Functional Vs OOP Mental Model

Amarjeet Yadav @amarjeet000

Just an opinion

It’s not about

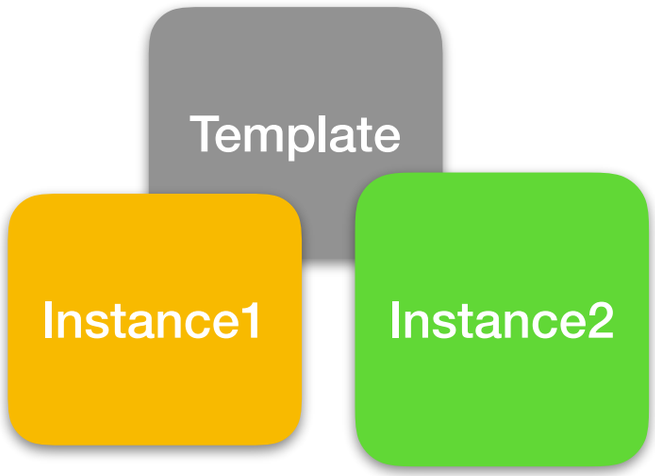
Pure Functions

or

Higher Order Functions

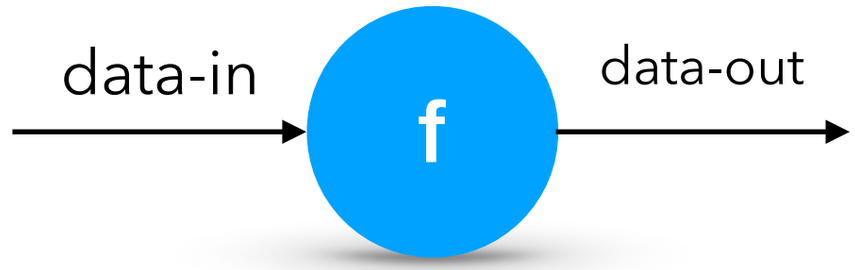
Object Oriented view of the World

- Class
- Object



Functional view of the world

- Immutable Data
- Function



State & Behaviour => Application (System)

“Application as a unit of programming”

“Data & Function as units of programming”

Programming is about building systems

- fast **enough**

- simple to collaborate on
- simple to extend/evolve/maintain
- simple to test correctly

Simple to **reason** about

OOP

"Application as a unit of programming"

Many systems in one

FP

"Data & Function as units of programming"

One system

Clojure

“Clojure is predominantly a functional programming language, and features a rich set of immutable, persistent data structures.

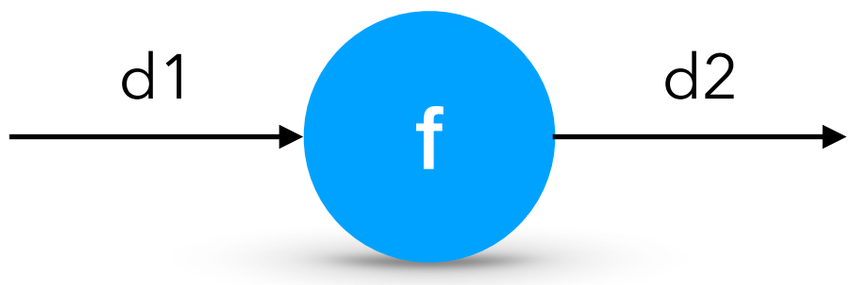
When mutable state is needed, Clojure offers a software transactional memory system and reactive Agent system that ensure clean, correct, multithreaded designs.”

Source: clojure.org

Clojure

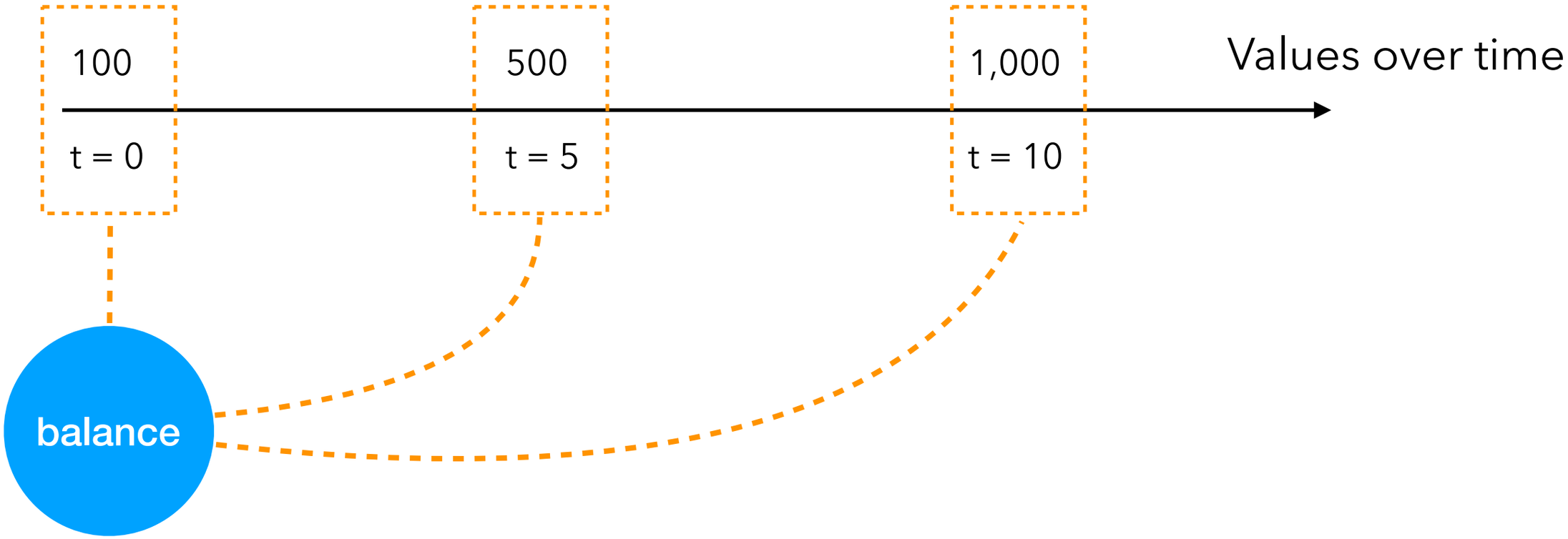
Immutable Data
Expressed in the forms of different data structures

Function
Operates on data and produce new data



Safe and Clean mechanism to deal with the state (**mutation**)

Immutability & State



Identity

At t = 0
my-bal: (balance)
my-dream-bal: (my-bal + 200)

=> my-bal => 100
=> my-dream-bal => 300

At t = 10
my-curr-bal: (balance)
my-new-bal: (my-bal + 300)

=> my-curr-bal => 1,000
=> my-new-bal => 400
=> **my-dream-bal** => 300

my-bal, my-dream-bal => **Immutable data**
(balance), (my-bal + 100) => **Function call** with latest values attached to the arguments

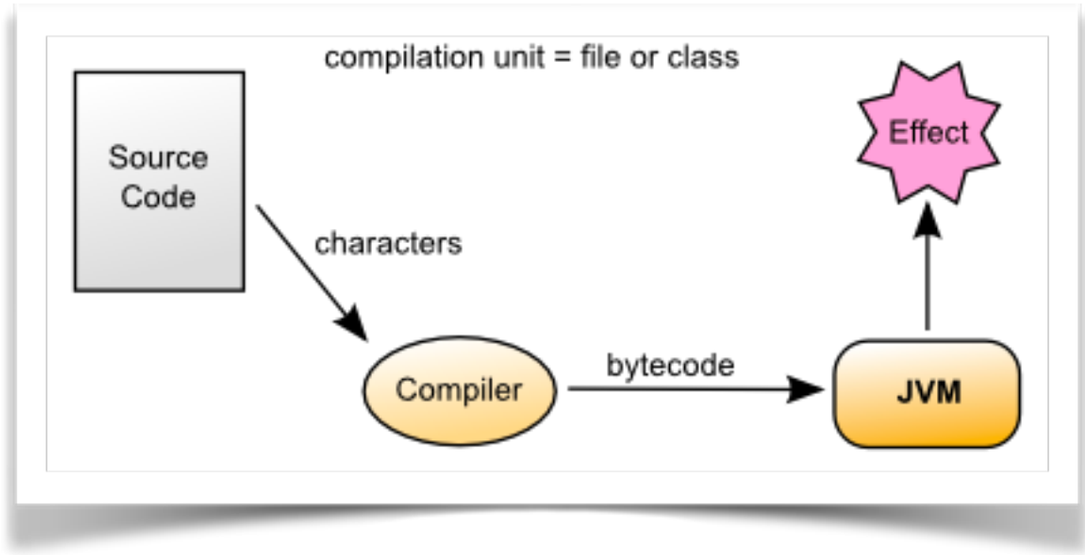
Concurrency & Multi-threading

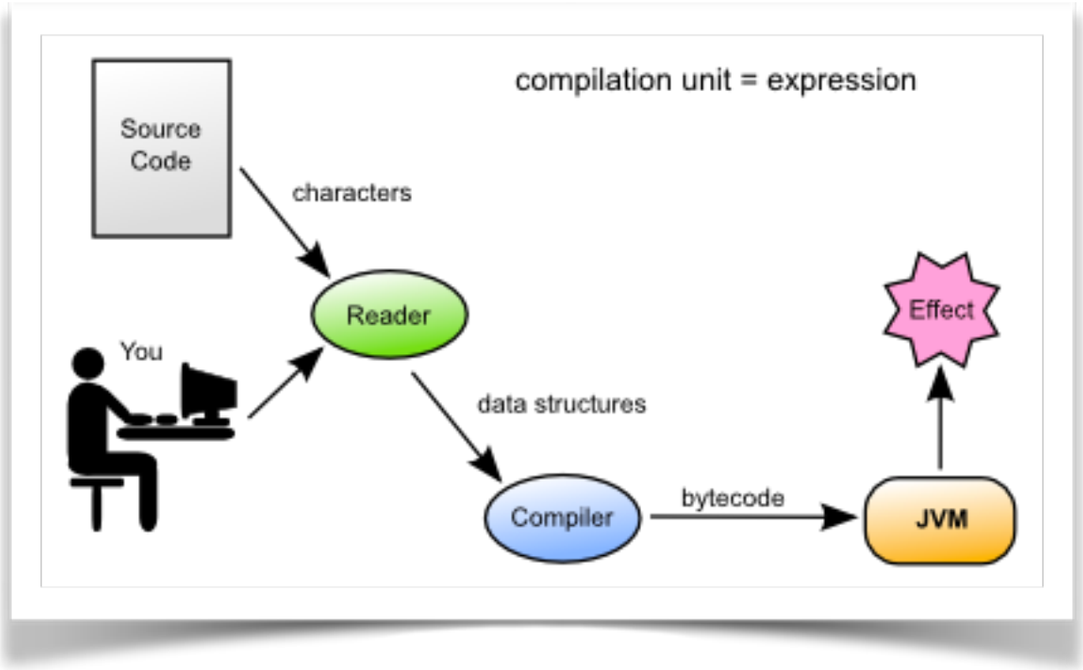
Software Transactional Memory (STM)
MVCC

atom | ref | agent

Expressiveness

Critical for reasonability





source: clojure.org

So, writing a mini language (DSL) is super easy. The compiler for your DSL is just another Clojure program that transforms one data structure to another.

No Breaking Changes (Principle)

Update your project from Clojure 1.8 to 1.10 without a single worry.

Clojure 101 (for absolute beginners)

Your code goes inside two parens

```
(…your code…)
```

Define immutable data

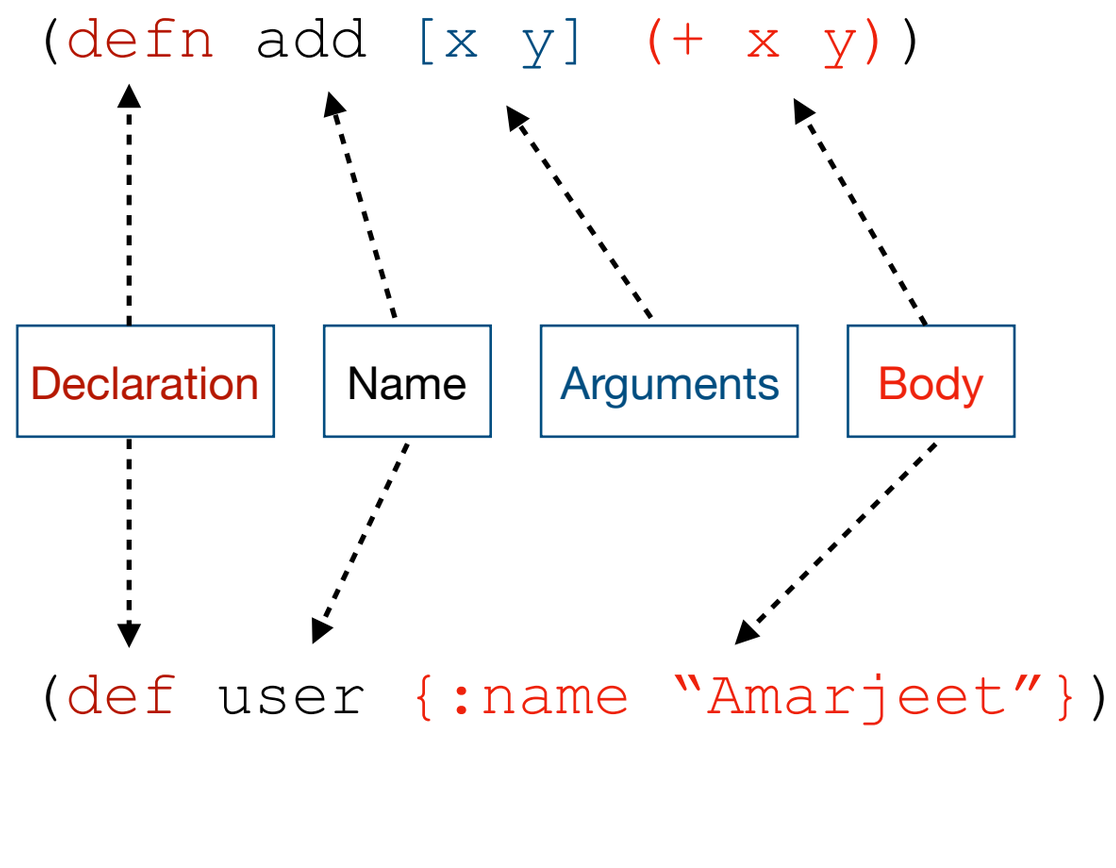
```
(def user {:name "Amarjeet"})
(def balance 1000)
```

Define function

```
(defn add [x y] (+ x y))
(defn arbit-fn [a] (add a 10))
```

Calling a function

```
(fn-name …arguments…)
(add 2 3)
(arbit-fn 5)
```



Return of a function

- the computational result of the body is the return of the function

Reach

Clojure
- JVM
- CLR
- GraalVM (community work in progress)

ClojureScript (Compiles to JavaScript)
- Browser
- Mobile
- Desktop

Use Java or JavaScript libraries via Clojure-Java and ClojureScript-JS interop

Community

One of the most active and helpful dev community today.

Majority of the conversation/help happen on **Slack**. There are some other channels, such as **ask.clojure.org** (stackoverflow for Clojure), **Google group**, **Zulip**, and **ClojureVerse**.

Each community platform adheres to community guideline to get most out of it - please follow guidelines.

Welcome to Clojure!