https://craigbuchek.com/fifty

RubyConf 2024

Fifty Years of Ruby

Craig Buchek

@CraigBuchek

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

— Alan Kay

https://craigbuchek.com/fifty

@CraigBuchek

https://craigbuchek.com/fifty
@CraigBuchek
1968

https://craigbuchek.com/fifty
@CraigBuchek

The First Dynabook Idea – Fall 1968

Alan Kay

https://craigbuchek.com/fifty
parc
A Xerox Company
3333 COYOTE HILL ROAD
@CraigBuchek
1969

Smalltalk-72

```
to box b | x y size tilt "x, y, size, tilt are instance variables"
  (ISNEW    »    (SELF undraw. 'size ← size + :. SELF draw. ⇑SELF)
  ⊂draw     »    (😀 place x y turn tilt. square size.)
  ⊂undraw   »    (😀 white. SELF draw. 😀 black)
  ⊂turn     »    (SELF undraw. 'tilt ← tilt + :. SELF draw)
  ⊂size     »    (⇑size))

to square size
  (poly 4 size)

to poly sides size
  (👉sides ← :. 👉size ← :.
    do sides (😀 go size turn 360/sides))

👉mybox ← box 0 0 50 0
mybox turn 45  "redraw box turned 45°"
```

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/1eft 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: [:n | n > 1].  "result: #(2 3)"
x collect: [:n | n * n]. "result: #(0 1 4 9)"
x inject: 0 into: [:a :n | a + n]. "result: 6"

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

https://craigbuchek.com/fifty

@CraigBuchek

1983

https://craigbuchek.com/fifty

@CraigBuchek

1984

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";
```


1991

https://craigbuchek.com/fifty

Ruby as an Idea

```
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.
```

https://craigbuchek.com/fifty

Ruby 0.95

@CraigBuchek

1995

https://craigbuchek.com/fifty

Community | Development | Documentation | Downloads | Features | Projects | Research | License | Donat

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.

macOS | Windows (x64 | Linux (x64)

All-in-One · Linux (ARMv8) · Release Notes · More...

Engage | Bugs | Gifts | Donate

@CraigBuchek

1996

https://craigbuchek.com/fifty

@CraigBuchek

1996

Ruby 1.0

Ruby 1.2

- First stable release
- `true` and `false` keywords
- `&&=` and `||=`
- `defined?`

https://craigbuchek.com/fifty

@CraigBuchek

1999

https://craigbuchek.com/fifty

@CraigBuchek

2000

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

- String interpolation of expressions
- Enumerable `all?`, `any?`
- Enumerable `inject`
  - AKA `reduce`
- Net::HTTP breaking changes
- Libraries (lots!)
- Hooks

https://craigbuchek.com/fifty

RAILS

@CraigBuchek

2004

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"
```


2004

https://craigbuchek.com/fifty

@CraigBuchek

2005

https://craigbuchek.com/fifty
@CraigBuchek
2007

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).

2008

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")
```

Rust

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

```rust
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 2.6

- MJIT (experimental)
- Bundler included
- Endless ranges
- Function composition operators
  - `<<` and `>>`
- `then`

```
"My long string"[8..]  # => "string"

f = proc{|x| x + 2}
g = proc{|x| x * 3}
(f >> g).call(3)  # Same as g(f(3))
(f << g).call(3)  # Same as f(g(3))

def filter_by_status(posts)
  return posts unless @status
  posts.where(status: Post.statuses[@status])
end

def order(posts)
  posts.order('published_at DESC')
end

Post.all
  .then(&method(:filter_by_status))
  .then(&method(:order))
```

Ruby 2.7

- Pattern matching (experimental)
  - `case` `in`
- Numbered block parameters
  - `_1`, `_2`, `_3`, etc

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

[1, 2, 3].each { puts _1 }
```

Ruby 3.0

- 3x3 performance improvements
  - 3 times faster than Ruby 2.0
  - JIT
- `end`-less method definition
- Ractors
  - Experimental
- RBS static analysis
- One-line pattern matching
  - Experimental

```
def pow(num, exp) = num ** exp

receiver = Ractor.new do
  message = Ractor.receive
  puts "received #{message}"
end
receiver.send("Hi!")  # prints "received Hi!"
received message is Hi!

sender = Ractor.new do
  message = 'Hi!'
  Ractor.yield(message)
end
sender.take  # => "Hi!"
```

Ruby 3.1

- Hash and keyword argument shorthand
- 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

```ruby
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
  - Not default yet
- RJIT: pure Ruby JIT compiler
- IRB improvements

Ruby 3.4

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

```
[1, 2, 3].each { puts it }
```

Ruby Today

- Over 30 implementations
- Innovations
- New features
- Community

Ruby's Future

- Ruby is alive and well!
- New features
  - Borrowing/stealing
  - Innovations
- Something will replace Ruby
- Ruby will survive

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

“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 gave me feedback
  - **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