Clojure
Programming language

Lucas Castro (PED) MC346 2021.2
lucas.castro@ic.unicamp.br

PED CLASSES

01 RUST LANGUAGE
09/11

02 GOLANG
11/11

03 CLOJURE
16/11

04 SCALA
18/11

CLOJURE HIGHLIGHTS

JVM

CLOJURE WAY

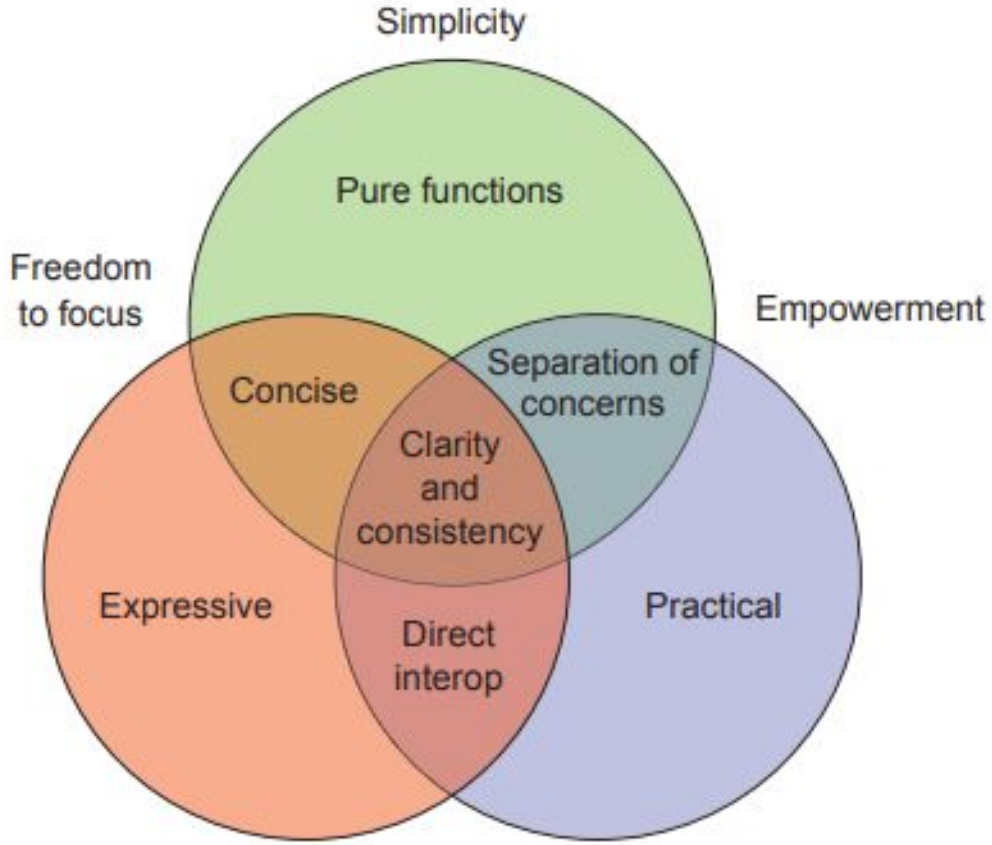
✓ FUNCTIONAL
✗ OBJ.-ORIENTED

CLOJURE-PY
CLOJERL

CORE FUNCTIONS

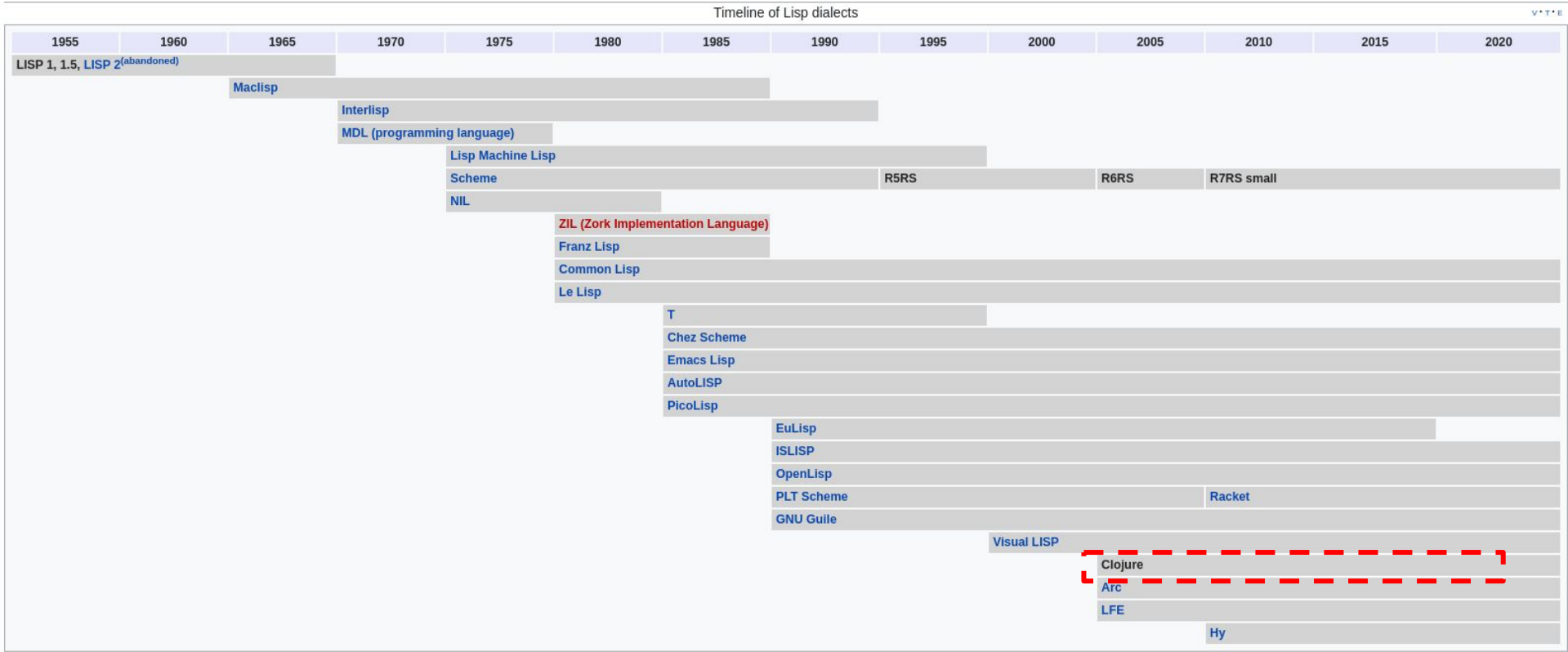
DIALETO LISP

CLOJURE WAY


Simplicity
Pure functions
Freedom
to focus
Empowerment
Concise
Separation of
concerns
Clarity
and
consistency
Expressive
Practical
Direct
interop

LISP DIALECTS TIMELINE

- Rich Hickey 2007
- ClojureScript, CljPerl, clojure-py



BEAUTY CODE

```
362 (defun prime-factors (figure &optional siblings)
363   (when (> figure 1)
364     (if (evenp figure)
365         (prime-factors (/ figure 2) (cons 2 siblings))
366         (loop with max-factor = (isqrt figure)
367           for factor? = 3 then (1+ factor?)
368           for (quotient remainder) = (multiple-value-list (truncate figure factor?))
369           do (cond ((> factor? max-factor)
370                     (return (cons figure siblings)))
371                    ((zerop remainder)
372                     (return (prime-factors quotient
373                                            (cons factor? siblings))))))))))
374
```

"Joy awaits the Lisp neophyte"

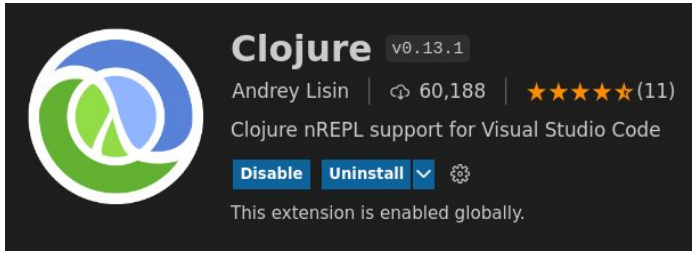
The Joy of Clojure book (2014)
Michael Fogus and Chris Houser

Clojure na sua máquina

- Clojure compilador (REPL)
  - curl -O https://download.clojure.org/install/linux-install-1.10.2.774.sh
  - ./clojureInstall.sh

- Leininger Project tool
  - curl https://raw.githubusercontent.com/technomancy/leiningen/stable/bin/lein > lein
  - sudo mv lein /usr/local/bin/lein
  - sudo chmod a+x /usr/local/bin/lein
  - lein version
  - lein new app helloworld

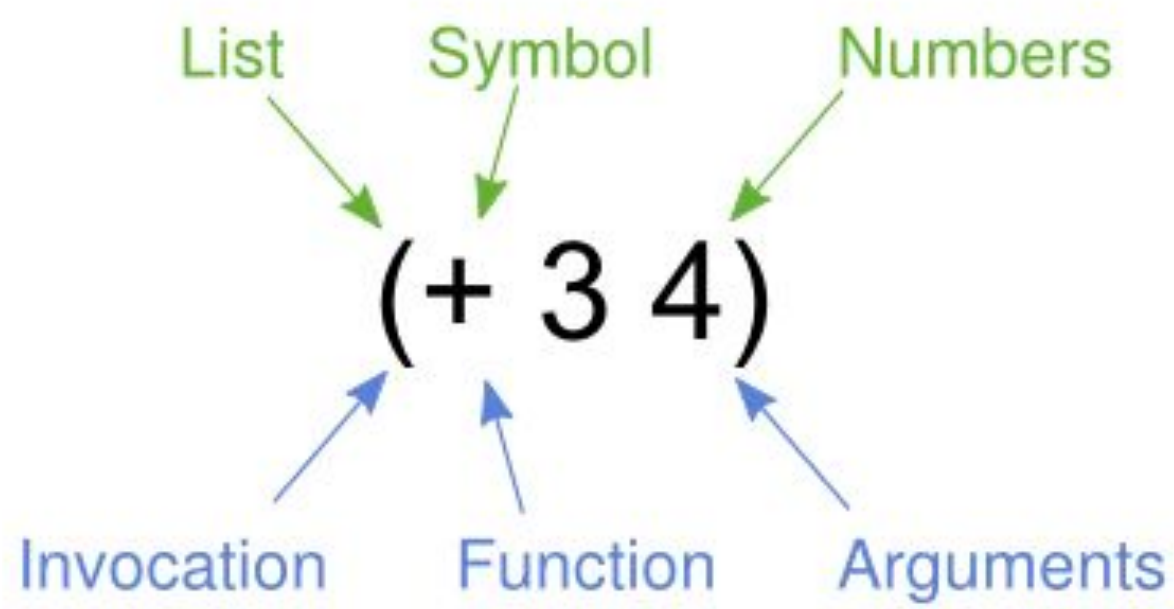
- Extensão VSCode
  - Clojure extension - Andrey Lisin



CLOJURE FEATURES

- Funcional e não orientada a objetos
  - Imutabilidade;
  - *“ The classical object-oriented model allows unrestrained mutation of object properties without a willingness to preserve historical states” - Joy of Clojure*
  - Orientada a valores;
  - Objetos vs valores;
- REPL
- Interoperabilidade com o Java
- Desenvolvimento web (ringo lib)
- Código como um dado

(CLOJURE (FUNÇÕES))



Clojure Sintax

(Clojure (funções)) - REPL

- (função parametros)
  - Notação polonesa

- (+ 3 4)
- (- 20 10)
- (* 30 10)
- (/ 50 10)
- (println “Hello World!)
- (println “A soma de 3 + 2 é = “(+ 3 2))
- (if...)
- (def)
- (defn)

Clojure Sintax

Clojure tipo de dados e collections (seq)

- Int,float,char, string...
- Listas
  - *LinkedList*
  - *‘(1 2 3 4 (a b c) 5)*
- Vectors
  - *ArrayList*
  - [1 2 3 true :a]
- Mapas
  - {1 “a”, “b” 2)
- Sets
  - #{1,2,3,4,5,6}
- Funções sobre coleções
  - get
  - count
  - conj/disj
  - def

Clojure - coleções (hash)

- Mapas
  - {1 “a”, “b” 2)
- Sets
  - #{1,2,3,4,5,6}
- Record
  - (defrecord Name [fields])

- Funções sobre coleções (hash)
  - into
  - assoc
  - dissoc
  - keys/vals
  - merge

	Vector	**List**	**Map**	**Set**
Combine	concat	concat	merge	union
Prepend	(Don't)	conj	N/A	N/A
Append	conj	(Don't)	N/A	N/A
Add or replace	N/A	N/A	conj	conj

Source

(CLOJURE (FUNÇÕES))

- Criando funções
  - # parametros
    - (defn <functionName> [name age])
  - N parametros (variadic)
    - (defn <functionName> [first & parameters]

```
(defn greeting [name]
  (println "Hello, " name))
```

- Funções com n-aridades

```
(defn messenger
  ([]     (messenger "No message here"))
  ([name]  (greeting name))
  ([name & msg] (println name " says: " msg)))
```

- Funções anônimas
  - #(* 10 %)

```
(map #(* 10 %) (range 1 99 10))    (10 110 210 310 410 510 610 710 810 910)
```

(CLOJURE (FUNÇÕES)) CONT.

- Função apply
  - (max 1 2 3) =>?
  - (max [1 2 3]) =>?
  - (+ 1 2) =>
  - (+ [1 2]) =>
  - (str(reverse “mc346”))
  - apply...
- Função partial
  - Espera por um parâmetro

```
(def add10 (partial + 10))
(add10 20)
```

```
(def add-domain (partial (str "@mc346.com")))
(str "lucas" add-domain)
```

- Mais clojure.core functions!

(CLOJURE (FUNÇÕES)) IF.

- (defn <functionName> [parameter]
  (if (cond)
  (faça algo)
  (senão)
)
)

```
(defn testGreaterAge [age]
  (if (>= age 18)
    (println "greater")
    (println "not")))
```

THREAD FIRST / LAST

- Thread First

```
(defn tfirst []
    (->
    {1 "ic", 2 "unicamp", 3 "mc346"}
    (assoc 4 "clojure")
    (dissoc 1))
  )

(tfirst)
```

THREAD FIRST / LAST

- Thread Last

```
(defn tLast []
  (->>
    (range 1 20)
    (map inc)
    (filter odd?)
    (into []))

  )

(tLast)
```

JAVA INTEROP.

- Podemos executar código puro Java em Clojure

```
(.format (SimpleDateFormat. "MM/dd/yyyy") (Date.))
(.toUpperCase "ic-unicamp")

(defn workingJava[]
  (doto (java.util.HashMap.)
  (.put "banana" 1)
  (.put "laranja" 2)
  (.put "melancia" 3))
)

(workingJava)
```

Clojure is a Java better than Java

Aprenda mais & work in clojure

- Clojure Learn Guide
- Clojure Book - For the brave and true
- 4Clojure
- Exercism.io
- Ebook Clojure - Casa do Código
- Vagas Clojure