Clojure & Functional Programming?

Eric Normand

PurelyFunctional.tv

Outline

Eric Normand

Functional Programming

Clojure

Clojure SYNC

Eric Normand

PurelyFunctional.tv

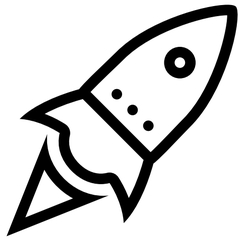
Who am I?

- UNO Alumnus (Bachelor's and Masters)
- Functional programming (Lisp) since 2001
- Got into Clojure in 2008
- Professional FP 2010-present
- Teaches Functional Programming and Clojure

@ericnormand - lispcast.com

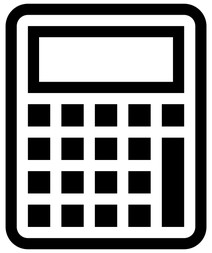
Three Domains

Functional Programming is a programming paradigm that views solutions to problems in terms of three domains. All objects in the solution fall into one of these three domains.



Actions

Actions are the broadest domain. It is composed of anything that depends on when it is run or how many times it is run.



Calculations

Calculations are the domain of mathematical computation and other “pure” calculations. They depend on execution to be understood.

Data

Data is the domain of inert values. They depend on interpretation to be understood.

Actions

the process of doing something, typically to achieve an aim

Examples

- Send an email
- Launch a missile
- Modify or read global mutable state
- Read from disk
- Send a web request

Criterion

Depend on **when they are run** or **how many times they are run**. They **change the world**.

Discipline

Concurrency guarantees

- Idenpotence
- Transactional properties
- Freedom from side-effects
- Exactly-once reads

Calculations

computation from inputs to outputs

Examples

- Summing numbers
- Statistical calculations
- Transforming from one format to another

Criterion

Depend on **being run to know what they do.** They are **opaque.**

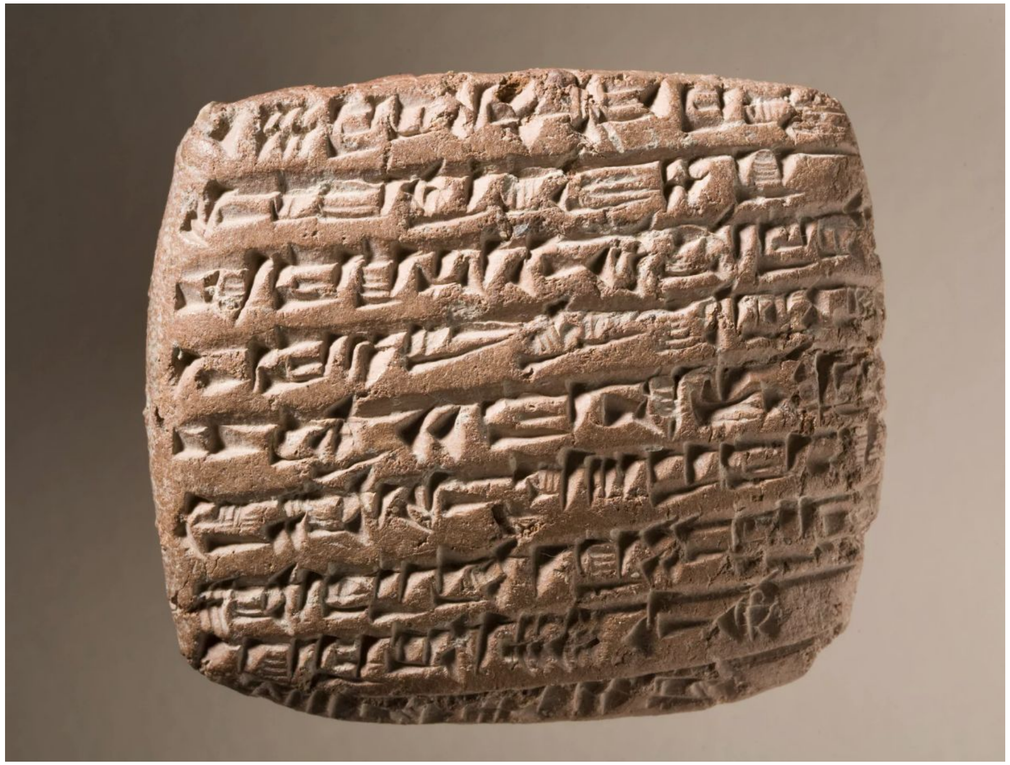
Discipline

Pure functions

- Rely only on arguments
- Return a value
- No side-effects

Data

factual information used as a basis for reasoning, discussion, or calculation



Examples

- Numbers
- Lists
- Dictionaries
- Tables
- Strings

Criterion

Depend on **interpretation**. They are **inert**.

Discipline

Immutability

- Copy-on-write
- Persistent data structures
- Append-only data stores

Clojure

Clojure is a Lisp

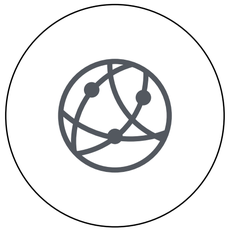
Lisp is a family of programming languages invented in 1958 by John McCarthy.

Functional Programming

Written by Rich Hickey

First released in 2007.

Concurrency

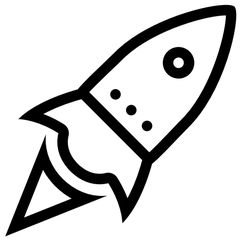


Designed to be hosted

Clojure is designed to be hosted on the JVM. It now also compiles to JavaScript.

Data-Orientation

Functional Programming



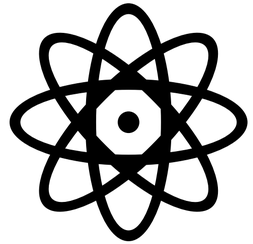
Actions

Atoms

Queues

Communicating Sequential Processes

Clojure



Concurrency Primitives

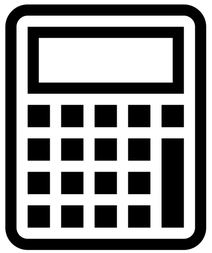
Functional Programming

Clojure

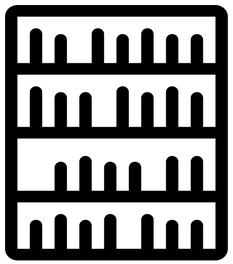
Data transformation

Pipelining

Higher-order functions



Calculations



Rich core library

Functional Programming

Data

Atomic values

Collections

Clojure

123

{}

[]

“abc”

Immutable Data

Eric Normand

LispCast

Follow Eric on:

Eric Normand

@EricNormand

lispcast.com

eric@lispcast.com