INTRO TO RUBY

INTRO TO RUBY

Programmer Happiness

LANGUAGE OVERVIEW

RUBY

- Created by Matz* in 1995
- MINASWAN
- Created for "Programmer Happiness"

*Yukihiro Matsumoto

INSTALLATION

- OSX: Ruby comes pre-installed
- Windows: http://rubyinstaller.org

This course was written using Ruby version 2.2.2

EXECUTION

- Use Ruby to run a Ruby file. Ruby is interpreted, not compiled.
- Use interactive Ruby shell for testing Ruby Code.

EXECUTION

- Use Ruby to run a Ruby file. Ruby is interpreted, not compiled.

```
ruby file.rb
```

EXECUTION

- Use interactive Ruby shell for testing Ruby Code.


INTERACTIVE
EXERCISE!!!

TRY IT OUT

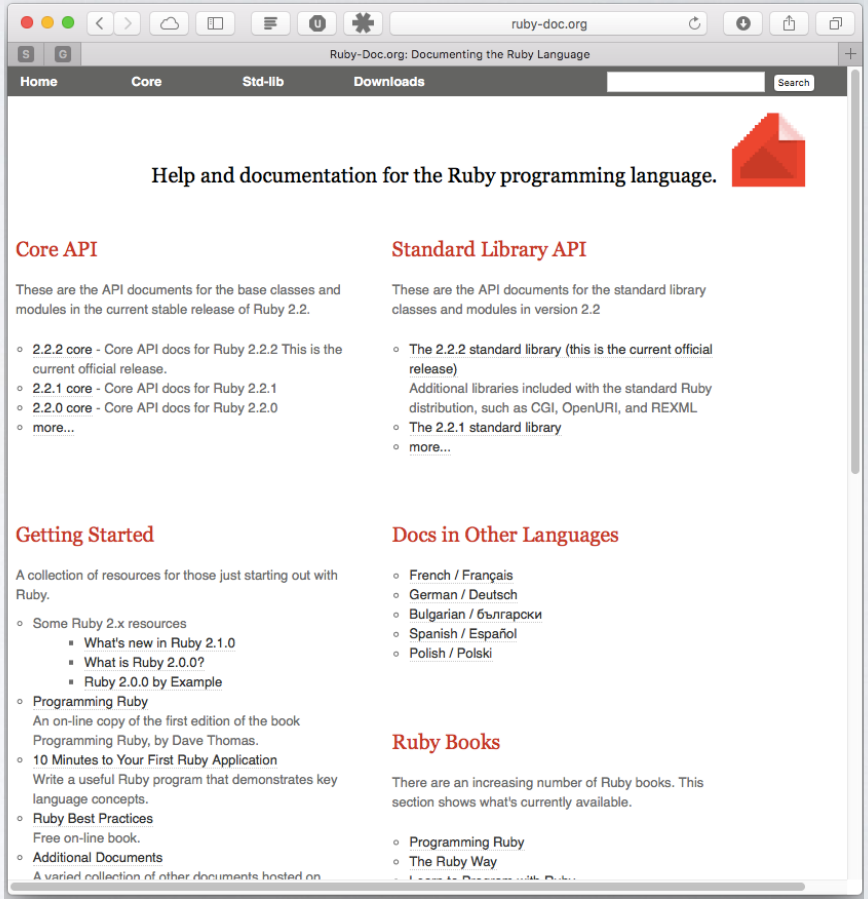
- In a file or irb, try the following code to output a hello world message

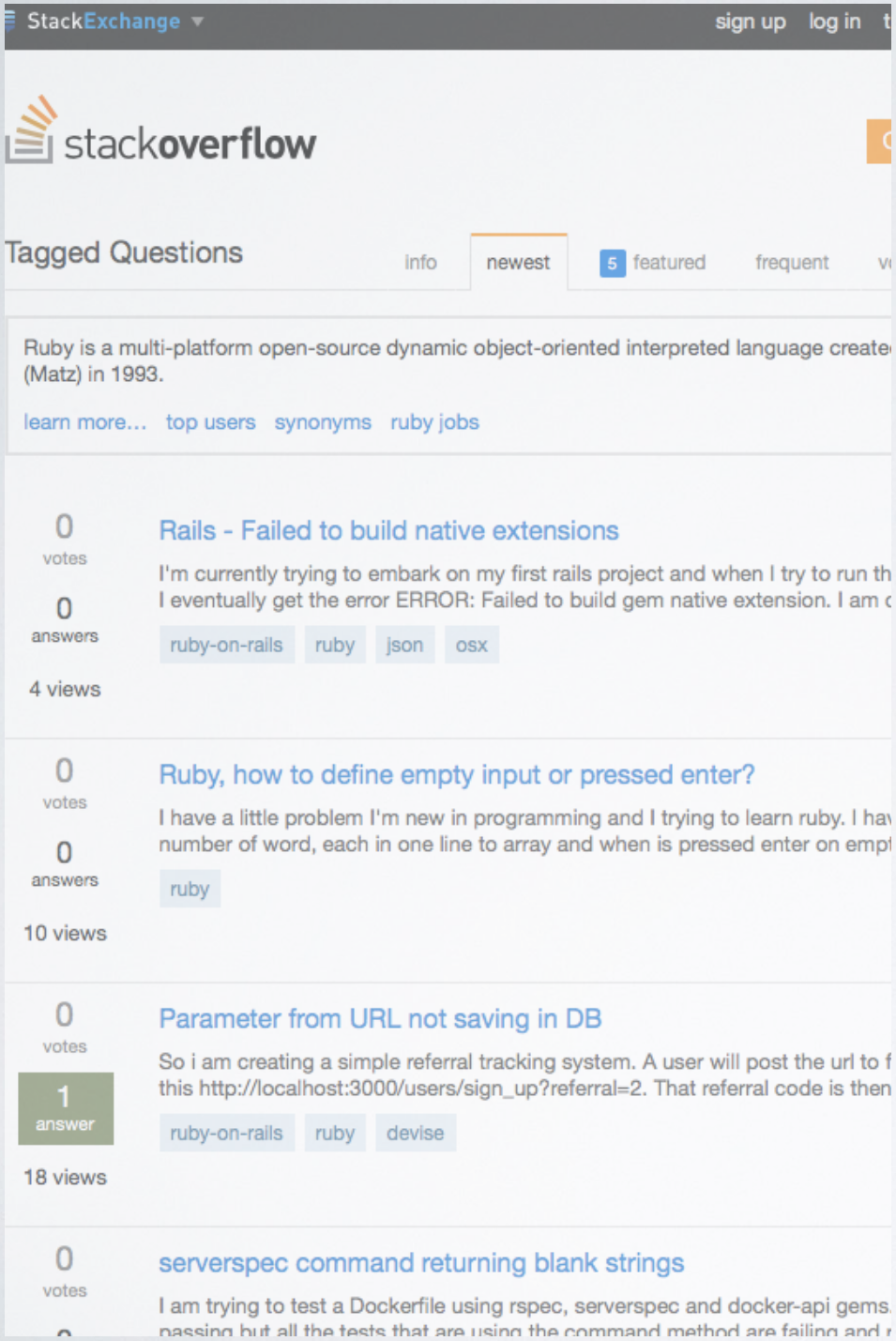
```
puts "Hello, world!"
```

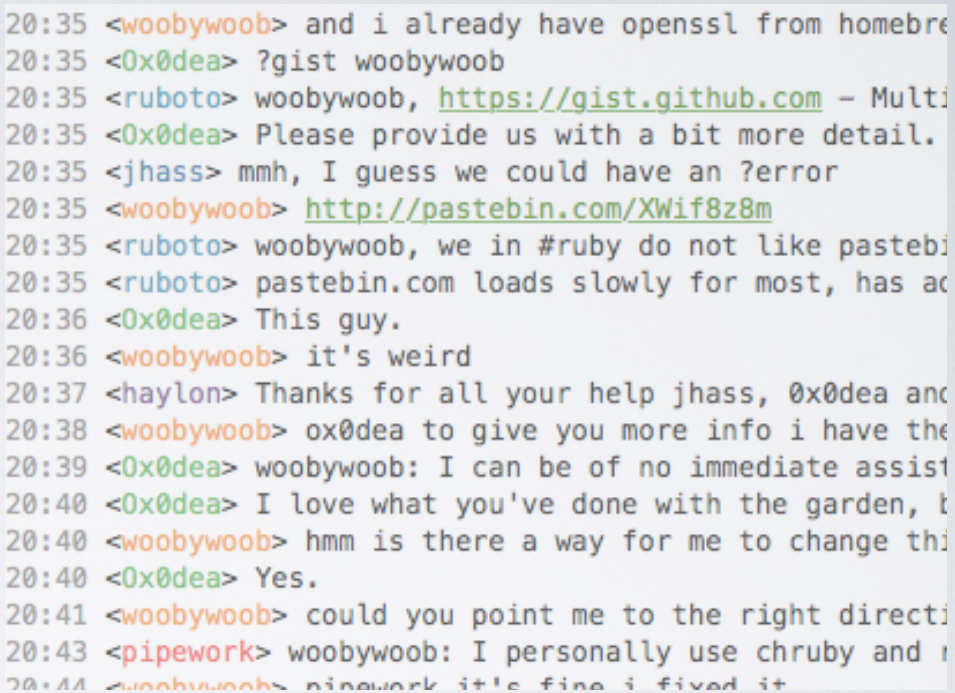
FINDING HELP

RUBY DOCS

- ruby-doc.org is fantastic
- Learn to browse modules



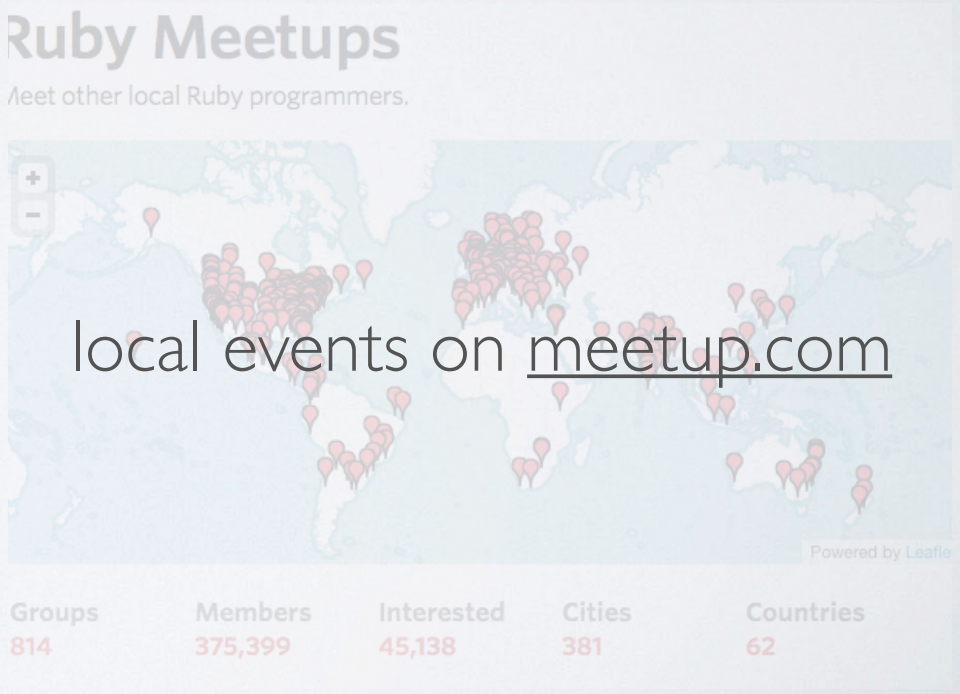

StackExchange
sign up
log in
stackoverflow
Tagged Questions
info
newest
5 featured
frequent
Ruby is a multi-platform open-source dynamic object-oriented interpreted language create
(Matz) in 1993.
learn more... top users synonyms ruby jobs
0 votes
0 answers
4 views
Rails - Failed to build native extensions
I'm currently trying to embark on my first rails project and when I try to run th
I eventually get the error ERROR: Failed to build gem native extension. I am
ruby-on-rails ruby json osx
0 votes
0 answers
10 views
Ruby, how to define empty input or pressed enter?
I have a little problem I'm new in programming and I trying to learn ruby. I hav
number of word, each in one line to array and when is pressed enter on empt
ruby
0 votes
1 answer
18 views
Parameter from URL not saving in DB
So i am creating a simple referral tracking system. A user will post the url to f
this http://localhost:3000/users/sign_up?referral=2. That referral code is then
ruby-on-rails ruby devise
0 votes
serverspec command returning blank strings
I am trying to test a Dockerfile using rspec, serverspec and docker-api gems.


20:35 <woobywoob> and i already have openssl from homebre
20:35 <0x0dea> ?gist woobywoob
20:35 <ruboto> woobywoob, https://gist.github.com - Multi
20:35 <0x0dea> Please provide us with a bit more detail.
20:35 <jhass> mmh, I guess we could have an ?error
20:35 <woobywoob> http://pastebin.com/XWif8z8m
20:35 <ruboto> woobywoob, we in #ruby do not like pastebi
20:35 <ruboto> pastebin.com loads slowly for most, has ac
20:36 <0x0dea> This guy.
20:36 <woobywoob> it's weird
20:37 <haylon> Thanks for all your help jhass, 0x0dea and
20:38 <woobywoob> ox0dea to give you more info i have the
20:39 <0x0dea> woobywoob: I can be of no immediate assist
20:40 <0x0dea> I love what you've done with the garden, b
20:40 <woobywoob> hmm is there a way for me to change thi
20:40 <0x0dea> Yes.
20:41 <woobywoob> could you point me to the right directi
20:43 <pipework> woobywoob: I personally use chruby and


Ruby Meetups
Meet other local Ruby programmers.
Powered by Leafle
Groups
814
Members
375,399
Interested
45,138
Cities
381
Countries
62

stackoverflow.com

irc #ruby on freenode



local events on meetup.com

OBJECT ORIENTED PROGRAMMING REFRESH


INTERACTIVE
EXERCISE!!!

SHOW OF HANDS

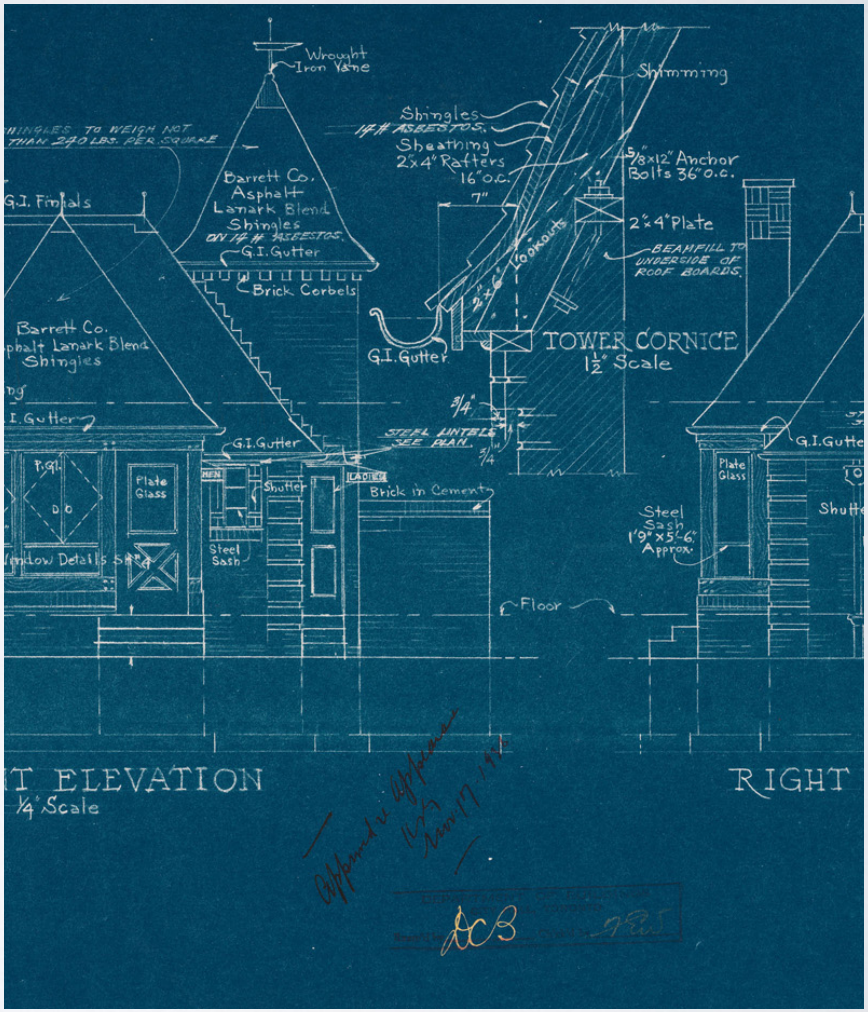
"I am familiar with object oriented programming"

"I use object oriented programming every day"

REFRESHER

CLASSES

- Blueprints that define how objects will be built
- Represent real things
- Have assignable attributes
- Can inherit functionality from other classes



OBJECT INSTANCES

- A single implementation of a class
- Can do things based on their class
- Have assigned attributes

KEY TO REMEMBER

- Class vs Instance
- Inheritance

BASIC CONCEPTS

EVERYTHING IS AN OBJECT

- Everything in Ruby is an object represented by some class
- Ruby has no primitives
- Can ask what the class is of anything with the method "class()"
- Except keywords and operators

EVERYTHING IS AN OBJECT DESCENDENT

- All objects in Ruby inherit from the base Object class
- Creates a limited amount of shared features across all objects
- Useful methods include: nil?, eq?, kind_of?, class, inspect

TYPES

- Ruby doesn't exactly have "types" as you'd expect in other programming languages
- Everything is an object of some kind
- Standard library classes in lieu of "types"

DUCK-TYPING

- Typed based on how an object acts rather than how it is defined

DUCK-TYPING

- "If it walks like a duck, and it quacks like a duck, then it is a duck"

EXAMPLE

```
class Duck
  def quack(self)
    puts("Quaaaaaack!")
  end
  def walk(self)
    puts("Waddles.")
  end
end

class Person
  def quack(self)
    puts("The person imitates a duck.")
  end
  def walk(self)
    puts("Walks normally.")
  end
  def name(self)
    puts("John Smith")
  end
end
```

```
def in_the_forest(duck)
    duck.walk()
    duck.quack()
end

def game
    donald = Duck()
    john = Person()
    in_the_forest(donald)
    in_the_forest(john)
end
```

DYNAMIC TYPING

- Ruby is typed dynamically
- Types of objects can be changed
- Types are inferred

```
greeting = "hello world"
greeting.class
# => String

greeting = 22
greeting.class
# => Fixnum
```

CORE CLASSES

ArgumentError
Array
BasicObject
Bignum
Binding
Class
Comparable
Complex
Complex::compatible
ConditionVariable
Continuation
Data
Dir
ENV
EOFError
Encoding
Encoding::CompatibilityError
Encoding::Converter
Encoding::ConverterNotFoundError
Encoding::InvalidByteSequenceError
Encoding::UndefinedConversionError
EncodingError
Enumerable
Enumerator
Enumerator::Generator
Enumerator::Lazy
Enumerator::Yielder
Errno
Exception
FalseClass
Fiber
FiberError
File
File::Constants
File::Stat
FileTest
Fixnum
Float
FloatDomainError
GC
GC::Profiler
Hash
IO
IO::EAGAINWaitReadable
IO::EAGAINWaitWritable
IO::EINPROGRESSWaitReadable
IO::EINPROGRESSWaitWritable
IO::EWOULDBLOCKWaitReadable
IO::EWOULDBLOCKWaitWritable
IO::WaitReadable
IO::WaitWritable
IOError
IndexError
Integer
Interrupt
Kernel
KeyError
LoadError
LocalJumpError
Marshal
MatchData
Math
Math::DomainError
Method
Module
Mutex
NameError
NilClass
NoMemoryError
NoMethodError
NotImplementedError
Numeric
Object
ObjectSpace
ObjectSpace::WeakMap
Proc
Process
Process::GID
Process::Status
Process::Sys
Process::UID
Process::Waiter
Queue
Random
Range
RangeError
Rational
Rational::compatible
Regexp
RegexpError
RubyVM
RubyVM::Env
RubyVM::InstructionSequence
RuntimeError
ScriptError
SecurityError
Signal
SignalException
SizedQueue
StandardError
StopIteration
String
Struct
Symbol
SyntaxError
SystemCallError
SystemExit
SystemStackError
Thread
Thread::Backtrace::Location
ThreadError
ThreadGroup
Time
TracePoint
TrueClass
TypeError
UnboundMethod
UncaughtThrowError
ZeroDivisionError
fatal
unknown

THE "TYPE" CLASSES

ArgumentError
Array
BasicObject
Bignum
Binding
Class
Comparable
Complex
Complex::compatible
ConditionVariable
Continuation
Data
Dir
ENV
EOFError
Encoding
Encoding::CompatibilityError
Encoding::Converter
Encoding::ConverterNotFoundError
Encoding::InvalidByteSequenceError
Encoding::UndefinedConversionError
EncodingError
Enumerable
Enumerator
Enumerator::Generator
Enumerator::Lazy
Enumerator::Yielder
Errno
Exception
FalseClass
Fiber
FiberError
File
File::Constants
File::Stat
FileTest
Fixnum
Float
FloatDomainError
GC
GC::Profiler
Hash

IO
IO::EAGAINWaitReadable
IO::EAGAINWaitWritable
IO::EINPROGRESSWaitReadable
IO::EINPROGRESSWaitWritable
IO::EWOULDBLOCKWaitReadable
IO::EWOULDBLOCKWaitWritable
IO::WaitReadable
IO::WaitWritable
IOError
IndexError
Integer
Interrupt
Kernel
KeyError
LoadError
LocalJumpError
Marshal
MatchData
Math
Math::DomainError
Method
Module
Mutex
NameError
NilClass
NoMemoryError
NoMethodError
NotImplementedError
Numeric
Object
ObjectSpace
ObjectSpace::WeakMap
Proc
Process
Process::GID
Process::Status
Process::Sys
Process::UID
Process::Waiter
Queue
Random

Range
RangeError
Rational
Rational::compatible
Regexp
RegexpError
RubyVM
RubyVM::Env
RubyVM::InstructionSequence
RuntimeError
ScriptError
SecurityError
Signal
SignalException
SizedQueue
StandardError
StopIteration
String
Struct
Symbol
SyntaxError
SystemCallError
SystemExit
SystemStackError
Thread
Thread::Backtrace::Location
ThreadError
ThreadGroup
Time
TracePoint
TrueClass
TypeError
UnboundMethod
UncaughtThrowError
ZeroDivisionError
fatal
unknown

CLASSES YOU SHOULD KNOW RIGHT NOW

```
null    = nil
boolean = true
string  = "Hello"
integer = 1
float   = 1.5
array   = [1,2,3]
hash    = {a: 1, b: 2}
symbol  = :symbol
```

NIL

- The absence of any value
- Value of any variable that hasn't been set or has been cleared
- Identified by the keyword "nil" representing a NilClass object

```
nil
```

BOOLEAN

- Used to represent truthiness
- Only two values
- Identified by the keywords "true" and "false" representing TrueClass and FalseClass objects

```
true
```

```
false
```

STRING

- Used to store text
- Identified by either single or double quotes
- Escape quotes with \

```
"Justin"

'Hello world'

'Some other phrase'

'Wubuluba dub dub'

'Justin\'s class'
```

INTEGER AND FLOAT

- Integers are any whole number
- Floats are any number with a decimal

```
1
2
-3
4
15
1.2
```


8
-3
166
3
70
37
2
21
.01
99
44
.33
41

ARRAY

- Used to store a list of items with no identifiers
- Identified by square brackets [ ]
- Delimited with commas
- Can store anything
- Can store mixed types

```
[2,3,5,7,11,13,17,19,23,29,
31,37,41,43,47]

['Bill','Ted']

['Romeo', 'Tybalt', 4, 3]
```

HASH

- Key value store
- Some languages call this a dictionary
- Good for lists of items with an identifier
- Keys and values can be any class, but are typically symbols
- Comma delimited
- Identified by curly brackets { }

```
{
title: "Back to the Future",
description: "Time travel movie",
price: 2900
}

{
"Bob's Date" => "Susan",
"Joanne's Date" => "Rufio"
}

{
1 => "b",
2 => "c"
}
```

SYMBOL

- Identifiers
- Called "keywords" and "atoms" in some languages
- Value is its name
- Identified by starting with a colon

```
:input_available
```


INTERACTIVE
EXERCISE!!!

DISCUSSION

WHICH CLASS TO USE?

- Name of a book
- Price of a book
- Person's age
- List of quotes and their speaker
- List of prime numbers
- Attributes about a person
- Customer loan eligibility

USING THESE CLASSES

VARIABLE NAMING

- Ruby style is to name variables with snake_case
- start with a lower case letter and separate words with underscores
- Constants start with a capital letter

```
name = "Justin"

age_today = 28

AGE_AT_BIRTH = 0
```

RUBY KEYWORDS

Avoid using these for naming things in your code

```
__ENCODING__
__LINE__
__FILE__
BEGIN
END
alias
and
begin
break
case
class
def
defined?
do
else
elsif
end
ensure
false
for
if
in
module
next
nil
not
or
redo
rescue
retry
return
self
super
then
true
undef
unless
until
when
while
yield
```

HASH ACCESS

```
bob = {name: 'Bob', age: 22}
bobs_age = bob[:age]

sarah = {"Name" => "Sarah", "Age" => 22}
sarahs_age = sarah["Age"]
```

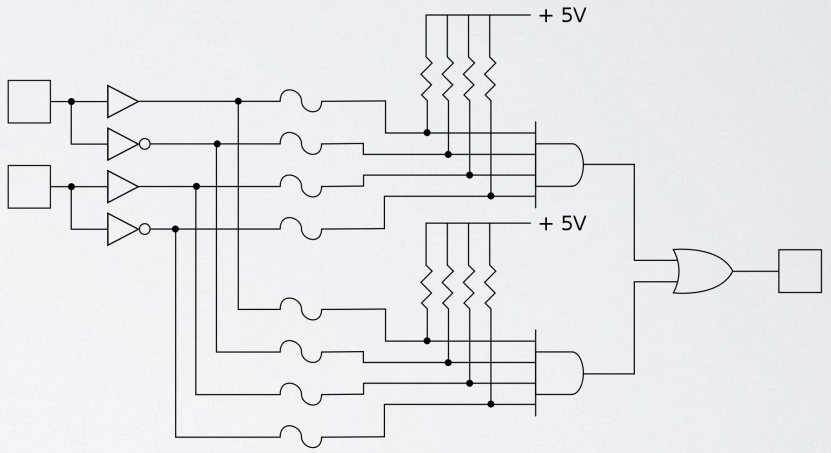
ARRAY ACCESS

```
prime_numbers = [2, 3, 5, 7, 11, 13, 17, 19, 23,
                 29, 31, 37, 41, 43, 47, 53, 59, 61,
                 67, 71, 73, 79, 83, 89, 97]


fifth_prime = prime_numbers[4]
```

LOGIC

LOGICAL DECISIONS

- How to control the flow through your Ruby program



Simplified programmable logic device

WHILE

```
while true
  puts "I'm in an infinite loop!"
end
```

UNTIL

```
until false
  puts "Will this never end?"
end
```

IF

```
puts "foo"
puts "bar"

if false
  puts "baz"
end
```

ELSE

```
puts "foo"
puts "bar"

if false
  puts "baz"
else
  puts "qux"
end
```

ELSIF

```
puts "foo"
puts "bar"

if false
  puts "baz"
elsif false
  puts "qux"
else
  puts "norf"
end
```

IF CONDITIONS

```
puts "foo"
puts "bar"
puts_baz = false

if puts_baz
  puts "baz"
end
```

IF CONDITIONS

```
puts "foo"
puts "bar"
should_i_put_baz = "no"

if should_i_put_baz == "yes"
  puts "baz"
end
```

IF CONDITIONS

```
puts "foo"
puts "bar"

if 2 + 2 == 5
  puts "baz"
end
```

SHORTCUT

If can go at the end of a line

```
puts "baz" if 2 + 2 == 5
```

ALTERNATIVE

Unless is the opposite of if.
Written the same way.

```
puts "baz" unless 2 + 2 == 5
```

CASE STATEMENT

```
x = 10

case x
when 1,2,3
  puts "1, 2, or 3"
when 10
  puts "10"
else
  puts "Some other number"
end
```

COMBINING CONDITIONS

- Most times you'll need to make decisions based on complex criteria
- You can logically combine conditions with "and", "or", and "not"

```
and
or
not

&&
||
!
```

LOGICAL OPERATORS

```
yes = true
no = false

if yes
  puts "baz"
end

if yes and yes
  puts "buz"
end

if no or yes
  puts "biz"
end
```

LOGICAL OPERATORS

```
yes = true
no = false

if yes
  puts "baz"
end

if yes && yes
  puts "buz"
end

if no || yes
  puts "biz"
end
```

LOGICAL OPERATORS

```
yes = true
no = false

if not yes
  puts "baz"
end

if !yes
  puts "buz"
end

if !(no or yes)
  puts "biz"
end
```


INTERACTIVE
EXERCISE!!!

CODING

SHORT EXERCISE

- Assign variables for a first and last name of a person
- If the person's first name is the same as your name, output their last name
- If it isn't your first name, output "not me"

NAVIGATING COLLECTIONS

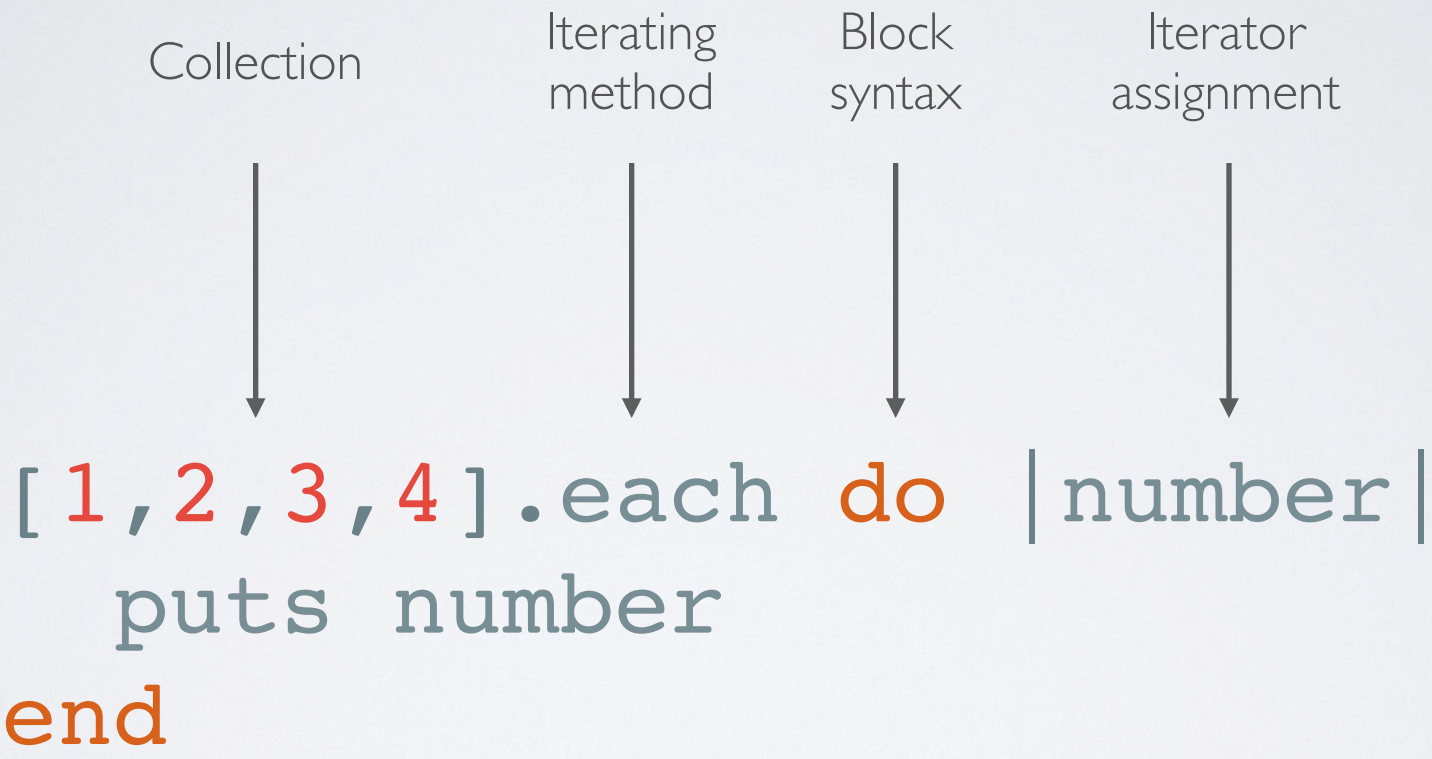
- Any Enumerable object including Hash and Array objects will need to be acted on
- Let's talk about ways to navigate through collections

EACH

- Most Rubyists use the "each" method from Enumerable
- There are others, like "for", but they aren't used often

```
[1,2,3,4].each do |number|
  puts number
end
```

WHAT'S HAPPENING



SPECIAL ACTIONS

You can skip an iteration

```
[1,2,3,4].each do |number|
  next if number == 2
  puts number
end
```

SPECIAL ACTIONS

You can stop iterating

```
[1,2,3,4].each do |number|
  break if number == 2
  puts number
end
```

SPECIAL ACTIONS

You can start over

```
[1,2,3,4].each do |number|
  retry if number == 2
  puts number
end
```


INTERACTIVE
EXERCISE!!!

CODING

FIZZ BUZZ

- You now have the tools to make a basic Ruby program
- Let's write fizz buzz
  - For every number 1 through 100, output that number.
  - Unless the number is divisible by 3. Then put "Fizz".
  - Unless the number is divisible by 5. Then put "Buzz"
  - Unless the number is divisible by 5 and 3. Then put "FizzBuzz"
  - Hint: look at the Integer documentation page for easier ways to get 1 through 100

ENCAPSULATION

METHODS

Store reusable code in methods

```
def greet
  puts "hello"
end

def greet(person)
  puts "hello"+person
end
```

METHODS

Call methods by their signature

```
def greet
  puts "hello"
end

def say_hi(name, other_thing)
  puts "hi"+name
end

greet
say_hi('Justin', false)
```

MODULES

Store similar methods in modules

```
module Greetings
  def greet
    puts "hello"
  end

  def greet(person)
    puts "hello"+person
  end
end
```

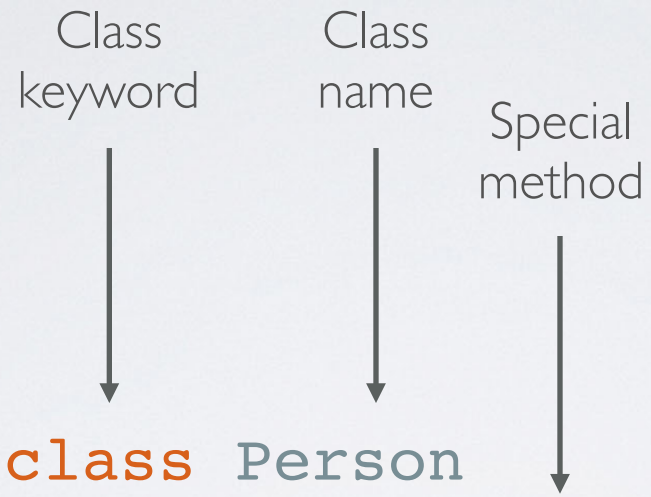
OBJECT-ORIENTED RUBY

CLASS DEFINITION

- Define the blueprint of an object
- Has an initializer
- Instances created by calling .new on the class
- Named with CamelCase

```
class Person
  def initialize
    puts "a new instance!"
  end
end

bob = Person.new()
```

CLASS DEFINITION




```
class Person
  def initialize
    puts "a new instance!"
  end
end

bob = Person.new()
```

WRITING CLASSES

```ruby
class Person
  attr_accessor :first_name, :last_name

  def initialize
    puts "a new instance!"
  end

  def self.about
    "A person is a human"
  end

  def full_name
    @first_name + @last_name
  end

  private
    def deepest_secret
      "Loves Taylor Swift"
    end
end
```

WRITING CLASSES

Attributes →

Initializer →

Class method →

Instance method →

Start of private methods →

```
class Person
attr_accessor :first_name, :last_name

  def initialize
    puts "a new instance!"
  end

  def self.about
    "A person is a human"
  end

  def full_name
    @first_name + @last_name
  end

  private
    def deepest_secret
      "Loves Taylor Swift"
    end
end
```

CLASS & INSTANCE METHODS

- Class methods are called directly on the class
- Instance methods are called on any instance of the class

CLASS & INSTANCE EXAMPLE

```ruby
class Person
  attr_accessor :first_name, :last_name

  def initialize
    puts "a new instance!"
  end

  def self.about
    "A person is a human"
  end

  def full_name
    @first_name + @last_name
  end

  private
    def deepest_secret
      "Loves Taylor Swift"
    end
end
```

```ruby
puts Person.about

joe = Person.new

puts joe.full_name
```

METHOD PRIVACY

- Methods are public by default
- Can be made private or protected by defining them below the keyword "private" or "protected"


INTERACTIVE
EXERCISE!!!

CODING

BLOG POST MODEL

- Write a Ruby class that describes a blog post

INHERITANCE

- Classes can inherit functionality from other classes
- Inherit all methods and attributes of the parent class
- Denoted with a <

```ruby
class Person
  attr_accessor :first_name, :last_name

  def full_name
    @first_name + @last_name
  end
end

class Teacher < Person
  def lesson_plan
    "Teach some Ruby"
  end
end
```

OVERRIDING

Methods in child classes can override methods in parent class

```
class Person
  attr_accessor :first_name, :last_name

  def full_name
    @first_name + @last_name
  end
end

class Teacher < Person
  def full_name
    "Professor"+@first_name+@last_name
  end
end
```

Same signature

Same signature

MONKEY PATCHING

- Another way to add functionality to a class is to "monkey patch" it
- This means to re-open a class and add new things to it

MONKEY PATCH A STRING

```
class String
  def inception
    chars.map{|char| char.upcase+'  '}.join.chomp
  end
end

"hello".inception
# => "H  E  L  L  O  "
```

MONKEY PATCHING

- Should be used sparingly
- Leads to unexpected behavior for other developers
- Pollutes namespaces

REFINEMENTS

- Refinements are an experimental feature to make monkey patching better
- Won't cover how-to in the material - know it's there


INTERACTIVE
EXERCISE!!!

CODING

BLOG POST MODEL

- Write a Ruby class that describes a blog post with comments and an author name
- Write another Ruby class that describes an anonymous blog post where comments are disabled and the author name is obscured

WRITING "GOOD" CODE

MODULES

- Used to encapsulate code
- Separate responsibilities into modules
- Each module in its own file
- Can be mixed into classes to add functionality

REQUIRING MODULES

In the same file

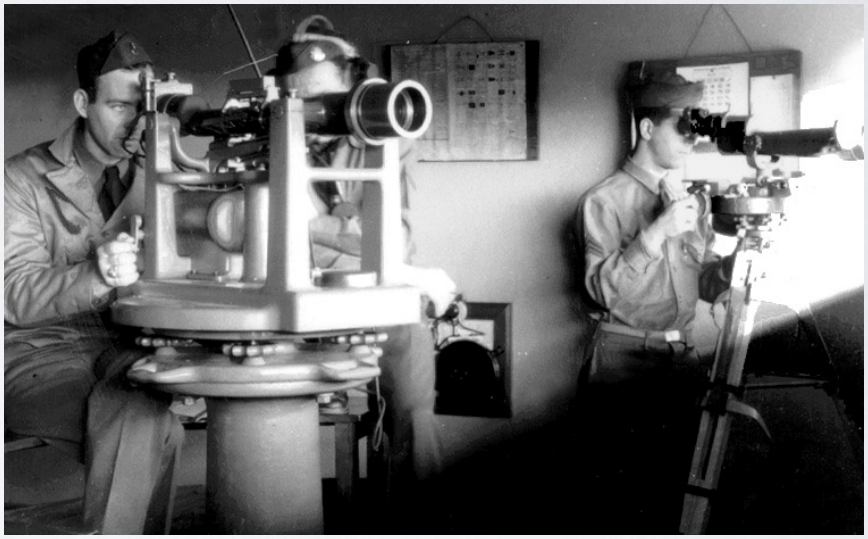
```
module Greetings
  def hello
    puts "hello"
  end
end

include Greetings
```

Require in
a separate file

```
require 'greetings'

include Greetings
```

SCOPE

- Think about scope of variables. Only make variables available where you NEED them.
- Local variables are available within the module or method where they are defined.
- Instance variables, starting with an "@", will be available for the lifetime of an instance
- Class variables, starting with "@@", and constants are accessible anywhere that class is accessible



SCOPE EXAMPLES

```
module Greetings
  default_greeting = "hello"
  def say_hi
    puts default_greeting
  end
end

puts default_greeting
# Nope
```

SCOPE EXAMPLES

```ruby
class Person
  def set_name
    @name = "paul"
  end

  def say_hi
    puts "hi"+@name
  end
end

p = Person.new
p.set_name
p.say_hi
# hi paul
```

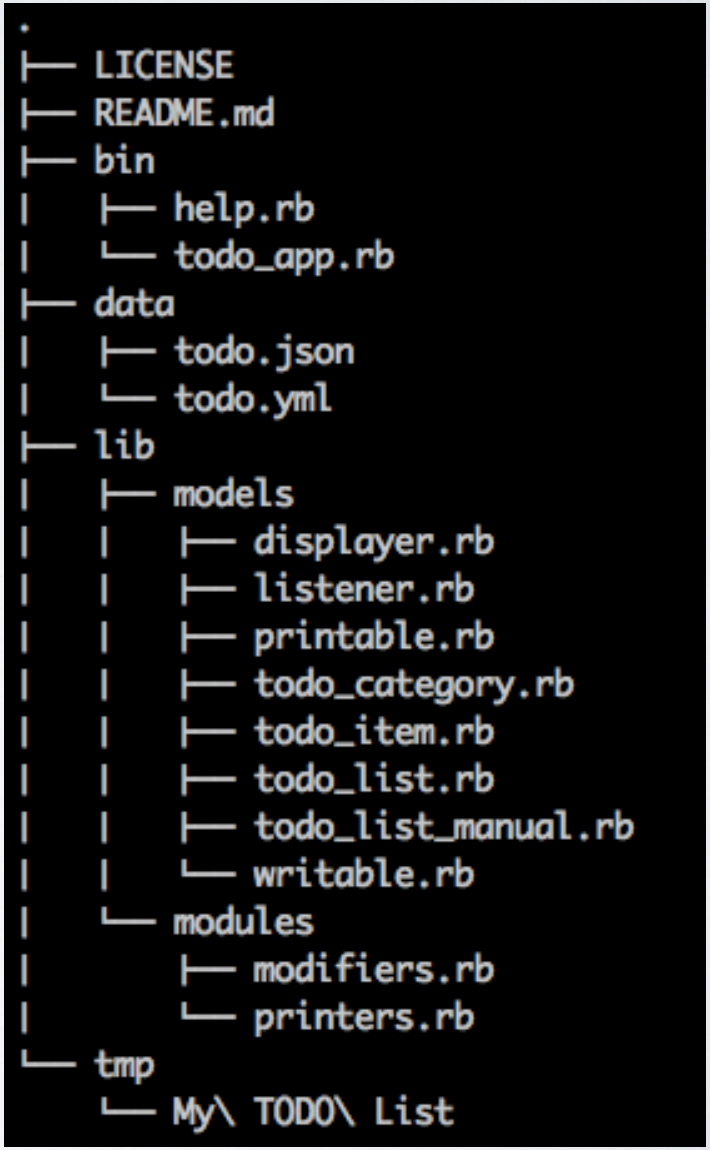
PROJECT STRUCTURE

ANYTHING

- Anything will work, but it's up to you and your team to follow some standards

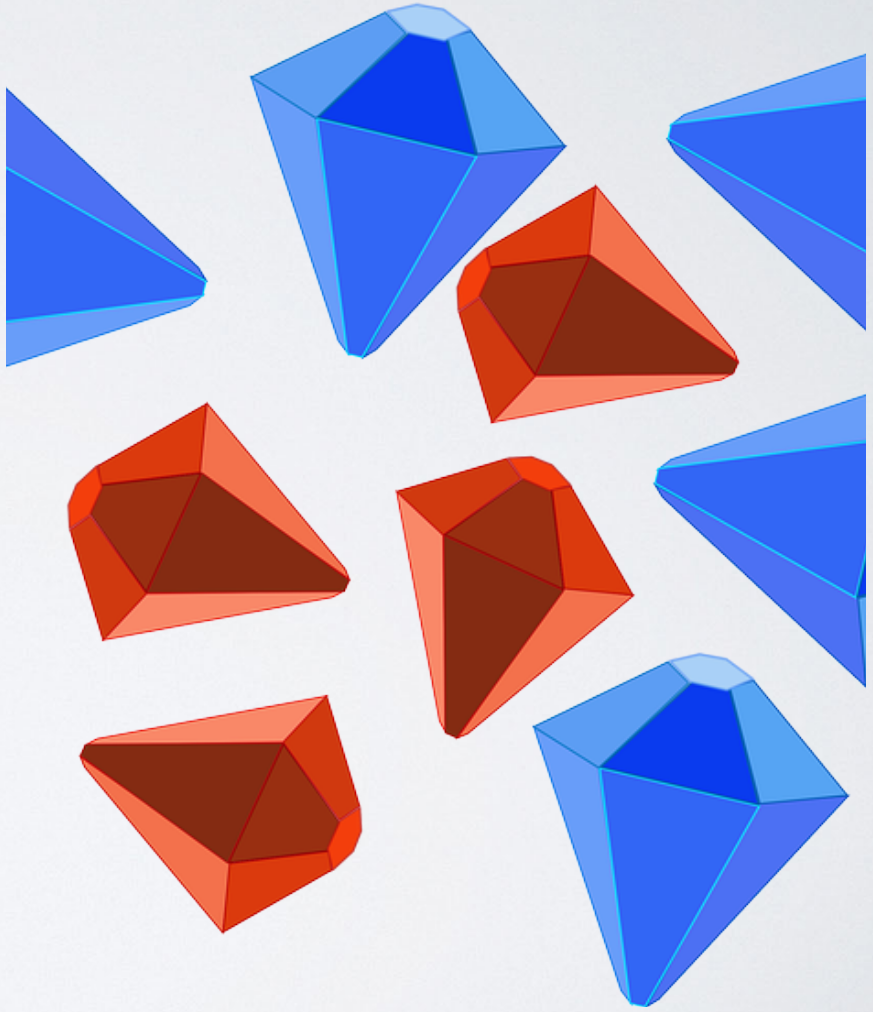
DIRECTORY STRUCTURE

- Prescribed, not dictated
- bin for executables
- data for application data fixtures
- lib for business logic
- spec for tests and specifications



.RUBY-VERSION

- A file dedicated just to maintaining the version of Ruby the project is built for
- Can be read by most version managers: rvm, chruby, rbenv, rbfu
- Common Ruby versions
  - MRI (Matz' Ruby Implementation - the default. Shown as numbered versions: 1.9.2, 1.8.7, 2.2.2, etc)
  - IronRuby (.Net under the hood)
  - jRuby (Ruby on the JVM)



RAKEFILE

- "Ruby Make"
- Rakefile used to define tasks that Rake will know about
- Great way to codify and run tasks

GEMFILE

- Gemfile and Gemfile.lock are used for managing project dependencies with Bundler
- Create one with the command "bundler init"

RUBY GEMS

RUBY GEMS

- External libraries to a project are called "gems"
- Gems can be required in the Gemfile and installed with Bundler
- Once installed, parts of the Gem can be required throughout the project

SYSTEM LEVEL GEMS

- Ideally only Bundler is installed at a system level
- Other Gems are managed by Bundler on a project level

GEMFILE EXAMPLE

```
source 'https://rubygems.org'

gem 'rails', '4.2.1'
gem 'omniauth-facebook'
gem 'active_model_serializers', '~> 0.9.2'

group :development, :test do
  gem 'rspec-rails'
end

group :test do
  gem 'shoulda-matchers'
  gem 'database_cleaner'
  gem 'turnip'
end

group :development do
  gem 'pry-rails'
end
```

GEMFILE EXAMPLE

Server →

```
source 'https://rubygems.org'
```

Gem with version →

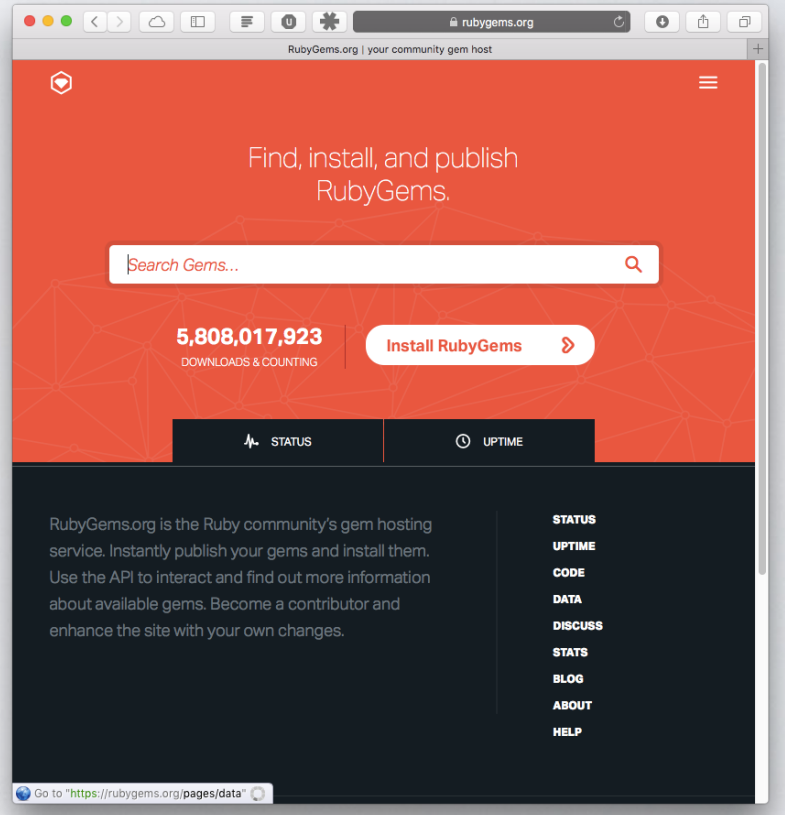
```
gem 'rails', '4.2.1'
gem 'omniauth-facebook'
gem 'active_model_serializers', '~> 0.9.2'
```

Only loaded sometimes →

```
group :development, :test do
  gem 'rspec-rails'
end

group :test do
  gem 'shoulda-matchers'
  gem 'database_cleaner'
  gem 'turnip'
end

group :development do
  gem 'pry-rails'
end
```

RUBYGEMS.ORG

- Centralized and public Gem repository
- Default server used by Bundler and RubyGems
- Searchable



USING BUNDLER

Once you have a Gemfile, you can manage the project's installed Gems with bundler's commands

```
bundle install
# Install project dependencies

bundle exec rake
# Run rake in the context of the bundle

bundle update
# Update all gems to latest version

bundle update gem_name
# Update a single gem
```

SPECIAL PURPOSE TOOLS

REGULAR EXPRESSIONS

REGEX IN RUBY

- Match patterns in text using common language
- http://rubular.com is a great resource for testing your syntax
- Surrounded by slashes /
- Matched with =~

```
/my regex/ =~ "Some phrase"
#nil (didn't find expression)

/phrase/ =~ "Some phrase"
#5 (regex found at index 5)

/phrase/.match("some phrase").pre_match
# "some "
```

CAPTURE GROUPS

```
m = /(.)(.)(\d+)(\d)/.match("THX1138.")
m             #=> #<MatchData "HX1138" 1:"H" 2:"X" 3:
                "113" 4:"8">
m[0]          #=> "HX1138"
m[1, 2]       #=> ["H", "X"]
m[1..3]       #=> ["H", "X", "113"]
m[-3, 2]      #=> ["X", "113"]

m = /(?<foo>a+)b/.match("ccaaab")
m             #=> #<MatchData "aaab" foo:"aaa">
m["foo"]      #=> "aaa"
m[:foo]       #=> "aaa"
```

TIME CLASSES

TIME CLASS

```
Time.now
#=> 2015-05-18 18:53:13 -0400

Time.now.hour
#=> 18

irb(main):014:0> Time.now.monday?
#=> false

Time.now.strftime('Printed on %m/%d/%Y')
#=> "Printed on 08/18/2015"
```

foragoodstrftime.com

DATE AND DATETIME

Basically the same, but need to call require 'date' to get them

```
require 'date'
d = Date.parse('3rd Feb 2001')
#=> #<Date: 2001-02-03 ...>
d.year
#=> 2001
d.mon
#=> 2
d.mday
#=> 3
d.wday
#=> 6
d += 1
#=> #<Date: 2001-02-04 ...>
d.strftime('%a %d %b %Y')
#=> "Sun 04 Feb 2001"
```

```
require 'date'

d = DateTime.parse('3rd Feb 2001 04:05:06+03:30')
#=> #<DateTime: 2001-02-03T04:05:06+03:30 ...>
d.hour
#=> 4
d.min
#=> 5
d.sec
#=> 6
d.zone
 #=> "+03:30"

#=> #<DateTime: 2001-02-04%16:05:06+03:30 ...>
d = d.new_offset('+09:00')

#=> #<DateTime: 2001-02-04%21:35:06+09:00 ...>
d.strftime('%I:%M:%S %p')
#=> "09:35:06 PM"
```

METHODS

```
!
!=
!~
+
-
<
<=
<=>
==
===
=~
>
>=
__id__
__send__
asctime
between?
class
clone
ctime
day
define_singleton_meth
od
display
dst?
dup
enum_for
eql?
equal?
extend
freeze
friday?
frozen?
getgm
getlocal
getutc
gmt?
gmt_offset
gmtime
gmtoff
hash
hour
inspect
instance_eval
instance_exec
instance_of?
instance_variable_def
ined?
instance_variable_get
instance_variable_set
instance_variables
is_a?
isdst
kind_of?
localtime
mday
method
methods
min
mon
monday?
month
nil?
nsec
object_id
private_methods
protected_methods
public_method
public_methods
public_send
remove_instance_varia
ble
respond_to?
round
saturday?
sec
send
singleton_class
singleton_method
singleton_methods
strftime
subsec
succ
sunday?
taint
tainted?
tap
thursday?
to_a
to_enum
to_f
to_i
to_r
to_s
trust
tuesday?
tv_nsec
tv_sec
tv_usec
untaint
untrust
untrusted?
usec
utc
utc?
utc_offset
wday
wednesday?
yday
year
zone
```

INTERACTING WITH FILES

FILE CLASS

Ruby has a File class built into the standard library

```
File.exists?("/training_ruby/exercises/data/albums.md")
# => true

File.open("/training_ruby/exercises/data/albums.md")
# => #<File:
# /Users/justin/github/builderworksco/training_ruby/exercises/#
# data/albums.md>

File.open("/training_ruby/exercises/data/albums.md").read
#=> "Illinoise\nFuneral\nSuburbs\nModern vampires
# in the city\nBlack album\nRockin the
# suburbs\nYoshimi\nMore
# adventurous\nGraduation\n"

f = File.open("/training_ruby/exercises/data/albums.md")
# => #<File:/Users/justin/github/builderworksco/training_ruby/
# exercises/data/albums.md>
```

FINDING A FILE

- Use File.join to create file search string
- It will automatically escape directories correctly according to the system it is running on (Windows vs Unix flavored slashes)

```
File.join("dir1", "dir2", "file")
# => "dir1/dir2/file" on Mac
# => C:\dir1\dir2\file on Windows
```

OPENING

```
path = File.join("training_ruby","exercises","data","albums.md")
File.open(path)
```

GETTING INFO

```
path = File.join("training_ruby","exercises","data","albums.md")
File.birthtime(path)
# => 2015-08-16 16:58:30 -0400
```


INTERACTIVE
EXERCISE!!!

CODING

REGEX, FILES, TIME

- Open a file on your computer
- If it was created on a Friday output "Good job working hard on a Friday"
- If it wasn't created on a Friday, output "This file was created on a " and then the day of the week it was created
- Use Regex and the "read" method to see if your name is anywhere in the file

CREATING/WRITING FILES

```
f = File.new("file.txt", "w")
f.write("1234567890")
f.close
```

CREATING/WRITING FILES


INTERACTIVE
EXERCISE!!!

CODING

WRITING TO FILES

- Write the numbers 1 through 1024 into a file on your Desktop

LINE BY LINE

```
IO.readlines(".testfile")[0]
IO.readlines(".testfile")[1]
IO.readlines(".testfile")[2]
```

FILES IN DIRECTORY

Dir will provide an enumerable object of file names in a directory

```
Dir.new('/Users/justin/web_testing/').map(&:to_s)
#=> [".", "..", ".ruby-version", "Gemfile", "Gemfile.lock",
# "hello.html", "test.feature", "test.rb"]
```

INPUT/OUTPUT

GETS

Built in gets method will listen for input

```
puts "What do you want me to echo?"
input = gets
puts input
```

KEEP LISTENING

Will do something each time user presses enter.
Exit with ctrl-D

```
puts "What do you want me to echo?"
while input = gets
  puts input
end
```

DON'T WAIT FOR ENTER

```
require 'io/console'
while input = STDIN.getch
  do_something_with(input)
end
```


INTERACTIVE
EXERCISE!!!

CODING

ACCEPTING USER INPUT

- Play Rock, Paper, Scissor against your computer
- Do it in an object-oriented way
- Create classes that respond to:

```
User.throw
Game.start
Game.result
```

ERROR HANDLING

BUILT IN EXCEPTIONS

```
NoMemoryError
ScriptError
LoadError
NotImplementedError
SyntaxError
SignalException
Interrupt
StandardError # default for
rescue
ArgumentError
IndexError
StopIteration
IOError
EOFError
LocalJumpError
NameError
NoMethodError
RangeError
FloatDomainError
RegexpError
```

```
RuntimeError # default for raise
SecurityError
SystemCallError
Errno#::*
SystemStackError
ThreadError
TypeError
ZeroDivisionError
SystemExit
fatal # impossible to rescue
```

RESCUE

- Exceptions can be caught with rescue
- Rescue will know what type of exception it received and the backtrace

```
begin
  this code will error
rescue => e
  e.backtrace
end
```

RESCUE

- Can rescue based on type of exception
- Also can use 'ensure' to ensure a block always runs even if code ahead of it raised an exception
- 'retry' can be used to try begin block again

```
begin
  this code will error
rescue ZeroDivisionError => e
  puts "you tried to divide by zero!"
  retry
  e.backtrace
rescue NoMethodError
  puts "that method doesn't exists"
ensure
  puts "wrapping up"
end
```

MAKE YOUR OWN

- You can create your own exceptions and raise them
- Typically inherit from standard error

```
class OnlyFiveException < StandardError
  def initialize(input, msg = "We only
accept 5 here")
    msg += 'you put '+input
    super(msg)
  end
end

if user_input != '5'
  raise OnlyFiveException.new(user_input)
end
```

TESTING

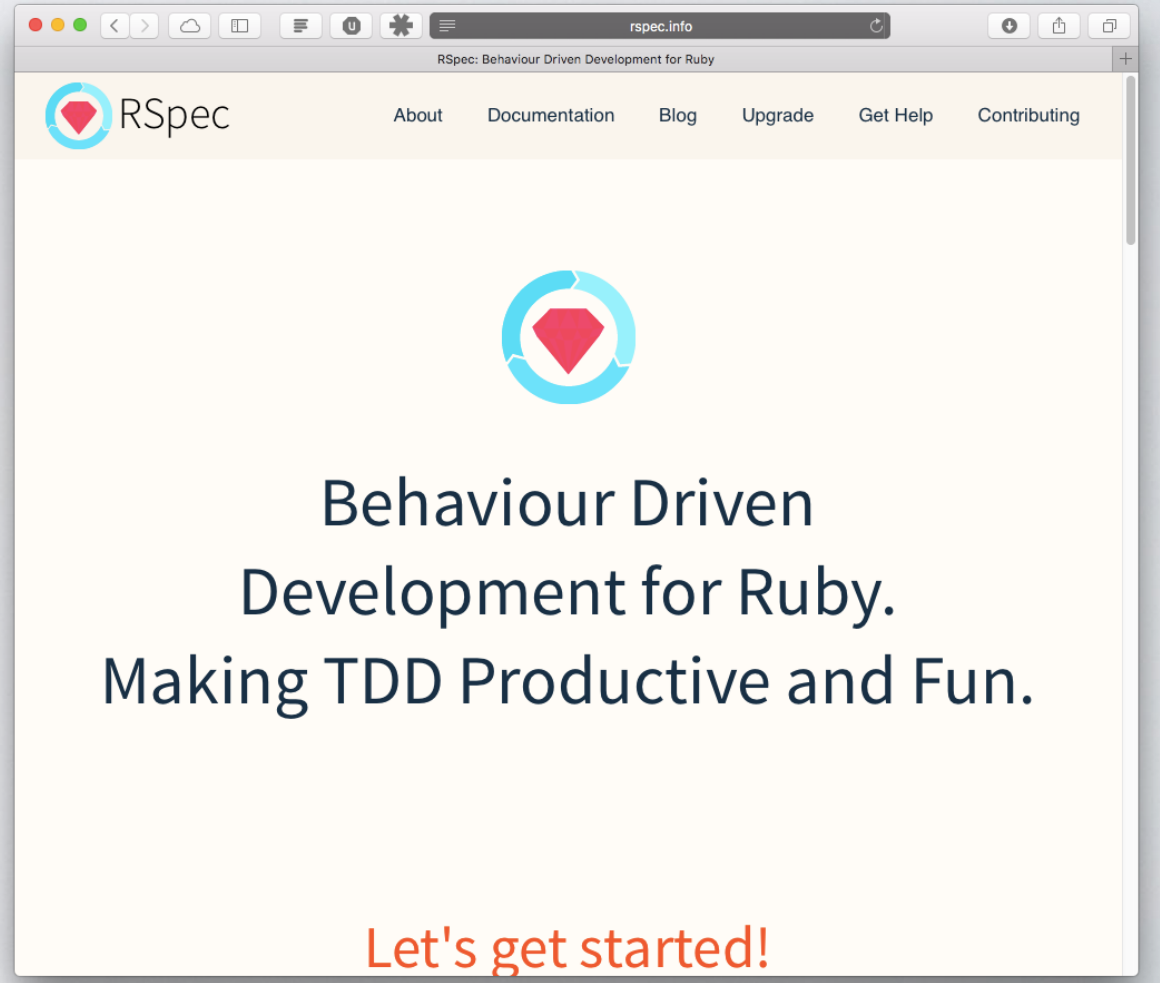
TEST::UNIT

- Built into the standard library
- Adds testing methods
- Require 'test/unit' module

```
require "test/unit"

class TestExample < Test::Unit::TestCase

  def test_simple
    assert_equal(4, 4)
    assert_equal(6, 6)
  end

end
```

RSPEC

- Popular alternative to Test::Unit
- gem 'rspec'



CUCUMBER

- Popular acceptance testing tool
- Translates Gherkin to Ruby code
- gem 'cucumber'

METAPROGRAMMING

PASSING CLOSURES

```
outer = 1

def m a_proc
  inner = 99
  a_proc.call(inner)
  puts "inner var = #{inner}"
  puts "argument is a #{a_proc.class}"
end

m proc {|inner| outer += inner}
# we can also use Proc.new instead of proc, with the same effect:
# m Proc.new {|inner| outer += inner}
puts "outer var = #{outer}"
```

LAMBDAS VS. PROCS

Basically, lambdas care about arguments

YIELD

```ruby
def calculation(a, b)
  yield(a, b)
end

puts calculation(5, 6) { |a, b| a + b } # addition
puts calculation(5, 6) { |a, b| a - b } # subtraction
```

DSL CREATION

```
module TestLang
  def self.blast blast_title, &block
    puts '*********'+blast_title+'*********'
    yield
    puts '*********'+blast_title+'*********'
  end
end

module TestLang
  blast 'HEYYY' do
    puts 'hello'
  end
end
```

METHOD MISSING

```
class Numeral
  def roman_to_int(str)
    # ...
  end

  def method_missing(methId)
    str = methId.id2name
    roman_to_int(str)
  end
end
```

DYNAMIC METHOD ADDING

```
class Numeral
  for(1..20) do |n|
    define_method('translate_'+n) do
      'The roman numeral for '+n+'hasn\'t been defined yet'
    end
  end
end

# same as
class Numeral
    def translate_1
      'The roman numeral for 1 hasn\'t been defined yet'
    end

    def translate_2
      'The roman numeral for 2 hasn\'t been defined yet'
    end
end
```

POPULAR FRAMEWORKS

SINATRA

- Lightweight micro framework for web applications

RAILS

- Framework for building web applications
- Strongly opinionated
- Gives you a lot out of the box

CHEF

- Systems integration framework

CHEF™

PUPPET

- Configuration management framework

CAPISTRANO

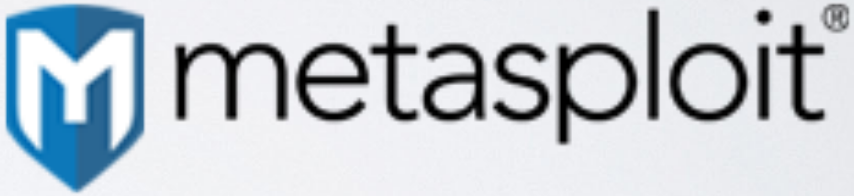
- Server automation and deployment

VAGRANT

- Virtual machine configuration

METASPLOIT

- Penetration testing framework



RUBY MOTION

- Write OSX, iOS and Android applications in Ruby

- All images licensed for commercial reuse with modification

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS