S/SL Dataless PL

- S/SL is dataless
- programmer can declare existence of data
- programmer can move data (using handles)
- programmer *implements* data and operations in an underlying language (a Toolbox language)
- removing data operations expresses DI (Design Intent) without drowning the design in low-level details

S/SL Pipeline PL

- S/SL programs composed of “little” passes pipelined together to make a whole system
- Each pass is completely *isolated*
  - build and forget
  - no implicit dependencies between passes
- pass-based, isolated architecture is the epitome of recursive design (aka divide and conquer)
- passes are “bite-sized” actions
  - with explicit input and output APIs
  - sequenced by input token stream
- e.g. Concurrent Euclid - scanner, parser, semantic analyzer, allocator, emitter