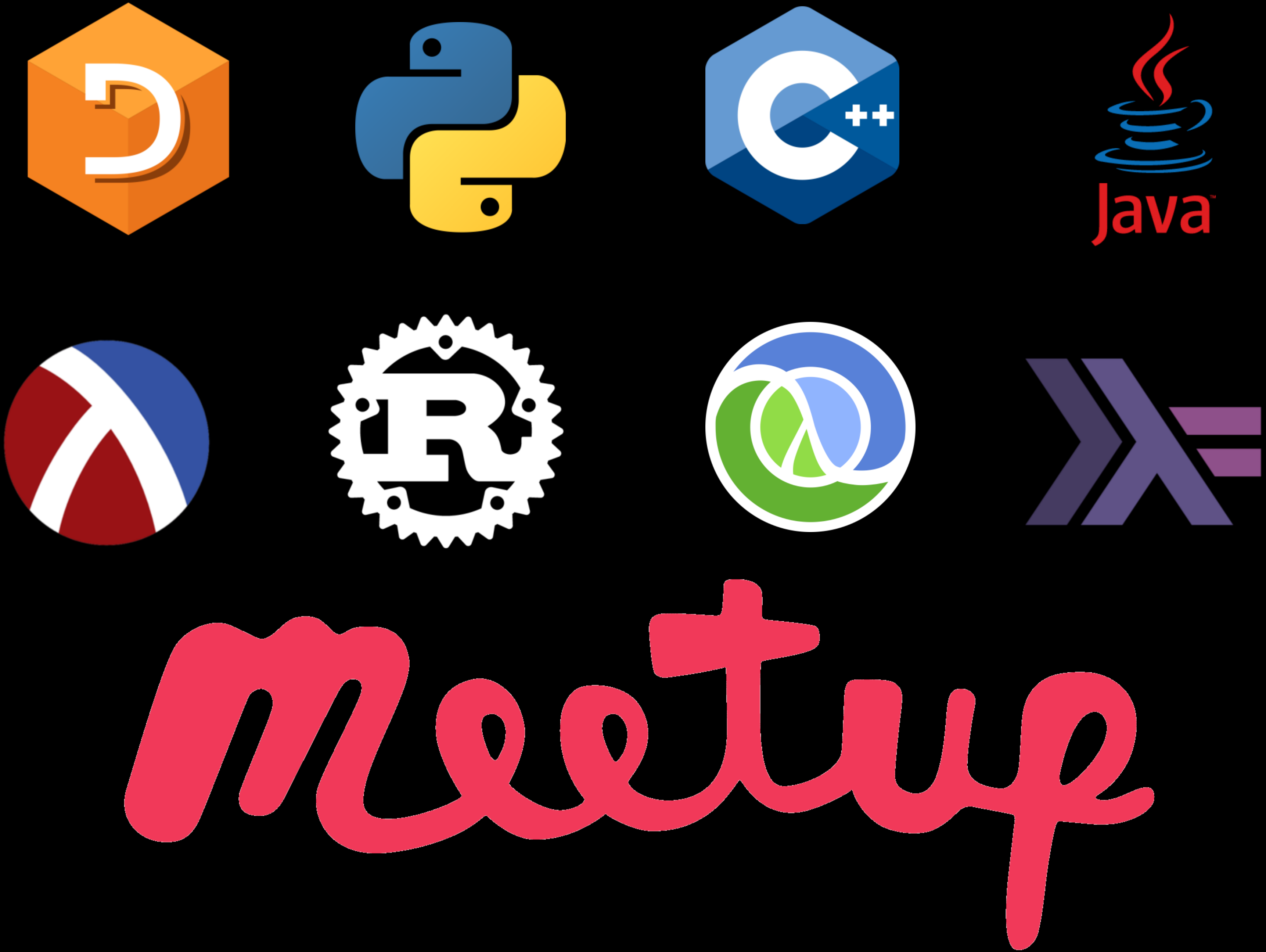
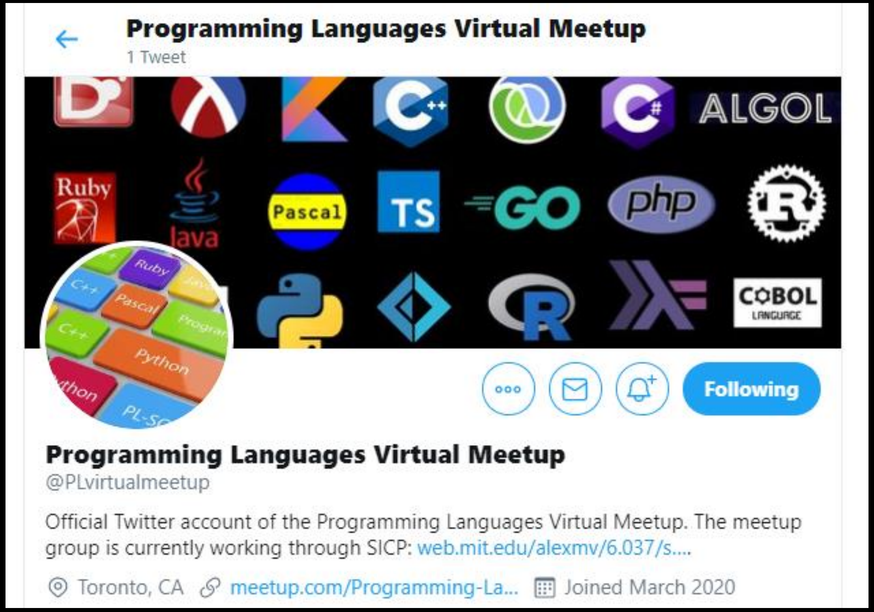

Java
Meetup

Friendly Environment Policy

Berlin Code of Conduct

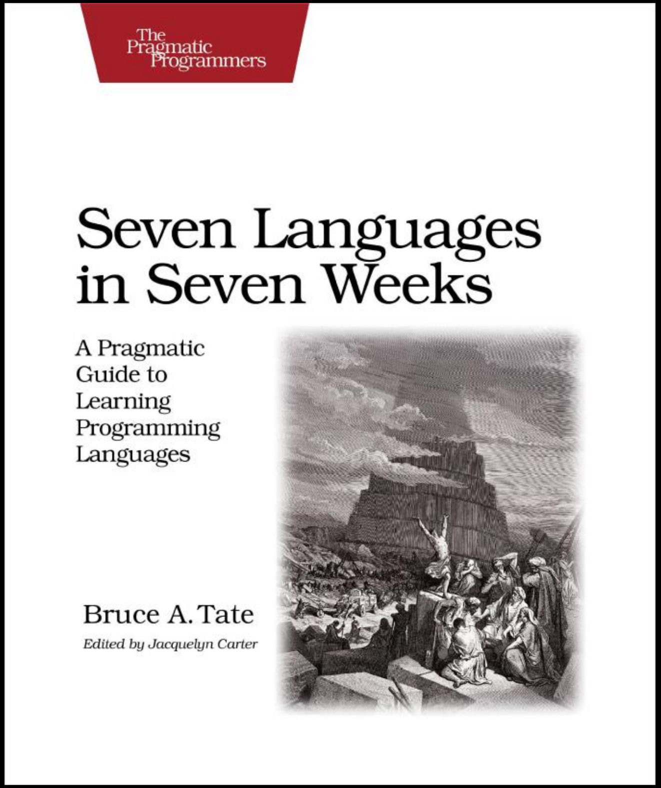


Programming Languages Virtual Meetup
1 Tweet

Programming Languages Virtual Meetup
@PLvirtualmeetup

Official Twitter account of the Programming Languages Virtual Meetup. The meetup group is currently working through SICP: web.mit.edu/alexmv/6.037/s....

Toronto, CA meetup.com/Programming-La... Joined March 2020



The Pragmatic Programmers

Seven Languages in Seven Weeks

A Pragmatic Guide to Learning Programming Languages

Bruce A. Tate

Edited by Jacquelyn Carter

7 Languages in 7 Weeks

Ruby

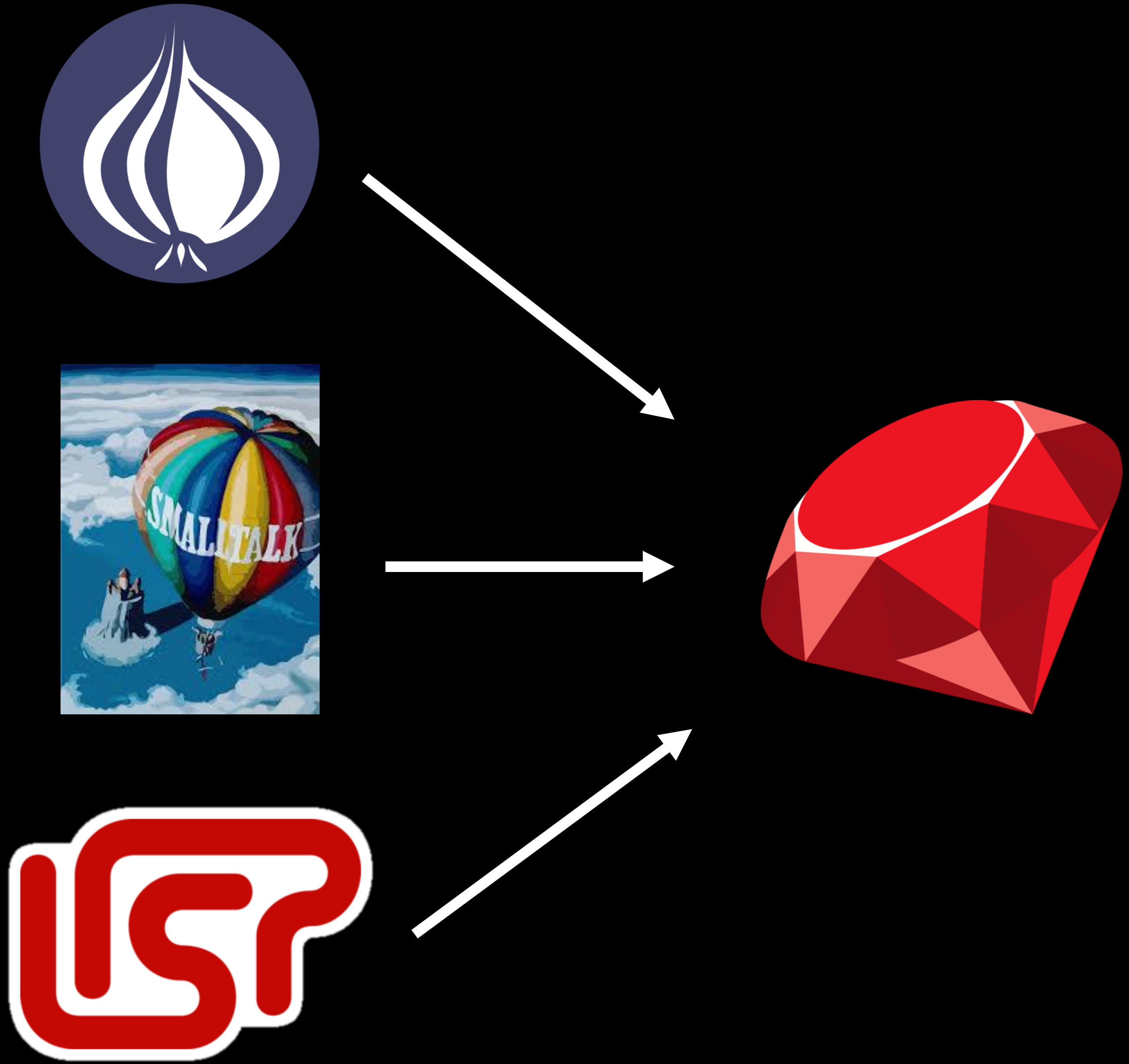
Io

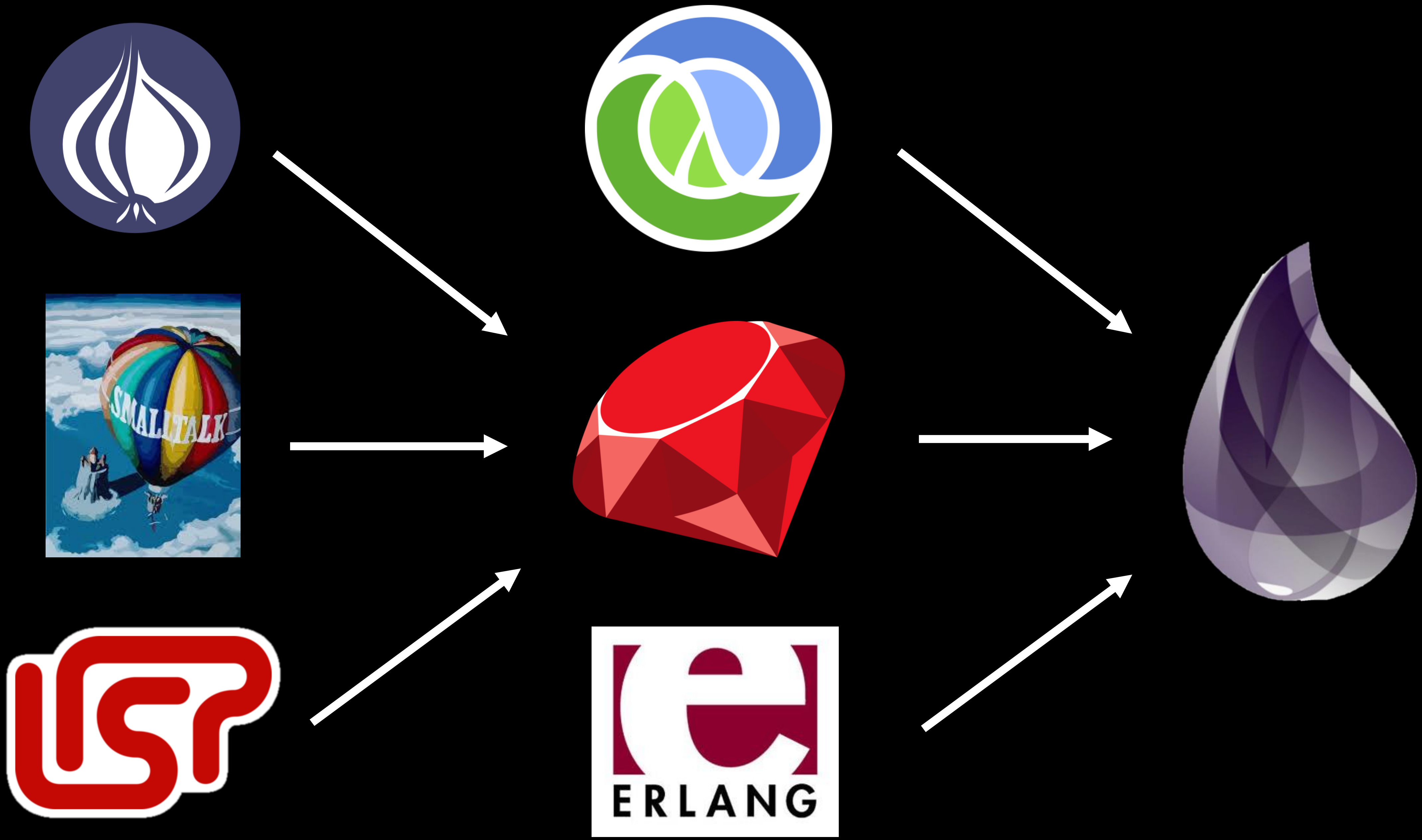
ERLANG

2.1 Quick History

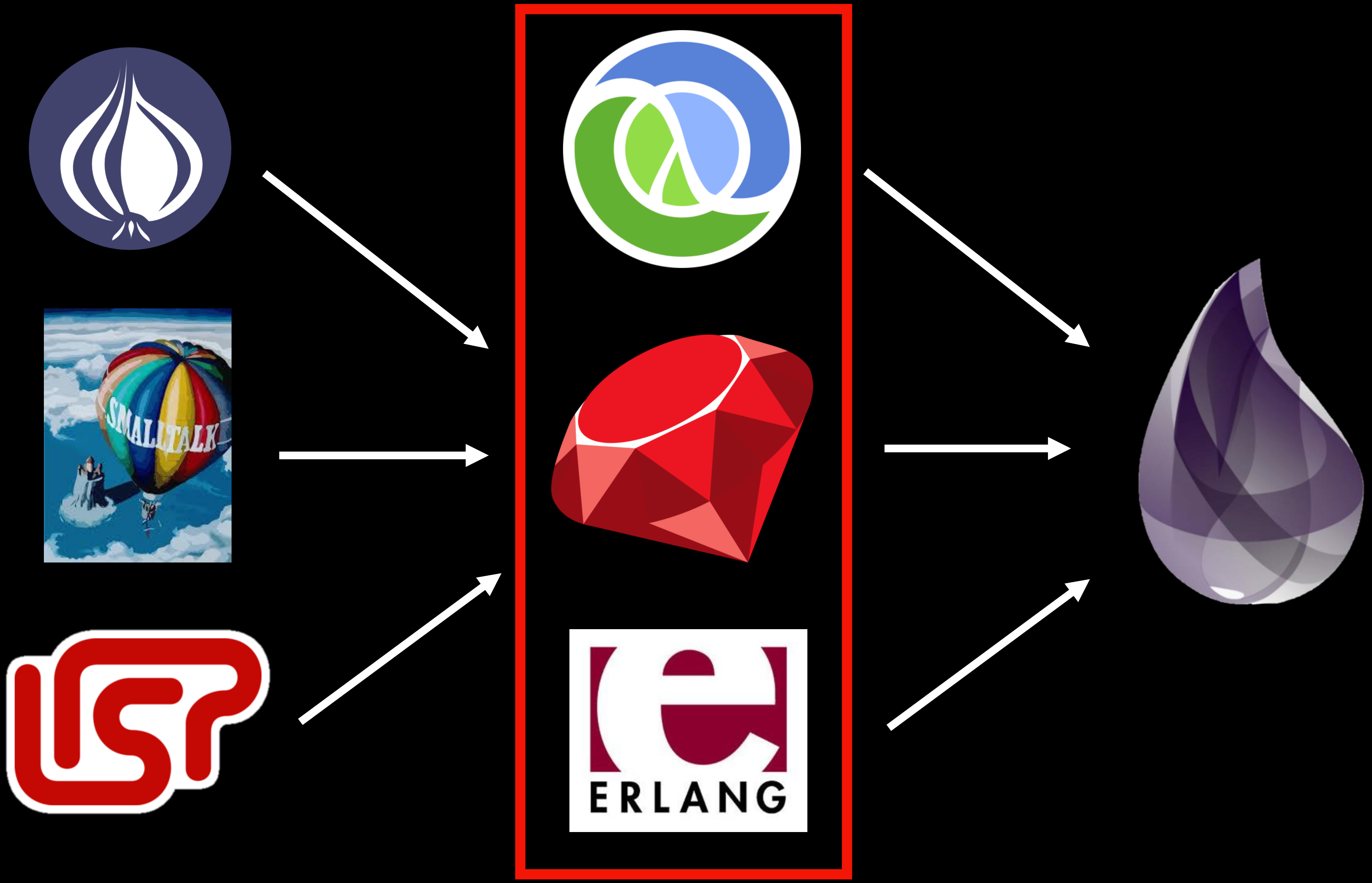
Yukihiro Matsumoto created Ruby in about 1993. Most people just call him Matz. As a language, Ruby is an interpreted, object-oriented, dynamically typed language from a family of so-called scripting languages. Interpreted means that Ruby code is executed by an interpreter rather than a compiler. Dynamically typed means that types are bound at execution time rather than compile time. In general, the trade-off for such a strategy is flexibility versus execution safety, but we'll get into that a little more later. Object-oriented means the language supports encapsulation (data and behavior are packaged together), inheritance through classes (object types are organized in a class tree), and polymorphism (objects can take many forms). Ruby patiently waited for the right moment and then burst onto the scene around 2006 with the emergence of the Rails framework. After wandering for ten years in the enterprise jungles, programming was fun again. Ruby is not hugely efficient in terms of execution speed, but it makes programmers very productive.

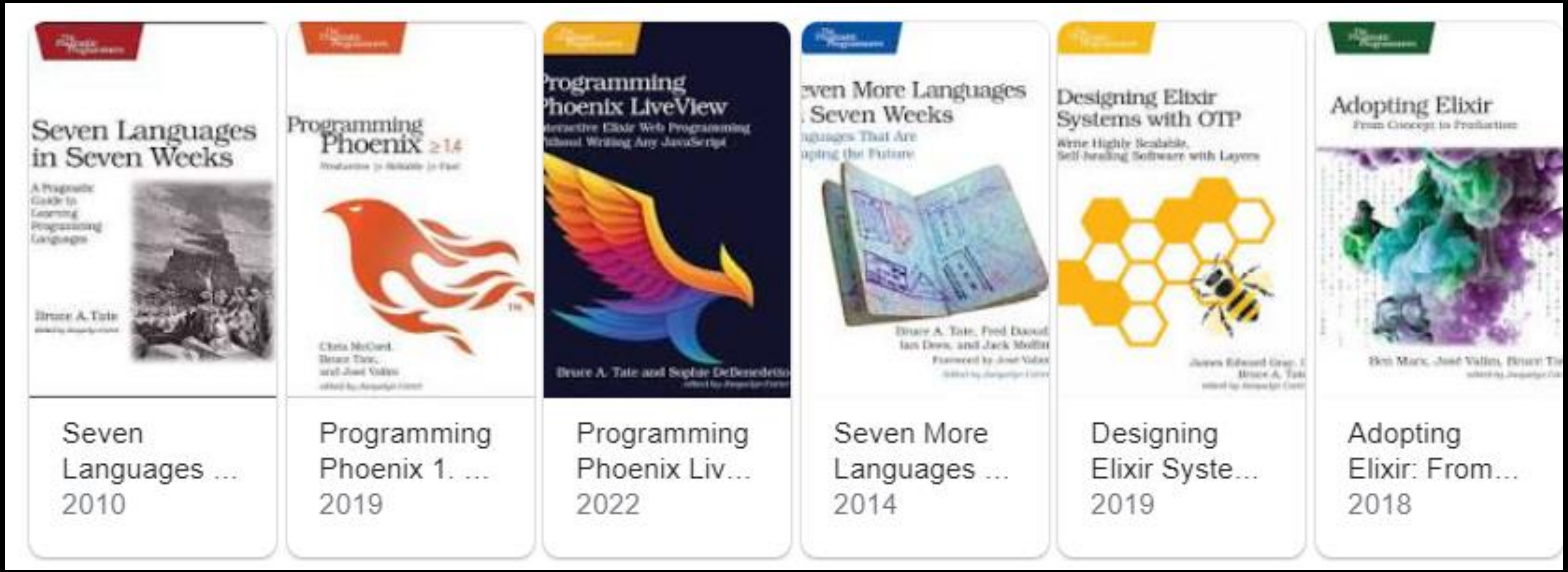
In 1993, when I saw Perl, I was somehow inspired that an object-oriented language that combines characteristics from Lisp, Smalltalk, and Perl would be a great language to enhance our productivity.


SMALLTALK
LISP



SMALLTALK
ERLANG


SMALLTALK
LISP
ERLANG



Seven Languages ... 2010

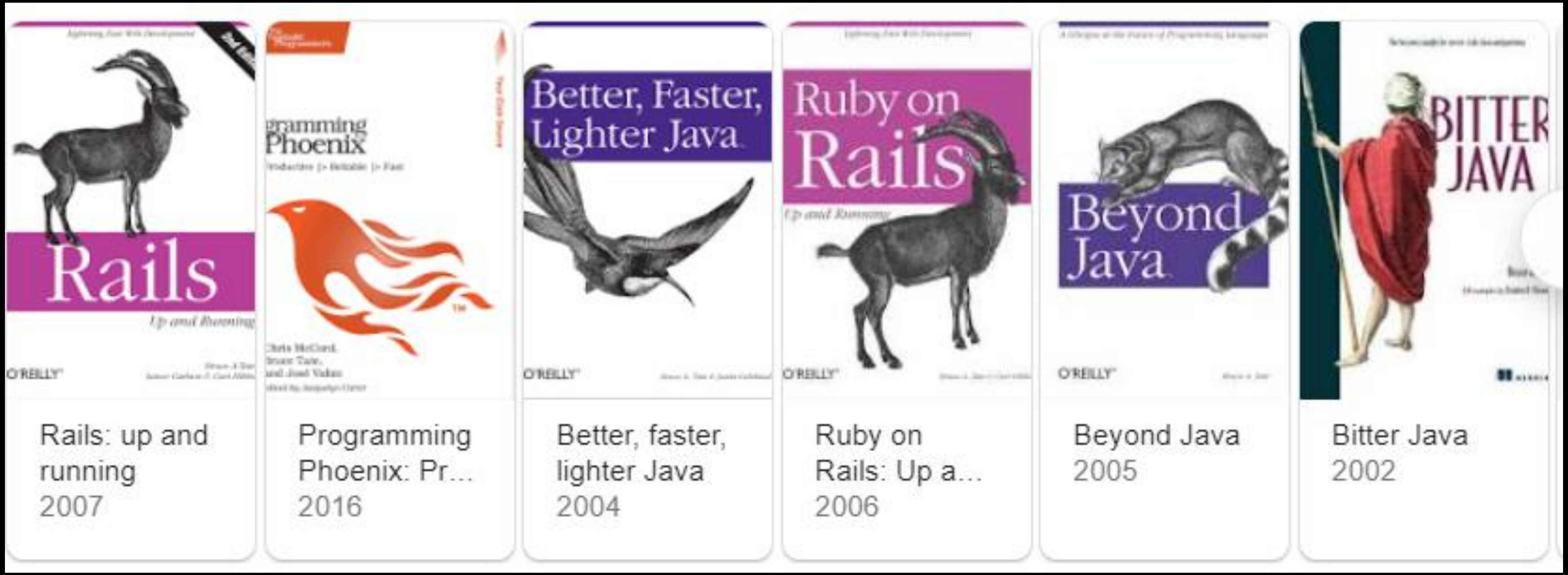
Programming Phoenix 1. ... 2019

Programming Phoenix Liv... 2022

Seven More Languages ... 2014

Designing Elixir Syste... 2019

Adopting Elixir: From... 2018



Rails: up and running 2007

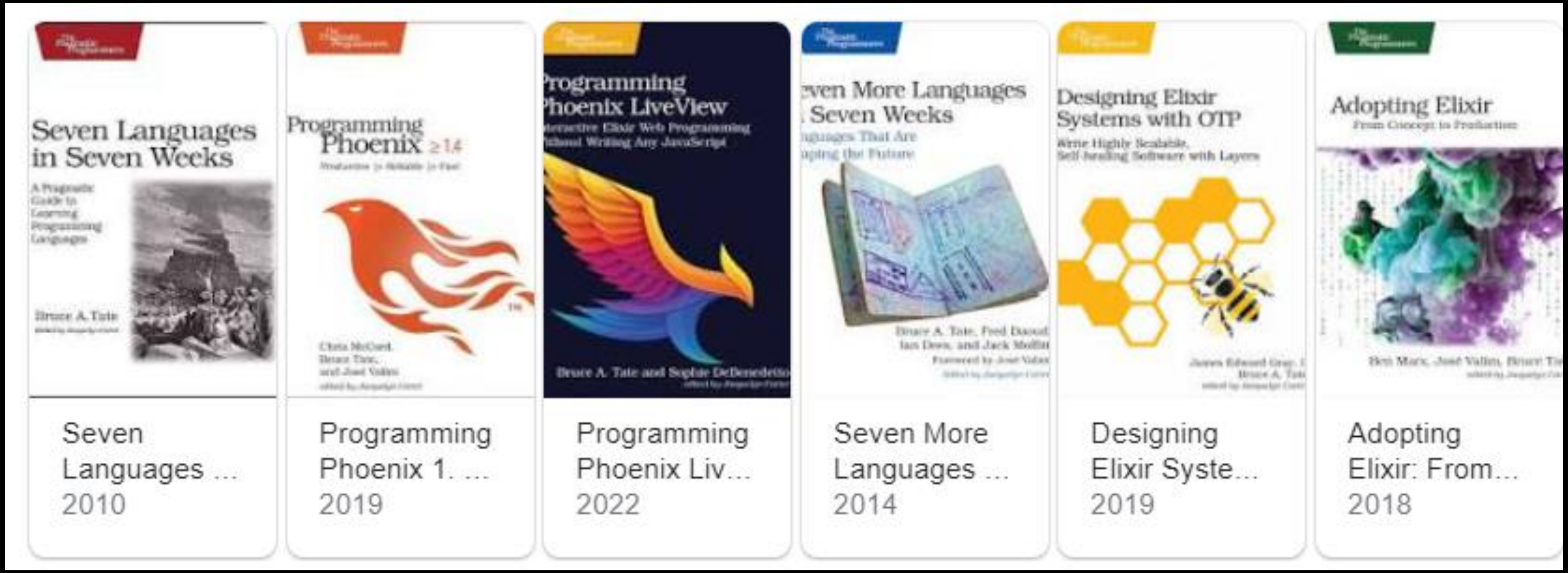
Programming Phoenix: Pr... 2016

Better, faster, lighter Java 2004

Ruby on Rails: Up a... 2006

Beyond Java 2005

Bitter Java 2002



Seven Languages ... 2010

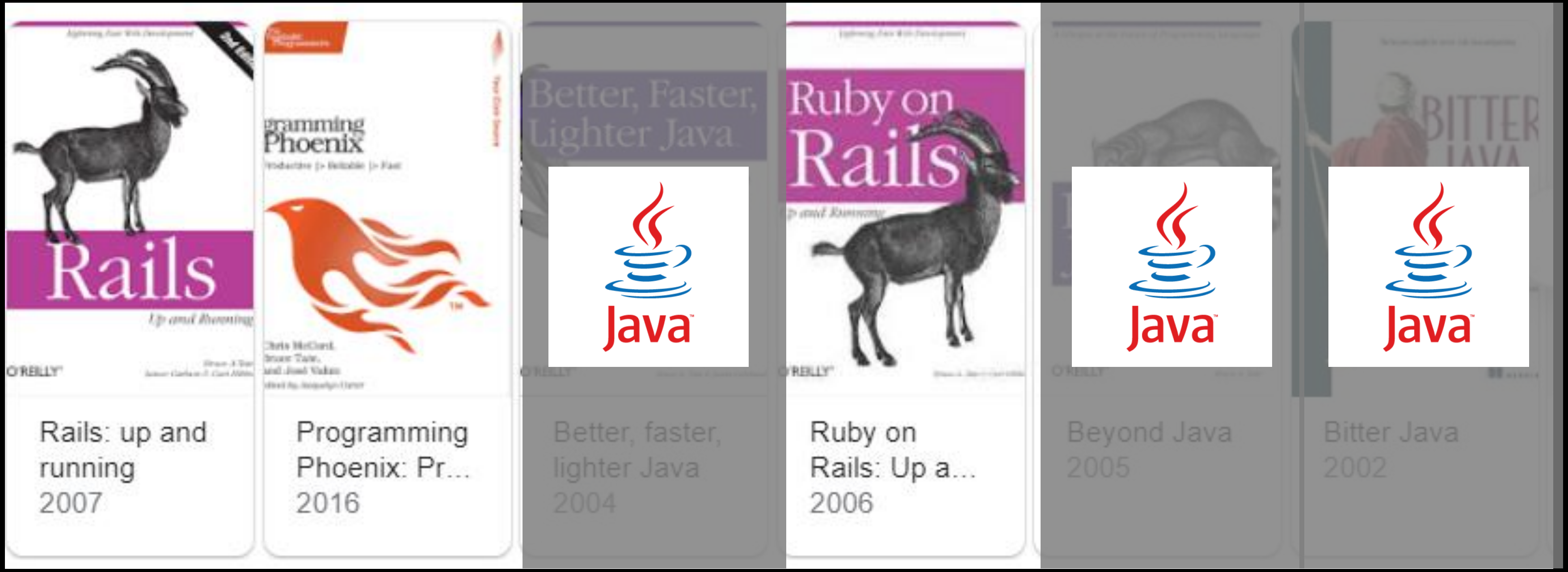
Programming Phoenix 1. ... 2019

Programming Phoenix Liv... 2022

Seven More Languages ... 2014

Designing Elixir Syste... 2019

Adopting Elixir: From... 2018



Rails: up and running 2007

Programming Phoenix: Pr... 2016

Better, faster, lighter Java 2004

Ruby on Rails: Up a... 2006

Beyond Java 2005

Bitter Java 2002

Seven Languages ... 2010

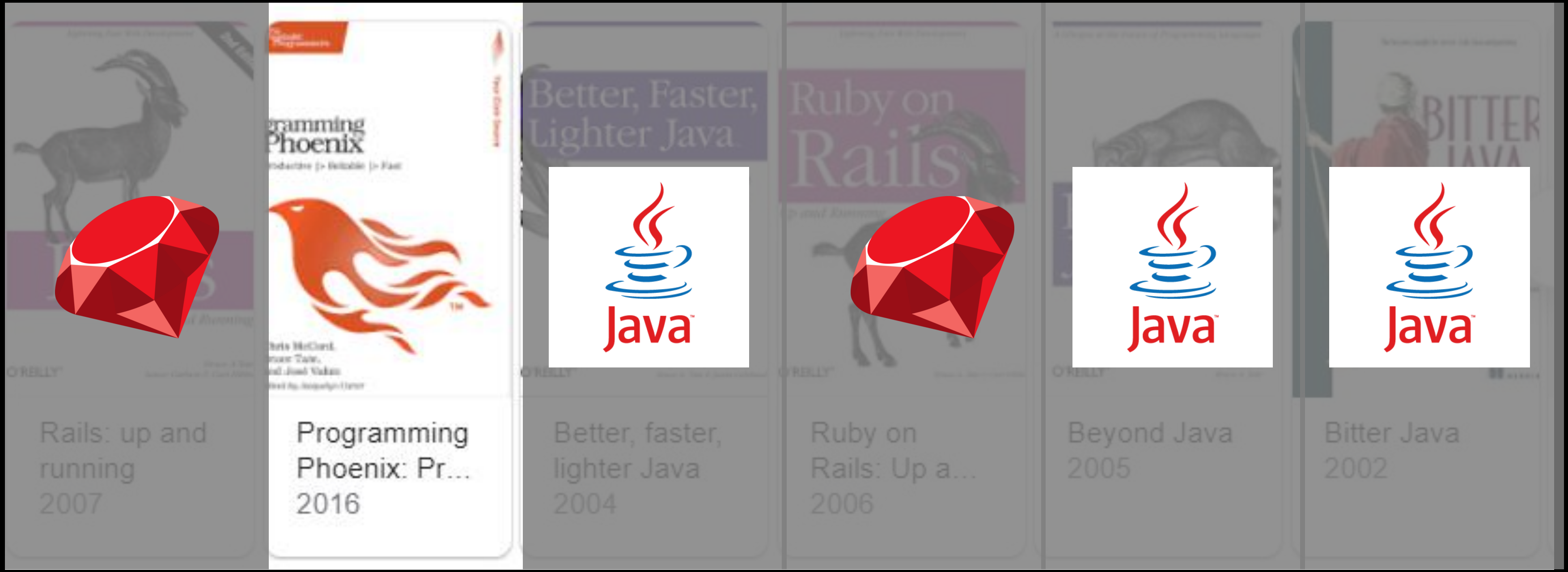
Programming Phoenix 1. ... 2019

Programming Phoenix Liv... 2022

Seven More Languages ... 2014

Designing Elixir Syste... 2019

Adopting Elixir: From... 2018



Rails: up and running 2007

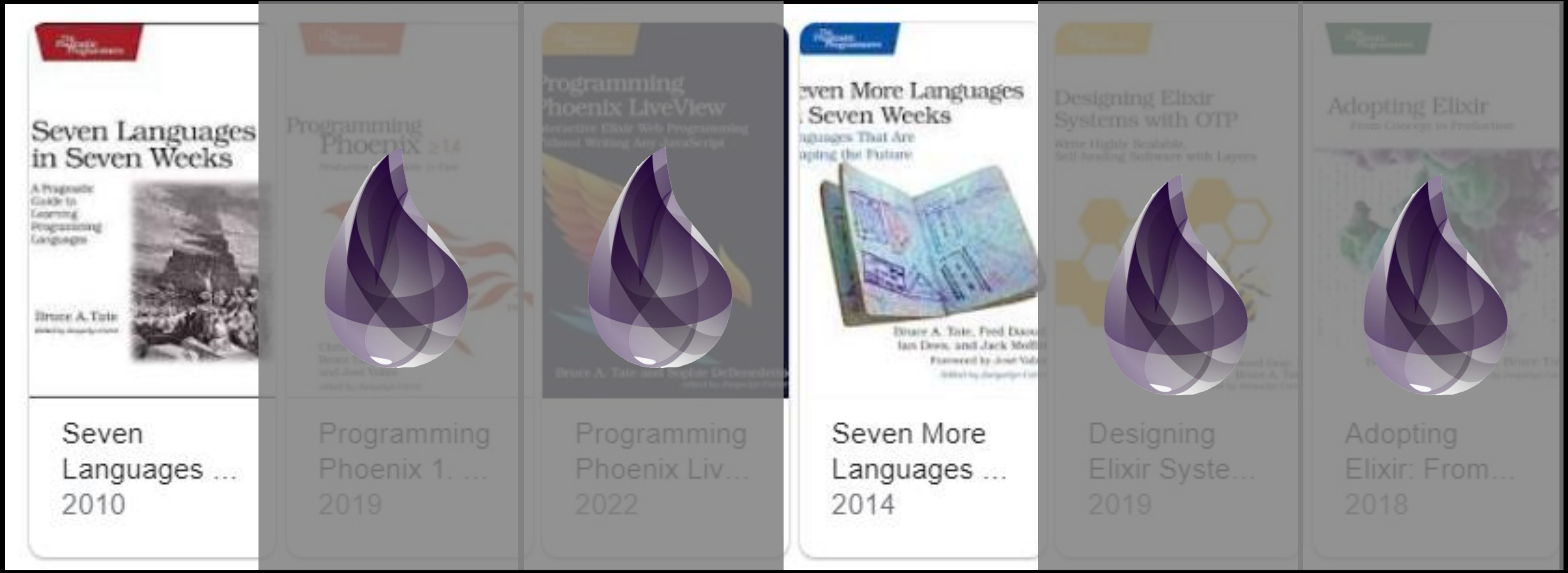
Programming Phoenix: Pr... 2016

Better, faster, lighter Java 2004

Ruby on Rails: Up a... 2006

Beyond Java 2005

Bitter Java 2002



Seven Languages ... 2010

Programming Phoenix 1. ... 2019

Programming Phoenix Liv... 2022

Seven More Languages ... 2014

Designing Elixir Syste... 2019

Adopting Elixir: From... 2018

Rails: up and running 2007

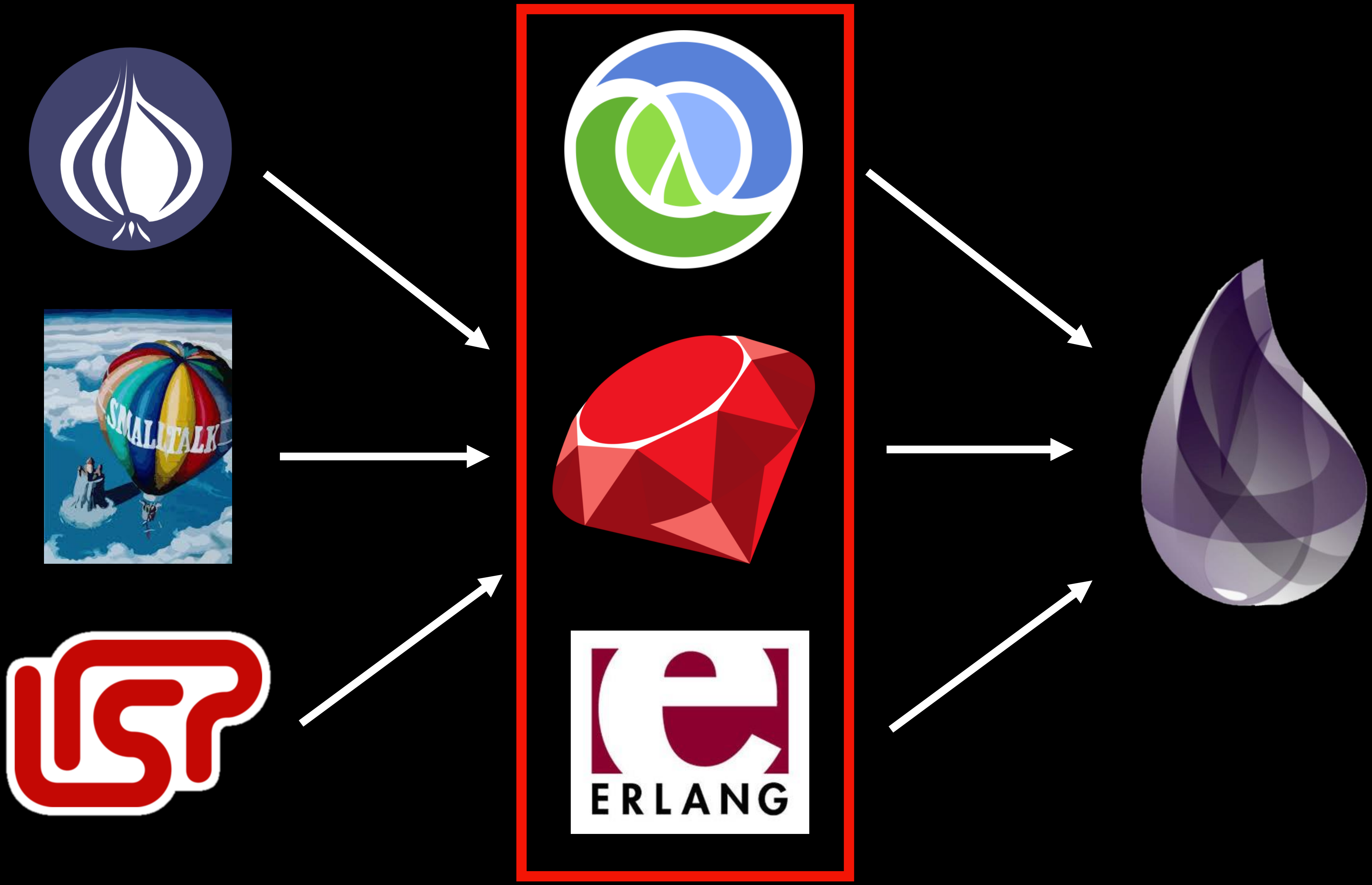
Programming Phoenix: Pr... 2016

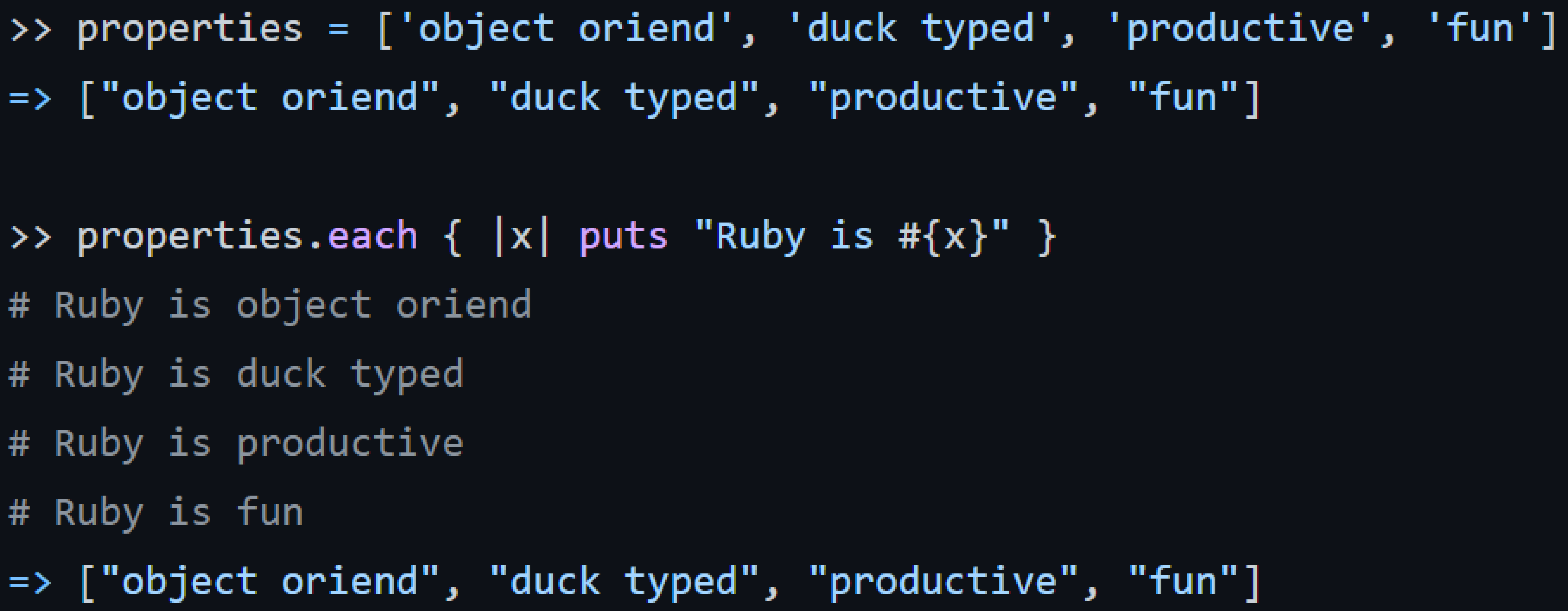
Better, faster, lighter Java 2004

Ruby on Rails: Up a... 2006

Beyond Java 2005

Bitter Java 2002


SMALLTALK
ERLANG

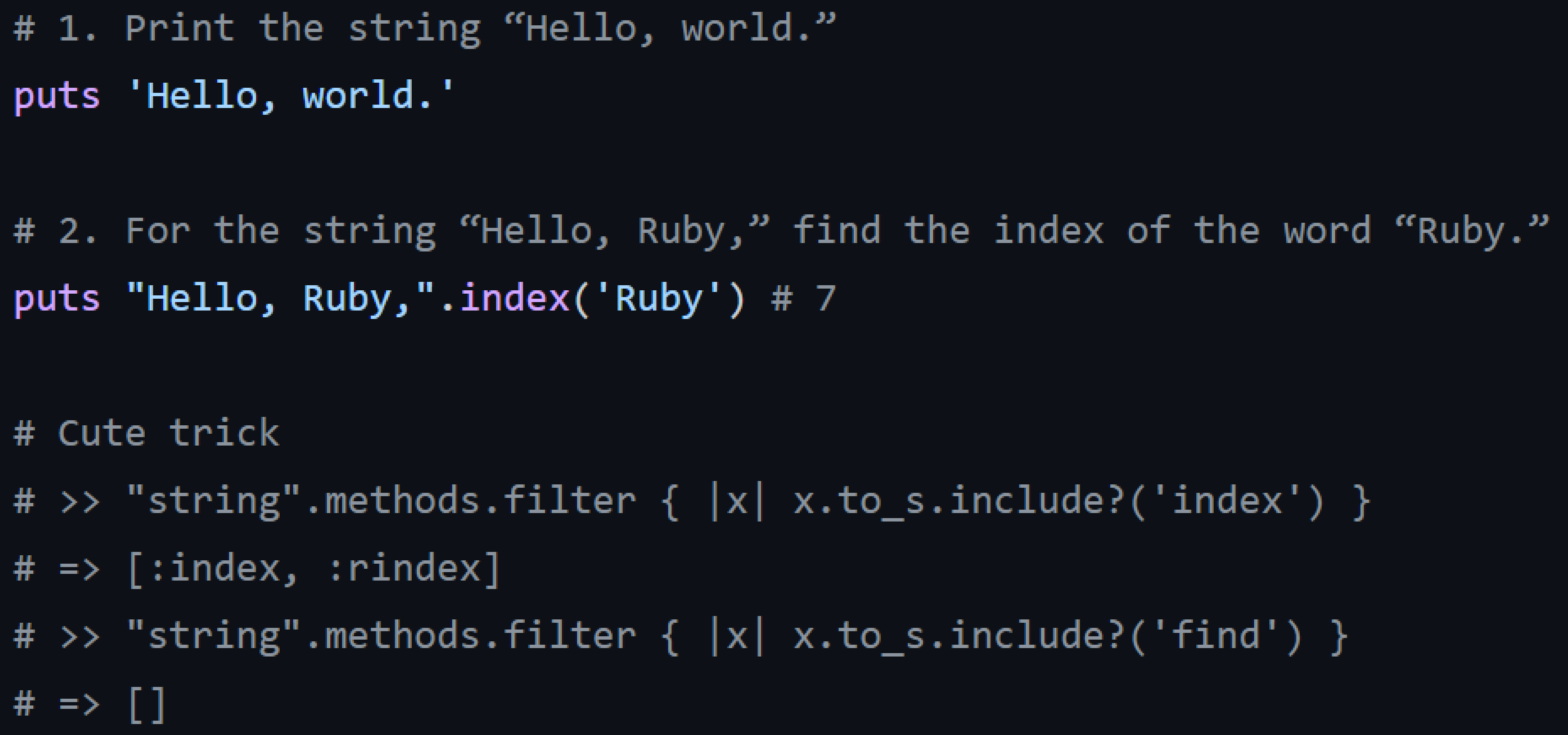


```
>> properties = ['object oriend', 'duck typed', 'productive', 'fun']
=> ["object oriend", "duck typed", "productive", "fun"]

>> properties.each { |x| puts "Ruby is #{x}" }
# Ruby is object oriend
# Ruby is duck typed
# Ruby is productive
# Ruby is fun
=> ["object oriend", "duck typed", "productive", "fun"]
```

```
>> 4
=> 4
>> 4.class
=> Integer
>> 4.methods
=> [:bit_length, :digits, :|, :numerator, :gcd, :-@, :**, :<=>, :<<, :>>, :<=, :>=, :==, :===, ... ]
>> Integer.methods
=> [:sqrt, :allocate, :superclass, :<=>, :<=, :>=, :==, :===, :included_modules, :include?, ... ]
>> Integer.methods.sort
=> [:!, :!=, :!~, :<, :<=, :<=>, :==, :===, :=~, :>, :>=, :__id__, :__send__, :alias_method, ... ]
>> 4.methods.sort
=> [:!, :!=, :!~, :%, :&, :*, :**, :+, :+@, :-, :-@, :/, :<, :<<, :<=, :<=>, :==, :===, :=~, :>, ... ]
```

Do:

- Print the string “Hello, world.”
- For the string “Hello, Ruby,” find the index of the word “Ruby.”
- Print your name ten times.
- Print the string “This is sentence number 1,” where the number 1 changes from 1 to 10.



```ruby
# 1. Print the string “Hello, world.”
puts 'Hello, world.'

# 2. For the string “Hello, Ruby,” find the index of the word “Ruby.”
puts "Hello, Ruby,".index('Ruby') # 7

# Cute trick
# >> "string".methods.filter { |x| x.to_s.include?('index') }
# => [:index, :rindex]
# >> "string".methods.filter { |x| x.to_s.include?('find') }
# => []
# 3. Print your name ten times.
puts "Conor Hoekstra\n" * 10

# 4. Print the string “This is sentence number 1,” where the number 1 changes from 1 to 10.
(1..10).each { |i| puts "This is sentence number #{i},\n" }
```

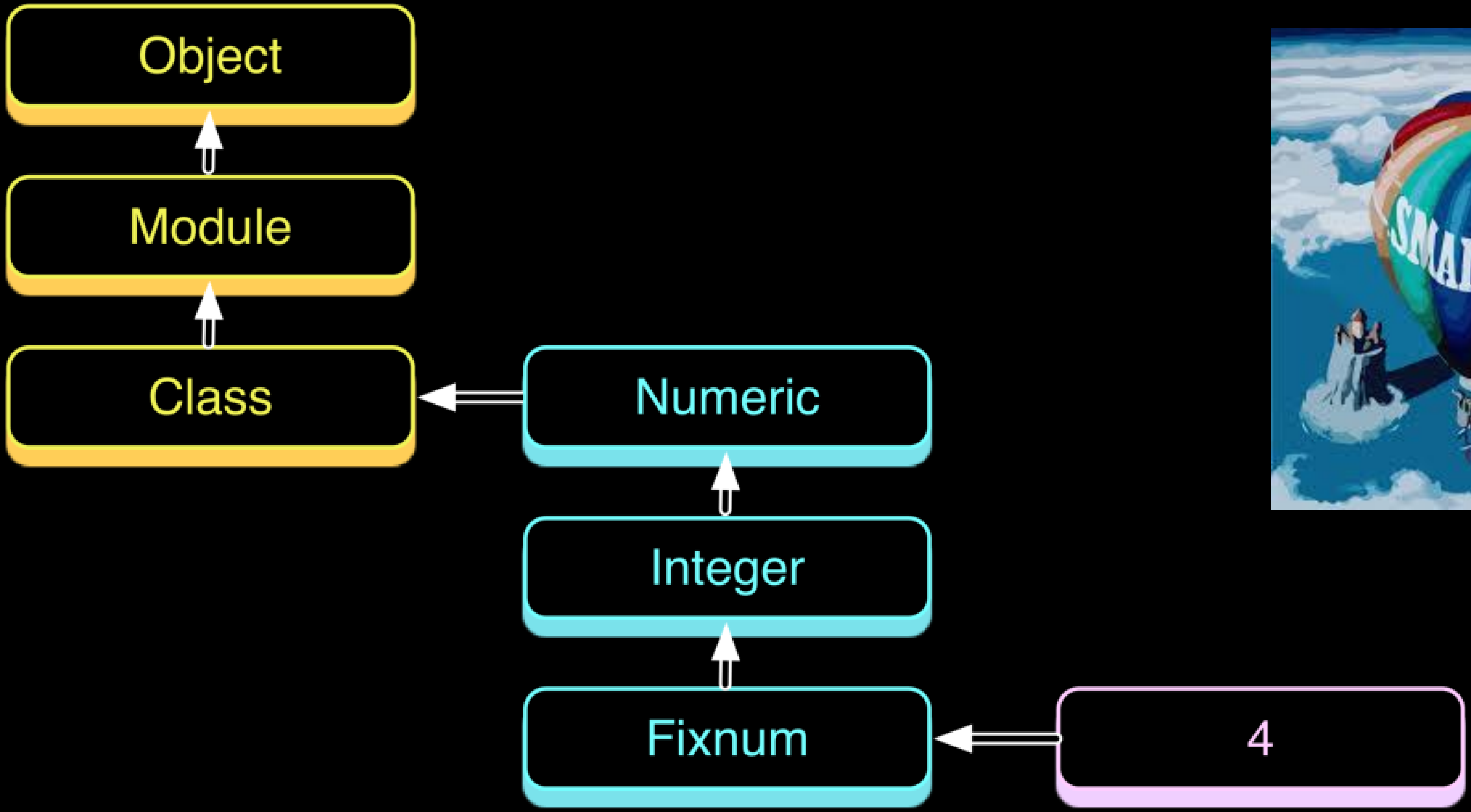
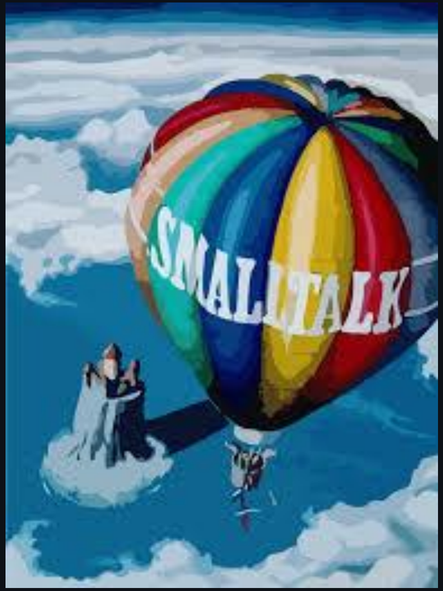
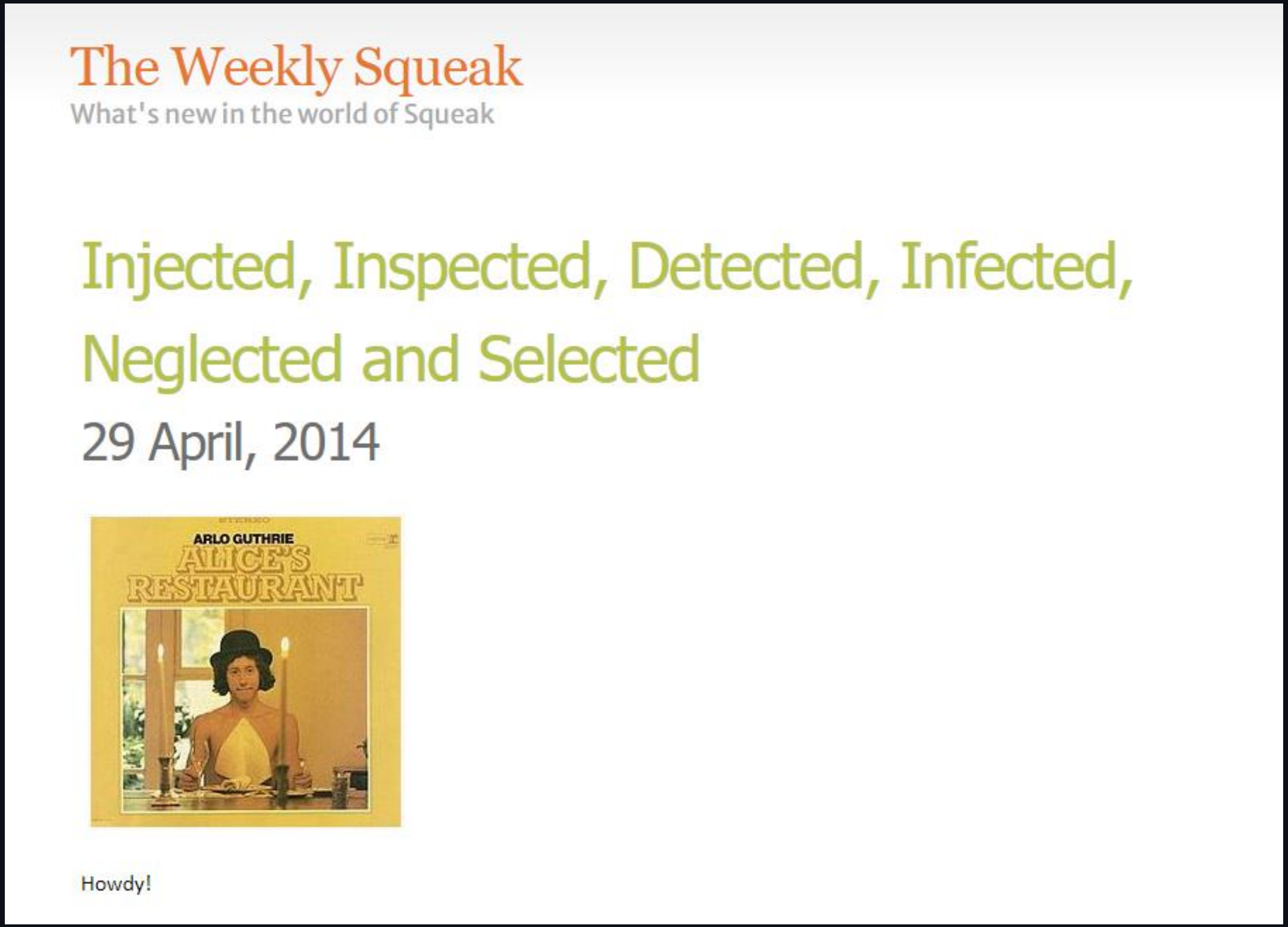


Figure 2.1: Ruby metamodel

```
>> (1..5).collect { |x| x * 2 }
=> [2, 4, 6, 8, 10]
>> (1..5).select { |x| x > 2 }
=> [3, 4, 5]
>> (1..5).inject(0) { |a,b| a + b }
=> 15
>> (1..5).inject { |a,b| a + b }
=> 15
>> (1..5).inject( &:+ )
=> 15
>> (1..5).sum
=> 15
```




The Weekly Squeak

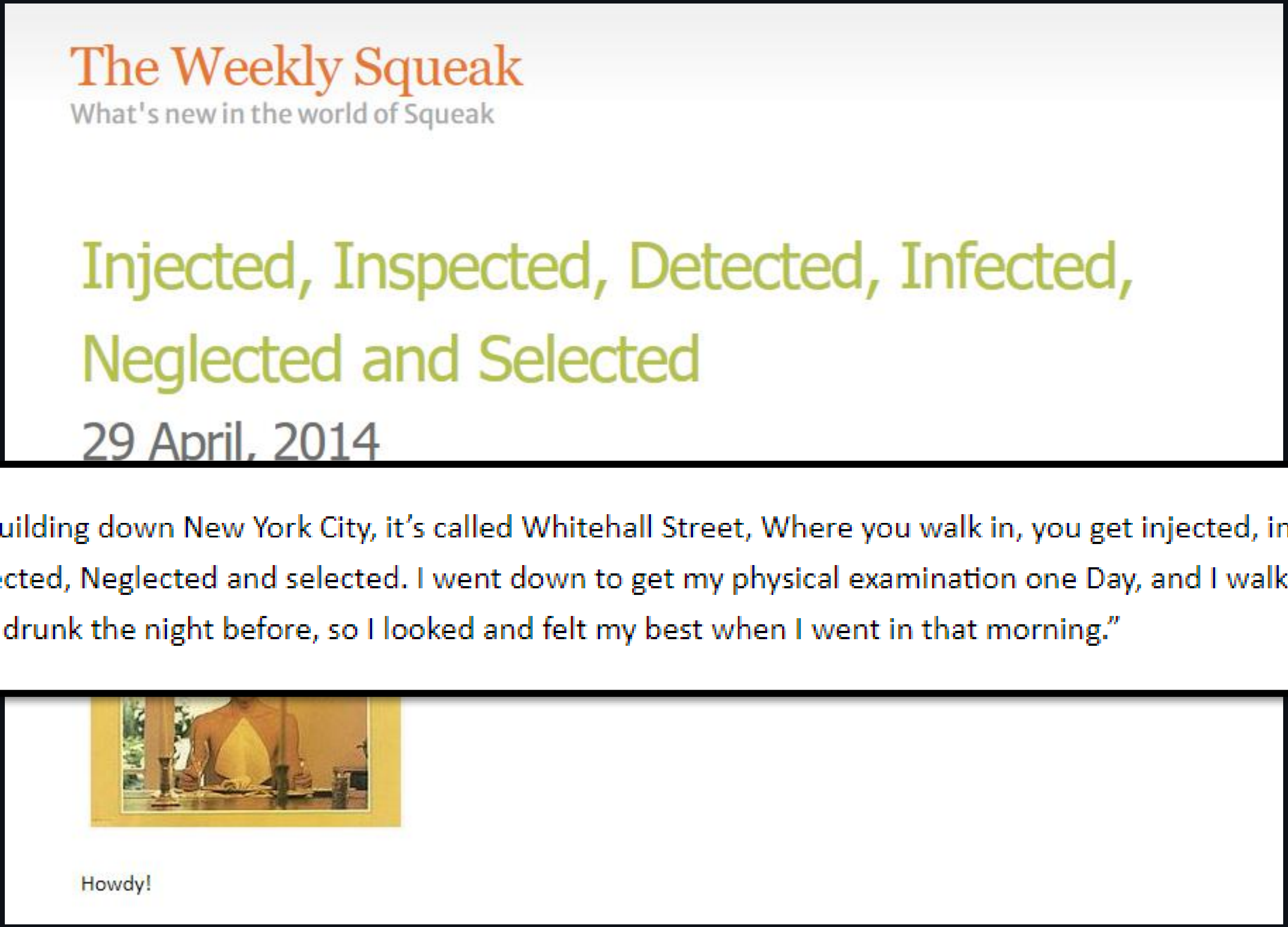
What's new in the world of Squeak

Injected, Inspected, Detected, Infected, Neglected and Selected

29 April, 2014

Howdy!

https://news.squeak.org/2014/04/29/injected-inspected-detected-infected-neglected-and-selected/



The Weekly Squeak

What's new in the world of Squeak

Injected, Inspected, Detected, Infected, Neglected and Selected

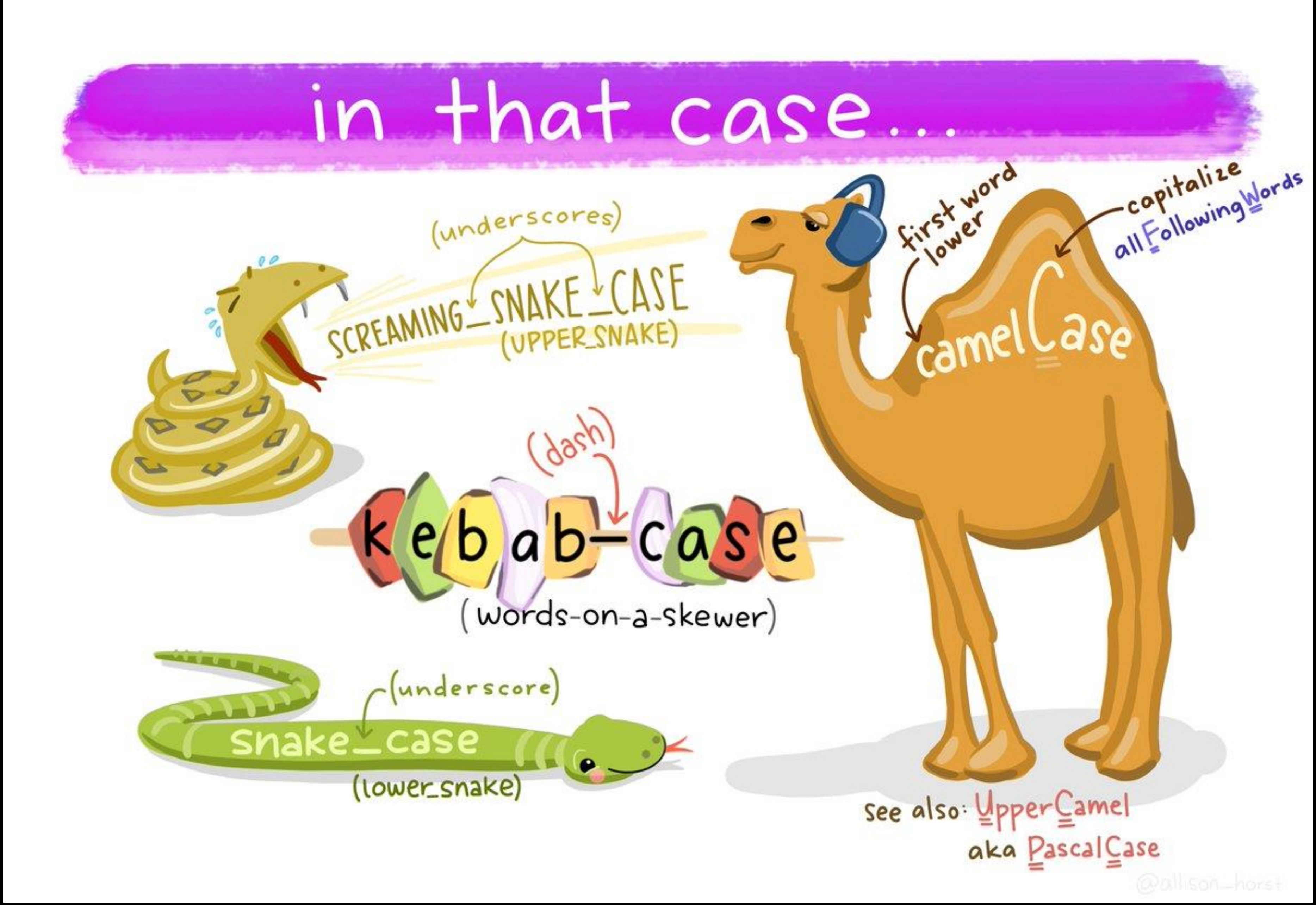
29 April, 2014

"They got a building down New York City, it's called Whitehall Street, Where you walk in, you get injected, inspected, detected, infected, Neglected and selected. I went down to get my physical examination one Day, and I walked in, I sat down, got good and drunk the night before, so I looked and felt my best when I went in that morning."

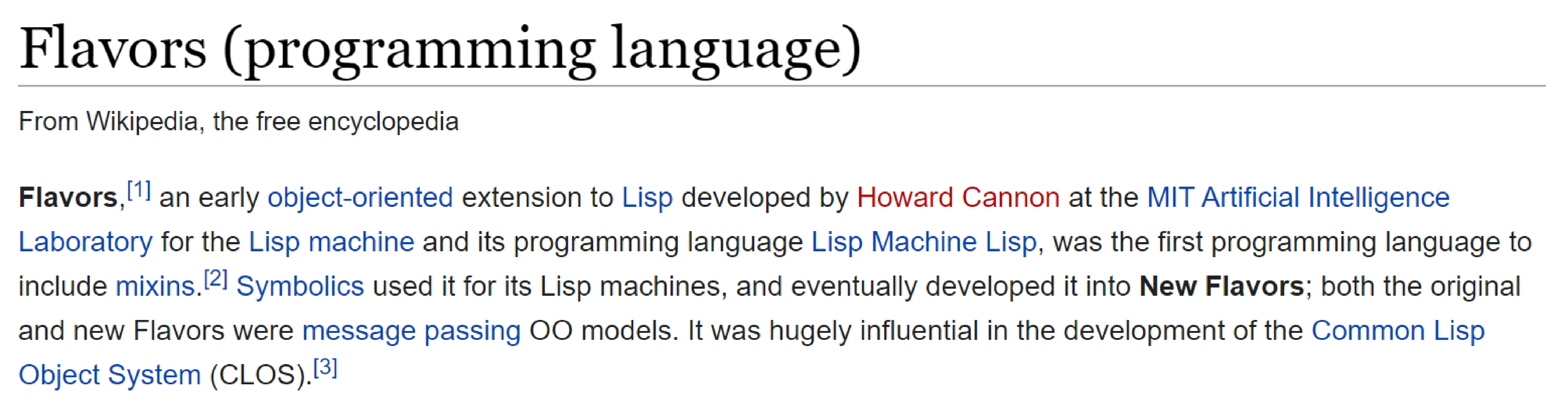
Howdy!

https://news.squeak.org/2014/04/29/injected-inspected-detected-infected-neglected-and-selected/

I should point out a few conventions and rules for Ruby. Classes start with capital letters and typically use CamelCase to denote capitalization. You must prepend instance variables (one value per object) with @ and class variables (one value per class) with @@. Instance variables and method names begin with lowercase letters in the underscore_style. Constants are in ALL_CAPS. This code defines a tree class. Each tree has two instance variables: @children and @node_name. Functions and methods that test typically use a question mark (if test?).


in that case...
(underscores)
SCREAMING_SNAKE_CASE
(UPPER_SNAKE)
(dash)
kebab-case
(words-on-a-skewer)
(underscore)
snake_case
(lower_snake)
first word lower
capitalize allFollowingWords
camelCase
see also: UpperCamel
aka PascalCase
@allison_horst

This style of programming, introduced in Flavors and used in many languages from Smalltalk to Python, is called a *mixin*. The vehicle that carries the mixin is not always called a module, but the premise is clear. Single inheritance plus mixins allow for a nice packaging of behavior.



Flavors (programming language)

From Wikipedia, the free encyclopedia

Flavors,[1] an early object-oriented extension to Lisp developed by Howard Cannon at the MIT Artificial Intelligence Laboratory for the Lisp machine and its programming language Lisp Machine Lisp, was the first programming language to include mixins.[2] Symbolics used it for its Lisp machines, and eventually developed it into **New Flavors**; both the original and new Flavors were message passing OO models. It was hugely influential in the development of the Common Lisp Object System (CLOS).[3]

- Write a simple grep that will print the lines of a file having any occurrences of a phrase anywhere in that line. You will need to do a simple regular expression match and read lines from a file. (This is surprisingly simple in Ruby.) If you want, include line numbers.

```ruby
puts File.open('conor_hoekstra_solutions.rb')
         .each_line
         .each_with_index
         .select { |line, i| line.include?('puts') }
         .collect { |line, i| i.to_s.ljust(4, ' ') + line.gsub(' ', '-') }
```

```ruby
puts File.open('conor_hoekstra_solutions.rb')
         .each_line
         .each_with_index
         .select { |line, i| line.include?('puts') }
         .collect { |line, i| i.to_s.ljust(4, ' ') + line.gsub(' ', '-') }
```

```
3   puts-'Hello,-world.'
6   puts-"Hello,-Ruby,".index('Ruby')-#-7
15  puts-"Conor-Hoekstra\n"-*-10
18  (1..10).each-{-|i|-puts-"This-is-sentence-number-#{i},\n"-}
27  ----puts-'Please-guess-a-number-between-1-and-100:'
33  ------------puts-'Your-guess-was-too-high'
35  ------------puts-'Your-guess-was-too-low'
37  --------puts-'Please-guess-again:'
42  ----puts-"Amazing,-you-guessed-#{target}-in-#{total_guesses}-tries!"
55  puts-File.open('conor_hoekstra_solutions.rb')
57  ---------.select-{-|line|-line.include?('puts')-}
59  puts-File.open('conor_hoekstra_solutions.rb')
62  ---------.select-{-|line,-i|-line.include?('puts')-}
92  csv.each-{-|row|-puts-row.one-}
```

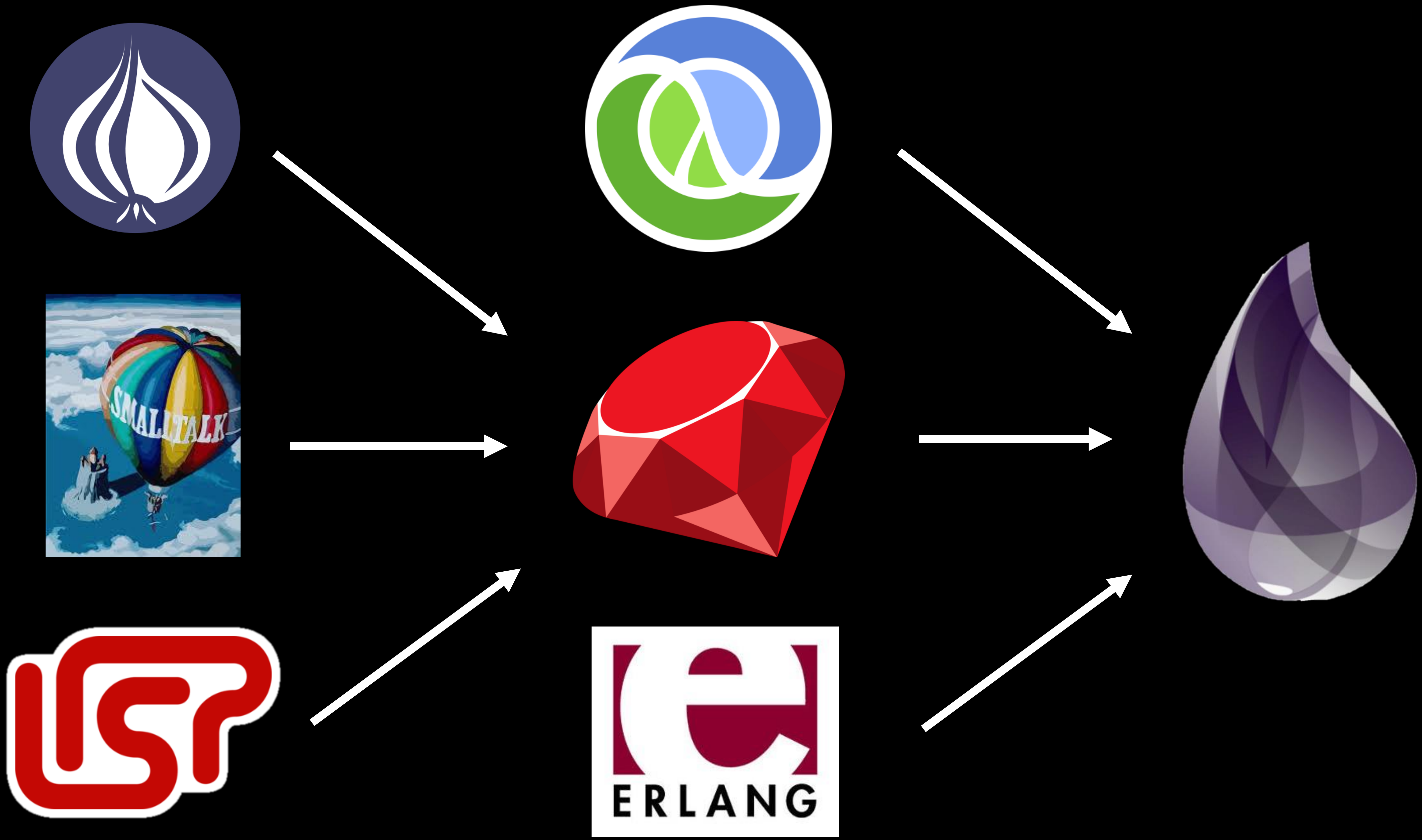
Do:

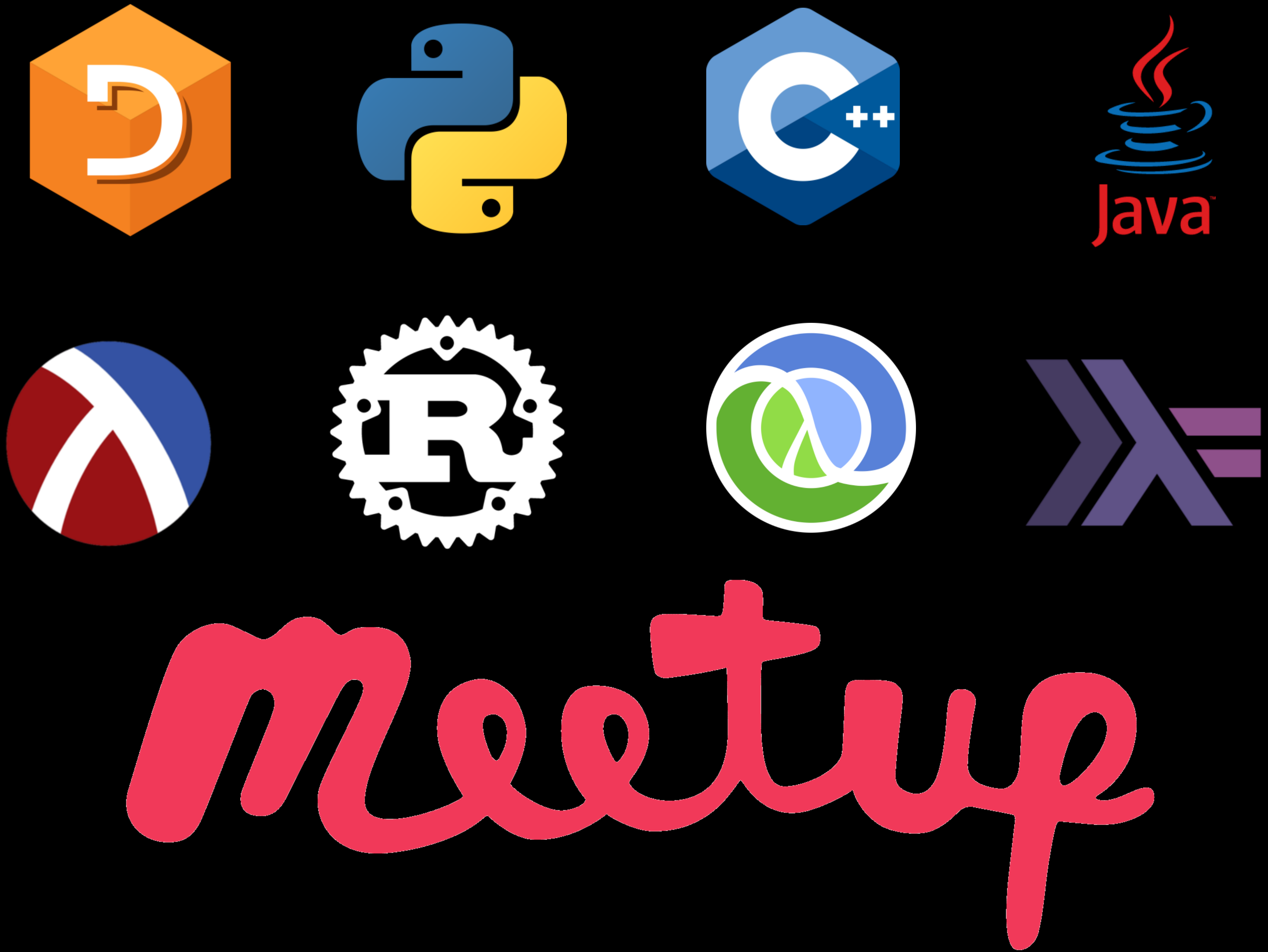
Modify the CSV application to support an each method to return a CsvRow object. Use method_missing on that CsvRow to return the value for the column for a given heading.

```ruby
class ActsAsCsv

    # ...

    # Add this
    def each(&block)
        @result.each do |row|
            block.call CsvRow.new(row, @headers)
        end
    end
end
```

```ruby
class ActsAsCsv

    # ...

    # Add this
    def each(&block)
        @result.each do |row|
            block.call CsvRow.new(row, @headers)
        end
    end
end
```

```ruby
class CsvRow
    def initialize(row, headers)
        @row = row
        @headers = headers
    end

    def method_missing(name)
        @row[@headers.index(name.to_s)]
    end
end
```


SMALLTALK
LISP
ERLANG


Java
Meetup