Separated Concerns

- Clojure's design separates fundamental concerns:
    - Value: Modeled as data
    - Transition: Pure functions
    - State: Atoms, agents, refs (STM)
    - Shape: spec
- Clojure applications have well-defined boundaries between the model and view layers. It is easy to create a domain model as pure Clojure and write desktop and web front ends.
- Functional languages with strong Object/Type systems complect concerns
    - E.g. Scala

Interactive

- The REPL (Read-Evaluate-Print Loop) is an environment that allows for interactive development
- Also found in Scala, Python, Ruby, and Haskell (among others)
- Java 9 has a REPL, but it is pretty weak
- Developing/debugging in Clojure is often done via generating inputs as data, interactively developing transformations, and lifting the result into a function
- You rarely need an interactive debugger or tools to step through code (but you can)
    - Note: Stack traces can be ugly