Types and Classes and Prototypes

- Types are a skin on prototype-based substrate
- Classes are types
- Types help optimize apps by separating what can be done early vs. later
  - “early” is currently called “compile time”
  - “later” is currently called “run time”
  - *early —> later* is a continuum, compile-time vs. run-time is an artificial distinction
  - in the end, *everything* is interpreted (hardware is an interpreter)

Layers of Functionality

- Many layers of simple-input-API —> simple-output-API —> simple-input-API —> simple-output-API —> …
- ‘+’ is an implementation detail
- implementation details appear *only* in bottom-most layer
  - e.g. all other layers have no syntax for ‘+’
  - e.g. layers can call functions in layer immediately below them (not up, not layers below-below, etc.)
  - e.g. diagrams makes this layering painfully clear
    - use DaS (Diagrams as Syntax)
    - diagrams show input APIs as a set of input ports
    - diagrams show output APIs as set of output ports (N.B. *output* API as opposed to return value(s))
    - diagrams show *composition* as boxes on diagram
      - no need to dig any further down than one layer, before switching to another diagram
      - hierarchy of diagrams (3D across and in, as opposed to 2D across-only)
  -