JAVASCRIPT'S

FUNCTIONAL WAY

INDEX.JS

- Why FP?
- Lambda vs State (Holly Paradigm wars)
- FP+OO
- FP Patterns in JS
- Monads and Functors in Real Life
- FRP concepts

WHY FP?

- Minimize side effects of
- Minimize amount of code
- Simplify testing

FP STATEMENTS

- Category Theory
- Lambda calculous
- Functional approach
- Expressions and constants
- Determinism/abstraction

IMPERATIVE STATEMENTS

- Automata theory
- State machine
- Procedural approach
- Variables
- Butterfly effect/changing

BENEFITS

- No side effects
- Highly abstract
- Less of code

RESTRICTIONS

- Entry-barrier is high
- Hard to store/cache/memoize
- Hard to IO

ABSTRACTION, COMPOSITION, IDENTITY

- Abstraction:
  what can be more abstract than y=f(x)...?
- Composition:
  if a=>b and b=>c, so there is identical way a=>c, so that
  b = f(a), c = g(b), there is c=h(a), which composes:
  h = (a) => g(f(a))
- Identity: a => a (way to itself)
- Associativity
  h->(g->f)=(h->g)->f
- Determinism

OOP FEATURES

- Composing:
  - bind cohesive things together,
    - context functions (methods)
- Abstracting: hide details, expose interface

OOP DRAWBACKS

- Sharing data
- Mutation
- Data racing

THE LAST THING YOU WANTED ANY PROGRAMMER TO DO IS MESS WITH INTERNAL STATE EVEN IF PRESENTED FIGURATIVELY. INSTEAD, THE OBJECTS SHOULD BE PRESENTED AS SITE OF HIGHER LEVEL BEHAVIORS MORE APPROPRIATE FOR USE AS DYNAMIC COMPONENTS. IT IS UNFORTUNATE THAT MUCH OF WHAT IS CALLED "OBJECT-ORIENTED PROGRAMMING" TODAY IS SIMPLY OLD STYLE PROGRAMMING WITH FANCIER CONSTRUCTS. MANY PROGRAMS ARE LOADED WITH "ASSIGNMENT-STYLE" OPERATIONS NOW DONE BY MORE EXPENSIVE ATTACHED PROCEDURES.

Alan Kay,
The Early History of Smalltalk

OOP PRINCIPLES

- Encapsulation?
- Inheritance?
- Polymorphism?

COMPOSITIONS, MONADS AND FUNCTORS

- Why Monads?
- B/C we can!

MONADS IRL

- Promise
- Pipelines
- Observable streams

FRP CONCEPTS

- Event-Driven
- Continuous data-flow
- Immutable
- Pure-functional