Programming Languages and Ecosystems

John Leung 2015

"7 Languages and Ecosystems in 70 minutes"

Topics

1. Languages
2. Drawbacks
3. Alternatives

THE C PROGRAMMING LANGUAGE


THE JAPAN PRIZE FOUNDATION

C

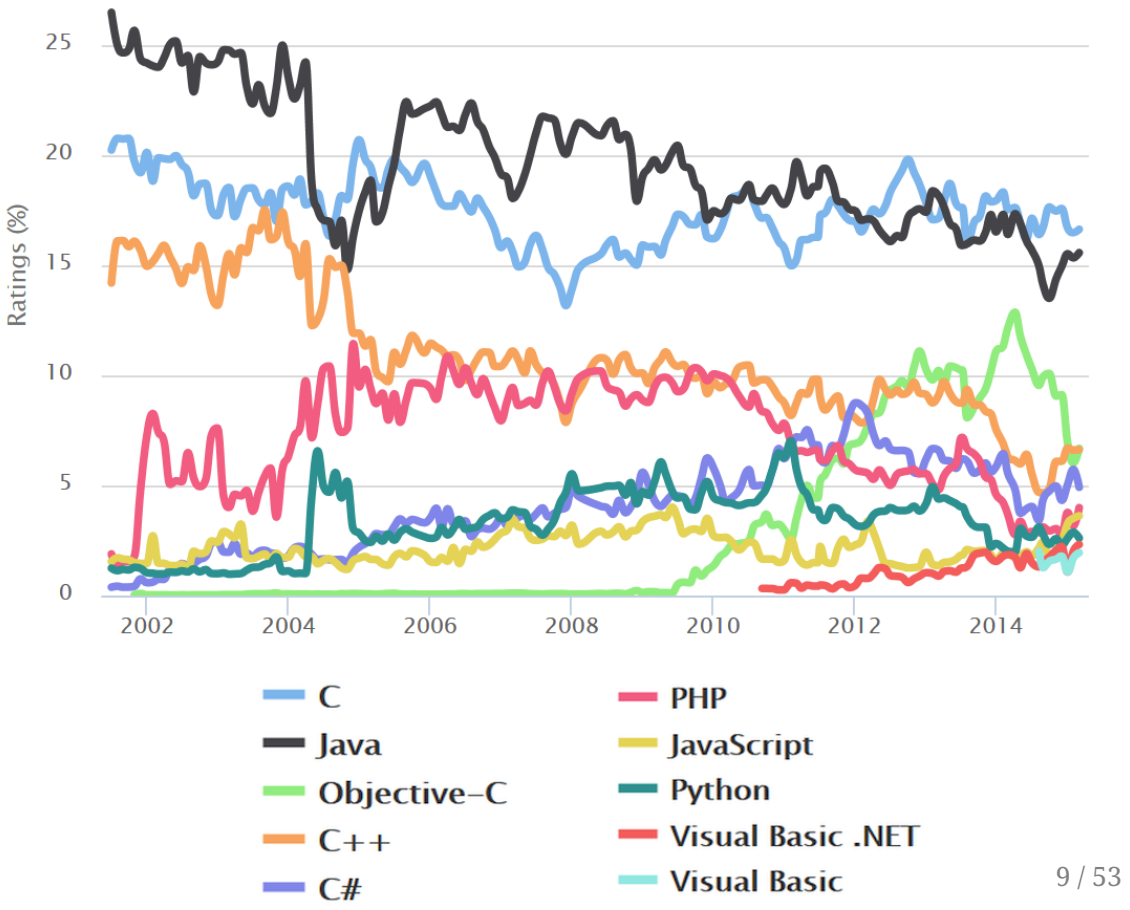
- developed by Dennis Ritchie 1969 - 1973 at AT&T Bell Labs
- general-purpose, imperative, statically typed
- constructs that map efficiently to machine instructions
- *"The new assembly"*
- typical applications: OS, embedded systems
- Influenced by:
  - Fortran (John Backus, 1957)
  - ALGOL (Committe of 8, including John Backus, '58, '60, '68)
  - PL/1 (IBM, *SHARE* Committee, 1964)
  - B (Ken Thompson and Dennis Ritchie, 1969)

C Standards

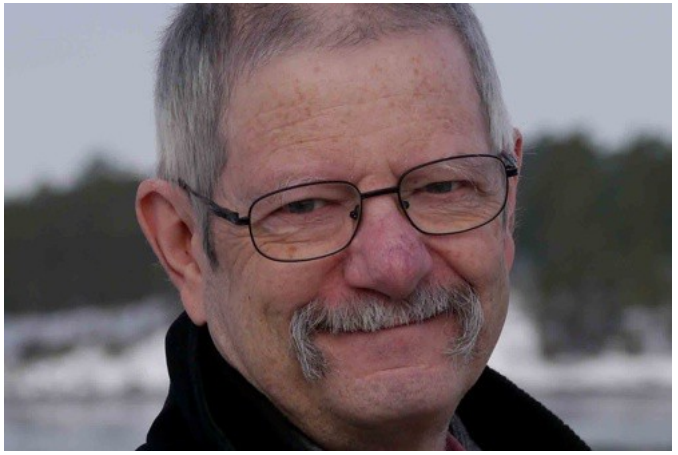
- K&R C, 1978
  - Brian Kernighan and Dennis Ritchie published *The C Programming Language* in 1978
  - the book served for many years as an informal specification of C, aka *K&R C*
  - large number of extensions
- ANSI C, 1983
  - large number of extensions led to the necessity of standardization
  - the *American National Standards Institue* (ANSI) formed a committee in 1983 to standardize C
  - ANSI C standard adopted by ISO in 1990

C Standards

- C99, 1999 - added:
  - inline functions
  - `long long int` and `complex` data types
  - variable length array, flexible array members
  - better floating point support
  - macros of variable arity
  - `// single line comments`
- C11, 2011 - added:
  - type generic macros
  - anonymous structures
  - better Unicode support
  - atomic operations
  - multi-threading
  - bounds-checked functions
  - better C++ compatibility


25
20
15
10
5
0
Ratings (%)
2002
2004
2006
2008
2010
2012
2014
C
Java
Objective-C
C++
C#
PHP
JavaScript
Python
Visual Basic .NET
Visual Basic


ERLANG



Erlang

- developed by Joe Armstrong, Robert Virding, and Mike Williams in 1986
- general-purpose, concurrent, functional programming language and runtime system
- dynamically typed, automatic memory management
- originally a proprietary language within Ericsson (*"Er"*), open sourced in 1998
- designed to support distributed, fault-tolerant, soft real-time, highly available, non-stop applications
- supports *hot swapping* (code can be changed without stopping the system)
- active, 18.0, 18.1 released in June, Sep 2015

Erlang Processes

- main strength in concurrency via light weight processes
- Erlang processes are not OS processes nor OS threads
- like OS processes, Erlang processes share no state
- processes communicate via share-nothing message passing, removing need for explicit locks
- processes are very light weight, only 300 words overhead
- millions of processes can be created without much degrade in performance

Erlang - language for fault tolerant distributed systems

- in GPRS, 3G, LTE mobile networks worldwide
- Yaws - web server
- Riak, CouchDB, SimpleDB - distributed databases
- ejabberd - XMPP server powering Facebook Chat
- Chef - configuration management tool re-written from Ruby
- RabbitMQ - Message Queuing Protocol implementation
- RPC proxies to ruby processes for GitHub
- WhatsApp, Goldman Sachs HFT programs
- Call of Duty, League of Legends, Battlestar Online game servers


python™


Goog

Python

- developed by Guido van Rossum, released in 1991
- general-purpose, object-oriented, high-level scripting language
- design philosophy emphasizes code readability
- dynamically typed, automatic memory management
- *"The new BASIC"*
- *"Batteries Included"*
- active, 3.5 released in Sep 2015
- langauge proposals via PEPs (Python Enhancement Proposals)

PEP 20 (The Zen of Python)

- Beautiful is better than ugly
- Explicit is better than implicit
- Simple is better than complex
- Complex is better than complicated
- Readability counts

Python Implementations

- CPython - official implementation, written in C
- PyPy - much faster JIT runtime
- Jython, IronPython - JVM, .NET runtimes
- Cython - python-like language with cdef type declarations, compiles to C++
- Numba - JIT LLVM compiler via declarator signatures

Python - language for Science and Machine Learning

- NumPy - matrix library that uses BLAS, allowing matrix calculations to be close to speed other languages using BLAS, even closer if used in PyPy
- SciPy, scikit-learn, BioPython, Sage, iPython notebook, Pandas
- many deep neural network libraries
- best supported language for OpenCV
- NLTK
- "machine learning" in github: 3337 in python, 2264 in Matlab, 1050 in R, 1006 in Java, 496 in C++

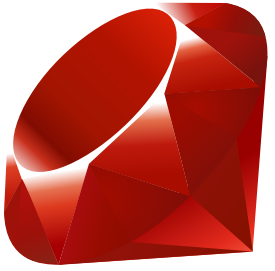

Lua

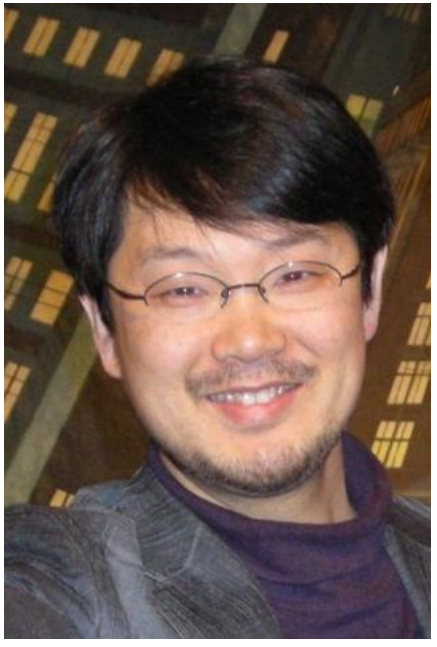
Lua

- created by Roberto Ierusalimschy, Luiz Henrique, Waldemar Celes in 1993
- designed to be a general purpose scripting language with embeddability and extensibilty as primary goals
- lightest weight (180kB for full reference interpreter) scripting language
- first-class functions, GC, closures, proper tail calls, coroutines
- ~50% - 3x faster than python in general benchmarks
- dynamically typed,
- single data structure: table
- simple C API

Lua - language for Games, UI, and Embedded Systems

- Mods, including Minecraft, Half-Life 2, many others
- World of Warcraft UI subsystem
- Game Engine for Corona, Moai, and many others
- full or large portion of many games
- over 40% of Adobe Lightroom is written in Lua
- eLua - embedded software applications builder
- OpenResty - web server bundling Nginx core





Ruby

- created by Yukihiro Matsumoto (松本行弘) "Matz", released in 1995
- general purpose, object oriented, dynamically typed scripting language
- Why? "Didn't like Perl because it smells like a toy language. Didn't like Python because it's not a true object oriented language. I wanted a genuine object oriented, easy to use scripting language" - Matz 1999
- everything is a true object

```
4.times do |i|
  puts i
end
```

Ruby - language for web applications with

Rails

(and DSLs, web (Github, old Twitter), and other usages where speed isn't critical)


JS

Javascript

- developed in 10 days by Brendan Eich in May 1995
- originally called LiveScript, renamed to JavaScript when shipped in Netscape 2.0 in Sep 1995
- designed to be a lightweight interpreted language used in a browser
- part of to the ECMA standard in June 1997, but no official "governing" body, hence no official logo
- fun fact - *JavaScript* is a trademark of Oracle
- predicted by Steve Yegge in 2007 to be *"The Next Big Langauge"* (web2.0 just coined 1+ years ago and ajax just a year ago)
- cemented its dominance with introduction of ajax (2005), jQuery (2006), node.js (2009)

Javascript - language for web and mobile

- *"The new C"*, *"The web assembly"*
- now a *"platform"* - languages that compiles to javascript: over 100
- a real platform - WebAssembly (supersedes asm.js) - ports C code without modifications with less than 50% performance hit
- web worker - spawn workers that unblocks the UI thread
- server application and server with node.js
- native mobile applications with Titanium and React native
- non-native mobile applications with phonegap and many other frameworks


CoffeeScript

CoffeeScript

- developed by Jeremy Ashkenas in 2009
- (also creator of Backbone.js and Underscore.js)
- syntax heavy borrows from Ruby and Python
- most adapted alt-js language

Drawbacks and Alternatives (7 More Languages)

C Drawbacks

- No runtime checking
- No strict type checking
- No easy object oriented programming support
- No exception handling support
- Not easy
- Solution: C++?, D?


nimrod

Nim

- created by Andreas Rumpf, v0.6 release 2008
- high level, statically typed, imperative language
- fast compilation
- compiles to C and javascript
- supports both automatic and manual memory management
- macros, templates, inline, iterators, generators
- universal function call syntax
- fast GC supporting soft real time systems and pluggable GC module (3 supported)

Erlang Drawbacks

- Slow for heavy computations
- weird syntax
- String manipulation

elixir

Elixir

- created by José Valim in 2012
- functional, concurrent, general purpose language
- compiles to bytecode that runs on BEAM - the Erlang VM
- same fault tolerant, soft real time, non-stop distributed applications capabilities
- can call Erlang functions and vice versa
- builtin macros, pattern matching everything is an expression
- lazy and async collections with streams
- biggest usage in Phoenix Web Framework

Python Drawbacks

- Still not fast enough for non-matrix computations
- Not functional enough - Eg. lambda functions are limited and side effect functions do not return

Solutions

- PyPy - for speed
- Mochi - for functional programming - Yasushi Itoh, first commit Nov 2014

Lua Drawbacks

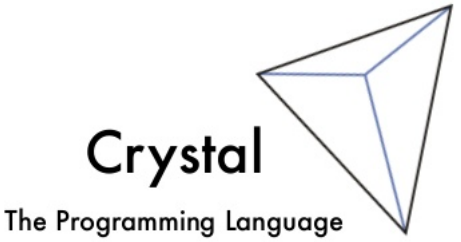
- indexes start at 1
- both speed and syntax are good, but can still be better

Solutions

- LuaJIT - for speed
- MoonScript - for coffeescript-like syntax - created by Leaf 2011

Ruby Drawbacks

- slow



@KamilLelonek

Crystal

- created by Ary Borenszweig, release in 2013
- Statically type-checked but without having to specify the type of variables or method arguments
- Compile to efficient native code
- able to call C code by writing bindings to it in Crystal
- Have compile-time evaluation and generation of code, to avoid boilerplate code
- gaining momentum, 3.5k github stars

JavaScript Drawbacks

- truthy evaluation and type coercion not make much sense
- floats and integers have same type Number (64 bits, 53 bits max for integers)
- typescript and flowtype for typing

Livescript

- derived from coffeescript
- adds many features to assist in functional style programming
- supports pipes, list comprehension, partial application, function composition
- better require of files and functions