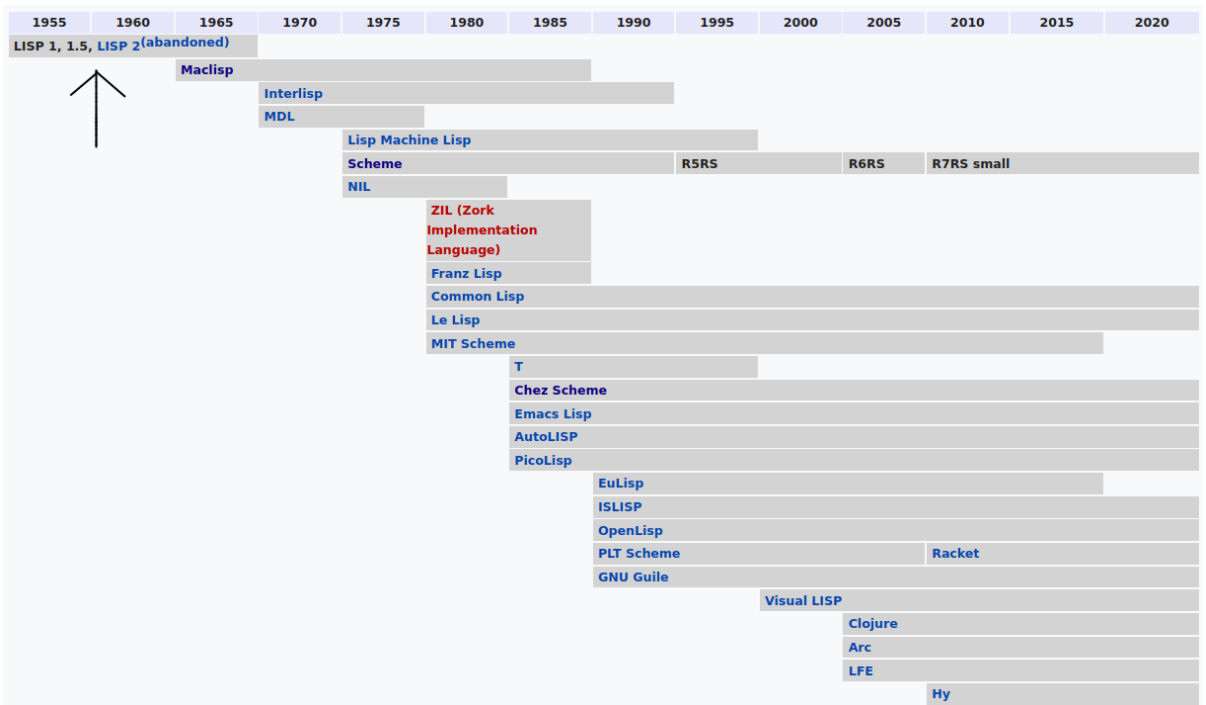
A TALE OF LISP

FROM LISP TO CLOJURE (AND BACK)

LONDON CLOJURIANS MEETUP - JUNE 7TH 2022

PART I: EARLY LISP 1955-1960



ONCE UPON A TIME, IN 1950

- John McCarthy (1927-2011)
- Ph.D. mathematics from Princeton 1951
- Then assistant professor at Dartmouth, 1955
- Soon after researcher at MIT, 1956

WHAT IS INTELLIGENCE?

- Hot topic in the ’50
- Shannon’s first chess program 1949
- First automated reasoning app Logic Theorist 1955
- Alan Turing’s Computing Machinery and Intelligence 1950

“ARTIFICIAL INTELLIGENCE” STUDY PROPOSAL

A Proposal for the

DARTMOUTH SUMMER RESEARCH PROJECT ON ARTIFICIAL INTELLIGENCE

June 17 - Aug. 16

We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer.

FEATURES FOR THE NEW LANGUAGE

- Abstracted away from hardware constraints
- Not much larger than the sentence it describes
- Allowing representable logical rules
- Enabling symbolic computations (not just numbers)

REALITY CHECK: PROGRAMMING IN 1958

- Fortran: symbolic, arithmetic
- IPL: list based, low level
- F-LPL: Fortan-compiled for list processing
- Algol: McCarthy tried to influence unsuccessfully

All too restrictive, in one way or another

THE PERFECT STORM

- September 1958 funding was granted
- Creation of a new MIT AI Lab with the following
  - 2 basement rooms
  - 2 programmers, 6 students
  - 1 secretary, 1 typewriter
  - Some limited CPU time

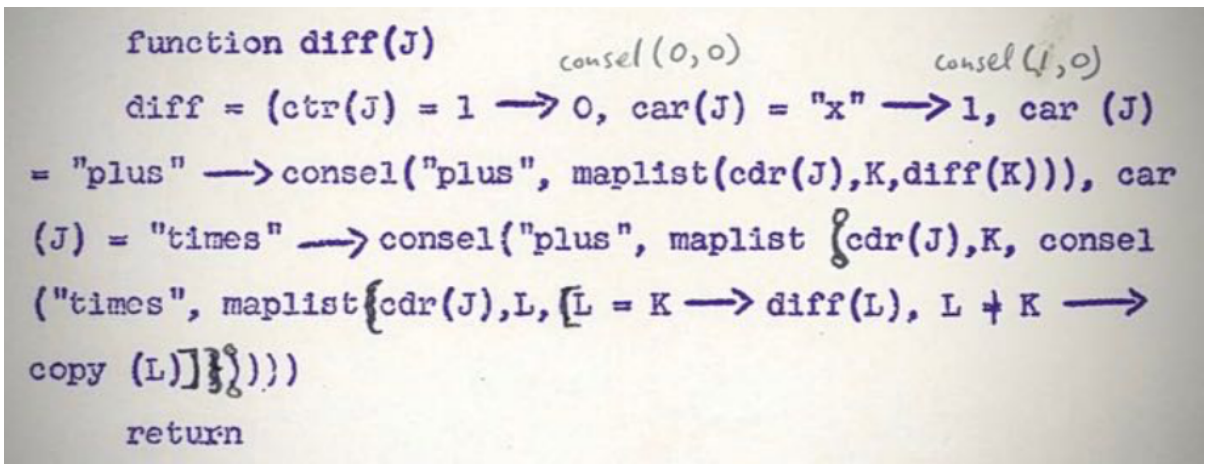
REALITY CHECK: IBM 704

CAR, CDR AND FRIENDS

Part	Name	Bits	Fn
w	whole-word	0-35	cwr
p	prefix	0-1-2	cpr
i	indicator	1-2	cir
s	sign-bit	0	csr
d	decrement	3-17	cdr
t	tag	18-20	ctr
a	address	21-35	car

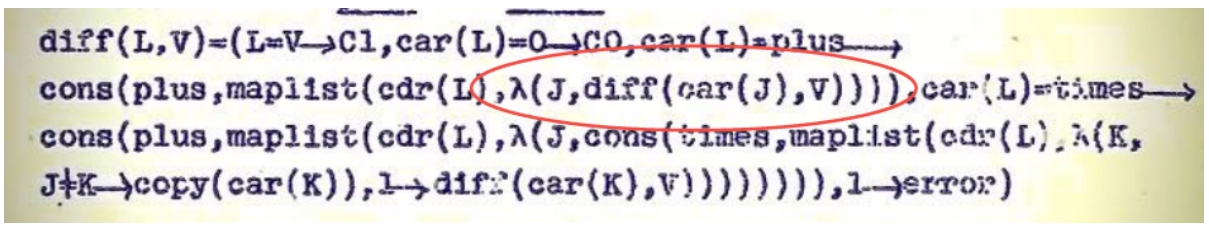
C (content of the) **R** (egistry)

PROGRESSIVE REFINEMENTS



- APL-ish single liner
- Algol like-ish
- “K”: a dummy var to iterate memory addresses

LAMBDA TO THE RESCUE



- Inspiration coming from Church's Lambda Calculus
- **λ** now allow `maplist` to declare the meaning of "J"

Still not very lispy!

TURNING POINT: S-EXPRESSIONS

- Challenge: demonstrate Turing completeness
- Proof: feed Lisp to another Lisp
- Alternative syntax required for easier “punching”
- Extend `apply` to receive Lisp forms as lists
- `eval` would need to interpret 6-7 special forms

...AND THEN STEVE RUSSELL CAME AROUND

- *Steve:* "Why don't we write this apply-eval down to assembly?"
- *McCarthy:* "ho, ho, you're confusing theory with practice… this `eval` is intended for reading not for computing."

APPLY <=> EVAL

LISP 1 WAS BORN

Tim Hart

LISP I

PROGRAMMER'S MANUAL

March 1, 1960

Artificial Intelligence Group

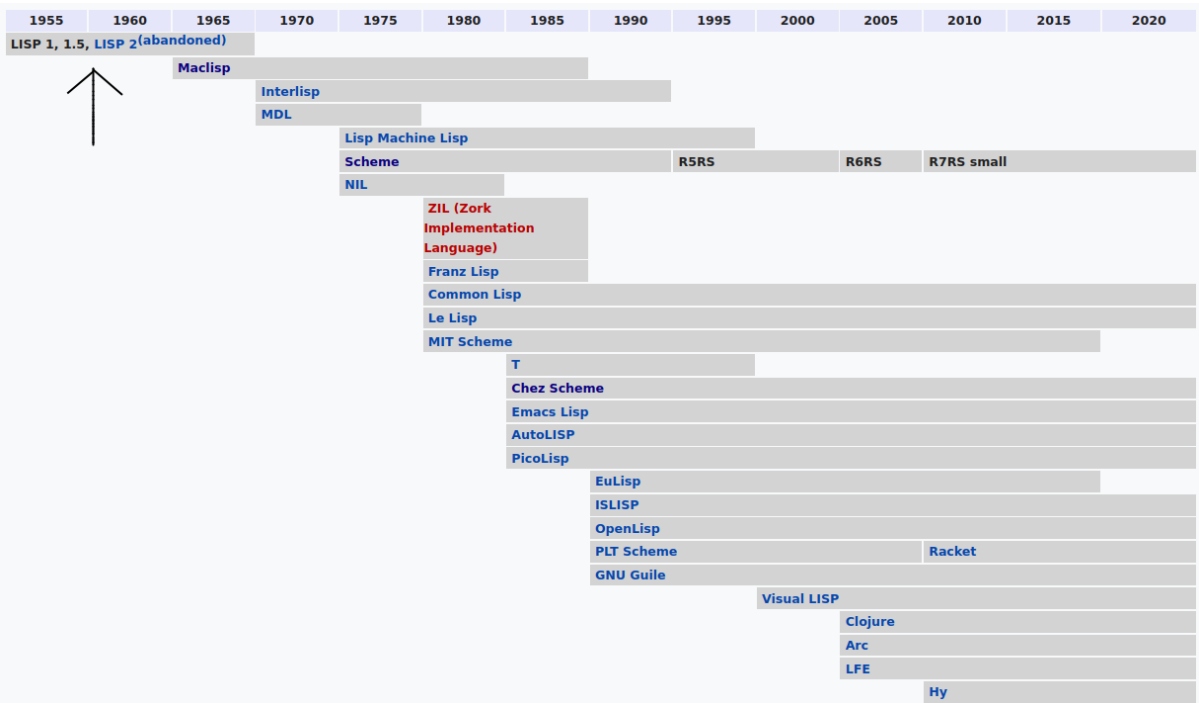
J. McCarthy
R. Brayton
D. Edwards
P. Fox
L. Hodes
D. Luckham
K. Maling
D. Park
S. Russell

COMPUTATION CENTER
and
RESEARCH LABORATORY OF ELECTRONICS
Massachusetts Institute of Technology
Cambridge, Massachusetts

LESSON LEARNED

- Innovative, ground-breaking goals
- Baby steps, small increments, refined versions
- If the alternative doesn’t work, make your own
- Constrained resources, more time to think
- Team interplay

=> FROM LISP TO COMMON LISP 1960-1990



AFTER LISP 1.0

- 1961, earliest surviving compiler by Hart-Levin
- 1962, Lisp 1.5 came out with minor improvements
- 1964, PDP-1 port by Peter Deutch
- Essentially any AI lab with a Lisp wizard down the hall
- We'll go fast and touch on a few implementations

MACLISP 1966-1990

- Targeting PDP 6-10, a production grade Lisp
- Lisp implementation for Project MAC (time-sharing)
- Consolidation of implementation techniques
- Style: small assembly core, Lisp library on top

INTERLISP 1968-1990

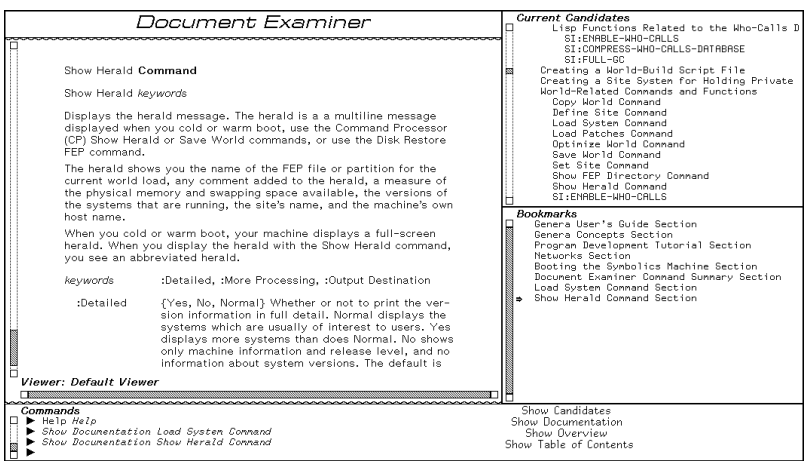
- Born as BBN Lisp, by Bobrow and al.
- Style: Emphasis on user interaction
- IDE, debuggers, auto-completion, spell checking
- Infinite login sessions: moving away from files
- Ended up at Xerox, inspiring Smalltalk
- Microcoded for the Alto by Deutch

LISP MACHINES

- First serious proposal by Deutch 1973
- Microcoded for Lisp efficiency
- 1974, the first MIT Lisp machine
- Large bitmapped display inspired by the Alto
- Introduced many industrial grade changes to Lisp

LISP MACHINES INNOVATIONS

- IDE-like environments, in the Interlisp spirit
- Specifically Symbolics Lisp:
  - `defmacro` and syntax quoting
  - `defstruct` record definition facility
  - `format`
  - Flavours OO



SCHEME

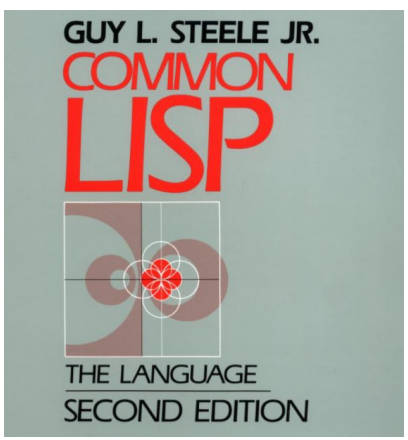
- 1975, initially to study aspects of the Actor Model
- Lexical closures were unusual and avoided in Lisp
- First-class continuations as support for actors
- TCO always possible thanks to continuations
- `?` or `!` for predicates and side effects
- Style: minimalism and centralized specifications

LISP FRAGMENTATION PROBLEM

- Many dialects with diverging philosophies
- Multiple Lisp machines from different vendors
- Multiple OO systems
- Multiple graphic libraries and GUIs

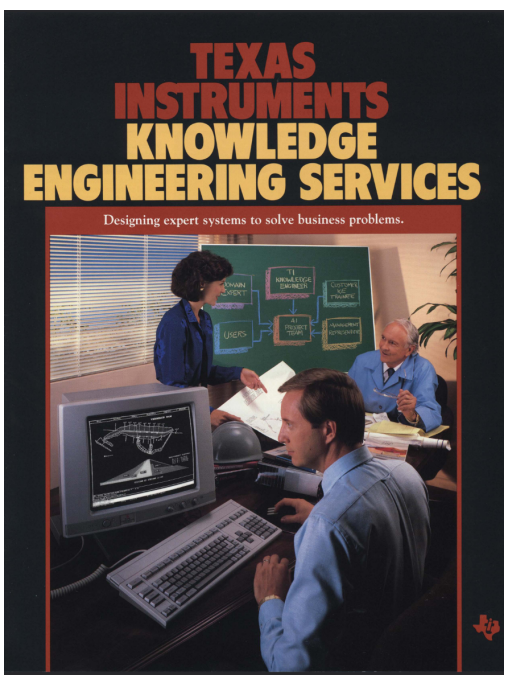
COMMON LISP

- An initiative started in the early ’80
- Goal: create a new Lisp with all the best features
- Initial discussions between MacLisp and Interlisp
- Priority given to Symbolics (largest money maker)
- Little or no interest in European Lisp strands



THE GOLDEN ERA 1984-1988

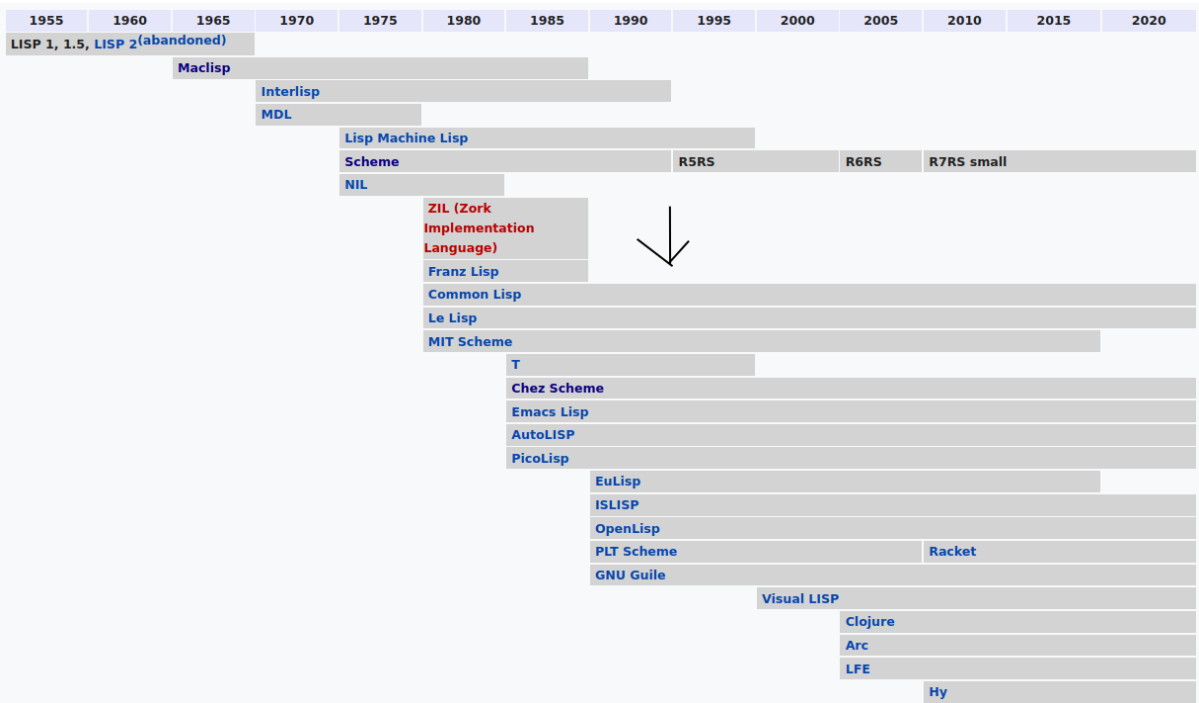
- AI was on the rise, Lisp was the language of AI
- Expert systems was the “killer app”
- The allure of AI attracted business and VCs
- Lisp was granted unprecedented popularity
- Lisp companies thrived thanks to quality and performance



LESSON LEARNED

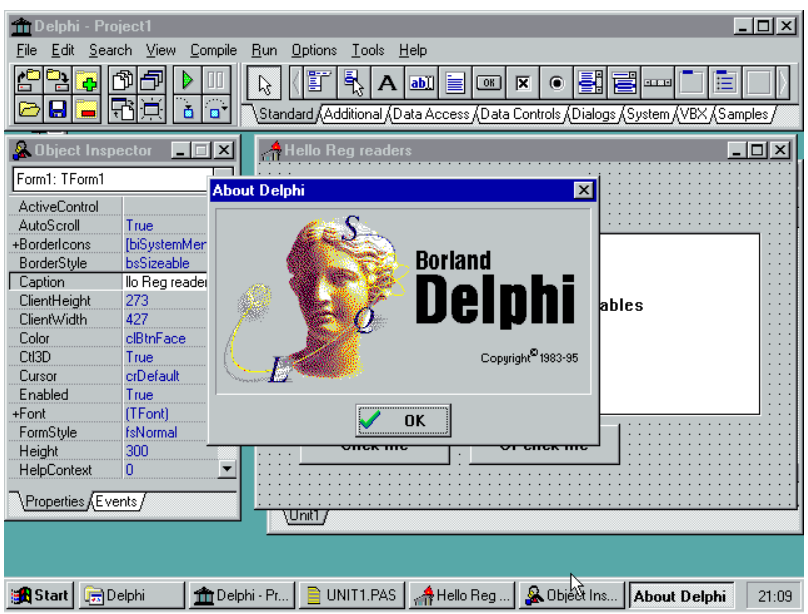
- Importance of theoretical foundations, Lisp is rooted in math
- Expressiveness power, the freedom from lower level details
- Easy DSLs, similar to the core language
- Interactive and incremental nature
- Pivotal to fundamental innovation

=> FROM CL TO CLOJURE 1990 - 2007



TIMES ARE CHANGING

- Advent of the web: shift from personal to distributed
- OO-based RAD IDE toolkits (like Delphi)
- Unix (with C) installed on 90% of servers
- Scripting languages: Python (1991), Java (applets) (1995), JavaScript (1996)



THE (LAST) AI WINTER, 1990+

- Was Lisp killed by its killer app?
- In typical bubble style, AI was over hyped
- Lisp machines power gradually eroded by Moore's law
- Negative impact of large failures like the Fifth Generation System
- Counterproductive self-blaming of Expert Systems as non-AI

LISP: SOME OTHER CITED PROBLEMS

- Imperative, OO or functional? All of them!
- Perhaps even too expressive: the Lisp curse
- King of the “throw-away design”: the Bipolar Lisp Programmer
- Pointy-haired bosses adverseness to risk (including dynamic types)
- “Many Lisps” even after going common

2022: IS LISP DEAD? NOT REALLY

- SBCL (CL)
- ABCL (CL)
- Allegro (CL)
- LispWorks (CL)
- Clozure CL (CL)
- Racket (not CL)
- Emacs Lisp (Not CL)
- Shen (Not CL)
- …

CLOJURE, THE BUILD UP 2001-2006

- 2001, Rich worked on a couple of production CL applications
- 2003, DotLisp (interpreted Lisp for the CLR)
- 2004, jFli (embedded JVM in CL)
- 2005, Foil (JVM to CL via IPC)
- 2006, Clojure first commit

STEADY GROWTH, 2007-2022

- 2007, first public release
- 2008, Rich invited at Lisp50 (OOPSLA 2008)
- 2009, JVM Language Summit Are we there yet? influential talk
- 2009, Programming Clojure first ed.
- 2010, First Clojure Conj
- 2012, Datomic announced, first EuroClojure
- 2013, Relevance becomes Cognitect
- 2014, and following new tools, releases, libraries, features etc.
- 2020, Clojure at HOPL IV, Nubank buys Cognitect
- 2022, 20+ books, conferences, 23k+ Slack users, many job posts

IS FUNCTIONAL FINALLY TAKING OVER?

- We should perhaps ask if things have fundamentally changed
- Is multi-core parallelism essential?
- Was programming liberated from the Von Neumann style?
- Has overall complexity of software changed?
- “Just” FP might not be enough to get out the tar pit

> *The complexity of software is an essential property, not an accidental one.*
>
> *Fred Brooks, No Silver Bullet*

WHAT MAKES A LANGUAGE POPULAR?

- ☐ Free, efficient, expressive, well designed
- ☐ Documented and beginner friendly
- ☐ Presence of a charismatic leader
- ☐ Scripting language for a popular OS or browser
- ☐ Distributed by a large vendor (marketing and money)
- ☐ Has a “killer app”
- ☐ Batteries included (Json, Web, etc.)
- ☐ Backward compatible and stable
- ☐ Easily contributed, hackable
- ☐ Pivotal to fundamental innovation

IS CLOJURE POPULAR?

- ☑ Free, efficient, expressive, well designed
- ☐ Documented and beginner friendly (maybe)
- ☑ Charismatic leader
- ☐ Scripting language for a popular OS or browser
- ☐ Distributed by a large vendor (marketing and money)
- ☐ Has a “killer app”
- ☐ Batteries included (Json, Web, etc.)
- ☑ Backward compatible and stable
- ☐ Easily contributed, hackable
- ☐ Pivotal to fundamental innovation

DOES IT NEED TO BE POPULAR?

> *I needed another choice more like Common Lisp or I wouldn't be able to continue as a professional software developer.*

Rich Hickey, A history of Clojure, 2020

> *Popularity further separates good languages from bad ones, because feedback from real live users always leads to improvements. So whether or not a language has to be good to be popular, I think a language has to be popular to be good. And it has to stay popular to stay good.*

Paul Graham, Hackers and Painters, 2010

CAN IT BE POPULAR?

- Popularity doesn't need to happen fast and can trigger anytime
- Lisp it's rooted in math: are we teaching it wrong?

> *Many young people perception of mathematics is as being "boring and irrelevant"*

maths inquiry report, 2004

> *"The popularity of mathematics diminishes during the time students are exposed to mathematics at school"*

The popularization of mathematics 1990

IS THERE HOPE?

- Of course!
- Human beings are short lived
- Knowledge persists through books, media, etc.
- Humans forget, distort, make mistakes
- As a consequence, history repeats
- Learn the past to shape a better future

WHAT CAN I DO?

- Tell the Lisp tale to friends and family
- Talk at conferences and meetups
- Write blogs, libraries, books
- Finally crack the “intelligence” problem
- Never stop learning!

~ fin ~