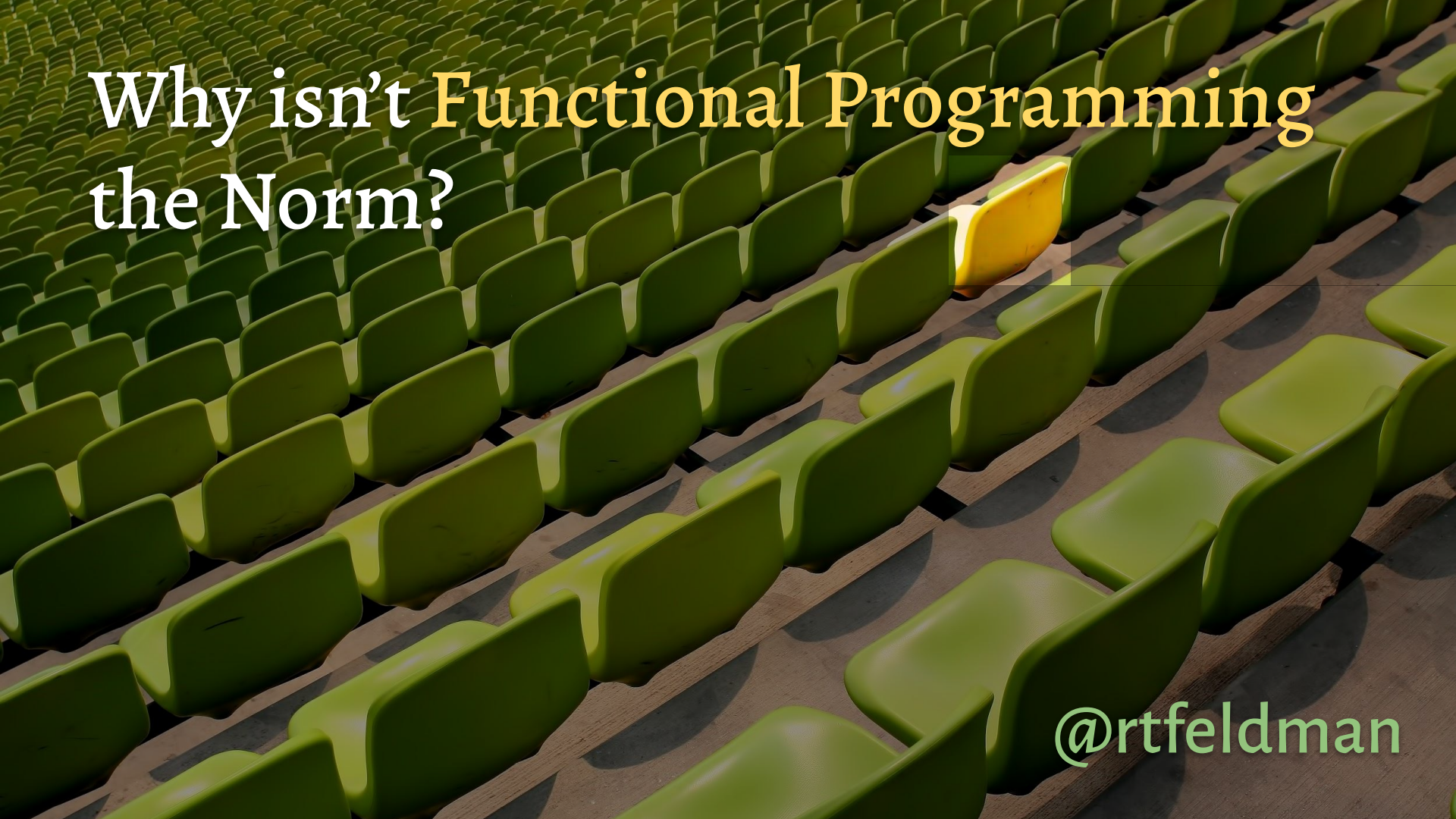

Why isn't Functional Programming the Norm?
@rtfeldman

# Why are things the way they are?

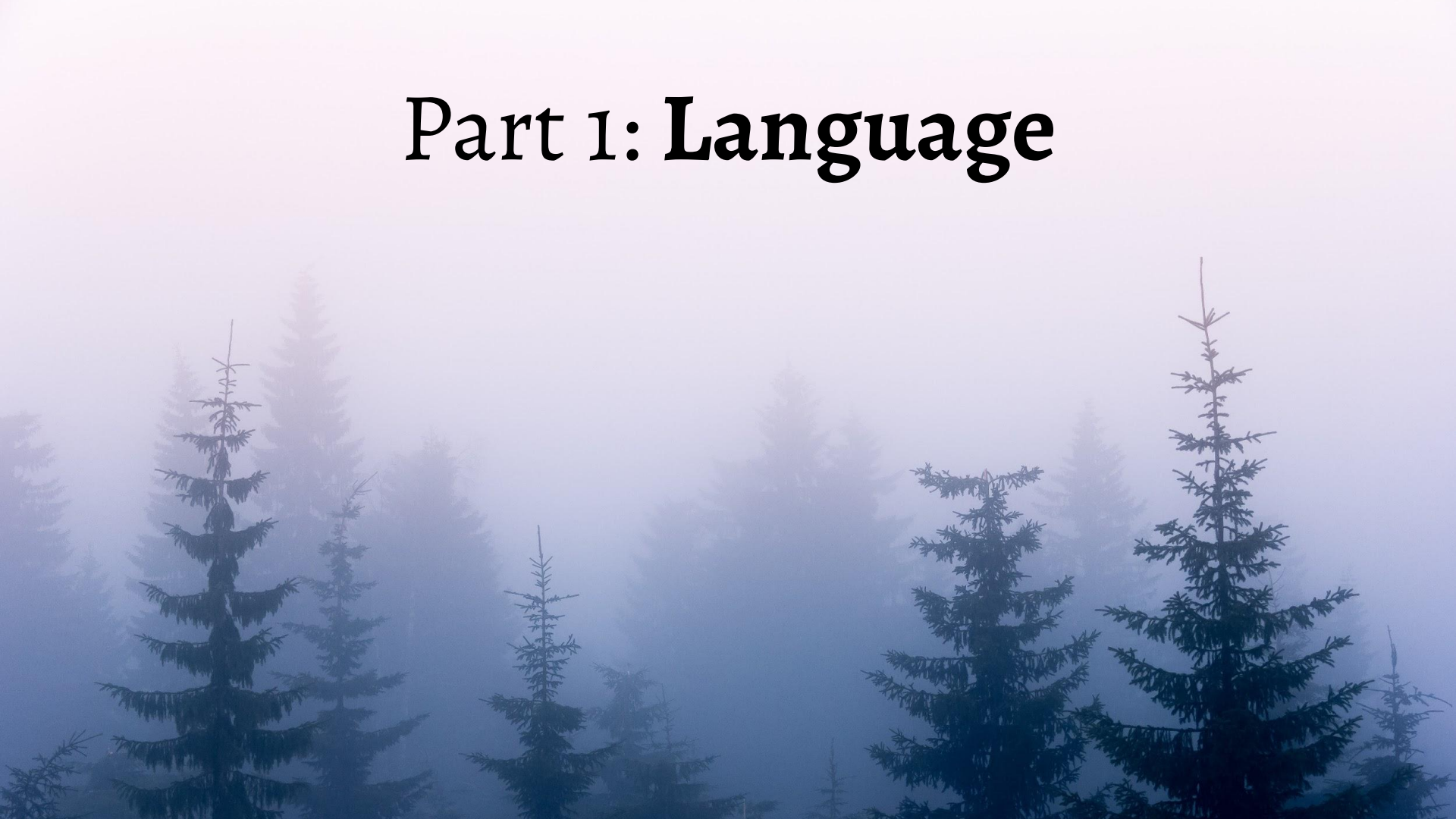

It’s complicated!

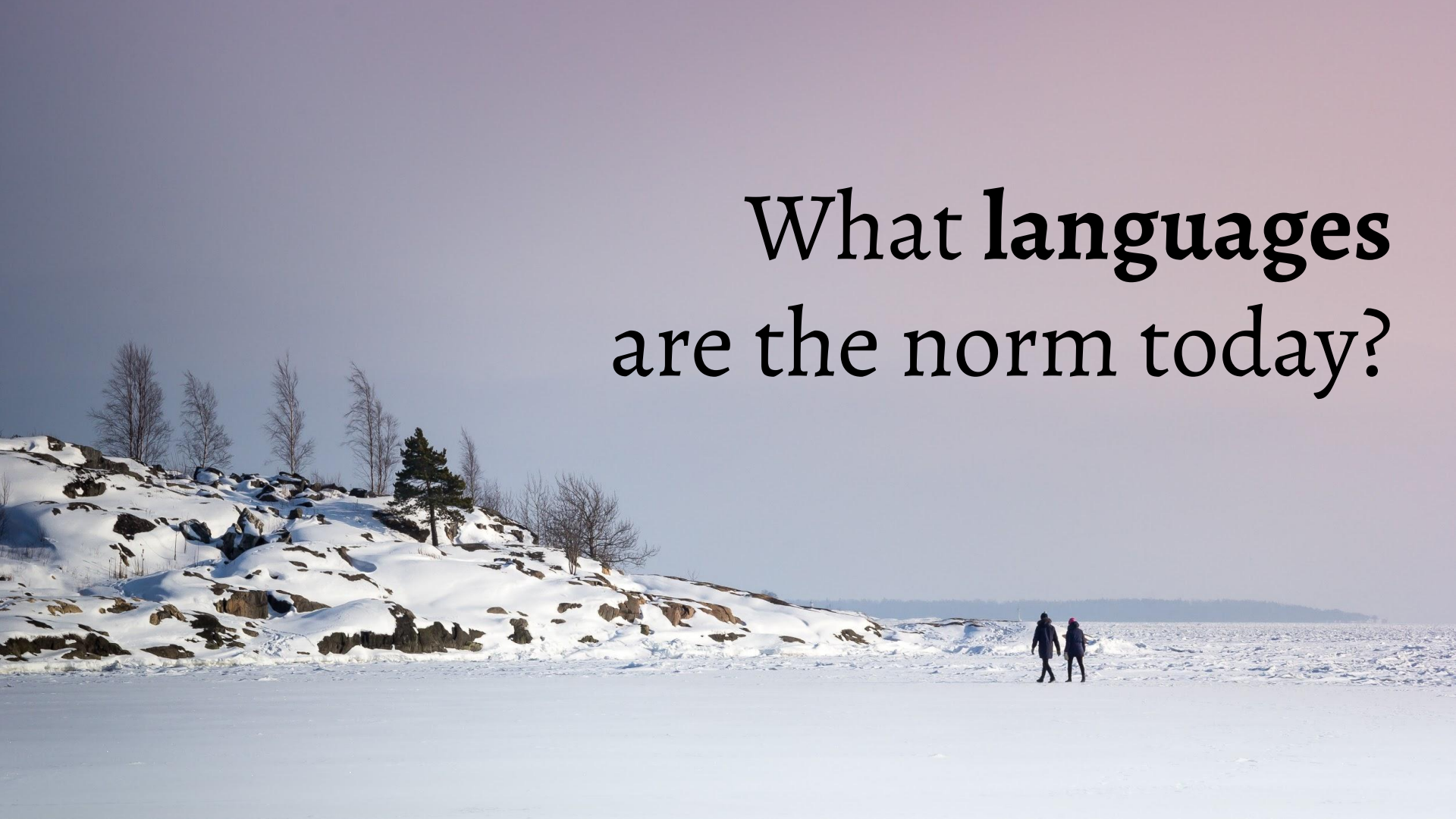
Part 1: **Language**

Part 2: **Paradigm**

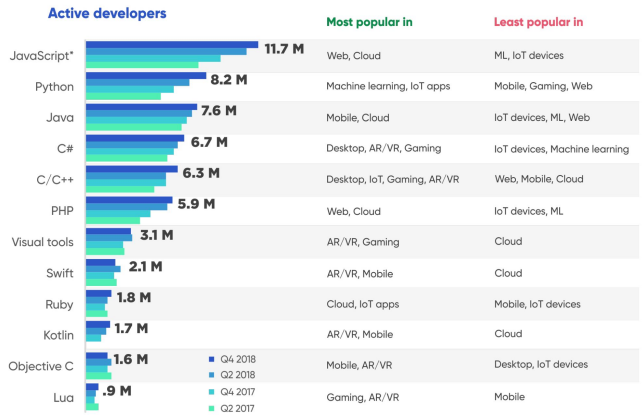
Part 3: **Style**

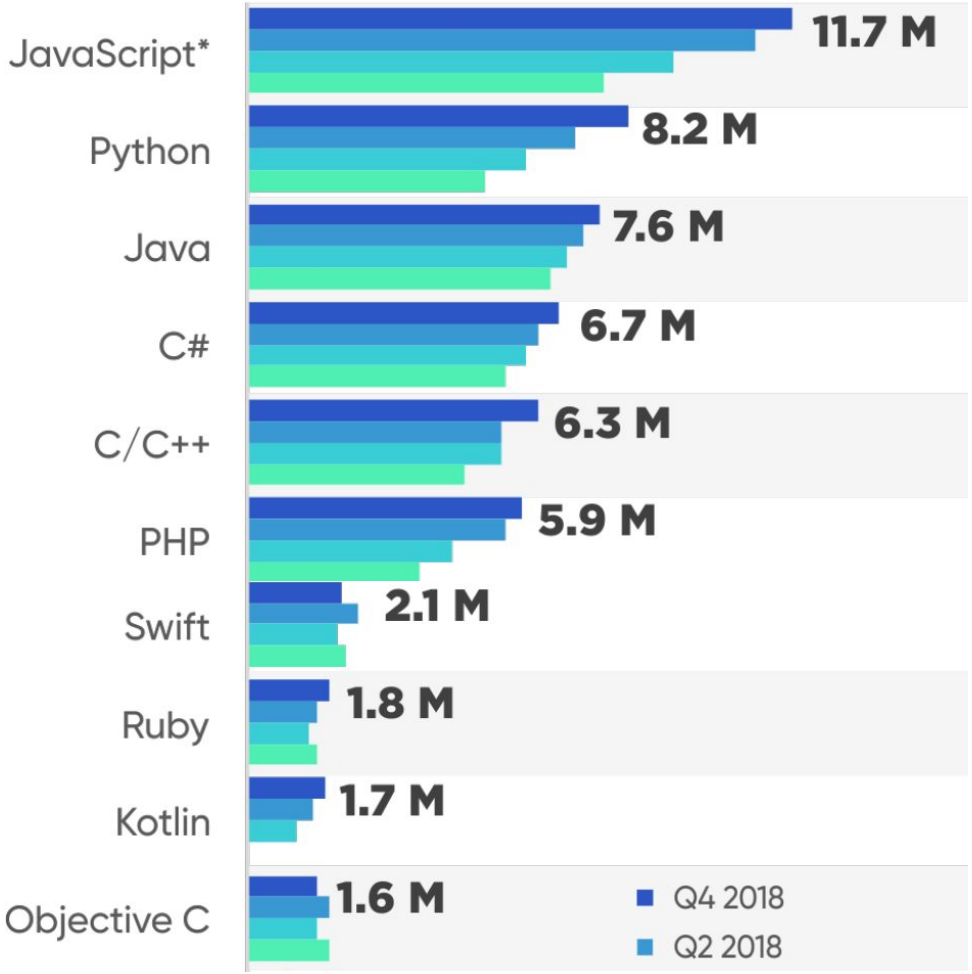
# Part 1: **Language**




What **languages**
are the norm today?

# Global developer population report 2019






# Total functional programming languages:

# 0

(*) JavaScript includes CoffeeScript, TypeScript



/DATA

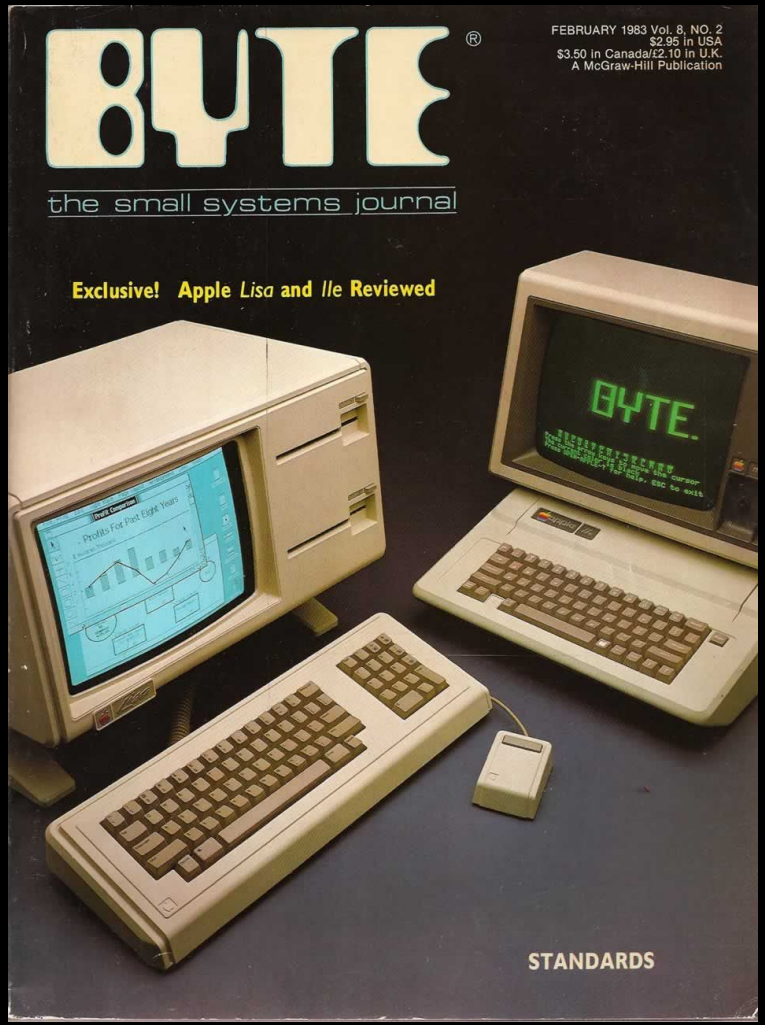
# How did they get popular?

1. Killer App
2. Platform Exclusivity
3. Quick Upgrade

# 1. Killer App


Software:
**$100**

Hardware:
**$10000**

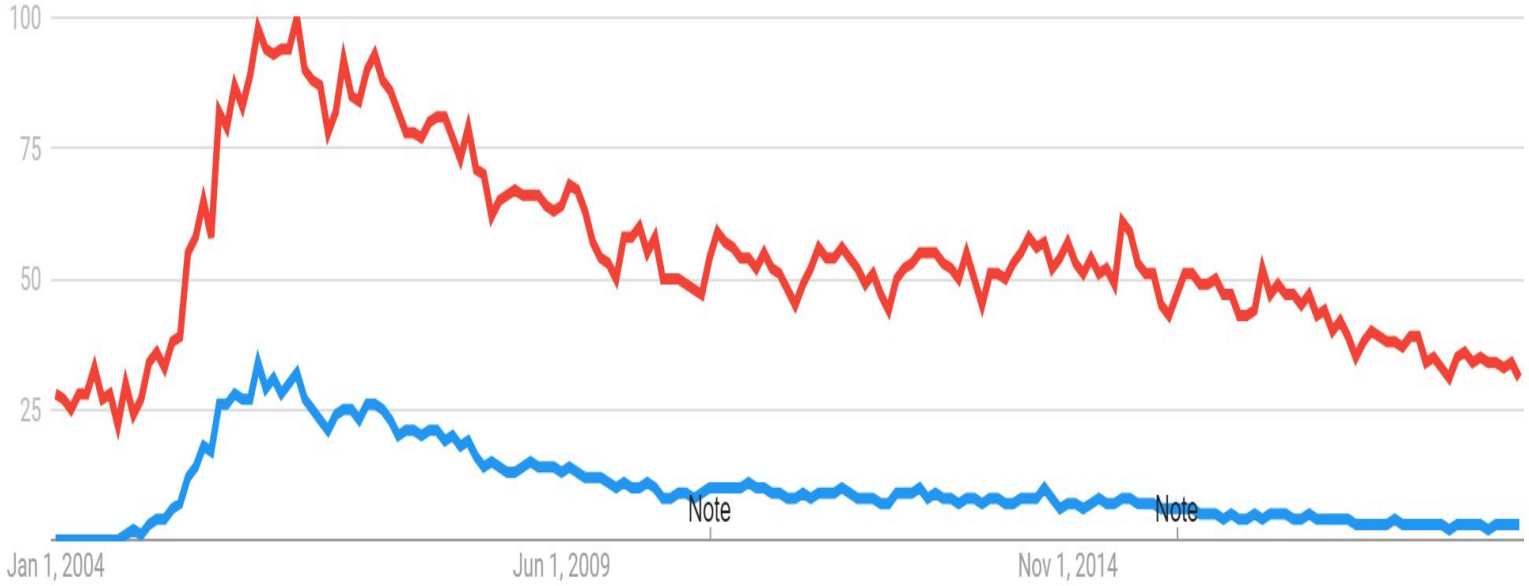



"VisiCalc is the first program available on a microcomputer that has been responsible for sales of entire systems."

-BYTE magazine, 1980

# the killer app for

Ruby

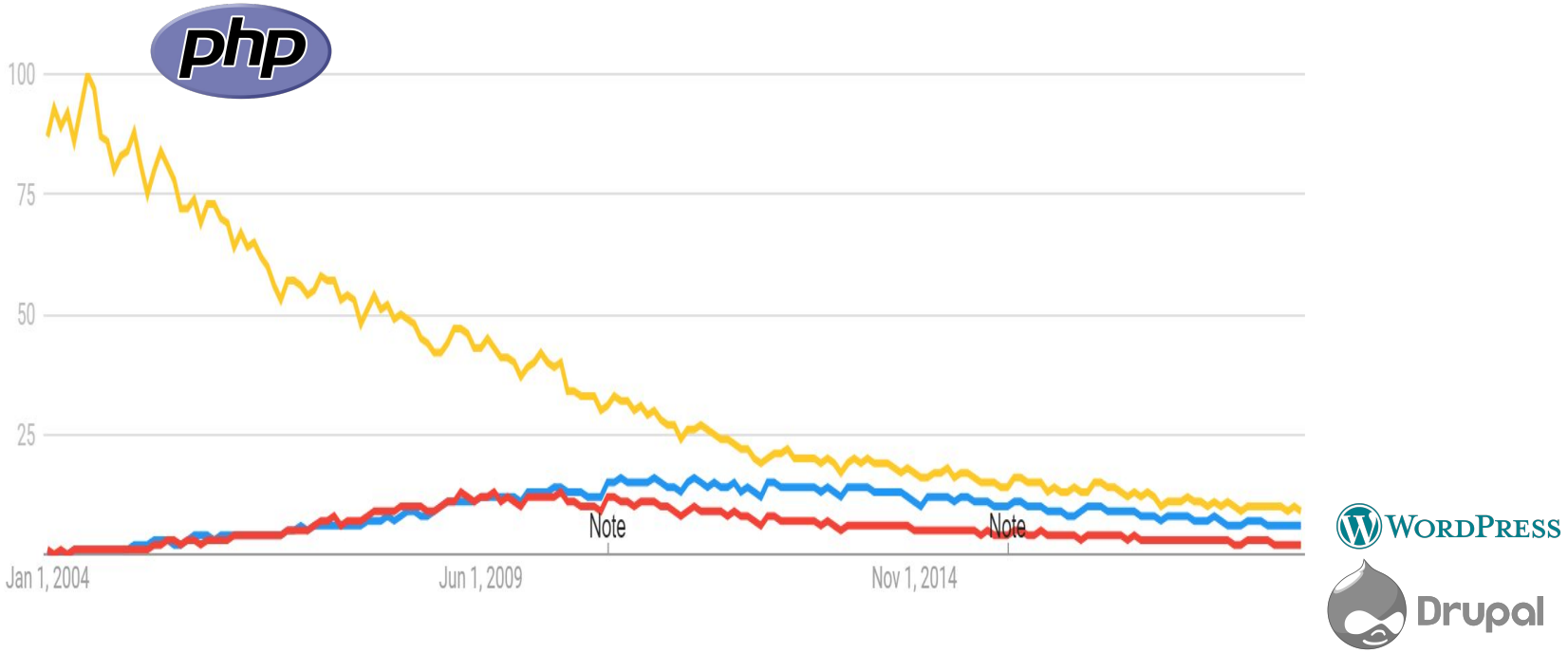



Ruby

RAILS

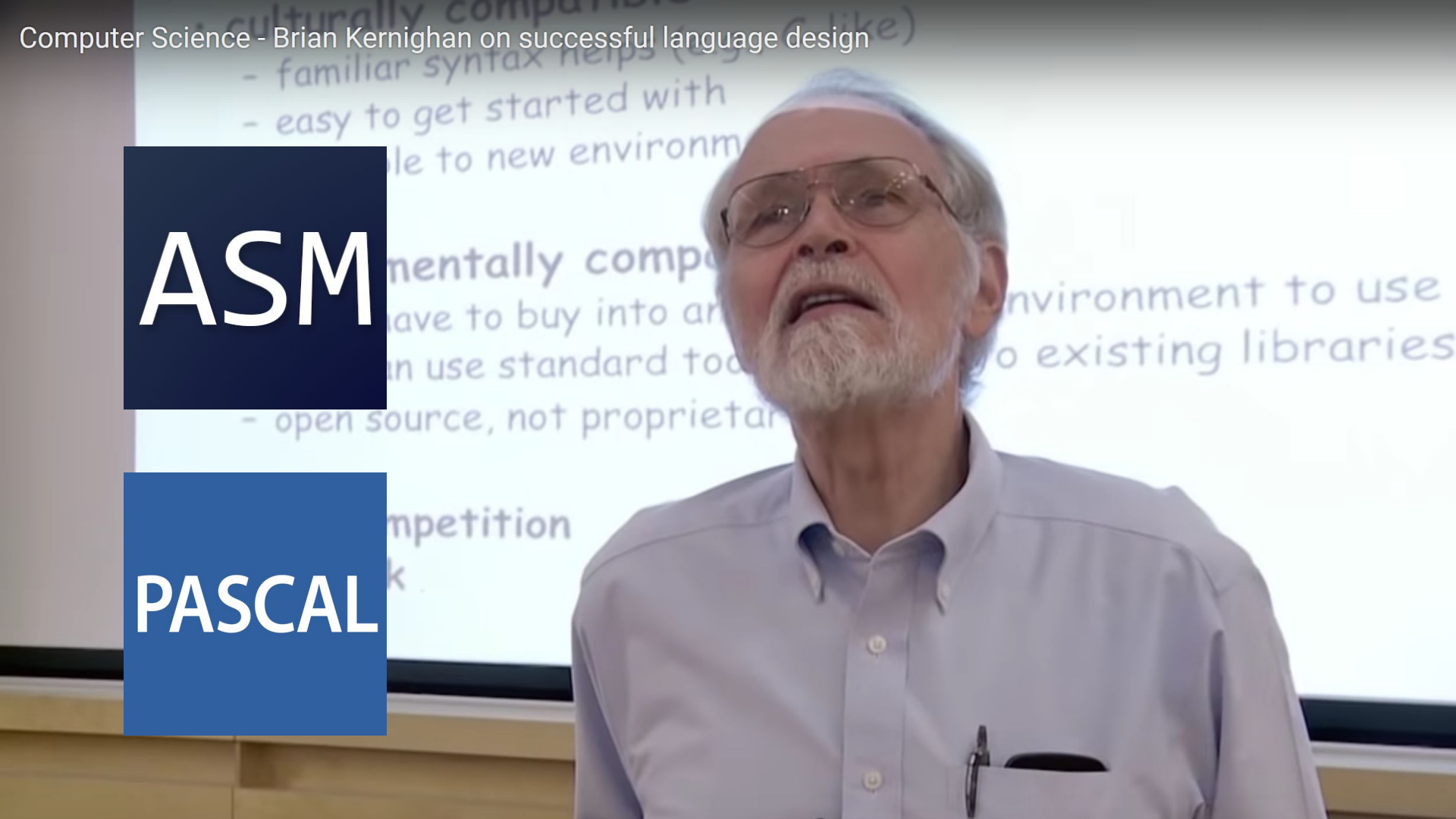
# the killer app for


# making HTML dynamic


php
100
75
50
25
Note
Note
Jan 1, 2004
Jun 1, 2009
Nov 1, 2014
WordPress
Drupal

# the killer app for

# systems programming


Computer Science - Brian Kernighan on successful language design
ASM
PASCAL

# Potential FP Killer Apps

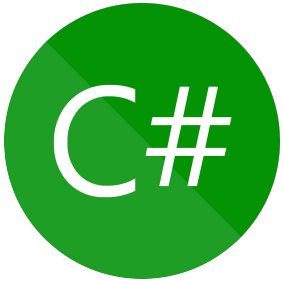
Elm `elm-ui`

Clojure Datomic

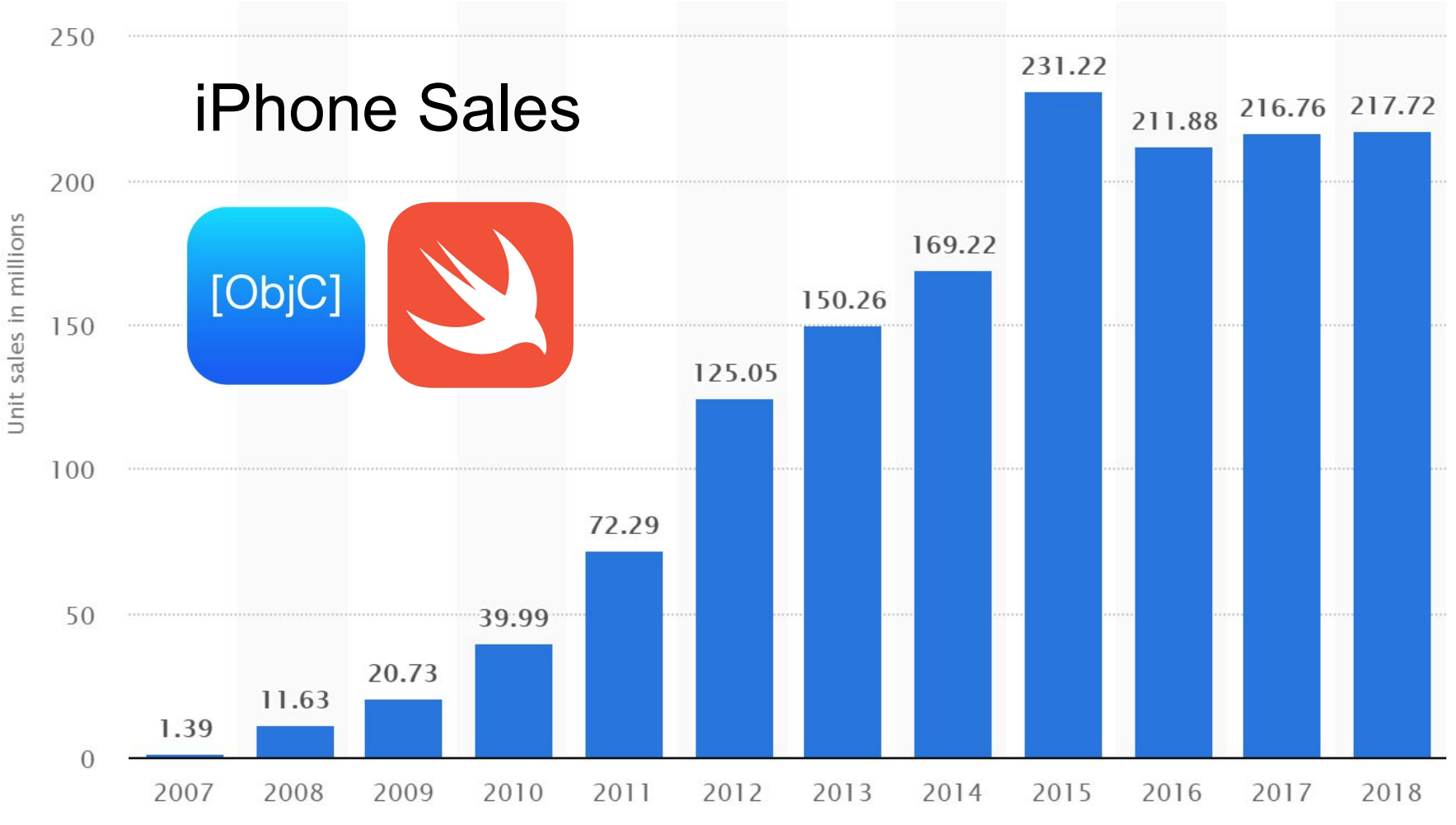

ReasonML `revery`

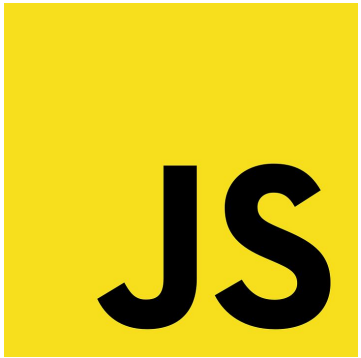
# 2. Platform Exclusivity


[ObjC]

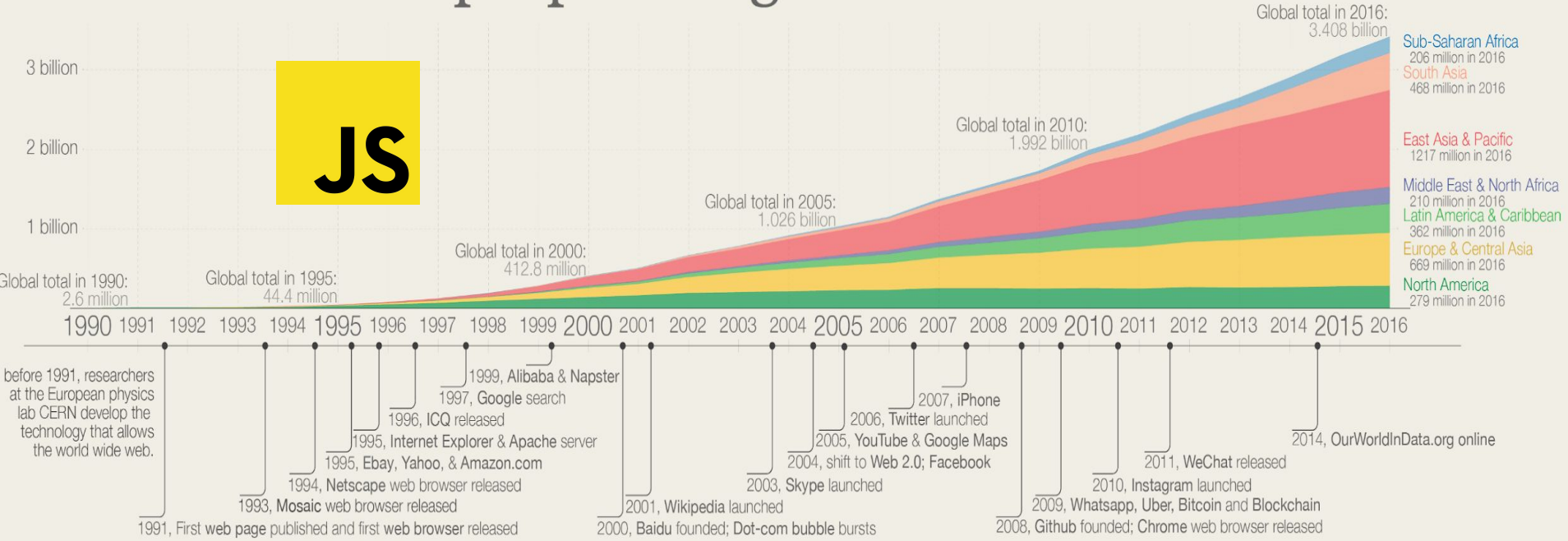

JS

C#

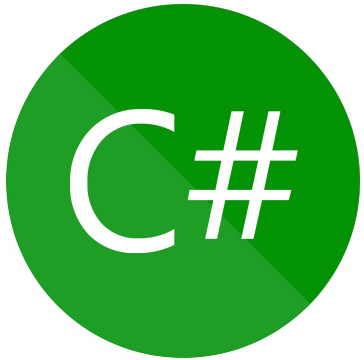

[ObjC]


iPhone Sales
[ObjC]
Unit sales in millions
250
200
150
100
50
0
1.39
11.63
20.73
39.99
72.29
125.05
150.26
169.22
231.22
211.88
216.76
217.72
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018


JS

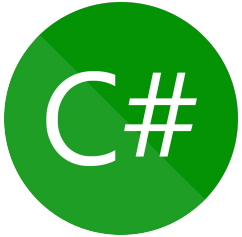
# Total number of people using the Internet




C#



"Write once,
run anywhere."



# 3. Quick Upgrade

# Considerations

"It's just JavaScript"


Benefits

Familiarity

Learning Curve

Ecosystem Access

Code Migration Effort

# Considerations

- Benefits
- Familiarity
- Learning Curve
- Ecosystem Access
- Code Migration Effort

"It's just JavaScript"

TypeScript

*strict superset of JS*

# Considerations

- Benefits
- Familiarity
- Learning Curve
- Ecosystem Access
- Code Migration Effort

*near-superset of C*

# Considerations

- Benefits
- Familiarity
- Learning Curve
- Ecosystem Access
- Code Migration Effort

Kotlin

"Kotlin is **100% interoperable** with [Java] and major emphasis has been placed on making sure that your **existing codebase** can interact properly with Kotlin."

# How did they get popular?

| | |
|---|---|
| 1. Killer App | C, Ruby, PHP |
| 2. Platform Exclusivity | JS, Objective-C, Swift, C# |
| 3. Quick Upgrade | C++, Kotlin, TypeScript |

# How did they get popular?

| | |
|---|---|
| 1. Killer App | C, Ruby, PHP |
| 2. Platform Exclusivity | JS, Objective-C, Swift, C# |
| 3. Quick Upgrade | C++, Kotlin, TypeScript |
| 4. Epic Marketing | Java |

THE WALL STREET JOURNAL.

U.S. Edition ▾ | September 26, 2019 | Print Edition | Video



# Sun Microsystems Boots Up Blitz To Create 'Java Powered' Loyalty

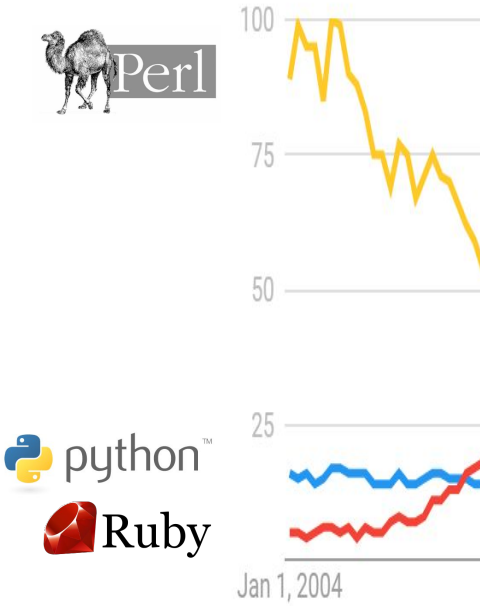

$500M Java marketing campaign in 2003


Java™


JavaScript™

# How did they get popular?

| | |
|---|---|
| 1. Killer App | C, Ruby, PHP |
| 2. Platform Exclusivity | JS, Objective-C, Swift, C# |
| 3. Quick Upgrade | C++, Kotlin, TypeScript |
| 4. Epic Marketing | Java |
| 5. Slow & Steady | Python |

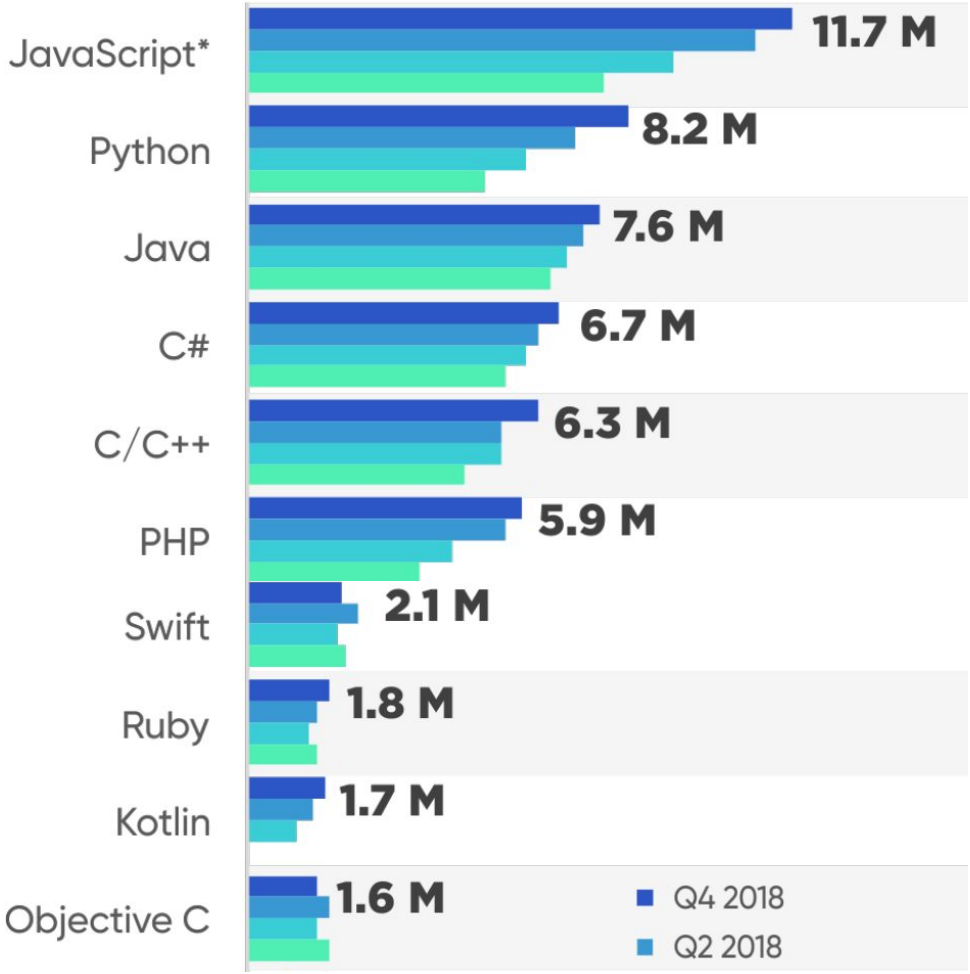

Perl
100
75
50
25
python
Ruby
Jan 1, 2004

# Other Popularity Factors

Syntax

Job Market

Community



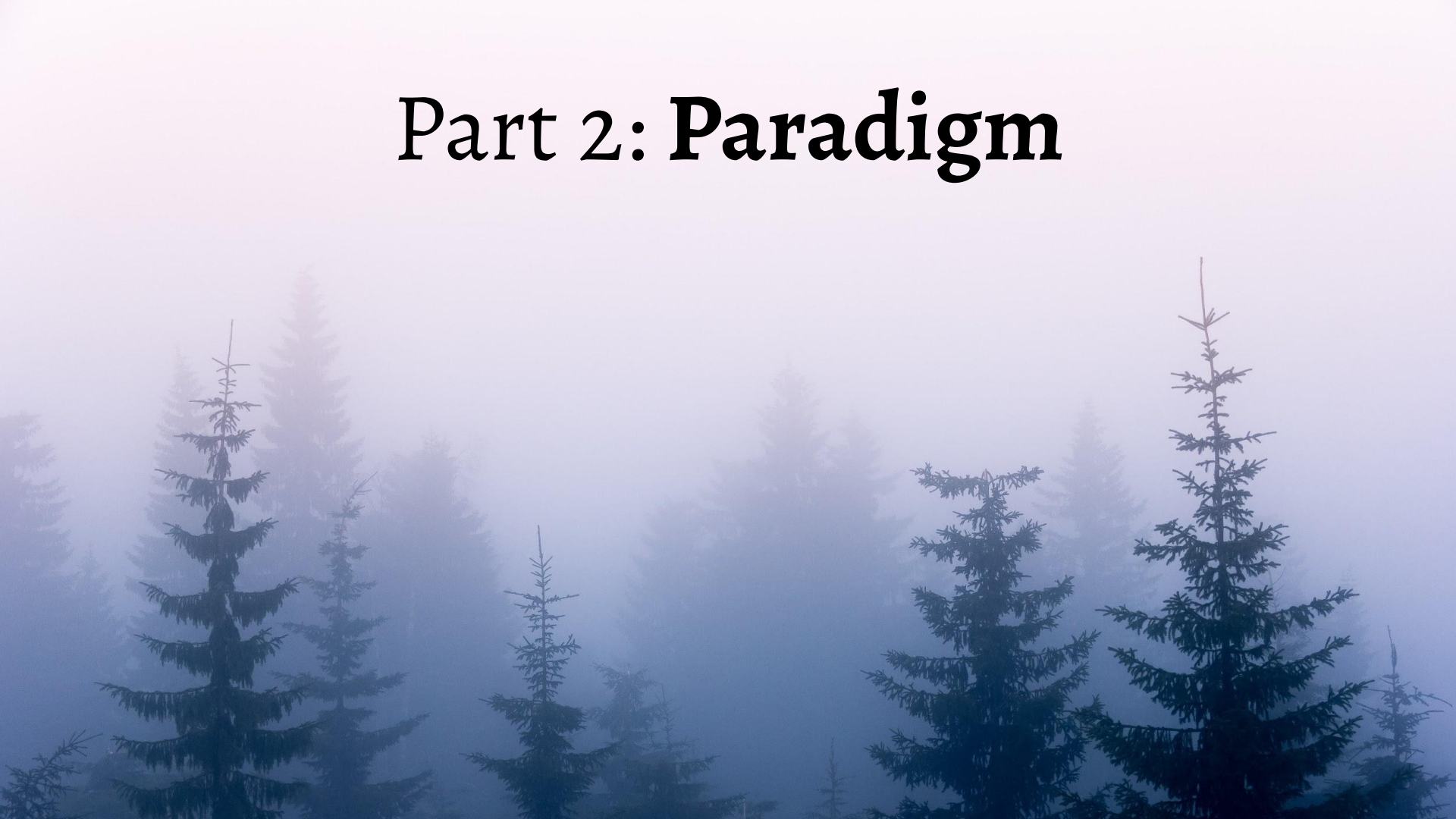

# Why are the most popular languages **OO** (except C)?

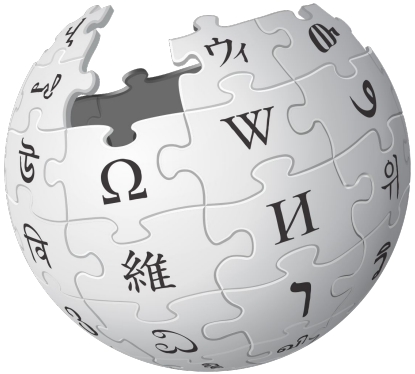
(*) JavaScript includes CoffeeScript, TypeScript

**Global developer population report 2019 .** https://sdata.me/GlobalDevPop19 · ©SlashData

/DATA

# Part 2: **Paradigm**





Programming paradigms are a way to
classify programming languages based on their features.
Languages can be classified into multiple paradigms.

Are OO languages the norm because of **uniquely OO features**?

# What are **uniquely OO features**?

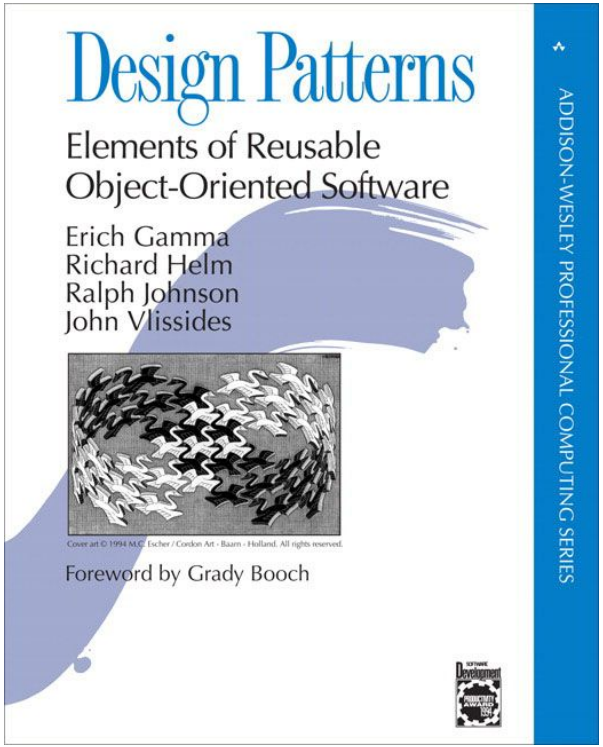
1. Encapsulation?
2. Inheritance?
3. Objects?
4. Methods?

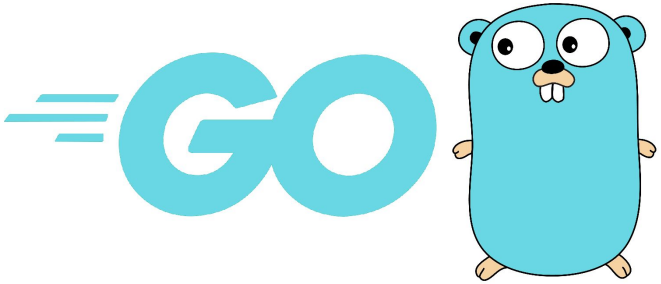
# Inheritance

*Interface* Inheritance (aka **subtyping**)

*Implementation* Inheritance (aka **inheritance**)

# “Composition over inheritance”





“Supports an **object-oriented style**” encapsulation?

**No inheritance**

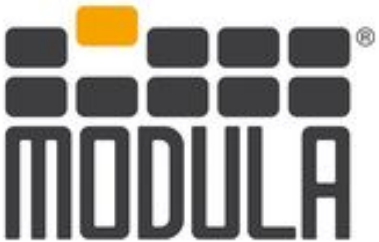
“Objects and methods” are

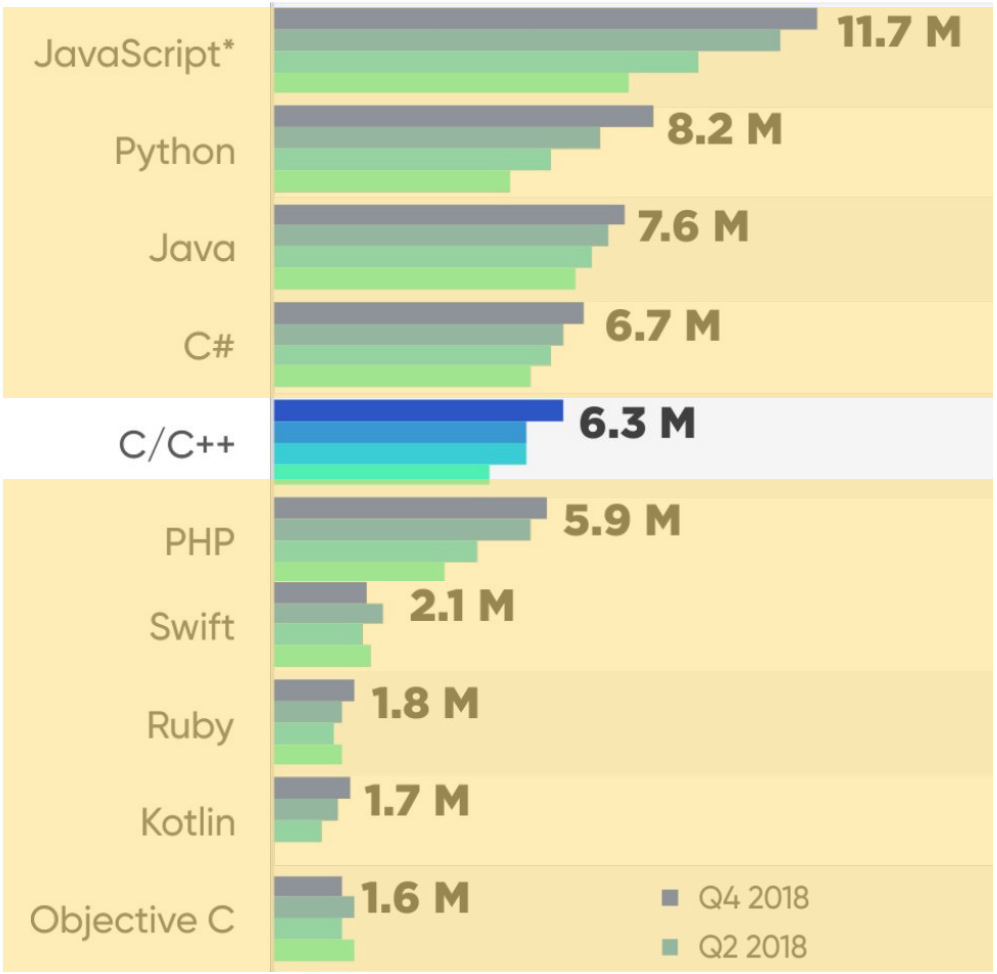
```
circle.grow(3)
grow(circle, 3)
```

syntax sugar for structs and procedures

# Modular Programming

*Modularity* lets you define a **public interface** to hide **private implementation details**






# Languages with modules

*coming in C++20

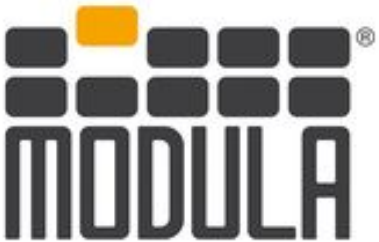
(*) JavaScript includes CoffeeScript, TypeScript

**Global developer population report 2019 .** https://sdata.me/GlobalDevPop19 · ©SlashData

/DATA

# Modular Programming

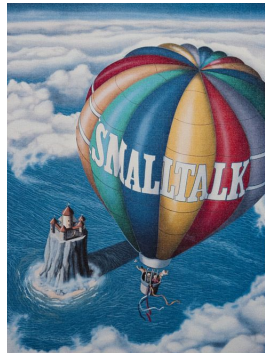
*Modularity* lets you define a **public interface** to hide **private implementation details**



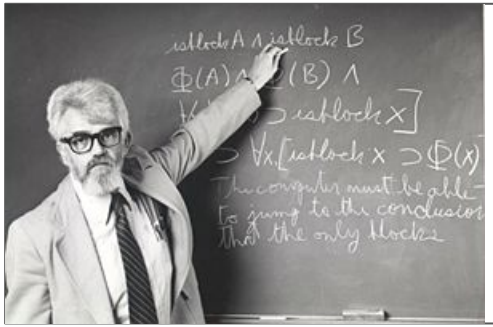
# Encapsulation

*Encapsulation* lets you define a **public interface** to hide **private implementation details** about an object's state

ALGOL → simula →




Objects
Classes
Inheritance
Garbage Collection



John McCarthy,
Creator of **Lisp**

ALGOL → simula →

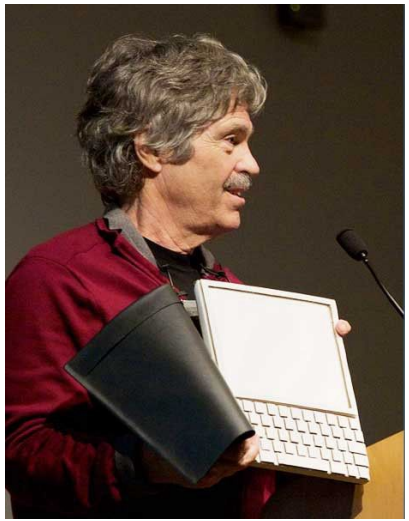

Objects
Classes
Inheritance
Garbage Collection

bit.ly/2kqxVWO

**everything** is an object

# “Object-Oriented” -Alan Kay

*OOP to me means only **messaging, local retention** and **protection and hiding of state-process**, and **extreme late-binding of all things**. It can be done in Smalltalk and in LISP. There are possibly other systems in which this is possible, but I’m not aware of them.*



ALGOL → simula → 

→ [ObjC]

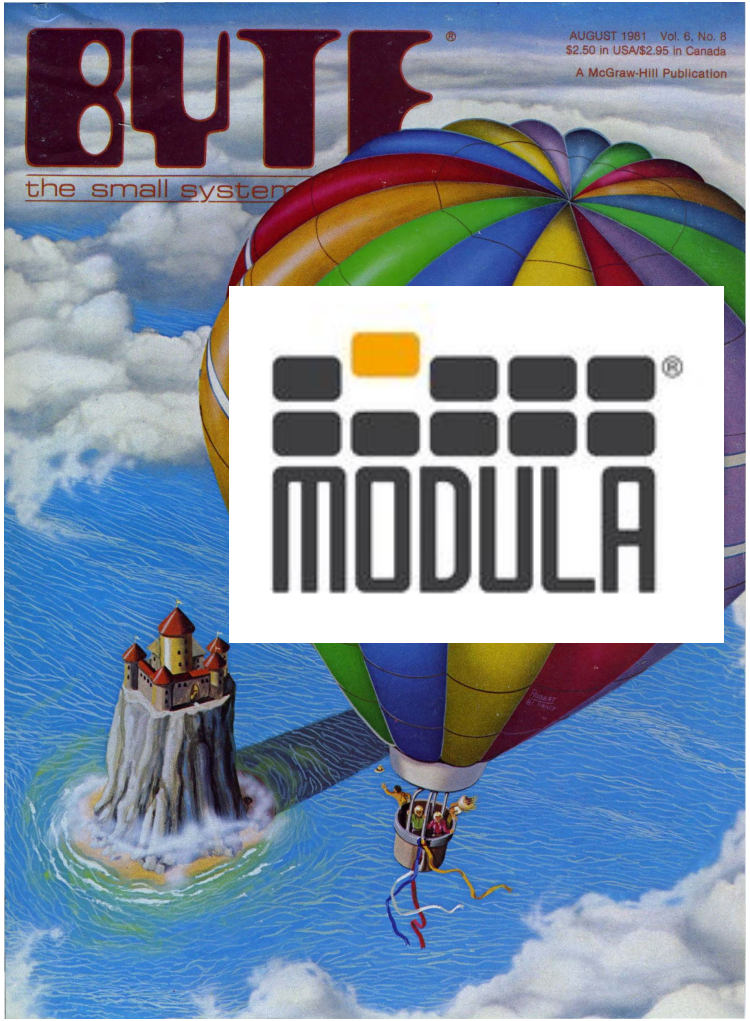
youtu.be/1xrL2d5omuA

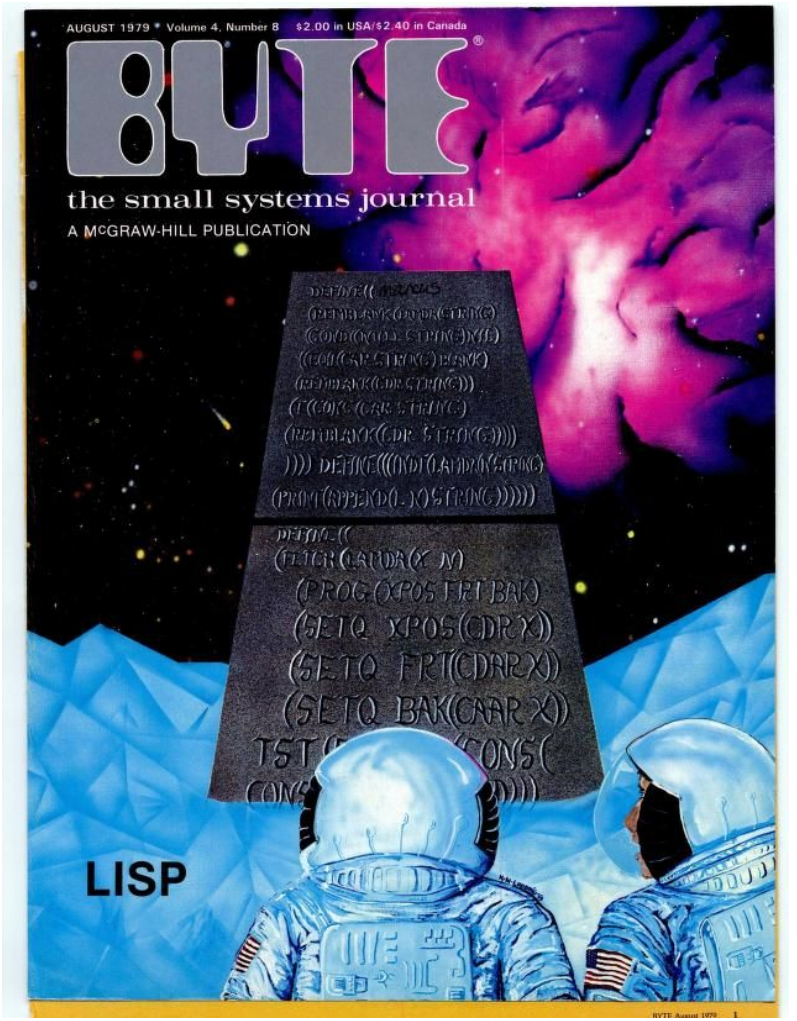
Objects
Classes
Inheritance
Garbage Collection

# Brad Cox

"I wasn't happy with **C as a productivity foundation**, and I was lobbying around for **anything that could help**. That was about the time the **Byte Magazine** issue came out."
"There were a whole bunch of things I thought vaguely might help. **Encapsulation** for sure…C is so bad at it. **Everything is public**; it just turns into soup.
**The pain of that is what I was trying to escape.**"

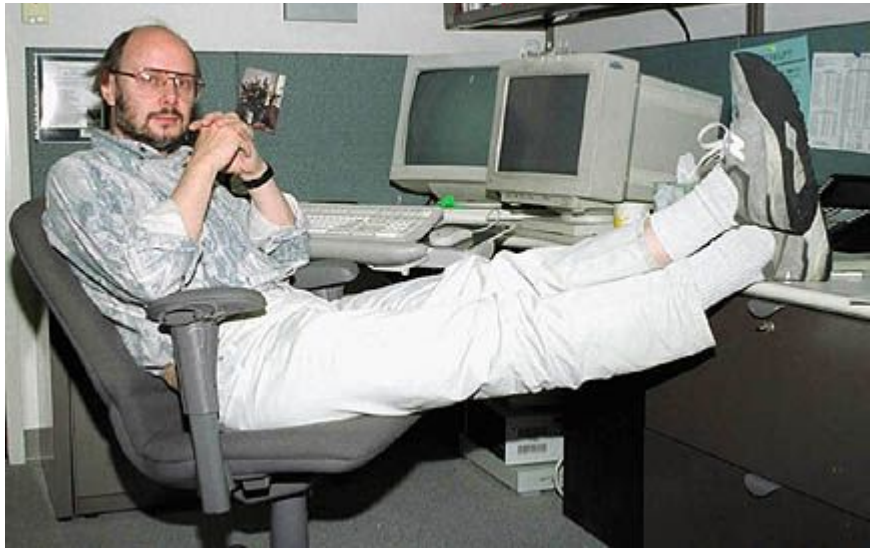
UNIX → C


BYTE
the small system
AUGUST 1981 Vol. 6, No. 8
$2.50 in USA/$2.95 in Canada
A McGraw-Hill Publication
MODULA


AUGUST 1979 • Volume 4, Number 8 $2.00 in USA/$2.40 in Canada
BYTE
the small systems journal
A McGRAW-HILL PUBLICATION
LISP

simula

Objects
Classes
Inheritance
Garbage Collection



## C with Classes: Why Classes?

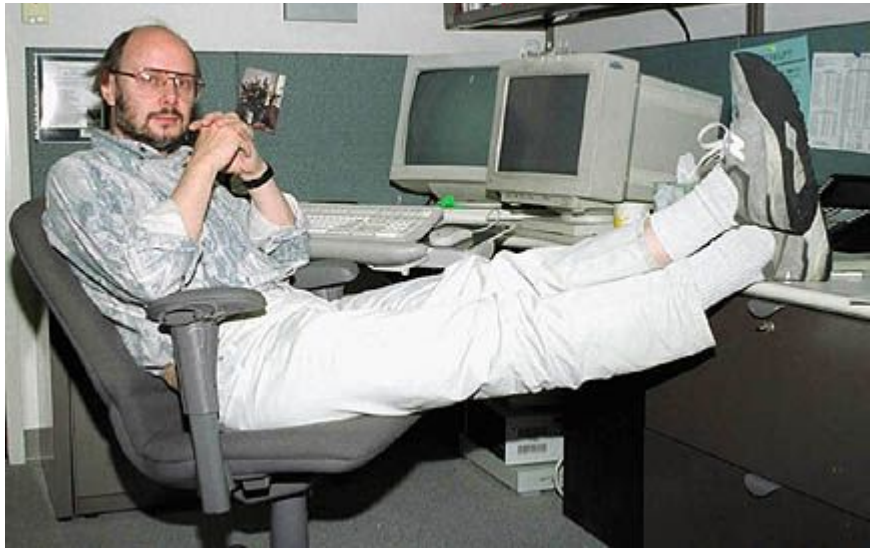
- Program organization
- Mapping of concepts
  - A class is a type
  - Static checking

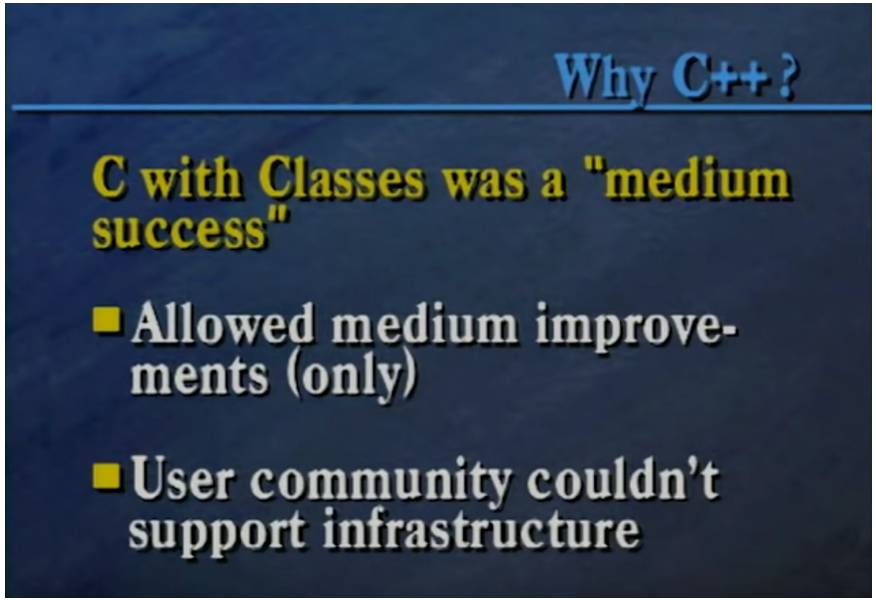
Bjarne Stroustrup

simula → With Classes → C++

youtu.be/69edOm889V4

# Drop-in C Replacement (!)






Bjarne Stroustrup

UNIX

C with **OOP** (“C with Classes”) **wasn’t sufficient** for popularity

Sun wanted **familiarity** for C++ programmers


Microsoft wanted a **proprietary Java** alternative

UNIX

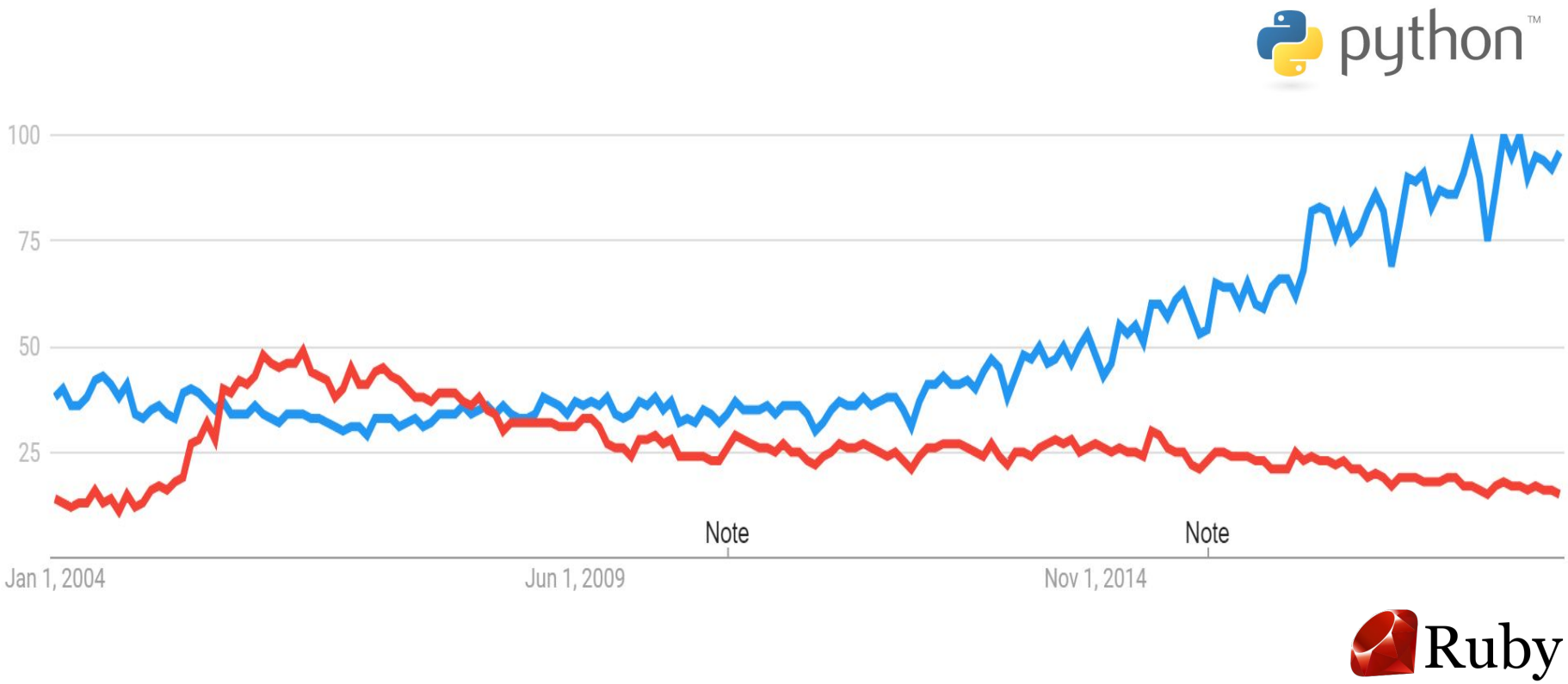

Brad Cox wanted **modularity**

Apple wanted to **improve on ObjC**


python™

“I wanted a scripting language that was more powerful than Perl, and **more object-oriented than Python**.”

—Yukihiro “Matz” Matsumoto, creator of **Ruby**

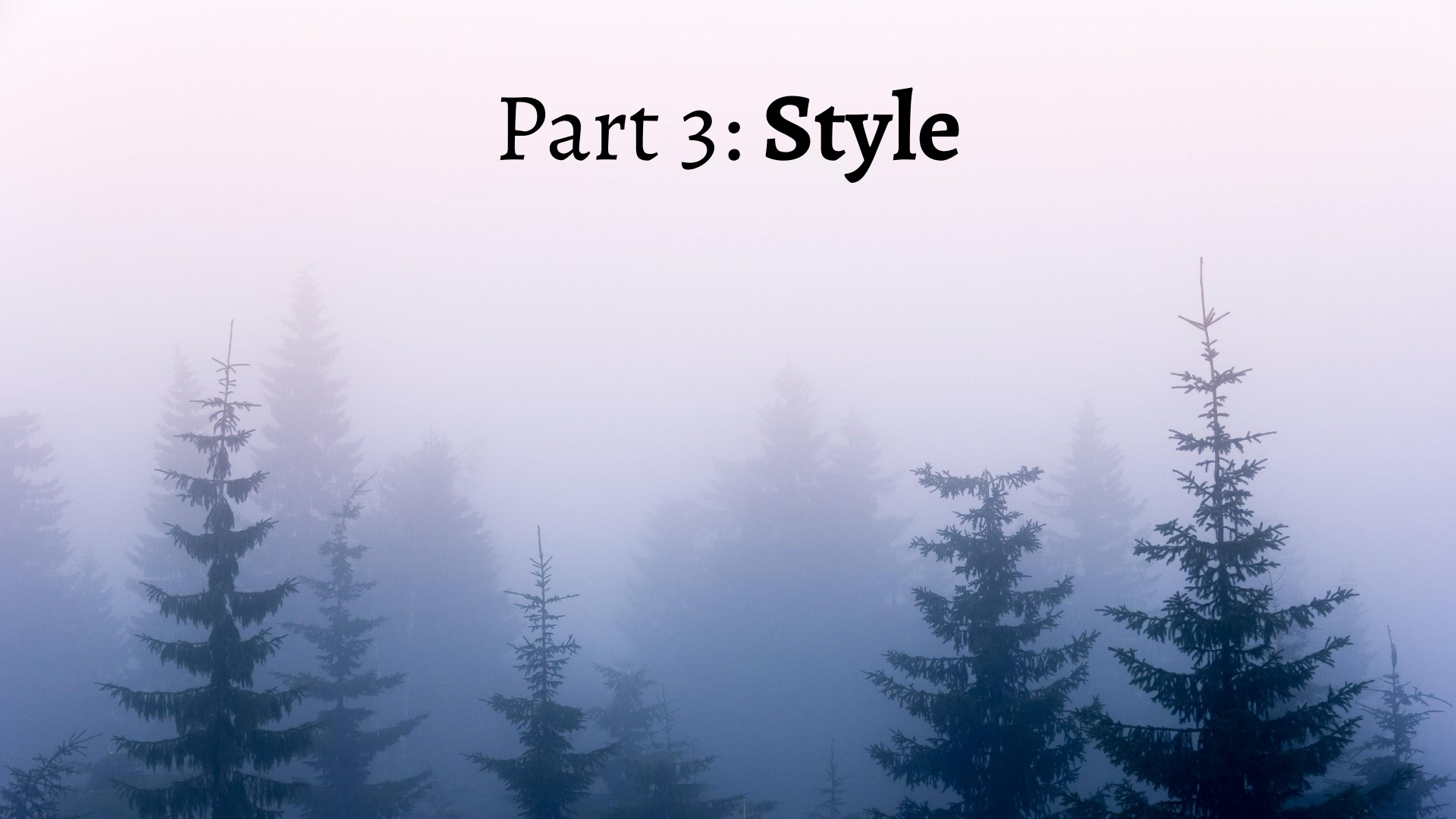

python™
100
75
50
25
Note
Note
Jan 1, 2004
Jun 1, 2009
Nov 1, 2014
Ruby

Are OO languages the norm because of **uniquely OO features**?

**No.**

They're OO because **modularity is a good idea**, and they originally got it from OO **by chance**.

# Part 3: **Style**



# Functional Programming **Style**

"Avoid mutation and side effects."

No language features **required**

Languages differ in their **support** for this style

# Why isn't FP **style** the norm?

# Kotlin FAQ

**Is Kotlin an object-oriented language or a functional one?**

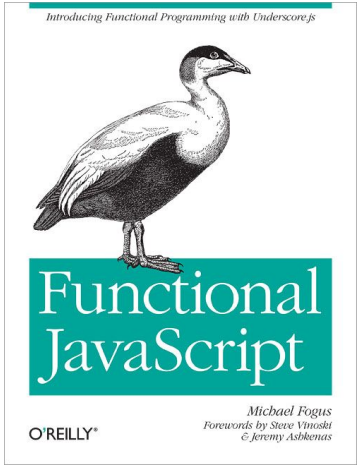
Kotlin has **both** object-oriented and functional constructs.

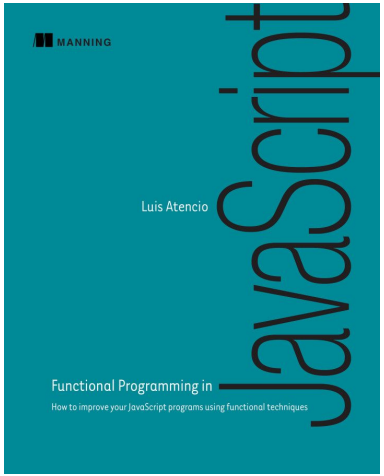
You can use it in both OO and FP styles, or mix elements of the two.

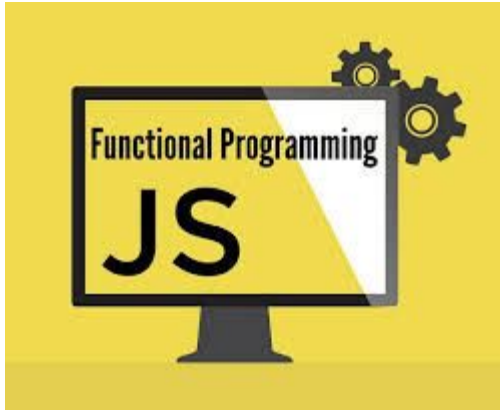
# swift.org

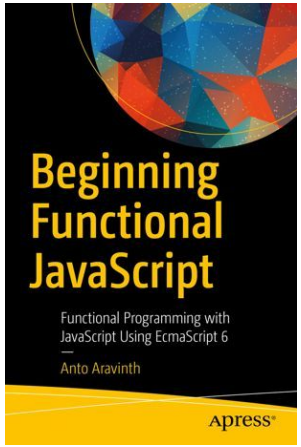
**Some additional features of Swift include:**

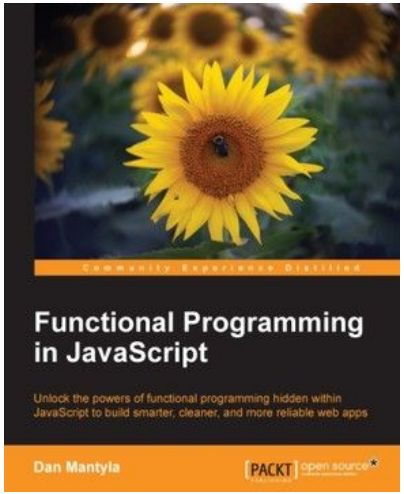
Functional programming patterns, e.g., **`map`** and **`filter`**

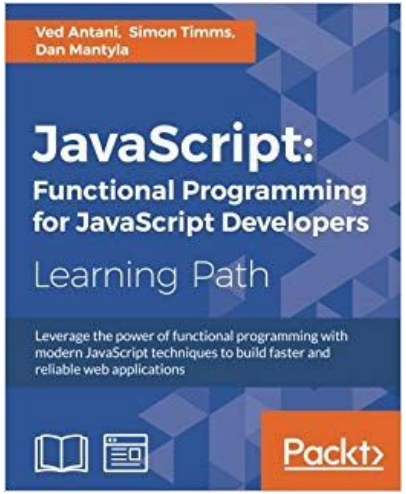

Introducing Functional Programming with Underscore.js
Functional JavaScript
O'REILLY
Michael Fogus
Forewords by Steve Vinoski & Jeremy Ashkenas

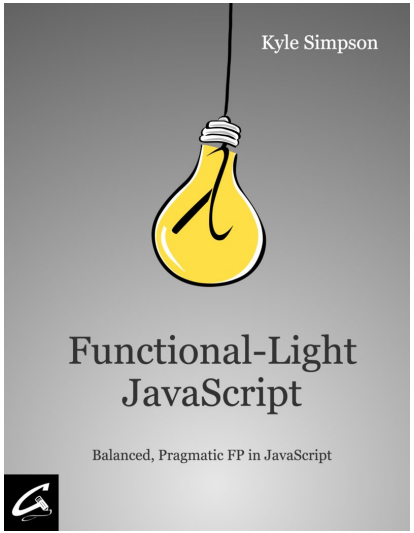

MANNING
Luis Atencio
Functional Programming in JavaScript
How to improve your JavaScript programs using functional techniques

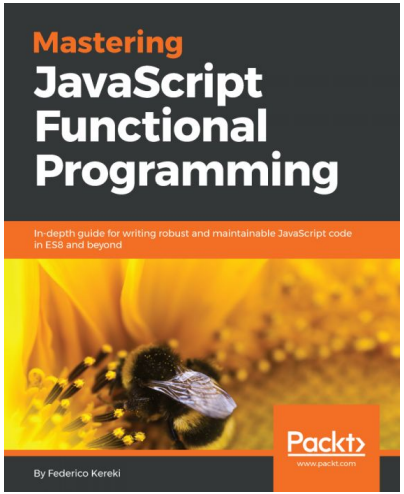

Functional Programming
JS

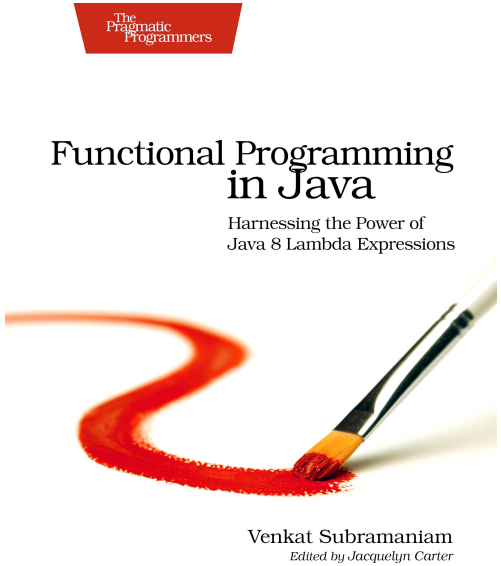

Beginning Functional JavaScript
Functional Programming with JavaScript Using ECMAScript 6
Anto Aravinth
Apress

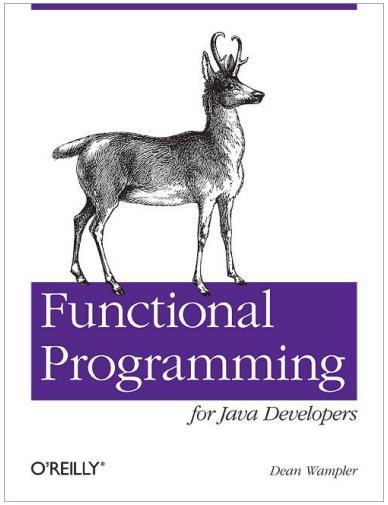

Functional Programming in JavaScript
Unlock the powers of functional programming hidden within JavaScript to build smarter, cleaner, and more reliable web apps
Dan Mantyla
PACKT open source

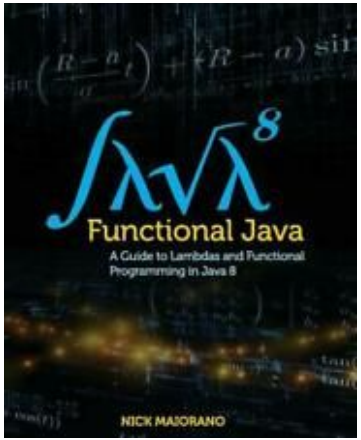

Ved Antani, Simon Timms, Dan Mantyla
JavaScript: Functional Programming for JavaScript Developers
Learning Path
Leverage the power of functional programming with modern JavaScript techniques to build faster and reliable web applications
Packt


Kyle Simpson
Functional-Light JavaScript
Balanced, Pragmatic FP in JavaScript

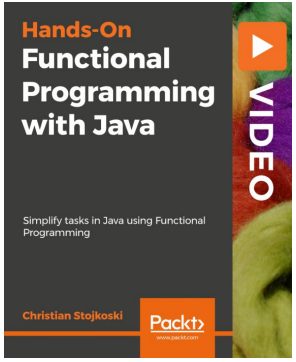

Mastering JavaScript Functional Programming
In-depth guide for writing robust and maintainable JavaScript code in ES8 and beyond
By Federico Kereki
Packt
www.packt.com

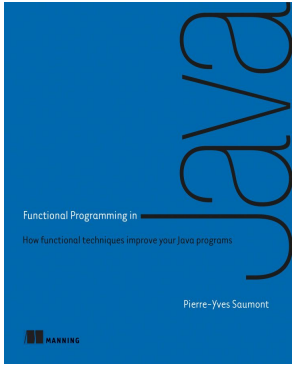

The Pragmatic Programmers
Functional Programming in Java
Harnessing the Power of Java 8 Lambda Expressions
Venkat Subramaniam
Edited by Jacquelyn Carter

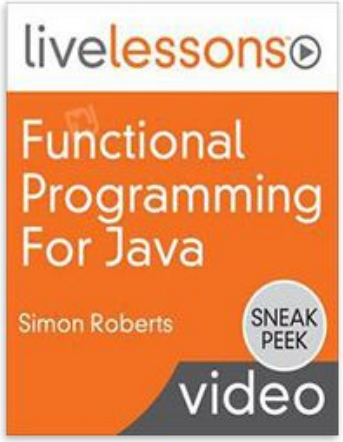

Functional Programming
for Java Developers
O'REILLY
Dean Wampler


Java 8
Functional Java
A Guide to Lambdas and Functional Programming in Java 8
NICK MAJORANO

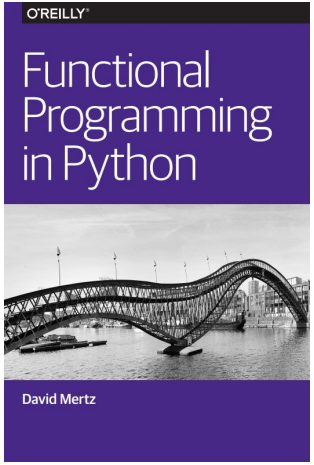

Colibri Ltd
Learning
Java 9 - Functional Programming
Create robust and maintainable Java applications using the functional programming style
Packt


Hands-On
Functional Programming with Java
VIDEO
Simplify tasks in Java using Functional Programming
Christian Stojkoski
Packt

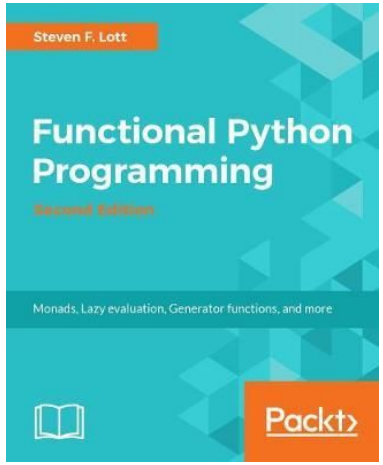

Java
Functional Programming in
How functional techniques improve your Java programs
Pierre-Yves Saumont
MANNING

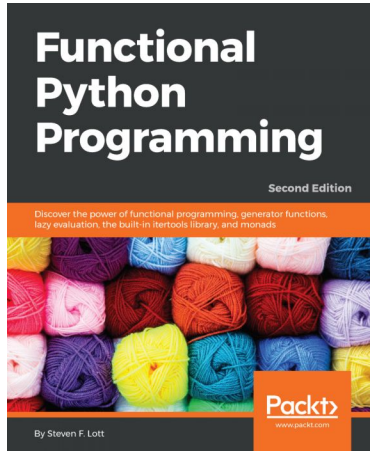

livelessons
Functional Programming For Java
Simon Roberts
SNEAK PEEK
video

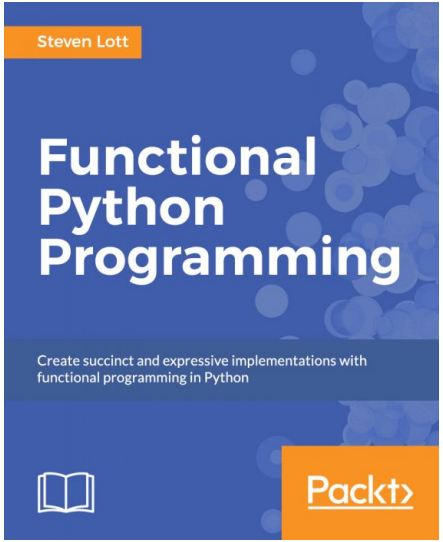

In28Minutes
Java 9
Functional
Programming


O'REILLY
Functional Programming in Python
David Mertz


O'REILLY
Functional Programming in Python
David Mertz
REPORT

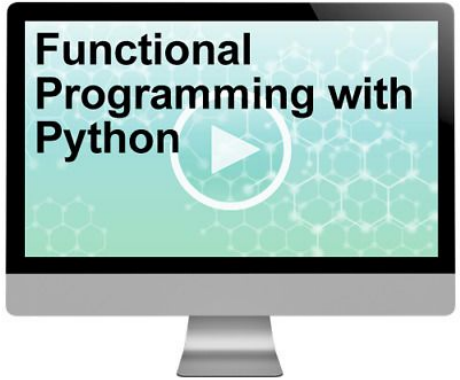

Steven F. Lott
Functional Python Programming
Second Edition
Monads, Lazy evaluation, Generator functions, and more
Packt

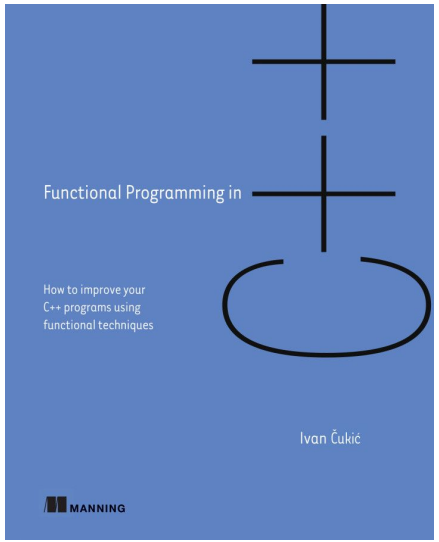

Functional Python Programming
Second Edition
Discover the power of functional programming, generator functions, lazy evaluation, the built-in itertools library, and monads
By Steven F. Lott
Packt

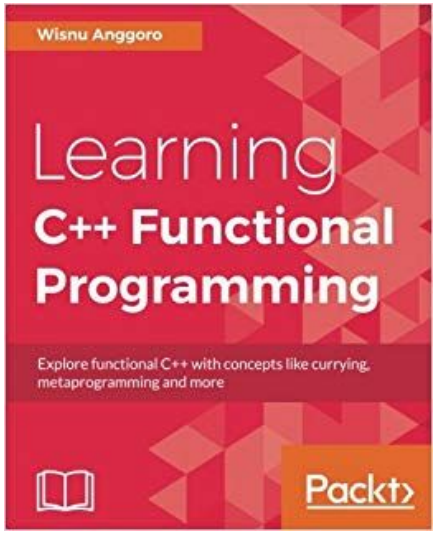

Steven Lott
Functional Python Programming
Create succinct and expressive implementations with functional programming in Python
Packt

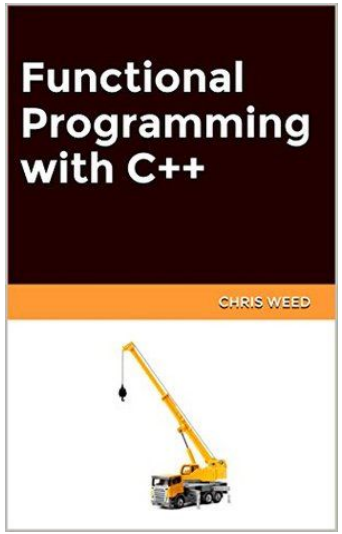

O'REILLY
Functional Programming with Python
VIDEO TRAINING

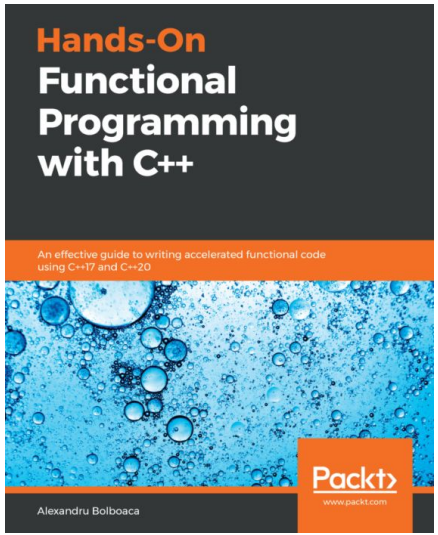

Sebastiaan Mathôt
Functional Programming in Python
Learn everything there is to know about Functional Programming in Python
Packt


Functional Programming with Python

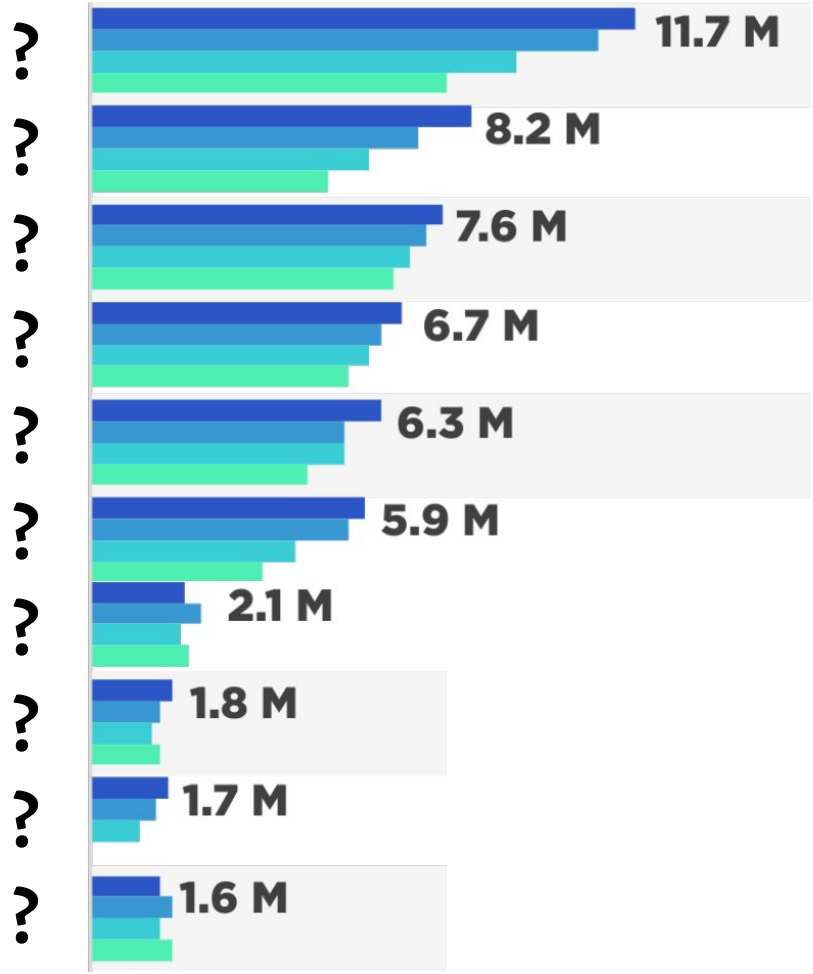

Functional Programming in C++
How to improve your C++ programs using functional techniques
Ivan Čukić
MANNING


Wisnu Anggoro
Learning C++ Functional Programming
Explore functional C++ with concepts like currying, metaprogramming and more
Packt


Functional Programming with C++
CHRIS WEED

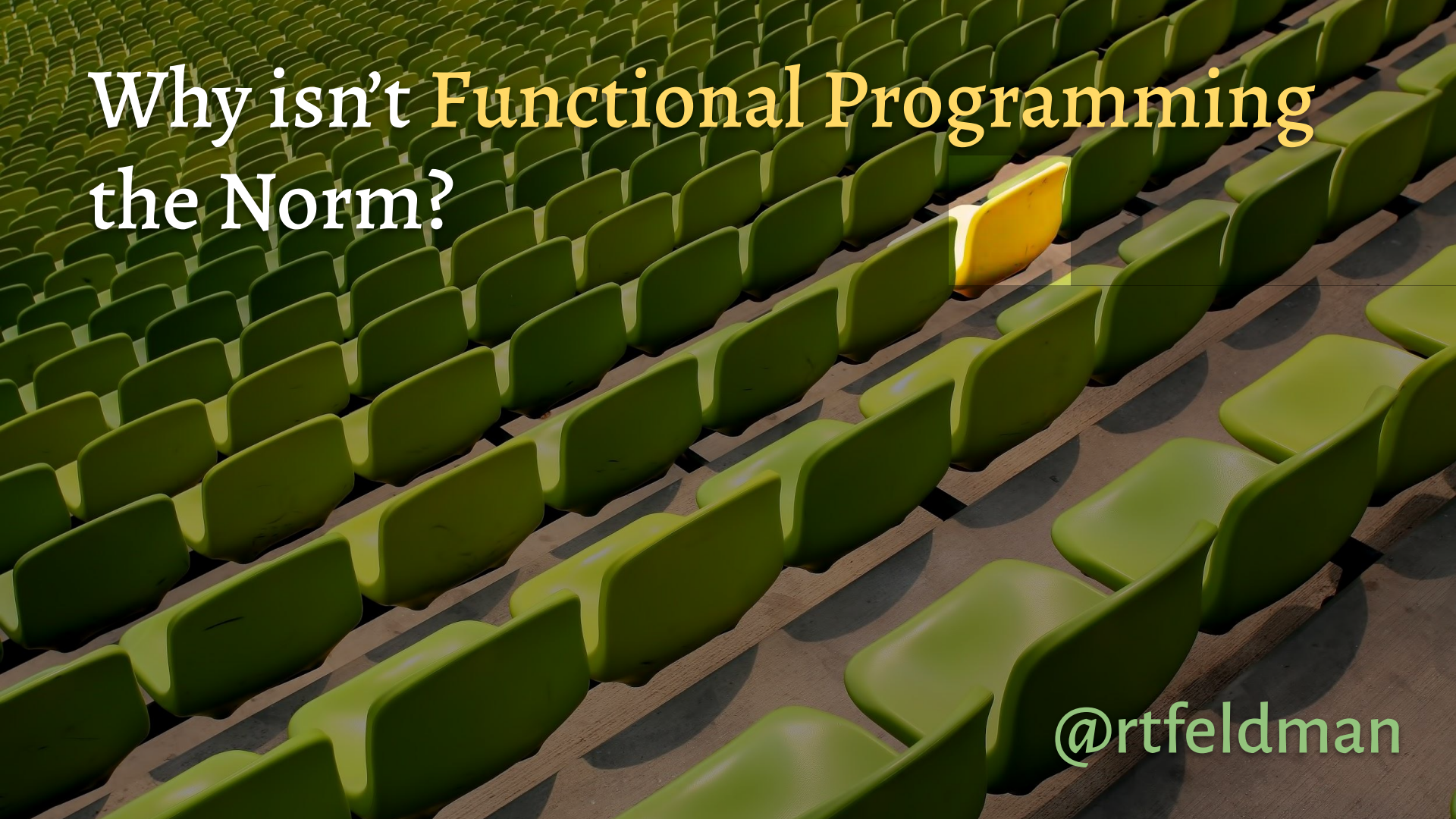

Hands-On Functional Programming with C++
An effective guide to writing accelerated functional code using C++17 and C++20
Packt
www.packt.com
Alexandru Bolboaca

OO style ——— hybrid ——→ FP style




"Wouldn't it be nice if my **language** had strong support for the **style** that's become the norm?"

(*) JavaScript includes CoffeeScript, TypeScript


Why isn't Functional Programming the Norm?
@rtfeldman

Part 1: **Language**

Part 2: **Paradigm**

Part 3: **Style**

# Why aren't FP **languages** the norm?

1. No sufficiently large "killer apps"
2. No exclusivity on large platforms
3. Can't be a quick upgrade if substantially different
4. No epic marketing budgets
5. Slow & steady growth takes decades

# Are OO languages the norm because of **uniquely OO features**?

1. **Information hiding** (encapsulation) is not a uniquely OO feature. **Modules** can do it too.
2. **Inheritance** is uniquely OO, but OO best practice encourages using **composition** instead.
3. Without inheritance, **objects and methods** are not significantly different from **structs and procedures**.

# Why isn't FP **style** the norm?

@rtfeldman