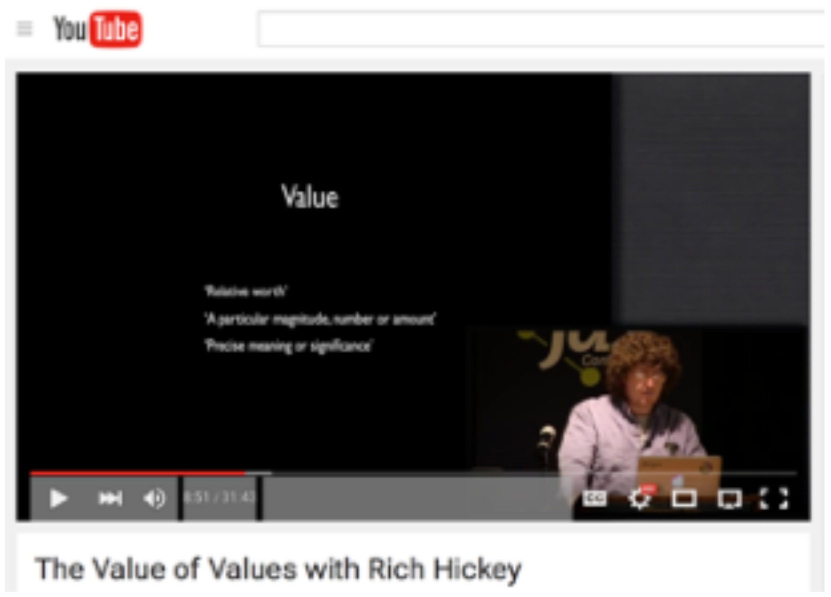
Data Oriented

- Clojure programs are written in the basic literal data structures of the Clojure programming languages
  - {}, [], #{}, ()
- The preferred way to encode domain models is as data, not classes (but classes are fully supported)
- Clojure has powerful data abstractions that extend across the language for reuse
- Clojure is Homoiconic, meaning it is written in its own data structures
  - The language can be extended at will using macros
  - Despite being a powerful feature, you generally don’t write many macros
- Clojure is easy to read, reason about, and debug
- There are no other mainstream JVM (or other?) languages that are as data-oriented as Clojure

Value Oriented



- Shareable
- Comparable
- Reproducible
- Synthesizable
- Universal
- Generic
- Aggregable

- Conveyable
- Perceivable
- Rememberable
- Interfaces
- Coordinated
- Movable

- Most languages: immutable classes, but generally no baked in value support
- Better languages: immutable collections, but no deep updates or retrieval
- Few have a model for handling state (e.g. Scala stops at immutability)
- Clojure:
  - All of the above
  - All of Rich's points
  - STM

https://www.youtube.com/watch?v=-6BsiVyC1kM