

Clojure programming language & it's ecosystem

Marius Rabenarivo



DEVELOPERS
CONFERENCE
2021

What is Clojure?

- LISP



Figure 1: Lisp Cycles, <https://xkcd.com/297/>

- Rich Hickey



Figure 2: Rich Hickey at the first Clojure Conj in 2010, <https://creativecommons.org/licenses/by-sa/2.0/>

What is LISP?

The image displays three lambda calculus definitions for the Y combinator, fib, and fac, followed by their application to mapcar and list processing. The definitions are written in a stylized, medieval manuscript font.

Definition 1 (Y combinator):

$$\text{Defun } \mathcal{Y} (f) ((\lambda (g) (\text{funcall } g \ g)) (\lambda (x) (\text{funcall } f (\lambda (\text{rest } a) (\text{apply } (\text{funcall } x \ x) \ a))))))$$

Definition 2 (fib):

$$\text{Defun fib } (n) (\text{funcall } (\mathcal{Y} (\lambda (f) (\lambda (n \ a \ b) (\text{if } (< \ n \ 1) \ a (\text{funcall } f \ (1- \ n) \ b \ (+ \ a \ b)))))) \ n \ 0 \ 1)$$

Definition 3 (fac):

$$\text{Defun fac } (n) (\text{funcall } (\mathcal{Y} (\lambda (f) (\lambda (n) (\text{if } (\text{zerop } \ n) \ 1 (\times \ n (\text{funcall } f \ (1- \ n)))))) \ n)$$

Application of Y combinator to mapcar:

$$\text{mapcar } \#'\text{fib } '(i \ ij \ iij \ iv \ v \ vi \ vij \ viij \ i\x)$$

Application of Y combinator to fac:

$$\text{mapcar } \#'\text{fac } '(i \ ij \ iij \ iv \ v \ vi \ vij \ viij \ i\x)$$

Figure 3: Y Combinator Codex by emacsomancer.

What is LISP?

- LISP Processing language
- 2nd oldest High-Level Language after FORTRAN
- 1958 - John McCarthy

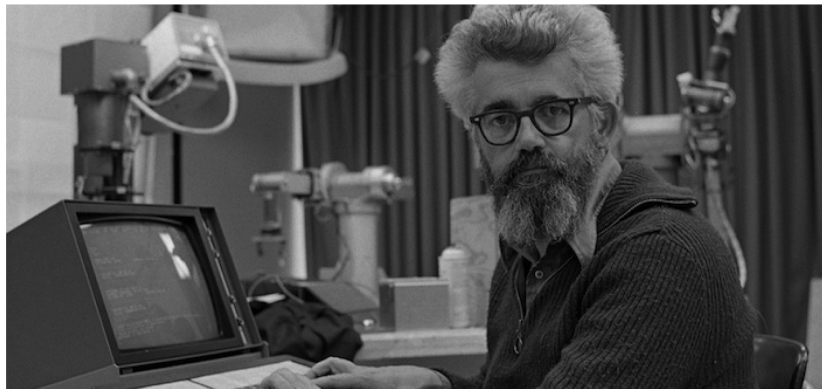


Figure 4: John McCarthy at work in his artificial intelligence laboratory at Stanford

Why Clojure?

- Designed with simplicity
- Functional Programming
- Data Oriented



Figure 5: Code? Data? Code? Data?

Why Clojure?

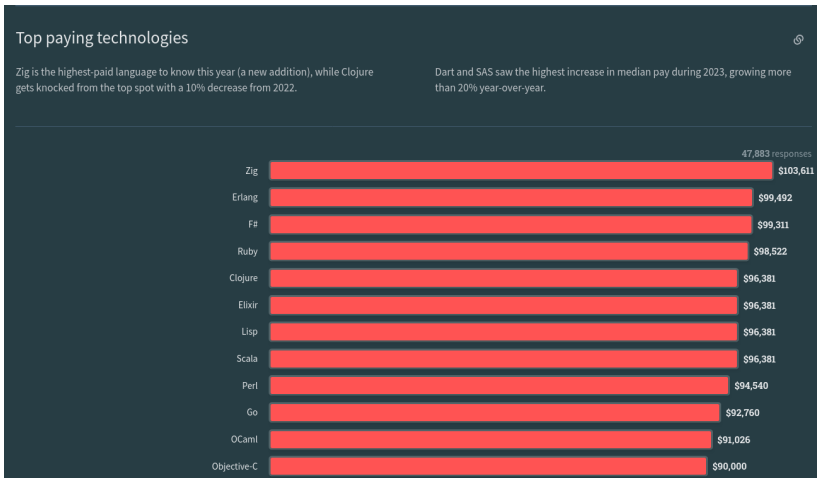


Figure 6: 2023 Developer Survey, Top paying technologies

Source: <https://survey.stackoverflow.co/2023/#section-top>

Why Clojure?

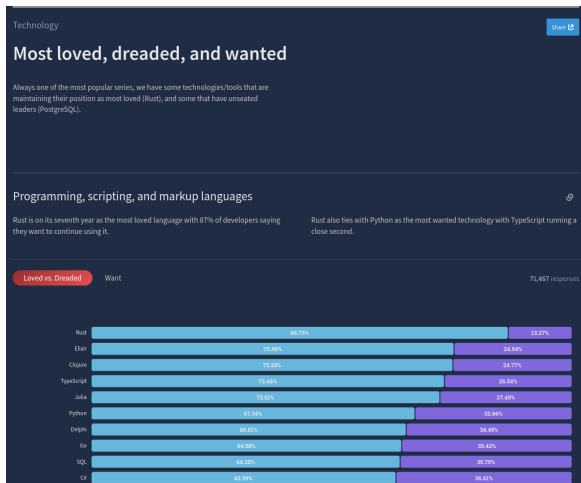


Figure 7: Stack Overflow Developer Survey 2022, Most loved, dreaded, and wanted

Functional Programming

- Immutability
- Referential transparency
- Easy to think about systems
- Parallel and concurrent programming