



Ruby 程式語言

入門導覽

<https://ihower.tw>

2015/6

Agenda

- 什麼是 Ruby
- 基本語法介紹
- 一些應用

什麼是 Ruby?

https://www.ruby-lang.org/zh_tw/

- 開放原碼、物件導向的動態直譯式 (interpreted) 程式語言
- 簡單哲學、高生產力
- 精巧、自然的語法
- 創造者 Yukihiro Matsumoto, a.k.a. Matz
 - 靈感來自 Lisp, Perl, 和 Smalltalk



Matz@RubyConf Taiwan 2012

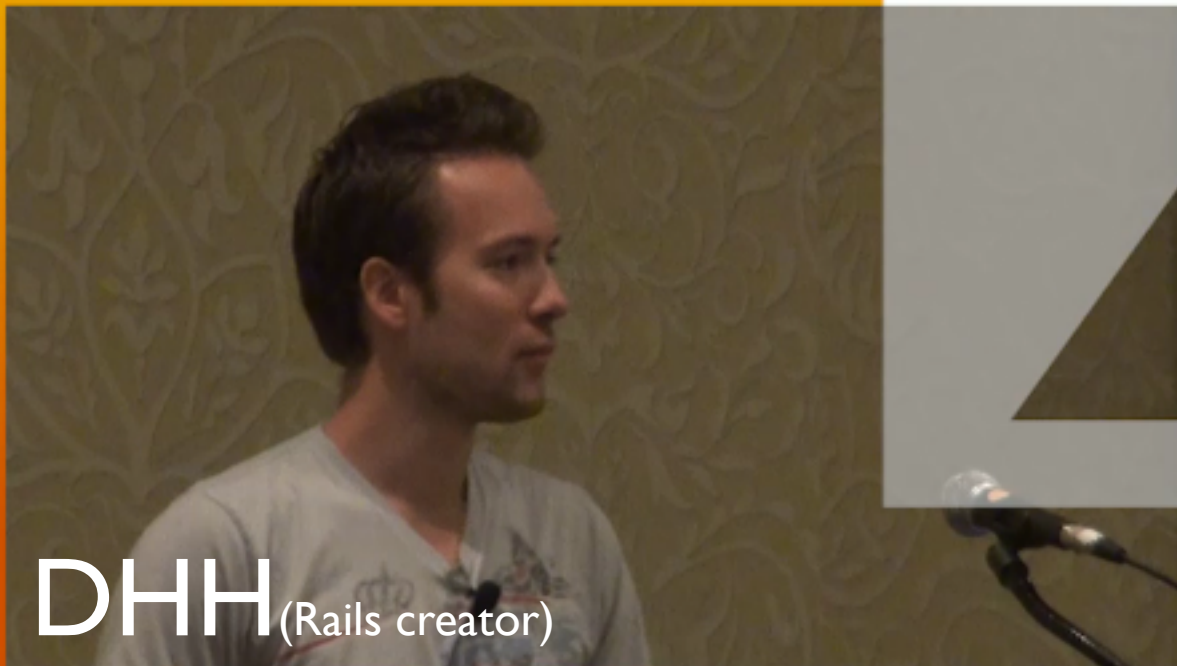
Happy?

dhh-final.mp4



Maslow's hierarchy of needs

馬斯洛需求層次理論

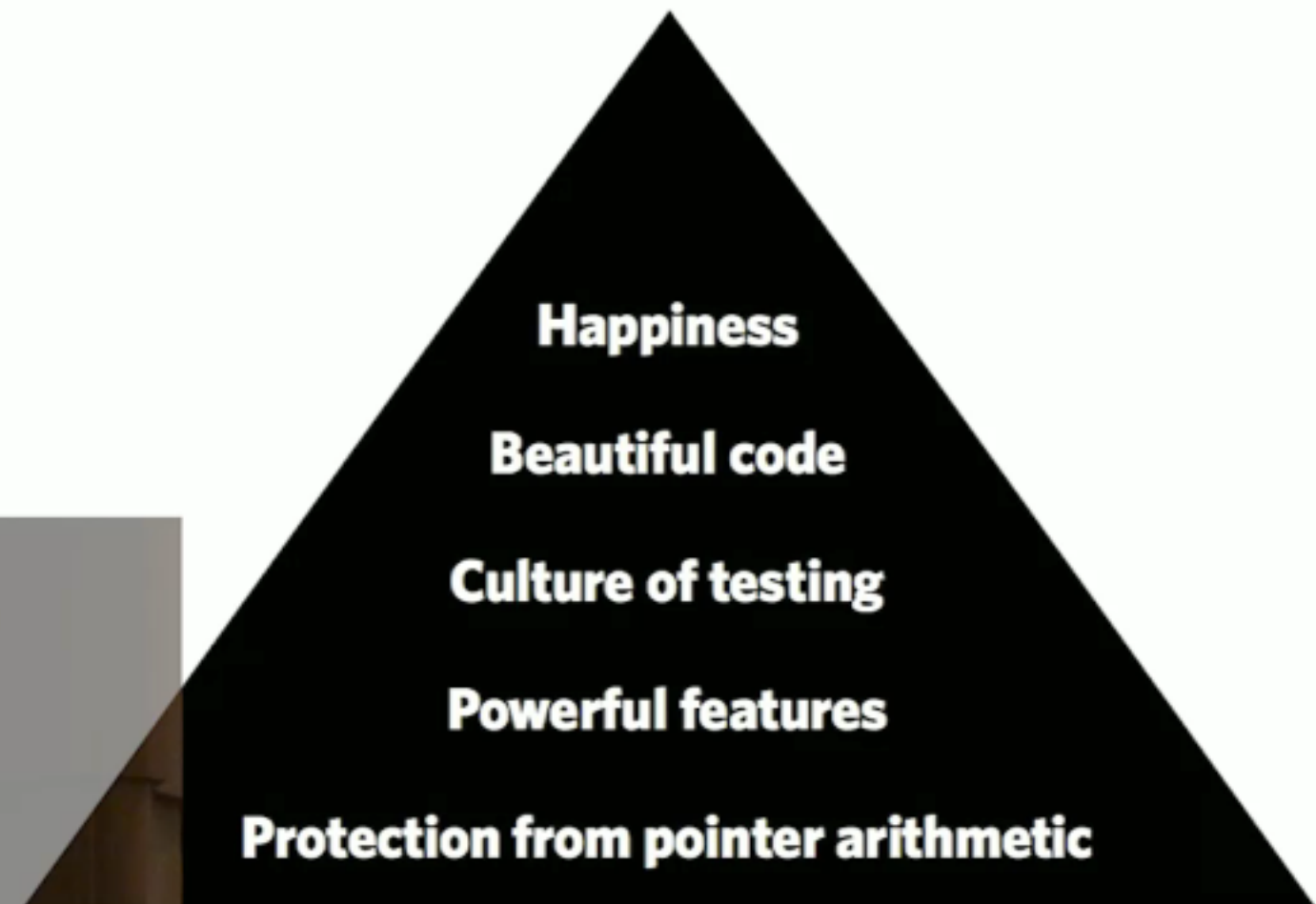


DHH (Rails creator)



17:05

Maslow's hierarchy of needs



Maslow's hierarchy of needs



Ruby

Smalltalk

Python

PHP

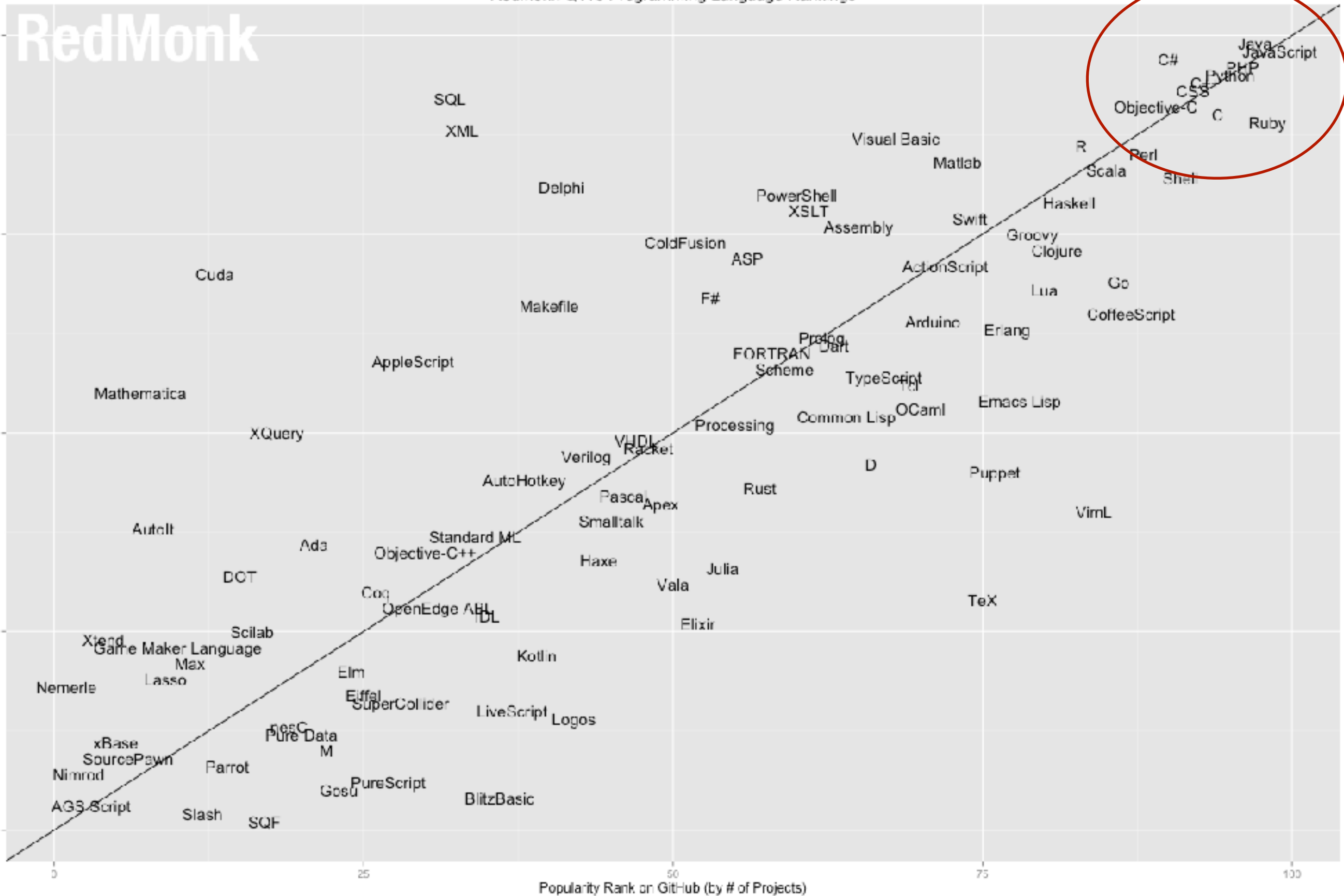
Basic



RedMonk Q115 Programming Language Rankings

RedMonk

Popularity Rank on Stack Overflow (by # of Tags)



irb: Interactive Ruby

```
irb(main):001:0>
```

```
irb(main):001:0> 1 + 1  
=> 2
```

```
irb(main):002:0>
```

PUTS 螢幕輸出

- 打開編輯器，編輯 hello.rb

```
puts "Hello World!"
```

- 執行 `ruby hello.rb`

Ruby 是動態強分型語言

- 動態 Dynamic v.s. 靜態 Static typing
 - Ruby/Perl/Python/PHP v.s. Java/C/C++
- 強 Strong v.s. 弱 Weak typing
 - Ruby/Perl/Python/Java v.s. PHP/C/C++

什麼強?弱?分型

PHP code:

```
$i = 1;  
echo "Value is " + $i  
# 1
```

C code:

```
int a = 5;  
float b = a;
```

Ruby code:

```
i=1  
puts "Value is " + i
```

```
#TypeError: can't convert Fixnum into  
String  
# from (irb):2:in `+'  
# from (irb):2
```

I. Ruby 基本語法

整數 Integer

5

-205

999999999999

0

浮點數 Float

後面有 .

54.321

0.001

-12.312

0.0

浮點數四則運算

```
puts 1.0 + 2.0
```

```
puts 2.0 * 3.0
```

```
puts 5.0 - 8.0
```

```
puts 9.0 / 2.0
```

```
# 3.0
```

```
# 6.0
```

```
# -3.0
```

```
# 4.5
```

整數四則運算

結果也會是整數

```
puts 1 + 2
```

```
puts 2 * 3
```

```
puts 5 - 8
```

```
puts 9 / 2
```

```
# 3
```

```
# 6
```

```
# -1
```

```
# 4
```

更多運算

```
puts 5 * (12-8) + -15
```

```
puts 98 + (59872 / (13*8)) * -52
```

字符串 String

```
puts 'Hello, world!'  
puts ''  
puts 'Good-bye.'
```

字串處理

```
puts 'I like ' + 'apple pie.'  
puts 'You\'re smart!'
```

```
puts '12' + 12  
#<TypeError: can't convert Fixnum into String>
```


更多字符串方法

```
var1 = 'stop'  
var2 = 'foobar'  
var3 = "aAbBcC"
```

```
puts var1.reverse # 'pots'  
puts var2.length # 6  
puts var3.upcase  
puts var3.downcase
```

字串內插

```
verb = 'work'  
where = 'office'
```

```
puts "I #{verb} at the #{where}"
```

Ruby 完全物件導向

每樣東西都是物件，包括字串和數字。

```
# 輸出 "UPPER"  
puts "upper".upcase
```

```
# 輸出 -5 的絕對值  
puts -5.abs
```

```
# 輸出 Fixnum  
puts 99.class
```

```
# 輸出 "Ruby Rocks!" 五次  
5.times do  
  puts "Ruby Rocks!"  
end
```

方法呼叫 Methods

- 所有東西都是物件(object)，可以呼叫物件的方法，例如字串、整數、浮點數。
- 透過逗點 . 來對物件呼叫方法

變數 Variable

小寫開頭，偏好單字之間以底線 _ 分隔

```
composer = 'Mozart'  
puts composer + ' was "da bomb", in his day.'
```

```
my_composer = 'Beethoven'  
puts 'But I prefer ' + my_composer + ', personally.'
```

型別轉換 Conversions

```
var1 = 2  
var2 = '5'
```

```
puts var1.to_s + var2 # 25  
puts var1 + var2.to_i # 7
```

```
puts 9.to_f / 2 # 4.5
```


常數 Constant

大寫開頭

```
foo = 1
```

```
foo = 2
```

```
Foo = 1
```

```
Foo = 2 # (irb):3: warning: already initialized constant Foo
```

```
RUBY_PLATFORM
```

```
ENV
```

nil

表示未設定値、未定義

```
nil # nil  
nil.class # NilClass
```

```
nil.nil? # true  
42.nil? # false
```

```
nil == nil # true  
false == nil # false
```

註解

偏好均使用單行註解

```
# this is a comment line  
# this is a comment line
```

```
=begin  
    This is a comment line  
    This is a comment line  
=end
```

陣列 Array

```
a = [ 1, "cat", 3.14 ]
```

```
puts a[0] # 輸出 1
```

```
puts a.size # 輸出 3
```

```
a[2] = nil
```

```
puts a.inspect # 輸出字串 [1, "cat", nil]
```

更多陣列方法

```
colors = ["red", "blue"]
```

```
colors.push("black")
```

```
colors << "white"
```

```
puts colors.join(", ") # red, blue, black, white
```

```
colors.pop
```

```
puts colors.last #black
```

走訪迴圈

each method

```
languages = ['Ruby', 'Javascript', 'Perl']
```

```
languages.each do |lang|  
  puts 'I love ' + lang + '!'  
end
```

```
# I Love Ruby  
# I Love Javascript  
# I Love Perl
```

雜湊 Hash

(Associative Array)

```
config = { "foo" => 123, "bar" => 456 }
```

```
puts config["foo"] # 輸出 123
```

字串符號 Symbols

唯一且不會變動的識別名稱

```
config = { :foo => 123, :bar => 456 }
```

```
puts config[:foo] # 輸出 123
```


更簡潔的語法

```
config = { foo: 123, bar: 456 }
```

```
puts config[:foo] # 輸出 123
```

走訪雜湊

each method

```
config = { :foo => 123, :bar => 456 }  
config.each do |key, value|  
  puts "#{key} is #{value}"  
end
```

```
# foo is 123  
# bar is 456
```

流程控制

Flow Control

比較方法

```
puts 1 > 2  
puts 1 < 2  
puts 5 >= 5  
puts 5 <= 4  
puts 1 == 1  
puts 2 != 1
```

```
puts ( 2 > 1 ) && ( 2 > 3 ) # and  
puts ( 2 > 1 ) || ( 2 > 3 ) # or
```

控制結構 If

Perl Style

```
if account.total > 100000
    puts "large account"
elsif account.total > 25000
    puts "medium account"
else
    puts "small account"
end
```

三元運算子

expression ? true_expression : false_expression

```
x = 3  
puts ( x > 3 )? "大於三" : "小於或等於三"  
  
# 輸出 小於或等於三
```

控制結構 Case

```
case name
  when "John"
    puts "Howdy John!"
  when "Ryan"
    puts "Whatz up Ryan!"
  else
    puts "Hi #{name}!"
end
```

迴圈

while, loop, until, next and break

```
i=0
while ( i < 10 )
  i += 1
  next if i % 2 == 0 #跳過雙數
  puts i
end
```

```
i = 0
i += 1 until i > 10
puts i
# 輸出 11
```

```
i = 0
loop do
  i += 1
  break if i > 10 # 中斷迴圈
end
```


真或假

只有 false 和 nil 是假，其他為真

```
puts "not execute" if nil  
puts "not execute" if false
```

```
puts "execute" if true # 輸出 execute  
puts "execute" if "" # 輸出 execute (和JavaScript不同)  
puts "execute" if 0 # 輸出 execute (和C不同)  
puts "execute" if 1 # 輸出 execute  
puts "execute" if "foo" # 輸出 execute  
puts "execute" if Array.new # 輸出 execute
```

Regular Expressions

與 Perl 接近的語法

抓出手機號碼

```
phone = "123-456-7890"
```

```
if phone =~ /(\d{3})-(\d{3})-(\d{4})/  
    ext  = $1  
    city = $2  
    num  = $3  
end
```

方法定義 Methods

def 開頭 end 結尾

```
def say_hello(name)
  result = "Hi, " + name
  return result
end
```

return
n 可省略，最
後一行就是


```
puts say_hello('ihower')
# 輸出 Hi, ihower
```

字
串相

預設參數

```
def say_hello(name="nobody")  
  result = "Hi, " + name  
  result  
end
```

```
puts say_hello  
# 輸出 Hi, nobody
```



括
號可以

? 與 ! 的慣例

方法名稱可以用?或!結尾，前者表示會回傳 Boolean，
後者暗示會有某種 side-effect。

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

```
array.empty? # false
```

```
array.sort # [1,2,3]
```

```
array.inspect # [2,1,3]
```

```
array.sort! # [1,2,3]
```

```
array.inspect # [1,2,3]
```

物件導向

Object-Oriented Programming

- OOP 一種將「資料」和「方法」封裝到物件的設計方式
- 使用「類別 Class」來定義出「物件 Object」，類別可說是物件的 Template 樣板

Car class



```
graph TD; A[Car class] --> B[car1 object]; A --> C[car2 object]; A --> D[car3 object];
```

car1 object

@name="ihower"
@color="red"

car2 object

@name="jimmy"
@color="yellow"

car3 object

@name="john"
@color="blue"

如何設計處理分數的相加，假設分子是 x 、分母是 y

```
def add_rational_numerator(x1, y1, x2, y2)
    x1*y2 + x2*y1
end
```

```
def add_rational_denominator(x1, y1, x2, y2)
    y1*y2
end
```

$2/3 + 3/4$

x1 = 2

y1 = 3

x2 = 3

y2 = 4

answer_x = add_rational_numerator(x1, y1, x2, y2)

answer_y = add_rational_denominator(x1, y1, x2, y2)

資料跟方法都沒有


```
class MyRational

  attr_accessor :x, :y

  def initialize(x, y)
    @x, @y = x, y
  end

  def add(target)
    MyRational.new(@x*target.y + @y*target.x, @y*target.y)
  end

end

# 2/3 + 3/4
a = MyRational.new(2,3)
b = MyRational.new(3,4)
a.add(b)
```

類別 Classes

大寫開頭，使用 new 可以建立出物件

```
color_string = String.new  
color_string = "" # 等同
```

```
color_array = Array.new  
color_array = [] # 等同
```

```
color_hash = Hash.new  
color_hash = {} # 等同
```

```
time = Time.new  
puts time
```

大
寫開頭的常

類別 Class

```
class Person
```

建構式

物件變數

```
def initialize(name)
```

```
  @name = name
```

```
end
```

```
def say(word)
```

```
  puts "#{word}, #{@name}"
```

```
end
```

字串

```
end
```

```
p1 = Person.new("ihower")
```

```
p2 = Person.new("ihover")
```

```
p1.say("Hello") # 輸出 Hello, ihower
```

```
p2.say("Hello") # 輸出 Hello, ihover
```

類別 Class (續)

```
class Person
```

類別變數

```
  @@name = "ihower"
```

```
  def self.say  
    puts @@name  
  end
```

類別方法

```
end
```

```
Person.say # 輸出 ihower
```

資料封裝

- 所有的物件變數(@開頭)、類別變數(@@開頭)，都是封裝在類別內部，類別外無法存取。
- 需透過定義 public 方法才可以存取到

```
class Person
  def initialize(name)
    @name = name
  end
end
```

```
p = Person.new('ihower')
p.name
=> NoMethodError
p.name='peny'
=> NoMethodError
```

```
class Person

  def initialize(name)
    @name = name
  end

  def name
    @name
  end

  def name=(name)
    @name = name
  end

end
```

```
p = Person.new('ihower')
p.name
=> "ihower"
p.name = "peny"
=> "peny"
```

方法封裝

預設是 public 公開

```
class MyClass
```

```
  def public_method  
  end
```

```
  private
```

```
  def private_method  
  end
```

```
  protected
```

```
  def protected_method  
  end
```

```
end
```

```
class MyClass
```

```
  def public_method  
  end
```

```
  def private_method  
  end
```

```
  def protected_method  
  end
```

```
  public :public_method  
  private :private_method  
  protected :protected_method
```

```
end
```

類別 Class body 也可以執行程式

attr_accessor, attr_writer, attr_reader

The diagram illustrates the equivalence between using `attr_accessor` and manually defining methods within a class body. It features three code snippets arranged in a triangular layout, with two arrows pointing from the leftmost snippet to the other two, and the Chinese text "等同於" (equivalent to) centered between the arrows.

```
class Person  
  attr_accessor :name  
end
```

等同於

```
class Person  
  def name  
    @name  
  end  
  
  def name=(val)  
    @name = val  
  end  
end
```


Class 繼承

```
class Pet  
  attr_accessor :name, :age  
end
```

```
class Cat < Pet  
end
```

```
class Dog < Pet  
end
```

Module (I) Namespace

```
module MyUtil  
  
  def self.foobar  
    puts "foobar"  
  end  
  
end
```

```
end
```

```
MyUtil.foobar  
# 輸出 foobar
```

Module(2) Mixins 繼承

```
module Debug
  def who_am_i?
    "#{self.class.name} (\##{self.object_id}): #{self.to_s}"
  end
end
```

```
class Foo
  include Debug # 這個動作叫做 Mixin
  # ...
end
```

```
class Bar
  include Debug
  include XXX
  include YYY
  # ...
end
```

```
ph = Foo.new("12312312")
et = Bar.new("78678678")
ph.who_am_i? # 輸出 "Foo (#330450): 12312312"
et.who_am_i? # 輸出 "Bar (#330420): 78678678"
```



Ruby
使用 Module 來解決
多重繼承問題

多型(Polymorphism)

即使不同類別，只要介面一致就可以處理

鴨子

```
class Duck
  def quack
    puts "quack!"
  end
end
```

野鴨 (不用繼承)

```
class Mallard
  def quack
    puts "qwuaacck!! quak!"
  end
end
```

動態型別 (duck typing)

```
birds = [Duck.new, Mallard.new, Object.new]
```

```
# 迭代陣列，並呼叫方法（無須擔心型別）
```

```
birds.each do |duck|
```

```
    duck.quack if duck.respond_to? :quack
```

```
end
```

迭代器 iterator

- 不同於 while 迴圈用法，each 是一個陣列的方法，走訪其中的元素，我們稱作迭代器(iterator)
- 其中 do ... end 是 each 方法的參數，稱作匿名方法(code block)

最簡單的迭代器

```
3.times do  
  puts 'Good Job!'  
end
```

```
# Good Job!  
# Good Job!  
# Good Job!
```

code block

一種匿名方法，或稱作 closure

```
{ puts "Hello" } # 這是一個 block
```

```
do
```

```
  puts "Blah" # 這也是一個 block
```

```
  puts "Blah"
```

```
end
```


code block

內部迭代器(iterator)

處理陣列 people


```
people = ["David", "John", "Mary"]  
people.each do |person|  
  puts person  
end
```

反覆五次

```
5.times { puts "Ruby rocks!" }
```

從一數到九

```
1.upto(9) { |x| puts x }
```



所以我們
將很少用到

code block

其他迭代方式

迭代並造出另一個陣列

```
a = [ "a", "b", "c", "d" ]
```

```
b = a.map { |x| x + "!" }
```

```
puts b.inspect
```

結果是 ["a!", "b!", "c!", "d!"]

找出符合條件的值

```
b = [1,2,3].find_all{ |x| x % 2 == 0 }
```

```
b.inspect
```

結果是 [2]

code block

當作判斷條件

迭代並根據條件刪除

```
a = [ "a", "b", "c" ]
```

```
a.delete_if { |x| x >= "b" }
```

結果是 ["a"]

客製化排序

```
[2,1,3].sort! { |a, b| b <=> a }
```

結果是 [3, 2, 1]

code block

有沒有 functional programming 的 fu?

計算總和

```
(5..10).reduce { |sum, n| sum + n }
```

找出最長字串 find the longest word

```
longest = ["cat", "sheep", "bear"].reduce do |memo,  
word|
```

```
  ( memo.length > word.length )? memo : word
```

```
end
```

code block

僅執行一次呼叫 pre- post- processing

```
file = File.new("testfile", "r")  
# ...處理檔案  
file.close
```

```
# 但 Ruby 習慣用以下寫法  
File.open("testfile", "r") do |file|  
    # ...處理檔案  
end  
# 檔案自動關閉
```

Yield

在方法中使用 yield 來執行 code block

定義方法

```
def call_block  
  puts "Start"  
  yield  
  yield  
  puts "End"  
end
```

```
call_block { puts "Blocks are cool!" }
```

輸出

"Start"

"Blocks are cool!"

"Blocks are cool!"

"End"

帶參數的 code block

```
def call_block  
  yield(1)  
  yield(2)  
  yield(3)  
end
```

```
call_block { |i|  
  puts "#{i}: Blocks are cool!"  
}
```

輸出

"1: Blocks are cool!"

"2: Blocks are cool!"

"3: Blocks are cool!"

Proc object

將 code block 明確轉成物件

```
def call_block(&block)
  block.call(1)
  block.call(2)
  block.call(3)
end
```

```
call_block { |i| puts "#{i}: Blocks are cool!" }
```

```
# 或是先宣告出 proc object
```

```
proc_1 = Proc.new { |i| puts "#{i}: Blocks are cool!" }
proc_2 = lambda { |i| puts "#{i}: Blocks are cool!" }
```

```
call_block(&proc_1)
call_block(&proc_2)
```

```
# 輸出
```

```
# "1: Blocks are cool!"
# "2: Blocks are cool!"
# "3: Blocks are cool!"
```


傳遞不定參數

```
def my_sum(*val)  
  val.inject(0) { |sum, v| sum + v }  
end
```

```
puts my_sum(1,2,3,4)
```

輸出 10

參數尾 Hash 可省略 { }

```
def my_print(a, b, options)
  puts a
  puts b
  puts options[:x]
  puts options[:y]
  puts options[:z]
end
```

```
my_print("A", "B", { :x => 123, :z => 456 } )
my_print("A", "B", :x => 123, :z => 456) # 結果相同
```

```
# 輸出 A
# 輸出 B
# 輸出 123
# 輸出 nil
# 輸出 456
```

例外處理

raise, begin, rescue, ensure

```
raise "Not works!!"  
# 丟出一個 RuntimeError
```

```
# 自行自定例外物件  
class MyException < RuntimeError  
end
```

```
raise MyException
```

```
begin  
  puts 10 / 0  
rescue => e  
  puts e.class  
ensure  
  # ...  
end
```

```
# 輸出 ZeroDivisionError
```

動手練習

Exercises

- <http://www.codecademy.com/en/tracks/ruby>
- <http://www.gotealeaf.com/books/ruby>
- learnrubythehardway.org/book/

3. Ruby 的應用

how ruby change the world!

Web framework

- MVC
- ORM
- URL Routing
- View Template

[Overview](#) | [Download](#) | [Deploy](#) | [Bugs/Patches](#) | [Screencasts](#) | [Documentation](#) | [Ecosystem](#) | [Community](#) | [Blog](#)



Web development that doesn't hurt

Ruby on Rails® is an open-source web framework that's optimized for programmer happiness and sustainable productivity. It lets you write beautiful code by favoring convention over configuration.

[Rails 4.0: Beta1 released](#), [Rails 3.2.13 released!](#), [The People Behind Rails 4](#)

Get Excited

```
class PostsController <
  # GET /posts
  # GET /posts.xml
  def index
    @posts = Post.find(

    respond_to do |form
      format.html # ind
      format.xml { ren
      format.json { ren
      format.atom
    end
  end
```

[Screencasts](#)

Get Started



[3.2.13 released Mar 18, 2013](#)

Get Better



[API, Guides, Books](#)

Get Involved

IRC
Mailing lists
Bug tracker
Wiki

[Join the community](#)

“Ruby on Rails is a breakthrough in lowering the barriers of entry to programming. Powerful web applications that formerly might have taken weeks or months to develop can be produced in a matter of days.”

-Tim O'Reilly, Founder of O'Reilly Media

[Read more quotes](#)

Who is already on Rails?

Tens of thousands of Rails applications are already live. People are using Rails in the tiniest part-time operations to the biggest companies.

Fork me on GitHub

Sinatra

README
DOCUMENTATION
CONTRIBUTE
CODE
CREW
ABOUT

Put this in
your pipe

```
require 'sinatra'

get '/hi' do
  "Hello world!"
end
```

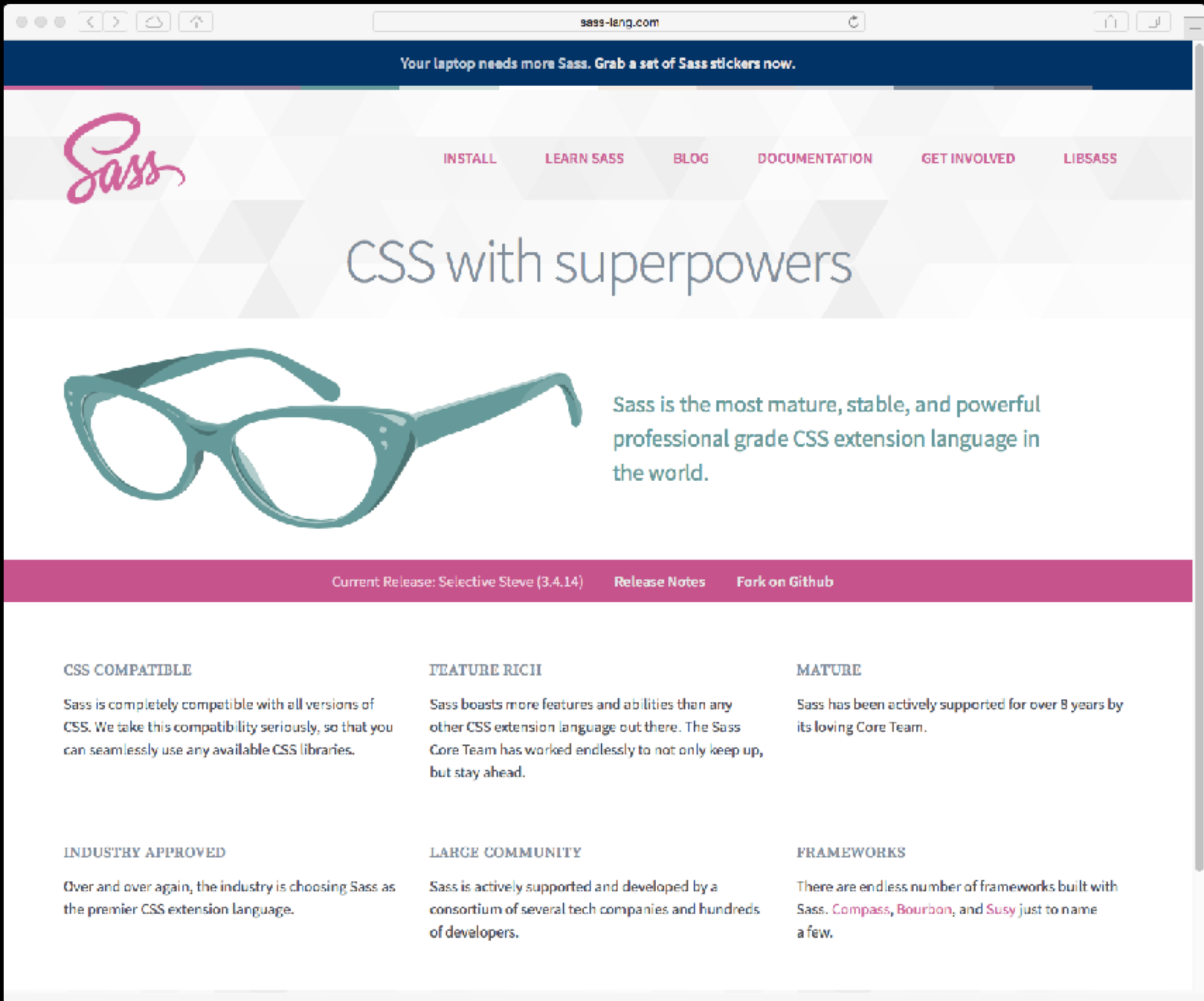
and smoke it

```
$ gem install sinatra
$ ruby hi.rb
== Sinatra has taken the stage ..
>> Listening on 0.0.0.0:4567
```


Web Designer Tools

- Sass/Less/Haml
- Compass
- Middleman





Your laptop needs more Sass. Grab a set of Sass stickers now.



[INSTALL](#)

[LEARN SASS](#)

[BLOG](#)

[DOCUMENTATION](#)

[GET INVOLVED](#)

[LIBSASS](#)

CSS with superpowers



Sass is the most mature, stable, and powerful professional grade CSS extension language in the world.

[Current Release: Selective Steve \(3.4.14\)](#)

[Release Notes](#)

[Fork on Github](#)

CSS COMPATIBLE

Sass is completely compatible with all versions of CSS. We take this compatibility seriously, so that you can seamlessly use any available CSS libraries.

FEATURE RICH

Sass boasts more features and abilities than any other CSS extension language out there. The Sass Core Team has worked endlessly to not only keep up, but stay ahead.

MATURE

Sass has been actively supported for over 8 years by its loving Core Team.

INDUSTRY APPROVED




Over and over again, the industry is choosing Sass as the premier CSS extension language.

LARGE COMMUNITY

Sass is actively supported and developed by a consortium of several tech companies and hundreds of developers.

FRAMEWORKS

There are endless number of frameworks built with Sass. [Compass](#), [Bourbon](#), and [Susy](#) just to name a few.

Use Sass / Compass in    easily

[Documentation](#) [FAQ](#) [Issues](#)



\$10 **BUY NOW**

30% of sales donated to [UMDF](#)

 Star 462

[Tweet Follow @handlino](#)

Compass is a stylesheet authoring framework that makes your stylesheets and markup easier to build and maintain. With compass, you write your stylesheets in **Sass** instead of plain CSS.

Compass.app is a menubar only app for Sass and Compass. It helps designers compile stylesheets easily without resorting to command line interface.

Compass.app is written in Java (JRuby), and works in mac, linux and pc. You do not need to install Ruby environment to use it.



We also made
Fire.app - The fast
prototyping tool

Features

- + Works in mac, linux and pc
- + Built-in Web server

Recommendation





“ I don't always use GUI to compile my Compass Project, But when I do, I use Compass app. - [Chris Epstein](#), creator of Compass framework



makes developing websites simple

```
$ gem install middleman
```

Middleman is a static site generator using all the shortcuts and tools in modern web development. [Getting started.](#)

 Watch  Follow @middlemanapp

Middleman Basics

Getting Started

Advanced Features

Local Data

Community

Community Extensions

Testing/BDD

Ruby community loves testing

- RSpec
- Cucumber

<http://ihower.tw/blog/archives/5438>

<http://ihower.tw/blog/archives/5983>

What's BDD?

- An improved xUnit Framework
- Focus on clearly **describe the expected behavior**
- The emphasis is **Tests as Documentation** rather than merely using tests for verification.

Terminology changed

New paradigm: Executable Specification

- “Test” becomes “Spec”
- “Assertion” becomes “Expectation”
- “test method” becomes “example” (RSpec)
- “test case” becomes “example group” (RSpec)

RSpec

Overview

RSpec is testing tool for the Ruby programming language. Born under the banner of Behaviour-Driven Development, it is designed to make Test-Driven Development a productive and enjoyable experience with features like:

- a rich command line program (the `rspec` command)
- textual descriptions of examples and groups (**rspec-core**)
- flexible and customizable reporting
- extensible expectation language (rspec-expectations)
- built-in mocking/stubbing framework (rspec-mocks)

Documentation

RDoc

- <http://rubydoc.info/gems/rspec-core>
- <http://rubydoc.info/gems/rspec-expectations>
- <http://rubydoc.info/gems/rspec-mocks>
- <http://rubydoc.info/gems/rspec-rails>

Cucumber Features

- <http://relishapp.com/rspec>

RSpec-1.x

- <http://old.rspec.info>

The RSpec Book

The RSpec Book will introduce you to RSpec, Cucumber, and a number of other tools that make up the Ruby BDD family. Replete with tutorials and practical examples, the RSpec Book will help you get your BDD on, taking you from executable requirements to working software that is clean, well tested, well documented, flexible and highly maintainable.



WIKI

Detailed documentation

EXAMPLES

Use your mother tongue

TUTORIALS

By the community

TICKETS

Requests/Bugs

MAILING LIST

Ask questions

IRC

Get instant help



Cucumber

behaviour driven development
with elegance and joy

1: Describe behaviour in plain text

```
Feature: Addition
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers

Scenario: Add two numbers
  Given I have entered 50 into the calculator
  And I have entered 70 into the calculator
  When I press add
  Then the result should be 120 on the screen
```

2: Write a step definition in Ruby

```
Given /I have entered (.*) into the calculator/ do |n|
  calculator = Calculator.new
  calculator.push(n.to_i)
end
```

3: Run and watch it fail

```
$ cucumber features/addition.feature
Feature: Addition # features/addition.feature
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers
Scenario: Add two numbers # features/addition.feature
  Given I have entered 50 into the calculator # features/step_definitions/calculator_steps.rb:2:in `Given /
  uninitialized constant Calculator (NameError)
  ./features/step_definitions/calculator_steps.rb:2:in `Given /
  features/addition.feature:7:in `Given I have entered 50 into
  And I have entered 70 into the calculator # features/step_definitions/calculator_steps.rb:3:in `And I have entered 70 into the calculator'
  When I press add # features/step_definitions/calculator_steps.rb:4:in `When I press add'
  Then the result should be 120 on the screen # features/step_definitions/calculator_steps.rb:5:in `Then the result should be 120 on the screen'
```

4. Write code to make the step pass

```
class Calculator
  def push(n)
    @args ||= []
    @args << n
  end
end
```

5. Run again and see the step pass

```
$ cucumber features/addition.feature
Feature: Addition # features/addition.feature
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers
Scenario: Add two numbers # features/addition.feature
  Given I have entered 50 into the calculator # features/step_definitions/calculator_steps.rb:2:in `Given /
  And I have entered 70 into the calculator # features/step_definitions/calculator_steps.rb:3:in `And I have entered 70 into the calculator'
  When I press add # features/step_definitions/calculator_steps.rb:4:in `When I press add'
  Then the result should be 120 on the screen # features/step_definitions/calculator_steps.rb:5:in `Then the result should be 120 on the screen'
```

6. Repeat 2-5 until green like a cucumber

```
$ cucumber features/addition.feature
Feature: Addition # features/addition.feature
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers
Scenario: Add two numbers # features/addition.feature
  Given I have entered 50 into the calculator # features/step_definitions/calculator_steps.rb:2:in `Given /
  And I have entered 70 into the calculator # features/step_definitions/calculator_steps.rb:3:in `And I have entered 70 into the calculator'
  When I press add # features/step_definitions/calculator_steps.rb:4:in `When I press add'
  Then the result should be 120 on the screen # features/step_definitions/calculator_steps.rb:5:in `Then the result should be 120 on the screen'
```

7. Repeat 1-6 until the money runs out

Cucumber lets software development teams describe how software should behave in

Learn more!

4th April 2013

Extended

Download

You need Ruby installed. Then just run
gem install cucumber

DevOps

- Chef
- Puppet
- Vagrant



OPSCODE
RULE THE CLOUD

[Customer Login](#) | [Sign Up](#) | [Account Management](#) | [Recover Password](#) [f](#) [t](#)

[Products & Services](#) ▾ | [Solutions](#) ▾ | [Customers](#) | [Support](#) ▾ | [About](#) ▾ | [Newsroom](#) ▾ | [Communi](#)

[Products](#)[Hosted Chef](#)[Private Chef](#)[Chef](#)

Chef

Accelerate your business

Chef is an open-source automation platform built to address the hardest infrastructure challenges on the planet. Chef gives you the power and flexibility you need to move faster in a complex world - from rapid provisioning and deployment of servers to the automated delivery of applications and services—at any scale.

How you use Chef is up to you. Here are the most common use cases

Configuration Management

Build a framework to consistently deploy servers and scale applications throughout your infrastructure. Implement policies to define and enforce software distribution, patch management, operating system and application compliance, security, and ad-hoc change processes. Get notifications around Chef runs to enable data gathering for trending analysis of usage. Create a blueprint of your infrastructure – so it can be built or rebuilt consistently from scratch in minutes.

[Learn more about Configuration Management](#)

Cloud Management

Automate and manage public, private, and hybrid cloud infrastructure to meet peak demand with no interruptions in service by rapidly provisioning and de-provisioning servers. Easily automate cloud infrastructure using plugins for Amazon Web Services, HP Cloud, VMware vCloud, Microsoft Windows Azure, Rackspace, Google Compute Engine, Eucalyptus, Openstack and many others.

[Learn more about Cloud Management](#)

Continuous Delivery

Reliably release high-quality software quickly, by automating the build, test, configuration and deployment functions. Give your organization the ability to make continuous incremental change with minimal risk. Get applications and updates to market in days or hours, instead of weeks or months.

Try Private Chef.
First 5 Nodes Free.

Enterprise-Grade Automation and
Configuration Management.

[Sign up now.](#) ➔



The power and speed of Opscode. The freedom and flexibility of Chef. Completely customized for your business behind your firewall. No matter how complex the realities of your business, Private Chef makes it easy to deploy servers and scale applications throughout your entire infrastructure.

[Learn more](#) ➔
[Start your free trial](#) ➔





What is Puppet?

Puppet

What is Puppet?

[Puppet Enterprise](#)[New in Puppet Enterprise](#)[Puppet Open Source Enterprise vs. Open Source](#)[Puppet Forge](#)[Components & Requirements](#)[FAQ](#)[How to Buy](#)

MCollective

[Introduction](#)[Screencasts](#)[EC2 Demo](#)[Terminology](#)[Security Overview](#)

Open Source Projects

[Facter](#)[Dashboard](#)

Related Content

[White Papers](#)

Puppet is IT automation software that helps system administrators manage infrastructure throughout its lifecycle, from provisioning and configuration to patch management and compliance. Using Puppet, you can easily automate repetitive tasks, quickly deploy critical applications, and proactively manage change, scaling from 10s of servers to 1000s, on-premise or in the cloud.

Puppet is available as both open source and commercial software. You can see the differences [here](#) and decide which is right for your organization.

How Puppet Works

Puppet uses a declarative, model-based approach to IT automation.

1. **Define** the desired state of the infrastructure's configuration using Puppet's declarative configuration language.
2. **Simulate** configuration changes before enforcing them.
3. **Enforce** the deployed desired state automatically, correcting any configuration drift.
4. **Report** on the differences between actual and desired states and any changes made enforcing the desired state.



1 Define: With Puppet's declarative language you design a graph of relationships between resources within reusable modules. These modules define your infrastructure in its desired state.

[Download Free Now](#)



VAGRANT

VMWARE FUSION

DOWNLOADS

DOCUMENTATION

SUPPORT

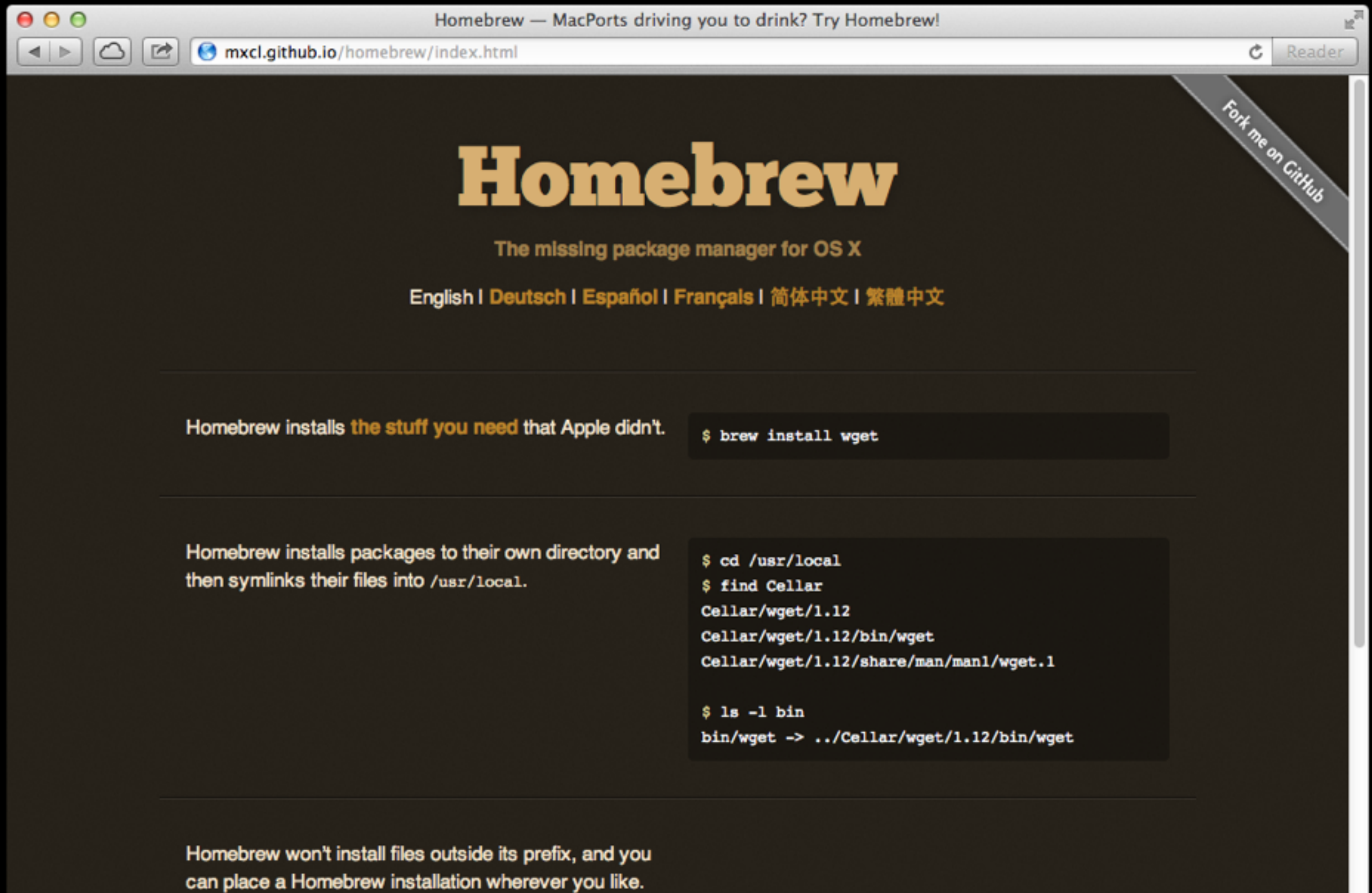
ABOUT

Development environments made easy.

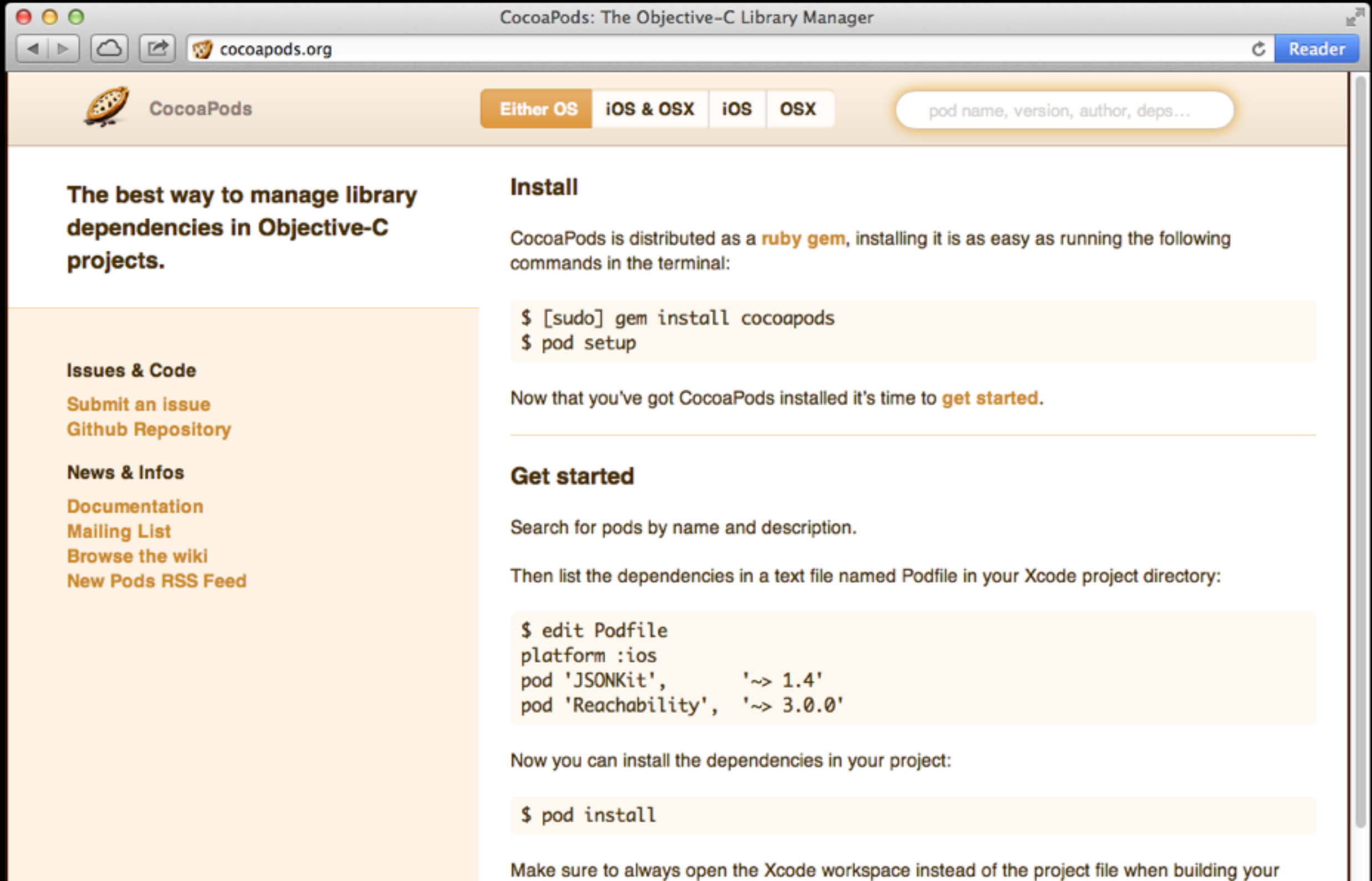
Create and configure lightweight, reproducible, and portable development environments.

[DOWNLOAD](#)[GET STARTED](#)

Mac



Mac



The screenshot shows a web browser window titled "CocoaPods: The Objective-C Library Manager". The address bar shows "cocoapods.org". The page has a navigation bar with the CocoaPods logo, a search bar with the placeholder "pod name, version, author, deps...", and tabs for "Either OS", "iOS & OSX", "iOS", and "OSX".

The best way to manage library dependencies in Objective-C projects.

Issues & Code
[Submit an issue](#)
[Github Repository](#)

News & Infos
[Documentation](#)
[Mailing List](#)
[Browse the wiki](#)
[New Pods RSS Feed](#)

Install

CocoaPods is distributed as a **ruby gem**, installing it is as easy as running the following commands in the terminal:

```
$ [sudo] gem install cocoapods  
$ pod setup
```

Now that you've got CocoaPods installed it's time to **get started**.

Get started

Search for pods by name and description.

Then list the dependencies in a text file named Podfile in your Xcode project directory:

```
$ edit Podfile  
platform :ios  
pod 'JSONKit', '~> 1.4'  
pod 'Reachability', '~> 3.0.0'
```

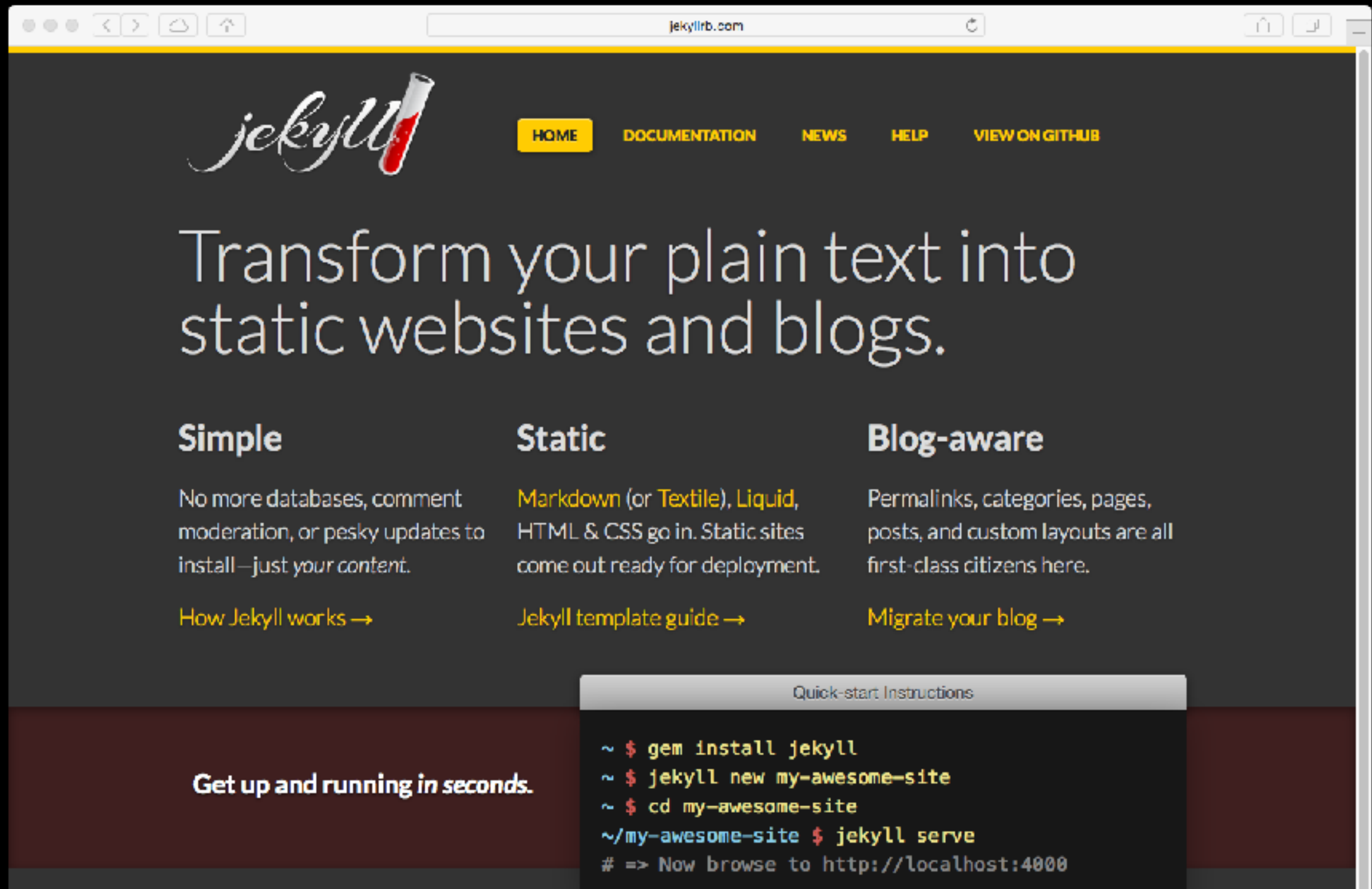
Now you can install the dependencies in your project:

```
$ pod install
```

Make sure to always open the Xcode workspace instead of the project file when building your

Blogging

(static HTML generator)



The image is a screenshot of a web browser displaying the Jekyll homepage. The browser's address bar shows 'jekyllrb.com'. The website has a dark grey background with a yellow header bar. The Jekyll logo, which includes the word 'jekyll' in a script font and a red test tube icon, is in the top left. To its right is a navigation menu with links: 'HOME' (highlighted in yellow), 'DOCUMENTATION', 'NEWS', 'HELP', and 'VIEW ON GITHUB'. Below the navigation, a large white text block reads 'Transform your plain text into static websites and blogs.' Underneath this, there are three columns of text, each with a heading and a description. The first column is titled 'Simple' and describes the ease of installation. The second is titled 'Static' and describes the input formats and deployment readiness. The third is titled 'Blog-aware' and describes the blogging features. Each column has a corresponding link at the bottom. At the bottom of the page, there is a dark red banner with the text 'Get up and running in seconds.' and a 'Quick-start Instructions' box containing terminal commands for installing and running Jekyll.

jeekyll

[HOME](#) [DOCUMENTATION](#) [NEWS](#) [HELP](#) [VIEW ON GITHUB](#)

Transform your plain text into static websites and blogs.

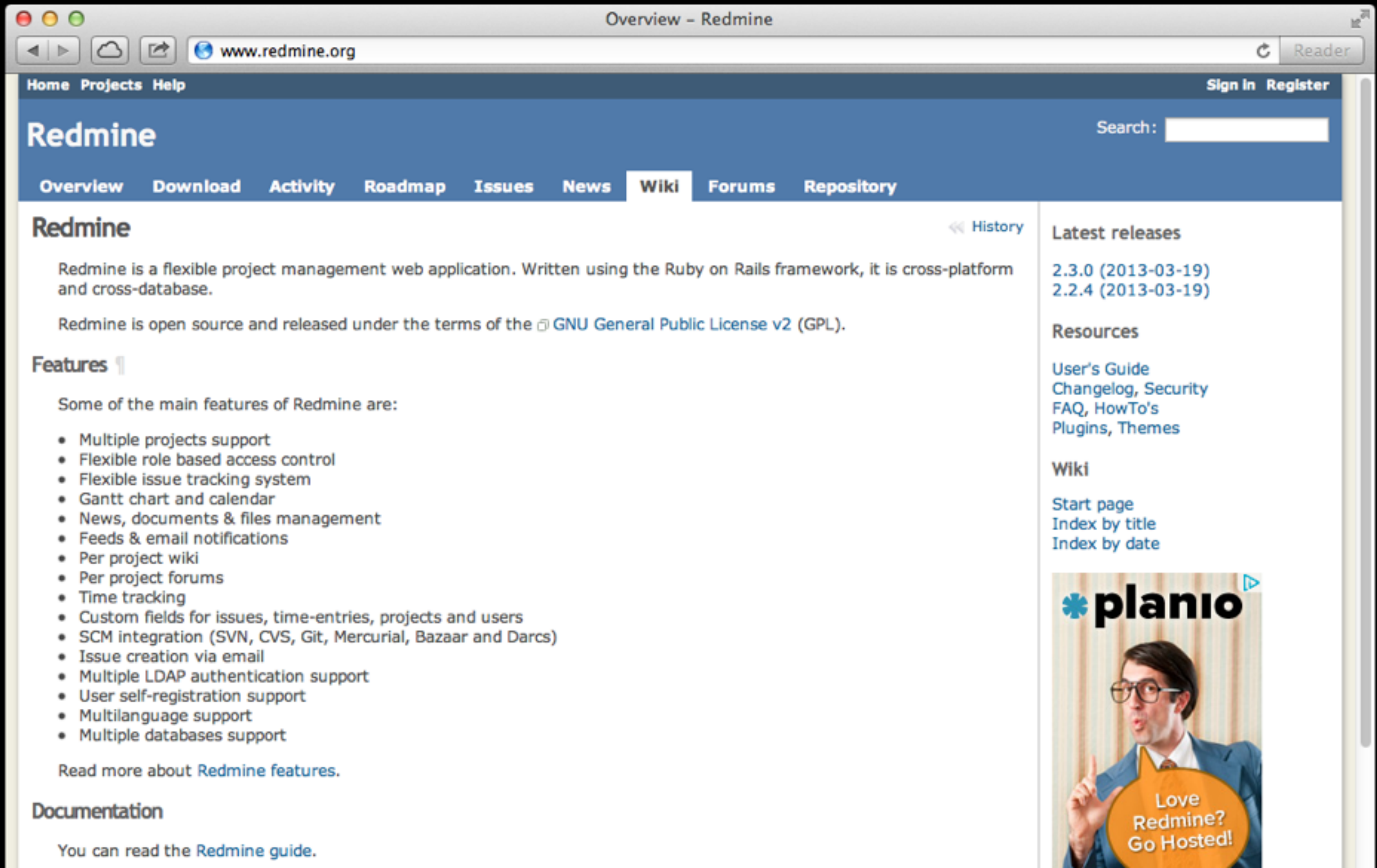
<h3>Simple</h3> <p>No more databases, comment moderation, or pesky updates to install—just <i>your content</i>.</p> <p>How Jekyll works →</p>	<h3>Static</h3> <p>Markdown (or Textile), Liquid, HTML & CSS go in. Static sites come out ready for deployment.</p> <p>Jekyll template guide →</p>	<h3>Blog-aware</h3> <p>Permalinks, categories, pages, posts, and custom layouts are all first-class citizens here.</p> <p>Migrate your blog →</p>
---	--	---

Get up and running in seconds.

Quick-start Instructions

```
~ $ gem install jekyll
~ $ jekyll new my-awesome-site
~ $ cd my-awesome-site
~/my-awesome-site $ jekyll serve
# => Now browse to http://localhost:4000
```


Redmine



The screenshot shows a web browser window titled "Overview - Redmine" with the URL "www.redmine.org". The page has a blue header with the "Redmine" logo and navigation links: Home, Projects, Help, Sign in, and Register. Below the header is a search bar and a secondary navigation bar with links: Overview, Download, Activity, Roadmap, Issues, News, Wiki (selected), Forums, and Repository. The main content area is titled "Redmine" and includes a brief description, license information, and a list of features. A right sidebar contains sections for "Latest releases", "Resources", and "Wiki". At the bottom right, there is an advertisement for "planio" featuring a man pointing and a speech bubble that says "Love Redmine? Go Hosted!".

Overview - Redmine

www.redmine.org

Home Projects Help Sign in Register

Redmine

Search:

Overview Download Activity Roadmap Issues News Wiki Forums Repository

Redmine

History

Redmine is a flexible project management web application. Written using the Ruby on Rails framework, it is cross-platform and cross-database.

Redmine is open source and released under the terms of the [GNU General Public License v2 \(GPL\)](#).

Features

Some of the main features of Redmine are:

- Multiple projects support
- Flexible role based access control
- Flexible issue tracking system
- Gantt chart and calendar
- News, documents & files management
- Feeds & email notifications
- Per project wiki
- Per project forums
- Time tracking
- Custom fields for issues, time-entries, projects and users
- SCM integration (SVN, CVS, Git, Mercurial, Bazaar and Darcs)
- Issue creation via email
- Multiple LDAP authentication support
- User self-registration support
- Multilanguage support
- Multiple databases support

Read more about [Redmine features](#).

Documentation

You can read the [Redmine guide](#).

Latest releases


- 2.3.0 (2013-03-19)
- 2.2.4 (2013-03-19)

Resources

- [User's Guide](#)
- [Changelog, Security](#)
- [FAQ, HowTo's](#)
- [Plugins, Themes](#)

Wiki

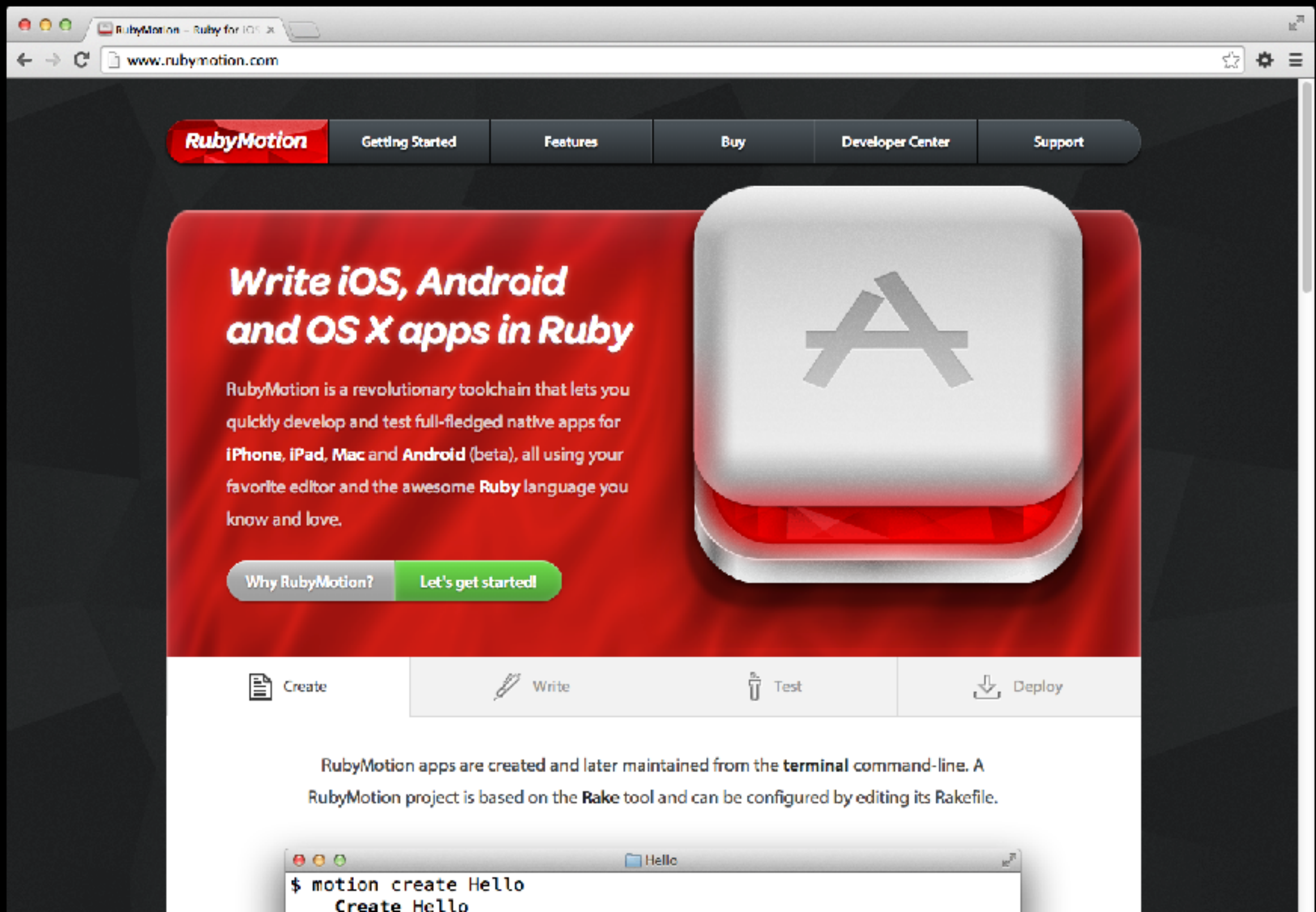
- [Start page](#)
- [Index by title](#)
- [Index by date](#)



planio

Love Redmine? Go Hosted!

RubyMotion



You can find more tools and libraries at:

[https://www.ruby-
toolbox.com/](https://www.ruby-toolbox.com/)

動手練習

Exercises

- 使用 Jekyll 產生靜態網頁
 - 並且部署到 GitHub Page 上
 - <http://jekyllrb.com/>
 - <https://pages.github.com/>

Thank you.

參考資料：

Beginning Ruby 2nd. (Apress)

Programming Ruby (The Pragmatic Programmers)

The Well-Grounded Rubyist (Manning)

Ruby 程式設計 (O'Reilly)

Foundation Rails 2 (friendsof)

<http://rubyonrails.org/ecosystem>

<http://infoether.com/ruby-and-rails-whitepaper>