Why OCaml?

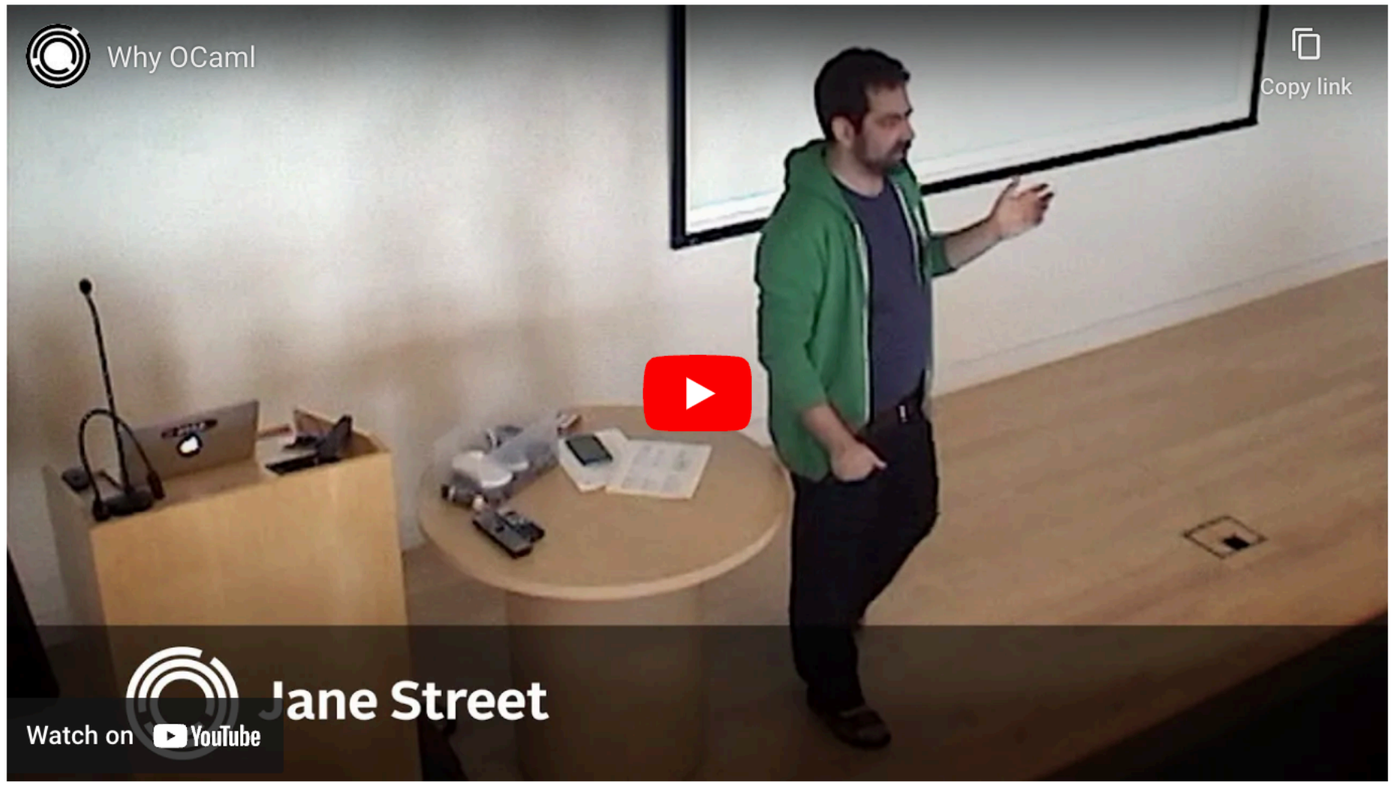
“KC” Sivaramakrishnan

- **Building functional systems** using OCaml
- We work on
  - **OCaml platform** — Compiler, Build system (dune), package manager (opam), documentation tools (odoc), editor support (LSP, merlin)
  - **OCaml community** — ocaml.org, CI for package repository, running conferences & events, managing community infrastructure
  - **OCaml consulting** — helping commercial users with OCaml, training
  - **Research** — formal verification, blockchain forensics, Unikernels support for space & IoT

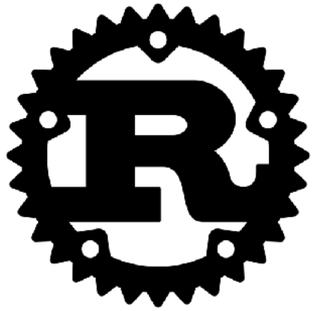
Why OCaml?

JAN 25, 2016 | 1 MIN READ

By: Yaron Minsky



Here's a post from a talk I gave this last summer during our internship program about why we use OCaml. It spends a lot of time on how OCaml fits into the space of programming language designs, and why we think OCaml is in a real sweet spot in that design space, especially for the kind of work we do at Jane Street.



Why OCaml in 2024?

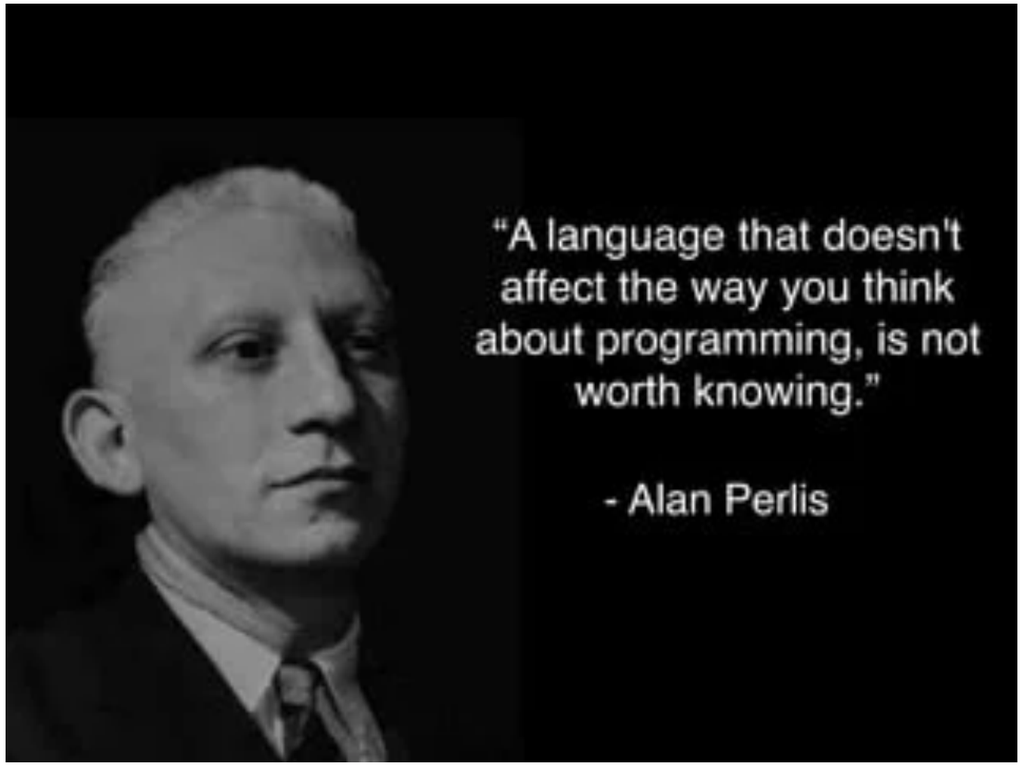
Language

- Algebraic data types and pattern matching
- Functions as a core primitive and data
- Parametric polymorphism
- Static types
- Type inference

Demo: Predicate langauge & evaluator

Staying ahead of the curve

- Algebraic data types and pattern matching
  - Released in Python 3.10 (2021)
  - Released in Java 17 (2021)
- Functions as a core primitive and data
  - Released in Java 8 (2014)
- Parametric polymorphism
  - Release in Go 1.18 (2022)
- ~~Strong static~~ types
  - TypeScript (JavaScript); Hack (PHP); Pyright, Pyre (Python) (2010s)
- Type inference
  - Local type inference for `var` in Java and `auto` in C++ (2010s)



Compiler

- Is fast and produces fast ***native*** code (x86, ARM, RISC-V, Power, …)
  - Static typing ensures efficient code is generated
- Has a bytecode interpreter
  - REPL
  - **Demo:** dune utop
- OCaml can be compiled to JavaScript (js_of_ocaml, ReasonML) and Wasm (wasm_of_ocaml)
  - *First GCed language to target the Wasm GC proposal*
  - **Demo:** OCaml playground

Build System — Dune

- Super fast, featureful
- **Demo:** Irmin
  - cloc
  - time dune build
  - Pattern matching — proof.ml:169
  - dune build --watch
    - merge.mli:19

Memory Safety & Security

Microsoft: 70 percent of all security bugs are memory safety issues

Percentage of memory safety issues has been hovering at 70 percent for the past 12 years.

Written by **Catalin Cimpanu,** Contributor

Feb. 11, 2019 at 7:48 a.m. PT

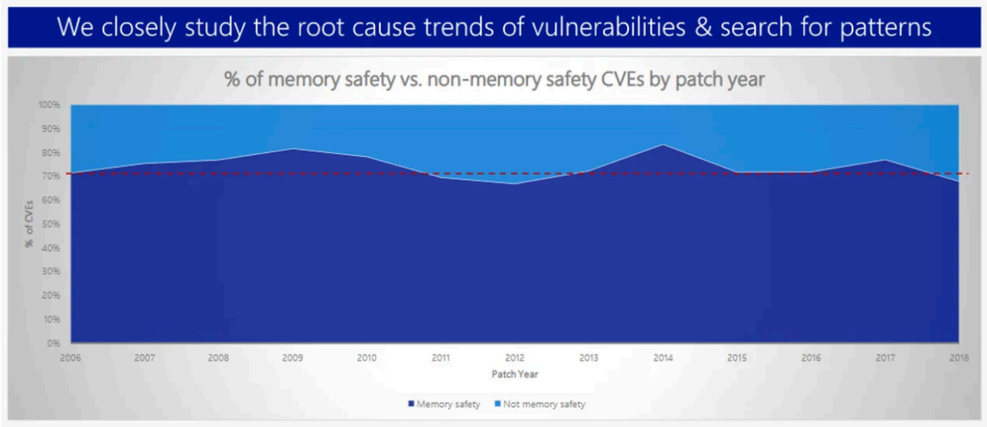


Image: Matt Miller

/ related

Worried about the Windows BitLocker recovery bug? 6 things you need to know

The Windows 10 clock is ticking: 5 ways to save your old PC in 2025 (most are free)

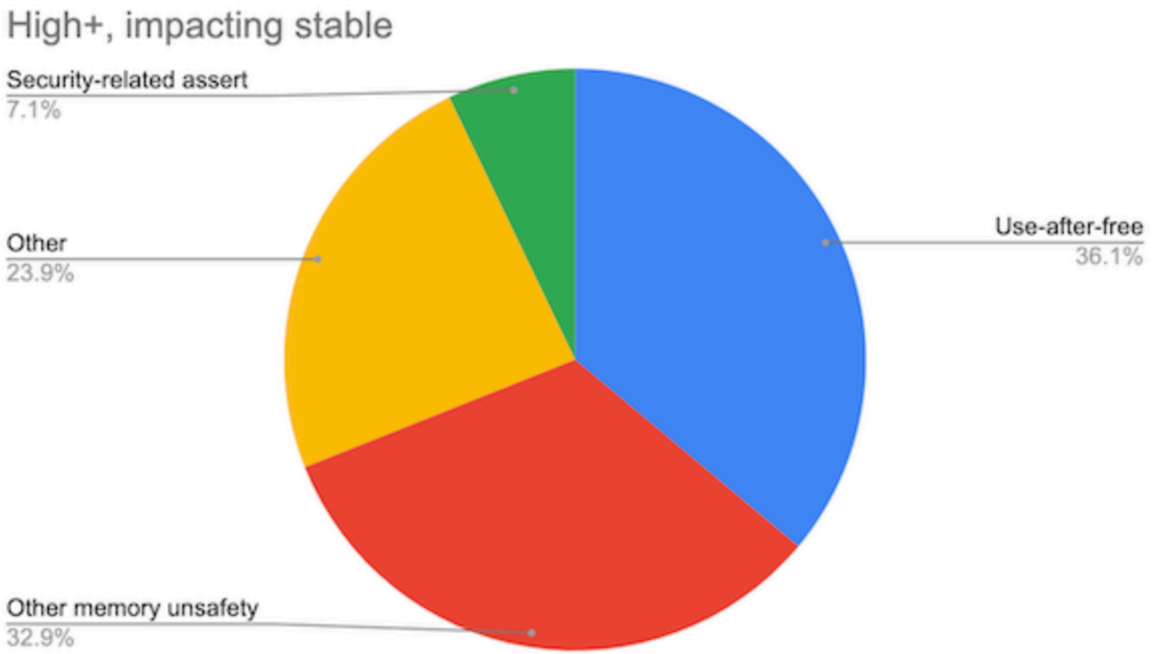
Memory Safety & Security

Memory safety

The Chromium project finds that around 70% of our serious security bugs are memory safety problems. Our next major project is to prevent such bugs at source.

The problem

Around 70% of our high severity security bugs are memory unsafety problems (that is, mistakes with C/C++ pointers). Half of those are use-after-free bugs.



Memory Safety & Security

The Case for Memory Safe Roadmaps

Why Both C-Suite Executives and Technical Experts Need to Take Memory Safe Coding Seriously

Publication: December 2023

United States Cybersecurity and Infrastructure Security Agency
United States National Security Agency
United States Federal Bureau of Investigation
Australian Signals Directorate's Australian Cyber Security Centre
Canadian Centre for Cyber Security
United Kingdom National Cyber Security Centre
New Zealand National Cyber Security Centre
Computer Emergency Response Team New Zealand

Memory Safety & Security

THE WHITE HOUSE

MENU

FEBRUARY 26, 2024

Press Release: Future Software Should Be Memory Safe

ONCD ▸ BRIEFING ROOM ▸ PRESS RELEASE

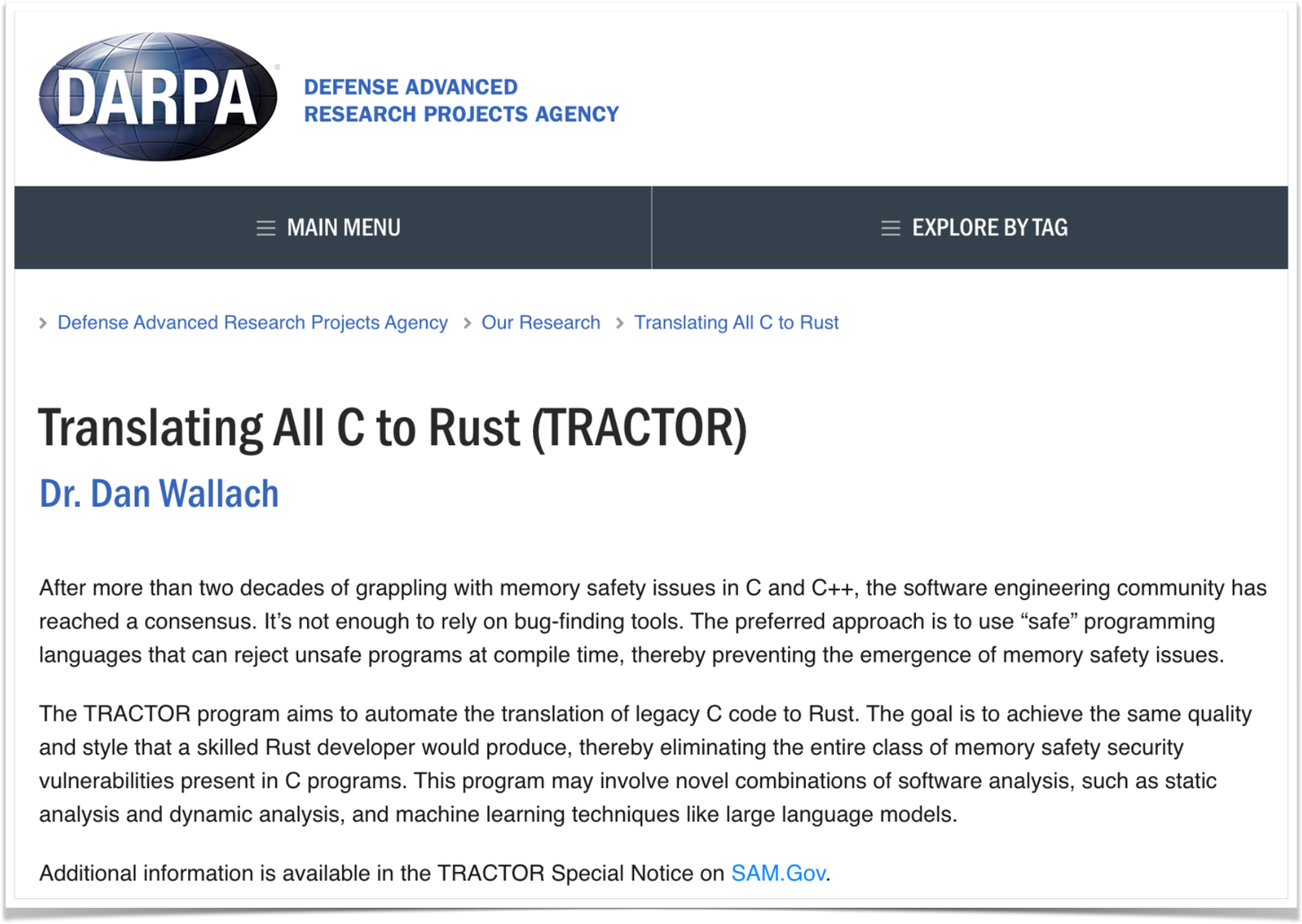
Leaders in Industry Support White House Call to Address Root Cause of Many of the Worst Cyber Attacks

Read the full report here

Memory Safety & Security

- OCaml is a type-safe language
  - Type safety ⇒ Memory safety
- Java, Python, JavaScript, Rust are also memory safe
- OCaml's has a great ***dynamic range*** compared to these languages
  - From small scripts, to Web, to long running services, to compilers, to embedded systems (Unikernels), to …
- OCaml vs Rust for memory-safety
  - OCaml has a GC (covers 95% of cases), for the rest use Rust
  - Pausetimes of < single-digit ms

Memory Safety & Security



DEFENSE ADVANCED
RESEARCH PROJECTS AGENCY

MAIN MENU | EXPLORE BY TAG

Defense Advanced Research Projects Agency › Our Research › Translating All C to Rust

Translating All C to Rust (TRACTOR)

Dr. Dan Wallach

After more than two decades of grappling with memory safety issues in C and C++, the software engineering community has reached a consensus. It's not enough to rely on bug-finding tools. The preferred approach is to use "safe" programming languages that can reject unsafe programs at compile time, thereby preventing the emergence of memory safety issues.

The TRACTOR program aims to automate the translation of legacy C code to Rust. The goal is to achieve the same quality and style that a skilled Rust developer would produce, thereby eliminating the entire class of memory safety security vulnerabilities present in C programs. This program may involve novel combinations of software analysis, such as static analysis and dynamic analysis, and machine learning techniques like large language models.

Additional information is available in the TRACTOR Special Notice on SAM.Gov.

Memory Safety & Security

Oxidizing OCaml with Modal Memory Management

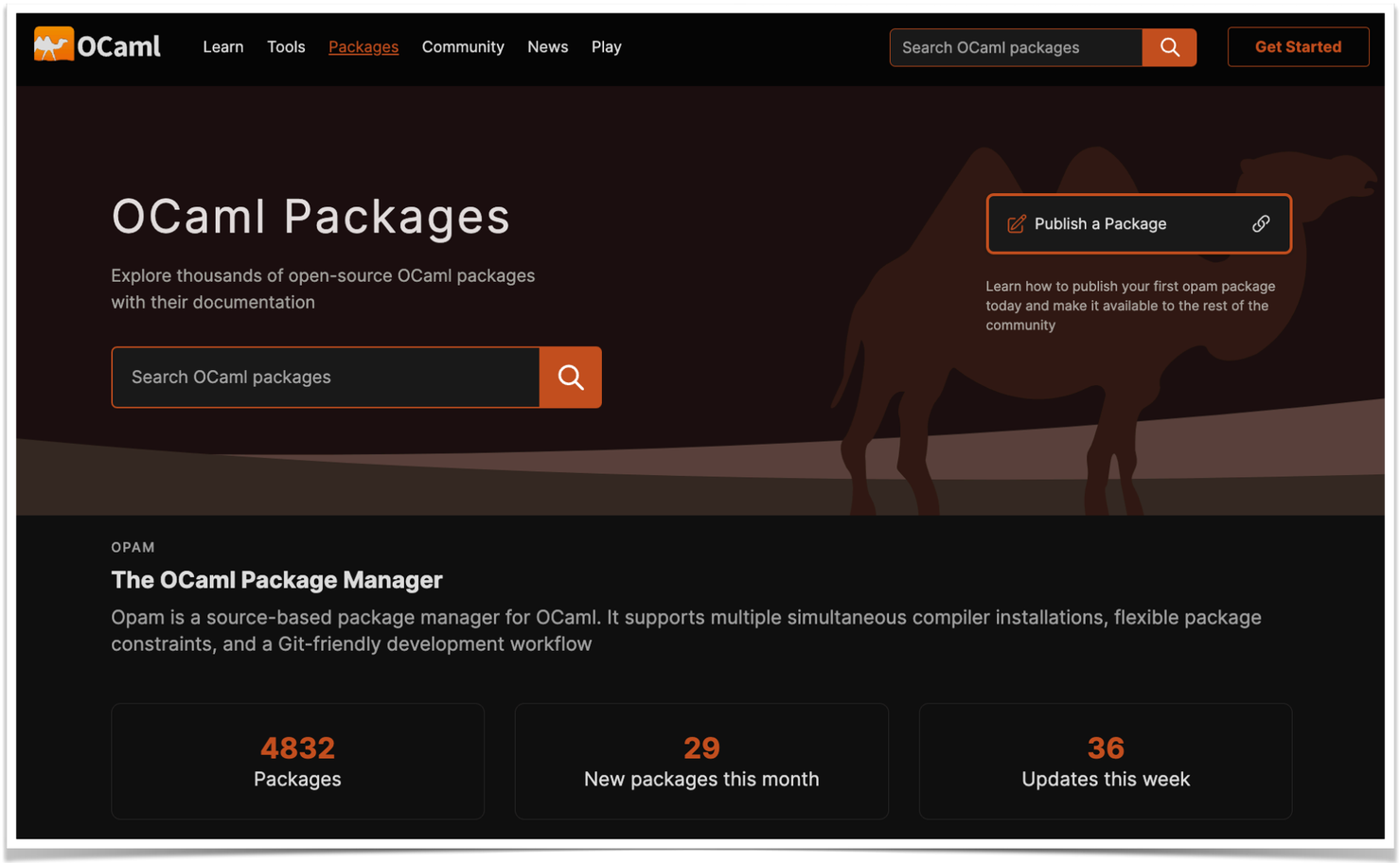
ANTON LORENZEN, The University of Edinburgh, UK
LEO WHITE, Jane Street, UK
STEPHEN DOLAN, Jane Street, UK
RICHARD A. EISENBERG, Jane Street, USA
SAM LINDLEY, The University of Edinburgh, UK

Programmers can often improve the performance of their programs by reducing heap allocations: either by allocating on the stack or reusing existing memory in-place. However, without safety guarantees, these optimizations can easily lead to use-after-free errors and even type unsoundness. In this paper, we present a design based on *modes* which allows programmers to safely reduce allocations by using stack allocation and in-place updates of immutable structures. We focus on three mode axes: affinity, uniqueness and locality. Modes are fully backwards compatible with existing OCaml code and can be completely inferred. Our work makes manual memory management in OCaml safe and convenient and charts a path towards bringing the benefits of Rust to OCaml.

Why *not* 

in 2024?

Ecosystem is small



… but very high quality and growing

Community is small

Industry

FACEBOOK Tarides Bloomberg

docker ahrefs SimCorp

Microsoft Tezos Jane Street

Projects

hack flow

COMPCERT

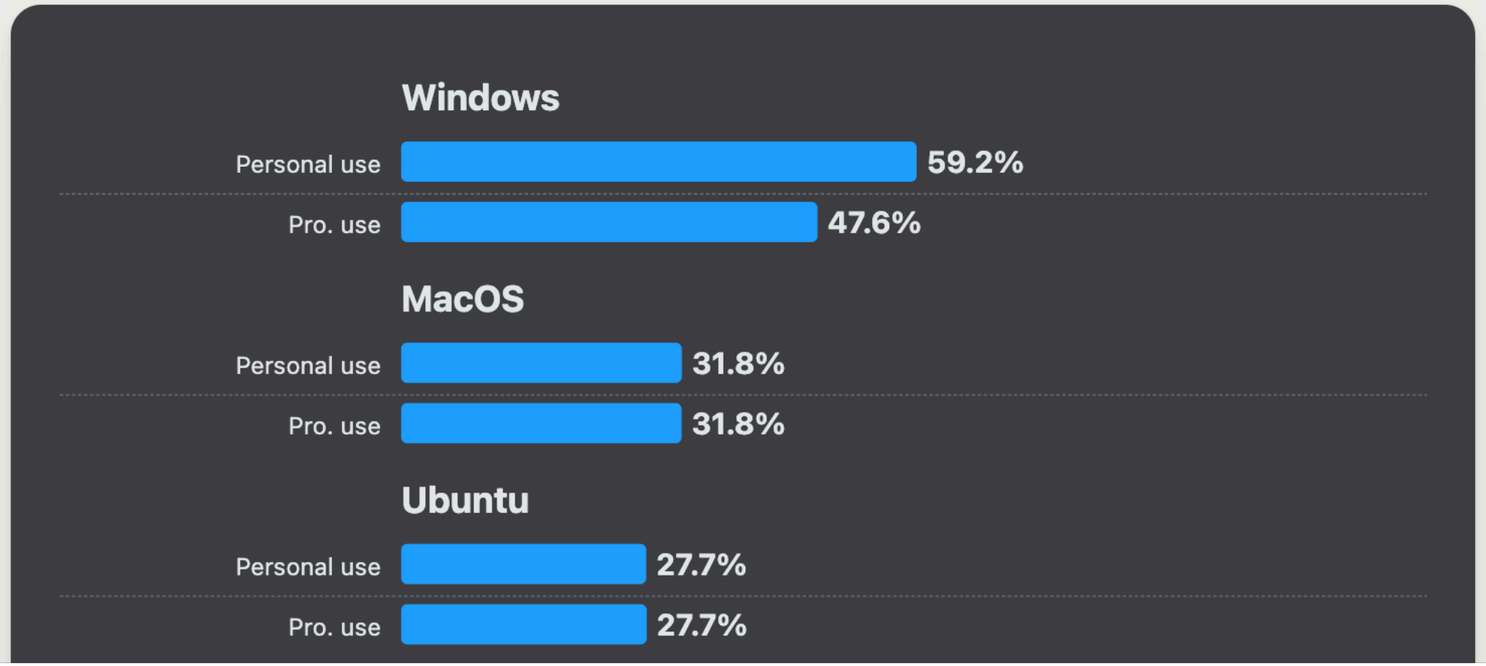
… with some high profile users

Windows is a second-class citizen

Operating system

Windows is the most popular operating system for developers, across both personal and professional use.

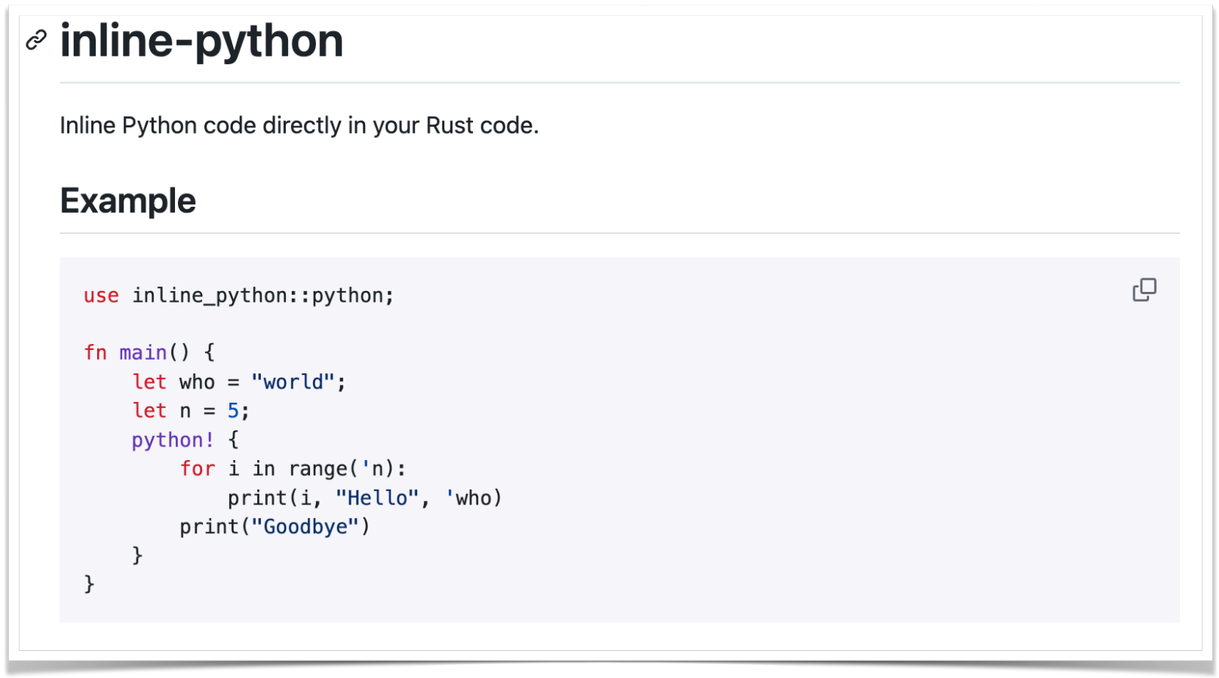
What is the primary **operating system** in which you work?



Windows is a second-class citizen

Interop remains challenging

- True for most languages, but necessary for OCaml adoption
- OCaml to C works best today
  - Wasm, JS, Python + respective build systems remains a challenge



inline-python

Inline Python code directly in your Rust code.

Example

```rust
use inline_python::python;

fn main() {
    let who = "world";
    let n = 5;
    python! {
        for i in range('n):
            print(i, "Hello", 'who)
        print("Goodbye")
    }
}
```

Tarides

Make OCaml succeed by playing well with others

Fin