

Fifty Years of Ruby

Craig Buchek





“The best way to predict the future is to invent it.”

— Alan Kay

<https://craigbuchek.com/fifty>



@CraigBuchek

1993

<https://craigbuchek.com/fifty>



@CraigBuchek

1968

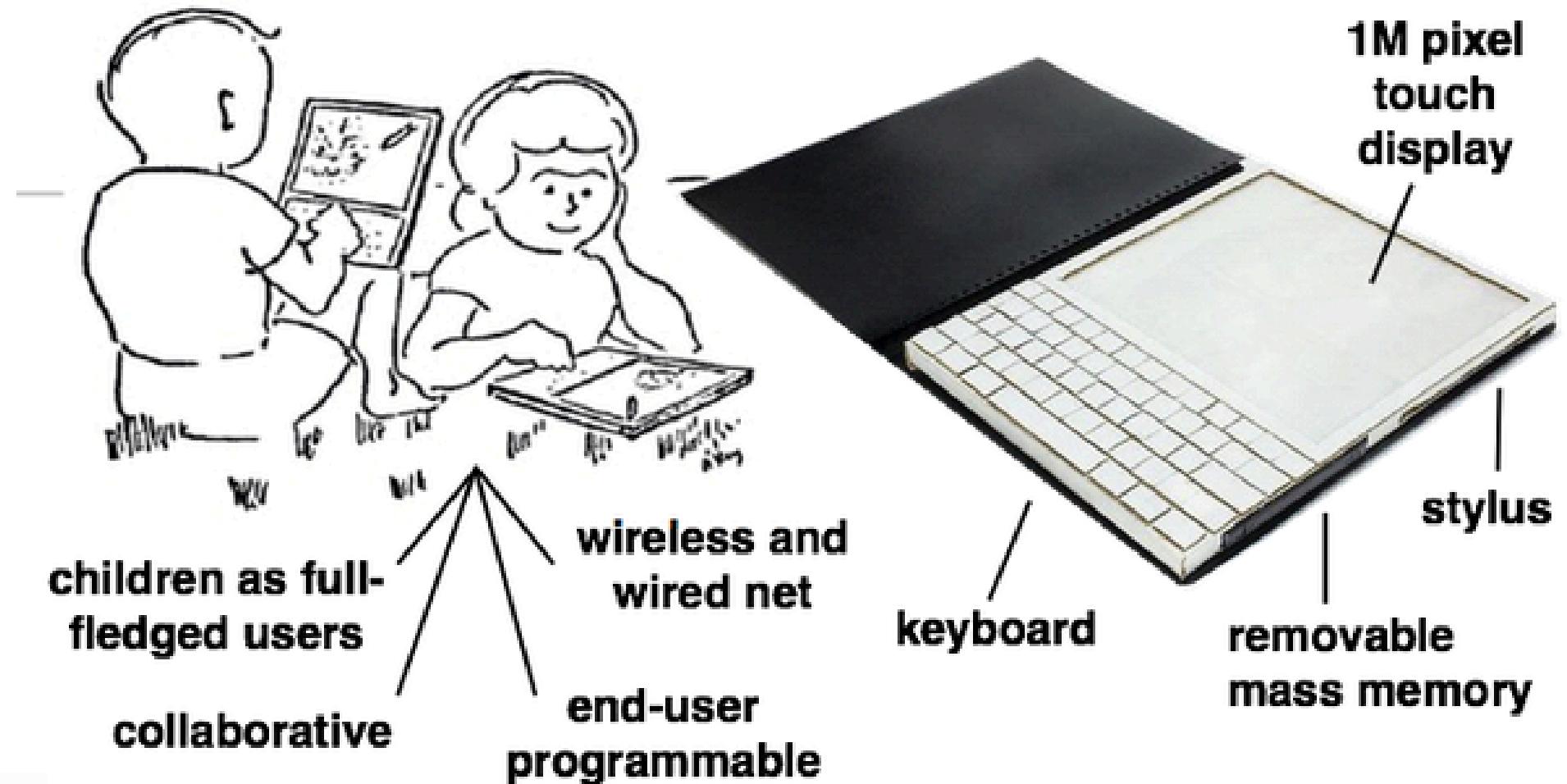
<https://craigbuchek.com/fifty>



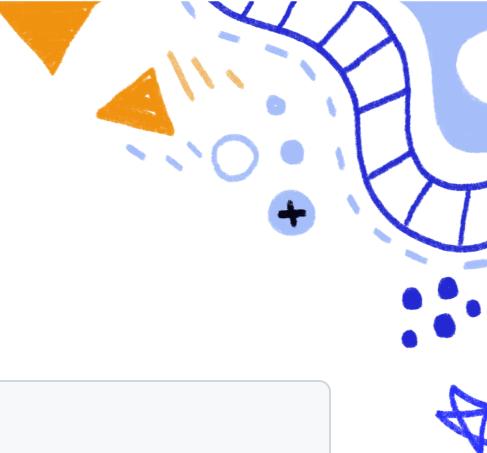
@CraigBuchek

The First Dynabook Idea – Fall 1968

Alan Kay

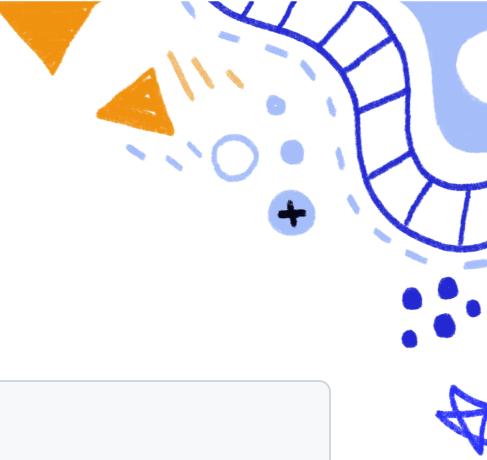






Smalltalk-71

```
to T 'and' :y do 'y'  
to F 'and' :y do F  
  
to 'factorial' 0 is 1  
to 'factorial' :n do 'n*factorial n-1'  
  
to 'fact' :n do 'to 'fact' n do factorial n. ^ fact n'
```



Smalltalk-72

```
to box var / x y size tilt
  (<(draw => (😊 place x y turn tilt. square size.))

to square
  (do 4
    (😊 go 50 turn. 90))

to square length
  (👉length ← :.
   do 4 (😊 go length turn 90))!

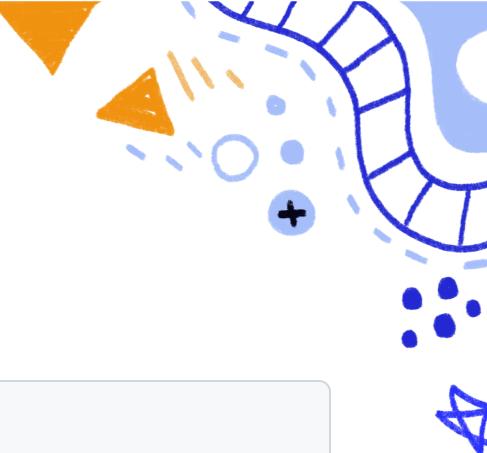
to poly sides size
  (👉sides ← :. 👉size ← :.
   do sides (😊 go size turn 360/sides))!
```

<https://craigbuchek.com/fifty>



@CraigBuchek

1973



Smalltalk-76

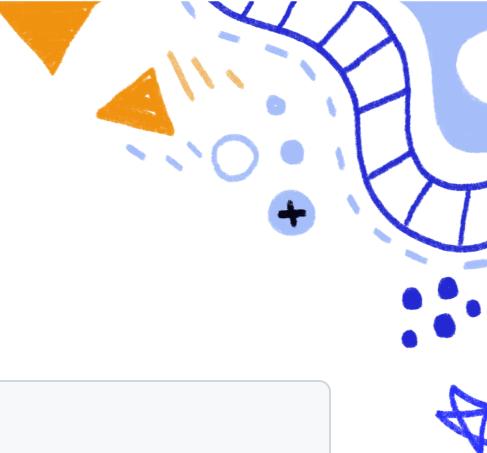
```
Class new title: 'Point';
    fields: 'x y'. "Cartesian coordinates"

"Access to fields"
x [↑x]
y [↑y]
x: x y: y

"Testing"
≤pt "return true if I am below/left of pt"
    [↑x≤pt x and: y≤pt y]

"Point arithmetic"
+ pt [↑Point new x: x+pt x y: y+pt y]
- pt [↑Point new x: x-pt x y: y-pt y]
* scale [↑Point new x; x*scale y: y*scale]
```

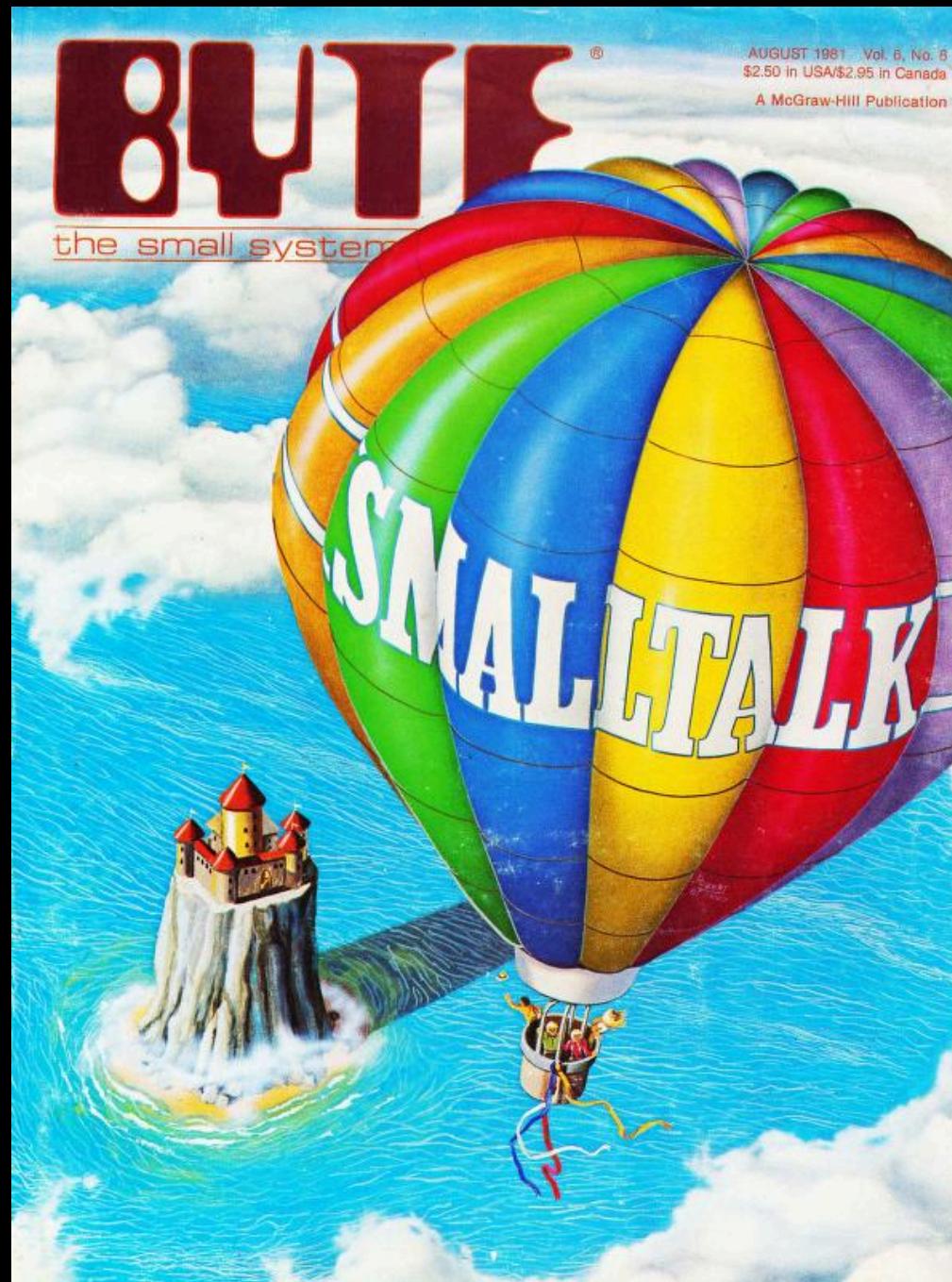




Smalltalk-80

```
x := OrderedCollection with: 1 with: 2 with: 3.  
x addFirst: 0.  
x select: [:a | a > 1]. "result: #(2 3)"  
x collect: [:a | a * a]. "result: #(0 1 4 9)"  
  
Object subclass: #MessagePublisher  
    instanceVariableNames: ''  
    classVariableNames: ''  
    category: 'Smalltalk Examples'  
publish  
    Transcript show: 'Hello, World!'  
multiply: i1 and: i2 by: n  
    | mul |  
    mul := i1 * i2.  
    ^mul * n "Return i1 * i2 * n"
```

<https://craigbuchek.com/fifty>

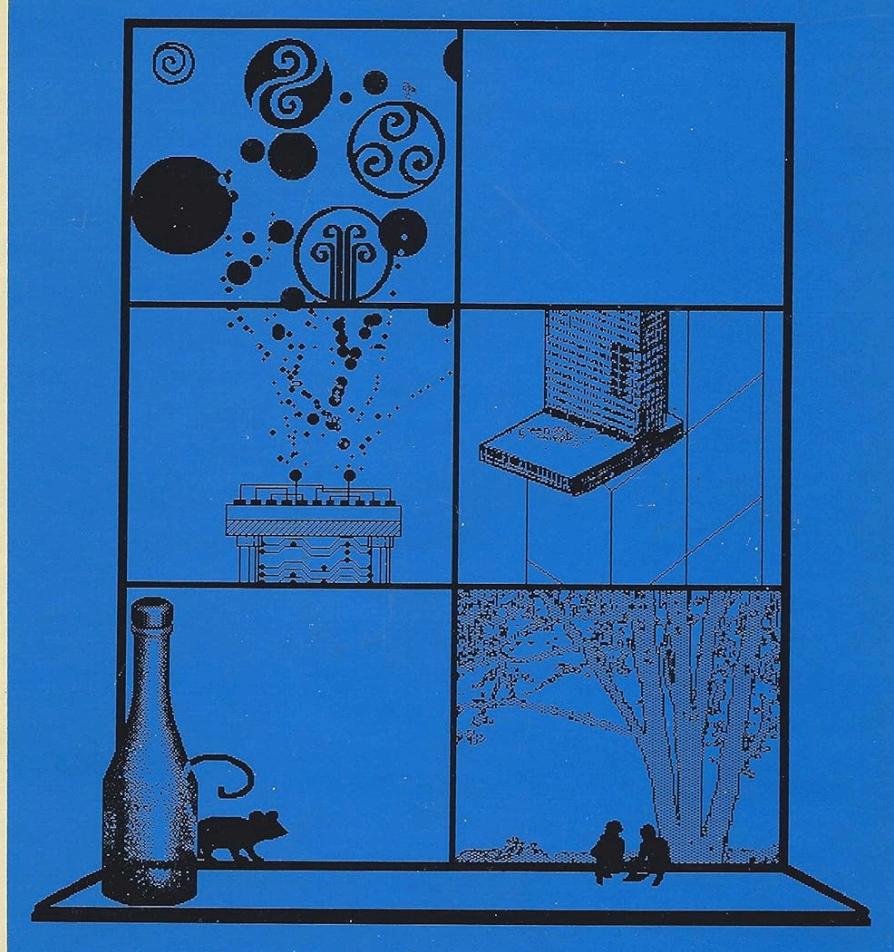


@CraigBuchek

1981

SMALLTALK-80

THE LANGUAGE AND ITS IMPLEMENTATION



Adele Goldberg and David Robson



@CraigBuchek

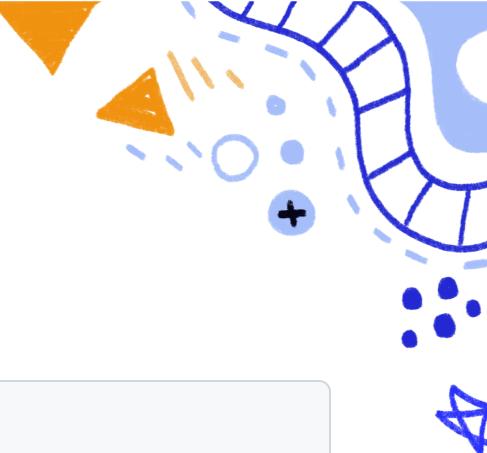
1983



1984



@CraigBuchek



Perl 4

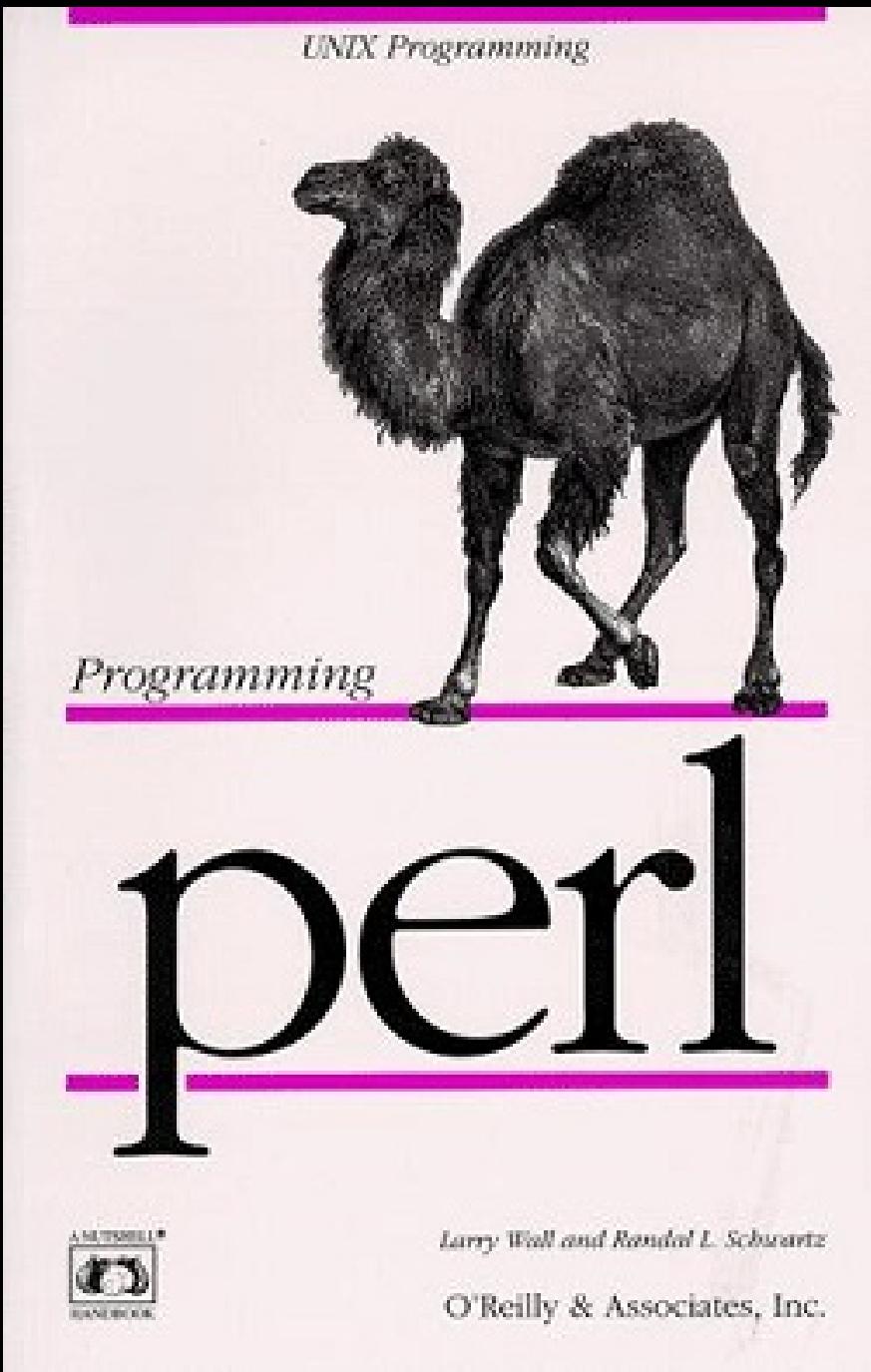
```
@numbers = (1, 2, 3, 4, 5);
$total = 0;
foreach $num (@numbers) {
    $total += $num;
}
print "Sum: $total\n";

$filename = "sample.txt";
open(FILE, $filename) || die "Cannot open $filename\n";
while ($line = <FILE>) {
    print $line;
}
close(FILE);

%ages = ("Alice" => 30, "Bob" => 25);
print "Alice is $ages{'Alice'} years old.\n";
```

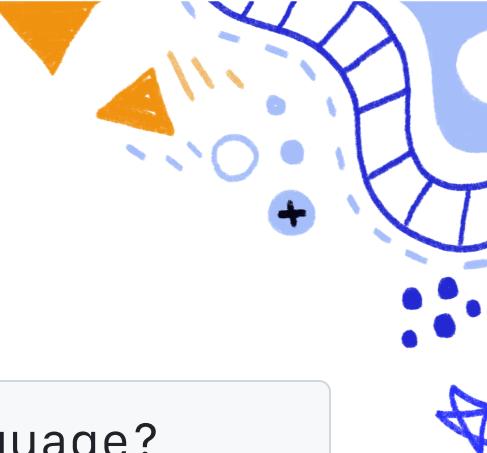


<https://craigbuchek.com/fifty>



@CraigBuchek

1991



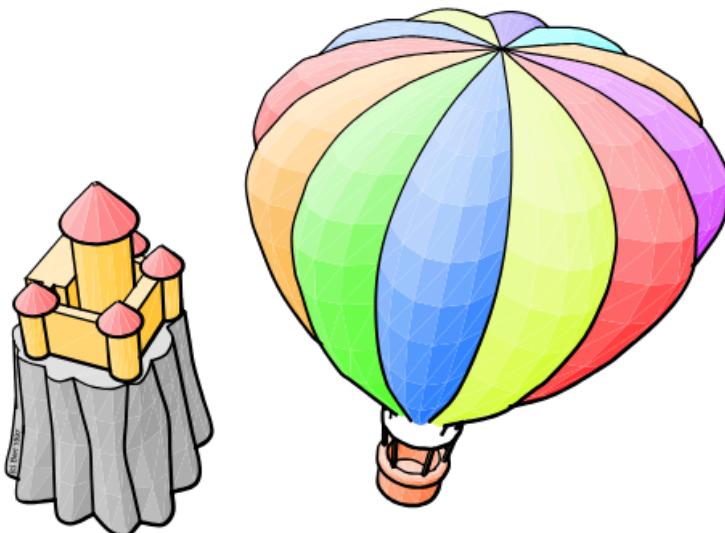
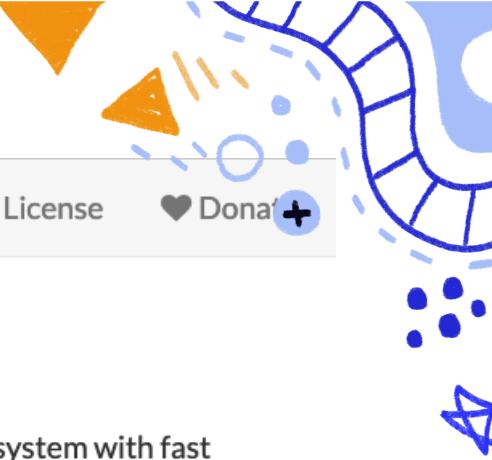
Pre-release Ruby

```
keiju> By the way, do you have considered the name of the language?  
matz> Well, Tish, if it's like shell enough.  
matz> But I want a smarter name.  
keiju> Toilet paper?  
...  
keiju> ruby  
keiju> a jewelry name after all  
matz> why jewel's name?  
keiju> perl  
matz> I see  
keiju> But, perl is related to a shell.  
...  
matz> What is your best up to now?  
keiju> I'm content with coral.  
matz> I thought ruby is cool as a codename, isn't it  
keiju> Well. Ruby is also good.
```

Ruby 0.95

Happy
BIRTHDAY





Welcome to Squeak/Smalltalk

Squeak is a modern, open-source Smalltalk programming system with fast execution environments for all major platforms. It features the Morphic framework, which promotes low effort graphical, interactive application development and maintenance. Many projects have been successfully created with Squeak. They cover a wide range of domains such as education, multimedia, gaming, research, and commerce.



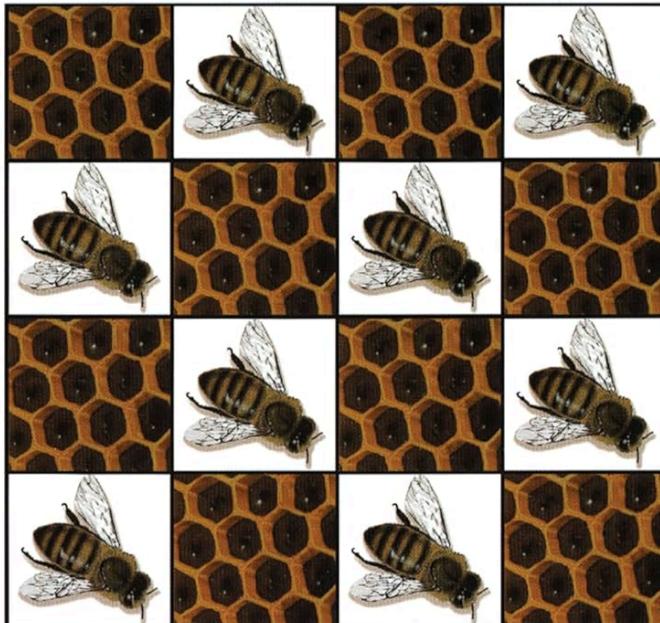
[All-in-One](#) · [Linux \(ARMv8\)](#) · [Release Notes](#) · [More...](#)



The screenshot shows the Squeak System Browser interface. The left pane displays a tree view of classes under the Kernel category, including Morph, event handling, and mouseDown. The right pane shows the details of the mouseDown class, including its implementors: AbstractResizerMorph, AlternatePluggableListMorphOfMany, BarometerMorph, BookPageThumbnailMorph, and BracketSliderMorph. A small pop-up window titled "Implementors of mouseDown: [88]" lists these implementors.



SMALLTALK BEST PRACTICE PATTERNS



KENT BECK



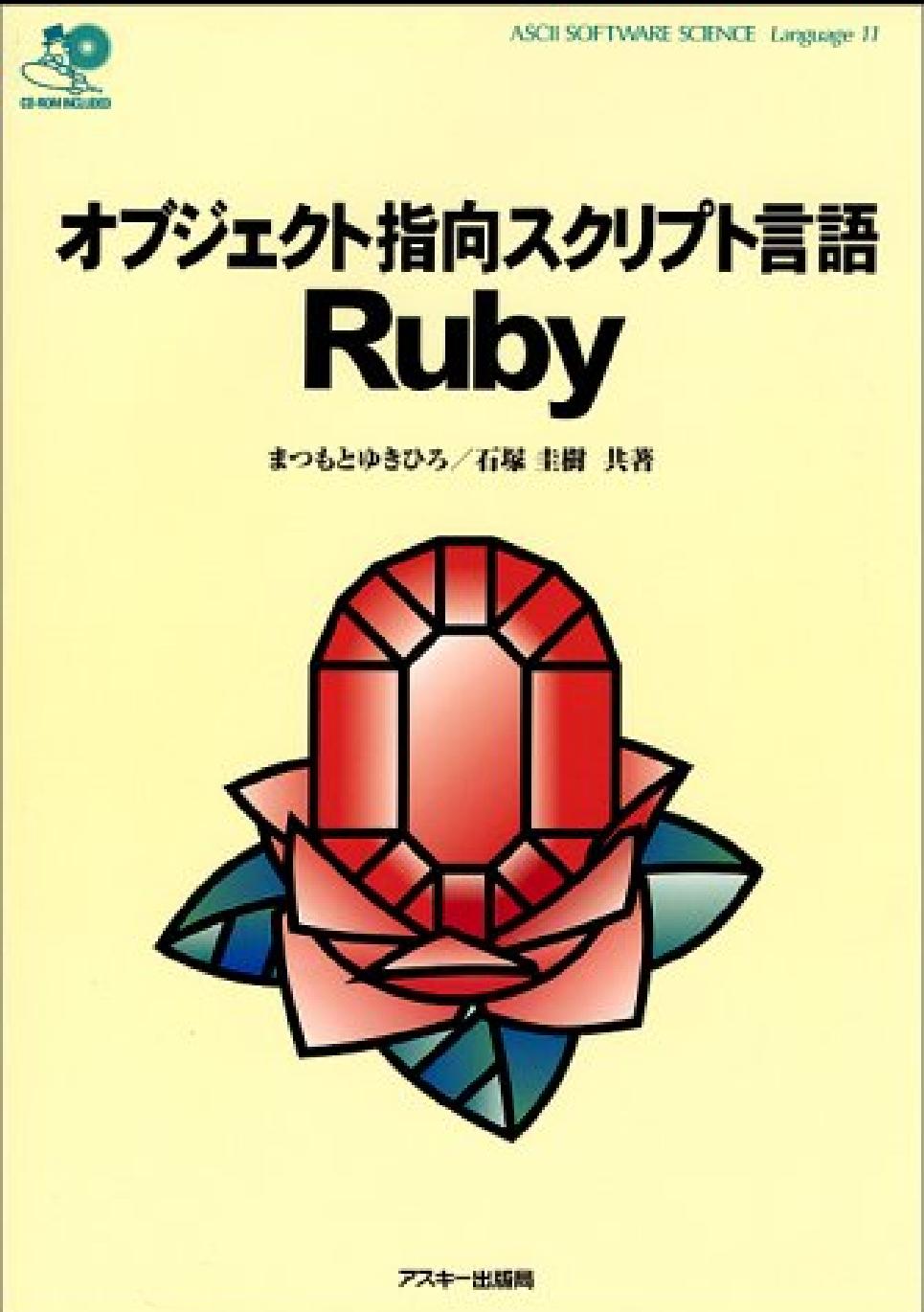
Ruby 1.0





Ruby 1.2

- First stable release





Programming Ruby



The Pragmatic Programmer's Guide

*Foreword by Yukihiro "Matz" Matsumoto
Creator of the Ruby Language*

David Thomas ♥ Andrew Hunt

<https://craigbuchek.com/fifty>



@CraigBuchek



2001





JRuby

- Runs on JVM
 - Faster
 - Java interoperability
 - Java libraries
- Jan Arne Petersen
- Charles Nutter
- Thomas Enebo



Ruby 1.8 (August 4)

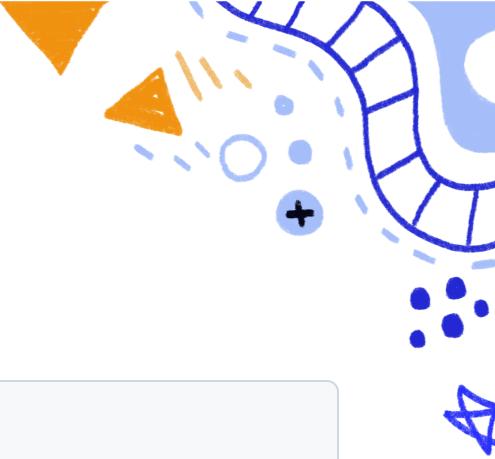
<https://craigbuchek.com/fifty>



2004



@CraigBuchek



Groovy

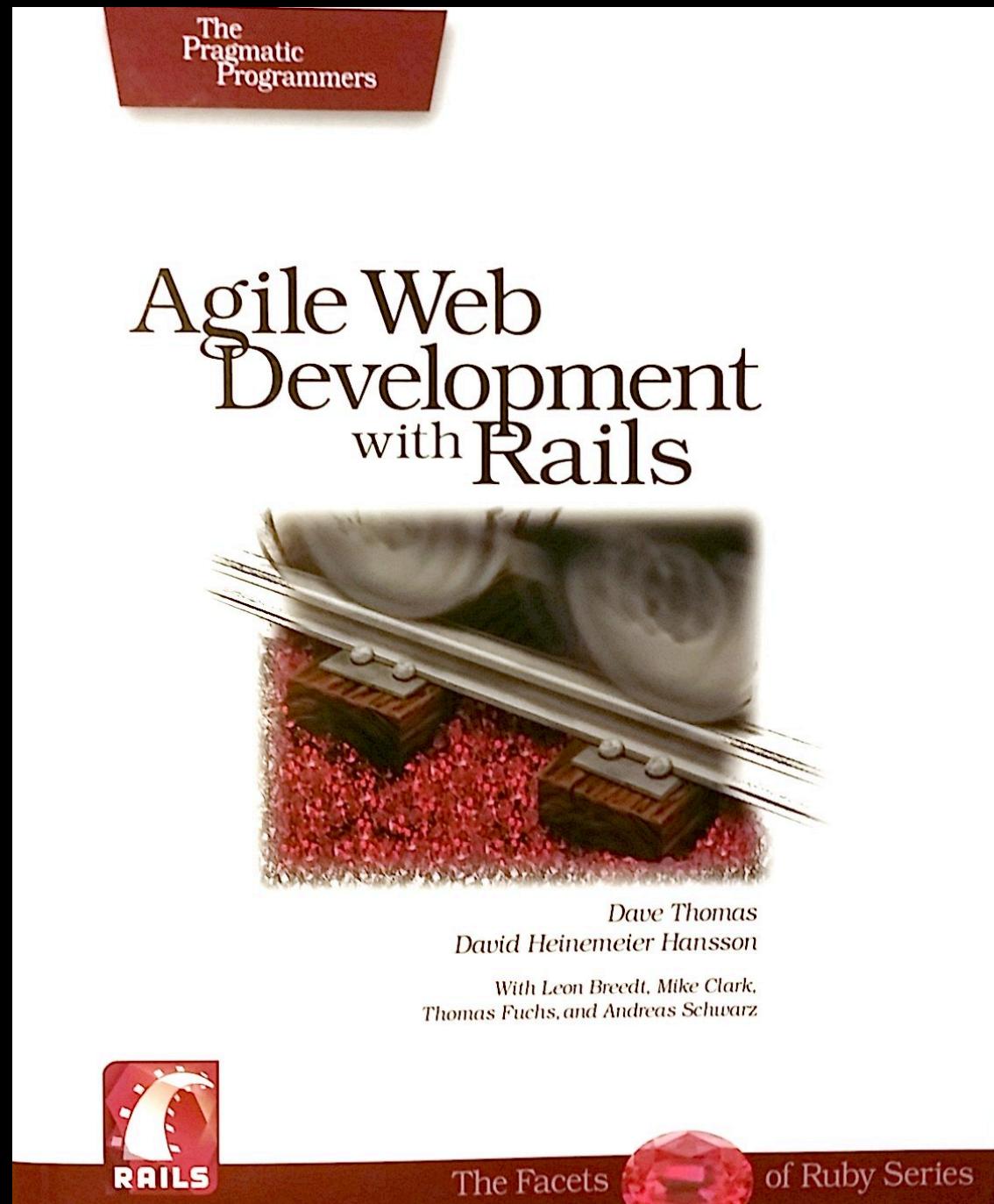
- JVM
- Concise syntax
- Dynamic typing
- Meta-programming
- DSLs
- Grails (2006)

```
languages = ["Ruby", "Java"]
languages << "Groovy"
languages.each { println "Language: $it"}

def say(msg = 'Hello', name = 'world') {
    "$msg $name!"
}
say
say()
say 'Hello'

class Test implements GroovyInterceptable {
    def sum(Integer x, Integer y) { x + y }
    def invokeMethod(String name, args) {
        System.out.println "Invoke method $name with args: $args"
    }
    def test = new Test()
    test?.sum(2,3)
    test?.multiply(2,3)

    class Foo {
        def propertyMissing(String name) { name }
    }
    def f = new Foo()
    assert f.boo == "boo"
```



<https://craigbuchek.com/fifty>

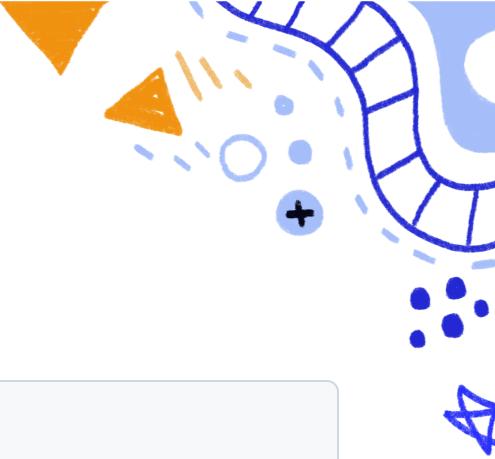


@CraigBuchek

2007

Ruby was a rediscovery of Smalltalk





Ruby 1.9

- YARV interpreter
- "Stabby" lambda
- Hash colons
- RubyGems
- Variables scoped to blocks
- Compatibility issues!

```
f = ->(a,b) { puts a + b }

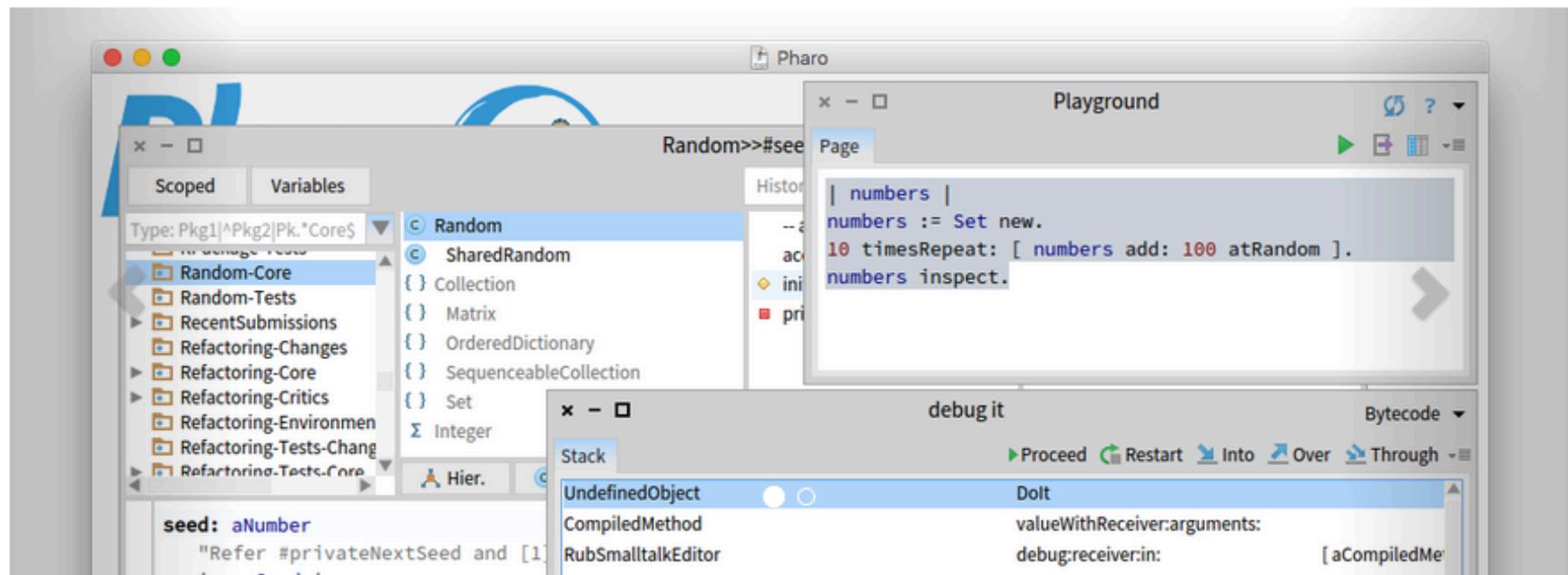
has_rockets = { :a => 1, :b => 2 }
has_colons = { a: 1, b: 2 }
has_colons == has_rockets

def foo
  1.times { x = 2 }
  puts x # Worked in 1.8; NameError in 1.9
end
```



The immersive programming experience

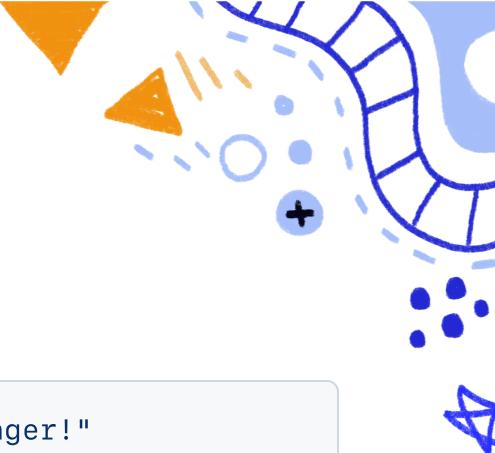
Pharo is a pure object-oriented programming language *and* a powerful environment, focused on simplicity and immediate feedback (think IDE and OS rolled into one).





Rubinius

- Written in Ruby
- Evan Phoenix
- Based loosely on Smalltalk-80 blue book



Elixir

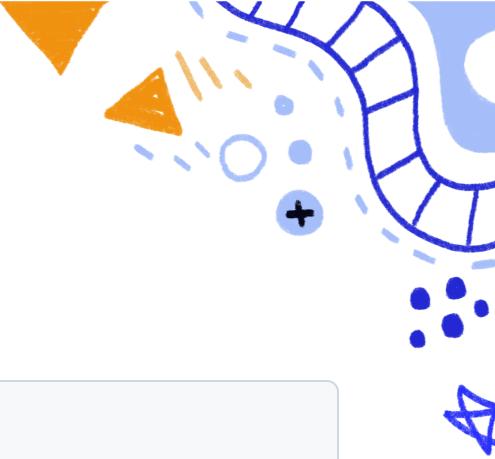
- FP
- Ruby-like syntax
 - DSLs
- Erlang VM
- Concurrency
 - Lightweight threads
- Fault tolerance
- Scalability
- Meta-programming

```
def hello(nil), do: IO.puts "Hello, Stranger!"  
def hello(name) do  
  IO.puts "Hello, " ++ name ++ "!"  
end  
  
defmodule User do  
  defstruct name: nil, email: nil  
end  
craig = %User{ name: "Craig", email: "craig@example.com" }  
craig = %{ craig | email: "craig@new-address.com" }  
  
defmodule Math do  
  def sum([]), do: 0  
  def sum([head | tail]), do: head + sum(tail)  
end  
  
[1, 2, 3, 4]  
|> Enum.map(&(&1 * 2))  
|> Enum.filter(&(&1 > 4))  
|> Enum.sum()
```



TruffleRuby

- Chris Seaton
- Fork of JRuby
- Graal JIT and VM



Ruby 2.0 (and 2.1)

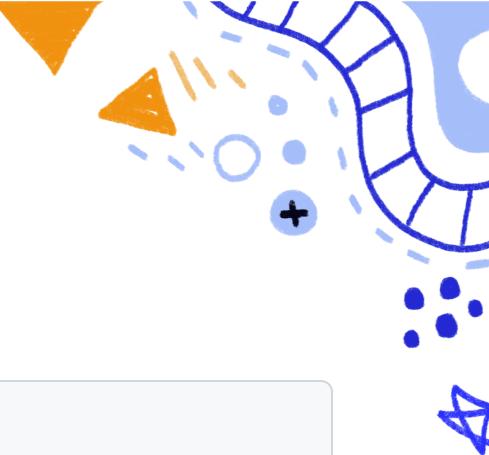
- Keyword arguments
- UTF-8
- Refinements (experimental)
- `def` now returns a symbol
- Rational literals

```
def foo(bar: 1)
  bar
end
def foo_old(*options) # Ruby < 2.0, we had to do:
  options[:bar] || 1
end

module M
  refine String do
    def reverse
      "reversed: #{super}"
    end
  end
end
using M
"hello".reverse # prints "reversed: olleh"

def bar(); end == :bar
private def baz(); end

1 // 2 == Rational(1, 2)
```



Crystal

- Effectively compiled Ruby
- Lose some meta-programming
- Typed, but minimal type hints
- Nil checking
 - No nil errors at runtime!
 - Eliminates class of bugs
- Fast
- Decent community/libraries

```
i = [] of Int32;

def maybe_upcase(name : String?)
  # Compile-time error: 'name' is a 'String?', not a 'String'
  name.upcase
end

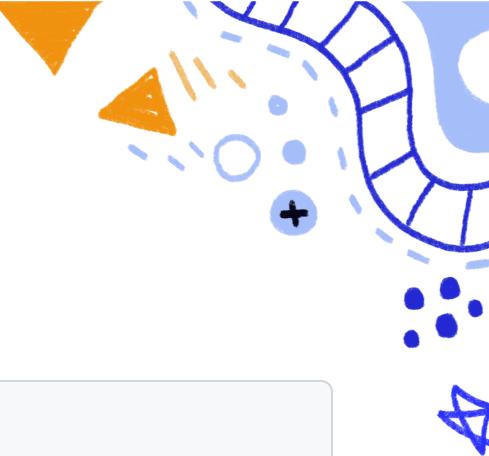
module Property
  macro property(name, type)
    def {{name.id}}
      @{{name.id}} : {{type}}
    end
    def {{name.id}}=(value : {{type}})
      @{{name.id}} = value
    end
  end
end

struct User
  extend Property
  property name, String
  property age, Int32?
  def initialize(@name : String, @age : Int32? = nil); end
end
craig = User.new(name: "Craig")
```



Ruby 2.2

- Incremental garbage collector
- Performance improvements



Rust

- Fast and memory-efficient
- Low-level
- Embedded
- Borrow checker
 - Ownership of mutable values
 - Eliminates a class of bugs

```
let numbers = vec![1, 2, 3, 4];
let sum: i32 = numbers
    .iter()
    .map(|x| x * 2)
    .filter(|x| x > &4)
    .sum();

fn divide(a: f64, b: f64) -> Result<f64, String> {
    if b == 0.0 { Err("Cannot divide by zero".to_string()) }
    else { Ok(a / b) }
}
match divide(10.0, 0.0) {
    Ok(result) => println!("Result: {}", result),
    Err(err) => println!("Error: {}", err),
}

let mut s = String::from("hello");
let len = calculate_length(&message); // Immutable borrow
let r1 = &mut message; // Mutable borrow
let r2 = &mut message; // Compile error! Can't mutable borrow twice
```



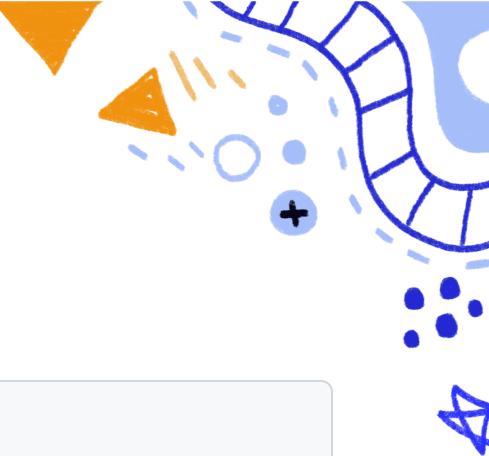
Ruby 2.3

- Safe navigation operator: `&.`
- `dig`
- Performance improvements



Ruby 2.4

- Unify Fixnum and Bignum into Integer
- Performance improvements



Ruby 2.5

- `rescue`, `ensure` don't require
`begin`, `end`
 - can use surrounding block

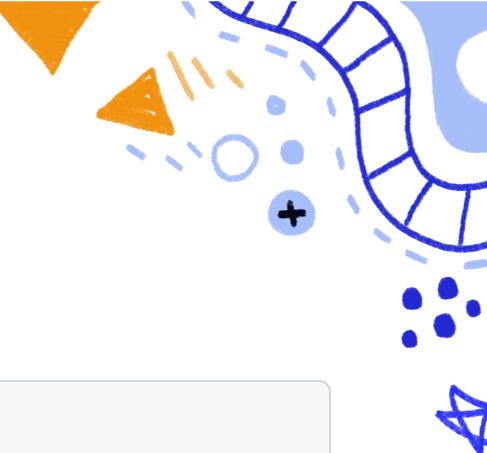
```
# Ruby 2.5
def foo
  f = File.open("file")
  raise "error"
rescue
  "rescued"
ensure
  f.close
end
```

```
# Ruby < 2.5
def foo
  begin
    f = File.open("file")
    raise "error"
  rescue
    "rescued"
  ensure
    f.close
  end
end
```



Ruby 2.6

- MJIT (experimental)

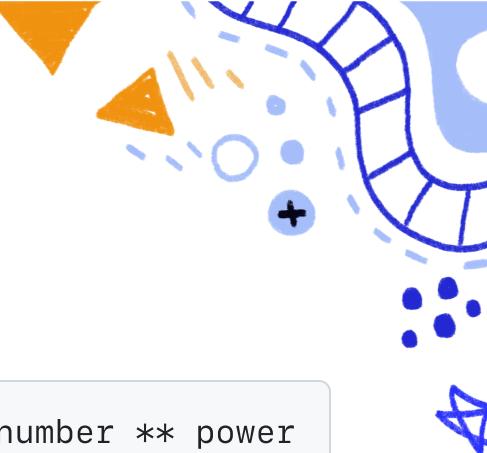


Ruby 2.7

```
people = [
    { name: "Craig", age: 53, children: [] },
    { name: "Alice", age: 30, children: [
        { name: "Bob", age: 2 }
    ]}
]

# Pattern matching (experimental)
people.each do |person|
  case person
  in {name: name, children: [{name: child, age: age}]}
    p "#{name} has a #{age}-year-old named #{child}"
  in {name: name, children: []}
    p "#{name} has no children"
  end
end
```

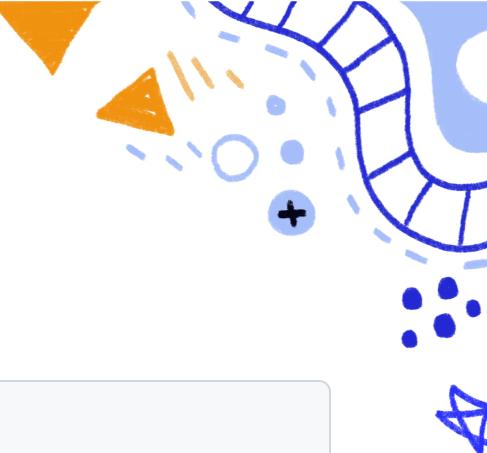




Ruby 3.0

- 3x3 performance improvements
 - 3 times faster than Ruby 2.0
 - JIT
- Ractors (experimental)
- RBS static analysis
- One-line pattern matching (experimental)
- Endless method definition

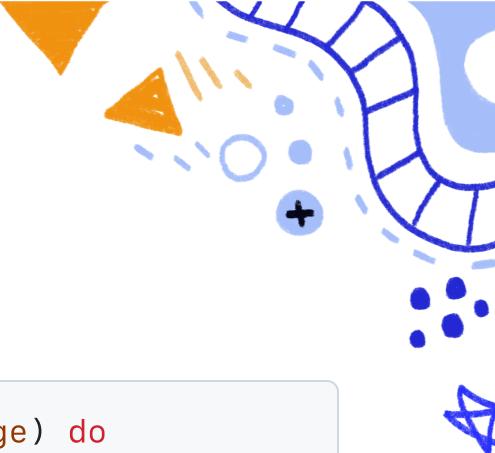
```
def raise_to_power(number, power) = number ** power
```



Ruby 3.1

- YJIT (experimental)
- `debug` gem (`rdbg`)
- `error_highlight` gem
- IRB autocomplete
- Pattern matching improvements
 - Pin operator (^)
- TypeProf

```
{x:, y:} == {x: x, y: y}  
foo(x:, y:) == foo(x: x, y: y)  
  
prime_pairs = Prime.each_cons(2).lazy  
prime_pairs.find_all{_1 in [n, ^(n + 2)]}.take(3).to_a  
#=> [[3, 5], [5, 7], [11, 13]]
```



Ruby 3.2

- Data class
 - immutable value objects
- WebAssembly
- YJIT rewrite
 - C -> Rust
 - no longer experimental

```
class Person < Data.define(:name, :age) do
  def initialize(name:, age:)
    super(name: name.to_s.capitalize, age: age.to_i)
  end
end
craig = Person.new(name: "craig", age: "53")
```



Ruby 3.3

- Prism parser
- pure-Ruby JIT compiler (RJIT)
- IRB improvements



Ruby 3.4

- Prism parser used by default
- Mutating string literals deprecated
- `it` as block parameter



Newer Ruby Features

- Refinements
- Data classes
- Pattern matching
- Concurrency
- Immutable objects



Ruby's Future

- Ruby will still get new features
 - Innovations
 - Borrowing/stealing
- Ruby will last a long time
- Something will replace Ruby



Ideas for Ruby's Future

- Ruby 4.0: To Infinity and Beyond
 - Bozhidar Batsov
- What If...?: Ruby 3
 - Eric Weinstein
- Steal This Talk: The Best Features Ruby Doesn't Have (Yet)
 - John Feminella
- Compiling Ruby
 - Kevin Deisz
- Keynote: Beyond Ruby 3.0 (RubyConf 2021)
 - Yukihiro Matsumoto





Take-aways

- Good solutions are rediscovered
- Ruby will last a long time
- Ruby will still get new features
- Something will replace Ruby
- TBD



“A change in perspective is worth 80 IQ points.”

— Alan Kay



“The best way to predict the future is to invent it.”

— Alan Kay



Thank You

- Attendees
- Viewers
- RubyConf organizers
- People that helped me with the talk
 - **Noel Rappin**
 - **Kerri Miller**



Come talk to me!

Ask me about:

- This talk (or others)
- Yoga
- Agile
- Job interviews

Tell me about:

- This talk (or others)
- Principal Engineer jobs
 - <https://resume.CraigBuchek.com>
- Hexagonal Architecture
- Cool language features



Contact Info

- Source: <https://github.com/booch/presentations/>
- GitHub: booch, boochtek
- Email: first.last@Gmail.com
- LinkedIn <https://linkedin.com/in/craigbuchek>
- Web page: <https://CraigBuchek.com>
- Everywhere else: CraigBuchek
 - Mastodon @ruby.social
 - Twitter