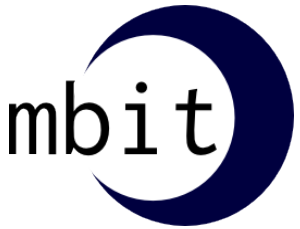
Introduction to Functional Programming

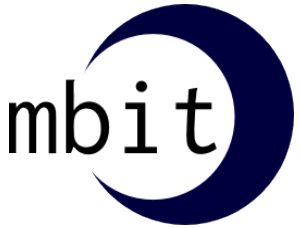
Greg Seaton

19 June 2013



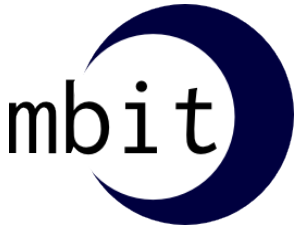
Agenda

- What is functional programming (FP)?
- How about object-oriented programming?
- Current functional languages
- Why FP?
- Why Clojure?
- Hello World
- Hello World Web
- What? No objects?
- Summary
- Links
- Questions?



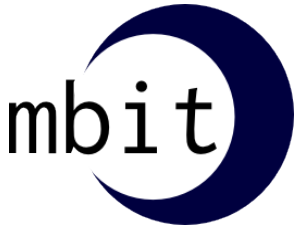
What is FP?

- Higher-Order Functions
- Pure Functions
- Anonymous Functions
- Composing
- Recursion
- Sequences and Maps
- Stateless
- Map/Reduce/Filter



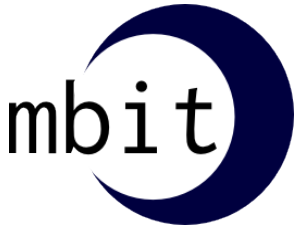
How about object-oriented?

- noun-based
- functionality may be silo'd
- may use 'pure functions' for FP-style
- mostly mutable
- limitations



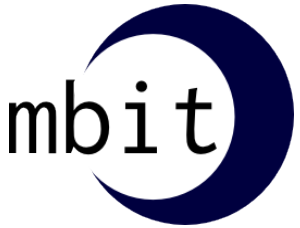
Current Functional Languages

- Erlang
- Haskell
- F#
- JVM
  - Scala
  - Clojure
- JavaScript + underscore.js
- Go
- ...and many more



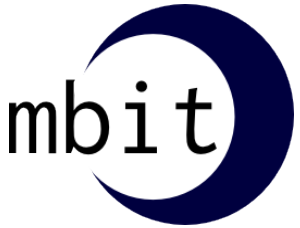
Why FP?

- elegant
  - less cruft
  - less ceremony
  - less code (locs)
- quality of code
  - immutable / mitigation of side-effects
  - easier to test
- performance
  - parallel
  - concurrent
- productivity
  - composable functions
  - REPL development



Why Clojure?

- JVM language
- interop w/ Java
- functional (impure)
- LISP dialect (since 1960)
- homoiconic (data <-> code)
- functional data structures
- productivity



Scala

- ornate
- ceremony
- powerful

mbit

Java

- cluttered
- verbose
- ceremony
- limiting

mbit

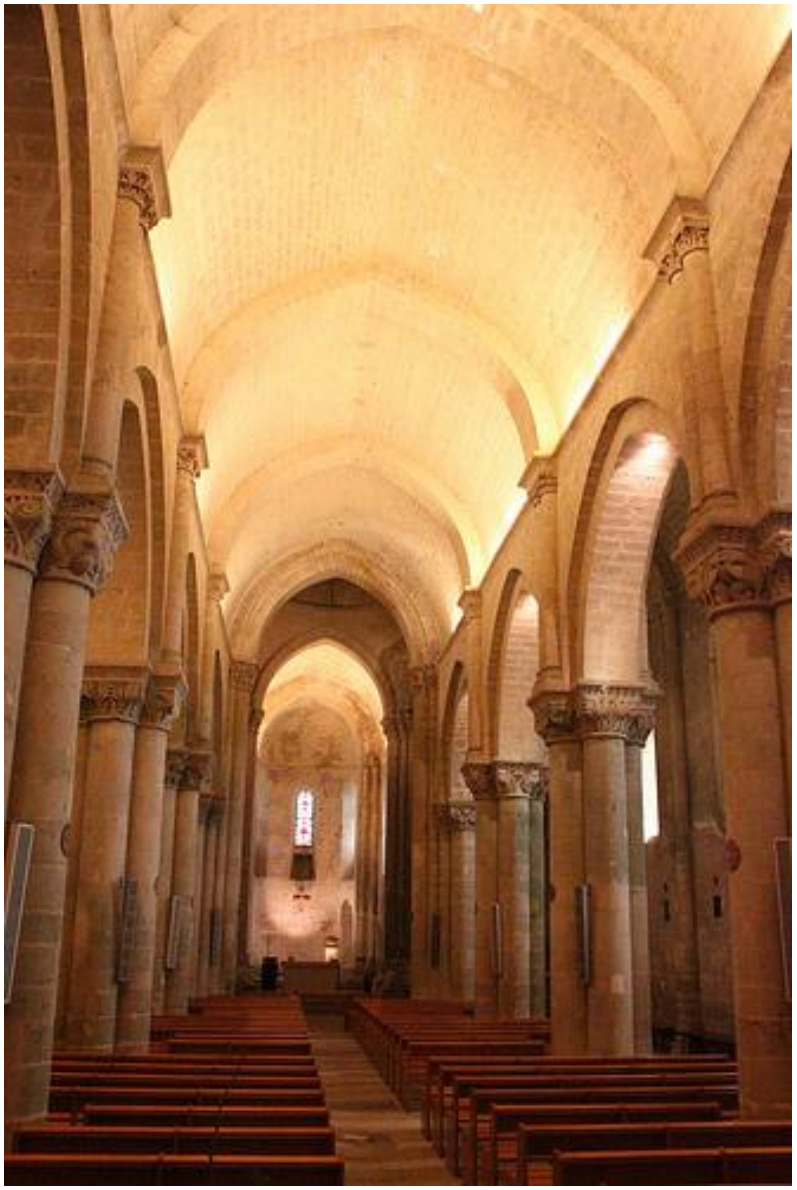
Java

- cluttered
- verbose
- ceremony
- limiting

mbit

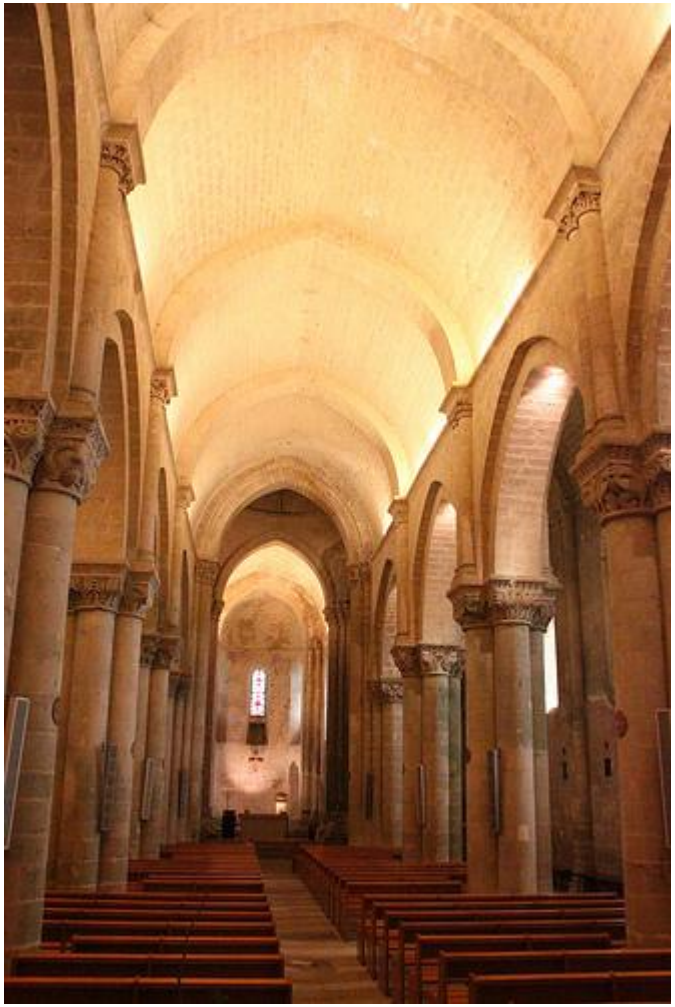
Clojure

- simple
- elegant
- powerful



mbit

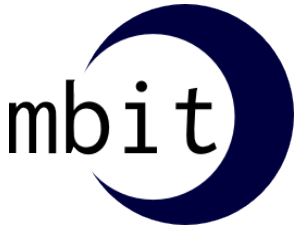
Choice?



mbit

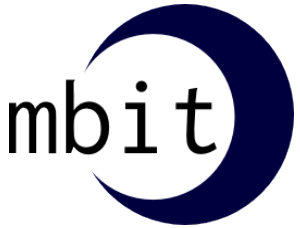
Hello World

- > lein new hi
- src/hi/core.clj
  - `(defn -main[who] (println "Hi," who "."))`
- > lein run hi.core Perry



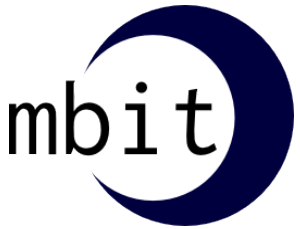
Hello World Web

- **>** lein new compojure howdy
- **howdy>** lein ring server
- src/howdy/handler.clj
  - `(GET "/howdy/:name" [] (str "howdy, " name))`
  - `(GET "/hi" [name] (str "hi, " name))`
- .../howdy/Perry
- .../hi?name=Barry



Map / Reduce / Filter

- Filter
  - `(filter even? [1 2 3 4]) -> [2 4]`
  - `(filter odd? [1 2 3 4]) -> [1 3]`
  - `(filter #(< % 3) [1 2 3 4]) -> [1 2]`
- Map
  - `(defn double [x] (* 2 x))`
  - `(map double [1 2 3]) -> [2 4 6]`
  - `(map inc [1 2 3]) -> [2 3 4]`
  - `(map #(* % %) [1 2 3]) -> [1 4 9]`
- Reduce
  - `(reduce + [1 2 3]) -> 6`
  - `(reduce + 5 [1 2 3]) -> 11`
  - `(reduce #(* %1 %2) [1 2 3 4]) -> 24`



What? No Objects?

- instances -> maps
  - **Java:**

```
public class Person {
    private String firstName;
    private String lastName;
    public Person(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }
    public String getFirstName() { return firstName; }
    public String getLastName() { return lastName; }
}
Person p = new Person("John", "Smith");
```

  - **Clojure:**

```
(def p {:ln "Smith" :fn "John" :hobbies [:read :code]})
```

mbit

What? No Objects? (cont'd)

- packages -> namespace
  - `package com.xyz.domain; // java`
  - `(ns xyz.domain) ;; clojure`

- class methods -> general functions
  - **Java:**

```
Person p = new Person("John", "Smith");
p.getFirstName(); // "John"
```

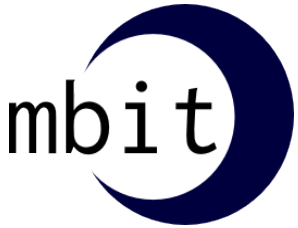
  - **Clojure:**

```
(let [p {:ln "Smith", :fn "John"}]
   (p :fn)) ;; "John"
```

mbit

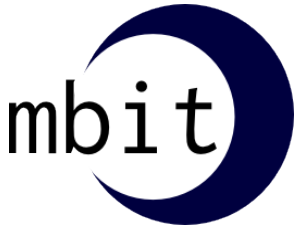
FP Challenges

- tooling
  - debuggers
  - ~editors/IDEs
- skill sets
  - most developers imperative (Java, C#, Python, et al)
  - different mindset / paradigm
- mindshare
  - management 'comfortable' w/ Java, et al
  - developers 'comfortable' w/ Java, et al



Summary

- FP
  - elegant
  - simple (![necessarily]= easy)
  - performant
- OO / imperative
  - adopted
  - works
  - limiting
- Clojure
  - JVM-based / interop
  - FP (impure)
  - pragmatic



Links

- Simple Made Easy http://www.infoq.com/presentations/Simple-Made-Easy-QCon-London-2012
- Simplicity Matters http://www.youtube.com/watch?v=rl8tNMsozo0
- Clojure in the Field http://www.infoq.com/presentations/Clojure-tips
- FP in the Enterprise http://skillsmatter.com/podcast/scala/functional-programming-in-the-enterprise-4235?goback=%2Egde_1058217_member_251421950
- Thinking in Clojure for Java Programmers http://devblog.factual.com/thinking-in-clojure-for-java-programmers-part-1-%E2%80%94-a-gentle-intro
- Clojure for Java Programmers
  - Part 1 : http://www.youtube.com/watch?v=P76Vbsk_3J0
  - Part 2 : http://www.youtube.com/watch?v=hb3rurFxrZ8
- Clojure : All Grown Up http://wit.io/posts/clojure-all-grown-up
- Twitter Storm http://www.ibm.com/developerworks/opensource/library/os-twitterstorm/

Questions?

mbit