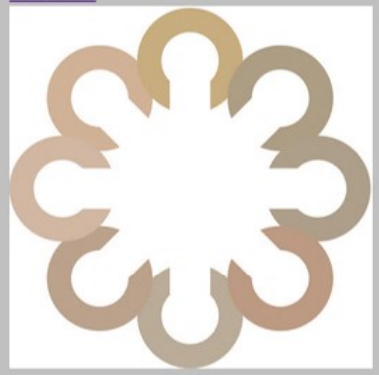
Monad is not a datatype.
It is a rule of composing functions.
Monads are simply a way to wrapping things and provide methods to do operations on the wrapped stuff without unwrapping it.

SVG-au Meetup group

SVG-AU

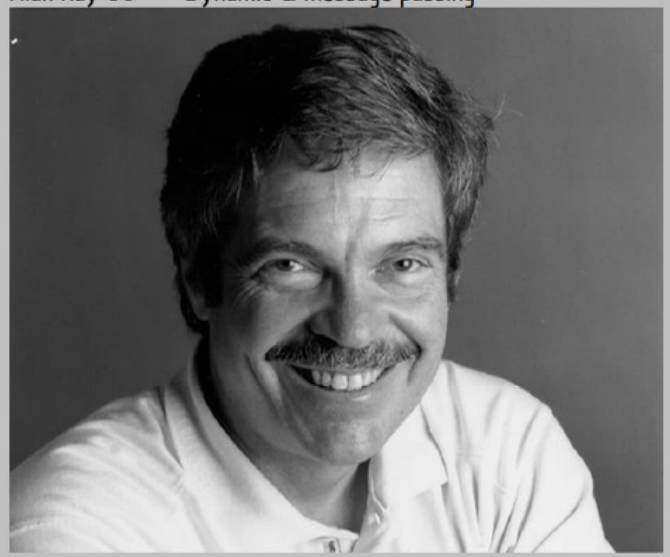


Slide references and links:

RORO

Ruby, dry-monad, intro

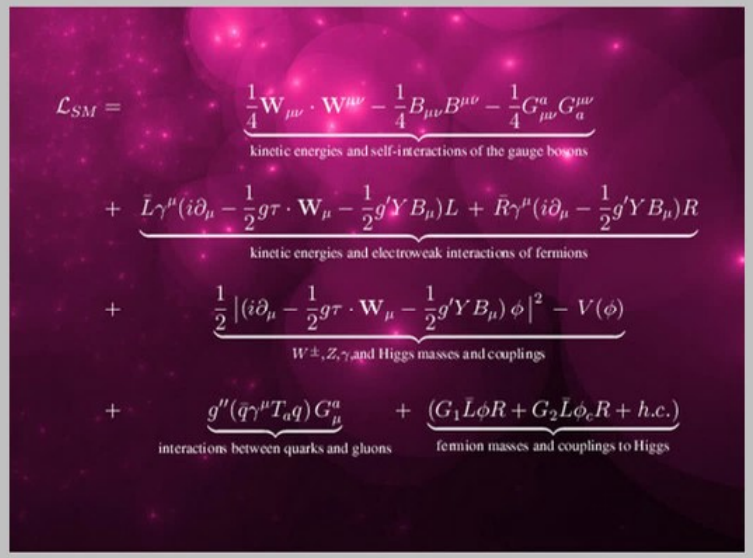
Alan Kay OO == Dynamic & message passing



Douglas Engelbart, OO == Human Modelling

Ruby, dry-monad, intro

MATHS



ScardeyCat

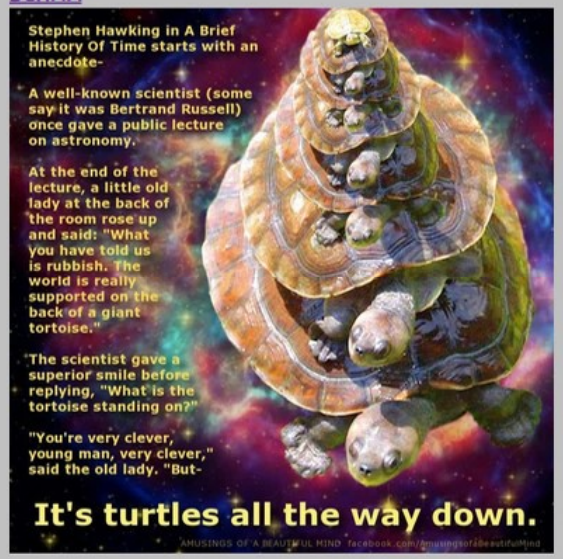
Cat Vid

How OO and Agile have lost thier way together, James Coplien

Composition
FrameworkHomework

FFT+ASIC+DESIGN

SCRAM



Ruby, dry-monad, intro

- Class Oriented vs Object Oriented
- Java vs Ruby
- Predictability vs Flexibility
- Provability: Secure Kernel SEL4
- OO message passing circles back with Erlang messages
- Inevitably Elixr
- Ruby flexibility ‘require’s monolithic Object tree.
- Ruby origins: C, Lisp, Perl
- Listishness brings enumeration happiness.
  Except for typish-functional langs.

- Closures, scope local
- Conceptual flexibility of Ruby with processing accuracy of f(n)
- Hazards of f(n), exceptions
- Control flow
- Concurrency
- Erlang-Elixr
- Scala, Clojure
- ES6, Rust, Go
- Use ES6 to learn functional
- GDPR
- Functional Sandwich
- #@MPJ
- Monadic Data Flow
- Railway-OP
- ScottWlaschin-ROP

Ruby, dry-monad, intro

- Compilers
- Transpilers
- Hazards of f(n), exceptions
- Control flow
- ES6 version management
- NodeJS module hazards
- debug issues
- Ruby gems, good to “go”
- WASM in browsers
- WRASM
- Popularity of browser extended functions. eg: webVR
- Railway-OP

Ruby, dry-monad, intro

map
bind
yield
run
find
each forEach
then
.then
.flatmap .chain
blocks {}
.call

Ruby, dry-monad, intro

- Blending of OO with f(n)?
- Advantage of monolithic Xception handling
- With microscopic failure indicators of monads
- Concurrency performant?
- Reliability & predictability of f(n) with dynamic?
- Keep the visibility of prying?
- Business/conceptual flexibility of dynamic modelling?

???