



Better Ruby

NaCl
OSS Vision
Ruby Association

まつもとゆきひろ
Yukihiro "Matz" Matsumoto
@yukihiro_matz
@matz@ruby.social



Ruby is Great



Because the Creator is a Genius



No



Because I had a clear target



- Who Loves Programming
- Who Loves Freedom
- Who Wants to Be Productive



Who is He?



Me



We Love Ruby



Because Ruby is Great



Ruby is Great



Because We **Love** the Language



Thus We Use the Language



Thus We Form the Community



Thus We Organize Conferences



Thus We Create Gems



Thus We Created RubyGems™



Thus We Improve the Language



The Amount of Love is Unique in the Ruby Community



That Love made Nice&Friendly Community



I am very proud of **You**



Ruby is Great



Because It Keeps Improving



Ruby is OSS



No one is forced to use Ruby



We are Driven by Love&Passion



If We are Bored, We may Leave



Motivation is the Key



Distance between Users and Developers



We Need to Feed the Community



Something Attractive



Something Interesting



Something Beneficial



New Features are for Us



For Profit and Fun



- YARV (1.9:2007)
- Multi-Encoding + Unicode (2.0:2013)
- Refinement (2.0:2013)
- Generational GC (2.1:2014)
- Incremental GC (2.2:2015)



- MJIT (2.6:2018)
- Pattern Matching (2.7:2019)
- Object Compaction (2.7:2019)
- Ractor (3.0:2020)
- Single-line def (3.0:2020)



- YJIT (3.2:2022)
- Ruby.wasm (3.2:2023)
- Data class (3.2:2023)
- Prism (3.2:2023)

YARV (1.9:2007)



- Yet Another Ruby VM
- Stack Based Bytecode Machine
- by Koichi

Multi-Encoding + Unicode (2.0:2013)



- pre-Unicode
- Legacy Encoding
- Multilingualization (M17N)
- UCS / CSI

UCS / CSI

- Universal Character Set (≡ Unicode)
- Character Set Independent (M17N)

Refinement (2.0:2013)



- Structured Monkey Patching
- Not Used Much (\doteq Failure)

Generational GC (2.1:2014)



- Narrow Object Tracing
- Young / Old Generations
- Minor / Major Collections
- Better Throughput

Incremental GC (2.2:2015)



- Interleave GC and App
- Incremental Mark
- Incremental Sweep
- Better Latency

MJIT (2.6:2018)



- First JIT Compiler for Ruby
- Generate C, compile
- Then dynamic load

Pattern Matching (2.7:2019)



- Big Syntax Enhancement in Years
- Branching
- Assignment
- Decomposition

Ractor (3.0:2020)



- Concurrency / Parallel
- GVL Free
- For CPU Intensive Tasks

Fiber Scheduler (3.2:2022)



- Concurrency / non preemptive
- No Parallelism
- Async
- For I/O Intensive Tasks

Single-line def (3.0:2020)



- `def square(x) = x**2`
- April Fool's Day Joke

YJIT (3.2:2022)



- Realistic JIT Compiler
- Implemented in C then Rust
- Faster Each Year

Ruby.wasm (3.2:2022)



- Ruby in Browsers
- Opens New Horizon
- e.g. Mastodon in the Browser
- by Yuta Saito

Data class (3.2:2022)



- Immutable Struct
- Value Object
- by Victor Shepelev (zverok)

Prism (3.3:2023)



- Recursive Decent Parser
- Universal
- Error Tolerant



We Want to Make Ruby Even **Better**

Biggest Known Prime Number



$$2^{136279841}-1$$

Pre-3.4

p (2**136279841-1).to_s # => *Infinity*

3.4

`p (2**136279841-1).to_s.size` # => *41024320*



The Future



Ruby2 (2003)



Not Ruby2.0 (2013)



I once tried to restart Ruby



Just like Perl6 or Python3000



We collected RCRs



Ruby Change Request



Ruby2 Ideas



- Selector Namespace
- Keyword Arguments
- Method Combination
- Unicode Support
- Pattern Matching
- Packages
- JIT Compiler



I was too lazy to book-keep



I got tired of managing them
and gave up Ruby2 project



That was lucky for us

Ruby2 Ideas



- Selector Namespace
- Keyword Arguments
- Method Combination
- Unicode Support
- Pattern Matching
- Packages
- JIT Compiler

Implementations



- Refinement: part of Selector Namespace (3.0)
- Real Keyword Arguments (3.0)
- Method Combination (Module#prepend: 2.0)
- Unicode Support (1.9)
- Pattern Matching (2.7)
- JIT Compiler (2.6)



- Selector Namespace
- Packages



Namespace



Namespace = Better Refinement + Package



Isolated Scope of Names



Load Multiple Versions of Gems in an App



Ruby3.4 with Experimental Namespace



Final API is under discussion



Bad News



Namespace would not come in 3.4



Satoshi is too busy recently



Namespace will be one of the key concepts
for Ruby4.0




Other Improvements



Pipeline Operator Returns



Pipeline Operator (2.7)



```
# Ruby2.7 Pipeline Operator (Canceled)
a |> b(c) |> d()
```

```
# Considered as:
a.b(c).d()
```



Difference from Elixir




```
# Elixir Pipeline Operator  
a |> b(c) |> d()
```

```
# Considered as:  
d(b(a, c))
```




Difference from F#



```
# F# Pipeline Operator  
a |> b(c) |> d()
```

```
# Considered as:  
d(b(c, a))
```



Elixir Map

```
map(f, seq)
```



F# Map

```
map(seq, f)
```



Ruby Map

```
seq.map(&f)
```



Placeholder Idea



```
# Ruby3.x Pipeline Operator (?)  
a |> b(_, c) |> _.d()
```

```
# Considered as:  
b(a, c).d()
```



Automatic Typing



20 Years Ago (2004)



Dynamic Typing was Popular



- Compact
- Concise



10 Years Ago (2010-2014)



Static Typing has been Popular



- Earlier Error Detection
- Better Performance
- Better IDE Support



New Age of Productivity



With (Static) Type Information,



- More Precise Error Detection
- More Chances to Optimize



That Reminds Me of Manual Transmission





- More Precise Control
- More Fuel Efficient



But We See Less MT car Now





- AT is As Quick As MT
- AT is Efficient Enough
- We don't care how AT works



Automatic is the way to go



So in Programming Languages



My Prediction



In 10 (or 20) Years,



We see Mainstream Automatic Typing



Precise Enough



Full IDE Support



And (I Hope) Ruby will be in the Frontline



- RBS
- TypeProf
- Sorbet



- Ruby-lsp
- Rubocop



Or Whatever Future Technology



In Any Way,



Enjoy Programming



Ruby will Help You

Sponsors



- **NaCl**
- **OSS Vision** & Others
- **GitHub Sponsors**

Sponsors



- **Shopify** [NEW]
- **JetBrains** [NEW]
- **Ruby Community**



Thank you