

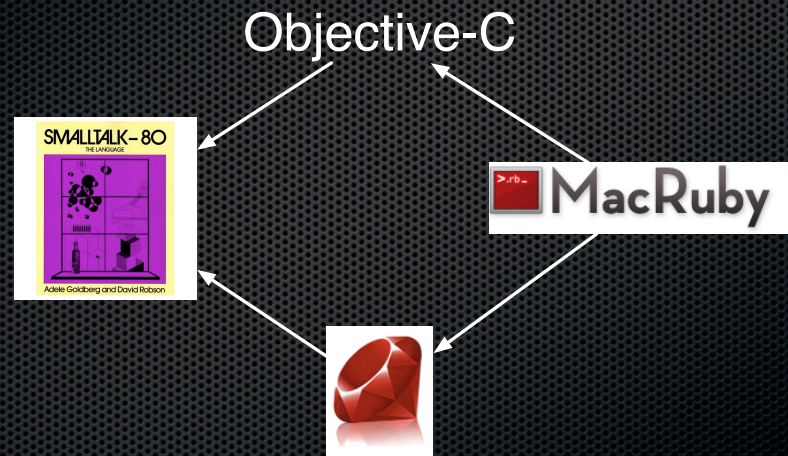


MacRuby 0.5 for OS X

Power to the Cats

MacRuby 0.5 – A Ruby specifically for and to take advantage of the Mac OS X Platform features.

MacRuby's Origins



Current Ruby on Mac

- 1.8.7
- Standard interpreter
- Bridge library called RubyCocoa for native apps
- Issues
 - Duplicate class trees in each runtime space
 - Seperate GC cycles
 - Type/object conversion across bridge

MacRuby

- Fully reimplemented on top of Obj-C runtime
- No YARV
- LLVM instead
 - Common compiler for Snow Leopard
- No bridging
 - All Ruby classes implemented on top of Obj-C
- Threaded GC

4 compilers available really but all headed towards LLVM core.

Ruby classes are Obj-C classes, Ruby methods are Obj-C methods, etc.


MacRuby

- Full support of all frameworks
 - (planned)
- Works with Grand Central Dispatch

4 compilers available really but all headed towards LLVM core.

Ruby classes are Obj-C classes, Ruby methods are Obj-C methods, etc.

MacRuby Site - www.macruby.org

 **MacRuby**

Current Version 0.4

HOME BLOG SOURCE DOCUMENTATION CONTACT MacRuby in 2 Easy Steps:

MacRuby is a version of Ruby 1.9, ported to run directly on top of Mac OS X core technologies such as the Objective-C common runtime and garbage collector, and the CoreFoundation framework. While still a work in progress, it is the goal of MacRuby to enable the creation of full-fledged Mac OS X applications which do not sacrifice performance in order to enjoy the benefits of using Ruby. [Read more...](#)

MACRUBY BLOG »

MacRuby 0.5 beta 2
2009-11-17 »

After a month of hard work we are pleased to offer the second beta of MacRuby 0.5. We are expecting one more beta before shipping the final 0.5.
[Read more...](#)


MacRuby 0.5 beta 1

HotCocoa Is For Me!

If you've done any amount of programming on OS X, you know that the API can be quite verbose. HotCocoa simplifies this down to very elegant and simple methods that then return super sexy UI elements. [Read more...](#)

```
require 'hotcocoa'
include HotCocoa
application do |app|
  win = window :size => [100,50]
```

DOWNLOAD MACRUBY:

 ZIP
or [checkout the source](#)

1.

GET STARTED!
Check out the [tutorial](#), [resources](#) and [examples](#) that are available for MacRuby.

2.

MACRUBY EVENTS »

19-21 Nov 2009 » [RubyConf](#)
San Francisco, CA
Laurent Sansonetti presents MacRuby

Gems

- Macgems is the command
- Still early in lifecycle so some gems work some don't
- Gems are currently put in:
 - /Library/Frameworks/MacRuby.framework/Versions/0.5/usr/lib/ruby/Gems/1.9.0/gems

```
$ macgem sources -a http://gemcutter.org  
$ sudo macgem install textorize-mr  
$ textorize -f"Papyrus" -s200 "MacRuby 0.5b1"  
$ open output.png
```

Gems are stored in:

```
/Library/Frameworks/MacRuby.framework/Versions/0.5/usr/lib/ruby/Gems/1.9.0/gems
```

Core

- Available in:
 - /Library/Frameworks/MacRuby.framework/Versions/0.5/usr/lib/ruby/1.9.0

Macirb

- `[~]$ macirb`
- `irb(main):001:0> "foo".class`
- `=> NSMutableString`

Macirb - cont 2

- `irb(main):002:0> Object.methods.sort`
- `=> [! , !=, !~ , ~, <, <=, <=>, ==, ===, =~, >, >=, :Complex, :Rational, :URI, :__callee__, :__id__, :__meta__?, :__method__, :__native__?, :__send__, :allocate, :ancestors, :autoload, :autoload?, :class_eval, :class_exec, :class_variable_defined?, :class_variable_get, :class_variable_set, :class_variables, :clone, :const_defined?, :const_get, :const_missing, :const_set, :constants, :define_singleton_method, :dup, :enum_for, :eql?, :equal?, :extend, :freeze, :frozen?, :gem, :gem_original_require, :hash, :include?, :included_modules, :inspect, :instance_eval, :instance_exec, :instance_method, :instance_methods, :instance_of?, :instance_variable_defined?, :instance_variable_get, :instance_variable_set, :instance_variables, :is_a?, :kind_of?, :load_bridge_support_file, :method, :method_defined?, :methods, :module_eval, :module_exec, :name, :new, :nil?, :object_id, :private_class_method, :private_instance_methods, :private_method_defined?, :private_methods, :protected_instance_methods, :protected_method_defined?, :protected_methods, :public_class_method, :public_instance_method, :public_instance_methods, :public_method, :public_method_defined?, :public_methods, :public_send, :remove_class_variable, :require, :respond_to?, :send, :singleton_methods, :taint, :tainted?, :tap, :to_enum, :to_s, :to_yaml, :to_yaml_properties, :to_yaml_style, :trust, :untaint, :untrust, :untrusted?, :y, :yaml_as, :yaml_new]`

Macirb - cont 3

- `irb(main):003:0> Object.ancestors`
- `=> [NSObject, Kernel]`
- Hmm - That's different?

XCode

- MacRuby projects just like any XCode project
- Lets take a look

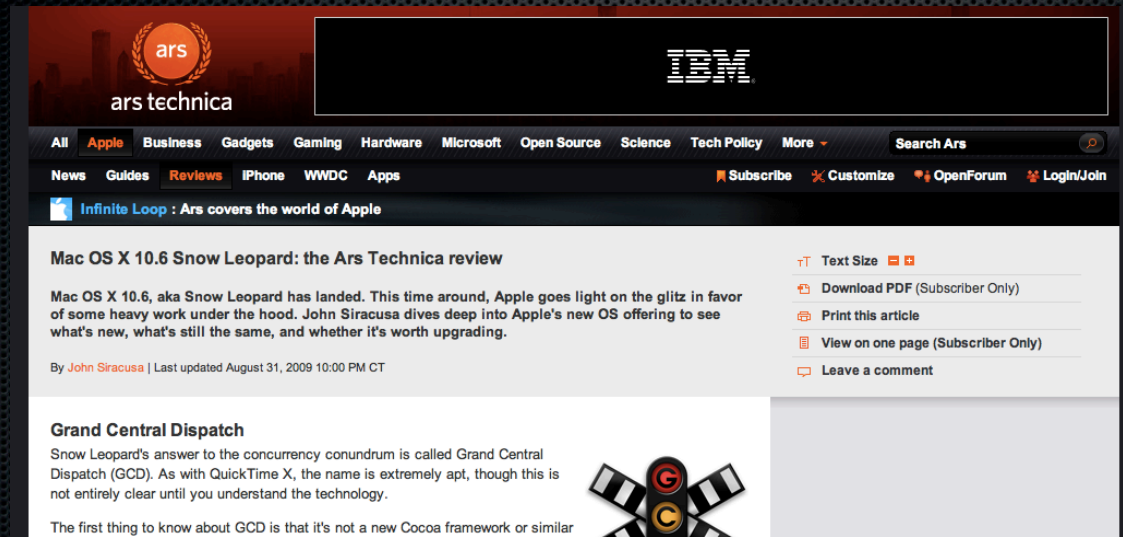
```
Add Dog class
class Dog
  def name
    @name
  end

  def setName(name)
    @name = name
  end
end
```

In main function put this. Then build and show console.

```
d = Dog.new
d.setName("Bob")
pp d
```


Grand Central Dispatch



```
Add Dog class
class Dog
def name
  @name
end

def setName(name)
  @name = name
end
end
```

In main function put this. Then build and show console.

```
d = Dog.new
d.setName("Bob")
pp d
```

GCD Sample Code

```
waiting_chairs = Dispatch::Queue.new('com.apple.waiting_chairs')
semaphore = Dispatch::Semaphore.new(3)
index = -1
while true
  index += 1
  success = semaphore.wait(Dispatch::TIME_NOW)
  if success != 0
    puts "Customer turned away #{index}"
  next
end
waiting_chairs.dispatch do
  semaphore.signal
  puts "Shave and a haircut #{index}"
end
```

Paste this into demo:

```
waiting_chairs = Dispatch::Queue.new('com.apple.waiting_chairs')
semaphore = Dispatch::Semaphore.new(3)
index = -1
while true
  index += 1
  success = semaphore.wait(Dispatch::TIME_NOW)
  if success != 0
    puts "Customer turned away #{index}"
  next
end
waiting_chairs.dispatch do
  semaphore.signal
  puts "Shave and a haircut #{index}"
end
end
```


Interface Builder

- Instance based GUI building
- Demo

Instruments

- Lets you perf-mon your MacRuby apps
 - Monitor memory, threads, # of cores used, etc.

Hot Cocoa

- DSL for building Cocoa GUI's if you don't want to do instance based building of your GUI's

Hot Cocoa Simplest App

```
require 'hotcocoa'

class Application
  include HotCocoa

  def start
    application(:name => "Postie") do |app|
      app.delegate = self
      window(:frame => [100, 100, 500, 500], :title => "Postie") do |win|
        win << label(:text => "Hello from HotCocoa", :layout => {:start => false})
        win.will_close { exit }
      end
    end
  end
end
```