Pontifícia Universidade Católica do Rio de Janeiro

Lua versus Javascript: Why do we need multiple languages?

Roberto Ierusalimschy
PUC-Rio, Lua.org

Disclaimer: I am hardly an unbiased observer to compare both languages.

On the other hand, several big companies support (or pretend to support) Javascript, so I will hardly affect the “average bias” of this discussion.

Why am I here?

- “Wikipedia Chooses Lua As Its New template language”
    - Slashdot: News for nerds, Feb 1, 2012
- “Meet Wikipedia, the Encyclopedia Anyone Can Code”
    - Wired, Mar 19, 2013

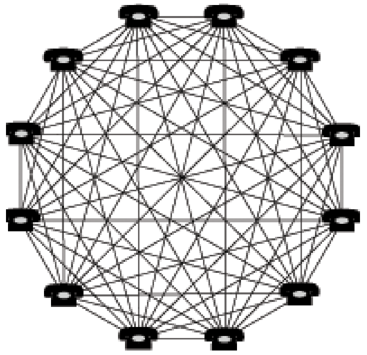
“Why not use javascript, the web's standard scripting language?”

Wired, Mar 19, 2013

Several Arguments in Favor of Javascript

- “The idea of having a single scripting language that can be used by our tools/gadgets community is certainly appealing, and the ever-growing ecosystem around both client-side and server-side JS is as well.”
- “I've settled on Lua [...] despite its relative unfamiliarity among our users ...”

Network Effect



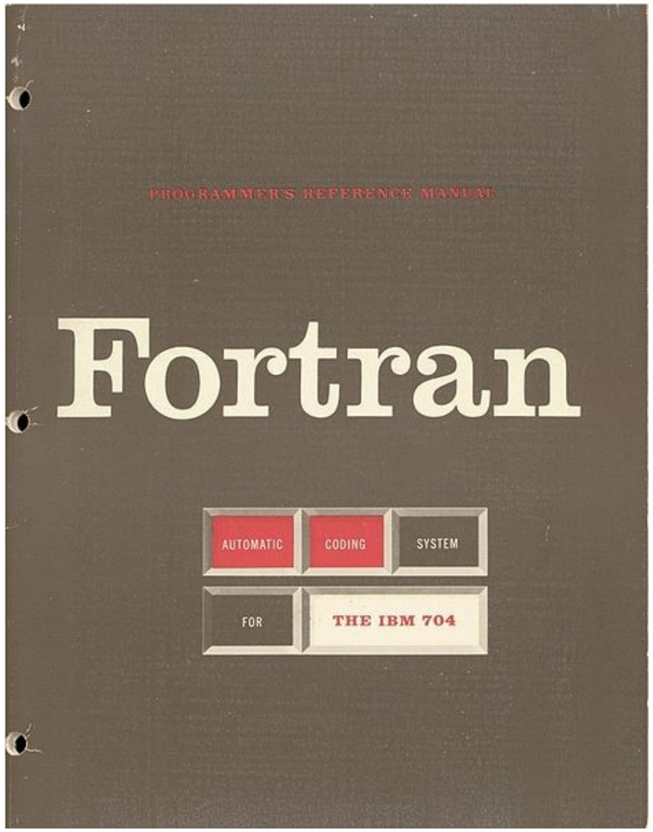
- I should learn the most popular language.
  - to find a job
- I should use the most popular language.
  - to find programmers
- I should use the most popular language.
  - to find libraries, documentation, etc.
- I should use the language I am already using.
  - to not have to learn a new language

Why do we need multiple languages?

Wouldn't the world be better if everybody used the same programming language?

Fact: if we all settled for arguments like “it would be good to use the same language as X” or “language X is better known than Y”, we would all be using Fortran today.

Why are we not?



Very Simplified Story

- FORTRAN (FORmula TRANslator) was specialized for scientific computation.
- COBOL (COmmon Business-Oriented Language) was created, specialized for commercial-oriented sofware.
- There were also Algol and Lisp...

But why do we need multiple languages?

PL/I

- A truly general-purpose programming language
  - *“FORTRAN VI is not intended to be compatible with any known FORTRAN IV. It includes the functional capabilities of FORTRAN IV as well as those capabilities normally associated with 'commercial' and 'algorithmic' languages.”*
- Championed by IBM
  - this was 1964
- Why was PL/I not a big success?
  - compare its usage with Fortran and Cobol

A programming language is not a pile of features.

The “Subset Fallacy”

- I can use only the features I like/need.
- Features that I do not use do not affect me.
- The bigger the pile, the better the language.

The “Subset Fallacy”

- Bugs frequently involve parts that you think you are not using
  - so you should know them for debugging
- Other people's code involves parts that you do not use
  - so you should know them for maintenance

The “Subset Fallacy”

- Compilers and interpreters must support the entire language
  - extra features make them larger, more complex, and slower
  - many features hamper optimizations

The design of a language involves many trade-offs, and we need explicit goals and priorities to settle these trade-offs. Different languages choose different goals, and therefore settle these trade-offs in different directions. Like any tool, no language is good for everything.

Some PL Trade-offs

- Safety versus flexibility
  - type checking
- Readability versus conciseness
  - Perl, regexs
- Performance versus abstractions
  - Assembler, automatic memory management

Some PL Trade-offs

- Flexibility versus good error messages
  - Haskell
- Libraries versus portability
  - Java (No, Java ME is not Java.)
- Simplicity versus expressiveness

Back to Lua and Javascript...

Several Unexpected Similarities

- All numbers are `double`
- Functions are first-class values with lexical scoping
  - but Lua has proper scope for variables
  - Functions in Lua are not objects
- Function are always anonymous
  - function definition is an assignment to a variable
- Objects are associative arrays
  - `t.x` as sugar for `t["x"]`
- Prototype-based OO

Several Unexpected Similarities

- Arrays as objects
  - that is, associative arrays
  - but Lua uses numbers as keys
- Expressive object constructors (“literal objects”)
- Optional semicolons
  - but Lua does not use semicolon insertion; the syntax itself dispenses semicolons

(Lua predates Javascript by two years)

But Several Important Differences

- Lua has much simpler objects
  - no property attributes, no object attributes, no getter-setter methods, no new
- Javascript has a C-like syntax
- Lua has a well-defined FFI
- Javascript has more syntactical constructs
  - exception handling, patterns

But Several Important Differences

- Javascript has built-in objects and more pre-defined functions
- Lua has semantics for garbage collection
  - finalizers and weak references
- Lua has coroutines

Why the Similarities?

- Simplicity
  - doubles
  - objects as associative arrays
  - arrays as associative arrays
- “Fashion” (plus Scheme and Self)
  - functions as first-class values
  - Functions are always anonymous
  - optional semicolons
  - prototype-based inheritance

Why the Differences?

- Better design :)
  - Lua has had the luxury of time to mature
- But mainly, different goals!

Javascript Goals

“ECMAScript was originally designed to be a Web scripting language, providing a mechanism to enliven Web pages in browsers and to perform server computation as part of a Web-based client-server architecture. ECMAScript can provide core scripting capabilities for a variety of host environments, and therefore the core scripting language is specified in this document apart from any particular host environment.”

ECMAscript language specification, 5.1 edition

Javascript Goals

- Despite the “can provide”, there is no sign of a strong commitment with this goal of being a scripting language for different hosts.
- Javascript is not the *scripting* language of the browser; it is the (only) programming language of the browser.

Javascript Goals

- Being THE language makes its goals quite different from those of a scripting language
  - better support for programming in the large
  - better fit as a target language for other language compilers
  - small emphasis on the specification of a FFI and the behavior of host objects

Lua Goals

“Lua is intended to be used as a powerful, lightweight, embeddable scripting language for any program that needs one.”

Lua 5.2 reference manual

Lua Goals

- Embeddability
  - *Lua is a library, not a language*
  - good integration with a host language
- Small size
- Simplicity
  - leave the complexity to the host

Scripted by Lua

Embedded Systems

TVs (Samsung), routers (Cisco), keyboards (Logitech), printers (Olivetti), set-top boxes (Verizon, Ginga), M2M devices (Sierra Wireless), telecommunications (Huawei), calculators (Texas Instruments), etc.

Configuration

Adobe Lightroom, Wireshark, Snort, Nmap, VLC Media Player, lighttpd, LuaTeX, Flame, …

`http://en.wikipedia.org/wiki/Category:Lua-scriptable_software`

Scripting the Internet of Things

m2m
eclipse.org

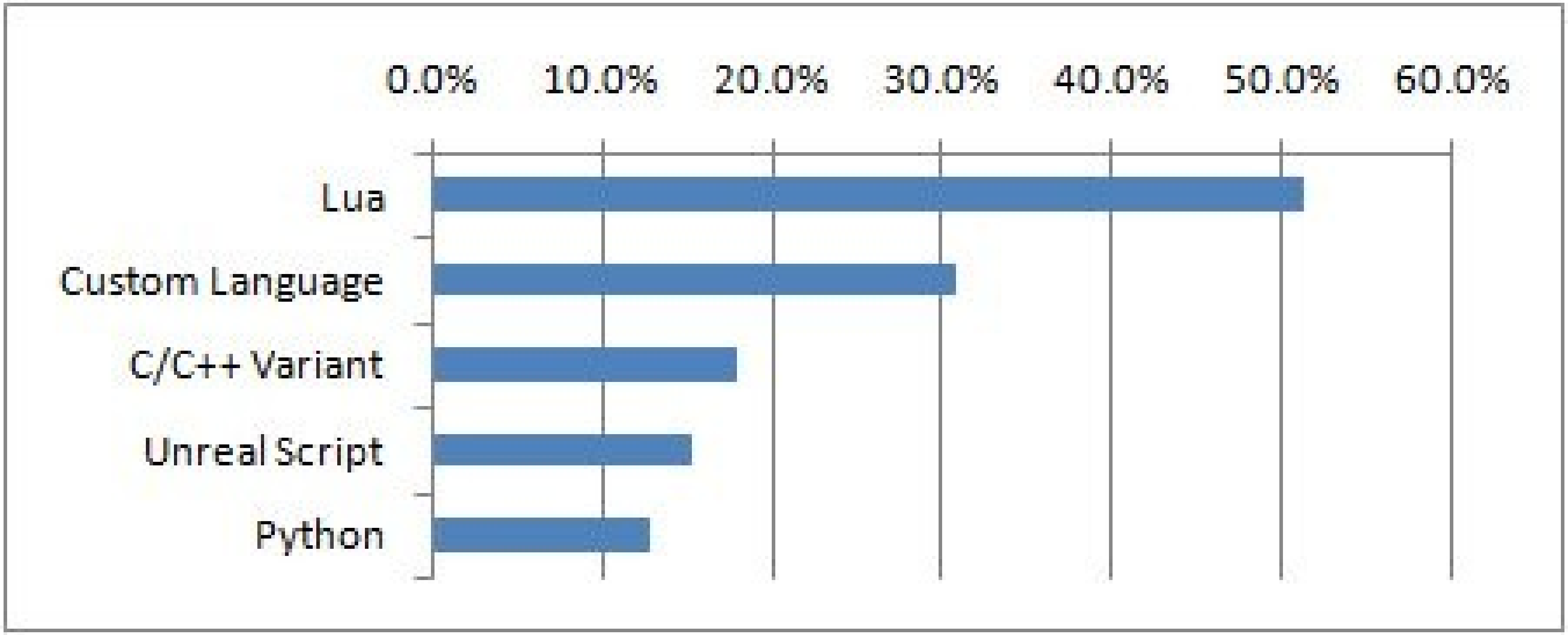
koneki

paho

mihini

Scripting Games

- The Engine Survey (03/02/09,Gamasutra):
- What script languages are most people using?



Company of Heroes
Universe at War
Dungeons
Tap Tap Revenge
Blitzkrieg
Monopoly Tycoon
Driver San Francisco
Roblox
Requiem Memento Mori
Grim Fandango
Amped
The Incredible Hulk
Destroy All Humans!
Baldur's Gate
Rail Simulator
World of Warcraft
Far Cry
Impossible Creatures
S.T.A.L.K.E.R.
Blossom
NFL Fever 2004
Soul Ride
Diner Dash 2
Natural Selection 2
Brütal Legend
SimCity 4
Angry Birds
Play
SpellForce
Napoleon Total War
Supreme Commander
Fable II
Buzz! Quiz TV
Top Spin 2
Sonic the Hedgehog
The Witcher
Crackdown
Star Wars Empire at War
The Guild 2
MDK2
Crysis
Supreme Commander Forged Alliance
Homeworld 2
Psychonauts
The Incredibles
Links
Dungeons
Brave
Empire Total War
Escape from Monkey Island
Civilization V
plus+
Crackdown
Heroes V
Dawn of War Dark Crusade
You Are Empty
Bubble Ball

Differences between Lua and Javascript (Revisited)

- Lua has a more verbose syntax
  - easier for end-user programmers
- Lua has a well-defined FFI
  - fundamental for a scripting language to be used by different hosts
- Lua has less syntactical constructs
  - easier to pass through an FFI (“the eye of the needle”)

Differences between Lua and Javascript (Revisited)

- Objects in Lua are simpler
  - easier to adapt to different objects in the host
  - protection provided (and needed) mostly by host objects - weaker mechanisms for modularity
  - DIY object systems: more work for programmers
- Lua has finalizers and weak references
  - essential for connecting host objects with native objects
- Lua has no built-in objects and few predefined functions
  - the host selects the appropriate libraries

Conclusions

- Language design involves several trade-offs
- Different languages solve the trade-offs in different directions
- No language is good for everything
- The Web is too vast for any single language
- Javascript does not have its design focused on being a general *scripting* language
- Lua does!