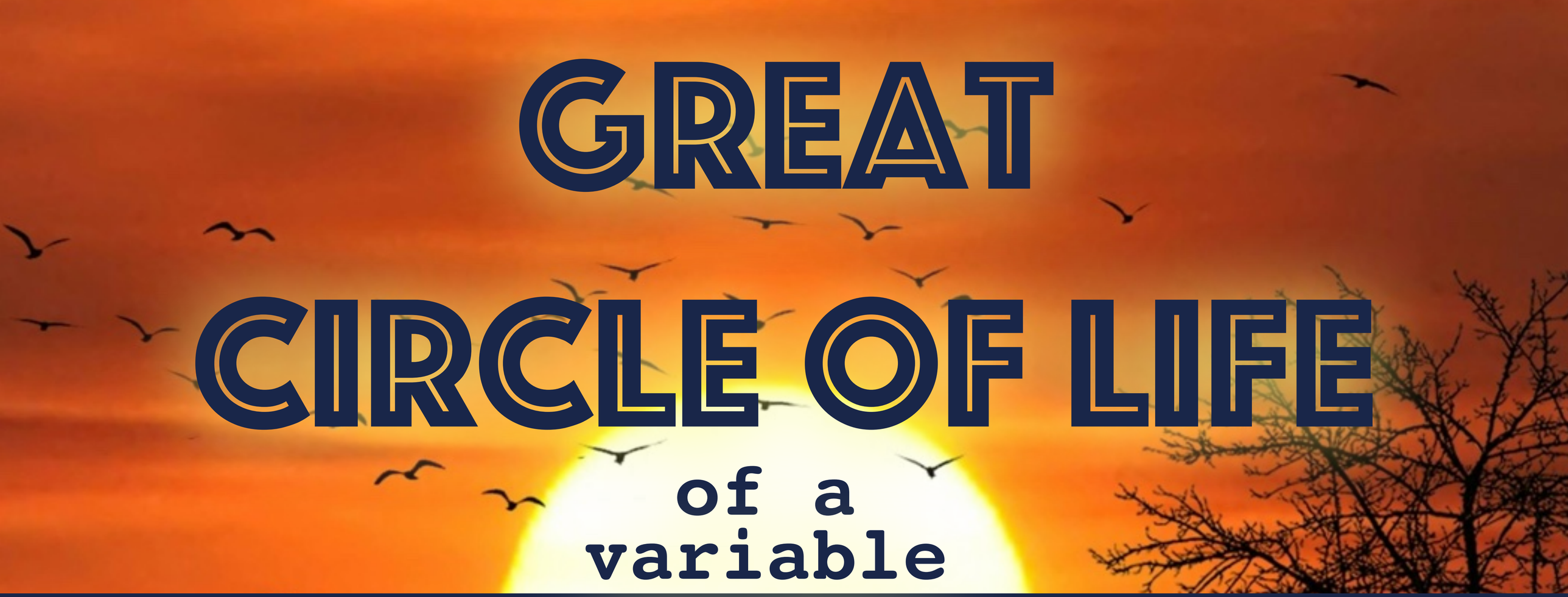
Example

```
foreach ($weakMap as $value) {
    echo $value . PHP_EOL;
}
// obj1 obj2 obj3

unset($obj2);
foreach ($weakMap as $value) {
    echo $value . PHP_EOL;
}
// obj1 obj3
```

CREATION

DESTRUCTION

BREAKING CYCLES



lendable

We're hiring!

Benoit VIGUIER
@b_viguier

FORUMPHP
PARIS2024