MacRuby and HotCocoa

Dominik Dingel

Worum geht es überhaupt?

- Motivation
- Ruby
- MacRuby
- HotCocoa

Motivation

- One size fits all?
- Syntax mit Zucker

Ruby ist:
Buzzword compliant

- Duck Typing
- Blocks
- OOP
- Mixins

dynamic, functional, reflective, meta programming, interpreted, operator overloading....

Duck Typing

„When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck.“
– James Whitcomb Riley

```ruby
class Ente
  def beschreibung
    "Eine graue Ente"
  end
  def sprechen
    "Quak!"
  end
end

class Kuh
  def beschreibung
    "Eine dicke Kuh"
  end
  def sprechen
    "Muuuh!"
  end
end

def lass_sprechen tier
  puts "#{ tier.beschreibung } macht:
#{ tier.sprechen }"
end

lass_sprechen Ente.new
lass_sprechen Kuh.new
```

Blocks in objC

```
#include <Block.h>
#include <stdio.h>
typedef int (^IntBlock)();

IntBlock CreateCounter(int start, int increment) {
  __block int c = start;
  return Block_copy(^{
    int result = c;
    c += increment;
    return result;
  });
}

int main(int argc, char *argv[]) {
  IntBlock counter = CreateCounter(7, 2);
  printf("1st: %d\n", counter());
  printf("2nd: %d\n", counter());
  Block_release(counter);
  return 0;
}
```

Blocks in Ruby

```ruby
def nTimes(aThing)
  return proc { |n| aThing * n }
end
p1 = nTimes(23)

puts p1.call(3)
puts p1.call(4)

p2 = nTimes("Hello ")
puts p2.call(3)
```

```
69
92
Hello Hello Hello
```

Ruby ist OOP

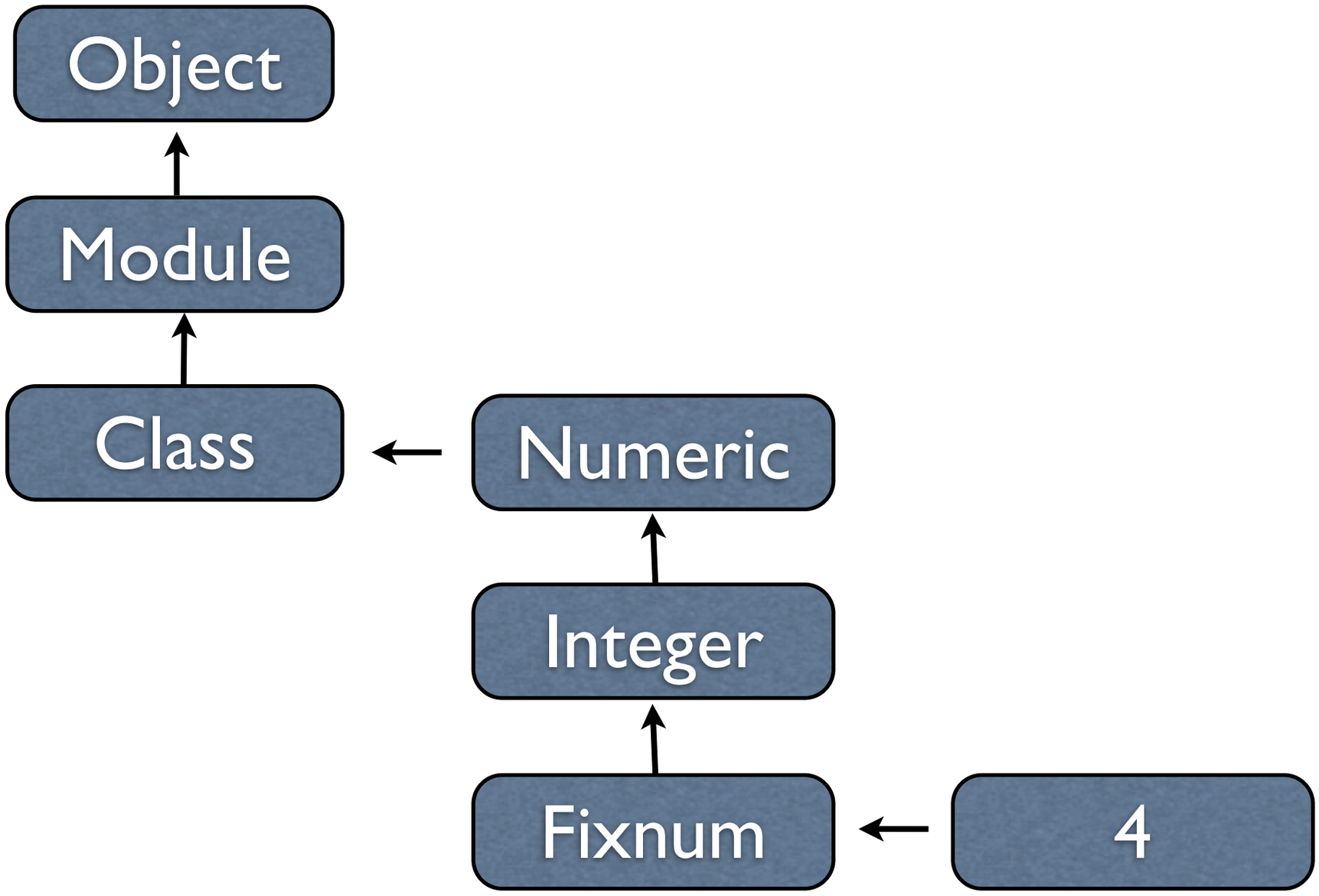
```
3.times{ puts "hallo" }
```

hallo
hallo
hallo

```
>> 4.class
=> Fixnum
>> 4.class.superclass
=> Integer
>> 4.class.superclass.superclass
=> Numeric
>> 4.class.superclass.superclass.superclass
=> Object
>> 4.class.superclass.superclass.superclass.superclass => nil
```

```
>> 4.class.class
=> Class
>> 4.class.class.superclass
=> Module
>> 4.class.class.superclass.superclass => Object
```

Ruby ist OOP



Mixins

Ruby besitzt keine Mehrfachverbung

```ruby
module Gaspedal
  def beschleunigen
    puts "brumm brumm brumm"
  end
end
```

```ruby
module Bremse
  def bremsen
    puts "quietsch"
  end
end
```

```ruby
class Auto
  include Gaspedal
end

class GutesAuto < Auto
  include Bremse
end
```

Mixins

- Interfaces mit Funktionalität
- einfaches Verändern von Klassen zur Laufzeit

```ruby
module BarModule
  def hello_world
    puts "Hello World"
  end
end
```

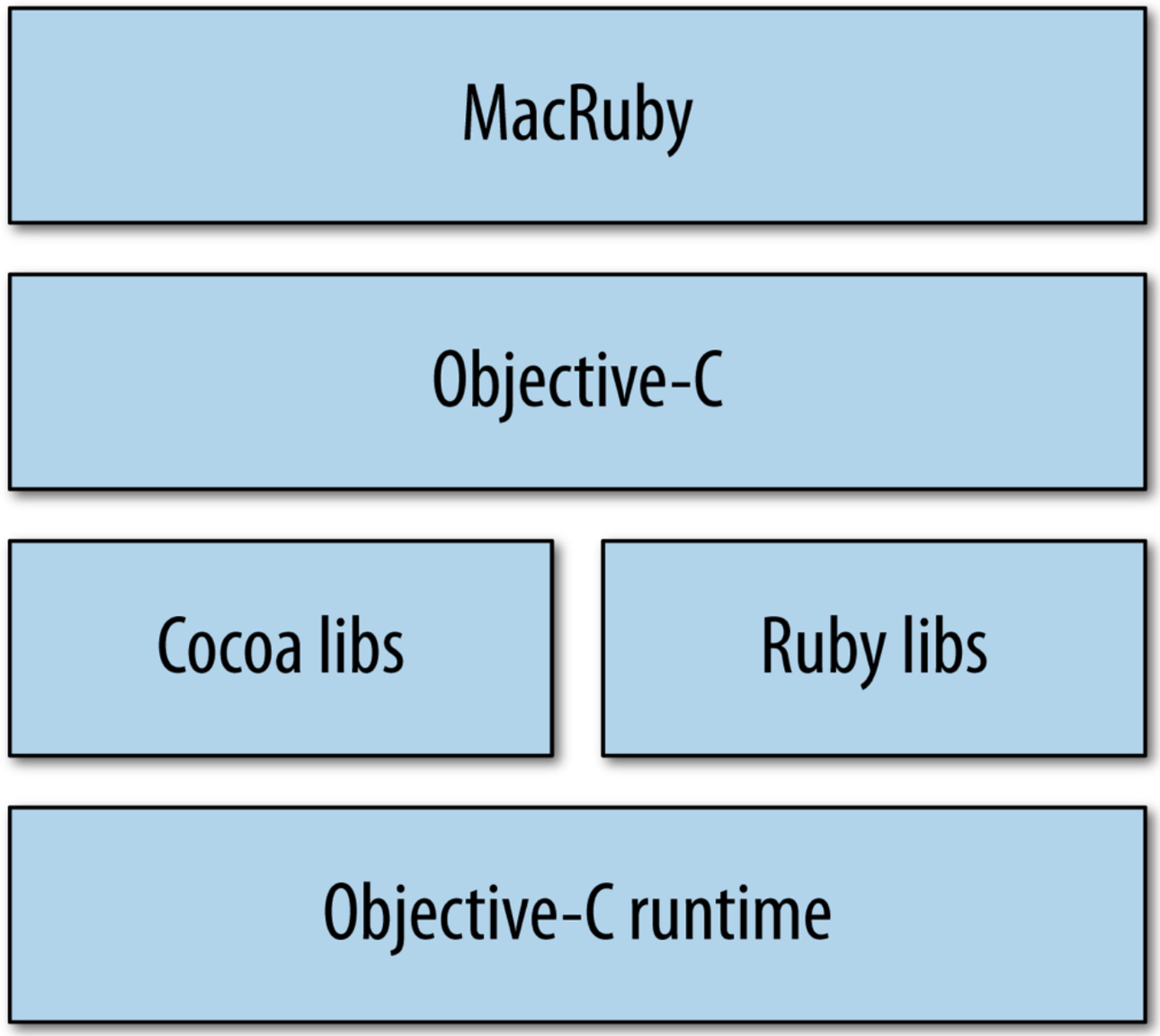
```ruby
String.send(:include, BarModule)
s = "Arbitrary String"
s.hello_world
```

Brücken bei Apple

- PyObjC
- Java
- RubyCocoa*

MacRuby

MacRuby


MacRuby
Objective-C
Cocoa libs
Ruby libs
Objective-C runtime

Ruby ist OOP

```
3.times{ puts "hallo" }
```

hallo
hallo
hallo

```
>> 4.class
=> Fixnum
>> 4.class.superclass
=> Integer
>> 4.class.superclass.superclass
=> Numeric
>> 4.class.superclass.superclass.superclass
=> NSNumber
>> 4.class.superclass.superclass.superclass.superclass
=> NSValue
...
```

```
>> 4.class.class
=> Class
>> 4.class.class.superclass
=> Module

>> 4.class.class.superclass.superclass => NSObject
```

MacRuby

- JIT
  - Ruby AST zu LLVM IR
  - LLVM zu Machine Code
  - Machine Code zu CPU
- AOT
  - Ruby AST zu LLVM IR
  - LLVM zu Machine Code
  - Machine Code zu Mach-O

MacRuby

- Nach der Installation verschiedene Tools: (macrake, macgem, macruby, macirb, **macrubyc**)
- möglich mit XCode Ruby Projekte zu entwickeln*
- Zugriff auf Frameworks (Foundation, AppKit, Core Data, ...)
- Libdispatch (GCD) nutzbar*

MacRuby..
in dem Sandkasten

MacRuby

MacRuby..
in dem Sandkasten

```
>> require "open-uri"
=> true
>>
begin
   open("http://www.google.de")
   rescue SystemCallError => exception
       puts exception
end

=> #<File:/var/folders/w6/zfmy00_n7m5g3g9wh5yl728h0000gn/T/open-
uri20111220-3834-5jgznl>
```

MacRuby..
in dem Sandkasten

```
>> require "open-uri"
=> true
>>
begin
   Sandbox.no_internet.apply!
   open("http://www.google.de")
   rescue SystemCallError => exception
       puts exception
end

Operation not permitted - connect(2)
=> nil
```

MacRuby..
in dem Sandkasten

- Einschränkung des Programms
- 5 Profile vorgefertigt (Internet, Daten..)

MacRuby

HotCocoa

Application

```
./hotcocoa siegen
cd siegen
macrake
```

```
total 16
-rw-r--r--  1 XXX  staff   127 20 Dez 17:53 Rakefile
drwxr-xr-x  3 XXX  staff   102 20 Dez 17:53 Siegen.app
-rw-r--r--  1 XXX  staff  2290 20 Dez 17:53 Siegen.appspec
drwxr-xr-x  4 XXX  staff   136 20 Dez 17:53 lib
drwxr-xr-x  3 XXX  staff   102 20 Dez 17:53 resources
```

HotCocoa

Library

```
win = NSWindow.alloc.initWithContentRect [10,20,300,300],
  :styleMask => (NSTitledWindowMask          |
                 NSClosableWindowMask        |
                 NSMiniaturizableWindowMask  |
                 NSResizableWindowMask)
```

MacRuby

HotCocoa

```
win = window :frame => [10,20,300,300]
```

HotCocoa

```
framework 'Cocoa'
app = NSApplication.sharedApplication
win = NSWindow.alloc.initWithContentRect([0,0,200,60],
  styleMask:NSTitledWindowMask | NSClosableWindowMask | NSMiniaturizableWindowMask |
NSResizableWindowMask,
  backing:NSBackingStoreBuffered, defer:false )
win.title = 'Hello World'
button = NSButton.alloc.initWithFrame(NSZeroRect)
win.contentView.addSubview(button)
button.bezelStyle = NSRoundedBezelStyle
button.title = 'Hello!'
button.sizeToFit
button.frameOrigin = NSMakePoint((win.contentView.frameSize.width / 2.0) -
(button.frameSize.width / 2.0),
                                 (win.contentView.frameSieze.height / 2.0) -
(button.frameSize.height / 2.0))
button_controller = Object.new

def button_controller.sayHello(sender)
  puts "Hello World!"
end
button.target = button_controller
button.action = 'sayHello:'

win.display
win.orderFrontRegardless

app.run
```

HotCocoa

```
require 'rubygems'
require 'hotcocoa'
include HotCocoa
application do
  win = window :title => 'hello world', :frame => [0,0,200,60]

  b = button :title => 'Hello!', :layout => {:align => :center}
  win << b

  b.on_action { puts 'Hello World!' }
end
```

..und wozu?

- Entwickeln von Mac OS X Apps
  - geringere Einstiegshürde
- Programme dürfen in Mac App Store
- bisher nicht für iOS

Ende