Lisp Speech Therapy

(Or: trying to explain some misconceptions about Lisp)

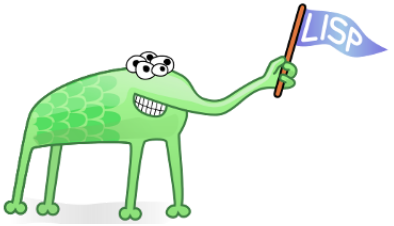
Bruno Jacquet

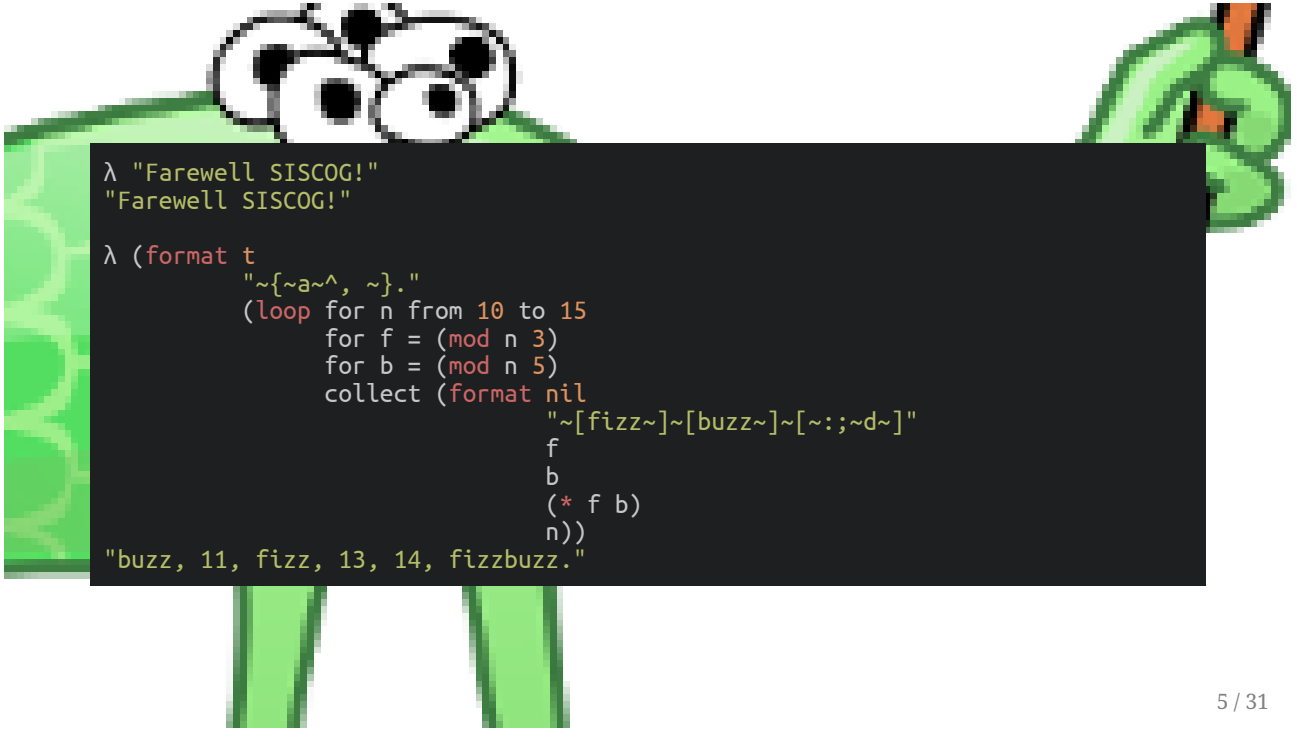
Bruno Jacquet

SISCOG

Bruno Jacquet

SISCOG





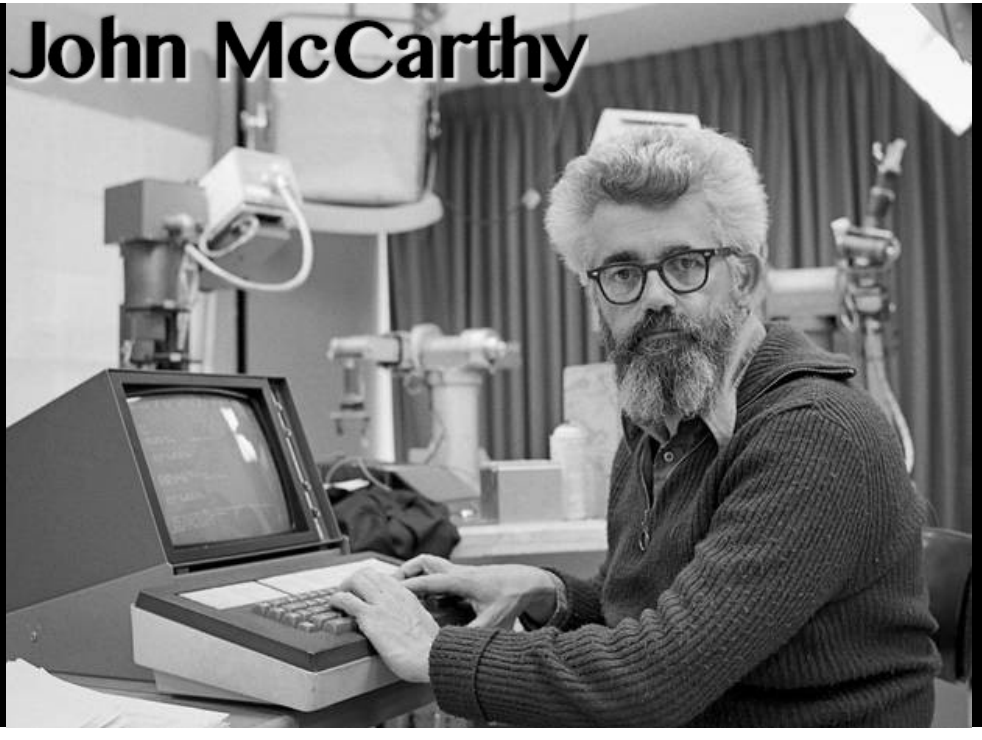
```
λ "Farewell SISCOG!"
"Farewell SISCOG!"

λ (format t
          "~{~a~^, ~}."
          (loop for n from 10 to 15
                for f = (mod n 3)
                for b = (mod n 5)
                collect (format nil
                                "~[fizz~]~[buzz~]~[~:;~d~]"
                                f
                                b
                                (* f b)
                                n))
"buzz, 11, fizz, 13, 14, fizzbuzz."
```

```
> "Hello, Runtime Revolution!"
"Hello, Runtime Revolution!"

> (10..15).map do |i|
    x = ''
    x += 'fizz' if i % 3 == 0
    x += 'buzz' if i % 5 == 0
    x.empty? ? i.to_s : x.to_s
  end.join(', ') + '.'
"buzz, 11, fizz, 13, 14, fizzbuzz."
```

Define Lisp


John McCarthy

Define Lisp in 2018

Define Lisp in 2018

Lisp is not a programming language

Define Lisp in 2018

Lisp is not a programming language

Lisp is a **family** of programming languages

Define Lisp in 2018

Lisp is not a programming language

Lisp is a **family** of programming languages

Dialects
Arc · AutoLISP · Clojure · Common Lisp · Emacs Lisp · EuLisp · Franz Lisp · Hy · Interlisp · ISLISP · LeLisp · LFE · Maclisp · MDL · newLISP · NIL · Picolisp · Portable Standard Lisp · Racket · RPL · Scheme · SKILL · Spice Lisp · T · Zetalisp

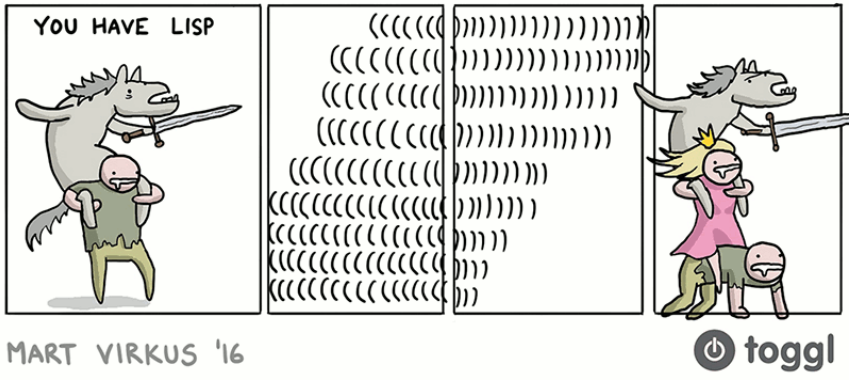
Define Lisp in 2018

Lisp is not a programming language

Lisp is a **family** of programming languages

Dialects	Major implementations
Arc · AutoLISP · Clojure · Common Lisp · Emacs Lisp · EuLisp · Franz Lisp · Hy · Interlisp · ISLISP · LeLisp · LFE · Maclisp · MDL · newLISP · NIL · Picolisp · Portable Standard Lisp · Racket · RPL · Scheme · SKILL · Spice Lisp · T · Zetalisp	Allegro CL, ABCL, CLISP, Clozure CL, CMUCL, ECL, GCL, LispWorks, Scieneer CL, SBCL, Symbolics Common Lisp

"...show me the code."



Parenthesis Everywhere

```
(defun birthday-paradox (probability people)
  (let ((new-probability (* (/ (- +year-size+ people)
                               +year-size+)
                            probability)))
    (if (< new-probability 0.5)
        (1+ people)
        (birthday-paradox new-probability (1+ people)))))
```

```
λ (list 1 "b" 'c #'+)
(1 "b" c +)
```

- Expression delimiters
- Represent data as lists

Delimiters Ruby

```
def permutation(l = [])
  for e in 1..6
    a, i = [], e-2
    (0..l.length-e+1).each do |g|
       if not (n = l[g..g+e-2]).uniq!
         a.concat(n[(a[0]? i : 0)..-1]).push(e).concat(n)
         i = e-2
       else
         i -= 1
       end
     end
     a.each { |n| print n }; puts "\n\n"
     l = a
  end
end
```

Prefix Notation

Prefix Notation


```
(if (< new-probability 0.5)
    (setq people (+ people 1))
    (birthday-paradox new-probability (1+ people))))
```

Prefix Notation

```
λ (/ (+ (- b)
        (sqrt (- (expt b 2)
                 (* 4 a c))))
     (* 2 a))
42
```

```
λ (+ 1 2 3 4)
10
```

Prefix Notation

```
λ (/ (+ (- b)
        (sqrt (- (expt b 2)
                 (* 4 a c))))
     (* 2 a))
42
```

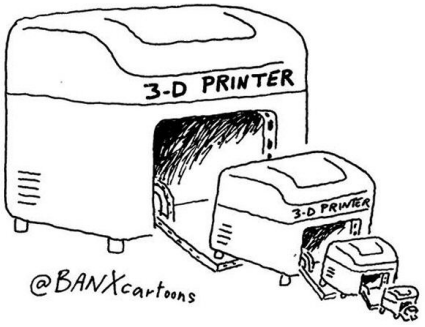
```
λ (+ 1 2 3 4)
10
```

Infix Notation

```
> (-b + Math.sqrt(b**2 - 4 * a * c)) /
  (2 * a)
42
```

```
> 1 + 2 + 3 + 4
=> 10
```

Recursion



Recursion

```
(defun factorial (n)
  (factorial-aux n 1))

(defun factorial-aux (n a)
  (if (= n 1)
      a
      (factorial-aux (- n 1) (* n a))))
```

Recursion

```
def factorial(n)
  factorial_aux(n, 1)
end

def factorial-aux(n, a)
  n == 1 ? a : factorial_aux(n - 1, n * a)
end
```

Recursion

```
def factorial(n)
  factorial_aux(n, 1)
end

def factorial-aux(n, a)
  n == 1 ? a : factorial_aux(n - 1, n * a)
end
```

```
def factorial(n)
  return 1 if n == 0
  (1..n).inject(:*)
end
```

Functional Programming

- Higher-order functions
- Purity
- Recursion

Functional Programming

Higher-order functions

Purity

Recursion

Lisp has these features and so does Ruby

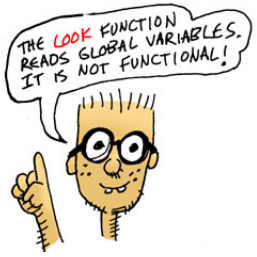
Yet neither are considered *pure* functional

Functional Programming

```
(defun look ()
  (append (describe-location *location* *map*)
          (describe-paths *location* *map*)
          (describe-floor *location* *objects* *object-locations*)))
```

Functional Programming

```
(defun look ()
  (append (describe-location *location* *map*)
          (describe-paths *location* *map*)
          (describe-floor *location* *objects* *object-locations*)))
```



Topics Covered

History

Syntax

Recursion

Functional Programming

Other Topics

Macros

Is sloooooow

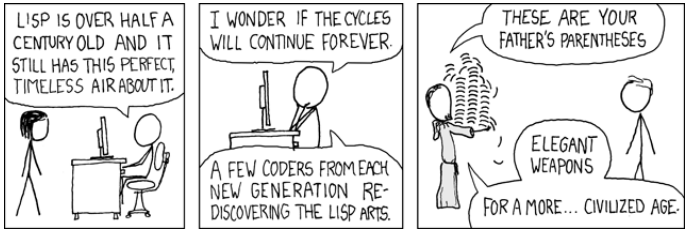
It's only for AI

No one else uses it

There's no GUI in Lisp

Doesn't have good IDE

Thank you!


LISP IS OVER HALF A CENTURY OLD AND IT STILL HAS THIS PERFECT, TIMELESS AIR ABOUT IT.
I WONDER IF THE CYCLES WILL CONTINUE FOREVER.
A FEW CODERS FROM EACH NEW GENERATION RE-DISCOVERING THE LISP ARTS.
THESE ARE YOUR FATHER'S PARENTHESES
ELEGANT WEAPONS
FOR A MORE... CIVILIZED AGE.

Notes & References

- *Casting Spels in Lisp*, http://www.lisperati.com/casting.html
- *Features of Common Lisp*, Abhishek Reddy, http://random-state.net/features-of-common-lisp.html
- *HaskellWiki*, https://wiki.haskell.org/Functional_programming
- *Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I*, John McCarthy, 1960
- *Successful Lisp*, David B. Lamkins, 2004