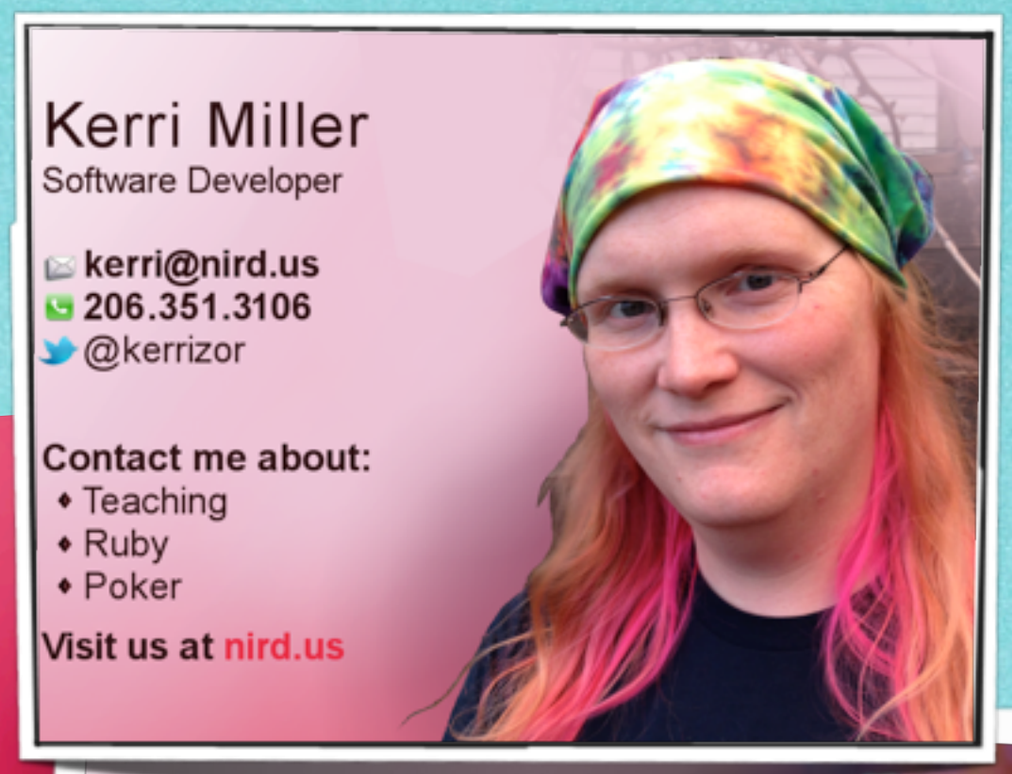
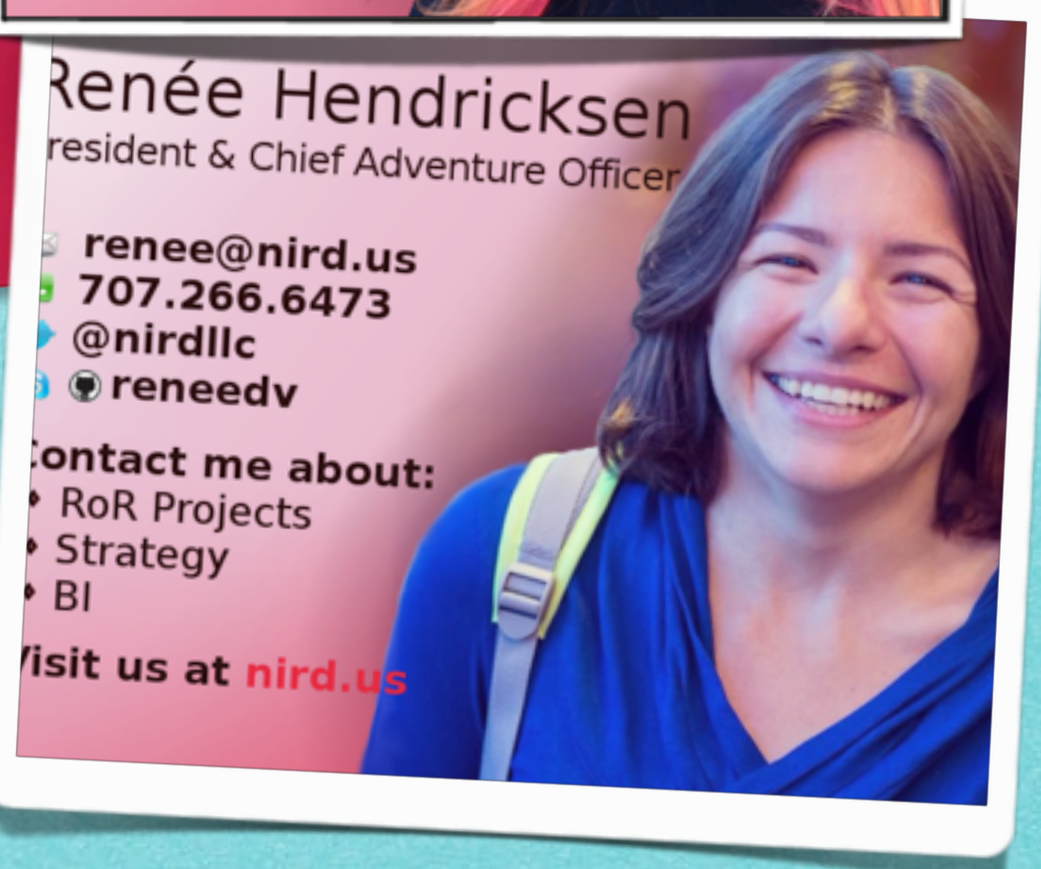
Advanced Ruby 3 Day

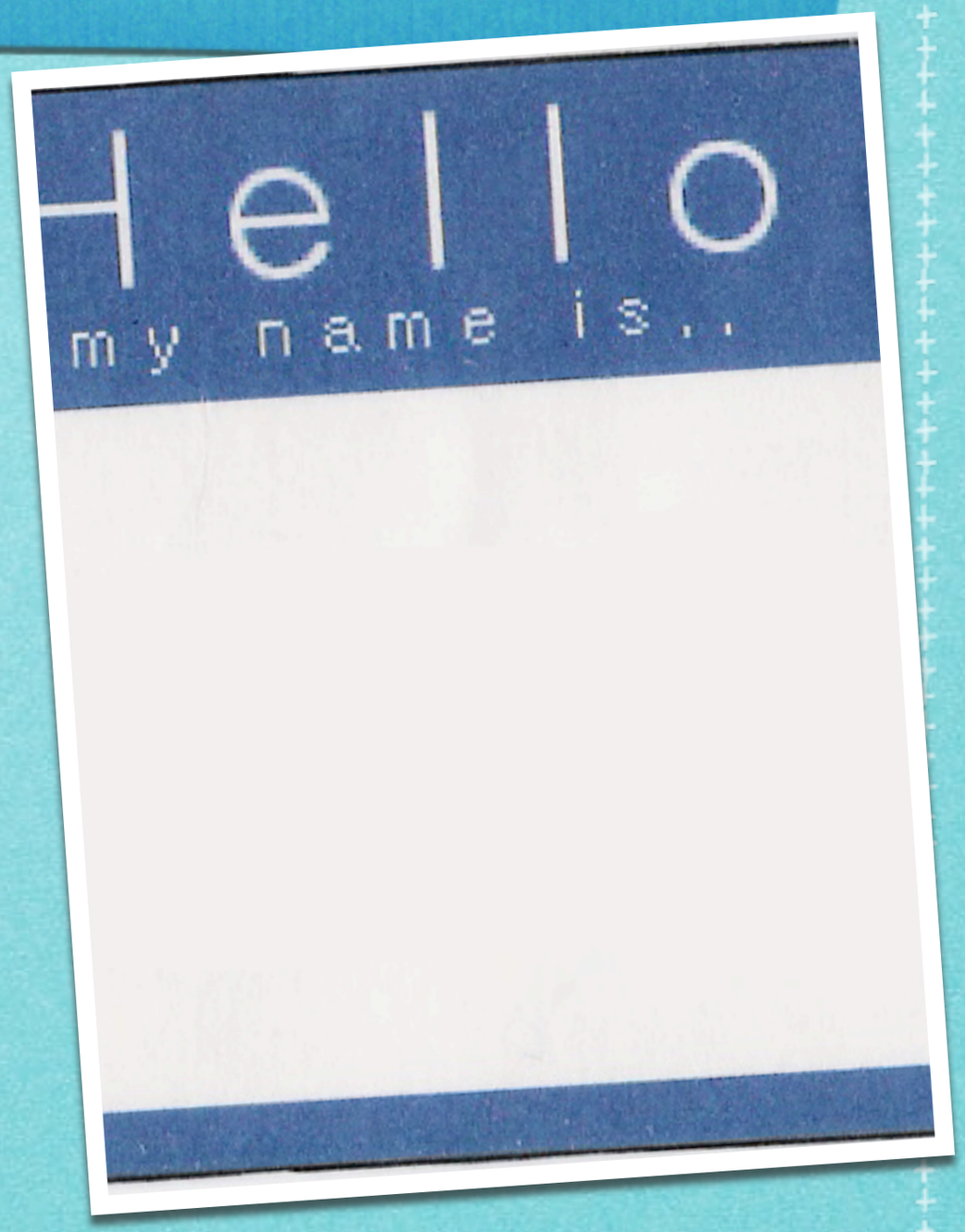
The Magical World of Ruby

By: Kerri Miller and Renée Hendricksen

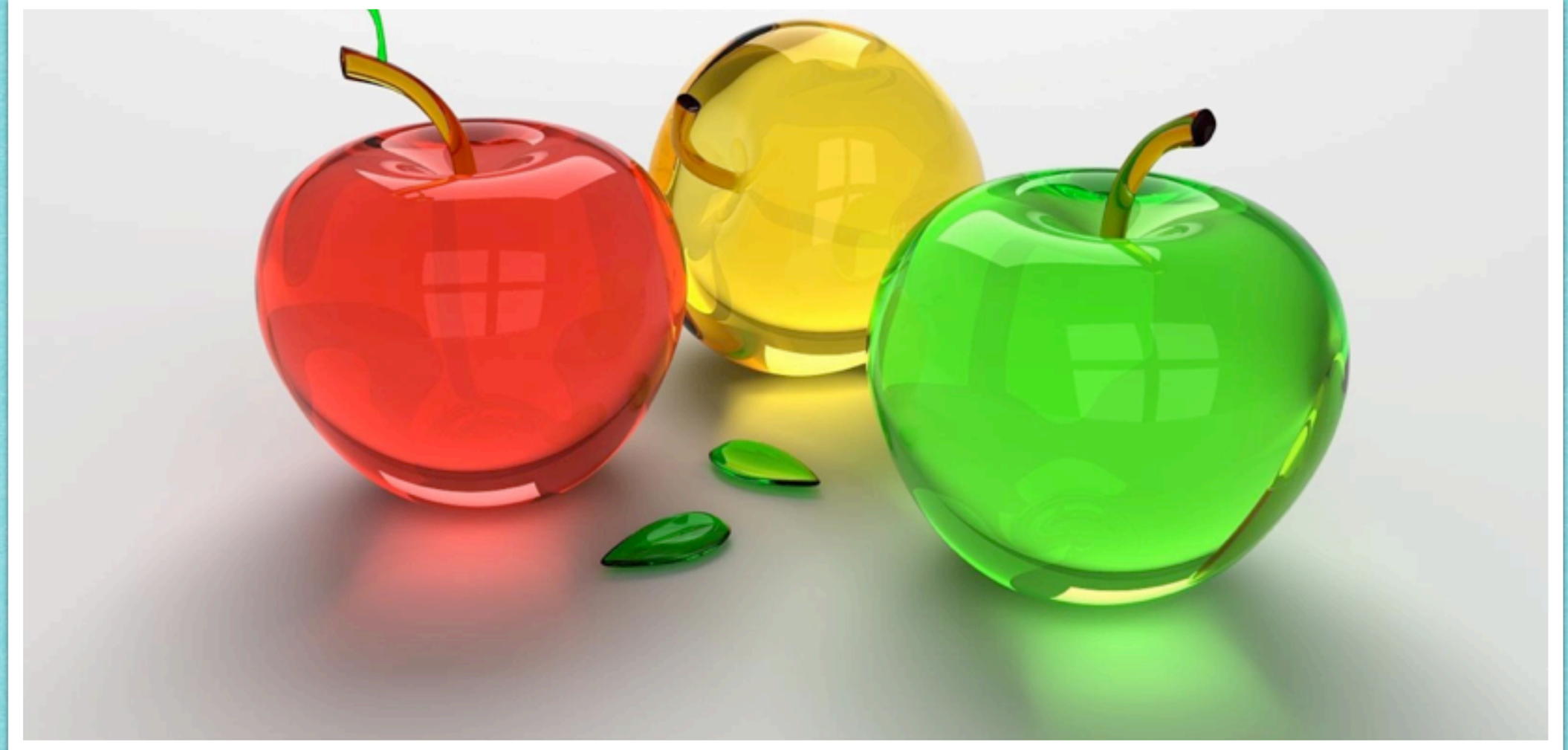

Kerri Miller
Software Developer
kerri@nird.us
206.351.3106
@kerrizor
Contact me about:
Teaching
Ruby
Poker
Visit us at nird.us


Renée Hendricksen
resident & Chief Adventure Officer
renee@nird.us
707.266.6473
@nirdllc
reneedv
ontact me about:
RoR Projects
Strategy
BI
isit us at nird.us

Welcome!



- Name
- Experience
- Class Goals



Objects

in Ruby

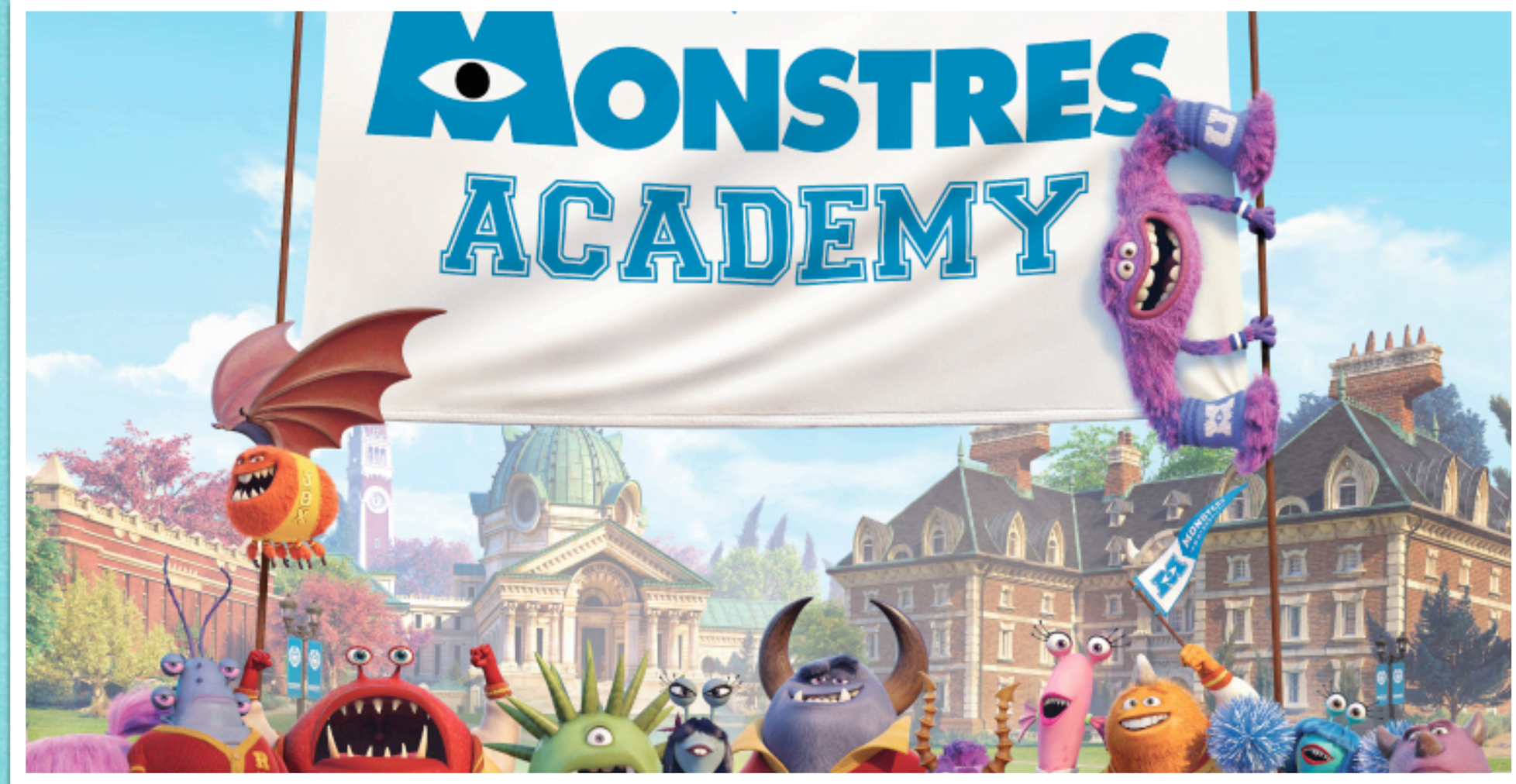
A quick tour of Ruby objects, classes, and modules

- Everything is an object
- metaclasses and metaobjects
  - Classes aren't that special!

method call chain

call chain self experiment

The object model
- from instance all the way up to BasicObject
- The changing face of `self`
  - class vs instance methods
  - the scope of `self`
- method call chains
- How methods are called dynamically
  - the magic of MethodMissing
- delegation patterns in Ruby
  - MethodMissing
  - StdLib: Forwardable



Inheritance

Privacy

Methods
- structure
- method dispatch
- public, protected, private
  - breaking the rules with .send
- Inheritance

Exercise!!

Exercise:
- Monster Academy
  - build a small menagerie of creatures that inherit from 2 or 3 base classes

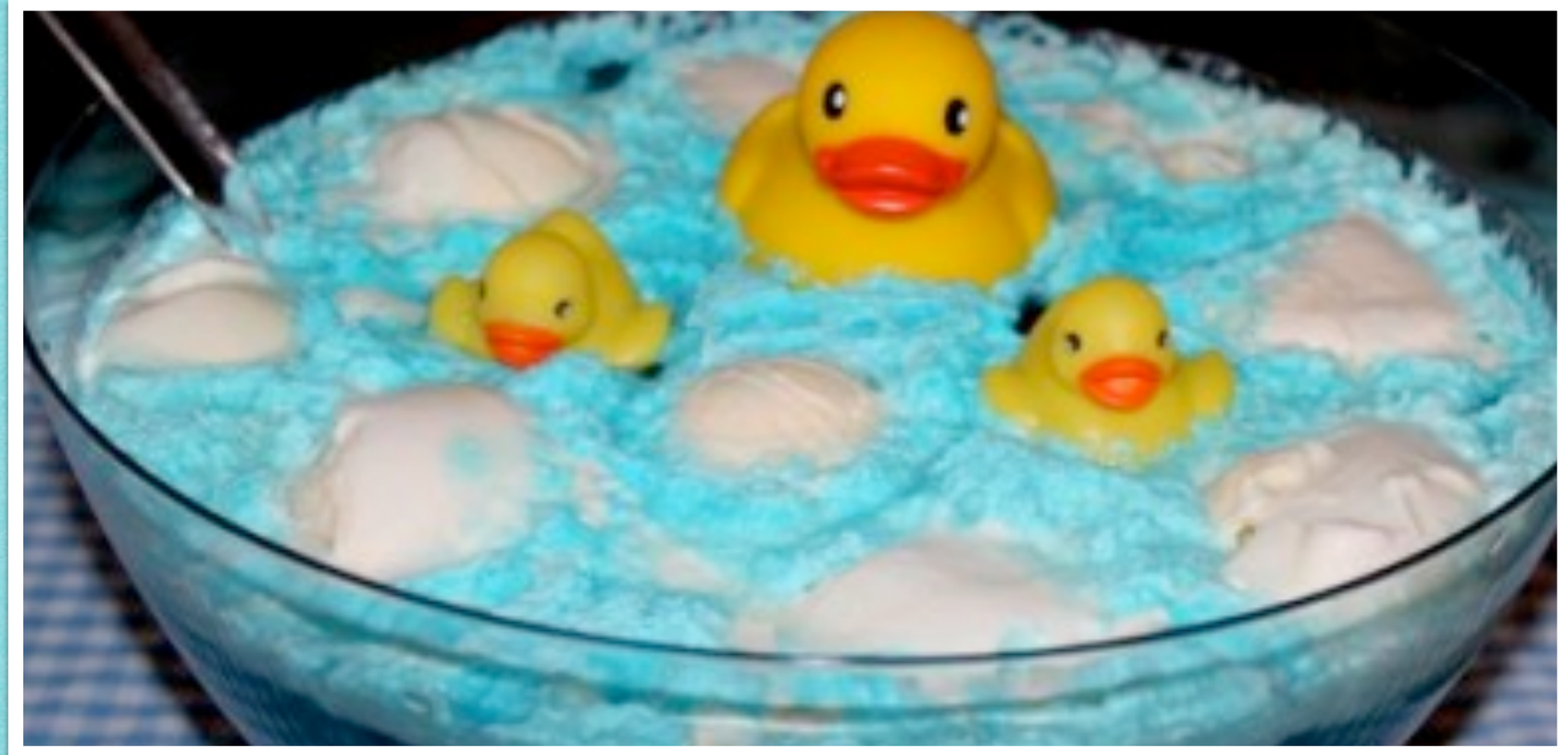
Modules

Shared Dry Behanvior

Modules & Mixins
- What are modules?
  - Modules as namespaces and reusable libraries
- Require VS Load VS Include VS Extend
- Mixing methods into classes using included
- Mixins: Ruby's path to Multiple Inheritance

Exercise!!

Exercise:
- extract common functionality from Monster Academy creatures into modules



Duck Punching

Runtime class extensions
- dynamically adding methods to a class
- "Monkey Patching" to overload a method's functionality
- creating methods on the fly using `define_method`

Runtime evaluation of code
- `eval`, `instance_eval`, and `class_eval`

Blocks

Blocks, Closures, Lambdas, Procs
- Ruby's syntax and implementation
- blocks as closures
- .yield
- self.yield
- .tap
- Symbol#to_proc (`&:name`)

Enumerable Demo

monsters.rb

standard enumerable demo w/ questions

Enumerable

Exercise:
- re-implement Enumerable

DSL

Bringing it together: Domain Specific Languages
How to write your own DSL
maybe look at rake

Additional Meta Exploration

Hook Methods
- Object lifestyle methods
  - methods_added
  - methods_removed
  - methods_undefined
- Decoupling code using inherited to keep track of subclasses
- Using `method_missing` to create magic methods
- Applying `const_missing` in both global and localized situations
- Common mistakes to avoid when metaprogramming (best practices)

Exercises:
- Accessing methods and variables from outside the class
- Add code to an existing method to perform additional work
- Examine Ruby with binding method
- Define a class and methods based on user-input

Concurrency and MultiThreading

Ruby's concurrency story - GITL
- A brief introduction to the available VMs
  - MRI, JRuby, Rubinius
  - Migrating between VMs - advantages and pitfalls
- Primitives - Threads and Mutexes and Fibers
- Advanced models via concurrent-ruby gem

Code Retreat Style Code

Exercise:

- Game of Life
  + complete a working example of Conway's Game of Life

Instrumentation

Best practices
- When it all goes sideways
  - Benchmark
  - pry
  - deep debugging and profiling
- Ruby best practices
  - idioms
  - code style
    - Rubocop
  - code metrics tools
    - Flog, Flay, Turbulence, Reek, SimpleCov, etc
  - Ruby and the SOLID principles of Object-Oriented Design

Perftools

Exercise!!

Exercise:
- generate benchmarks for each Game of Life solution and compare
- use debugging and style tools to refactor