# Array-based Lisp?

@raviqqe

# Daydreaming about an array-based Lisp language...

# Progress so far

- https://github.com/raviqqe/arachne
- Single-word runtime value with NaN boxing
- Reference counting GC
- Interpreter
  - AST-based implementation of primitive operations
  - Bytecode VM
    - Work in progress...

# Virtual machine

- Stack machine

## Instructions

`nil`, `float64`, `symbol`, `local`, `get`, `set`, `length`, `add`, `subtract`, `multiply`, `divide`, `call`, `closure`, `equal`, `array`, `drop`, `dump`, `jump`, `return`

- `nil`, `float64`, `symbol`: Pushes a constant.
- `local`: Gets a value of a local variable.
- `get`: Gets a value from an array.
- `set`: Sets a value to an array.

# Types

- Float64
- Symbol
- Function
- Array
- Nil
  - `()` = `0` = `false`

# Design decisions

- Operand evaluation order & argument order in a stack
  - Scheme doesn't specify its operand evaluation order in its specification.
- Tight or loose coupling between bytecode compiler and VM
  - Is it ok to embed runtime values into bytecodes?
  - Do we want to save bytecodes of modules in a file system?

# Next tasks...

- `call` instruction
- Closures

# Summary

- Daydreaming a language is fun.

# Ribbit

- https://github.com/udem-dlteam/ribbit
- AOT compiler + RVM
- Everything is a rib.
    - Rib is a three-word data structure.

## Objects

## Bytecodes

- You can GC bytecodes!