RubyCocoa & Max OSX

Bridging two worlds together

Ruby

- Interpreted dynamic language

- Modern garbage collected environment

- Wealth of libraries, open source code, and communities

- Many Ruby developers using Mac OSX

  - Lots of interest in building Mac OSX apps

RubyCocoa

- Bi-directional communication

  - Ruby within a Mac OSX app, internal scripting

  - Allows you to write Cocoa apps in Ruby

- Bridges two worlds together

```
require 'osx/cocoa'
diggy =
OSX::NSSound.alloc.initWithContentsOfFile_byReference("tiger.m4a",
false)
diggy.play
```

Sounds awesome!

- particularly if you don't know Objective-C
- ruby devs seem to dislike the [ ] brackets

There are a few caveats :)

Naming syntax

* Named parameters & conventions

* Objective-C

  - (id)tableView:(NSTableView *)aTableView objectValueForTableColumn:
  (NSTableColumn *)aTableColumn row:(NSInteger)rowIndex {

  }

* Ruby

  def tableView_objectValueForTableColumn_row(tableView, tableColumn, rowIndex)

  end

- Tough on the eyes for the ruby dev
- Objective-C dev looses their beloved named params

Type Hierarchies

- Objective-C

  - 'String' class is NSString

- Ruby

  - 'String' class is String

- RubyCocoa

  - 'String' class is String or OSX::NSString

  - String != OSX::NSString

There are 2 independent runtimes
- they don't know about each other
- what exists on the Cocoa side must be mapped to the Ruby side and vice versa
    - this is what the bridge does
- its also not for free

Learning Curve

* Learn Ruby

* Learn Cocoa

  * Apple documentation is in Objective-C

  * Learn Objective-C

* Learn RubyCocoa

* Now you can start your app

Also, Mac OSX only, RubyCocoa isn't on the phone/ipad

MacRuby is the shiny future

Both the desktop and the phone

– when its mature enough

Back into code!

– we'll be building a RubyCocoa app

– at the source level, RubyCocoa and MacRuby are quite similar