Rust advantage - static checking to prevent errors
Rust wants memory safety with "zero-cost" abstraction
ownership + borrowing

- static
- prevent modification of shared state
- attacking races
- immutable - state where thing has shared access
- data race = shared data + immutable

functional programming - no mutability
Rust: either shared data or mutability

- reduce the overhead of having a garbage collector

Also:

- runtime bounds checking, panic if out of bounds
- no pointer arithmetic
- no free/del
- no null ptrs

problems attacked:

- use after free
- double free
- buffer overrun
- data races
- mnull pointer dereference

Failure Handling (Rust)

- panic
    - prints backtrace
    - unwinds stack, dropping as it goes
    - exits
- enum

Cargo build system:

- dependencies
- build
- test benchmarks
- documentation

Problem: subscript checking