JUNE 2
JUST-IN-TIME (JIT) Compilation

Delay Compilation of (Parts of) Program
- Code is not all needed (at first)
- Compile w dynamic context

Unit of Compilation
- Function
- TRACE

JIT for Short Functions

Runtime Environment:
- Compiler
  → Produce machine code into memory
- Source code
  → GC

Start by compiling max(F)

```
max(F) f max:
```

```
F := max(F)
```

```
TRAPPOLE

SCAP T,F

COMPILES F

RUB CODE W

MEMORY

@ PROCESS F

OVERWROTES ITSEF

W/ A JUMP T TO F

OVERWRITE LAST F

JUMP TO F
```

Interpreter + JIT Compiler

When a function becomes "hot"

Then compile it.

Maybe have a dumb, fast JIT

```
A SMART, SLOW JIT
```

"Hotspot"
TRACES — AN ALTERNATIVE TO FUNCTIONS AS UNIT OF SITTING.

\[ f(c) = \begin{cases} \text{undefined, } k = 1, c < 2 \\ \text{blackness, } 3 \leq k < c \\ 3 \end{cases} \]

Compiler code for TRACES

- OCAML - Byte Code Interpreter or Machine Code Generation
- LLVM - CLANG
- Web Assembly
TARGETS
RISC, VM's
MULTI-PROCS, GPO's

SOURCES
NESTED
CH 2 FIRST-CLASS FUNCTIONS (CLOSURES)
CH 11 PARAMETRIC POLY TYPE3

- 63 RTS

EXCEPTIONS
MULTI-THREADING
LOGIC LANGUAGES e.g., PROLOG

CORRECTNESS
VERIFICATION & VALIDATION
COMPLECT, CAKEML, VELLUM