- Pinocchio
  - Problems with VMs
    - closed-box VM
      - VMs not adaptable
        - can't anticipate all needs
        - not open for extension
      - need to change and extend VMs
        - eg supporting tail call elimination for Lisp
        - eg supporting become: for Smalltalk
      - boundary between language and VM
        - forces specialized VM implementation
    - host-language lock-in
      - VMs not reusable
        - eg invokedynamic only available to Java 7
    - metaprogramming restrictions
      - extension points must be hardwired into the VM
        - eg using objects as methods in Pharo
    - need two-way interface between languages
  - Nutshell
    - The Message is the Medium
    - runtime, not VM
      - self-supporting
        - available at runtime
          - not just compiled away
          - in Squeak and PyPy, you have no access to runtime
      - only hardwires message sending
    - Message Sending
      - runtime library
      - invocation convention
      - prefilled monomorphic inline caches
        - caches prefilled at compile time
    - metalevel is Smalltalk code
      - ie message sending
        - so compiler builds MOP for you
      - compiles to machine code
      - unify language metalevel with user code
        - freely move between levels
  - Implementation
    - Object headers
      - Object = data plus pointer to behaviour
    - Message sending
      - runtime library
      - invocation conventions (interface)
        - native call to meta-level implementation
        - lookup & apply
          - meta-level invoke function performs lookup and apply
          - invoke can be applied to any object
          - looks up selector and performs found behaviour
      - prefilled inline caches
        - formal binding stays at runtime
          - actual binding at compile time
          - can be overridden at runtime
        - solves the bootstrapping problem
          - avoid meta-regression
        - type hints
          - avoid static compilation of metalevel
          - avoid need for statically typed language
          - tells runtime where to find compiled method if type matches
    - Native compilation
      - Conventional architecture
    - Bootstrapping
      - compiler cross-compiled from Pharo
    - Performance
      - 3 benchmarks
  - Applications
    - Replacing a MethodObject
      - foobar example :-(
    - DoesNotUnderstand
      - example of runtime extension to support new feature
    - Extending behaviour
      - eg prototype-based lookup
    - Object Flow Analysis
      - does not currently run in Pinocchio yet
      - either AST interpreter approach or full compiler approach
    - two-way communication with C
    - profiler
      - counting msg sends