The Julia programming language

Introduction to Julia and its ecosystems for OR

Mathieu Tanneau

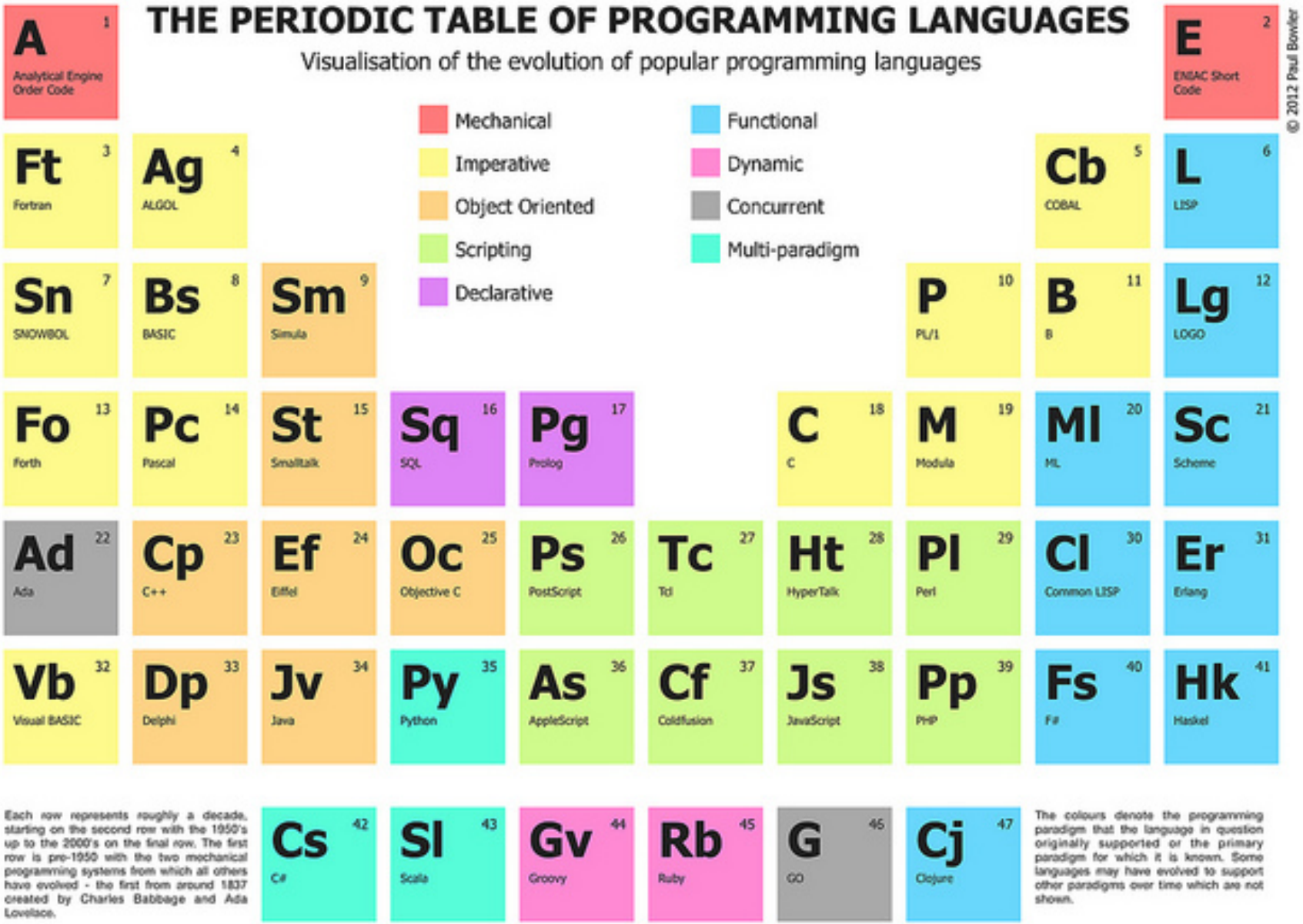
GERAD

November 20, 2018

Julia is a programming language.
You can write programs in Julia.

In this talk:

- How is Julia different from *other* languages?
- Numerical computing in Julia (focusing on linear algebra)
- How does it integrate with other languages?


THE PERIODIC TABLE OF PROGRAMMING LANGUAGES
Visualisation of the evolution of popular programming languages
Mechanical
Imperative
Object Oriented
Scripting
Declarative
Functional
Dynamic
Concurrent
Multi-paradigm
A 1 Analytical Engine Order Code
E 2 ENIAC Short Code
Ft 3 Fortran
Ag 4 ALGOL
Cb 5 COBAL
L 6 LISP
Sn 7 SNOWBOL
Bs 8 BASIC
Sm 9 Simula
P 10 PL/I
B 11 B
Lg 12 LOGO
Fo 13 Forth
Pc 14 Pascal
St 15 Smalltalk
Sq 16 SQL
Pg 17 Prolog
C 18 C
M 19 Modula
Ml 20 ML
Sc 21 Scheme
Ad 22 Ada
Cp 23 C++
Ef 24 Eiffel
Oc 25 Objective C
Ps 26 PostScript
Tc 27 Tcl
Ht 28 HyperTalk
Pl 29 Perl
Cl 30 Common LISP
Er 31 Erlang
Vb 32 Visual BASIC
Dp 33 Delphi
Jv 34 Java
Py 35 Python
As 36 AppleScript
Cf 37 Coldfusion
Js 38 JavaScript
Pp 39 PHP
Fs 40 F#
Hk 41 Haskel
Each row represents roughly a decade, starting on the second row with the 1950's up to the 2000's on the final row. The first row is pre-1950 with the two mechanical programming systems from which all others have evolved - the first from around 1837 created by Charles Babbage and Ada Lovelace.
Cs 42 C#
Sl 43 Scala
Gv 44 Groovy
Rb 45 Ruby
G 46 GO
Cj 47 Clojure
The colours denote the programming paradigm that the language in question originally supported or the primary paradigm for which it is known. Some languages may have evolved to support other paradigms over time which are not shown.
© 2012 Paul Bowler

The programmer's dilemma:

You can have a programming language that is
either fast (C) **or easy to use** (Py), but not both!

Julia's stance:

Let's have both!

"The speed potential of a language consists almost entirely of the properties that the compiler is able to prove ahead-of-time so that they don't need to be checked at runtime."

[compiler = a human-to-machine translator]

"The flexibility comes from being able to get those runtime checks automatically whenever they are needed." [1]

Back to the programmer's dilemma:
the more information you put in the code, the less readable it becomes

[1] Jameson Nash, https://juliacomputing.com/blog/2016/02/09/static-julia.html

See notebook

See notebook

See notebook

Wrap-up:

Julia is a programming language

- It tries to be fast *and* simple (it's OK to use 'for' loops)
- It is compiled
- It makes it simple to extend existing frameworks

Julia is built for numerical computing

- Native support for linear algebra
- External libraries available
- Many packages for specific applications

Julia can call and be called from other languages

Some useful links:

- Tutorials: `https://github.com/JuliaComputing/JuliaBoxTutorials`
- Official documentation: `https://docs.julialang.org/en/v1/`
- Other resources: `https://julialang.org/learning/`
- `www.google.com`

Questions?