

# Inside Your Ruby Implementation

Russ Olsen

Why Should  
You Care?

# YARV

Ruby 1.9

# JRuby

Ruby 1.8/1.9

Getting Started...

```
tar -xvf ruby-1.9.1-p378.tar
cd ruby-1.9.1-p378
./configure optflags=
make
# sudo make install
```

ruby-1.9.1-p378/

-> io.c

-> object.c

-> vm.c

-> parse.y

-> ... About 100 .c .h files

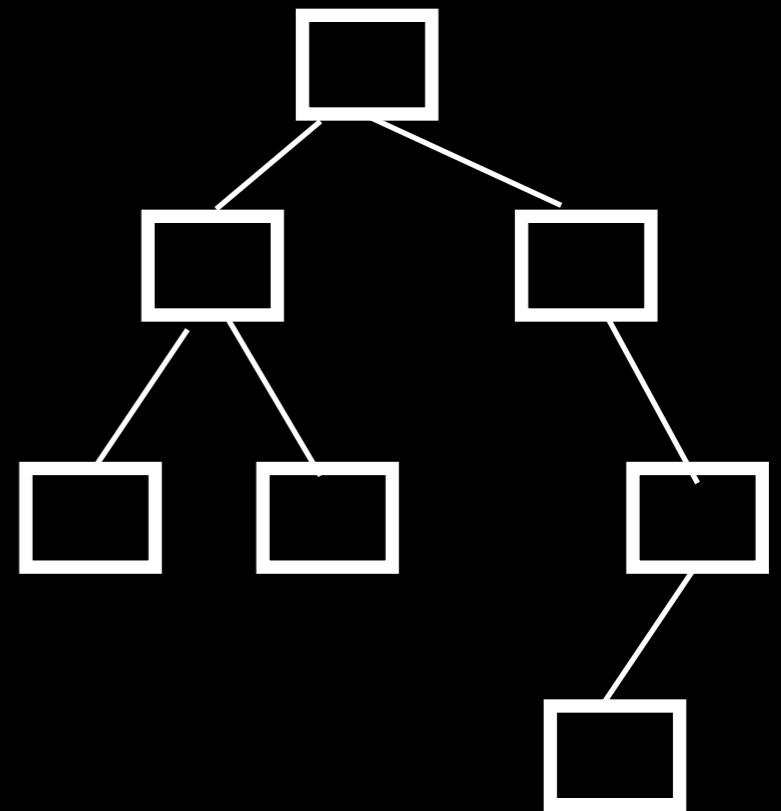
-> include/ruby/\*.h

-> ext/openssl...

# The Big Picture

```
if 1 > 2
  puts 'yes'
else
  puts 'no'
end
```

parse.y

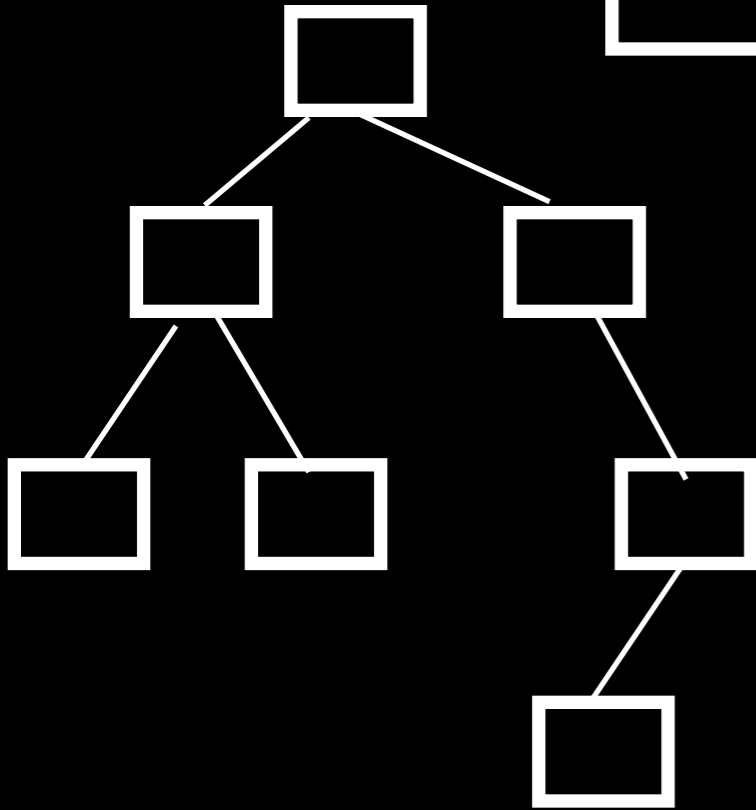


# The Big Picture

compile.c

vm\_exec.c

```
0000 trace          1
0002 putspecialobject 1
0004 putspecialobject 2
0006 putobject :hello
```



# Your Code ...

```
if 1 > 2
  puts "true"
else
  puts "false"
end
```

# Gets Parsed...

```
| k_if expr_value then
  compstmt
  if_tail
  k_end
  {
    ... Generate nodes for if ...
  }
| k_unless expr_value then
  compstmt
  opt_else
  k_end
  {
    ... Generate nodes for unless ...
  }
```

from parse.y

# ... into nodes

```
typedef struct RNode {
    unsigned long flags;
    char *nd_file;
    union {
        struct RNode *node;
        ID id;
        VALUE value;
        VALUE (*cfunc)(ANYARGS);
        ID *tbl;
    } u1;
    union {
        /* Stuff ... */
    } u2;
    union { /* More Stuff ... */ } u3;
} NODE;
```

from node.h

# Each node has a type, and...

```
enum node_type {  
    NODE_METHOD,  
#define NODE_METHOD        NODE_METHOD  
    NODE_FBODY,  
#define NODE_FBODY        NODE_FBODY  
    NODE_CFUNC,  
#define NODE_CFUNC        NODE_CFUNC  
    NODE_SCOPE,  
#define NODE_SCOPE        NODE_SCOPE  
    NODE_BLOCK,  
#define NODE_BLOCK        NODE_BLOCK  
    NODE_IF,  
    ...  
};
```

from node.h

# ... gets turned into bytecodes

`== disasm:`

`<RubyVM::InstructionSequence:<main>@example.rb>=====`

```
0000 trace          1          ( 2)
0002 putobject      1
0004 putobject      2
0006 opt_gt
0007 branchunless  22
0009 trace          1          ( 3)
0011 putnil
0012 putstring      "true"
0014 send           :puts, 1, nil, 8, <ic>
0020 leave          ( 2)
0021 pop
0022 trace          1          ( 5)
0024 putnil
```

# ... which get executed

```
INSN_ENTRY(branchunless) {  
  {  
    OFFSET dst = (OFFSET)GET_OPERAND(1);  
    VALUE val = TOPN(0);  
    DEBUG_ENTER_INSN("branchunless");  
    ADD_PC(1+1);  
    PREFETCH(GET_PC());  
    POPN(1);  
    ... Lot of stuff omitted ...  
    {  
      #line 1158 "insns.def"  
      if (!RTEST(val)) {  
        RUBY_VM_CHECK_INTS();  
        JUMP(dst);  
      } // ...
```

from vm.inc

puts

```
gdb ruby
```

```
(gdb) b rb_io_puts
```

```
(gdb) run -e 'puts "hello"'
```

```
...
```

```
(gdb) p argv[0]
```

```
/*  
 * call-seq:  
 *   ios.puts(obj, ...) => nil  
 * ...  
 */
```

**VALUE**

```
rb_io_puts(int argc, VALUE *argv, VALUE out)  
{  
    ...  
}
```

It's All About  
VALUEs

## Question

A VALUE IS:

- a) A Pointer To A Struct
- b) A Number
- c) A Bit Field

```
typedef unsigned long VALUE;
```

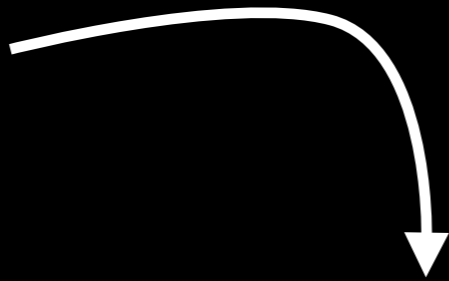
Often A  
VALUE Is A  
\*struct

# VALUE



```
struct RFloat {  
    struct RBasic basic;  
    double float_value;  
};
```

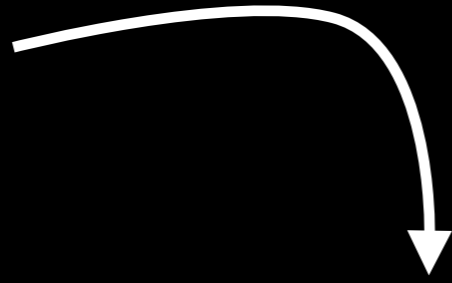
# VALUE



```
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;
};
```

from include/ruby/ruby.h

# VALUE



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

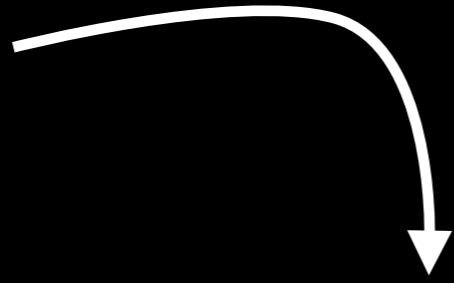
from include/ruby/ruby.h

# VALUE



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

# VALUE



```
struct RClass {  
    struct RBasic basic;  
    rb_classext_t *ptr;  
    struct st_table *m_tbl;  
    struct st_table  
    *iv_index_tbl;  
};
```

from include/ruby/ruby.h

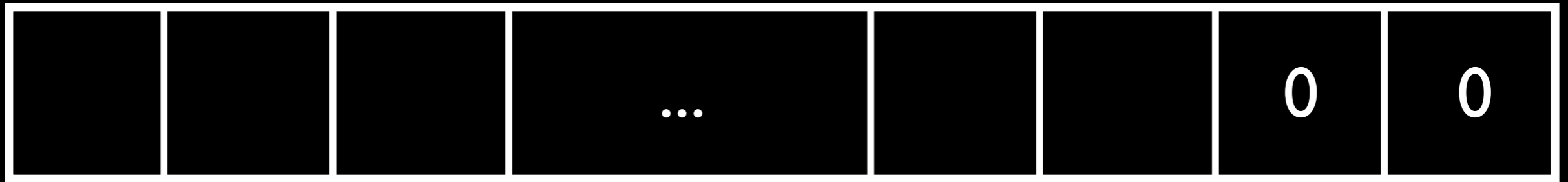
# Structs

## In Action

A Funny Thing  
About  
Struct Pointers

31

0



# Fixnum VALUE

31

0

<< the number >>

1

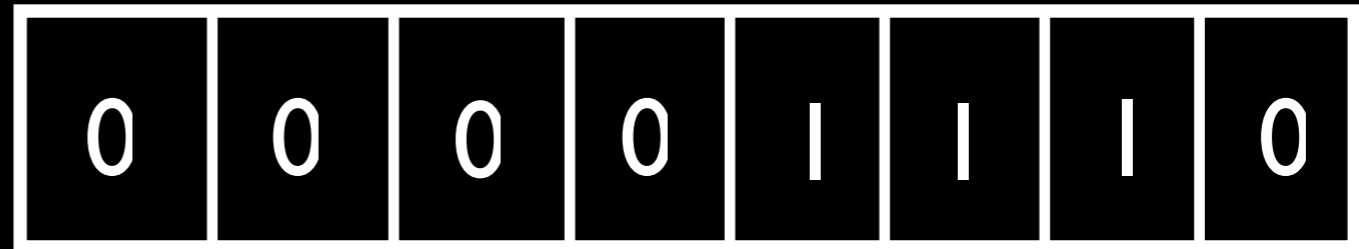
# Symbol VALUE

31

0

<< 24 bit index >>

0x0e



# Special VALUES

RUBY\_Qfalse = 0

RUBY\_Qtrue = 2

RUBY\_Qnil = 4

RUBY\_Qundef = 6

## Question

A VALUE IS:

- a) A Pointer To A Struct
- b) A Number
- c) A Bit Field
- d) All Of The Above!

An Insight

JRuby

```
tar -xvf jruby-src-1.4.0.tar
```

```
cd jruby-1.4.0
```

```
ant
```

Good News:

.project

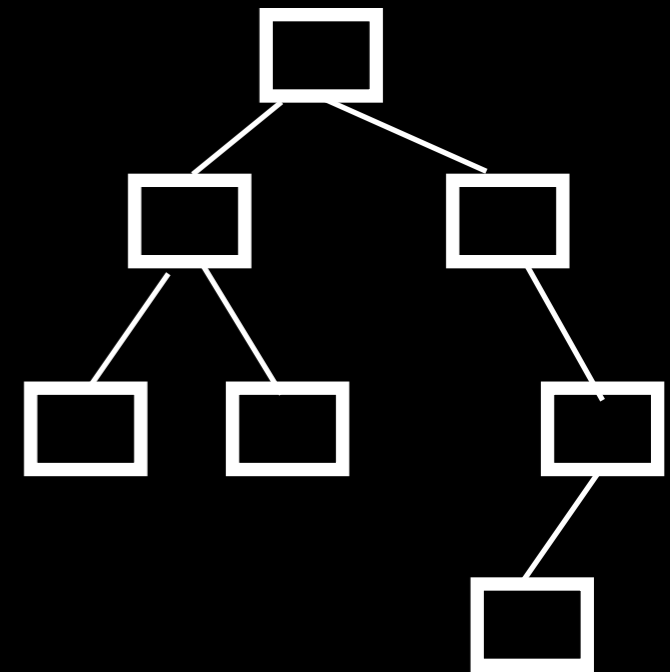
Good News:

nbproject

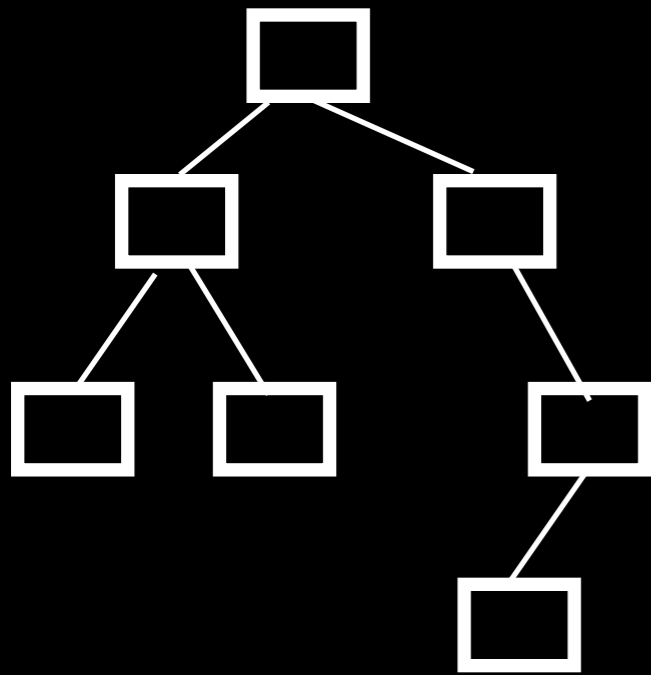
# The Big Picture

```
if 1 > 2
  puts 'yes'
else
  puts 'no'
end
```

DefaultRubyParser.y



# The Big Picture



JITCompiler.java

```
0 iload_1  
1 iload_2  
2 iadd  
3 istore_3
```

# The parser turns Ruby...

```
| kIF expr_value then compstmt if_tail kEND {  
    $$ = new IfNode(getPosition($1),  
        support.getConditionNode($2), $4, $5);  
}  
  
| kWHILE {  
    lexer.getConditionState().begin();  
} expr_value do {  
    lexer.getConditionState().end();  
} compstmt kEND {  
Node body = $6 == null ? NilImplicitNode.NIL : $6;  
$$ = new WhileNode(getPosition($1),  
    support.getConditionNode($3), body);  
}
```

# ... into Java nodes

```
public class IfNode extends Node {  
  
    private final Node condition;  
    private final Node thenBody;  
    private final Node elseBody;  
  
    // ...  
}
```

# ... which are interpreted

```
public class IfNode extends Node {  
  
    private final Node condition;  
    private final Node thenBody;  
    private final Node elseBody;  
  
    // ...  
  
    public IRubyObject interpret(Ruby runtime,  
        ThreadContext context, IRubyObject self,  
        Block aBlock)  
    {  
        // Evaluate condition; if it is true, evaluate  
        // thenBody. If it false, evaluate elseBody.  
    }  
}
```

# ... or compiled

```
public class ASTCompiler {
    // ...

    public void compile(Node node, BodyCompiler context,
        boolean expr) {

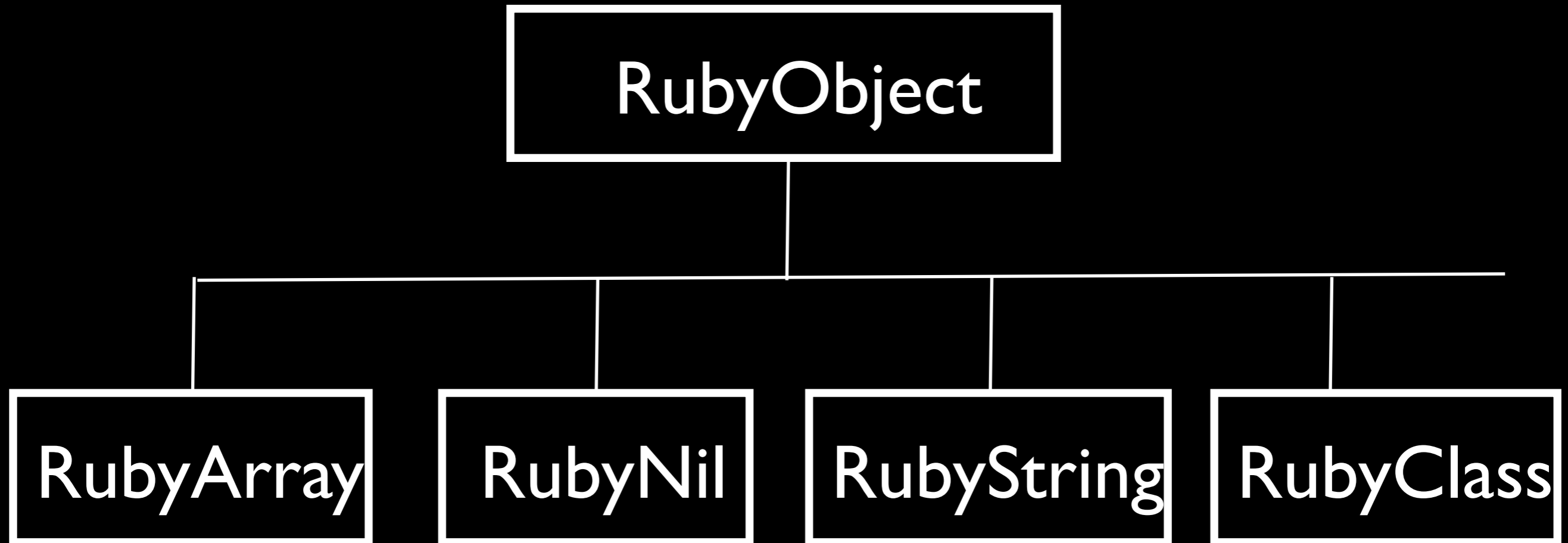
        // ...

        switch (node.getNodeType()) {
            case ALIASNODE:
                compileAlias(node, context, expr);
                break;
            case ANDNODE:
                compileAnd(node, context, expr);
                break;
            // ...
            case IFNODE:
                compileIf(node, context, expr);
                break;
            // ...
        }
    }
}
```

Java

Good!

# Easy Objects



Ruby Is Yours

Ruby Is Yours

Own It!

# Inside Your Ruby Implementation

Russ Olsen