

# CS 5142

# Scripting Languages

10/28/2012

Ruby, Rails

# Outline

- Ruby
- Rails

# Methods, Blocks, Procs

- Declaration: `def id [ (arg*) ] ... end`
  - Always invoked on an object (a.k.a., the receiver)
  - `self` refers to the object on which the method was invoked
- Any invocation may be followed by a *block*
  - `yield` statements will invoke the block
  - `def mymethod(x) yield x end`
  - `mymethod(3) { |arg*| ... }`
- A block is represented by a `Proc` object
  - `p = Proc.new { |x| puts x }`
  - `def mymethod(b, x) b.call(x) end`

## Lambdas

- `lambda` is a method in the Kernel module that also creates a `Proc`
  - `l = lambda { |x| puts x }`
  - `l.call()`
- A proc is the object form of a block, behaves like a block
- A lambda behaves like a method
- Lambdas are closures
  - Binds the the variables in lexical scope where the lambda is defined (including self)

# Lambdas vs. Blocks

```
def proc_return
  p = Proc.new { return "Proc.new" }
  p.call
  return "proc_return method finished"
end
```

```
def lambda_return
  l = lambda { return "lambda" }
  l.call
  return "lambda_return method finished"
end
```

```
puts proc_return
puts lambda_return
```

## Using Objects

```
a1 = Apple.new(150, "green")
```

```
a2 = Apple.new(150, "green")
```

Constructor calls

```
a2.color= "red"
```

Setter call

```
puts a1.prepare("slice") + "\n"
```

```
puts a2.prepare("squeeze") + "\n"
```

Method calls

# Defining Classes

```
class Fruit
  def initialize(weight_)
    @weight = weight_ end
  def weight
    @weight end
  def weight= (value)
    @weight = value end
  def pluck
    "fruit(" + @weight + "g)" end
  def prepare(how)
    how + "d " + pluck end
end
```

Fruit
@weight
<u>initialize()</u> pluck() prepare()

- All fields are private, external use requires accessors (e.g., @weight, weight, weight=)
- Classes are open, can add additional fields+methods

# Class Definition Gotcha

```
class Fruit
  @weight = 0
  def initialize(weight_)
    @weight = weight_
  end
end
```

These two `@weight` variables are different!



- Doesn't behave as you'd expect
- One is a class variable
- The other is an instance variable



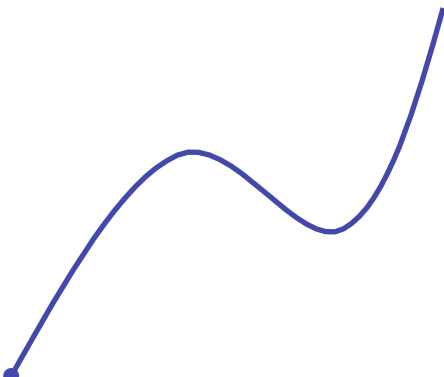
## Self

- Evaluates to the current object
- Remember that `class` is a function in Ruby
- When `class` is invoked, `self` points to the `Fruit` class, not an instance of the `Fruit` class
- When `initialize` is invoked, `self` points to the `Fruit` instance

# Self with Blocks

#<A:0x007fe42b1063c0>

```
class A
end
class B
  def initialize
    @a = A.new
  end
  def m
    @a.instance_eval { puts self }
  end
end
b = B.new
b.m
```

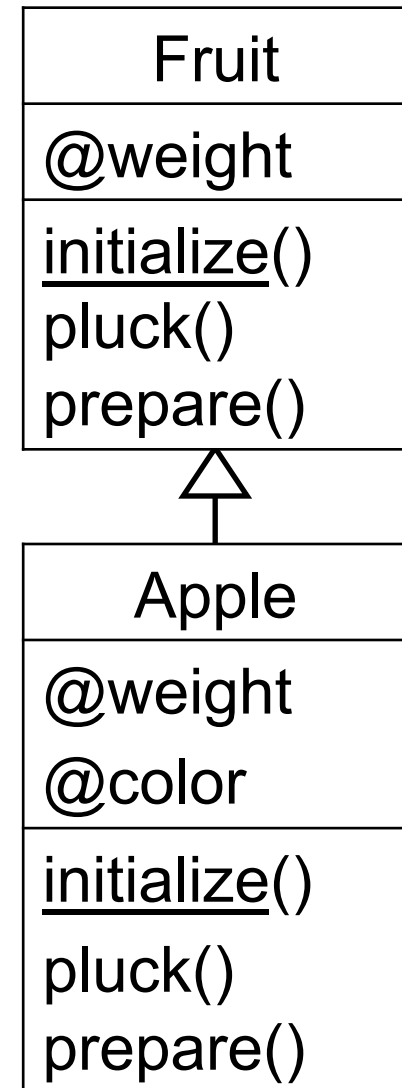


# Inheritance in Ruby

```
class Fruit
  def initialize(weight_)
    @weight = weight_ end
  def weight
    @weight end
  def weight= (value)
    @weight = value end
  def pluck
    "fruit(" + @weight + "g)" end
  def prepare(how)
    how + "d " + pluck end
end
```

---

```
class Apple < Fruit
  def initialize(weight_, color_)
    @weight = weight_
    @color = color_
  end
  def color
    @color end
  def color= (value)
    @color = value end
  def pluck
    self.color + " apple" end
end
```



## Scopes and Visibility

- Visibility of class members
  - All instance variables are private
  - Methods can be private, protected, or public

- Accessor generation

Generates `@weight` field  
and `weight/weight=` methods

```
class Fruit
  attr_accessor :weight
  def initialize(weight_)
    @weight = weight_
  end
  def pluck
    "fruit(" + @weight + "g)"
  end
  def prepare(how)
    how + "d " + pluck
  end
end
```

Fruit
@weight
<u>initialize()</u> pluck() prepare()

## Structure of a Ruby Application

- **require** *file*
- Module = like class, but can't be instantiated
  - Class can **include** (“mix in”) one or more modules
  - Members of mix-in module are copied into class
  - Later definition with same name overrides earlier
  - Module can inherit from other module, but not class
  - Module can contain methods, classes, modules
- Module **Kernel** is mixed into class **Object**
- Top-level subroutines are private instance methods of the Kernel module
  - Visible everywhere, can't call with explicit receiver

## Arrays

- Initialization: `$a = [1, 2, 3]`
  - With block: `$a = Array.new(10) { |e| 2*e }`
- Indexing: `$a[...]`
  - Zero-based, contiguous, integers only
  - Negative index counts from end
- Deleting: `$a.clear()`, `$a.compact()`, `$a.delete_at(i)`
- Lots of other methods

## Hashes

- Initialization:  
`$h = { 'lb' => 1, 'oz' => 16, 'g' => 453 }`
- Indexing: `$h[ 'lb' ]`
  - Can use any object as key, not just strings
- Deleting: `$h.clear()`, `$h.delete(k)`
- Lots of other methods
- Can have a “default closure”:  
return value for keys not explicitly stored

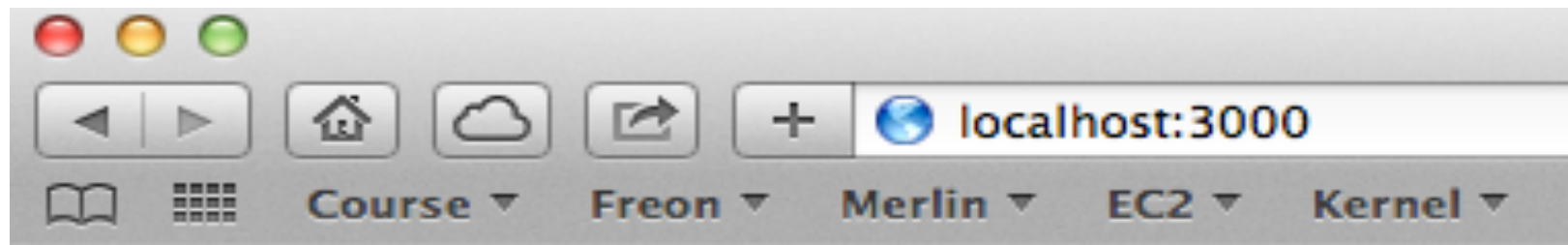
# Outline

- Ruby
- **Rails**



## Rails “Hello World”

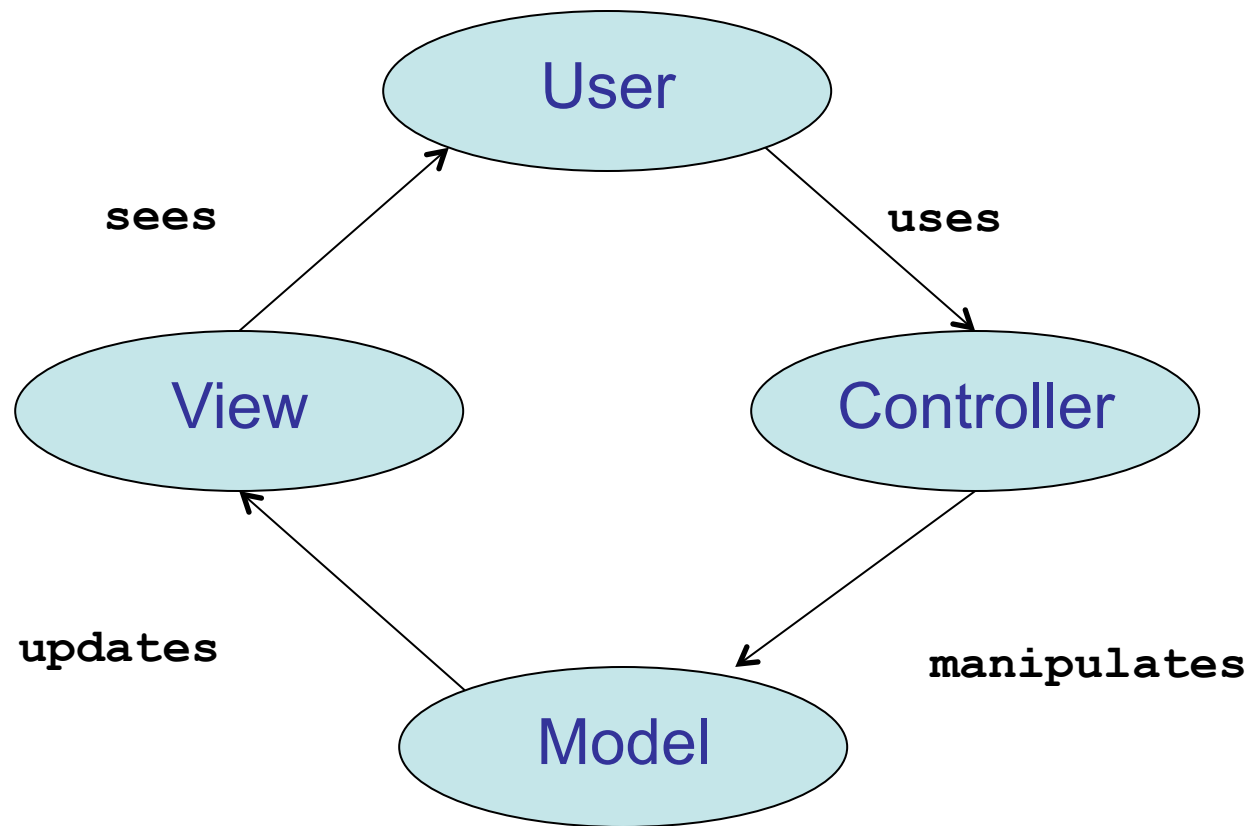
```
$ rails new Hello  
$ cd Hello  
$ rails server
```



# Welcome#index

Find me in `app/views/welcome/index.html.erb`

# Model View Controller

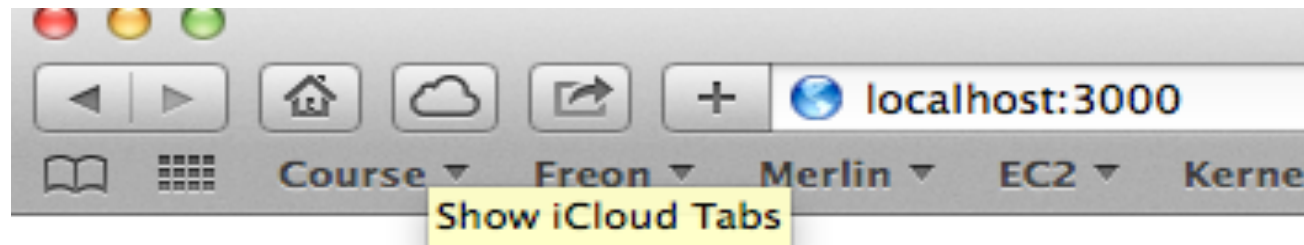


## Rails Routing

```
# Create a controller named welcome with  
# action index  
$ rails generate controller welcome index  
  
$ vi config/routes.rb  
# uncomment root to: "welcome#index"
```

## Modify View

```
$ echo "<p>Hello Class</p>" >> \
app/views/welcome/index.html.erb
```



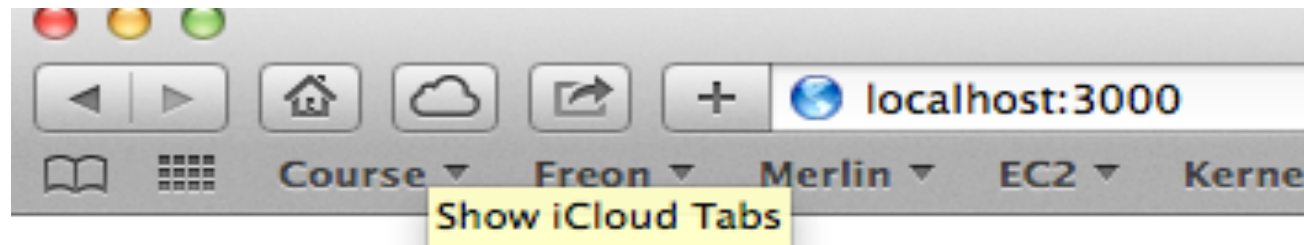
## Welcome#index

Find me in `app/views/welcome/index.html.erb`

Hello Class

## Modify View

```
$ echo "<p>Hello Class</p>" >> \
app/views/welcome/index.html.erb
```



## Welcome#index

Find me in `app/views/welcome/index.html.erb`

Hello Class

## Modify Controller

```
$ pwd  
~/rails/Hello/app/controllers
```

```
$ cat welcome_controller.rb  
class WelcomeController <  
  ApplicationController  
    def index  
    end  
end
```

# Ruby Documentation

- <http://www.ruby-lang.org>
- <http://www.rubyonrails.org>
- Book: The Ruby Programming Language.  
David Flanagan, Yukihiro Matsumoto.  
O' Reilly, 2008

# Evaluating Ruby

## Strengths

- Rails
- Purely object oriented
- Perl-like =~ and default variables

## Weaknesses

- Less popular than Java and PHP
- Unusual syntax



# Last Slide

- No announcements.
- Today' s lecture
  - Ruby