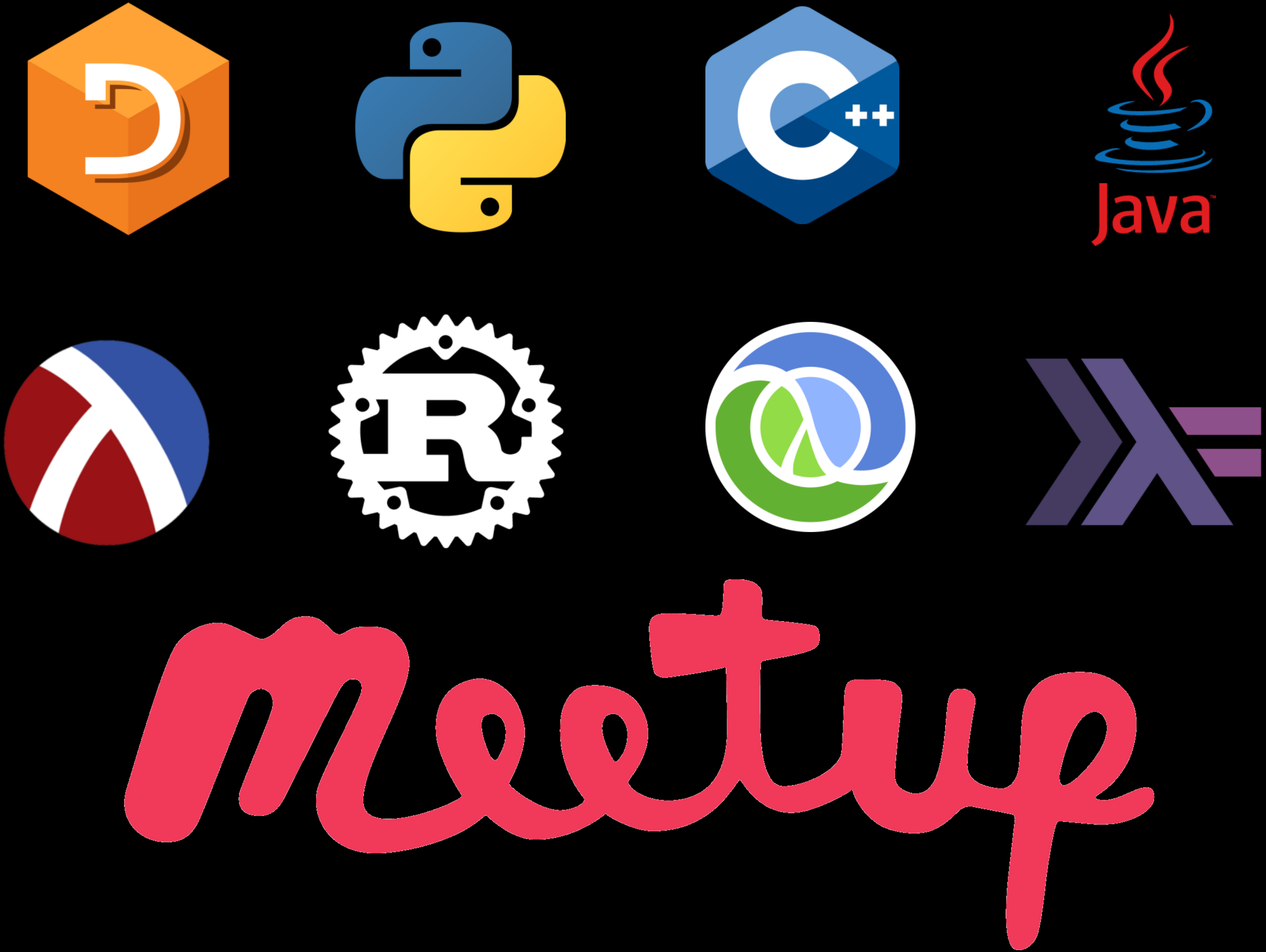
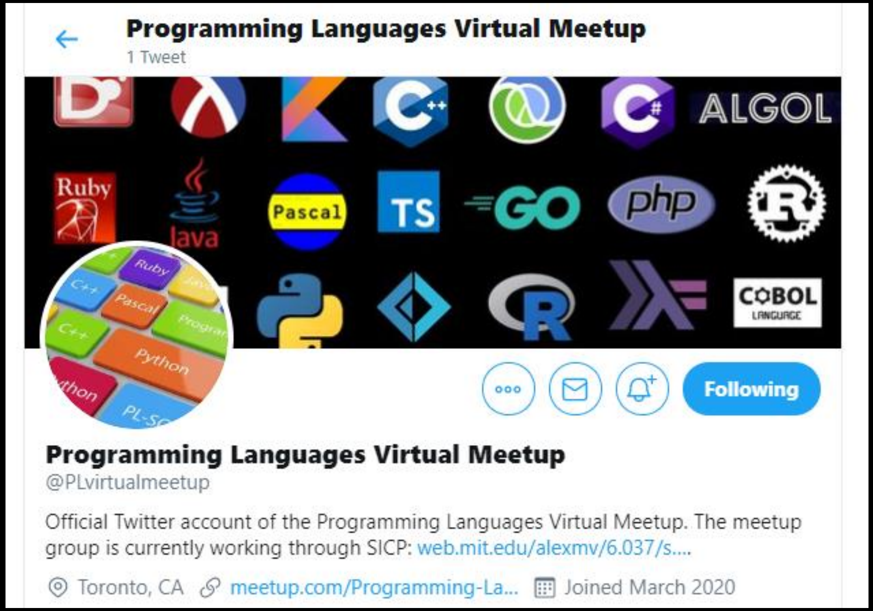

D
C++
Java
R
Meetup

Friendly Environment Policy

Berlin Code of Conduct



Programming Languages Virtual Meetup

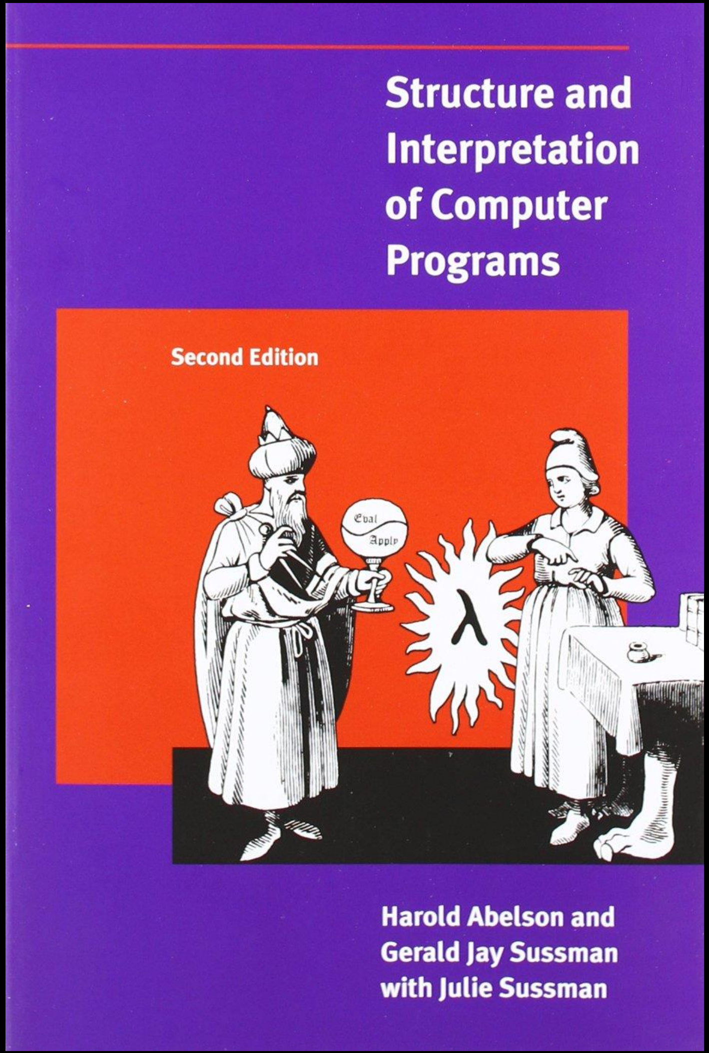
1 Tweet

Programming Languages Virtual Meetup

@PLvirtualmeetup

Official Twitter account of the Programming Languages Virtual Meetup. The meetup group is currently working through SICP: web.mit.edu/alexmv/6.037/s....

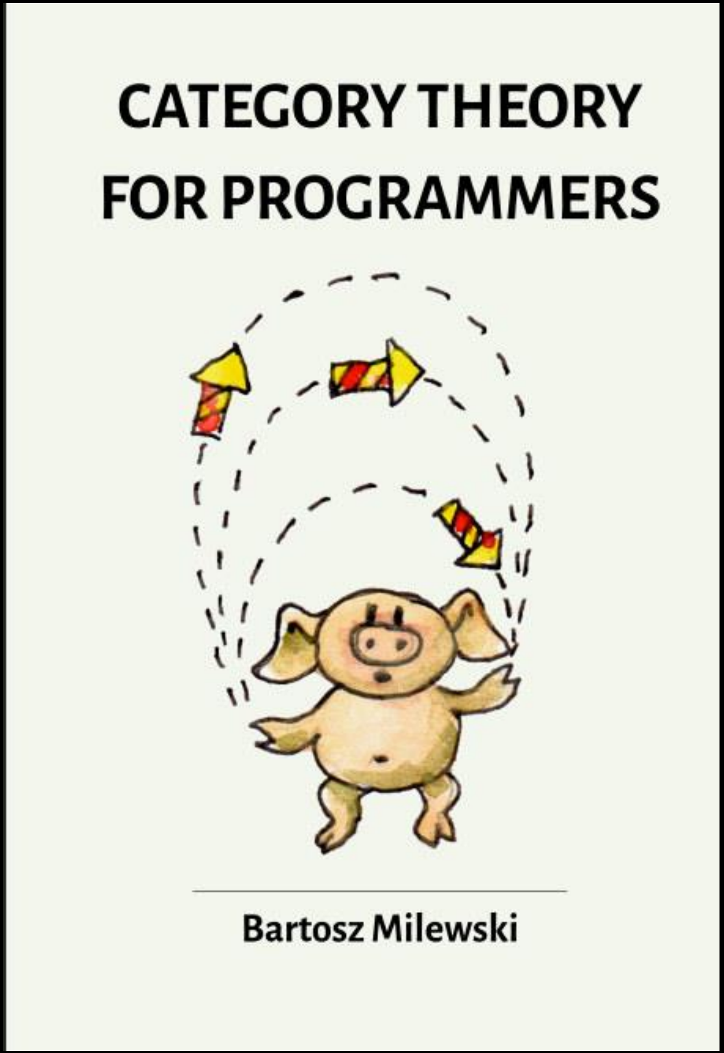
Toronto, CA meetup.com/Programming-La... Joined March 2020



Structure and Interpretation of Computer Programs

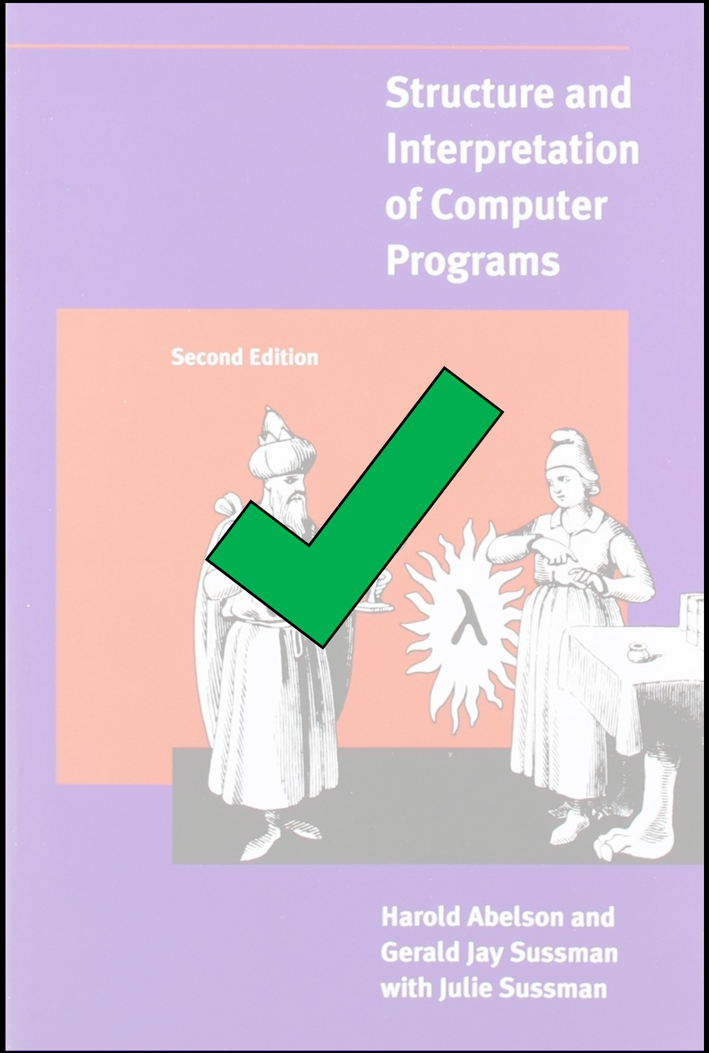
Second Edition

Harold Abelson and
Gerald Jay Sussman
with Julie Sussman



CATEGORY THEORY FOR PROGRAMMERS

Bartosz Milewski



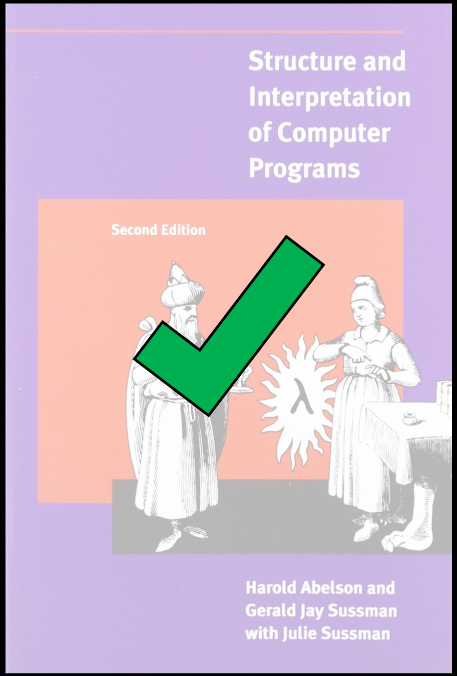
Structure and Interpretation of Computer Programs

Second Edition

Harold Abelson and Gerald Jay Sussman with Julie Sussman

CATEGORY THEORY FOR PROGRAMMERS

Bartosz Milewski



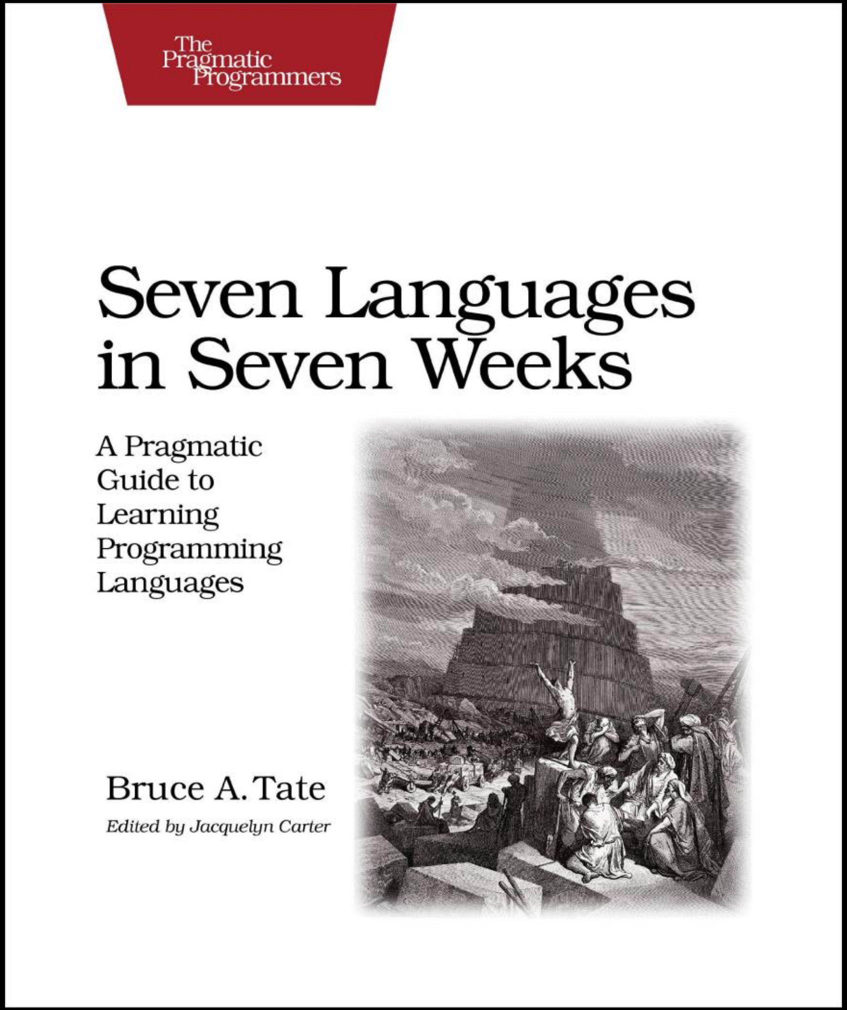
Structure and Interpretation of Computer Programs

Second Edition

Harold Abelson and Gerald Jay Sussman with Julie Sussman

CATEGORY THEORY FOR PROGRAMMERS

Bartosz Milewski



The Pragmatic Programmers

Seven Languages in Seven Weeks

A Pragmatic Guide to Learning Programming Languages

Bruce A. Tate

Edited by Jacquelyn Carter

Why 7L7W?

- It is a lighter read and we (I) need a break after CTfP
- Plus I am crazy busy for the next few months

Books to read:

Structure and Interpretation of Computer Programs

Second Edition

The Pragmatic Programmers

Seven More Languages in Seven Weeks

Languages That Are Shaping the Future

Bruce A. Tate, Fred Daoud, Ian Dees, and Jack Moffitt

Foreword by José Valim

Edited by Jacquelyn Carter

Mastering Dyal

A Complete Introduction to Dyalog APL

From Simple Machines to Impossible Programs

Understanding Computation

O'REILLY®

Tom Stuart

ACKET

ON TO LANGUAGE-ORIENTED

USING RACKET

TTERICK · VERSION 1.6

Design Patterns in Modern C++

Reusable Approaches for Object-Oriented Software Design

—

Dmitri Nesteruk

Apress®

The Pragmatic Programmers

Seven Languages in Seven Weeks

A Pragmatic Guide to Learning Programming Languages

Bruce A. Tate

Edited by Jacquelyn Carter

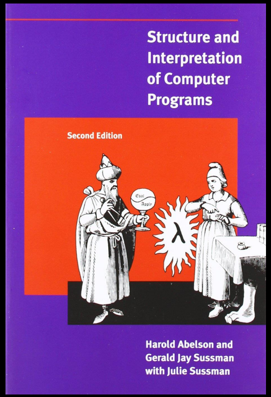
Types and Programming Languages

Functional Programming in C++

How to improve your C++ programs using functional techniques

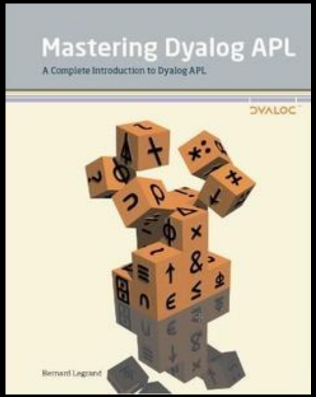
Ivan Čukić

MANNING


Structure and Interpretation of Computer Programs

Second Edition

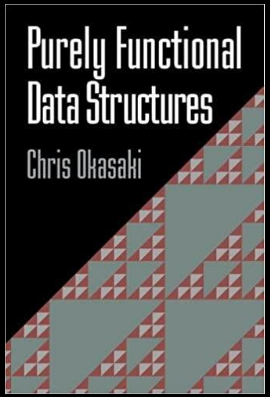
Harold Abelson and Gerald Jay Sussman with Julie Sussman


Mastering Dyalog APL

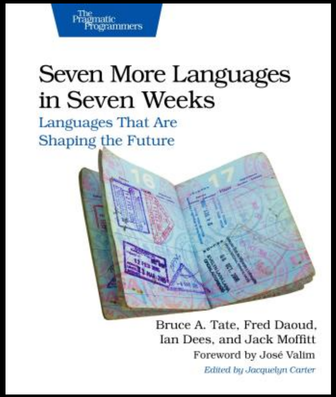
A Complete Introduction to Dyalog APL

DYALOG

Bernard Legrand


Purely Functional Data Structures

Chris Okasaki


The Pragmatic Programmers

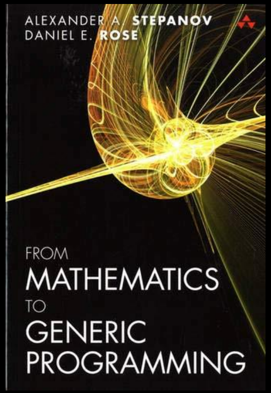
Seven More Languages in Seven Weeks

Languages That Are Shaping the Future

Bruce A. Tate, Fred Daoud, Ian Dees, and Jack Moffitt

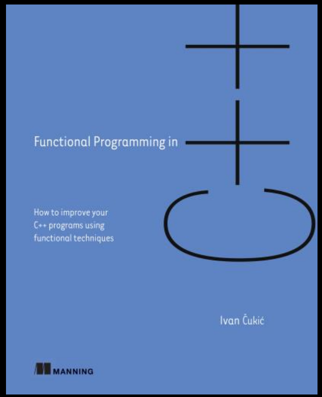
Foreword by José Valim

Edited by Jacquelyn Carter


ALEXANDER A. STEPANOV

DANIEL E. ROSE

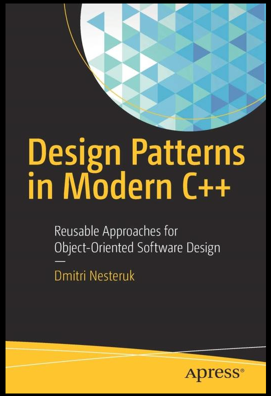
FROM MATHEMATICS TO GENERIC PROGRAMMING


Functional Programming in C++

How to improve your C++ programs using functional techniques

Ivan Čukić

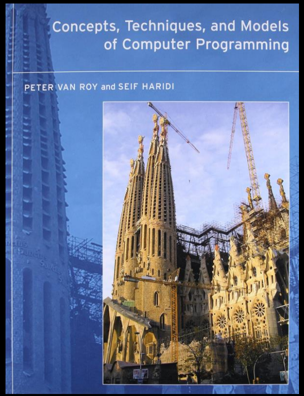
MANNING


Design Patterns in Modern C++

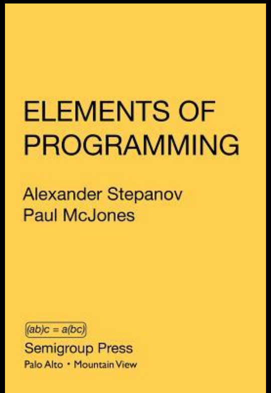
Reusable Approaches for Object-Oriented Software Design

Dmitri Nesteruk

Apress®


Concepts, Techniques, and Models of Computer Programming

PETER VAN ROY and SEIF HARIDI


ELEMENTS OF PROGRAMMING

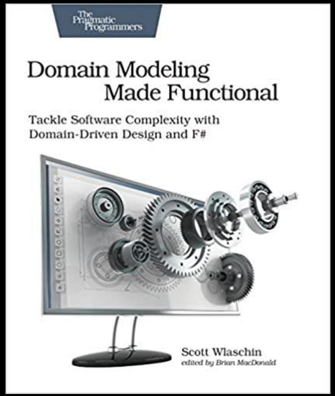
Alexander Stepanov

Paul McJones

(ab)c = a(bc)

Semigroup Press

Palo Alto • Mountain View

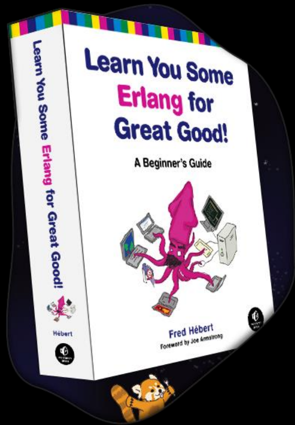

The Pragmatic Programmers

Domain Modeling Made Functional

Tackle Software Complexity with Domain-Driven Design and F#

Scott Wlaschin

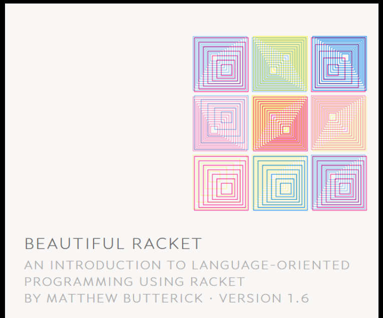
edited by Brian MacDonald


Learn You Some Erlang for Great Good!

A Beginner's Guide

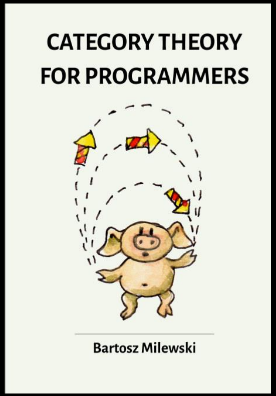
Fred Hébert

Foreword by Joe Armstrong

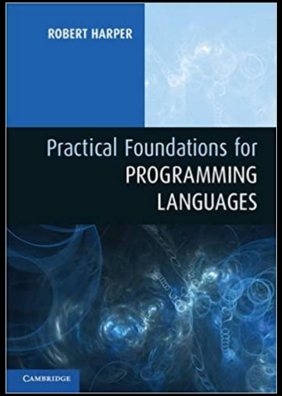

BEAUTIFUL RACKET

AN INTRODUCTION TO LANGUAGE-ORIENTED PROGRAMMING USING RACKET

BY MATTHEW BUTTERICK · VERSION 1.6

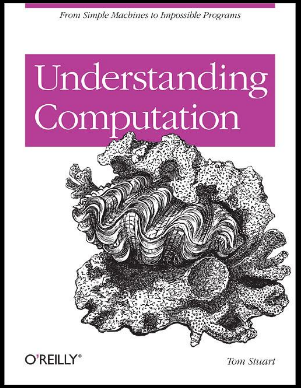

CATEGORY THEORY FOR PROGRAMMERS

Bartosz Milewski


ROBERT HARPER

Practical Foundations for PROGRAMMING LANGUAGES

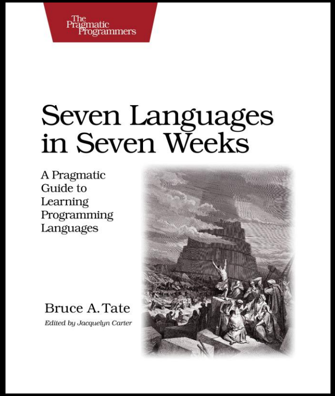
CAMBRIDGE


From Simple Machines to Impossible Programs

Understanding Computation

O'REILLY®

Tom Stuart

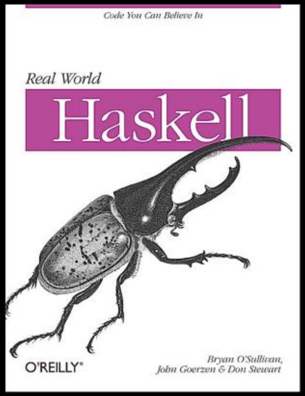

The Pragmatic Programmers

Seven Languages in Seven Weeks

A Pragmatic Guide to Learning Programming Languages

Bruce A. Tate

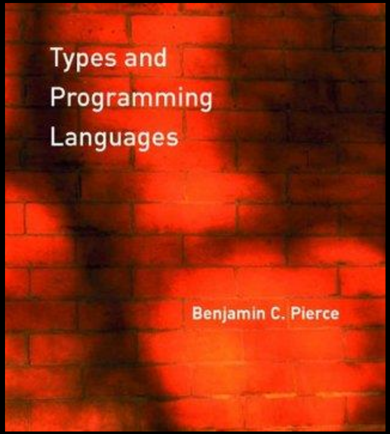
Edited by Jacquelyn Carter


Code You Can Believe In

Real World Haskell

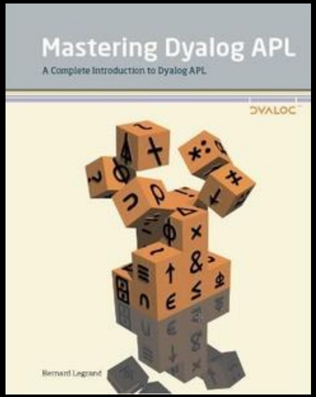
O'REILLY®

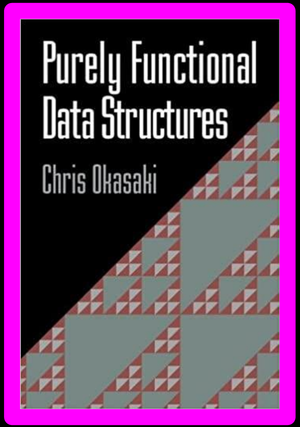
Bryan O'Sullivan, John Goerzen & Don Stewart

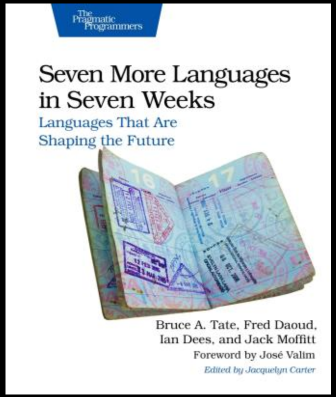

Types and Programming Languages

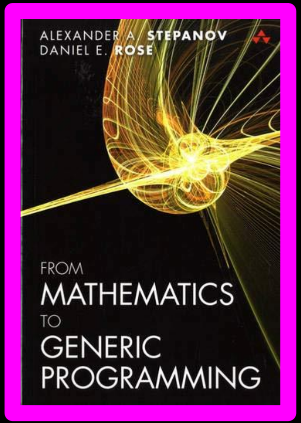
Benjamin C. Pierce

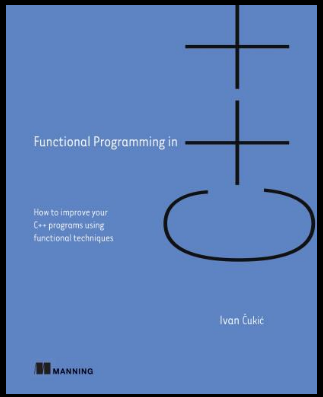
Structure and Interpretation of Computer Programs
Second Edition
Harold Abelson and Gerald Jay Sussman with Julie Sussman

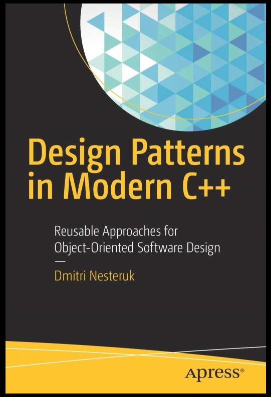

Mastering Dyalog APL
A Complete Introduction to Dyalog APL
DYALOG
Bernard Legrand

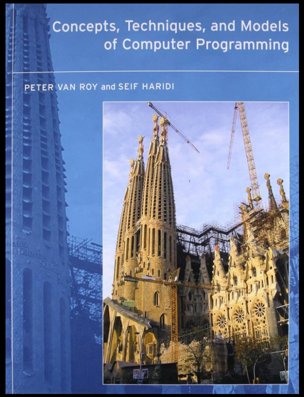

Purely Functional Data Structures
Chris Okasaki

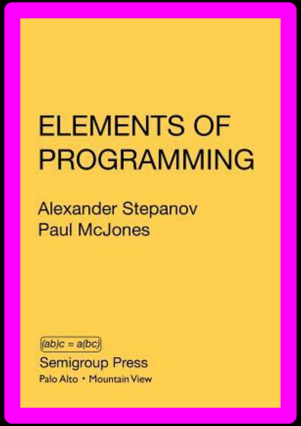

The Pragmatic Programmers
Seven More Languages in Seven Weeks
Languages That Are Shaping the Future
Bruce A. Tate, Fred Daoud, Ian Dees, and Jack Moffitt
Foreword by José Valim
Edited by Jacquelyn Carter

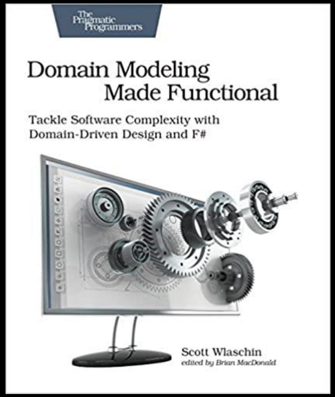

ALEXANDER A. STEPANOV
DANIEL E. ROSE
FROM MATHEMATICS TO GENERIC PROGRAMMING

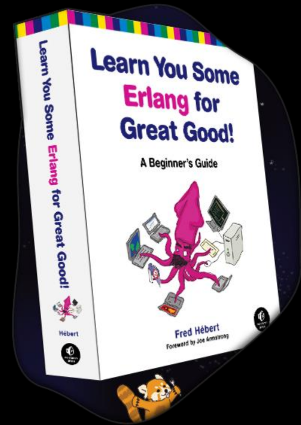

Functional Programming in C++
How to improve your C++ programs using functional techniques
Ivan Čukić
MANNING

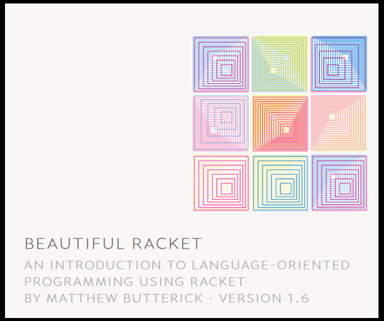

Design Patterns in Modern C++
Reusable Approaches for Object-Oriented Software Design
Dmitri Nesteruk
Apress®


Concepts, Techniques, and Models of Computer Programming
PETER VAN ROY and SEIF HARIDI

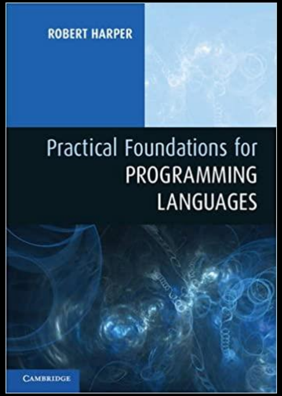

ELEMENTS OF PROGRAMMING
Alexander Stepanov
Paul McJones
(ab)c = a(bc)
Semigroup Press
Palo Alto • Mountain View

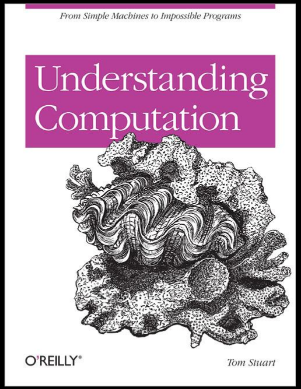

The Pragmatic Programmers
Domain Modeling Made Functional
Tackle Software Complexity with Domain-Driven Design and F#
Scott Wlaschin
edited by Brian MacDonald

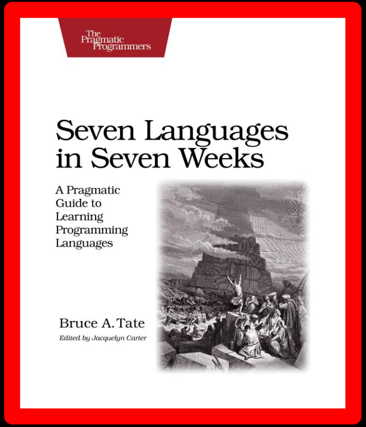

Learn You Some Erlang for Great Good!
A Beginner's Guide
Fred Hébert
Foreword by Joe Armstrong

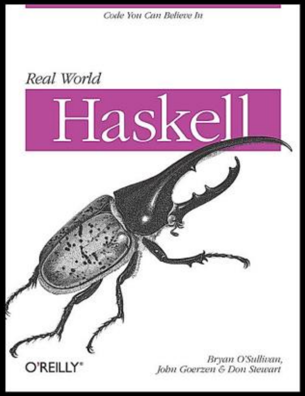

BEAUTIFUL RACKET
AN INTRODUCTION TO LANGUAGE-ORIENTED PROGRAMMING USING RACKET
BY MATTHEW BUTTERICK · VERSION 1.6

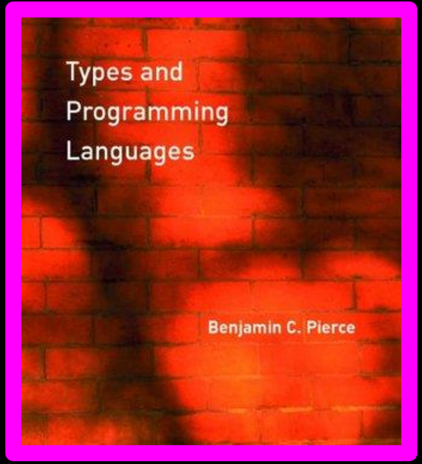
CATEGORY THEORY FOR PROGRAMMERS
Bartosz Milewski


ROBERT HARPER
Practical Foundations for PROGRAMMING LANGUAGES
CAMBRIDGE


From Simple Machines to Impossible Programs
Understanding Computation
O'REILLY®
Tom Stuart


The Pragmatic Programmers
Seven Languages in Seven Weeks
A Pragmatic Guide to Learning Programming Languages
Bruce A. Tate
Edited by Jacquelyn Carter


Code You Can Believe In
Real World Haskell
O'REILLY®
Bryan O'Sullivan, John Goerzen & Don Stewart


Types and Programming Languages
Benjamin C. Pierce

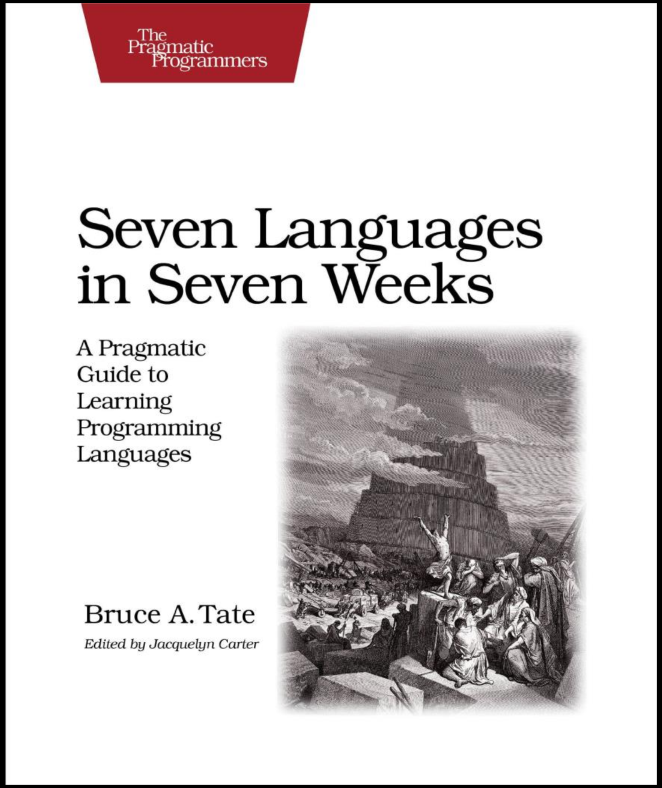
Goal of PLVM

- Work through books on programming languages together
- Grow knowledge on PLs and PL:
  - principles
  - design
  - implementation
- This ultimately will lead to ability to write **code** that is more:
  - readable & expressive
  - maintainable & scalable
  - beautiful & idiomatic

Format of PLVM

- 2 hour meeting once ~~a~~ every ~three weeks
- ~30 min presentation
- Discussion during and afterwards

- I will pre-record presentation and upload to YouTube for those unable to attend



The Pragmatic Programmers

Seven Languages in Seven Weeks

A Pragmatic Guide to Learning Programming Languages

Bruce A. Tate

Edited by Jacquelyn Carter

7 Languages in 7 Weeks

Introduction

"To better understand the essence of **OO programming**, you should study **logic** or **functional programming**."

Joe Armstrong

Foreward, 7L7W

✓ Foreword

➡ 1. Introduction

- 1.1 Method to the Madness
- 1.2 The Languages
- 1.3 Buy This Book
- 1.4 Don’t Buy This Book
- 1.5 A Final Charge

“You may even seek **enlightenment**, knowing every new language can shape the way you think.”

Bruce Tate

Introduction, 7L7W

✓ Foreword

✓ 1. Introduction

→ 1.1 Method to the Madness

1.2 The Languages

1.3 Buy This Book

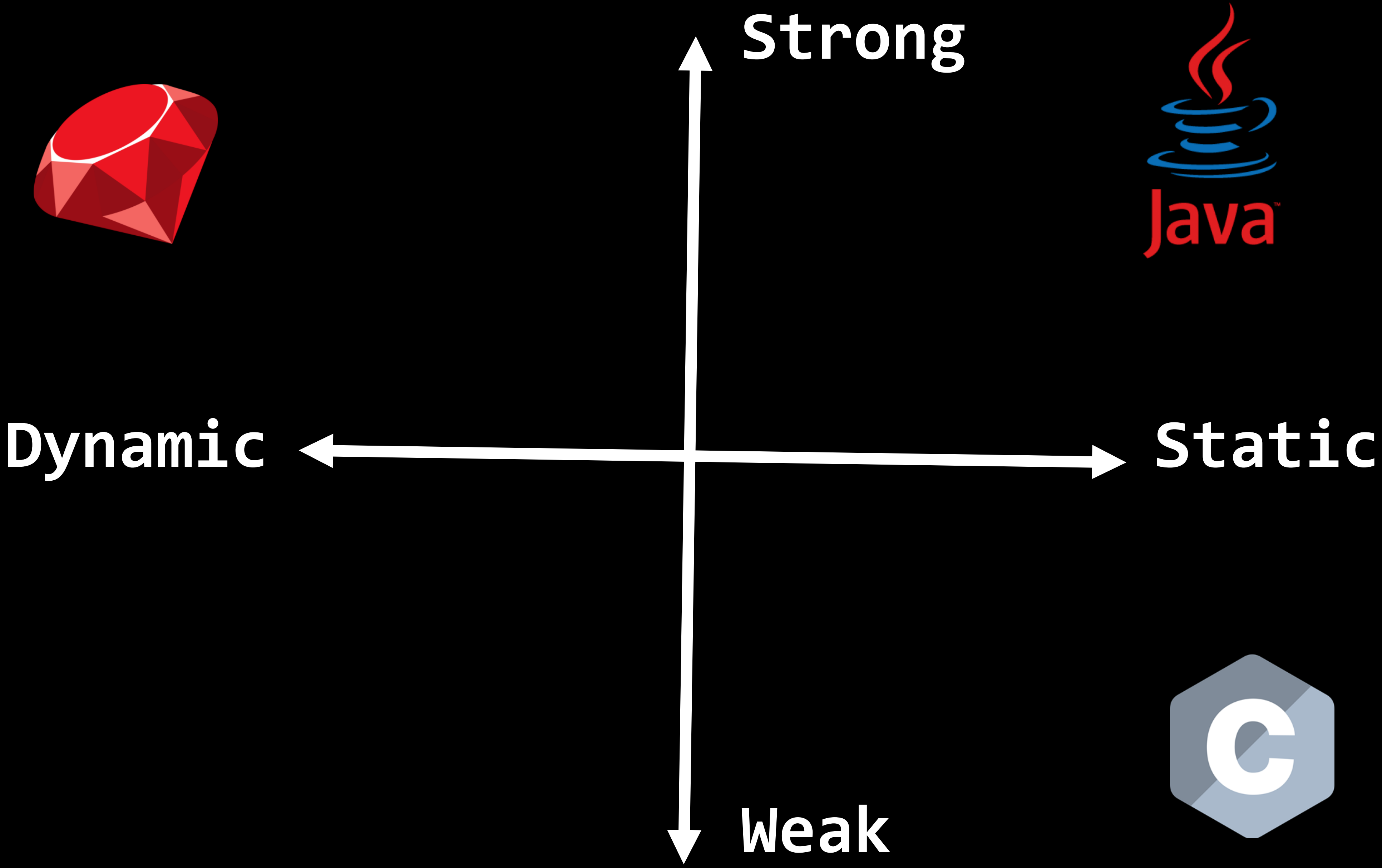
1.4 Don’t Buy This Book

1.5 A Final Charge

1. What is the typing model?
2. What is the programming model?
3. How will you interact with it?
4. What are the decision constructs and core data structures?
5. What are the core features that make the language unique?

What is the typing model?

- *What is the typing model?* Typing is strong (Java) or weak (C), static (Java) or dynamic (Ruby).

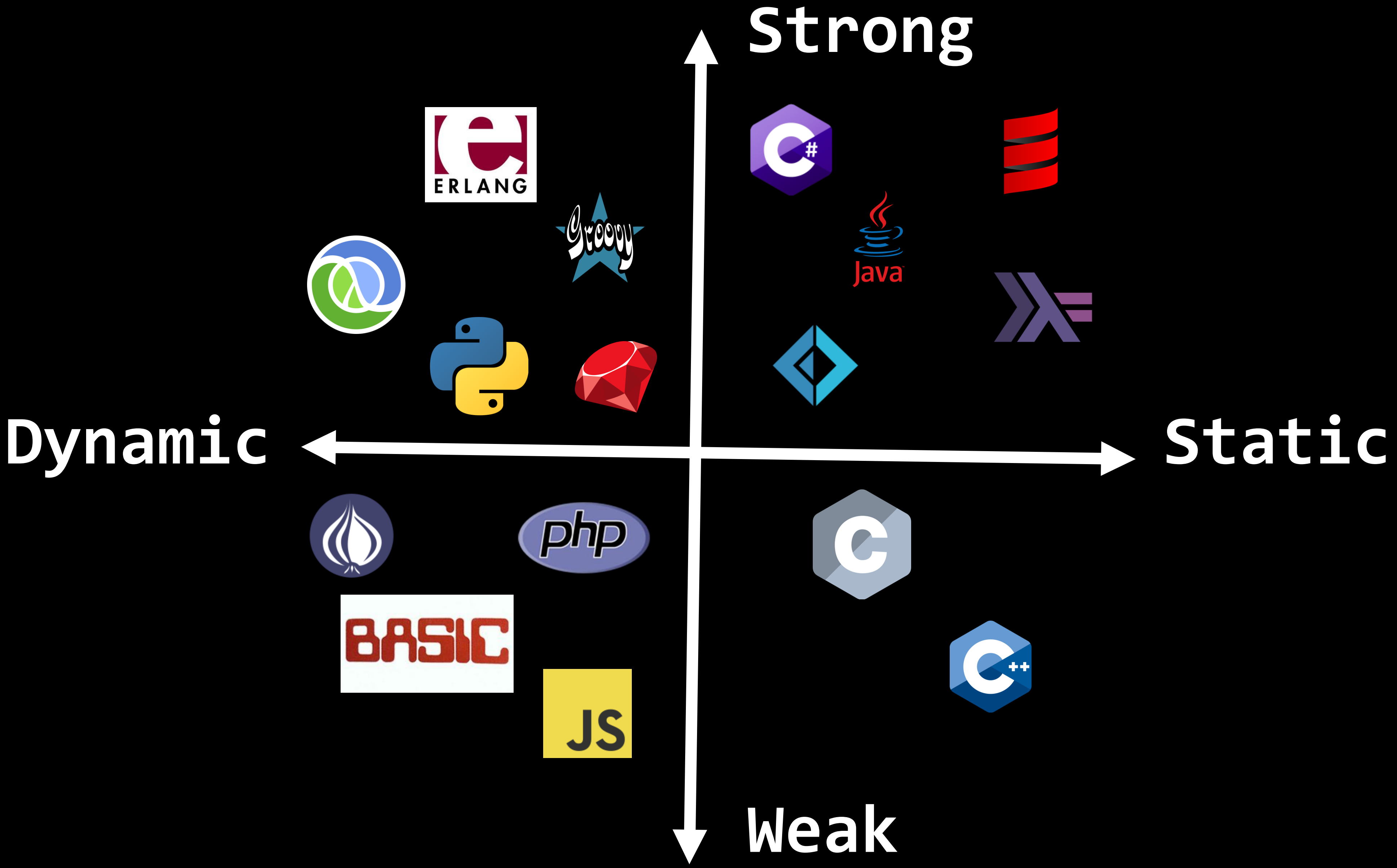


Strong

Dynamic

Static

Weak



Strong

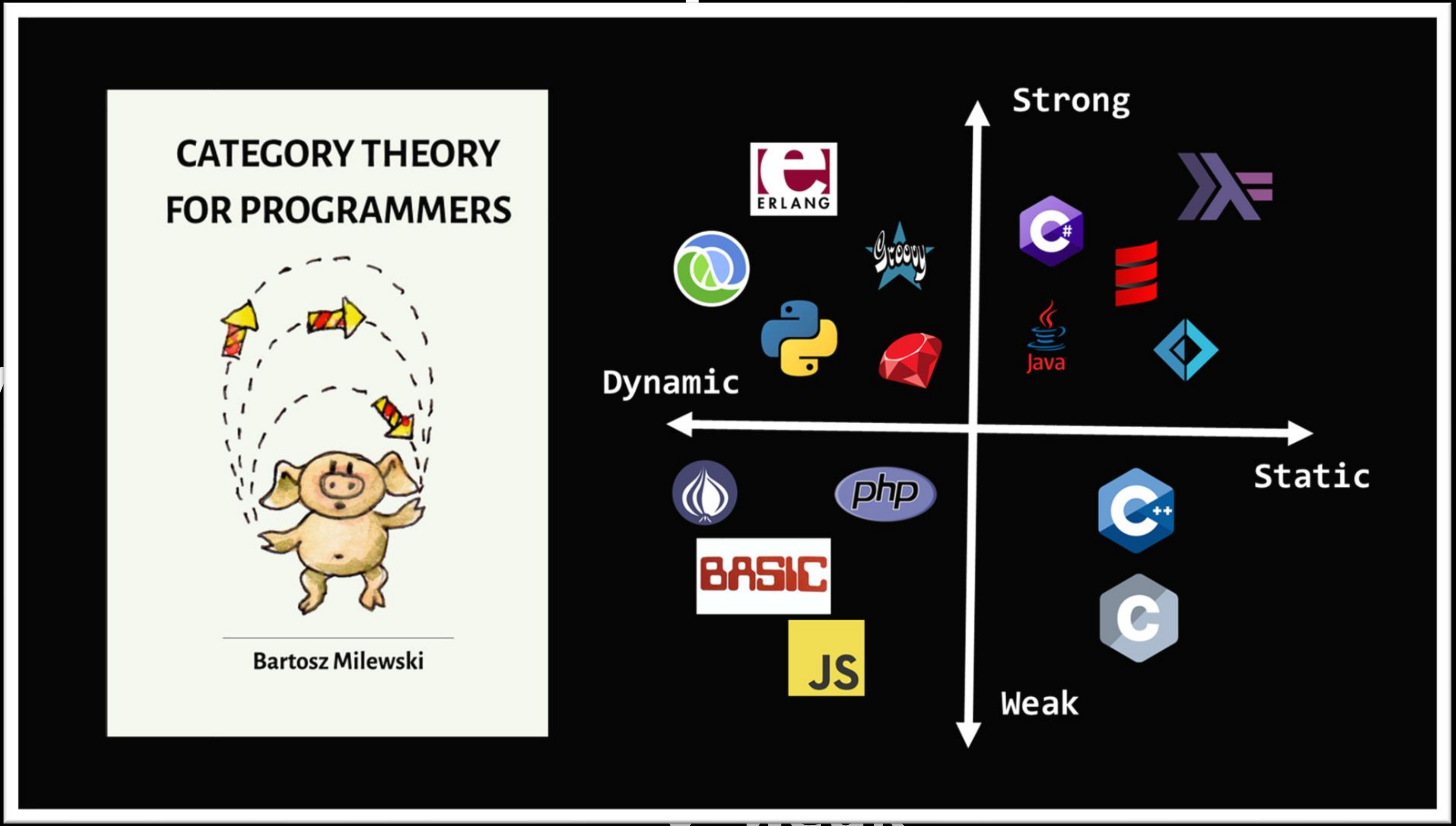
Dynamic

Static

Weak

Strong

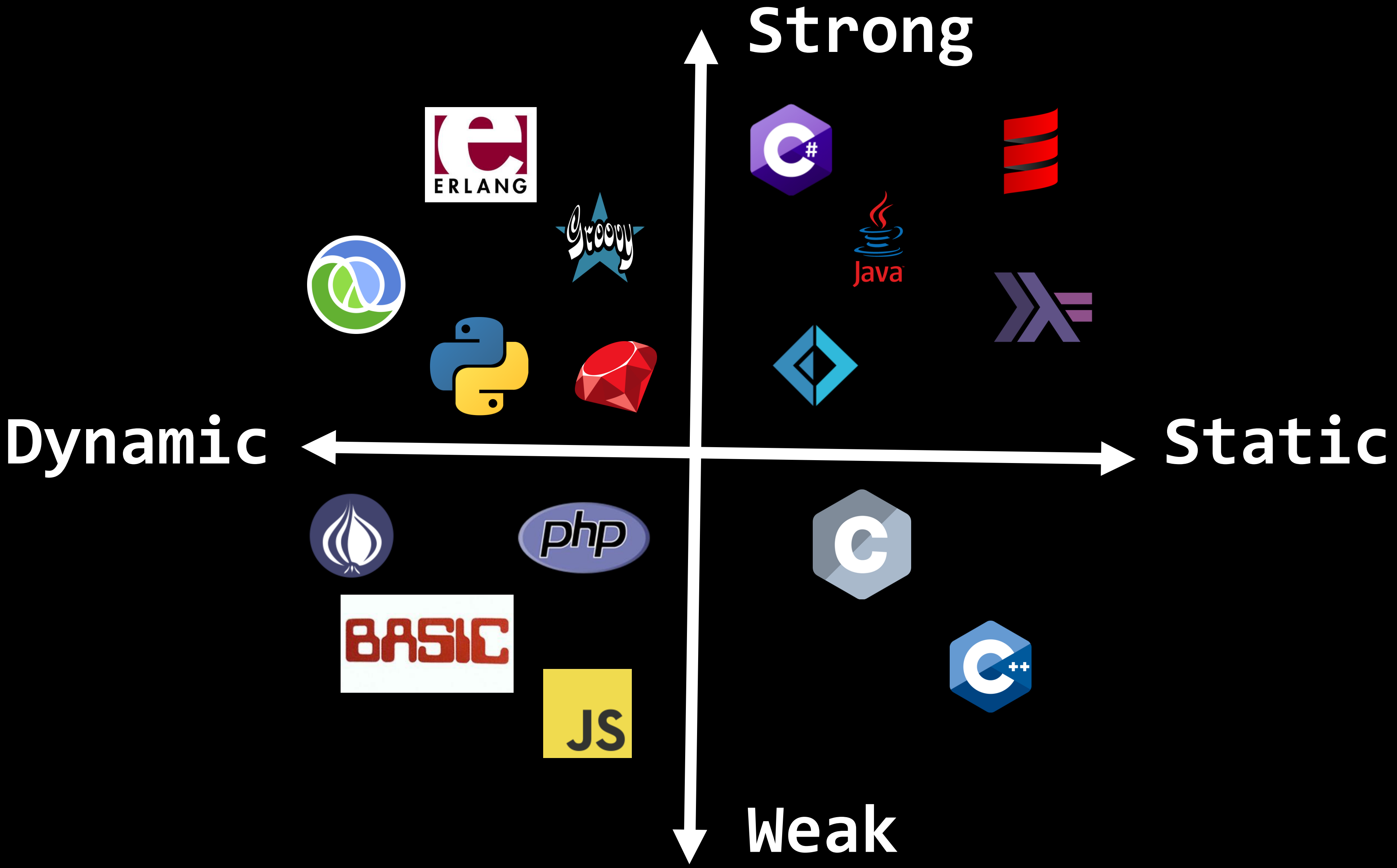
Dy



CATEGORY THEORY FOR PROGRAMMERS

Bartosz Milewski

Weak

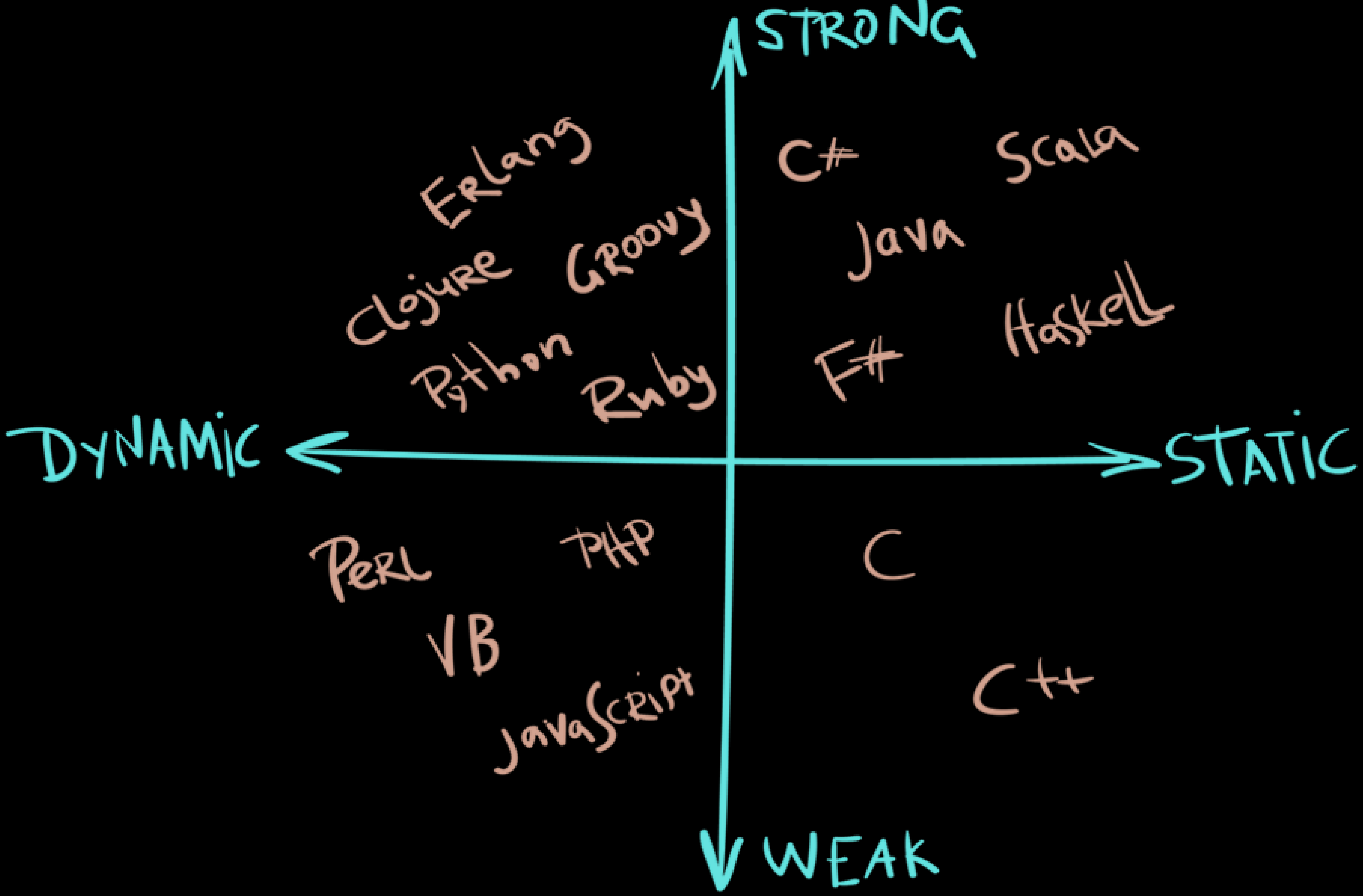


Strong

Weak

Dynamic

Static

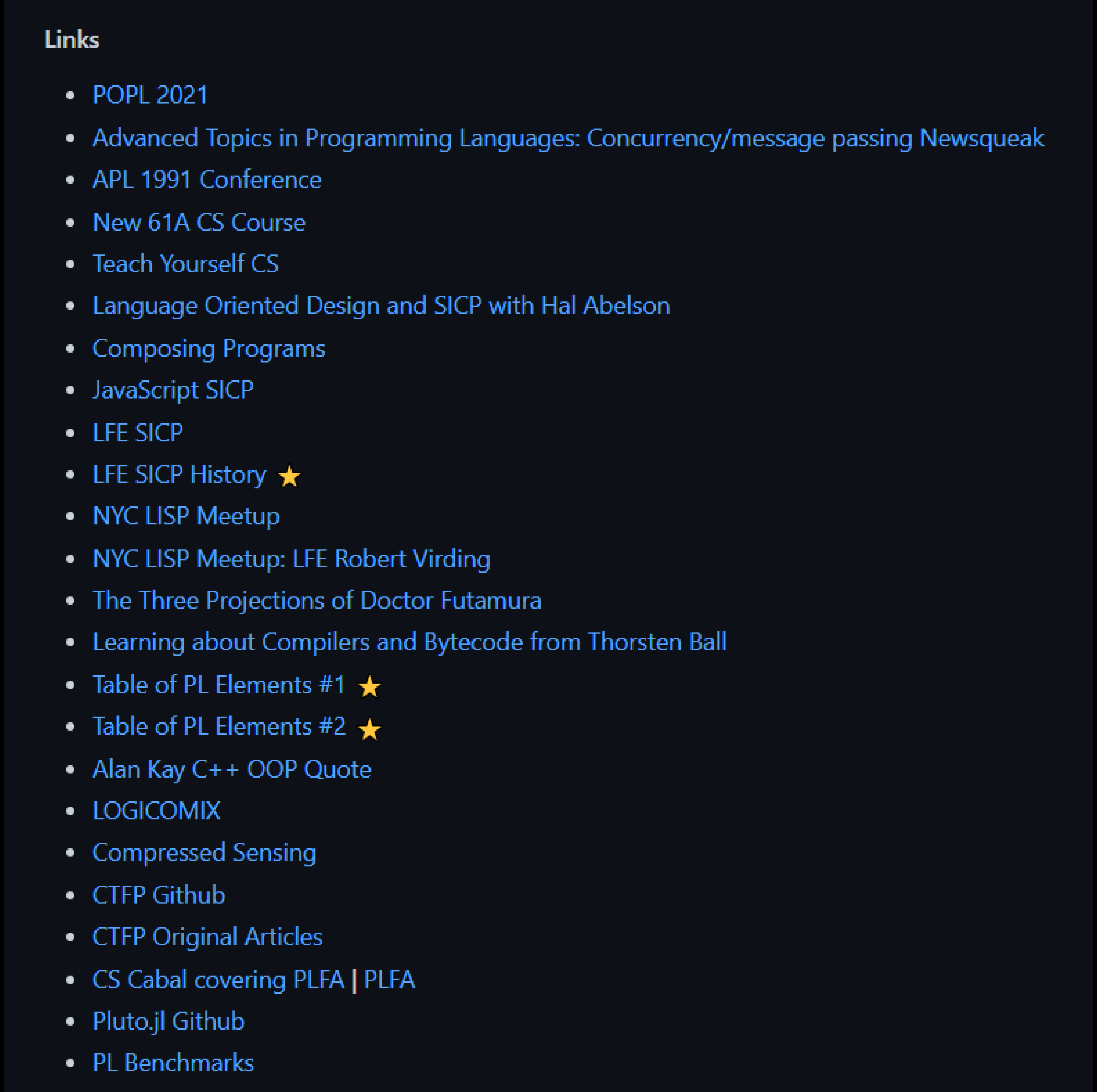


https://android.jlelse.eu/magic-lies-here-statically-typed-vs-dynamically-typed-languages-d151c7f95e2b

What is the programming model?

What is the programming paradigm?

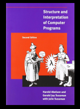
- *What is the programming model?* Is it object-oriented (OO), functional, procedural, or some type of hybrid? This book has languages spanning four different programming models and, sometimes, combinations of more than one. You will find a logic-based programming language (Prolog), two languages with full support for object-oriented concepts (Ruby, Scala), four languages that are functional in nature (Scala, Erlang, Clojure, Haskell), and one prototype language (Io). Several of

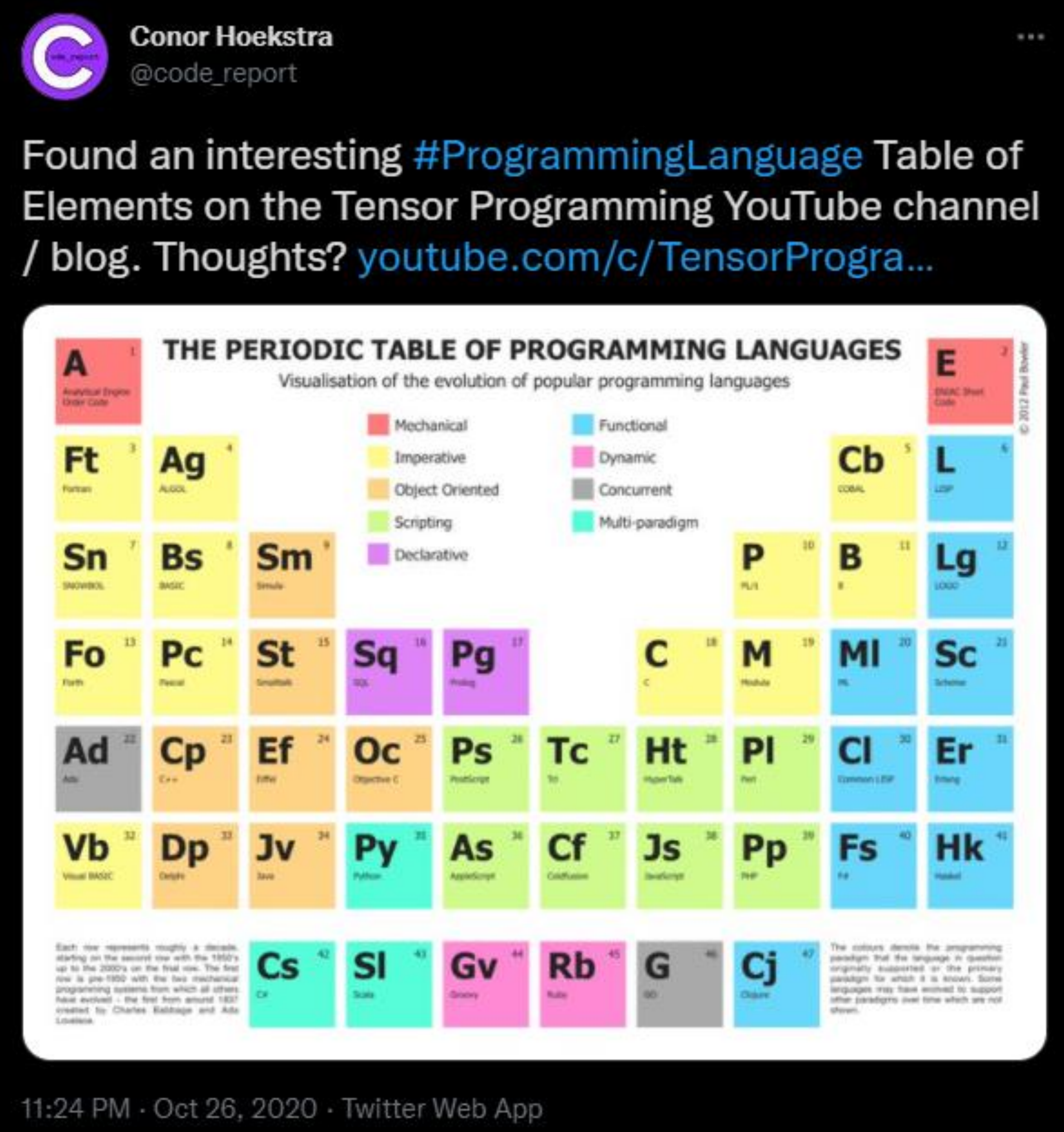


Links

- POPL 2021
- Advanced Topics in Programming Languages: Concurrency/message passing Newsqueak
- APL 1991 Conference
- New 61A CS Course
- Teach Yourself CS
- Language Oriented Design and SICP with Hal Abelson
- Composing Programs
- JavaScript SICP
- LFE SICP
- LFE SICP History ★
- NYC LISP Meetup
- NYC LISP Meetup: LFE Robert Virding
- The Three Projections of Doctor Futamura
- Learning about Compilers and Bytecode from Thorsten Ball
- Table of PL Elements #1 ★
- Table of PL Elements #2 ★
- Alan Kay C++ OOP Quote
- LOGICOMIX
- Compressed Sensing
- CTFP Github
- CTFP Original Articles
- CS Cabal covering PLFA | PLFA
- Pluto.jl Github
- PL Benchmarks

Ch 5.5 Compilation





Conor Hoekstra
@code_report

Found an interesting #ProgrammingLanguage Table of Elements on the Tensor Programming YouTube channel / blog. Thoughts? youtube.com/c/TensorProgra...

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 Haskell
		Cs 42 C#	Sl 43 Scala	Gv 44 Groovy	Rb 45 Ruby	G 46 GO	Cj 47 Clojure		

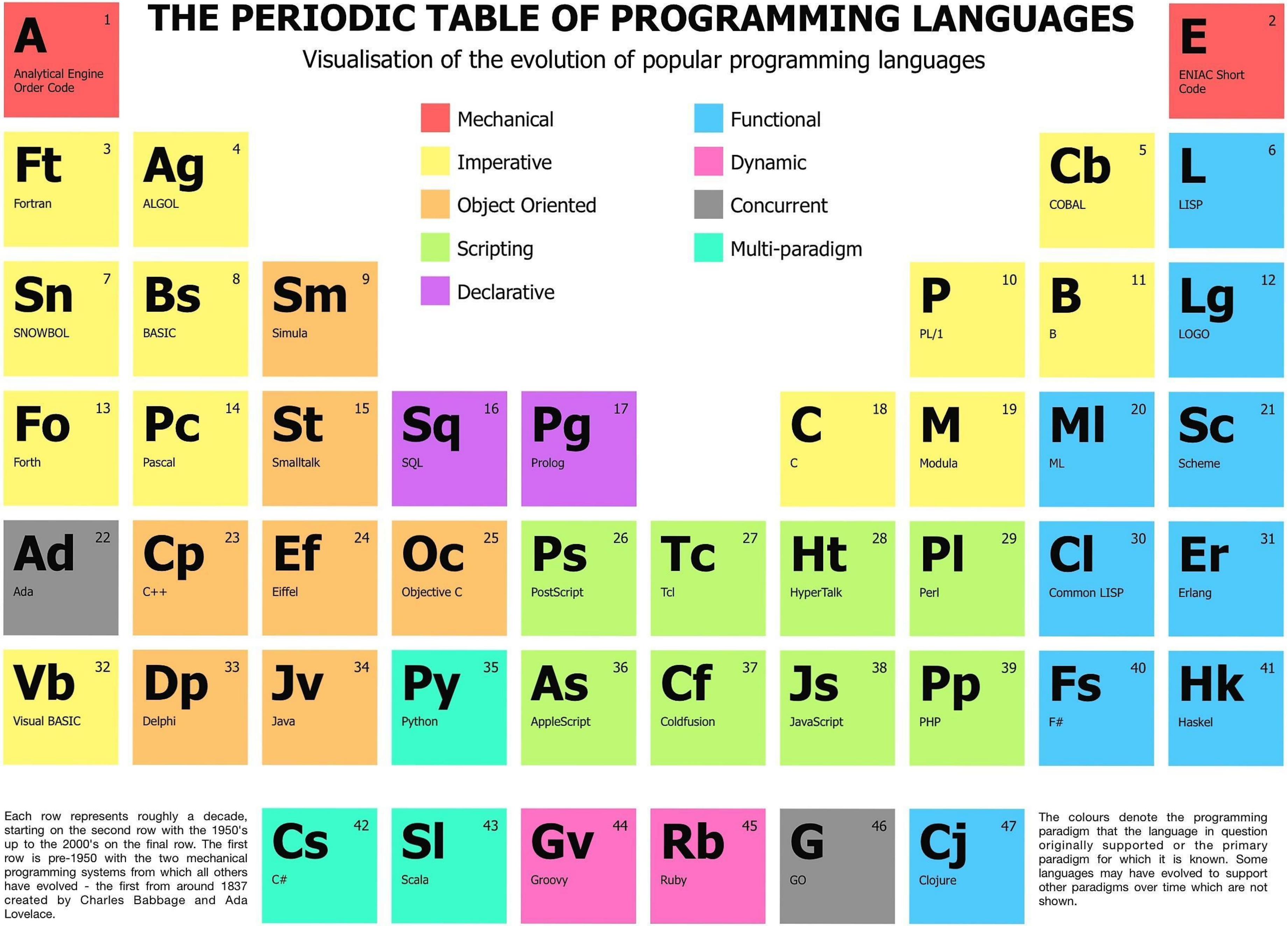
Each row represents roughly a decade, starting on the second row with the 1950's up to the 2000's on the final row. The first row is pre-1950 with the two mechanical programming systems from which all others have evolved - the first from around 1837 created by Charles Babbage and Ada Lovelace.

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.

11:24 PM · Oct 26, 2020 · Twitter Web App

THE PERIODIC TABLE OF PROGRAMMING LANGUAGES

Visualisation of the evolution of popular programming languages

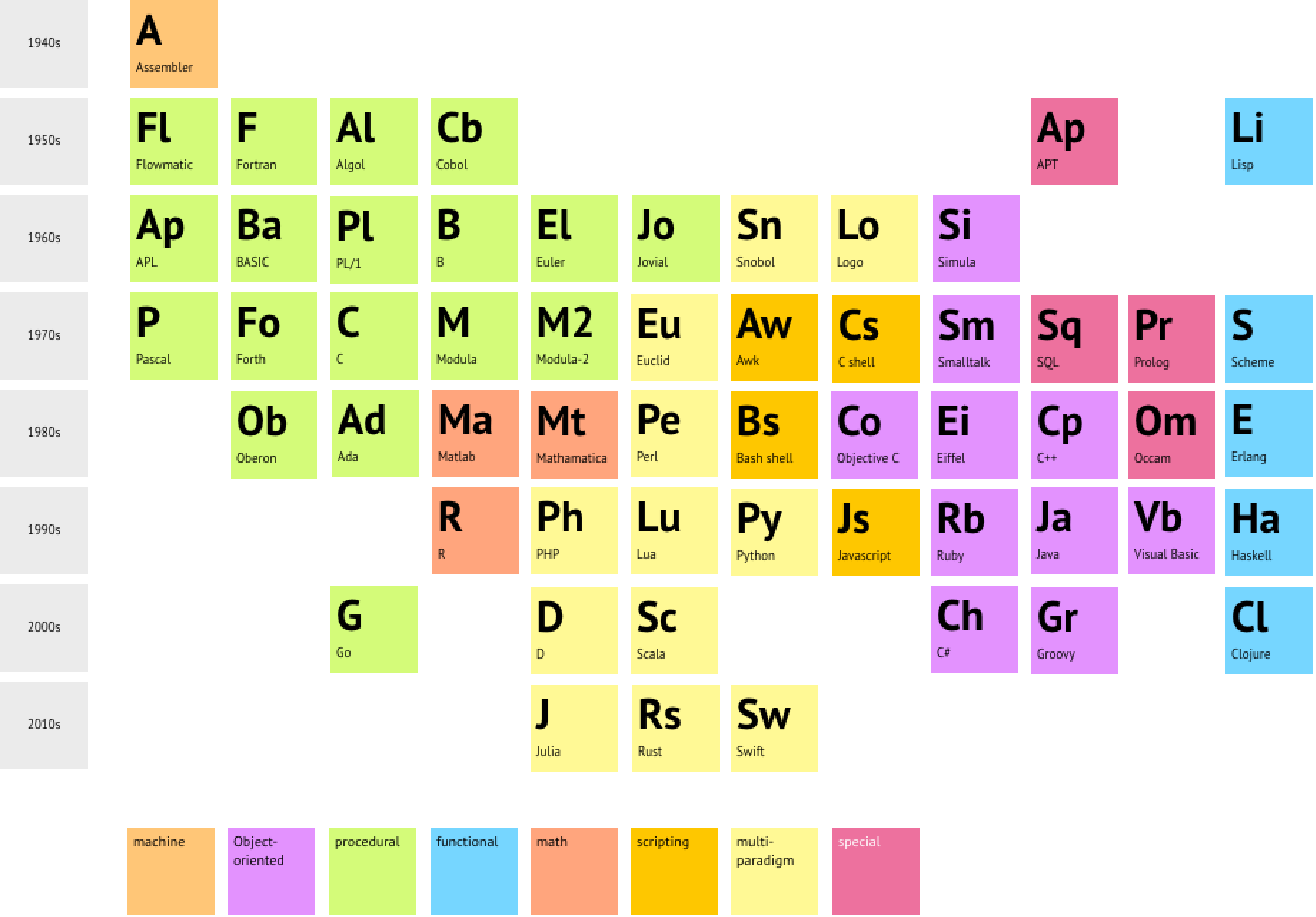


- Mechanical
- Imperative
- Object Oriented
- Scripting
- Declarative
- Functional
- Dynamic
- Concurrent
- Multi-paradigm

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/1	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
		Cs 42 C#	Sl 43 Scala	Gv 44 Groovy	Rb 45 Ruby	G 46 GO	Cj 47 Clojure		

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.

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.



1940s	A Assembler											
1950s	Fl Flowmatic	F Fortran	Al Algol	Cb Cobol						Ap APT		Li Lisp
1960s	Ap APL	Ba BASIC	Pl PL/1	B B	El Euler	Jo Jovial	Sn Snobol	Lo Logo	Si Simula			
1970s	P Pascal	Fo Forth	C C	M Modula	M2 Modula-2	Eu Euclid	Aw Awk	Cs C shell	Sm Smalltalk	Sq SQL	Pr Prolog	S Scheme
1980s		Ob Oberon	Ad Ada	Ma Matlab	Mt Mathamatica	Pe Perl	Bs Bash shell	Co Objective C	Ei Eiffel	Cp C++	Om Occam	E Erlang
1990s				R R	Ph PHP	Lu Lua	Py Python	Js Javascript	Rb Ruby	Ja Java	Vb Visual Basic	Ha Haskell
2000s			G Go		D D	Sc Scala			Ch C#	Gr Groovy		Cl Clojure
2010s					J Julia	Rs Rust	Sw Swift					

machine

Object-oriented

procedural

functional

math

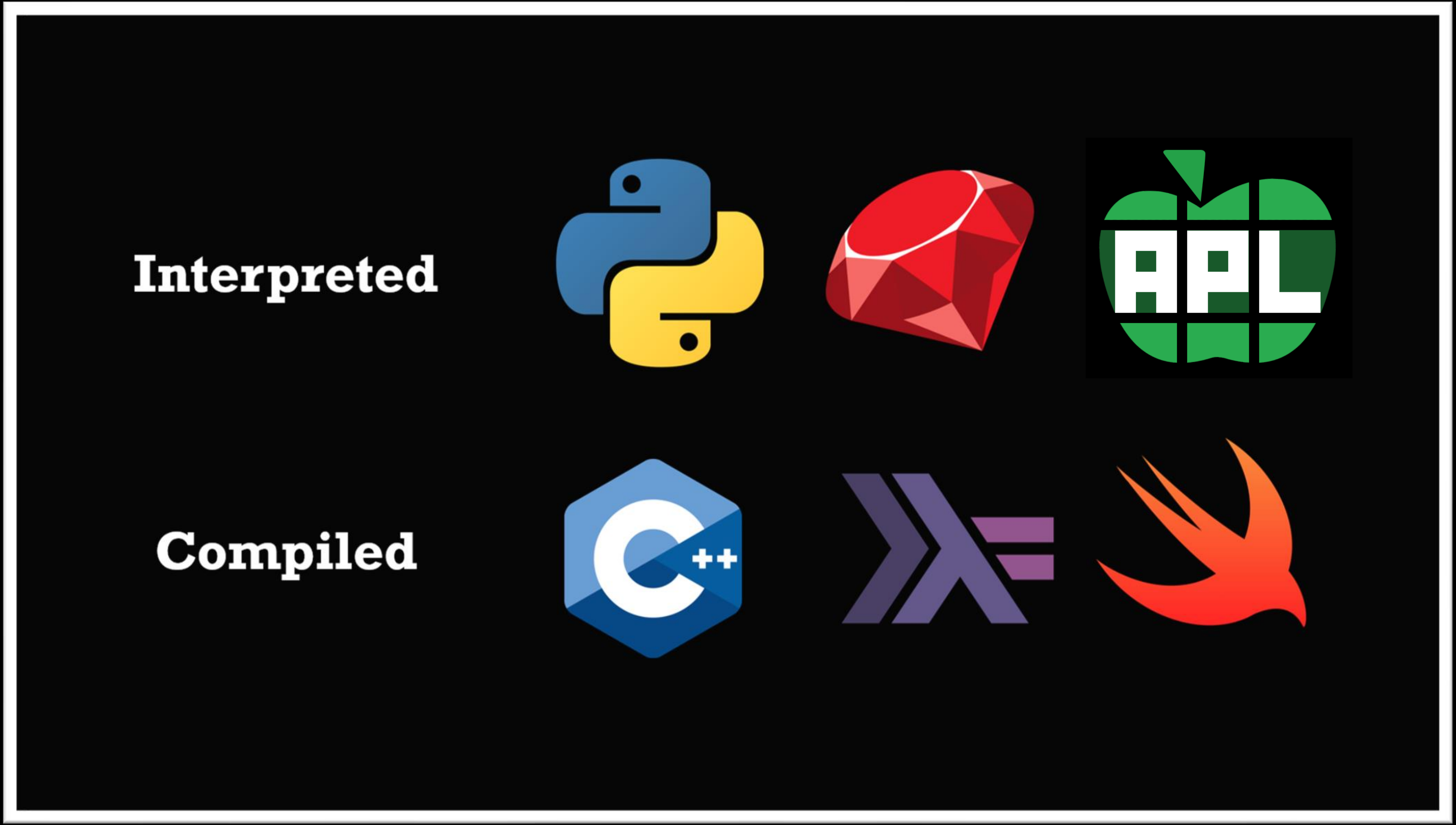
scripting

multi-paradigm

special

How will you interact with it?

- *How will you interact with it?* Languages are compiled or interpreted, and some have virtual machines while others don’t.



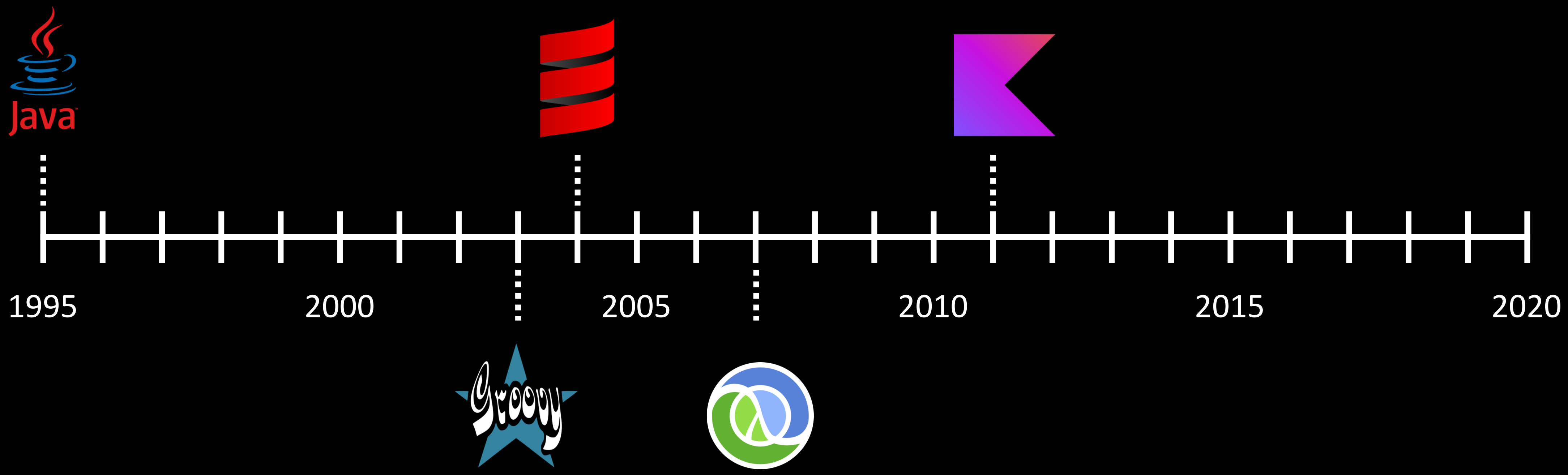
Interpreted

Compiled

Java™

Groovy

JVM Languages



What are the decision constructs and core data structures?

In languages such as Smalltalk and Lisp, the collections are defining characteristics of the language. In others, like C++ and Java, collections are all over the place, defining the user's experience by their absence and lack of cohesion.

✓ Foreword

✓ 1. Introduction

✓ 1.1 Method to the Madness

→ 1.2 The Languages

1.3 Buy This Book

1.4 Don’t Buy This Book

1.5 A Final Charge

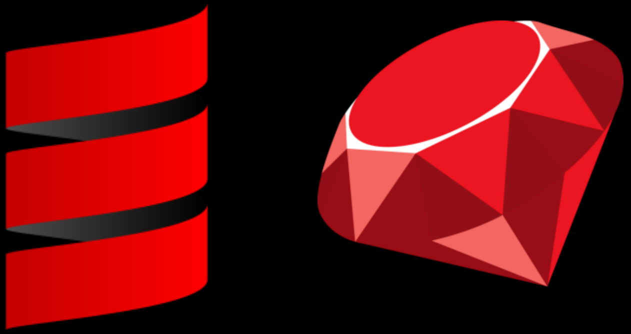
Io

ERLANG

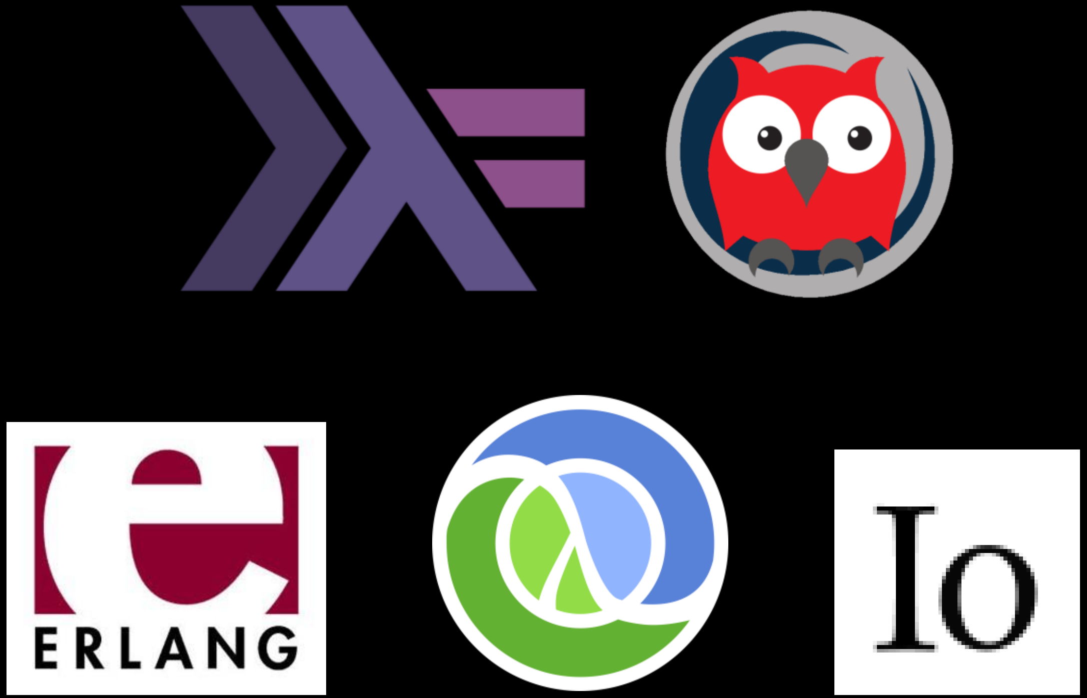
Functional

Non-Functional

Object Oriented



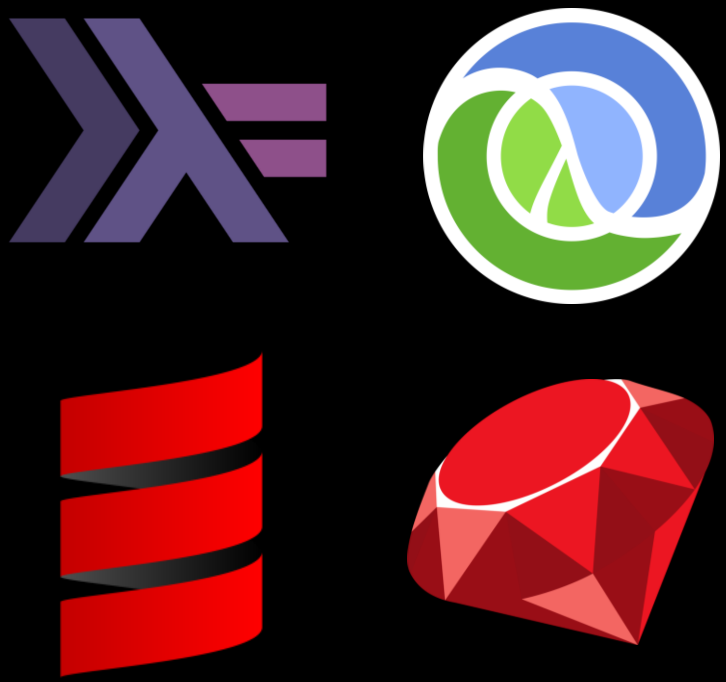
Non-OO



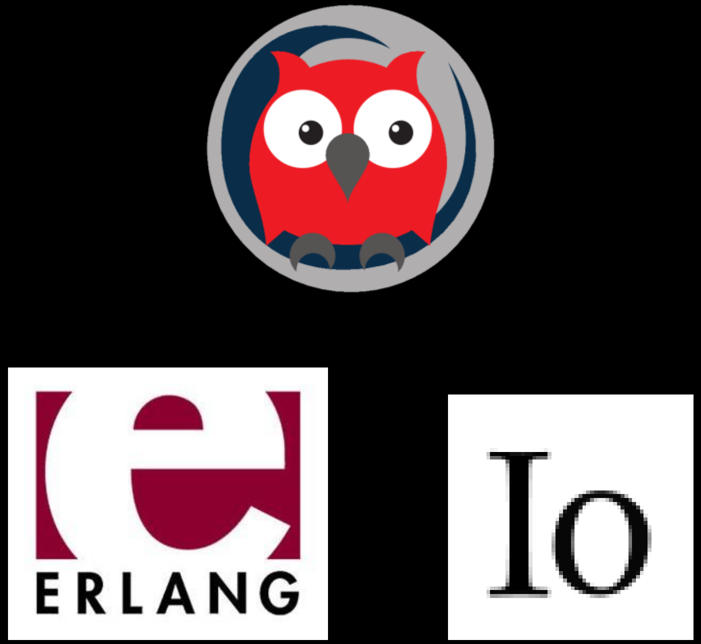
JVM Languages

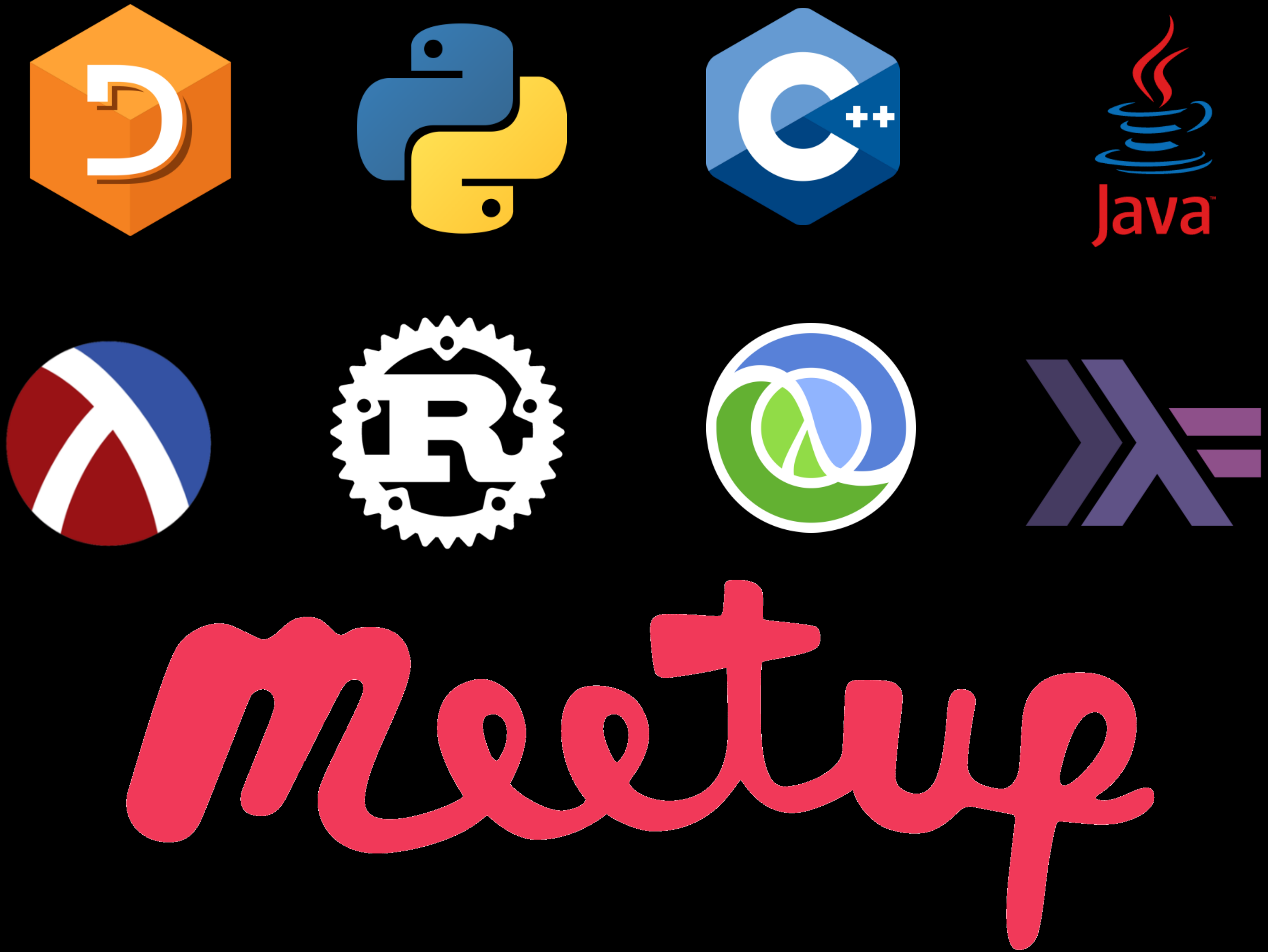
Non-JVM

I’ve Coded In



Have not




Java
Meetup