RubyMotion

---

a2rb / 2012.12.11

Andrew Sardone / @andrewa2

Foundation

Objective-C

- Strict superset of C
- Adds Smalltalk message passing OOP to C

Smalltalk

```
myObject say: 'Hello, world!'
```

Objective-C

```
[myObject say: @"Hello, world!"];
```

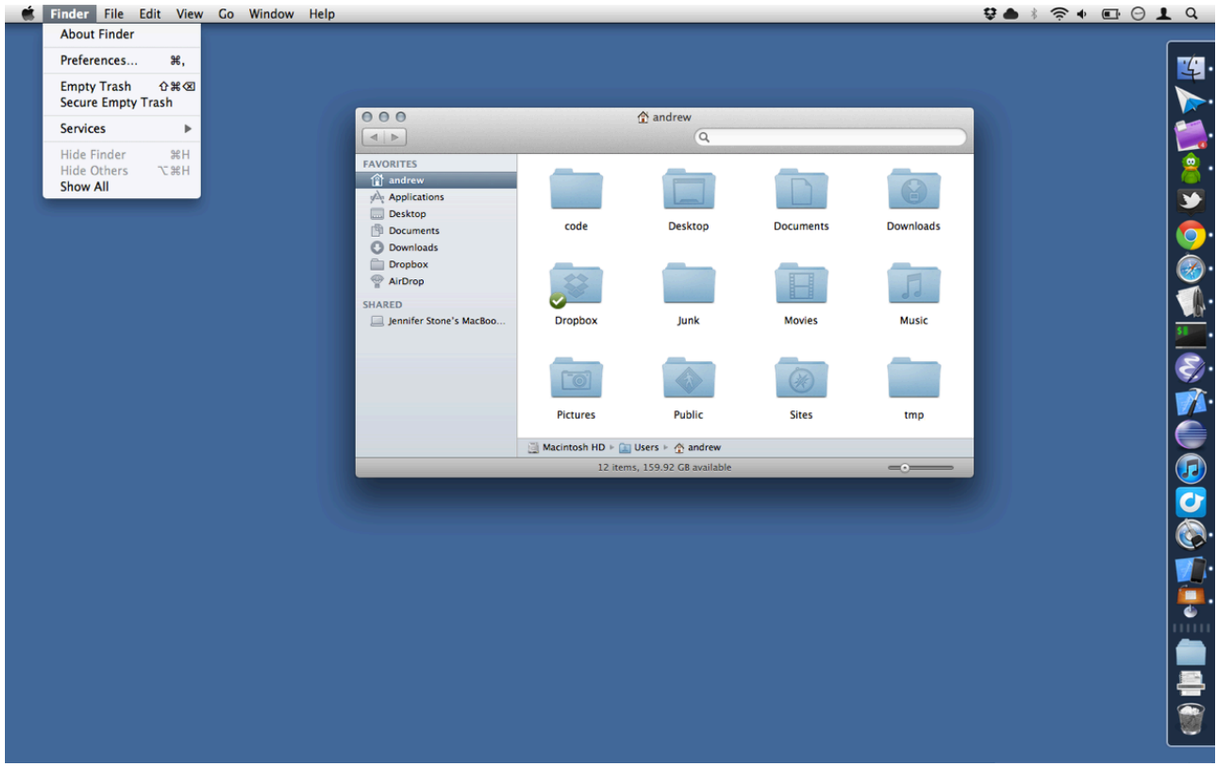
Ruby

- "A dynamic, open source programming language with a focus on simplicity and productivity. "
- Flexible, expressive syntax
- Smalltalk meets Perl

```
myObject.say "Hello, world!"
```


Workspace
Info
File
Edit
Disk
View
Tools
Windows
Services
Hide
Log Out
Disk
Eject
Initialize
Check for Disks
File Viewer
paul
tmp
LocalApps
ibmaaa.aff
home
paul
Adv.NetInfo.rtf
Attributes Inspector
Attributes
Adv.NetInfo.rtf
Path: /home/paul
Size: 30.2KB
Owner: paul
Group: wheel
Permissions
Read
Write
Owner Group Other
Changed
Revert
OK
Mailboxes
Name: OmniWeb.mbox
Delete
New
Open
Transfer
Spy

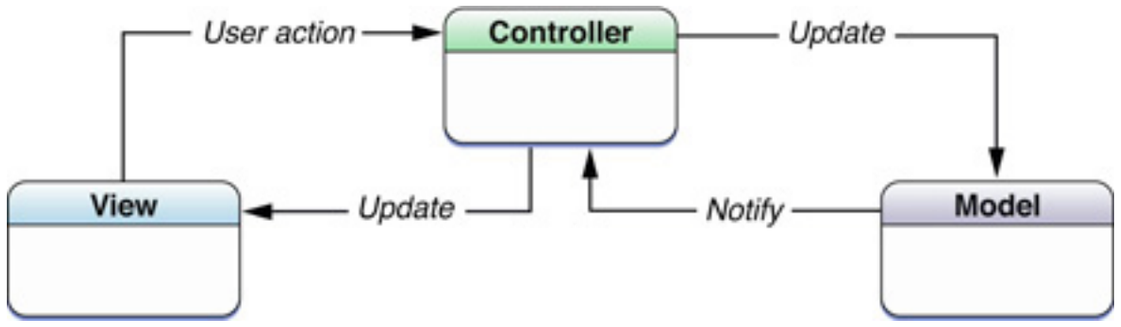
NeXTSTEP



NeXTSTEP ⟶ Mac OS X

OpenStep (APIs) ⟶ Cocoa

OpenStep / Cocoa	Cocoa Touch / iOS SDK
Foundation Kit	Foundation Kit
AppKit	UIKit
…	…



Cocoa Touch

Media

Core Services

Core OS

via developer.apple.com

MacRuby

by Laurent Sansonetti

An open source Ruby 1.9 implemented directly on top of Apple's Objective-C runtime, LLVM compiler infrastructure, and OS X Foundation frameworks.

RubyMotion

by Laurent Sansonetti

A commercial toolchain for iOS development

Similar to MacRuby, it's Ruby implemented on top of the Objective-C runtime

RubyMotion

Objective-C

Objective-C Runtime

iOS SDK

Foundation Kit

UIKit

…

Ruby Objects
on
Objective-C Runtime

```
dict = {}
# {}

dict.class
# Hash

dict.superclass
# NSMutableDictionary

dict.addValue("world", forKey: "hello")
# { "hello" => "world" }

dict["foo"] = "bar"
# { "hello" => "world", "foo" => "bar" }
```

```
str = "Ruby Motion"
# "Ruby Motion"

str.class.ancestors
# [String, NSMutableString, NSString,
    Comparable, NSObject, Kernel]

str.split.insertObject("Try", atIndex:0)
# ["Try", "Ruby", "Motion"]
```

Toolchain

- CLI-based workflow via `rake`
- Statically compiles Ruby code to LLVM byte code
- Automatic memory management, similar to Automatic Reference Counting (ARC) – not Ruby's normal garbage collection

Statically Compiled

```
class HelloView < UIView
  def drawRect(rect)
  end
end
```

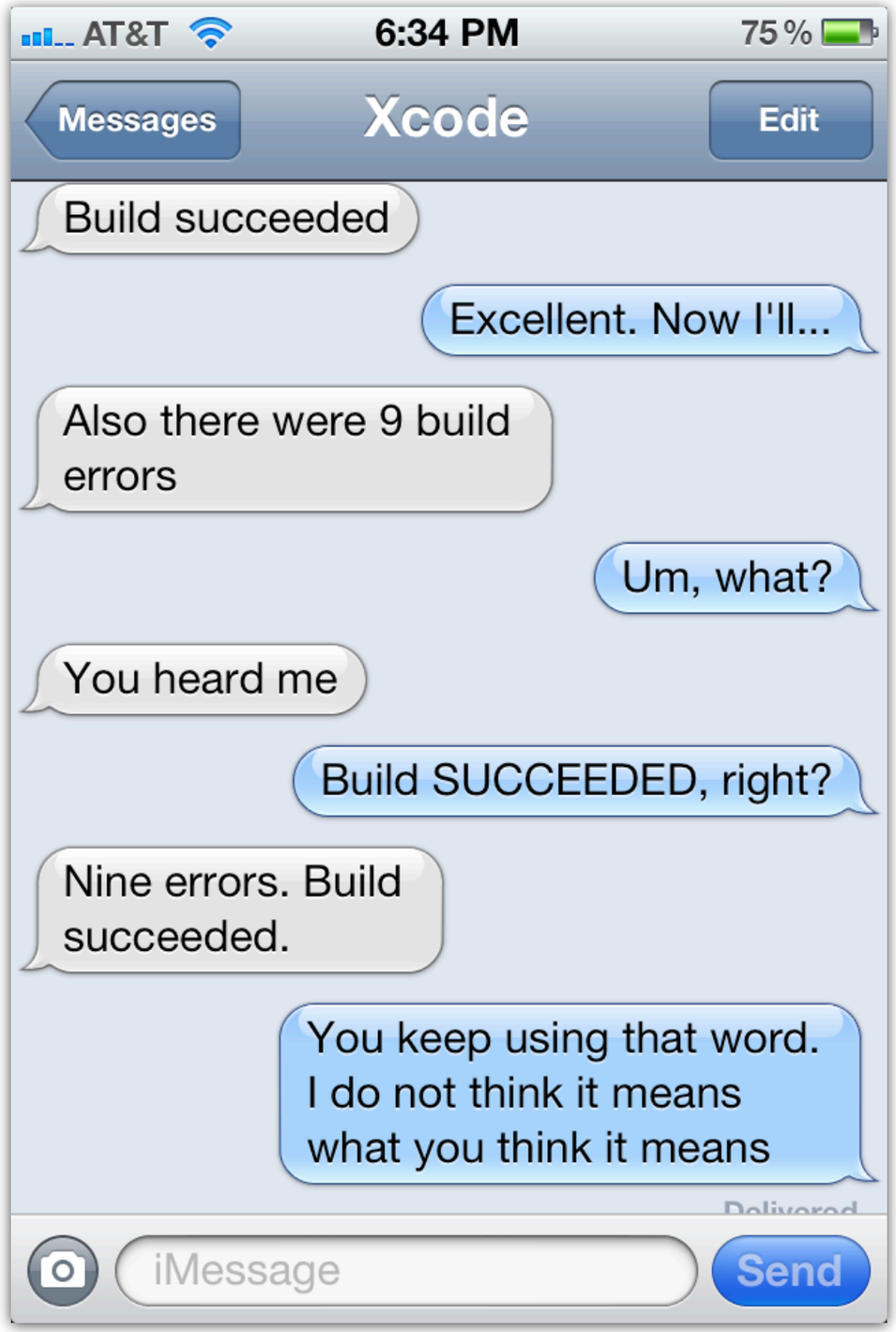
```
[:class, :HelloView,
  [:const, :UIView],
  [:defn,
    :drawRect,
    ...]]
```

```
define internal i32 @"rb_scope__drawRect:__"(i32
%self, i8* nocapture %sel, i32 %rect) {
MainBlock:
  %right_addr.i42 = alloca i32, align 4
  %right_addr.i30 = alloca i32, align 4
  %right_addr.i = alloca i32, align 4
  %argv29 = alloca [4 x i32]
  %argv29.sub = getelementptr inbounds [4 x i32]*
%argv29, i32 0, i32 0
  %0 = load i32* @5
  %1 = load i8** @6
  %2 = call fastcc i32 @vm_ivar_get(i32 %self, i32
%0, i8* %1)
  switch i32 %2, label %then [
    i32 0, label %else
    i32 4, label %else
```

Ruby ⟶ AST ⟶ LLVM IR ⟶ Assembly…

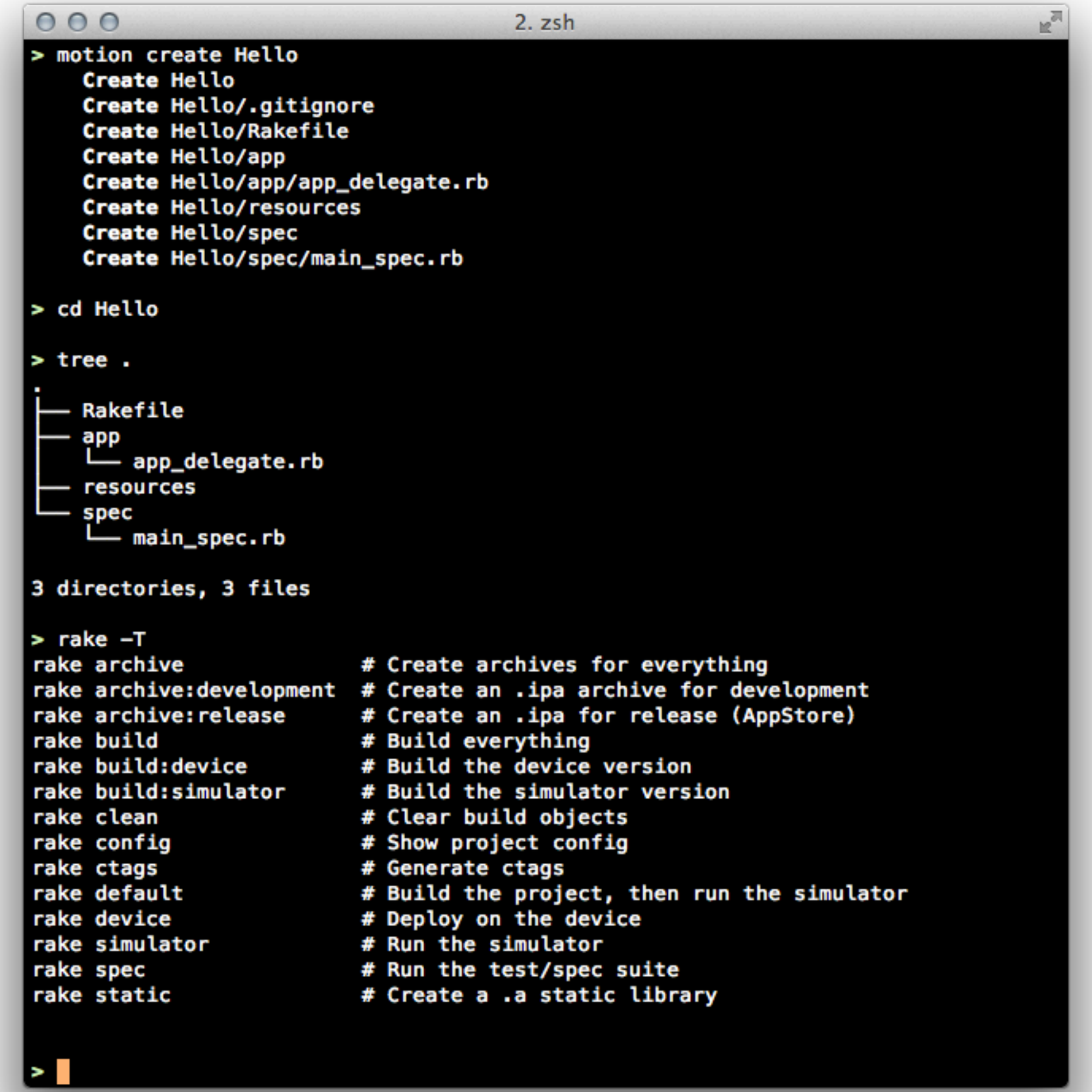
via Rich Kilmer's *RubyConf 2012 Presentation*

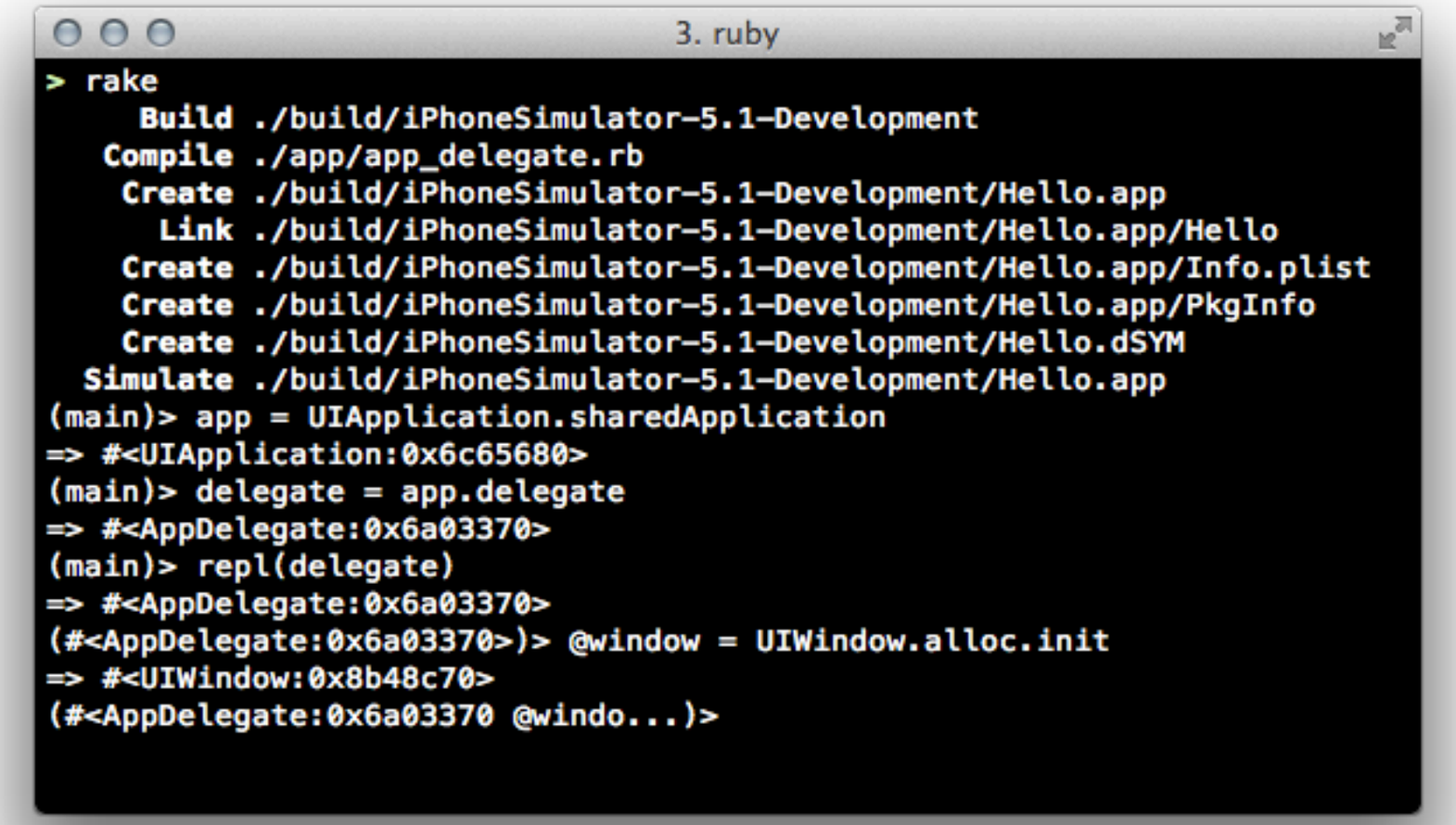
Advantages?



via textfromxcode.com

No Xcode


2. zsh
> motion create Hello
Create Hello
Create Hello/.gitignore
Create Hello/Rakefile
Create Hello/app
Create Hello/app/app_delegate.rb
Create Hello/resources
Create Hello/spec
Create Hello/spec/main_spec.rb
> cd Hello
> tree .
.
├── Rakefile
├── app
│ └── app_delegate.rb
├── resources
└── spec
└── main_spec.rb
3 directories, 3 files
> rake -T
rake archive # Create archives for everything
rake archive:development # Create an .ipa archive for development
rake archive:release # Create an .ipa for release (AppStore)
rake build # Build everything
rake build:device # Build the device version
rake build:simulator # Build the simulator version
rake clean # Clear build objects
rake config # Show project config
rake ctags # Generate ctags
rake default # Build the project, then run the simulator
rake device # Deploy on the device
rake simulator # Run the simulator
rake spec # Run the test/spec suite
rake static # Create a .a static library
>


3. ruby
> rake
Build ./build/iPhoneSimulator-5.1-Development
Compile ./app/app_delegate.rb
Create ./build/iPhoneSimulator-5.1-Development/Hello.app
Link ./build/iPhoneSimulator-5.1-Development/Hello.app/Hello
Create ./build/iPhoneSimulator-5.1-Development/Hello.app/Info.plist
Create ./build/iPhoneSimulator-5.1-Development/Hello.app/PkgInfo
Create ./build/iPhoneSimulator-5.1-Development/Hello.dSYM
Simulate ./build/iPhoneSimulator-5.1-Development/Hello.app
(main)> app = UIApplication.sharedApplication
=> #<UIApplication:0x6c65680>
(main)> delegate = app.delegate
=> #<AppDelegate:0x6a03370>
(main)> repl(delegate)
=> #<AppDelegate:0x6a03370>
(#<AppDelegate:0x6a03370>)> @window = UIWindow.alloc.init
=> #<UIWindow:0x8b48c70>
(#<AppDelegate:0x6a03370 @windo...)>

REPL

Ruby

```objc
NSArray *a = @[ @"Foo", @"Bar" ];

[a enumerateObjectsUsingBlock:^(id obj, NSUInteger idx, BOOL *stop) {
    NSLog(@"%@", obj);
}];

# vs.

["Foo", "Bar"].each_with_index { |obj, i| puts obj }
```

Ruby-fication

```objc
// UIKit in Objective-c
[button addTarget:self
           action:@selector(buttonTapped:)
 forControlEvents:UIControlEventTouchUpInside];

// Elsewhere
- (void)buttonTapped:(id)sender {
    self.backgroundColor = [UIColor redColor];
}
```

```ruby
# vs.

# BubbleWrap UIControl / UIButton helpers
button.when(UIControlEventTouchUpInside) do
  self.backgroundColor = UIColor.redColor
end
```

via http://clayallsopp.com/posts/the-ruby-motion-way/

```
describe "The Timer view controller" do
  tests TimerController

  it "has a timer label" do
    view('Tap to start').should.not == nil
  end

  it "starts a timer" do
    tap 'Start'
    controller.timer.isValid.should == true
  end

  it "increases the timer label value" do
    label = view('Tap to start')
    label.text.to_f.should == 0

    tap 'Start'
    proper_wait 1
    tap 'Stop'
    label.text.to_f.should > 1
    label.text.to_f.should < 2
  end
end
```

Testing

Disadvantages?

- Another toolchain dependency
- Core product is closed-source
- Some tools still missing (lint / better static analysis)

Demo

“I'm trying to convince Objective-C developers to look into Ruby and Ruby developers to look into Objective-C”

– Matt Aimonetti

NSBrief #72

Additional Resources

- RubyMotion Developer Center
  rubymotion.com/developer-center/

- RubyMotion Consoler – in-browser REPL
  pieceable.com/rubymotion-console

- RubyMotion Tutorial
  rubymotion-tutorial.com

- BubbleWrap – Cocoa wrappers/helpers
  bubblewrap.io

- pinboard.in/u:andrewsardone/t:rubymotion