

MacRuby





Jonathan Penn

@jonathanpenn Ohio

Thoughtful. Cautious. Bold. Stubborn. Good looking.

<http://cocoamanifest.net>



Jonathan Penn

@jonathanpenn Ohio

Thoughtful. Cautious. Bold. Stubborn. Good looking.

<http://cocoamanifest.net>



Navel Labs



Jonathan Penn

@jonathanpenn Ohio

Thoughtful. Cautious. Bold. Stubborn. Good looking.

<http://cocoamanifest.net>



Navel Labs

cocoamanifest.net

What is Ruby?



Yukihiro Matsumoto

Papa Perl,
Mama Smalltalk

Papa Perl,
Mama Smalltalk
(and a lil' Lisp)

Old

Is ***NOT*** Rails

Dynamically Typed

```
user = User.new
```

Dynamic Class Definitions

```
class User
  def first_name
    # ...
  end

  def last_name
    # ...
  end
end
```

Dynamic Class Definitions

```
# elsewhere in the code
```

```
class User
  def full_name
    [first_name, last_name].join(" ")
  end
end
```

Message Passing

```
user.send :first_name
```

```
# equivalent to
```

```
user.first_name
```

Syntax

```
puts("hello world")  
puts "hello world"
```

```
Dir["*.html"].each do |name|  
  puts name  
end
```

```
users.map { |u| u.first_name }.join(", ")
```


Plenty of opportunity to shoot
yourself in the foot

Learn Ruby

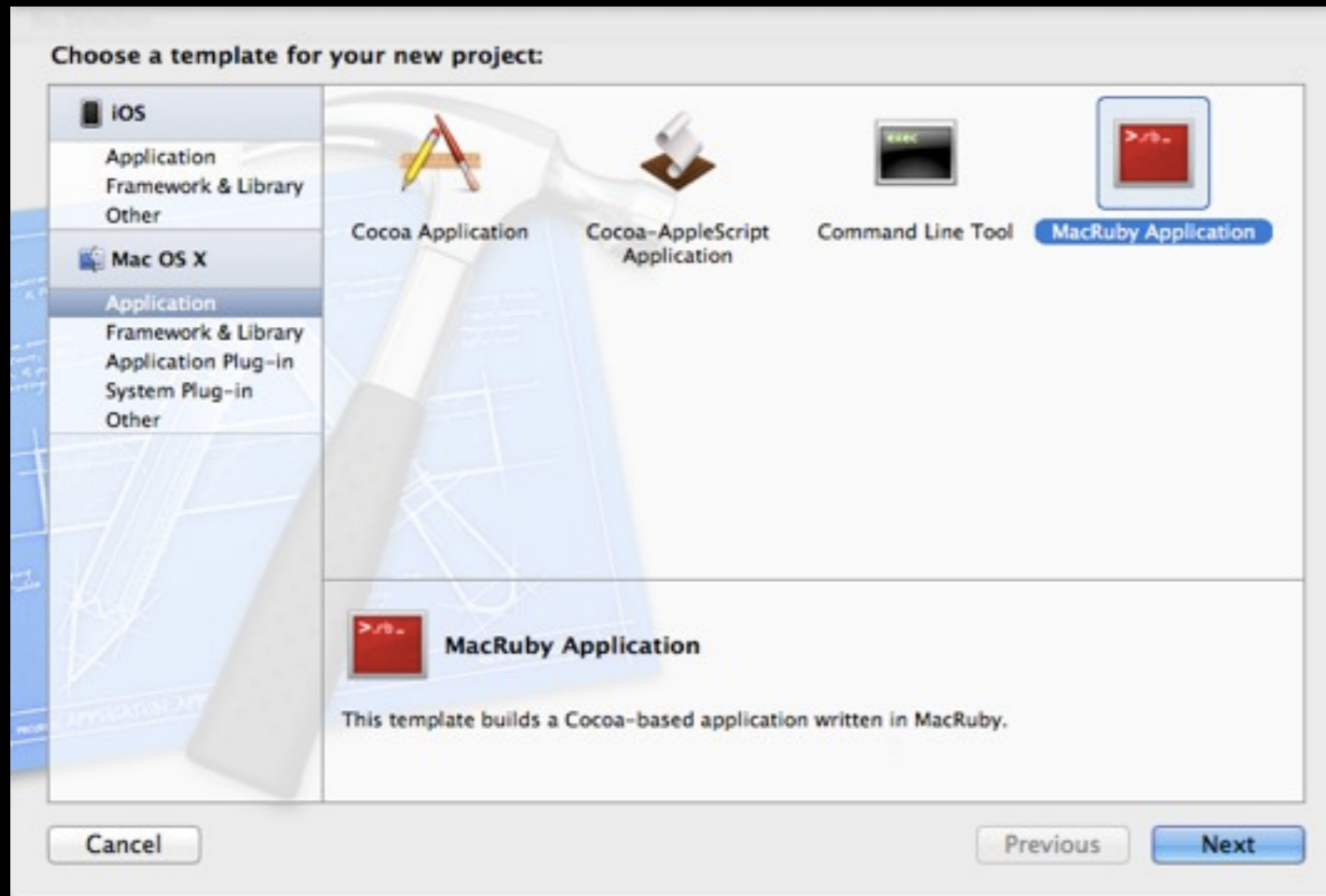
WITH THE EDGE CASE RUBY KOANS

The Koans walk you along the path to enlightenment in learn Ruby. The goal is to learn the Ruby language structure, and some common functions and libraries. We a you culture. Testing is not just something we pay lip servi something we live. It is essential in your quest to learn and things in the language.

Ruby Koans
rubykoans.com

MacRuby

Install from
www.macruby.org



New Project
from template

Resources

Submit your MacRuby app to the app store

<http://astonj.com/tech/how-to-submit-your-macruby-app-to-the-mac-app-store/>

Excellent two part video presentation on the low level details

<http://bostonrb.org/presentations/month/June-2011>

My thoughts on MacRuby and it's future

<http://cocoamanifest.net/articles/2011/05/life-after-objective-c.html>

MacRuby: The Definitive Guide

<http://ofps.oreilly.com/titles/9781449380373/>

MacRuby

Object creation, message passing
all on the ObjC runtime

Base object is NSObject

Common objects are NS equivalent

```
hash = {"key" => "value"}
```

```
NSMutableDictionary *dict =  
[NSMutableDictionary  
dictionaryWithValuesAndKeys:  
@"value", @"key", nil];
```

Common objects are NS equivalent

```
array = [1, 2, 3, 4]
```

```
NSMutableArray *array = ...
```

Use Objective C objects and Ruby objects interchangeably

```
dict = NSMutableDictionary.dictionary
```

```
dict.addValue "value", forKey: "key"
```

```
dict['foo'] = 'bar'
```

Compiles to LLVM byte code

Runs as fast as any Objective C objects

Uses Objective C garbage collector

Why?

Don't change just because

Less syntax noise

Quick prototyping

Excellent flexibility
with less ceremony

Dynamic typing is not evil

iOS?

iOS?

Technical limitations

iOS?

Technical limitations

Cultural limitations

Why?

Syntax level changes previously in the realm of Apple

@property ...

@autorelease { ... }

Ruby runtime definitions

(what if we had that power?)

```
class User

  def first_name
    @first_name
  end

  def first_name= new_name
    @first_name = new_name
  end

end
```



```
class User
  extend PropertyMethods

  property :first_name
  property :last_name
end
```

```
module PropertyMethods

  def property name
    define_method name do
      instance_variable_get "@#{name}"
    end

    define_method "#{name}=" do |value|
      instance_variable_set "@#{name}", value
    end
  end
end

end
```

```
class User
  attr_accessor :first_name
  attr_accessor :last_name
end
```

```
class User
  include AutoreleasePoolMethods

  def compute_friendships
    autorelease {
      # Do lots of stuff
    }
  end
end
```

```
module autoreleasePoolMethods
  # only for example, not needed in MacRuby

  def autorelease
    pool = NSAutoreleasePool.alloc.init
    yield
    pool.drain
  end
end

end
```

```
module autoreleasePoolMethods
  # only for example, not needed in MacRuby

  def autorelease
    pool = NSAutoreleasePool.alloc.init
    yield
    pool.drain
  end
end
```

Contrived, but you get the point

Future?


```
class User
  include DataMapper::Resource

  property :id, Serial
  property :first_name, String, length: 50
  property :last_name, String, length: 50

  validates_presence_of :first_name,
                       :last_name

  has 1, :account
  has n, :friends, 'User'

end
```

```
describe "User model" do

  let(:user) { User.new }

  describe "full_name" do
    it "generates from first and last" do
      user.first_name = "Jonathan"
      user.last_name = "Penn"
      user.full_name.should == "Jonathan Penn"
    end
  end
end

end
```


What if?

```
class TweetsViewController < UIViewController
  extend AwesomeViewController

  title "Tweets"

  on_appear do
    show_toolbar
  end

  after_appear do
    # ...
  end
end
```


MacRuby





Jonathan Penn

@jonathanpenn

jonathan@cocoamanifest.net

<http://cocoamanifest.net>



Jonathan Penn

@jonathanpenn

jonathan@cocoamanifest.net

<http://cocoamanifest.net>

Thank You.



Jonathan Penn

@jonathanpenn

jonathan@cocoamanifest.net

<http://cocoamanifest.net>

Questions?