# CSE 505
# Programming Languages

## *Intro*

# Today: Introduction

Administrivia

Motivation and Goals

Overview

Caml Crash Course

# Hello! My name is...

Zach Tatlock

ztatlock@cs

CSE 546

*door is open!*

New faculty ⇒ first ever course!

# Resources

Web:

`http://courses.cs.washington.edu/courses/cse505/13au/`

Mailing List:

`cse505a_au13@u.washington.edu`

Piazza:

`(your course email will be subscribed)`

# Structure

## 4-5 Individual Homeworks:

- “pen & paper” (TeX) proofs
- implementations in OCaml
- challenge problems optional

## Midterm & Final

## Useful, Optional Reference

Pierce’s *Types and Programming Languages*

© Alamy

# Academic Integrity

## Do. Not. Cheat.

Erodes the very foundation of academia.

Absolutely not worth the risk.

## Roughly:

Discuss problem and *sketch* ideas together.

Write your own solutions, note discussion partners.

## When in doubt, ask!

# Today: Introduction

**Administrivia**

Motivation and Goals

Overview

Caml Crash Course

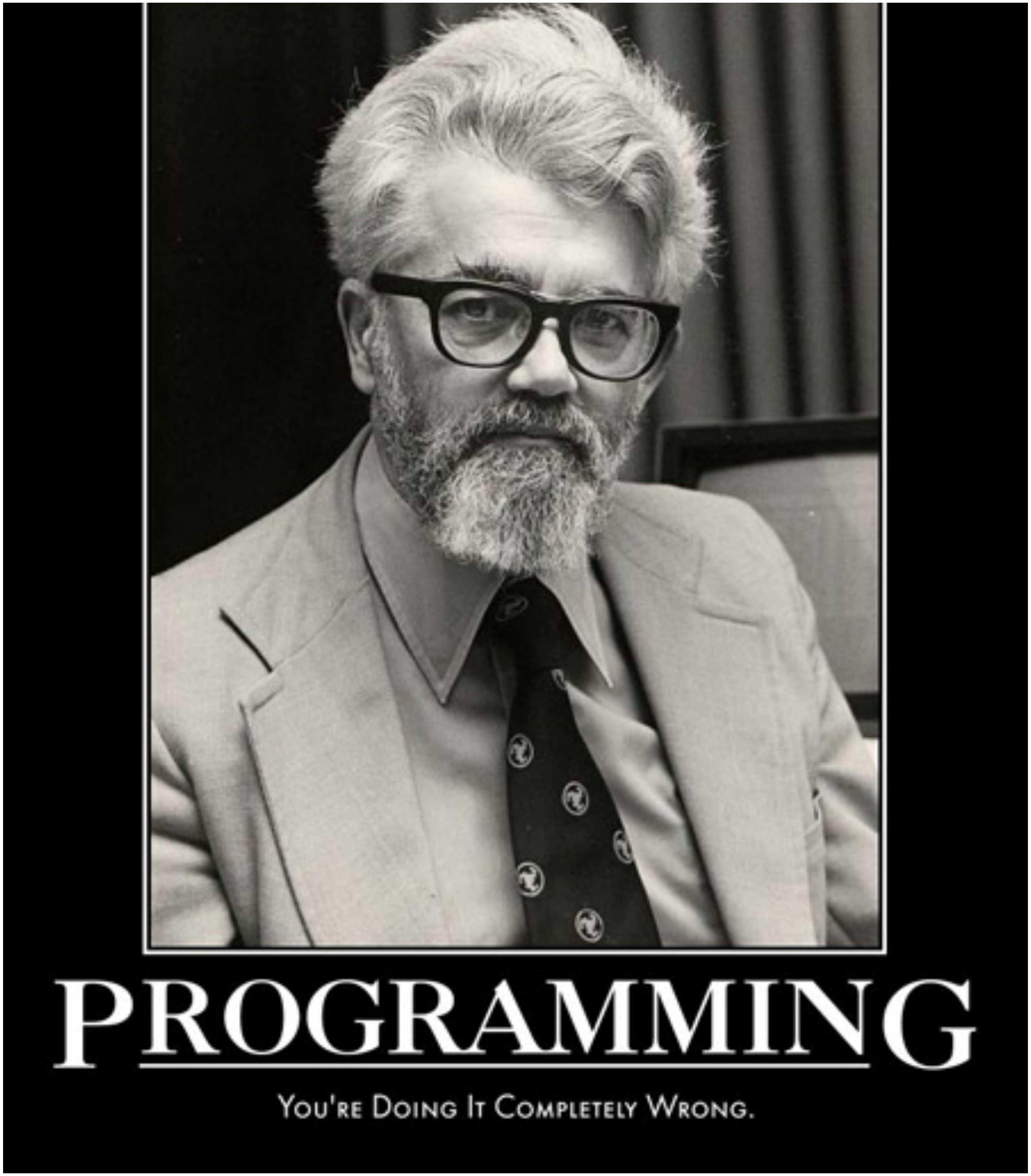
# Today: Introduction

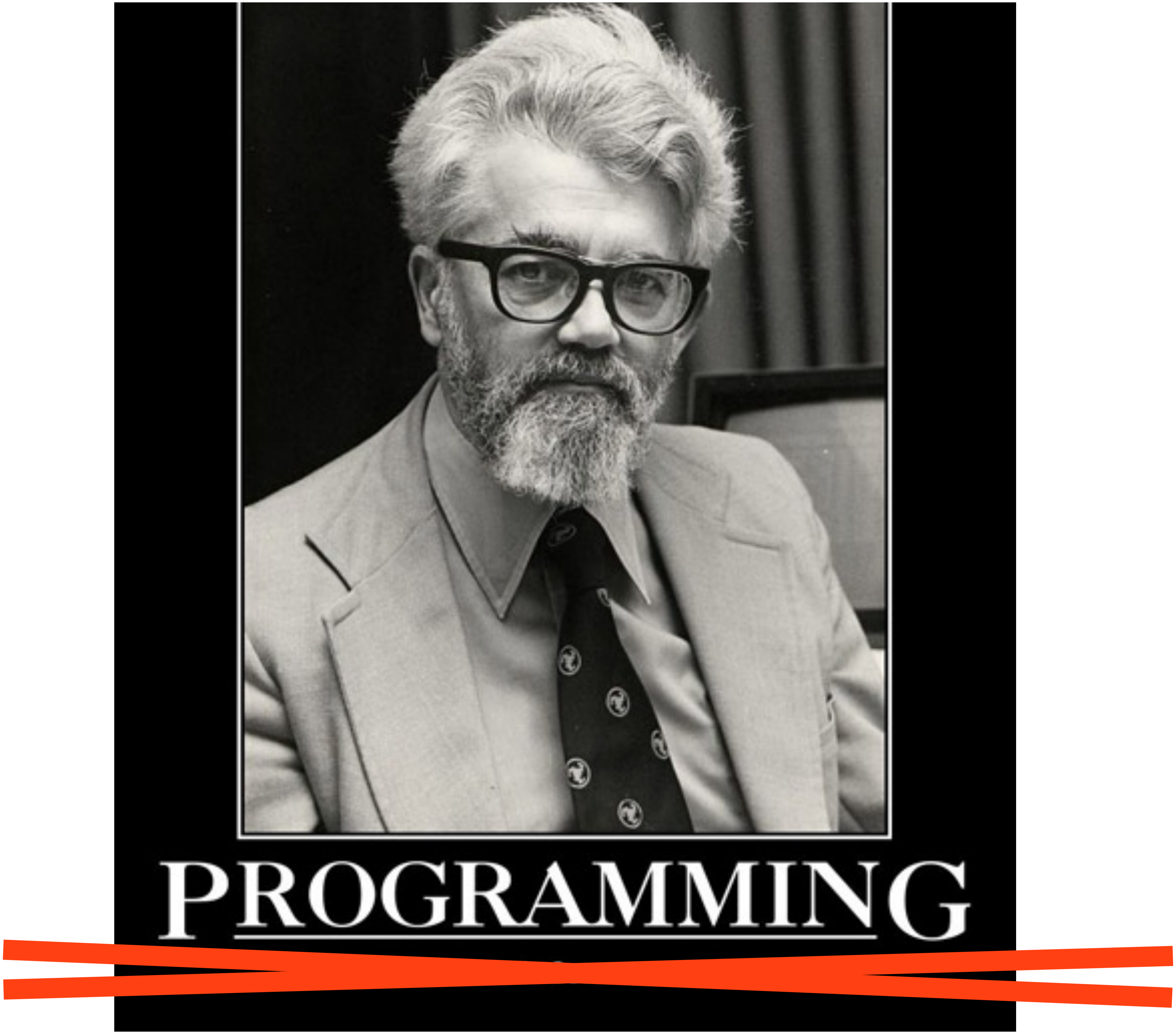
Administrivia

**Motivation and Goals**

Overview

Caml Crash Course


PROGRAMMING
You're Doing It Completely Wrong.

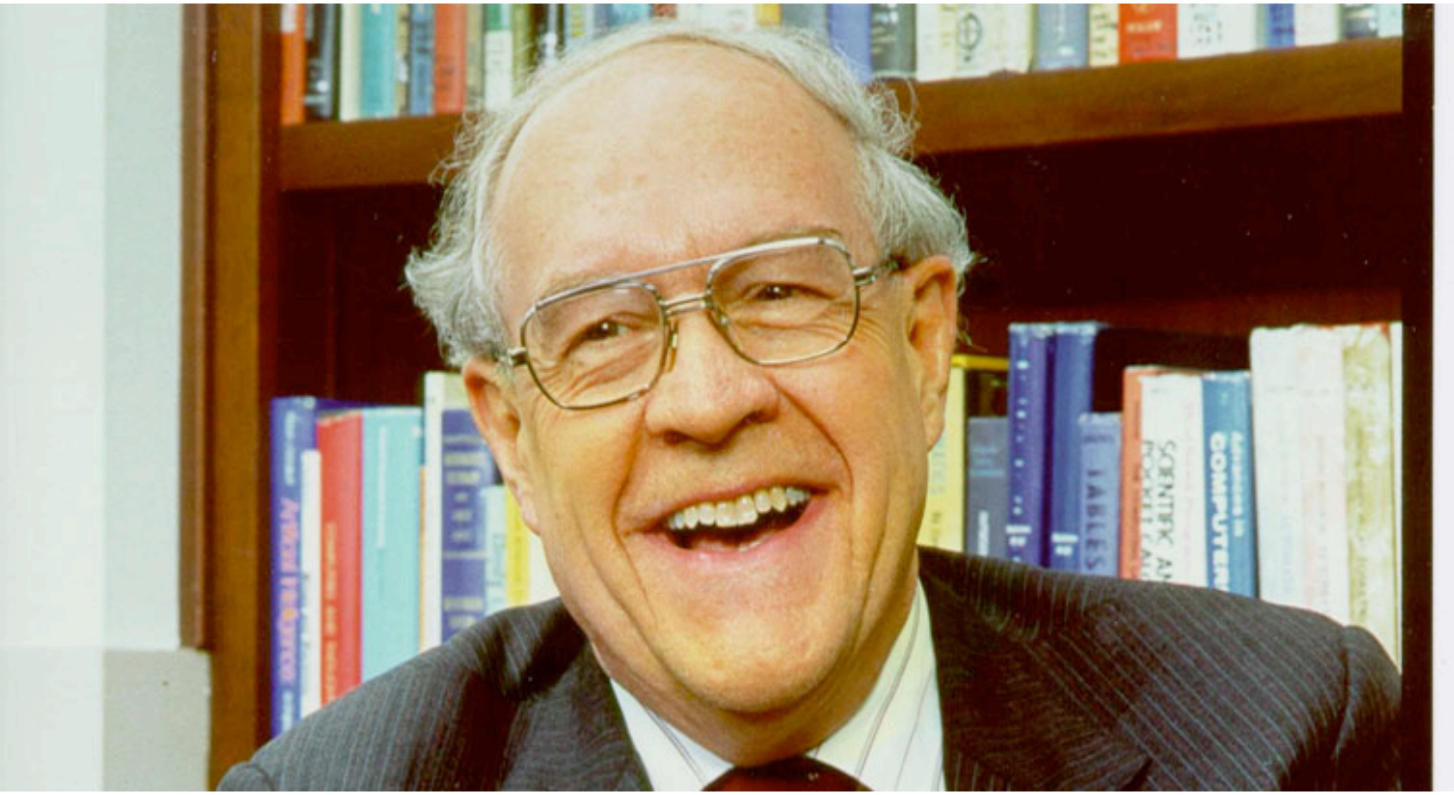

PROGRAMMING

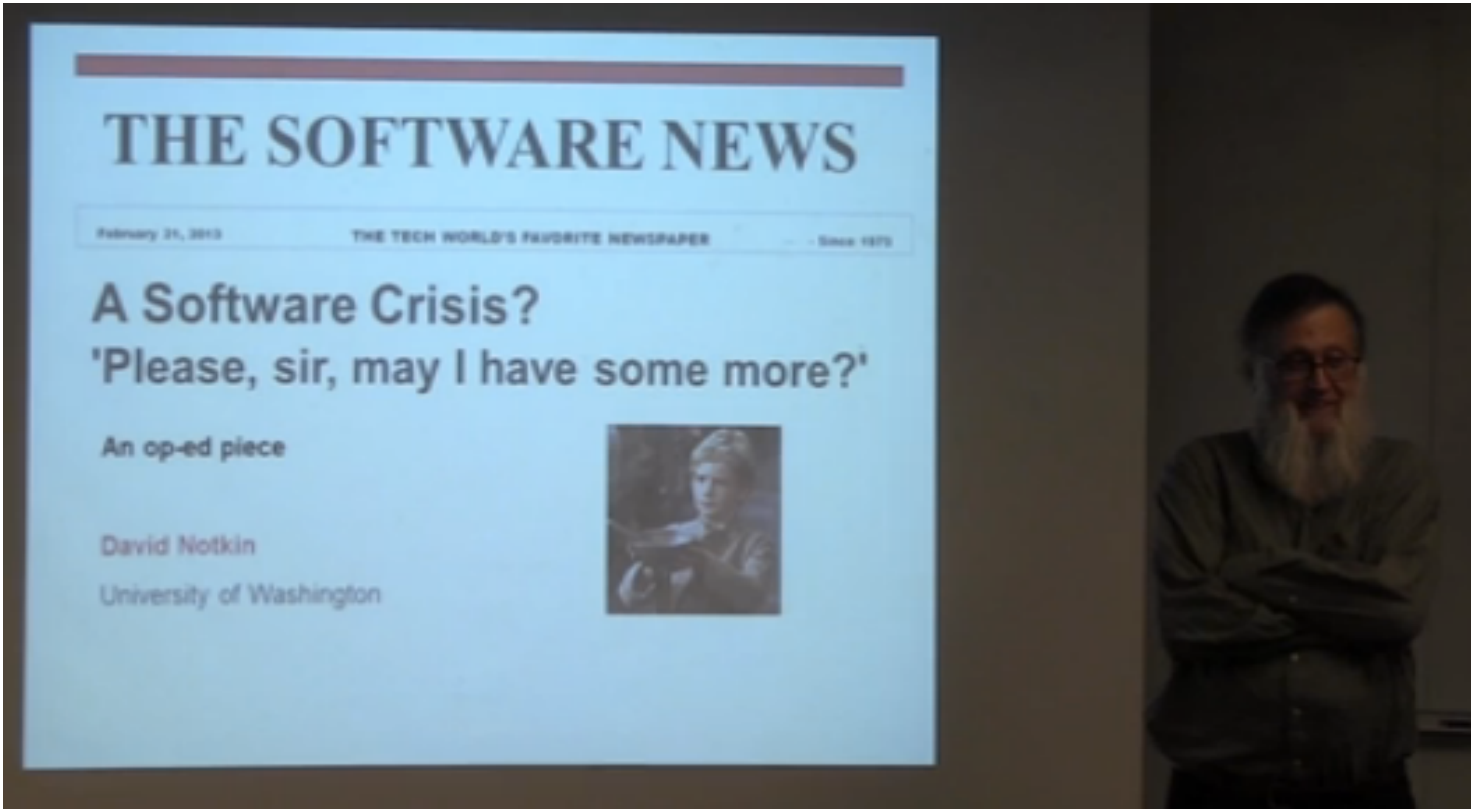
SOFTWARE

IS

GREAT!!!

> *The magic of myth and legend has been realized in our time. One simply types the correct incantation on a keyboard, and a display screen comes to life, showing things that never were nor could be.*
>
> *Fred Brooks*





THE SOFTWARE NEWS

February 21, 2013 THE TECH WORLD'S FAVORITE NEWSPAPER - Since 1975

A Software Crisis?

'Please, sir, may I have some more?'

An op-ed piece

David Notkin

University of Washington

http://cs.brown.edu/events/talks/notkin.html

SOFTWARE IS GREAT!!!

... but (often) still poorly understood!

# wat

Wat

@garybernhardt

https://www.destroyallsoftware.com/talks/wat

# Room For Improvement

## Development

quickly produce high quality code in teams

## Maintenance

comprehend, extend, fix bugs, tune

## Reliability

avionics, medicine, finance, nuclear power

**Require reasoning!**

# What do these do?

```
a=0?(3>2?23:(2>5?(7<6?34:48):64)):1;
printf("%d",a);
```

```
 printf("%d", printf("%d",
   printf("%d", printf("%s", "husky"))));
```

Realistic?

# Safe to optimize / refactor?

```
class A { int f() { return 0; } }
class B {
  int g(A x) {
    try { return x.f(); }
    finally { foo(); }
  }
}
```

⇓

```
class A { int f() { return 0; } }
class B {
  int g(A x) {
    return 0;
  }
}
```

# Safe to optimize / refactor?

... nope.

**`A`** could be extended

**`x`** could be null

**`s`** could have “side effects”

How can we handle general case?

How can we be sure it’s right?

How do we automate?

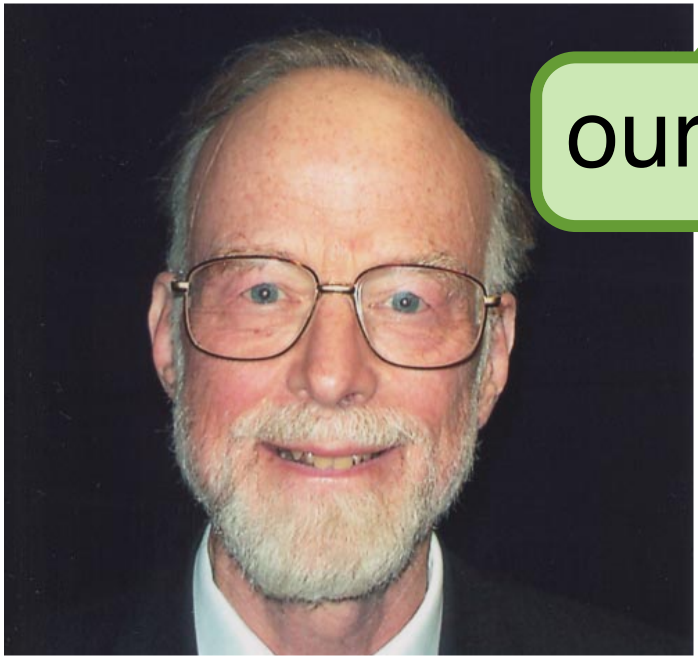
# Goal

Develop tools to **rigorously** study what programs mean.

*semantics*

*equivalence, termination, determinism, ...*

> *... [rigorous proofs about programs] are an absolute scientific ideal, like purity of materials in chemistry or accuracy of measurement in mechanics. The value of purity and accuracy (just like correctness) are often not appreciated until after the scientist has built the tools that make them achievable.*
>
> *Sir C.A.R. Hoare*



our goal is noble!

# Benefits

Writing a PL-ish thing is inevitable

*extensible systems, rich data structures, optimizer*

Build skill with “Theory B” formalisms

*all research needs precision, expressiveness, clarity*

Become better programmers

*travel to understand where you’re from*

# Today: Introduction

Administrivia

**Motivation and Goals**

Overview

Caml Crash Course

# Today: Introduction

Administrivia

Motivation and Goals

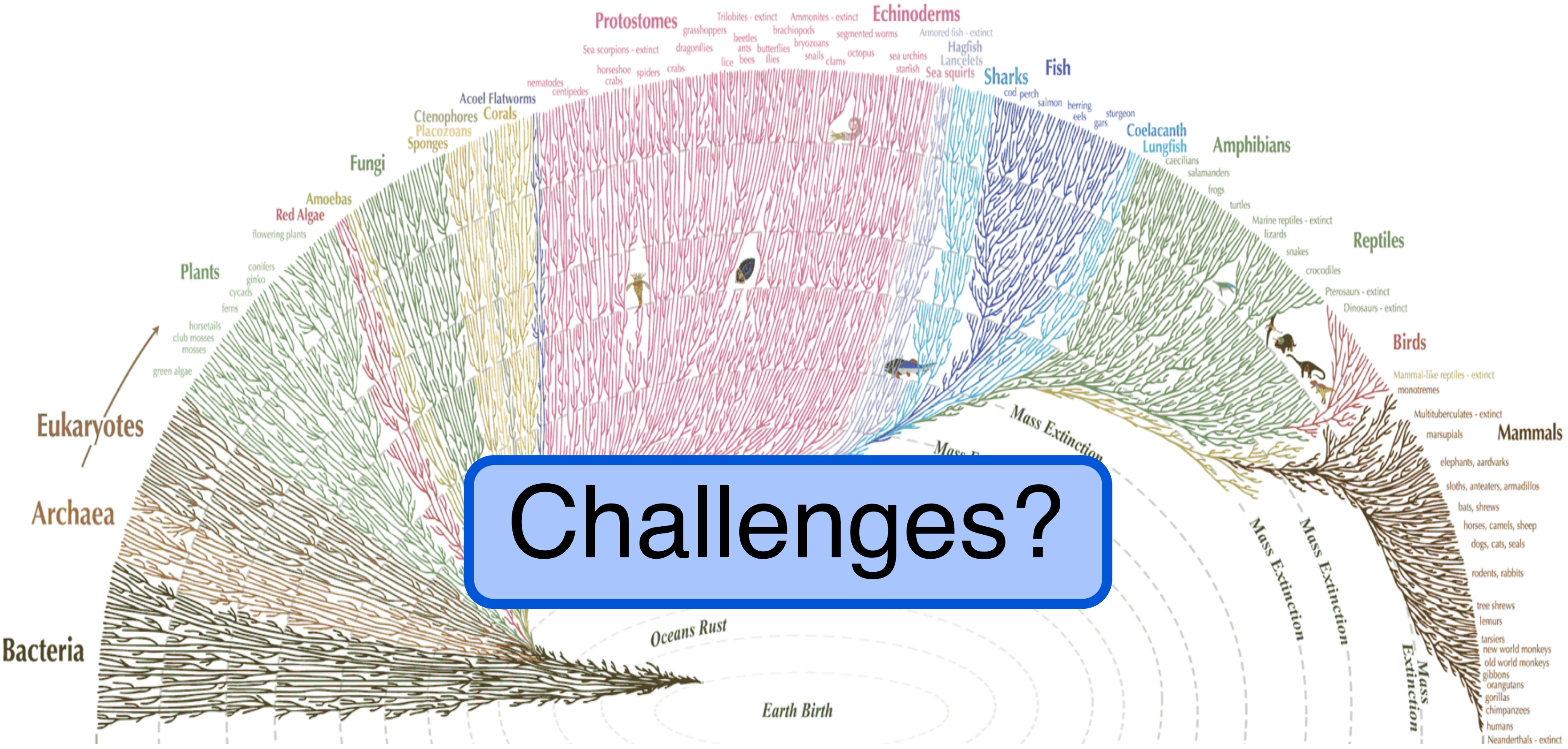
**Overview**

Caml Crash Course

# Which PL to Study?

Well... which is the best? Depends.

Aren’t they all the same? Yes and no.



Challenges?

# Approach

10 weeks is short!

Define small, tractable languages

*Turing complete, but not for "real" programming*

Extend with increasingly rich features

*extend reasoning techniques in parallel*

Sketch application to "real" PLs

*implement programs to connect theory with code*

# Subgoals

Develop tools for studying program behavior

*inductive defns, structural induction, inference rules*

Investigate core PL concepts

*types, functions, scope, mutation, iteration*

# Today: Introduction

Administrivia

Motivation and Goals

**Overview**

Caml Crash Course

# Today: Introduction

Administrivia

Motivation and Goals

Overview

**Caml Crash Course**