# down the **rb_newobj()** rabbit hole

**JUNE 28, 2013 • ATHENS, GREECE**

Good afternoon.

My name is **Chris Kelly**.

On the Internets, **amateurhuman**.

I work at **New Relic**.

1

2

3

4

1

2

3

4

What are we talking about

1 → 2 3 4

What are we talking about

Navigating CRuby

1 → 2 → 3 4

What are we talking about

Navigating CRuby

Object Creation

1 → 2 → 3 → 4

What are we talking about

Navigating CRuby

Object Creation

Garbage Collection

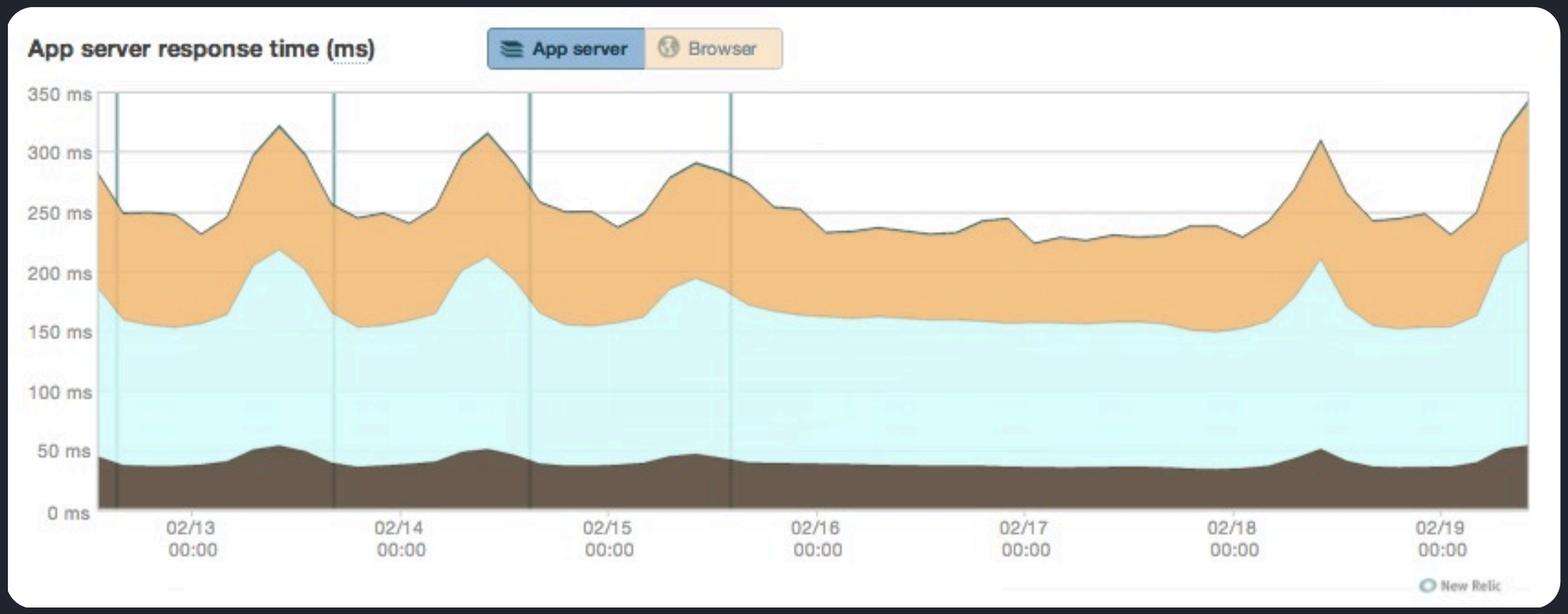
# 1

What are we talking about.

This is New Relic on **Ruby 1.8**

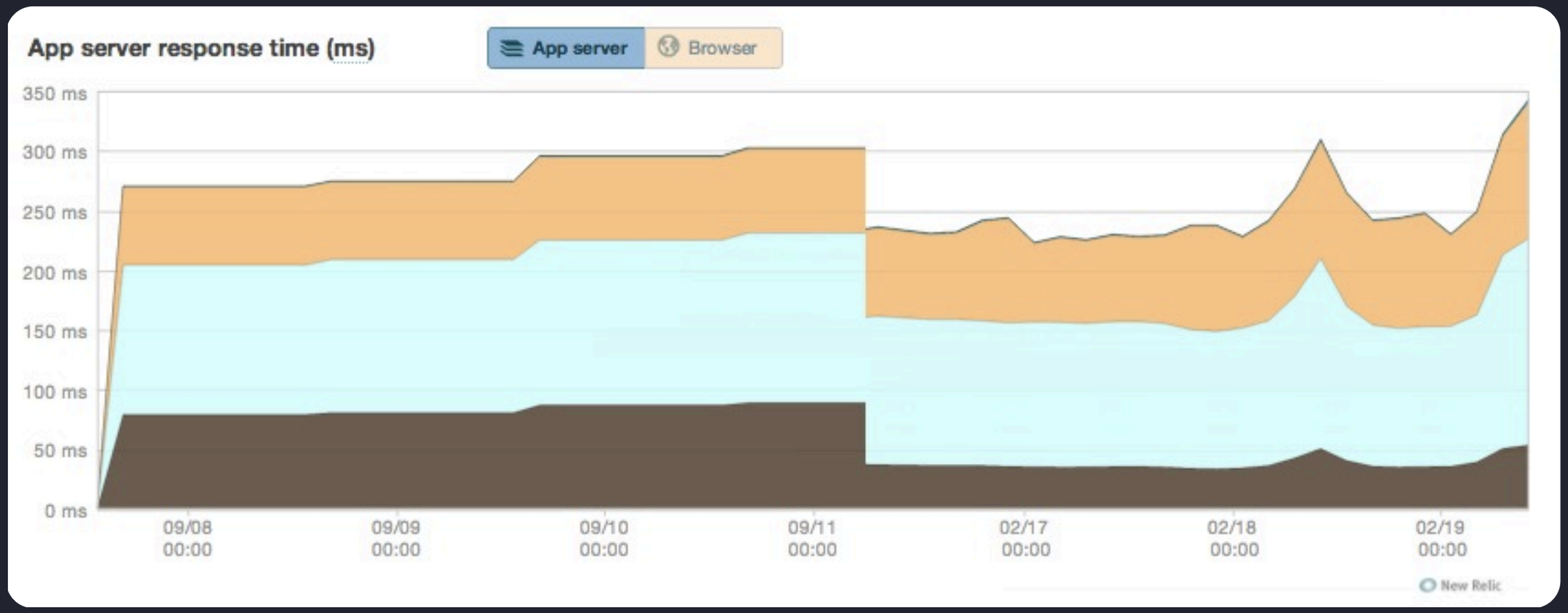

Average **80ms** in Garbage Collection

This is New Relic on **Ruby 1.9**




Average **42ms** in Garbage Collection

Ruby 1.8 Ruby 1.9



Ruby 1.8 Ruby 1.9


SAVE SAVE SAVE!

48%

UPGRADE NOW!

Ruby 1.8 Ruby 1.9

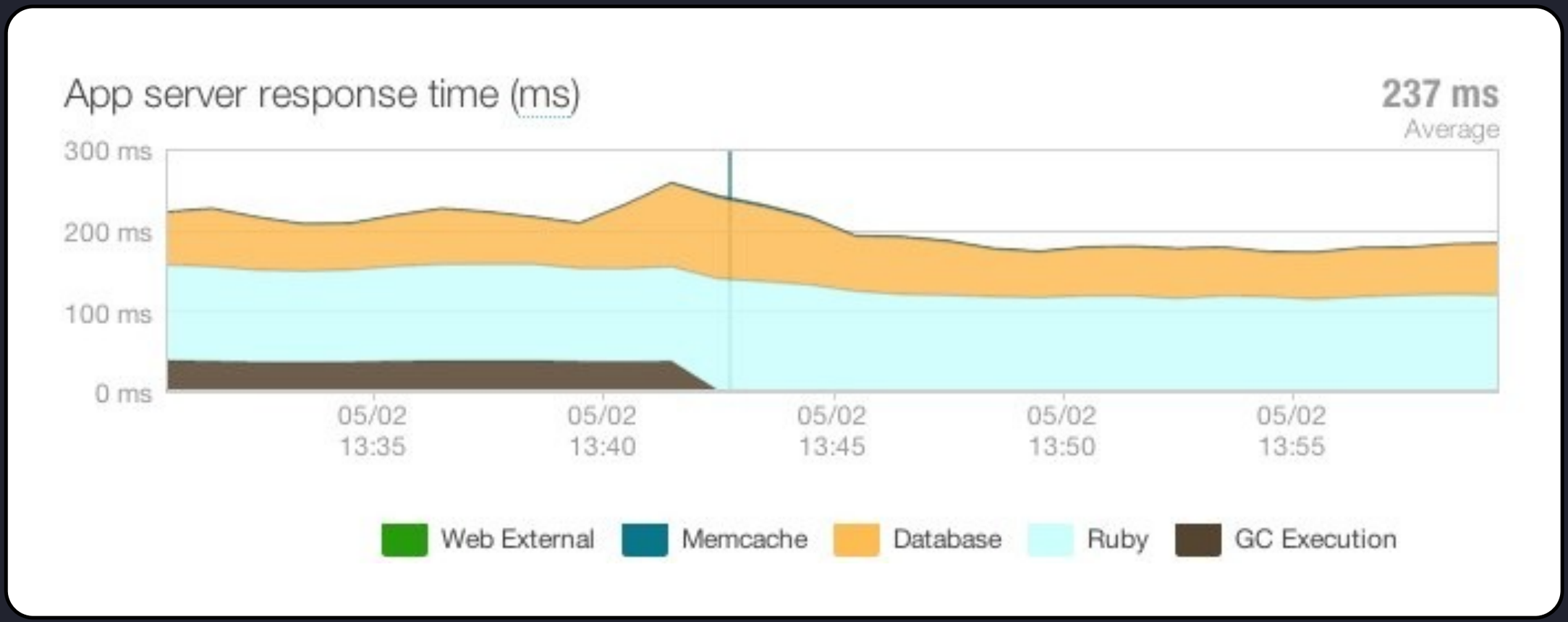

SAVE SAVE SAVE!

48%

UPGRADE NOW!*

* Some restrictions may apply

# Kick Garbage Collection **Out of the Band**




With **Unicorn OOB GC** + **Unicorn Slayer**

# Out of Band GC

```ruby
require_dependency 'unicorn/oob_gc'
require_dependency 'unicorn/unicorn_slayer.rb'

GC_FREQUENCY = 40

# Don't run GC during requests
GC.disable

# Run UnicornSlayer during every request
use(UnicornSlayer::Oom, ((1_024 + Random.rand(512)) * 1_024), 1)

# Run OOB GC every GC_FREQUENCY requests
use Unicorn::OobGC, GC_FREQUENCY


/* config.ru */
```

# Ruby

is all about objects.

**Ruby**

is all about objects.

Garbage collection is too.

# What is GC?

Garbage collector's function is to find data object that are no longer in use and make their space available for reuse by the running program.

An object is considered garbage if it is not reachable by the running program via a path of pointer traversal.

# ObjectSpace

A module for interacting with garbage collection and traversing all living objects.

```
ObjectSpace.count_objects

=> { :TOTAL     => 14716,
     :FREE      => 317,
     :T_OBJECT  => 8,
     :T_CLASS   => 478,
     :T_MODULE  => 21,
     :T_FLOAT   => 7,
     :T_STRING  => 6314,
     :T_REGEXP  => 24,
     :T_ARRAY   => 996,
     :T_HASH    => 14,
     :T_BIGNUM  => 3,
     :T_FILE    => 9,
     :T_DATA    => 402,
     :T_MATCH   => 104,
     :T_COMPLEX => 1,
     :T_NODE    => 5993,
     :T_ICLASS  => 19 }
```

```
ObjectSpace.count_objects

=> { :TOTAL      => 14716,
     :FREE       => 317,
     :T_OBJECT   => 8,
     :T_CLASS    => 478,
     :T_MODULE   => 21,
     :T_FLOAT    => 7,
     :T_STRING   => 6314,
     :T_REGEXP   => 24,
     :T_ARRAY    => 996,
     :T_HASH     => 14,
     :T_BIGNUM   => 3,
     :T_FILE     => 9,
     :T_DATA     => 402,
     :T_MATCH    => 104,
     :T_COMPLEX  => 1,
     :T_NODE     => 5993,
     :T_ICLASS   => 19 }
```

# Create a Class

```
#!/usr/local/ruby

Foo = Class.new
```

```
ObjectSpace.count_objects

=> { :TOTAL      => 14718,
     :FREE       => 317,
     :T_OBJECT   => 8,
     :T_CLASS    => 480,
     :T_MODULE   => 21,
     :T_FLOAT    => 7,
     :T_STRING   => 6314,
     :T_REGEXP   => 24,
     :T_ARRAY    => 996,
     :T_HASH     => 14,
     :T_BIGNUM   => 3,
     :T_FILE     => 9,
     :T_DATA     => 402,
     :T_MATCH    => 104,
     :T_COMPLEX  => 1,
     :T_NODE     => 5993,
     :T_ICLASS   => 19 }
```

```
ObjectSpace.count_objects

=> { :TOTAL     => 14718,
     :FREE      => 317,
     :T_OBJECT  => 8,
     :T_CLASS   => 480,
     :T_MODULE  => 21,
     :T_FLOAT   => 7,
     :T_STRING  => 6314,
     :T_REGEXP  => 24,
     :T_ARRAY   => 996,
     :T_HASH    => 14,
     :T_BIGNUM  => 3,
     :T_FILE    => 9,
     :T_DATA    => 402,
     :T_MATCH   => 104,
     :T_COMPLEX => 1,
     :T_NODE    => 5993,
     :T_ICLASS  => 19 }
```

# Create an Array

```
#!/usr/local/ruby

Foo = Class.new

objs = []
```

```
ObjectSpace.count_objects

=> { :TOTAL     => 14719,
     :FREE      => 317,
     :T_OBJECT  => 8,
     :T_CLASS   => 480,
     :T_MODULE  => 21,
     :T_FLOAT   => 7,
     :T_STRING  => 6314,
     :T_REGEXP  => 24,
     :T_ARRAY   => 997,
     :T_HASH    => 14,
     :T_BIGNUM  => 3,
     :T_FILE    => 9,
     :T_DATA    => 402,
     :T_MATCH   => 104,
     :T_COMPLEX => 1,
     :T_NODE    => 5993,
     :T_ICLASS  => 19 }
```

```
ObjectSpace.count_objects

=> { :TOTAL      => 14719,
     :FREE       => 317,
     :T_OBJECT   => 8,
     :T_CLASS    => 480,
     :T_MODULE   => 21,
     :T_FLOAT    => 7,
     :T_STRING   => 6314,
     :T_REGEXP   => 24,
     :T_ARRAY    => 997,
     :T_HASH     => 14,
     :T_BIGNUM   => 3,
     :T_FILE     => 9,
     :T_DATA     => 402,
     :T_MATCH    => 104,
     :T_COMPLEX  => 1,
     :T_NODE     => 5993,
     :T_ICLASS   => 19 }
```

# Fill it with Objects

```ruby
#!/usr/local/ruby

Foo = Class.new

objs = []

10000.times do |i|
  objs << Foo.new
end
```

```
ObjectSpace.count_objects

=> { :TOTAL      => 24719,
     :FREE       => 317,
     :T_OBJECT   => 10008,
     :T_CLASS    => 480,
     :T_MODULE   => 21,
     :T_FLOAT    => 7,
     :T_STRING   => 6314,
     :T_REGEXP   => 24,
     :T_ARRAY    => 997,
     :T_HASH     => 14,
     :T_BIGNUM   => 3,
     :T_FILE     => 9,
     :T_DATA     => 402,
     :T_MATCH    => 104,
     :T_COMPLEX  => 1,
     :T_NODE     => 5993,
     :T_ICLASS   => 19 }
```

# Try Garbage Collect

```ruby
#!/usr/local/ruby

Foo = Class.new

objs = []

10000.times do |i|
  objs << Foo.new
end
ObjectSpace.garbage_collect
```

```
ObjectSpace.count_objects

=> { :TOTAL      => 14716,
     :FREE       => 317,
     :T_OBJECT   => 10008,
     :T_CLASS    => 480,
     :T_MODULE   => 21,
     :T_FLOAT    => 7,
     :T_STRING   => 6314,
     :T_REGEXP   => 24,
     :T_ARRAY    => 997,
     :T_HASH     => 14,
     :T_BIGNUM   => 3,
     :T_FILE     => 9,
     :T_DATA     => 402,
     :T_MATCH    => 104,
     :T_COMPLEX  => 1,
     :T_NODE     => 5993,
     :T_ICLASS   => 19 }
```

# Empty the Array

```ruby
#!/usr/local/ruby

Foo = Class.new

objs = []

10000.times do |i|
  objs << Foo.new
end
ObjectSpace.garbage_collect

objs = []
```

```
ObjectSpace.count_objects

=> { :TOTAL      => 14716,
     :FREE       => 317,
     :T_OBJECT   => 10008,
     :T_CLASS    => 480,
     :T_MODULE   => 21,
     :T_FLOAT    => 7,
     :T_STRING   => 6314,
     :T_REGEXP   => 24,
     :T_ARRAY    => 997,
     :T_HASH     => 14,
     :T_BIGNUM   => 3,
     :T_FILE     => 9,
     :T_DATA     => 402,
     :T_MATCH    => 104,
     :T_COMPLEX  => 1,
     :T_NODE     => 5993,
     :T_ICLASS   => 19 }
```

# Try GC Again

```ruby
#!/usr/local/ruby

Foo = Class.new

objs = []

10000.times do |i|
  objs << Foo.new
end
ObjectSpace.garbage_collect

objs = []
ObjectSpace.garbage_collect
```

```
ObjectSpace.count_objects

=> { :TOTAL     => 14719,
     :FREE      => 317,
     :T_OBJECT  => 8,
     :T_CLASS   => 480,
     :T_MODULE  => 21,
     :T_FLOAT   => 7,
     :T_STRING  => 6314,
     :T_REGEXP  => 24,
     :T_ARRAY   => 997,
     :T_HASH    => 14,
     :T_BIGNUM  => 3,
     :T_FILE    => 9,
     :T_DATA    => 402,
     :T_MATCH   => 104,
     :T_COMPLEX => 1,
     :T_NODE    => 5993,
     :T_ICLASS  => 19 }
```

```
ObjectSpace.count_objects

=> { :TOTAL     => 14719,
     :FREE      => 317,
     :T_OBJECT  => 8,
     :T_CLASS   => 480,
     :T_MODULE  => 21,
     :T_FLOAT   => 7,
     :T_STRING  => 6314,
     :T_REGEXP  => 24,
     :T_ARRAY   => 997,
     :T_HASH    => 14,
     :T_BIGNUM  => 3,
     :T_FILE    => 9,
     :T_DATA    => 402,
     :T_MATCH   => 104,
     :T_COMPLEX => 1,
     :T_NODE    => 5993,
     :T_ICLASS  => 19 }
```

# Remove the Class

```ruby
#!/usr/local/ruby

Foo = Class.new

objs = []

10000.times do |i|
  objs << Foo.new
end
ObjectSpace.garbage_collect

objs = []
ObjectSpace.garbage_collect

Object.send(:remove_const, :Foo)
ObjectSpace.garbage_collect
```

```
ObjectSpace.count_objects

=> { :TOTAL     => 14716,
     :FREE      => 317,
     :T_OBJECT  => 8,
     :T_CLASS   => 478,
     :T_MODULE  => 21,
     :T_FLOAT   => 7,
     :T_STRING  => 6314,
     :T_REGEXP  => 24,
     :T_ARRAY   => 997,
     :T_HASH    => 14,
     :T_BIGNUM  => 3,
     :T_FILE    => 9,
     :T_DATA    => 402,
     :T_MATCH   => 104,
     :T_COMPLEX => 1,
     :T_NODE    => 5993,
     :T_ICLASS  => 19 }
```

# 2

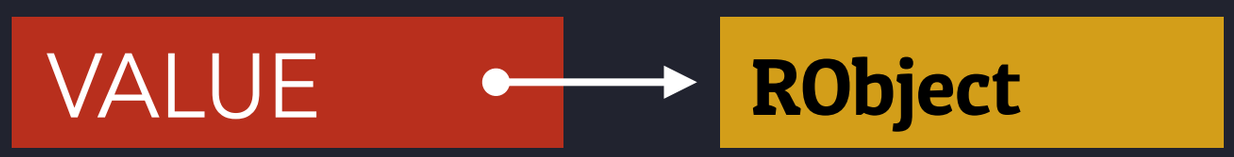
# Navigating CRuby.

# Ruby, Written in C

include/ruby/ruby.h
vm_method.c
object.c
gc.c

# VALUE and Objects

VALUE, an unsigned long, is a pointer to Ruby's objects.



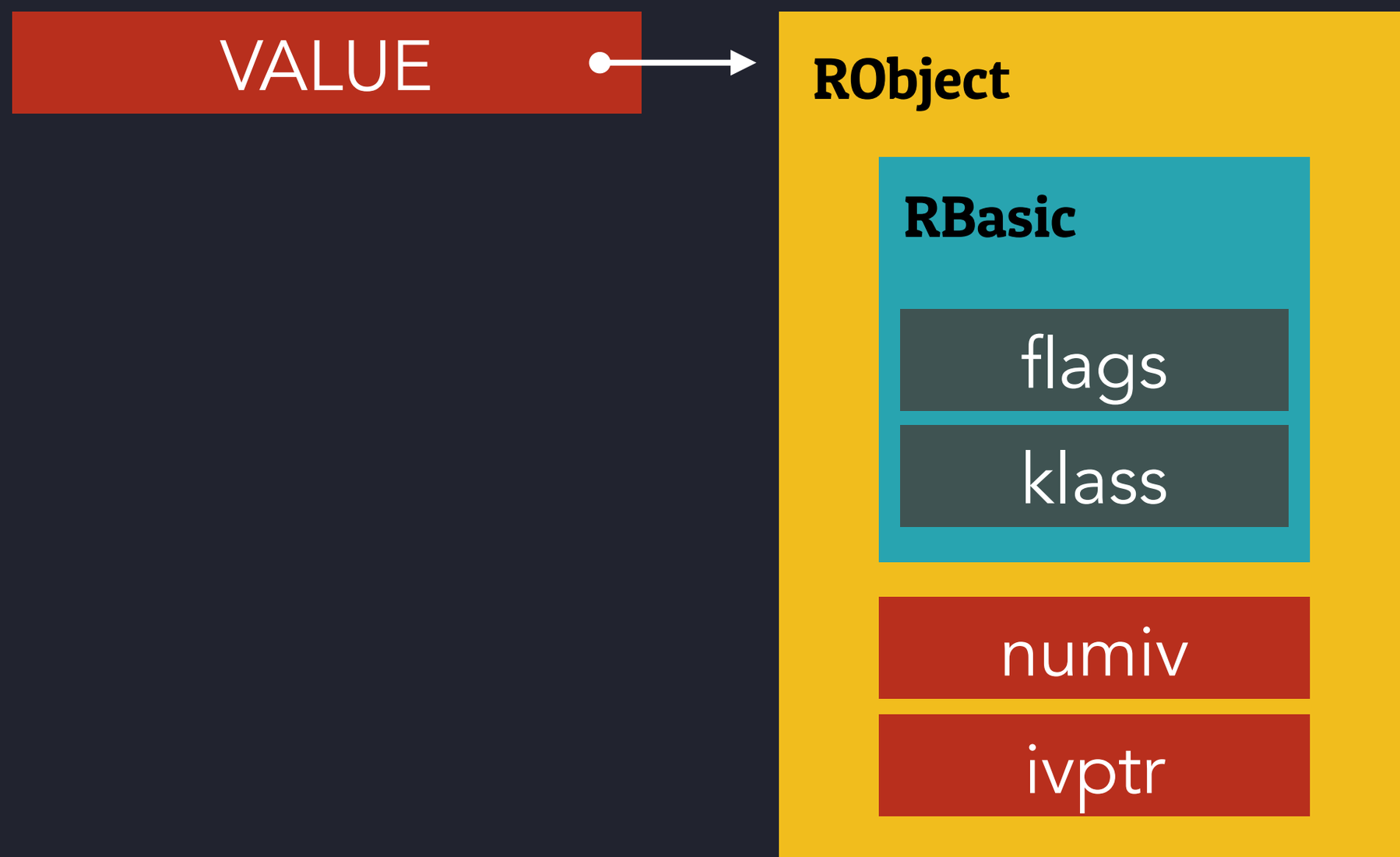
# Object Types

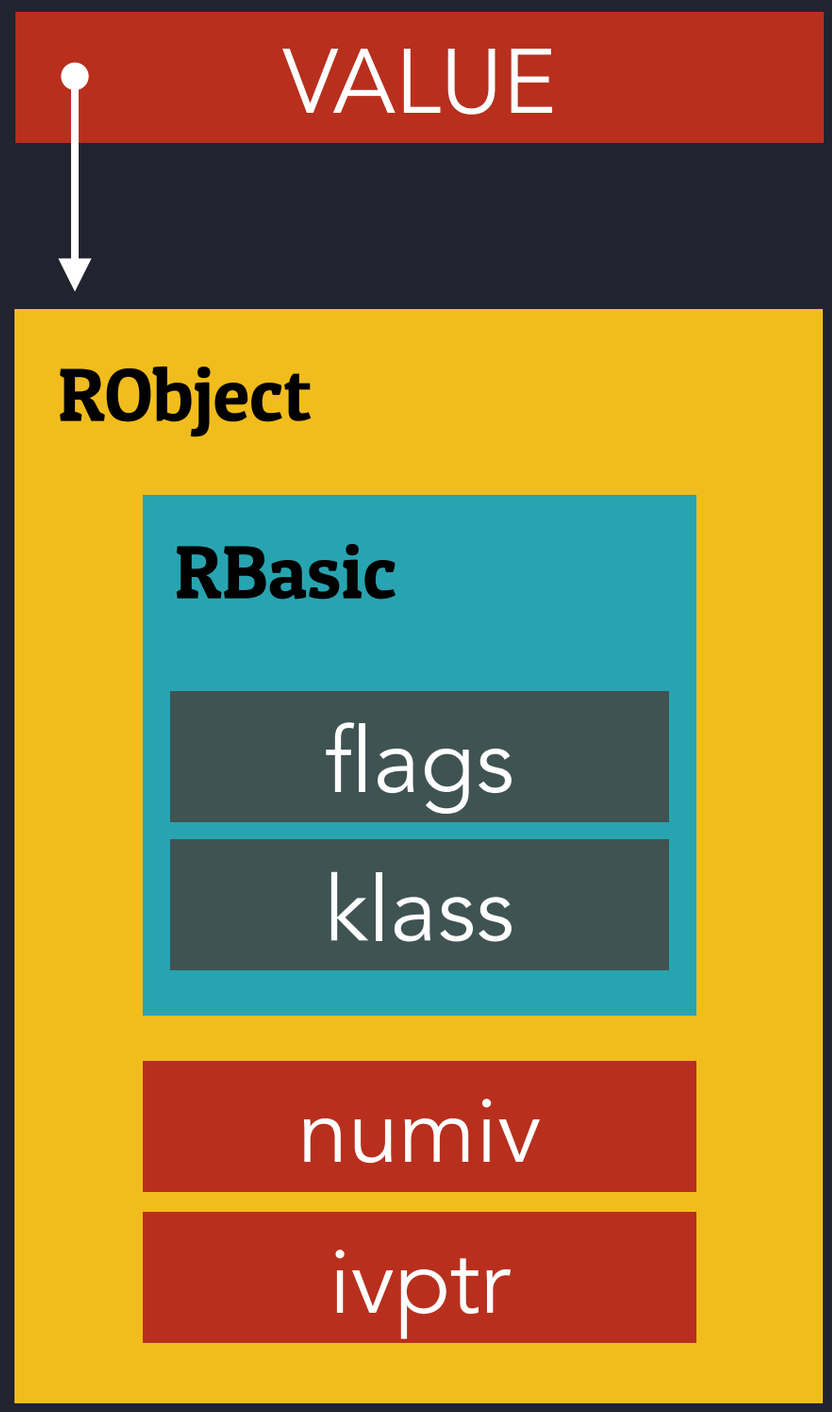
```
struct RBasic  basic;
struct RObject object;
struct RClass  klass;
struct RFloat  flonum;
struct RString string;
struct RArray  array;
struct RRegexp regexp;
struct RHash   hash;
struct RData   data;
struct RTypedData   typeddata;
struct RStruct rstruct;
struct RBignum bignum;
struct RFile   file;
struct RNode   node;
struct RMatch  match;
struct RRational rational;
struct RComplex complex;

/* gc.c */
```

# RBasic and RObject

```
struct RBasic {
    VALUE flags;
    VALUE klass;
};

struct RObject {
    struct RBasic basic;
    union {
        struct {
            long numiv;
            VALUE *ivptr;
            struct st_table *iv_index_tbl;
        } heap;
        VALUE ary[ROBJECT_EMBED_LEN_MAX];
    } as;
};

/* include/ruby/ruby.h */
```

# RObject Structure






```
struct RBasic {
    VALUE flags;
    VALUE klass;
};

struct RObject {
    struct RBasic basic;
    union {
        struct {
            long numiv;
            VALUE *ivptr;
            struct st_table *iv_...
        } heap;
        VALUE ary[ROBJECT_EMBED_...
    } as;
};

/* include/ruby/ruby.h */
```

# Ruby and Macros

Understanding C macros is essential to understanding Ruby source.

# RString Magic

```
struct RString {
    struct RBasic basic;
    union {
        struct {
            long len;
            char *ptr;
            union {
                long capa;
                VALUE shared;
            } aux;
        } heap;
        char ary[RSTRING_EMBED_LEN_MAX + 1];
    } as;
};

/* include/ruby/ruby.h */
```

# RString Magic

```
struct RString {
    struct RBasic basic;
    union {
        struct {
            long len;
            char *ptr;
            union {
                long capa;
                VALUE shared;
            } aux;
        } heap;
        char ary[RSTRING_EMBED_LEN_MAX + 1];
    } as;
};

/* include/ruby/ruby.h */
```

# RString Magic

```
struct RString {
    struct RBasic basic;
    union {
        struct {
            long len;
            char *ptr;
            union {
                long capa;
                VALUE shared;
            } aux;
        } heap;
        char ary[RSTRING_EMBED_LEN_MAX + 1];
    } as;
};

/* include/ruby/ruby.h */
```

# RString Macros

```
#define R_CAST(st)   (struct st*)
#define RSTRING(obj) (R_CAST(RString)(obj))

#define RSTRING_PTR(str) \
    (!(RBASIC(str)->flags & RSTRING_NOEMBED) ? \
     RSTRING(str)->as.ary : \
     RSTRING(str)->as.heap.ptr)
```

```
/* include/ruby/ruby.h */
```

# RString Macros

```
#define R_CAST(st)   (struct st*)
#define RSTRING(obj) (R_CAST(RString)(obj))

#define RSTRING_PTR(str) \
    (!(RBASIC(str)->flags & RSTRING_NOEMBED) ? \
     RSTRING(str)->as.ary : \
     RSTRING(str)->as.heap.ptr)
```

```
/* include/ruby/ruby.h */
```

# RString Macros

```
#define R_CAST(st)   (struct st*)
#define RSTRING(obj) (R_CAST(RString)(obj))

#define RSTRING_PTR(str) \
    (!(RBASIC(str)->flags & RSTRING_NOEMBED) ? \
     RSTRING(str)->as.ary : \
     RSTRING(str)->as.heap.ptr)
```

```
/* include/ruby/ruby.h */
```

# RString Macros

```
#define R_CAST(st)   (struct st*)
#define RSTRING(obj) (R_CAST(RString)(obj))

#define RSTRING_PTR(str) \
    (!(RBASIC(str)->flags & RSTRING_NOEMBED) ? \
     RSTRING(str)->as.ary : \
     RSTRING(str)->as.heap.ptr)
```

```
/* include/ruby/ruby.h */
```

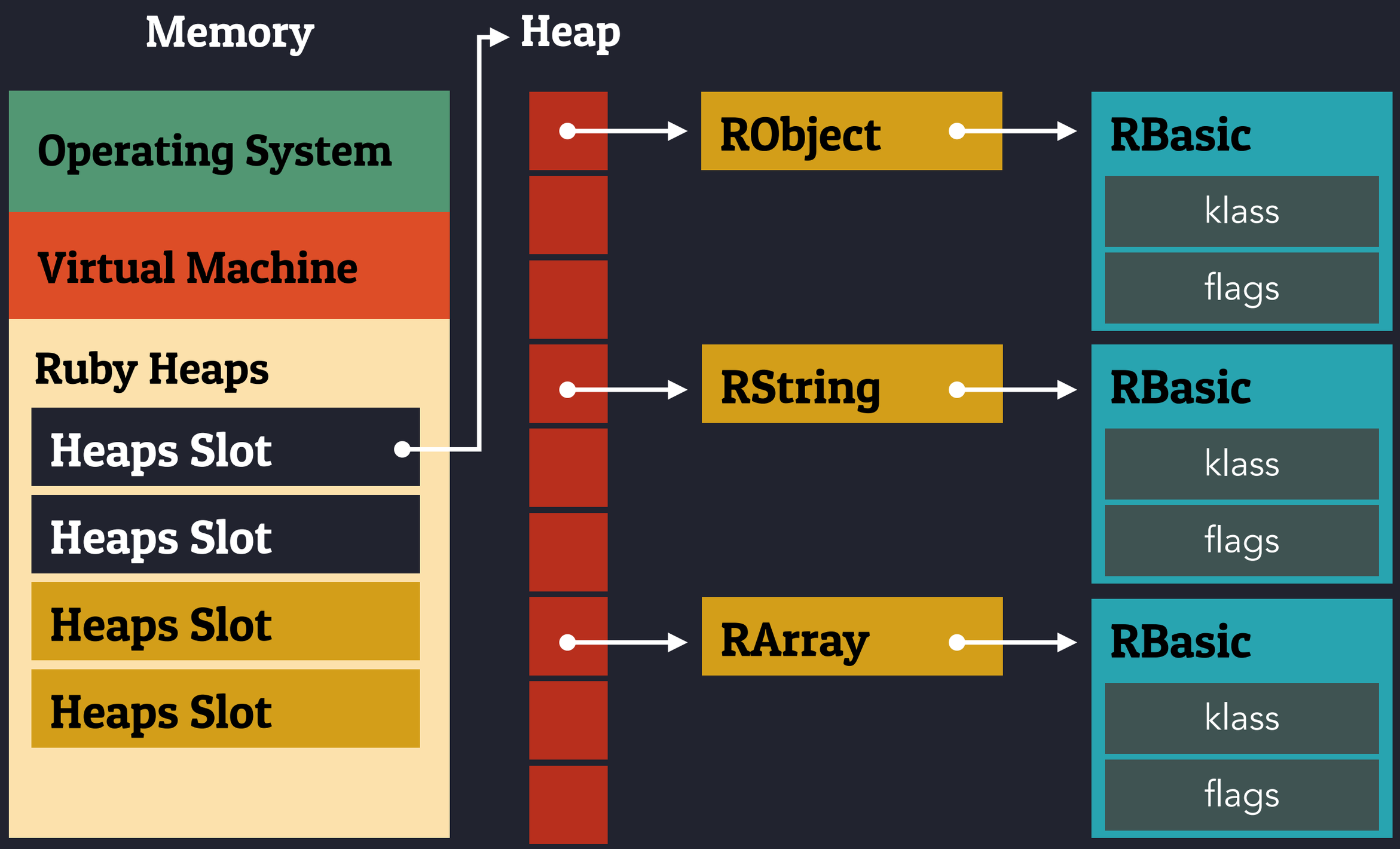
# RString Macros

```
#define R_CAST(st)   (struct st*)
#define RSTRING(obj) (R_CAST(RString)(obj))

#define RSTRING_PTR(str) \
    (!(RBASIC(str)->flags & RSTRING_NOEMBED) ? \
     RSTRING(str)->as.ary : \
     RSTRING(str)->as.heap.ptr)
```

```
struct RString(str)->as.heap.prt
```

```
/* include/ruby/ruby.h */
```

# Ruby Heaps


Memory
Heap
Operating System
Virtual Machine
Ruby Heaps
Heaps Slot
Heaps Slot
Heaps Slot
Heaps Slot
RObject
RBasic
klass
flags
RString
RBasic
klass
flags
RArray
RBasic
klass
flags

# 3

## Object Creation.

# Class#new

```
VALUE
rb_class_new_instance(int argc, VALUE *argv, VALUE klass)
{
    VALUE obj;

    obj = rb_obj_alloc(klass);
    rb_obj_call_init(obj, argc, argv);

    return obj;
}
```

```
/* object.c */
```

# Object Allocation

```
VALUE
rb_obj_alloc(VALUE klass)
{
    VALUE obj;
    rb_alloc_func_t allocator;
    /* ... */
    allocator = rb_get_alloc_func(klass);
    /* ... */
    obj = (*allocator)(klass);
    /* ... */
    return obj;
}
```

```
/* object.c */
```

# Find Alloc Method

```
rb_alloc_func_t
rb_get_alloc_func(VALUE klass)
{
    Check_Type(klass, T_CLASS);

    for (; klass; klass = RCLASS_SUPER(klass)) {
    rb_alloc_func_t allocator = RCLASS_EXT(klass)->allocator;
        if (allocator == UNDEF_ALLOC_FUNC) break;
        if (allocator) return allocator;
    }
    return 0;
}
```

```
/* vm_method.c */
```

# Define Alloc Method

```
void
rb_define_alloc_func(VALUE klass, VALUE (*func)(VALUE))
{
    Check_Type(klass, T_CLASS);
    RCLASS_EXT(klass)->allocator = func;
}
```

```
/* vm_method.c */
```

# Define Alloc Method

```
void
Init_Object(void)
{
    /* ... */
    rb_define_private_method(rb_cBasicObject,
                             "initialize", rb_obj_dummy, 0);
    rb_define_alloc_func(rb_cBasicObject,
                         rb_class_allocate_instance);
    rb_define_method(rb_cBasicObject, "==", rb_obj_equal, 1);
    rb_define_method(rb_cBasicObject, "equal?", rb_obj_equal, 1);
    rb_define_method(rb_cBasicObject, "!", rb_obj_not, 0);
    rb_define_method(rb_cBasicObject, "!=", rb_obj_not_equal, 1);
    /* ... */
}
```

```
/* object.c */
```

# Allocate Instance

```
static VALUE
rb_class_allocate_instance(VALUE klass)
{
    NEWOBJ_OF(obj, struct RObject, klass, T_OBJECT);
    return (VALUE)obj;
}
```

```
/* object.c */
```

# NEWOBJ_OF Macro

```
#define NEWOBJ_OF(obj,type,klass,flags) \
        type *(obj) = (type*)rb_newobj_of(klass, flags)
```

```
/* include/ruby/ruby.h */
```

# Create Ruby Object

```
VALUE
rb_newobj_of(VALUE klass, VALUE flags)
{
    VALUE obj;

    obj = newobj(klass, flags);
    OBJSETUP(obj, klass, flags);

    return obj;
}
```

```
/* gc.c */
```

# Get Object Space

```
static VALUE
newobj(VALUE klass, VALUE flags)
{
    rb_objspace_t *objspace = &rb_objspace;
    VALUE obj;
```

```
static VALUE
newobj(VALUE klass, VALUE flag
{
    rb_objspace_t *objspace =
    VALUE obj;

    if (UNLIKELY(during_gc)) {
       dont_gc = 1;
       during_gc = 0;
       rb_bug("object allocati
    }

    if (UNLIKELY(ruby_gc_stres
       if (!garbage_collect(ob
           during_gc = 0;
           rb_memerror();
       }
    }

    if (UNLIKELY(!has_free_obj
       if (!gc_prepare_free_ob
           during_gc = 0;
           rb_memerror();
       }
    }

    obj = (VALUE)objspace->hea
    objspace->heap.free_slots-
    if (objspace->heap.free_sl
        unlink_free_heap_slot(
    }

    MEMZERO((void*)obj, RVALUE
#ifdef GC_DEBUG
    RANY(obj)->file = rb_sourc
    RANY(obj)->line = rb_sourc
#endif
    objspace->total_allocated_

    return obj;
}
```

/* gc.c */

# Try Garbage Collect

```
if (UNLIKELY(ruby_gc_stress &&
                !ruby_disable_gc_stress)) {
    if (!garbage_collect(objspace)) {
        during_gc = 0;
        rb_memerror();
    }
}
```

```
static VALUE
newobj(VALUE klass, VALUE flag
{
    rb_objspace_t *objspace =
    VALUE obj;

    if (UNLIKELY(during_gc)) {
       dont_gc = 1;
       during_gc = 0;
       rb_bug("object allocati
    }

    if (UNLIKELY(ruby_gc_stres
       if (!garbage_collect(ob
           during_gc = 0;
           rb_memerror();
       }
    }

    if (UNLIKELY(!has_free_obj
       if (!gc_prepare_free_ob
           during_gc = 0;
           rb_memerror();
       }
    }

    obj = (VALUE)objspace->hea
    objspace->heap.free_slots-
    if (objspace->heap.free_sl
        unlink_free_heap_slot(
    }

    MEMZERO((void*)obj, RVALUE
#ifdef GC_DEBUG
    RANY(obj)->file = rb_sourc
    RANY(obj)->line = rb_sourc
#endif
    objspace->total_allocated_

    return obj;
}
```

/* gc.c */

# Get the Next Free Slot

```
obj = (VALUE)objspace->heap.free_slots->freelist;
objspace->heap.free_slots->freelist =
                        RANY(obj)->as.free.next;
if (objspace->heap.free_slots->freelist == NULL) {
    unlink_free_heap_slot(objspace,
                          objspace->heap.free_slots);
}
```

```
static VALUE
newobj(VALUE klass, VALUE flag
{
    rb_objspace_t *objspace =
    VALUE obj;

    if (UNLIKELY(during_gc)) {
       dont_gc = 1;
       during_gc = 0;
       rb_bug("object allocati
    }

    if (UNLIKELY(ruby_gc_stres
       if (!garbage_collect(ob
           during_gc = 0;
           rb_memerror();
       }
    }

    if (UNLIKELY(!has_free_obj
       if (!gc_prepare_free_ob
           during_gc = 0;
           rb_memerror();
       }
    }

    obj = (VALUE)objspace->hea
    objspace->heap.free_slots-
    if (objspace->heap.free_sl
        unlink_free_heap_slot(
    }

    MEMZERO((void*)obj, RVALUE
#ifdef GC_DEBUG
    RANY(obj)->file = rb_sourc
    RANY(obj)->line = rb_sourc
#endif
    objspace->total_allocated_

    return obj;
}
```

/* gc.c */

# Return the Object

```
    MEMZERO((void*)obj, RVALUE, 1);
#ifdef GC_DEBUG
    RANY(obj)->file = rb_sourcefile();
    RANY(obj)->line = rb_sourceline();
#endif

    objspace->total_allocated_object_num++;
    return obj;
}
```

```
static VALUE
newobj(VALUE klass, VALUE flag
{
    rb_objspace_t *objspace =
    VALUE obj;

    if (UNLIKELY(during_gc)) {
       dont_gc = 1;
       during_gc = 0;
       rb_bug("object allocati
    }

    if (UNLIKELY(ruby_gc_stres
       if (!garbage_collect(ob
           during_gc = 0;
           rb_memerror();
       }
    }

    if (UNLIKELY(!has_free_obj
       if (!gc_prepare_free_ob
           during_gc = 0;
           rb_memerror();
       }
    }

    obj = (VALUE)objspace->hea
    objspace->heap.free_slots-
    if (objspace->heap.free_sl
        unlink_free_heap_slot(
    }

    MEMZERO((void*)obj, RVALUE
#ifdef GC_DEBUG
    RANY(obj)->file = rb_sourc
    RANY(obj)->line = rb_sourc
#endif
    objspace->total_allocated_

    return obj;
}
```

/* gc.c */

# Mark & Sweep

```
static int
garbage_collect(rb_objspace_t *objspace)
{
    if (GC_NOTIFY) printf("start garbage_collect()\n");
    if (!heaps) { return FALSE; }
    if (!ready_to_gc(objspace)) { return TRUE }
    /* ... */

    during_gc++;
    gc_marks(objspace);
    /* ... */
    gc_sweep(objspace);
    /* ... */
    if (GC_NOTIFY) printf("end garbage_collect()\n");
    return TRUE;
}
/* gc.c */
```

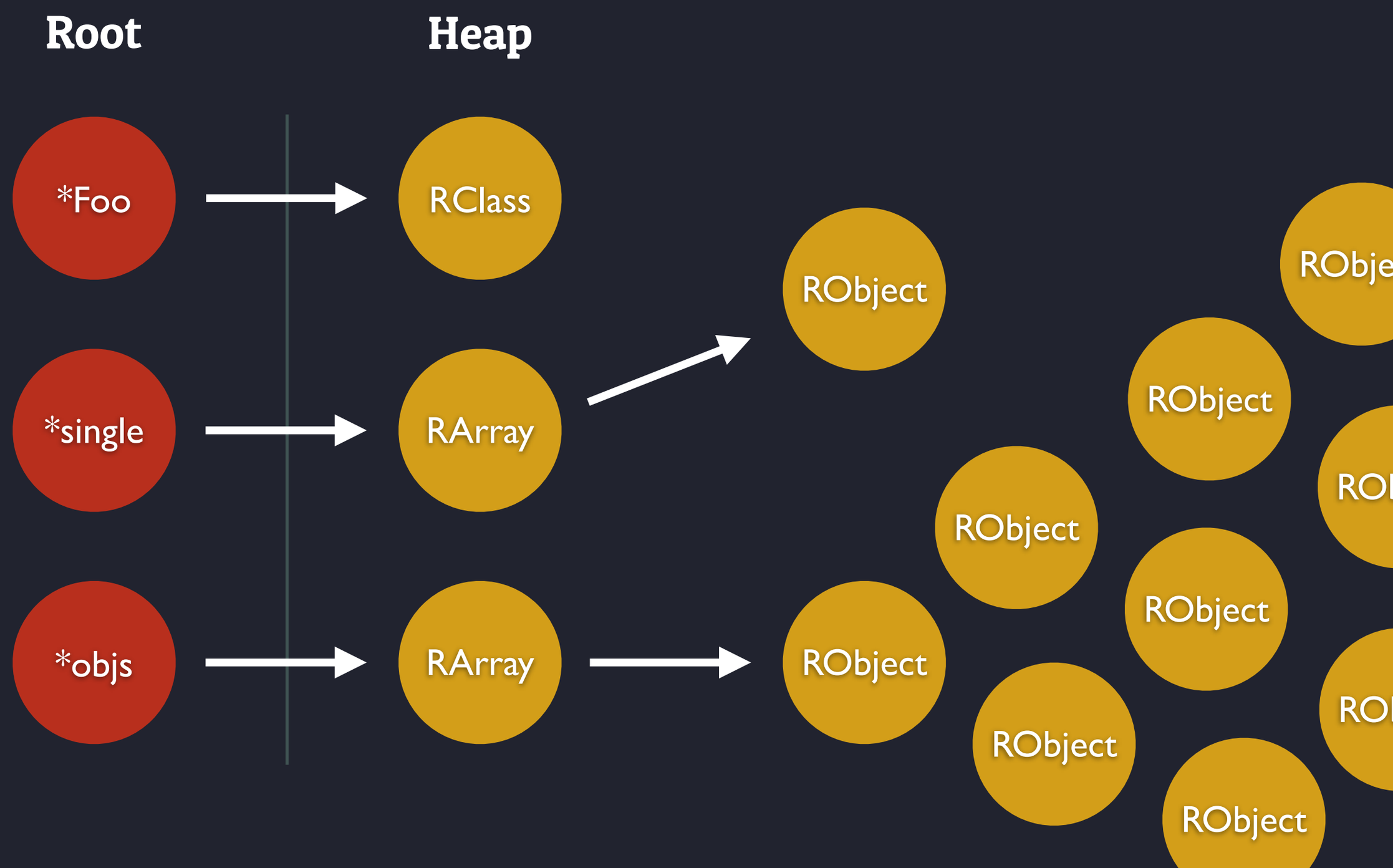
# 4

## Garbage Collection.

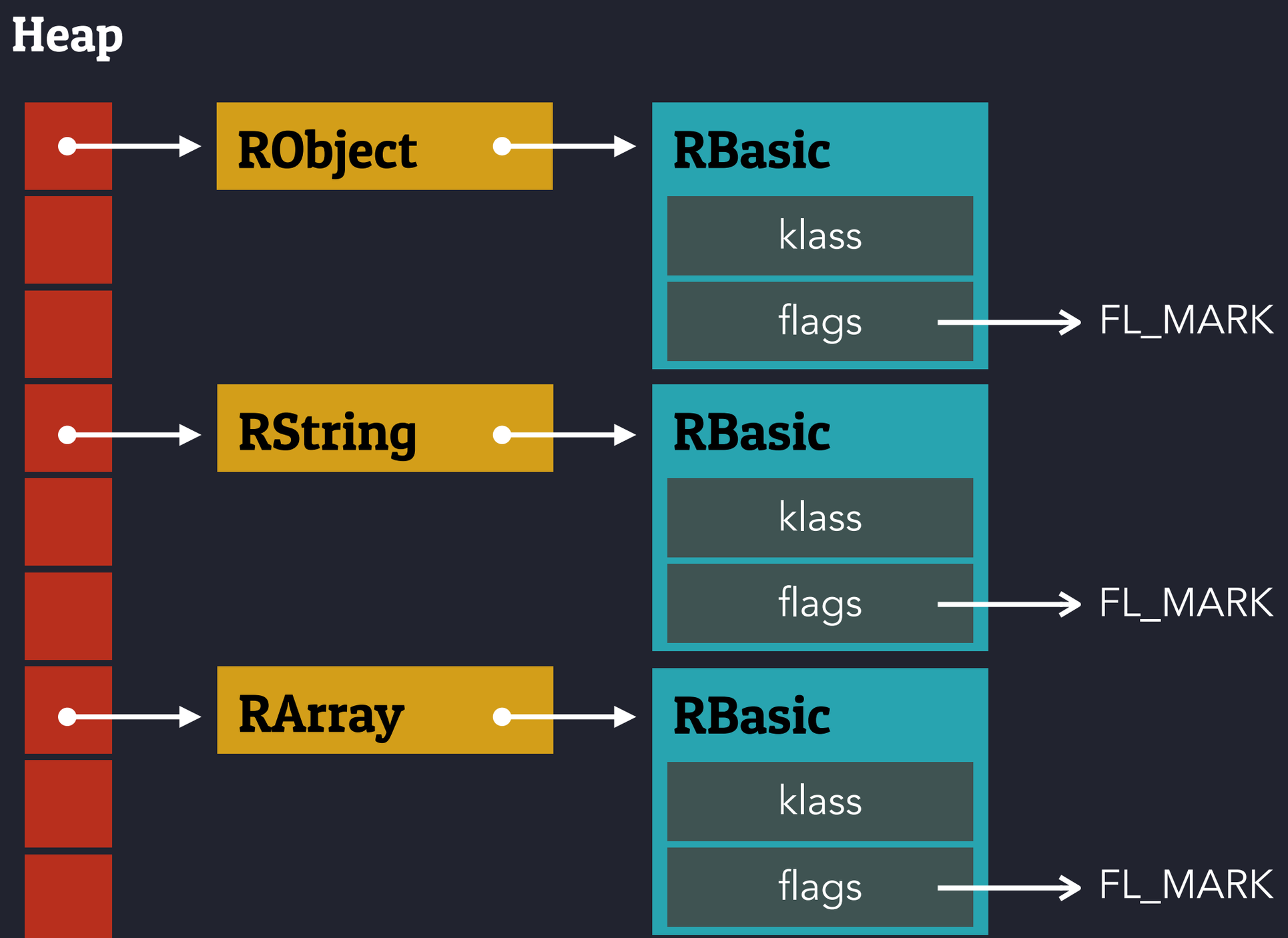
# Create Objects

```ruby
#!/usr/local/ruby

Foo = Class.new

single = []
single << Foo.new

objs = []

10000.times do |i|
  objs << Foo.new
end
```

# Object Allocation



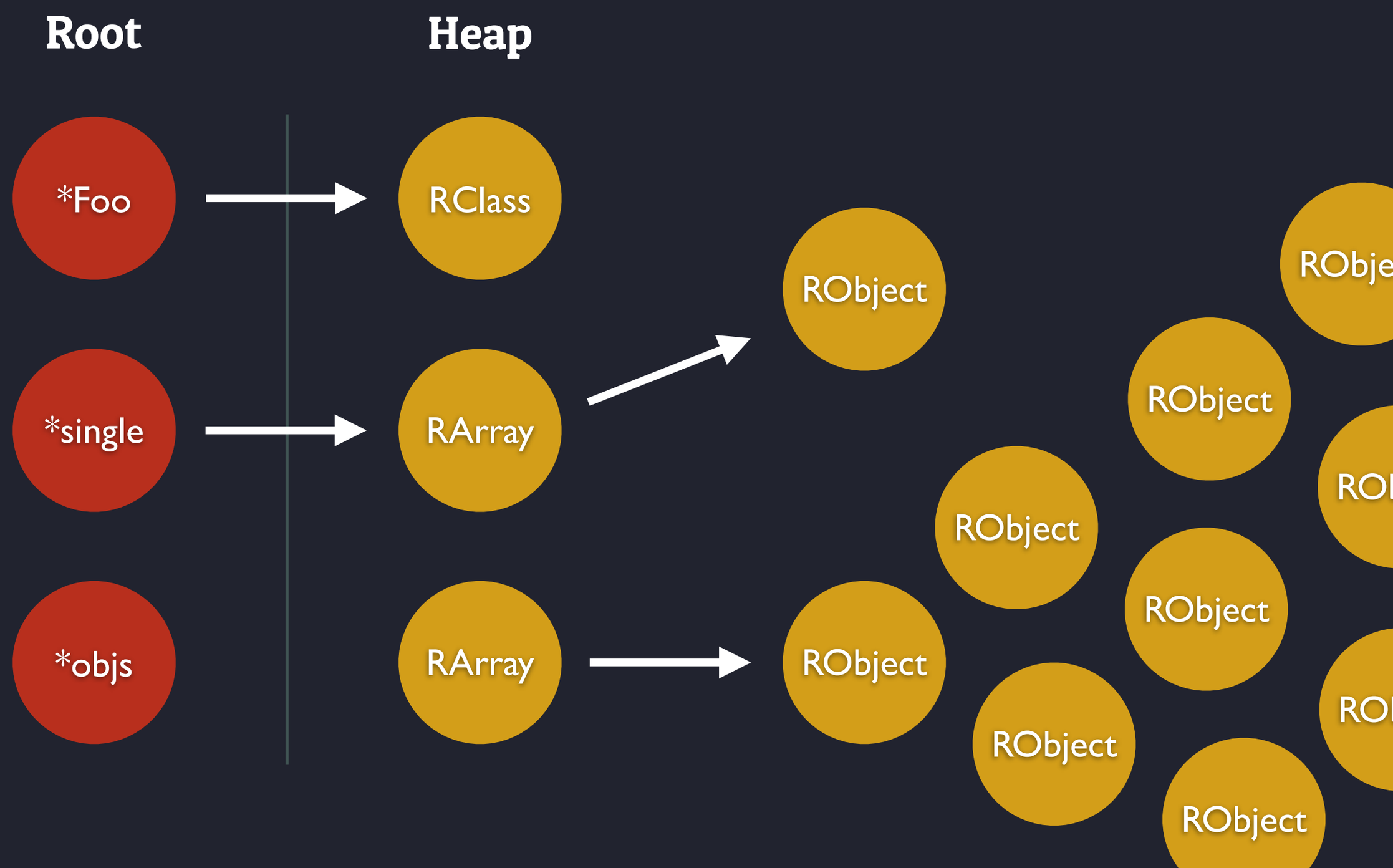
# Object Anatomy



# Create Garbage

```ruby
#!/usr/local/ruby

Foo = Class.new

single = []
single << Foo.new

objs = []

10000.times do |i|
  objs << Foo.new
end

objs = nil
```

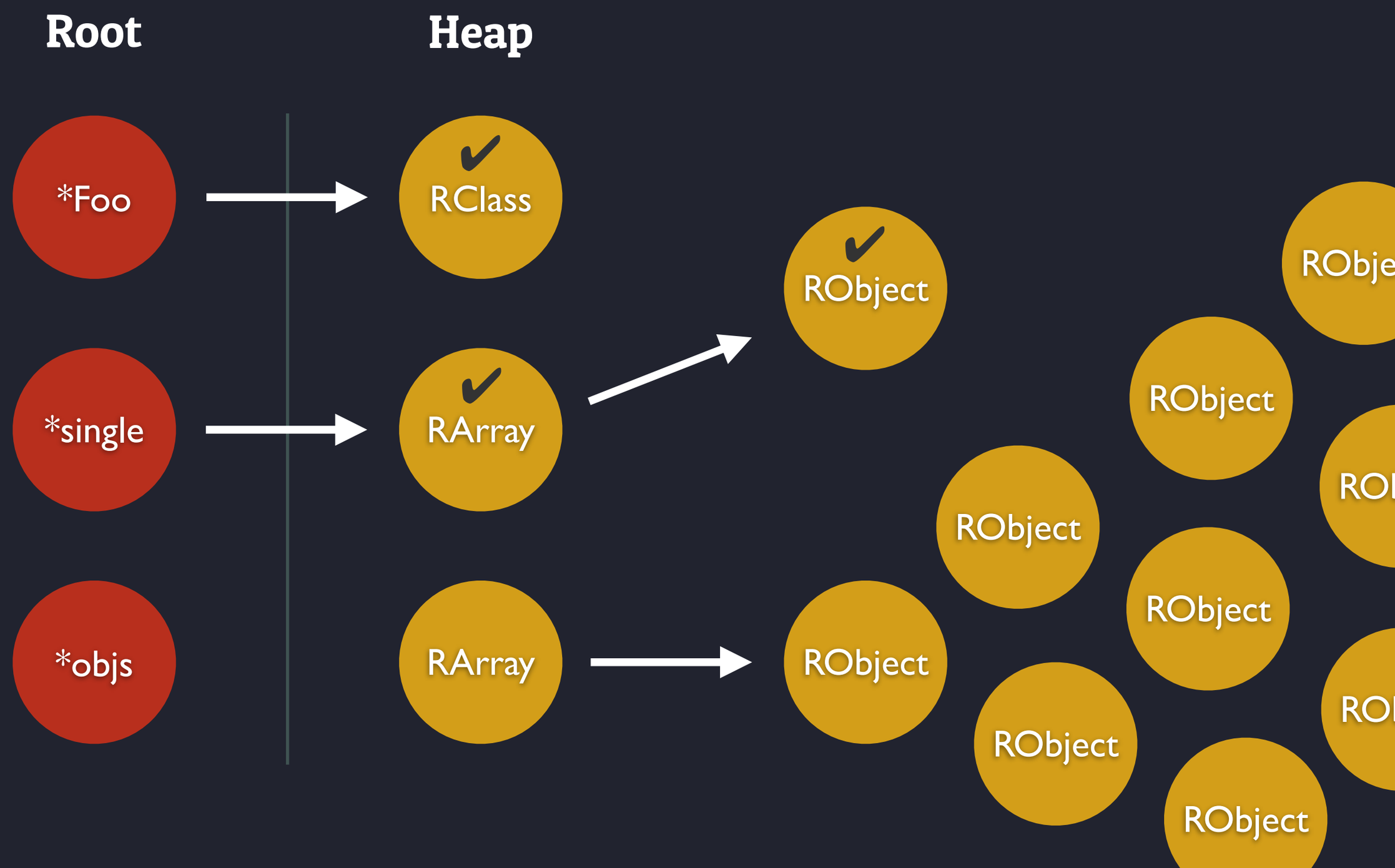
# Object Allocation



# Mark Phase

Traverse every object reachable from the Root set, and mark it.

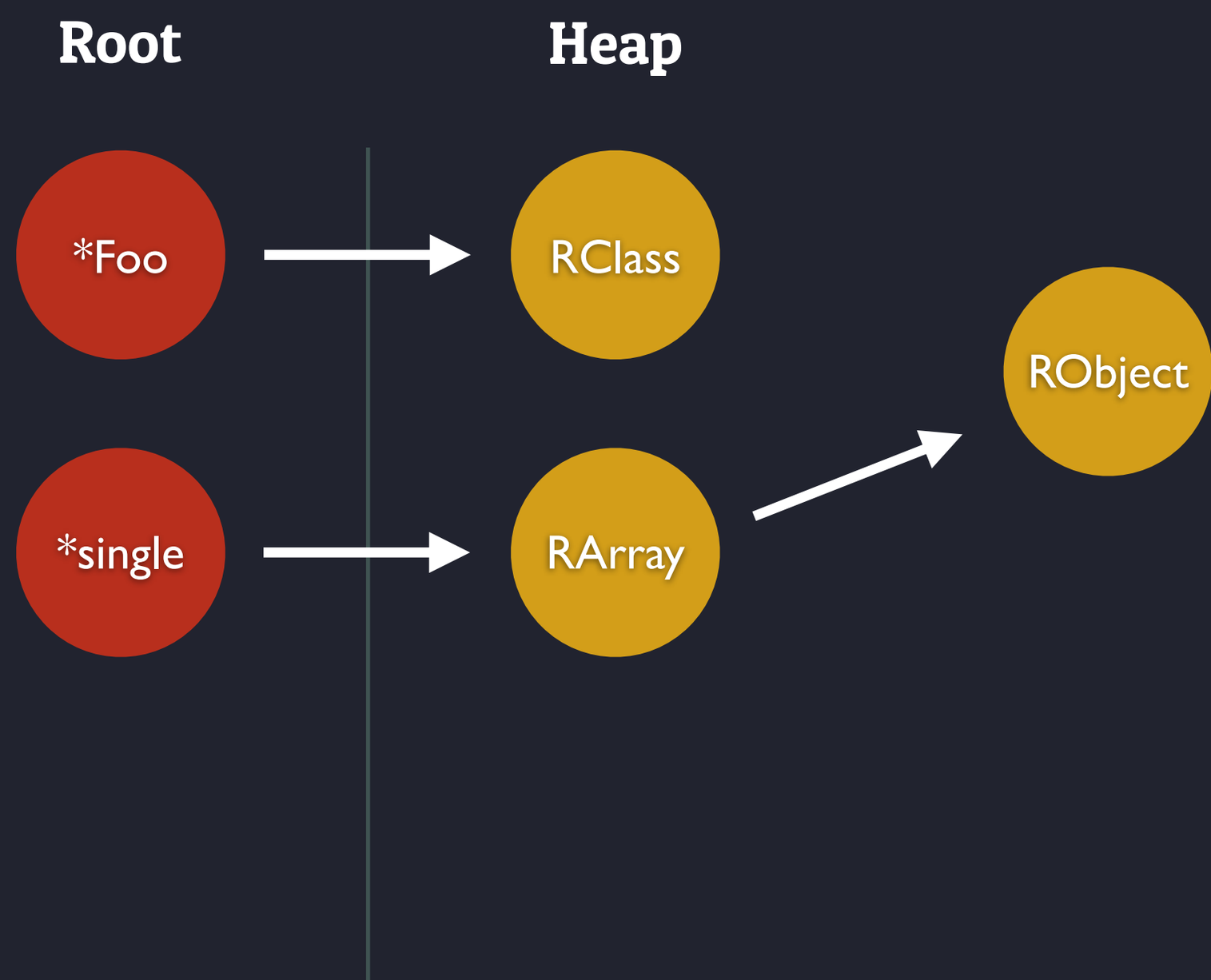
# Object Allocation



# Sweep Phase

Traverse every object in the object space. Those with a mark are unmarked. Those that are already unmarked are added to the free set.

# Object Allocation



# Stop the World

The mutator must stop while the garbage collector runs.

# Lazy Sweep

Sweep only in the case that there aren't any free objects and the heap cannot be increased.

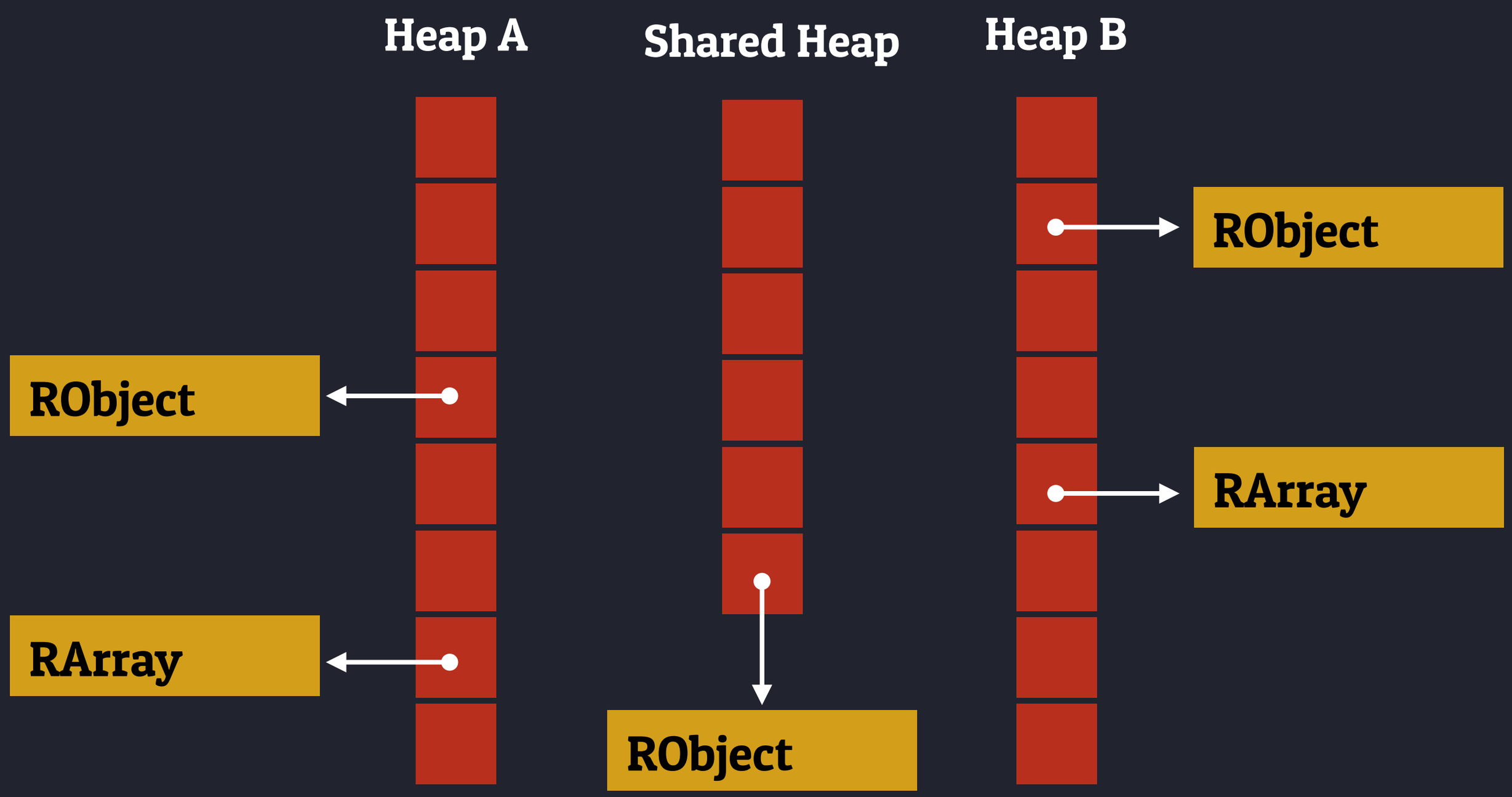
# GC Performance

Ordinary Mark & Sweep is expensive. Lazy Sweep only postpones the problem.

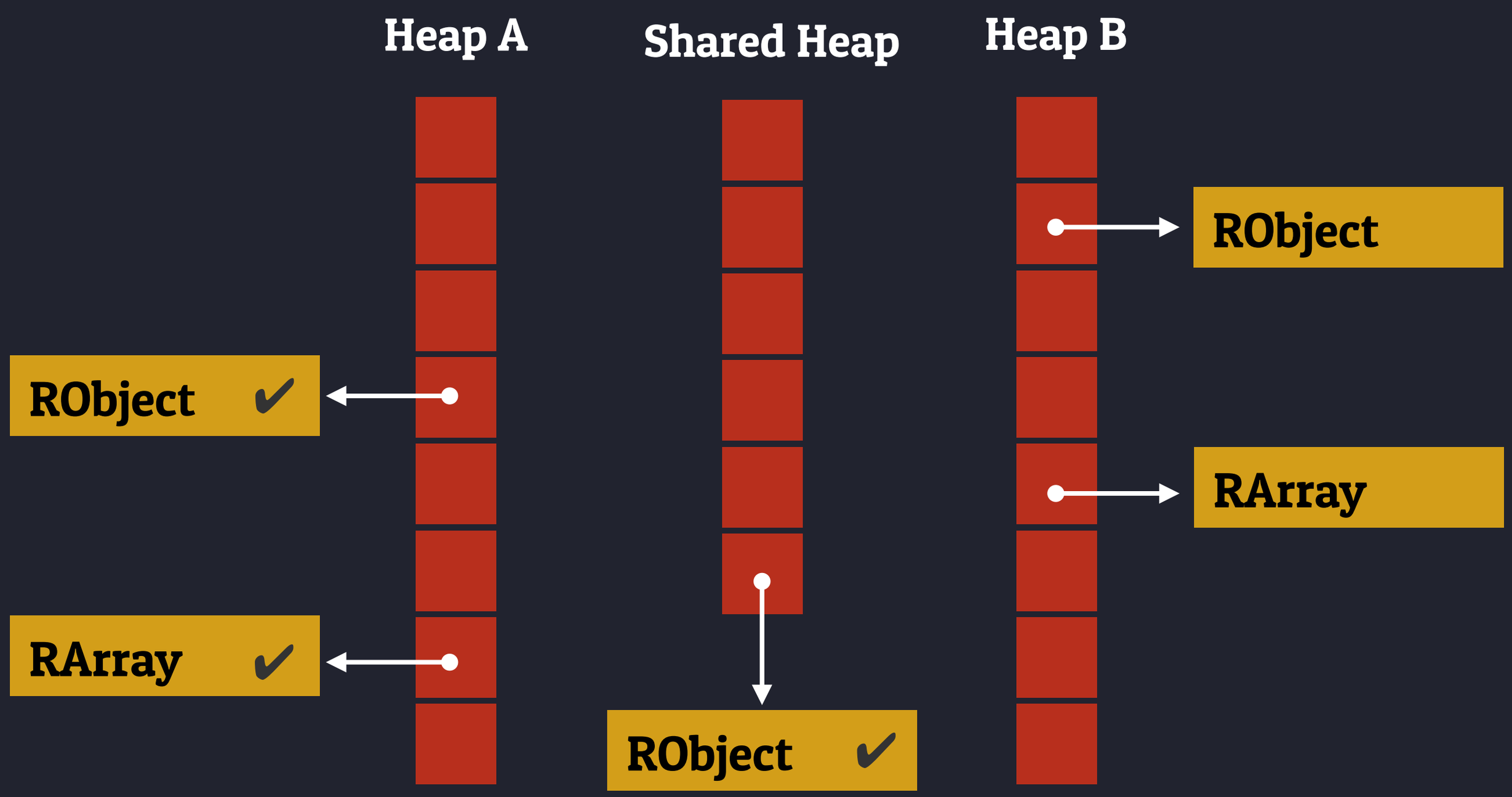
# Copy-on-Write

When a process is forked, memory is shared. Memory is copied only when it is changed. Marking an object is considered changing it.

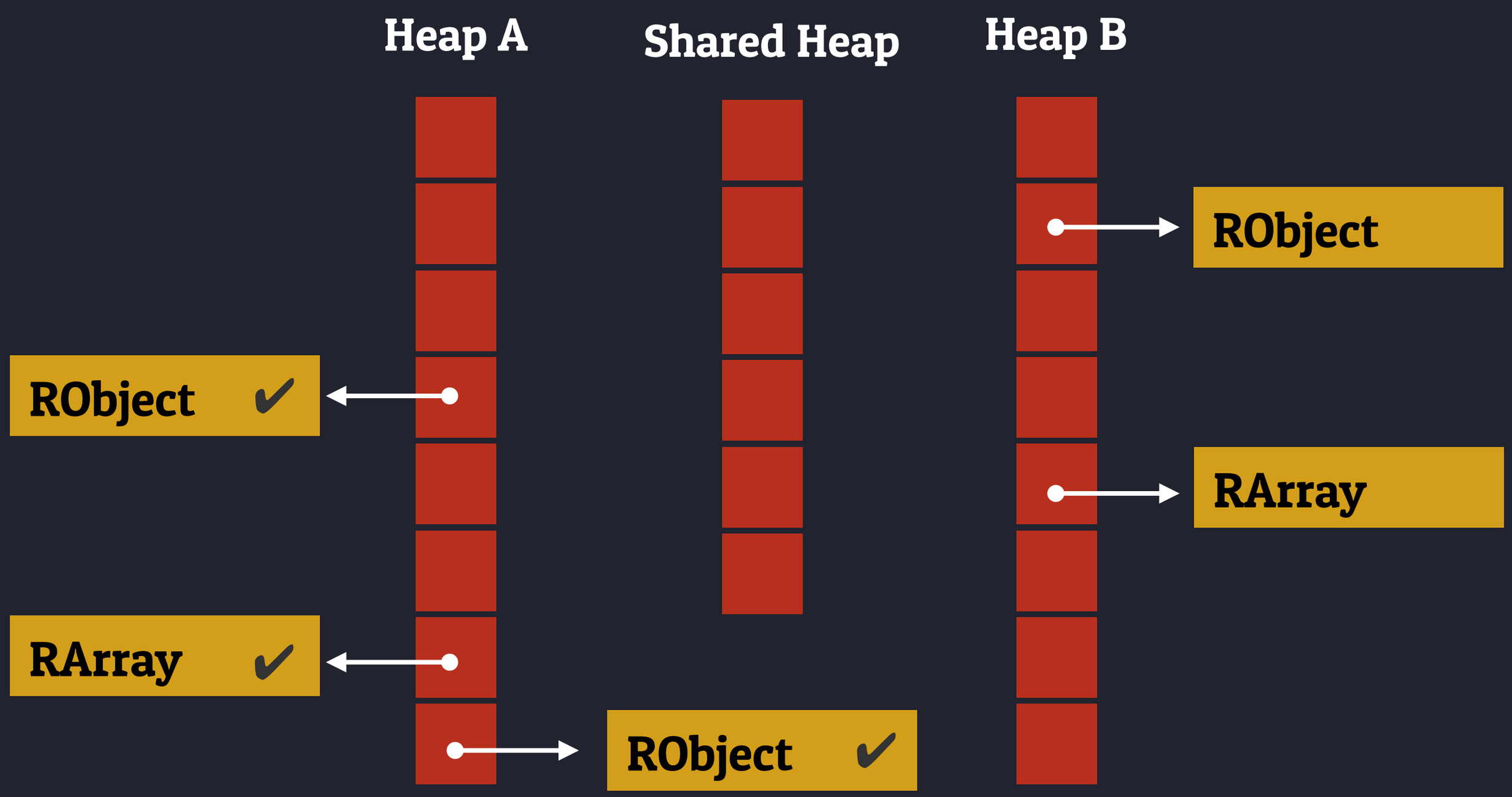
# CoW Anatomy


Heap A
Shared Heap
Heap B
RObject
RArray
RObject
RArray
RObject

# CoW Anatomy


Heap A
Shared Heap
Heap B
RObject ✔
RArray ✔
RObject ✔
RObject
RArray

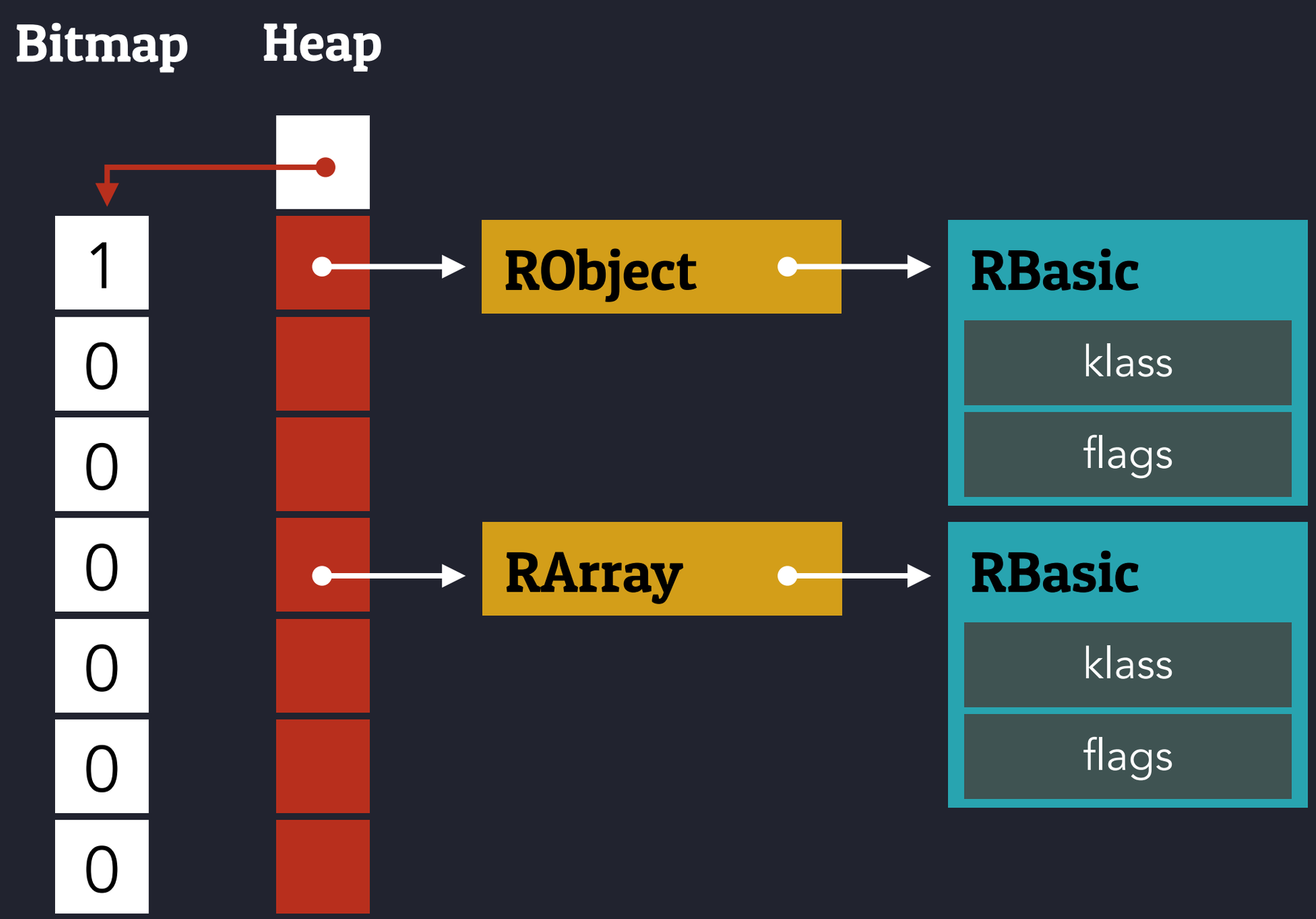
# CoW Anatomy


Heap A
Shared Heap
Heap B
RObject
RObject ✔
RArray
RArray ✔
RObject ✔

# Bitmap Marking

Ruby 2.0 introduces a new type of garbage collection. FL_MARK flag has been moved to a bitmap memory structure.

# New Heap Anatomy



# Now What?

# MRI's Slow?

Significant work has been done on GC. Method cache changes are coming. People are getting involved.

# Generational GC

Feature #8339: Introducing

https://bugs.ruby-lang.org/issues/8339

Home Projects Help Sign In Register

Ruby » ruby-trunk

Search:

Overview Activity Roadmap Issues Calendar Wiki Repository

## Feature #8339

### Introducing Geneartional Garbage Collection for CRuby/MRI

Added by ko1 (Koichi Sasada) 15 days ago. Updated 4 days ago.

[ruby-core:54621]

| | | | |
|---|---|---|---|
| **Status:** | Open | **Start date:** | 04/28/2013 |
| **Priority:** | Normal | **Due date:** | |
| **Assignee:** | ko1 (Koichi Sasada) | **% Done:** | 0% |
| **Category:** | core | | |
| **Target version:** | current: 2.1.0 | | |

**Description**

> | One day a Rubyist came to Koichi and said, "I understand how to improve
> | CRuby's performance. We must use a generational garbage collector." Koichi
> | patiently told the Rubyist the following story: "One day a Rubyist came
> | to Koichi and said, 'I understand how to improve CRuby's performance..."
> | [This story is an homage of an introduction in a paper:
> | "A real-time garbage collector based on the lifetimes of objects"
> | (by Henry Lieberman, Carl Hewitt)
> | <http://dl.acm.org/citation.cfm?id=358147&CFID=321285546&CFTOKEN=10963356>]

**Issues**

View all issues
Summary
Calendar

**Custom queries**

1.9.1 issues
1.9.2 bugs
1.9.3 issues
2.0.0 accepted features (need action!)
2.0.0 bugs
2.1.0 issues
Bugs unassociated with any target version
matz
showstoppers for 2.0.0-p0
unassigned bugs

# Learn More

**Uniprocessor GC Techniques, Paul Wilson**
https://ritdml.rit.edu/bitstream/handle/1850/5112/PWilsonProceedings1992.pdf

**Rare are GC Talks, nari**
http://furious-waterfall-55.heroku.com/ruby-guide/internals/gc.html

**The Garbage Collection Handbook, Jones, Hosking, Moss**
http://gchandbook.org

**The Ruby Hacker's Guide, Minero Aoki**
http://edwinmeyer.com/Integrated_RHG.html

**Ruby Under a Microscope, Pat Shaughnessy**
http://patshaughnessy.net/ruby-under-a-microscope

**C Programming Language, Brian Kernighan**
http://www.informit.com/store/c-programming-language-9780133086225

Thank you.

@amateurhuman