CLOJURE DATA TYPES

All collection types In Clojure Are Immutable BY Default

NUMBERS
→ BIG DECIMAL
→ BIG INT
  ↳ PRECISION NUMBERS (42 N)
↳ LONG → 64 BITS (SIGNED)
↳ DOUBLE → 64 BITS (WITH FLOATING POINTS)
→ RATIO
  ↳ EXACT FRACTIONS (1/3, 22/7)

TEXT
→ STRING
→ CHARACTER

COLLECTION TYPES

SPECIALIZED COLLECTIONS
→ RECORD → USER-DEFINED TYPES WITH NAMED FIELDS
  ↳ DEFRECORD
↳ QUEUE → FIFO
→ TYPE
  ↳ LOWER LEVEL USER-DEFINED
    ↳ DEFTYPE

SEQUENTIAL
→ LIST → OPTIMIZED FOR PREPENDING
↳ VECTOR → INDEXED ARRAYS, OPTIMIZED FOR RANDOM ACCESS
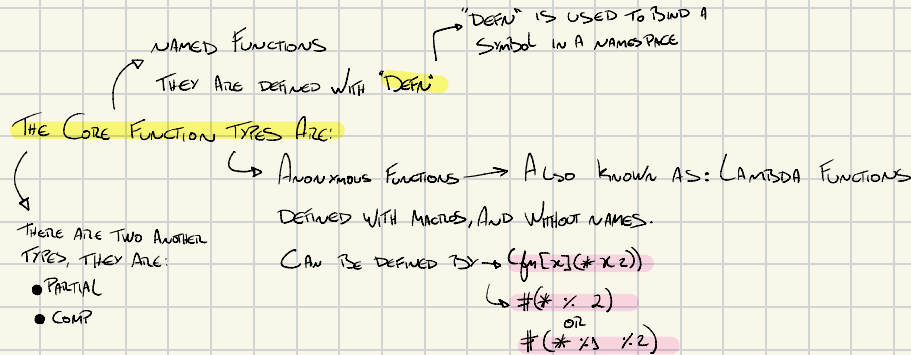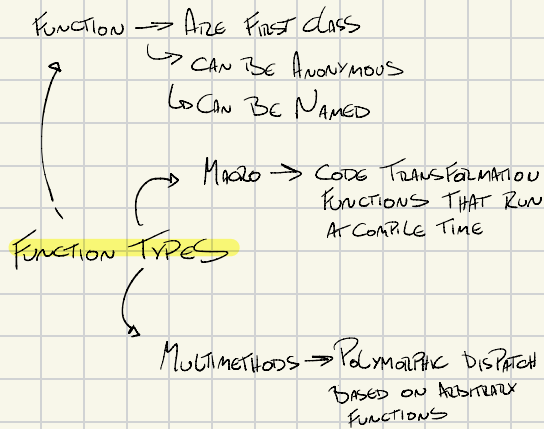
ASSOCIATIVE
→ MAP → KEY VALUE PAIRS
  ↳ {:NAME "JHON"}
↳ SET → UNIQUE ELEMENTS COLLECTION
  ↳ #{ 1 2 3}

FOR STATE MANAGEMENT
AGENT → ASYNCHRONOUS STATE CHANGES
REF → SYNCHRONOUS AND COORDINATED STATE CHANGES
ATOM → SYNCHRONOUS AND UNCOORDINATED STATE CHANGES
VAR ↳ THREAD-LOCAL STATE AND DYNAMIC BINDING

Made with Goodnotes

FUNCTION → ARE FIRST CLASS
↳ CAN BE ANONYMOUS
↳ CAN BE NAMED

MACRO → CODE TRANSFORMATION
FUNCTIONS THAT RUN
AT COMPILE TIME

==FUNCTION TYPES==

MULTIMETHODS → POLYMORPHIC DISPATCH
BASED ON ARBITRARY
FUNCTIONS

NAMED FUNCTIONS
THEY ARE DEFINED WITH `DEFN`

"DEFN" IS USED TO BIND A
SYMBOL IN A NAMESPACE

==THE CORE FUNCTION TYPES ARE:==

↳ ANONYMOUS FUNCTIONS → ALSO KNOWN AS: LAMBDA FUNCTIONS
DEFINED WITH MACROS, AND WITHOUT NAMES.
CAN BE DEFINED BY → (fn [x] (* x 2))
↳ #(* % 2)
or
#(* %1 %2)

THERE ARE TWO ANOTHER
TYPES, THEY ARE:
• PARTIAL
• COMP

THERE ARE SPECIFIC FUNCTIONS THAT ACCEPT
FUNCTIONS AS PARAMETERS:
• MAP
• FILTER
• REDUCE

IF YOU RESPECT THE LEXICAL SCOPE,
IS POSSIBLE TO CREATE FUNCTIONS
THAT RETURN OTHER FUNCTIONS.
↳ THIS IS DIFFERENT FROM
CURRIED FUNCTIONS. THOSE, ARE
FUNCTIONS THAT RETURN ANOTHER
FUNCTIONS TO ACHIEVE A PARTIAL
APPLICATION / COMPOSITION

==MULTIMETHODS==

USED TO CREATE BEHAVIOR BASED ON
==DEFMULTI== AND ==DEFMETHOD== FUNCTIONS
• DISPATCH FUNCTION DETERMINES
WHICH METHOD TO CALL
• CAN DISPATCH ON TYPE, VALUE,
OR ANY COMPUTED RESULT

==PROTOCOLS==

INTERFACE DEFINITION USING ==DEFPROTOCOL==
WITH IMPLEMENTATIONS VIA ==EXTEND PROTOCOL==
↳ MORE PERFORMANT THAN MULTIMETHODS
FOR TYPE-BASED DISPATCH
↳ CAN BE EXTENDED TO EXISTING TYPES