Ruby

Skylar Smith, Corey Finley, and Andrey Gavel
Problem Domain

- Balance Functional and Interpretive Programming Styles
- More Powerful than Perl
- More Object Oriented than Python
- Interpreted Language
  - Supports Many Platforms
Historical Context

- Lead Developer
  - Yukihiro Matsumoto

- Development Period
  - Mid-1990s

- Country of Origin
  - Japan

- Inspiration for Syntax
  - Perl
  - Smalltalk

- Also Influenced by
  - Eiffel
  - Lisp
Evolution

- Ruby 1.0 Released (1996)
- Ruby on Rails Released (2005)
  - Makes Ruby Very Popular
- Branches/Frameworks (Currently)
  - Ruby on Rails
    - Web Framework
  - JRuby
    - Integration into Java
  - IronRuby
    - Targeting .Net Framework
Concepts

- Everything is an Object
  - No Primitive Types
- Metaprogramming
  - Program Can Rewrite Itself
- Dynamic Typing
- Everything is *true*, Except *false* and *nil*
- Automatic Garbage Collection
- Centralized Package Management
  - RubyGems
Example 1 - Hello World

- `initialize` is used for creating a new object
- Methods begin with `def`
- Class variables prefixed by `@` symbol
- No need to declare type

```
# The Greeter class
class Greeter
  def initialize(name)
    @name = name.capitalize
  end

  def salute
    puts "Hello #{@name}!"
  end
end

# Create a new object
g = Greeter.new("world")
# Output "Hello World!"
g.salute
```
Example 2 - Flexibility

- Add custom method `plus` to built-in `Numeric` class
- Operators can also be overloaded and redefined

```ruby
class Numeric
  def plus(x)
    self.+((x)
  end
end

y = 5.plus 6  # y is now equal to 11
```
Example 3 - Collection Iteration

- Iterate over any collection with `.each` do |x|
- Inline array declaration with `[]`

```ruby
# define taxes class
class Taxes
  # set the tax rate
def initialize rate
    @rate = rate
  end

  # iterate through the collection with .each
  def add_rate collection
    collection.each do |cl|
      c = c*(1+@rate)
    end
  end
end

# create Taxes object with a 5% tax rate
t = Taxes.new(0.05)

# add the tax rate to a collection of taxes
transactions = [10.00, 15.32, 45.09]
t.add_rate transactions
```
<table>
<thead>
<tr>
<th>Ruby</th>
<th>Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpreted</td>
<td>Compiled to Bytecode</td>
</tr>
<tr>
<td>Dynamic Typing</td>
<td>Static Typing</td>
</tr>
<tr>
<td>Terse Syntax (Automatic Getters and Setters)</td>
<td>Verbose Syntax (Manually Write Getters and Setters)</td>
</tr>
</tbody>
</table>
# C++ Comparison

<table>
<thead>
<tr>
<th>Ruby</th>
<th>C++</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpreted</td>
<td>Compiled</td>
</tr>
<tr>
<td>Everything is an Object</td>
<td>Many Primitive Types</td>
</tr>
<tr>
<td>Automatic Garbage Collection</td>
<td>Manual Memory Management</td>
</tr>
</tbody>
</table>
Questions?