a systems language

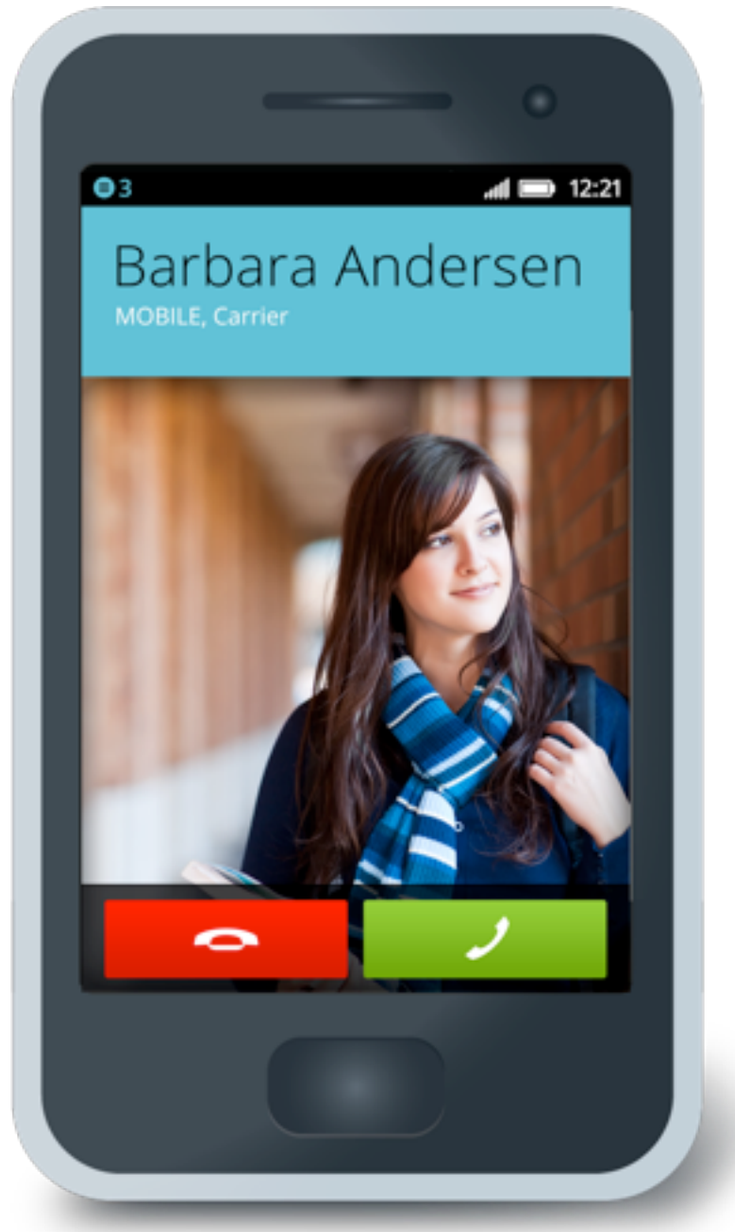
pursuing the trifecta

safe, concurrent, fast


FOSDEM
Firefox
Don't work for the man. Work for mankind.
mozilla.org/opportunities
mozilla


3
12:21
I LIKE EICH


3
12:21
Gallery
Camera
Contacts
Calendar
Marketplace
Clock
Videos
Music
FM Radio
Calculator
Settings
Data Usage

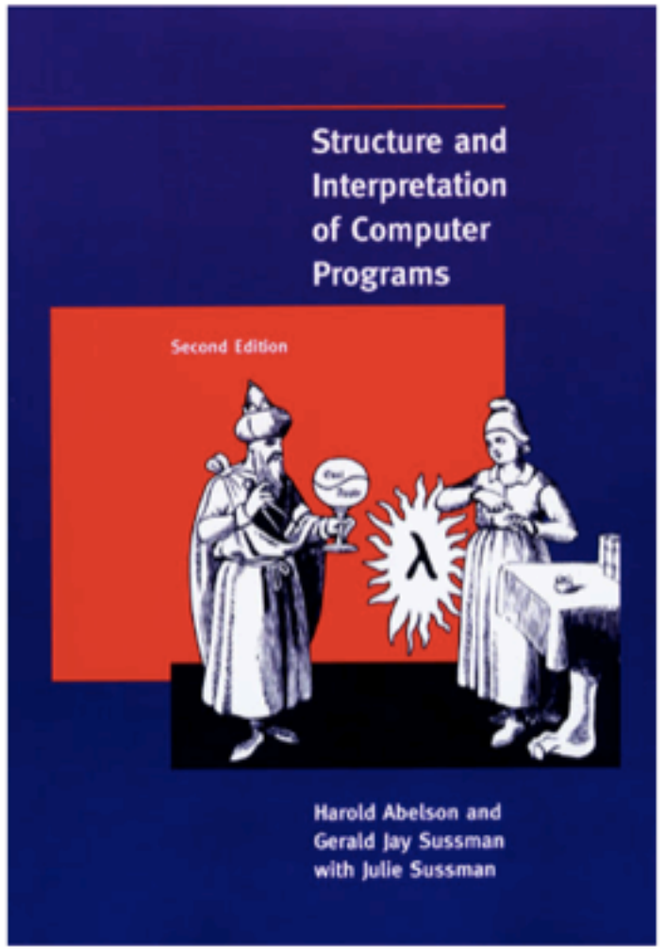

3
12:21
Barbara Andersen
MOBILE, Carrier

Structure and Interpretation of Computer Programs

Mozilla —... | Twitter / ... | Facebook | Gmail – In... | Rust | mozilla/r... | Home – St... | Structu...

mitpress.mit.edu/sicp/full-text/book/book.html

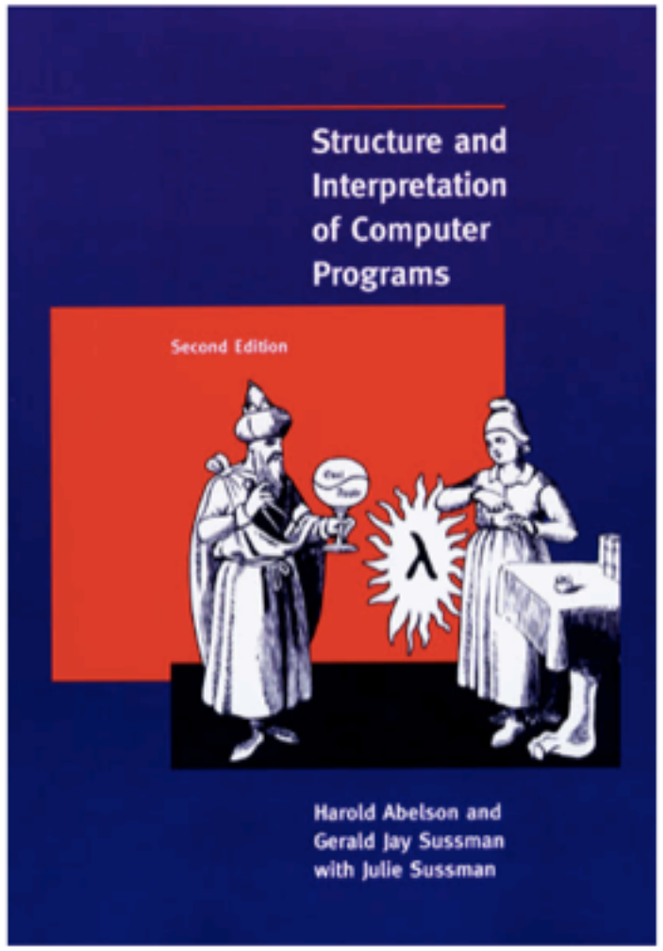
sicp

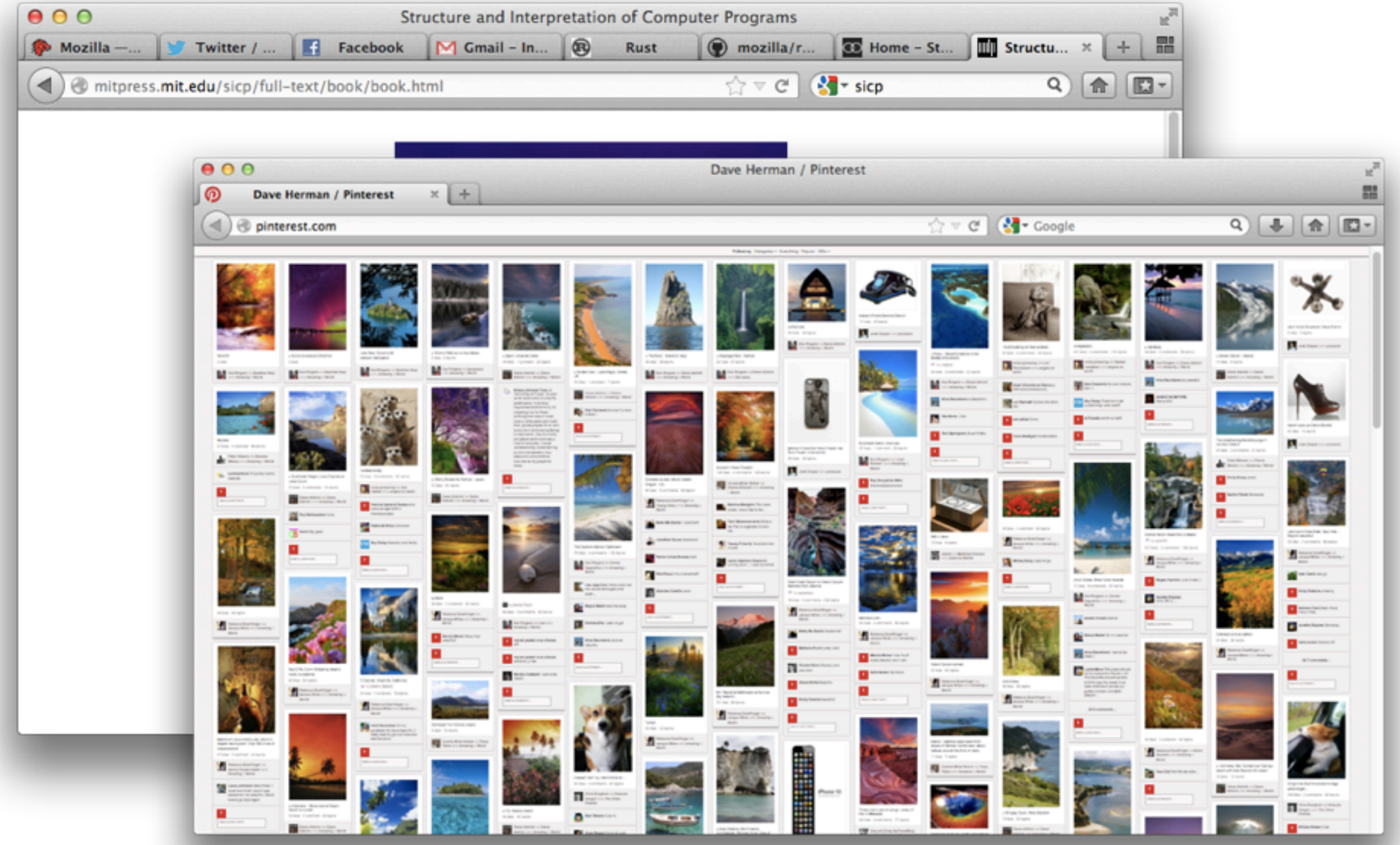


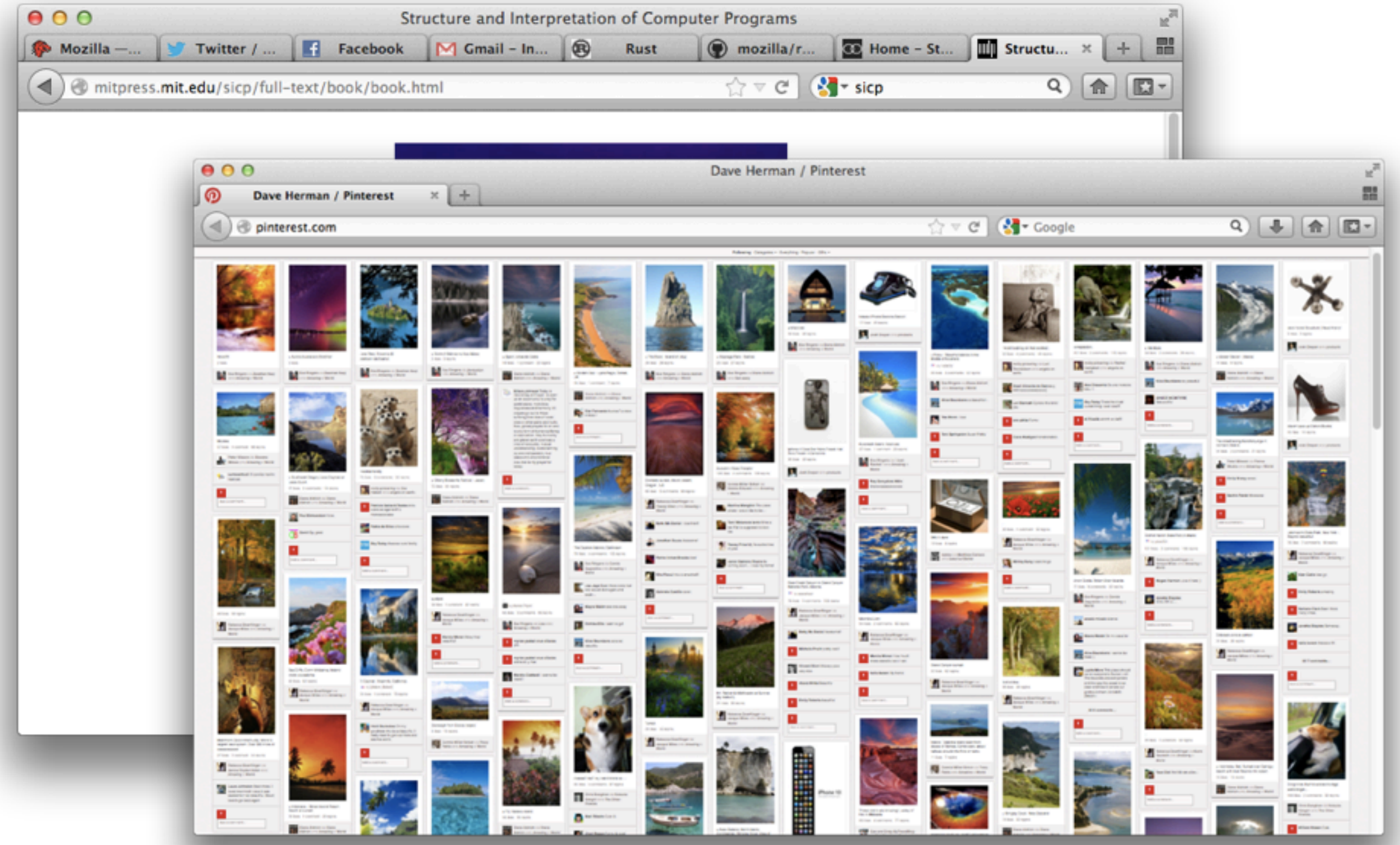
Structure and Interpretation of Computer Programs

Mozilla —... | Twitter / ... | Facebook | Gmail – In... | Rust | mozilla/r... | Home – St... | Structu...

mitpress.mit.edu/sicp/full-text/book/book.html




Structure and Interpretation of Computer Programs
Mozilla —...
Twitter / ...
Facebook
Gmail - In...
Rust
mozilla/r...
Home - St...
Structu...
mitpress.mit.edu/sicp/full-text/book/book.html
sicp
Dave Herman / Pinterest
Dave Herman / Pinterest
pinterest.com
Google



github.com/mozilla/servo

```
$ hg clone http://hg.mozilla.org/mozilla-central
...
66238 files updated, 0 files merged, 0 files removed,
0 files unresolved
$ cd mozilla-central
$ find . -name '*.c' -or -name '*.cpp' -or -name
'*.h' -or -name '*.tbl' | xargs wc -l | fgrep total |
awk '{total = total + $1}END{print total}'
```

5433119

2009: Graydon starts full-time work on Rust

2010: Team begins to grow

2011: Self-hosting via LLVM

2012: 0.1, 0.2, 0.3, 0.4 (soon), and beyond...

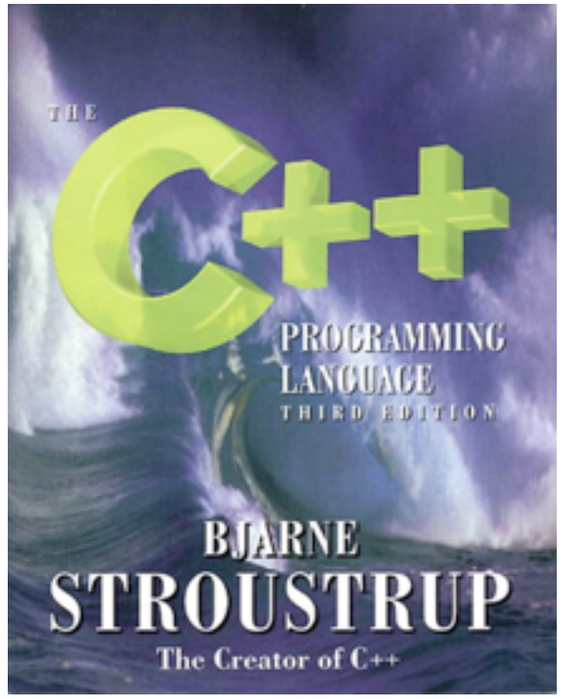
The Rust Team

Brian Anderson • Tim Chevalier •
Graydon Hoare • Niko Matsakis •
Patrick Walton

Interns and Alumni

Michael Bebenita • Ben Blum • Rafael Espíndola •
Roy Frostig • Marijn Haverbeke • Eric Holk •
Lindsey Kuper • Elliott Slaughter • Paul Stansifer •
Michael Sullivan

- stack allocation
- data ownership
- monomorphisation and inlining

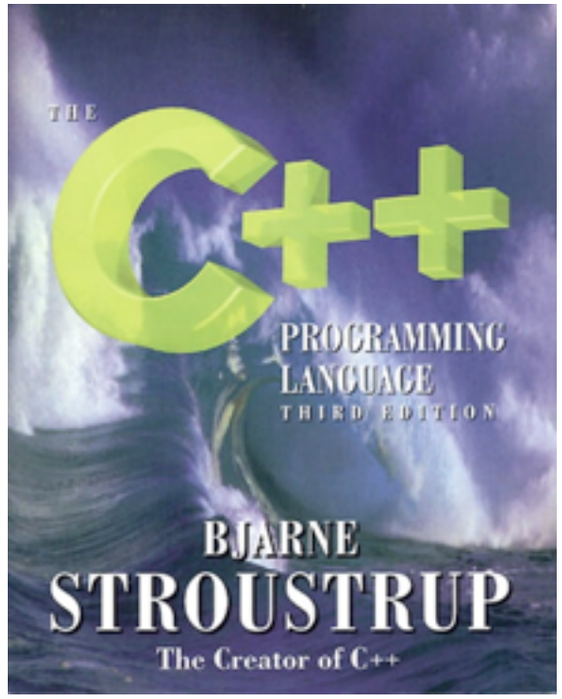


ERLANG

- type safety
- pattern matching
- type classes
- no **`null`**

- actors
- message-passing
- failure

PERFORMANCE



CONCURRENCY

TYPES


FAST

"Sometimes, cleaning up your code makes it slower even when it shouldn't."

— Robert O'Callahan

"Abstraction Penalties, Stack Allocation and Ownership Types"

http://j.mp/abstraction-penalties

```
let vec = [1, 2, 3];

vec.each(|item| {
    print(fmt!("%d ", item));
    true
});
```

```
let vec = [1, 2, 3];

for vec.each |item| {
    print(fmt!("%d ", item));
}
```

```
fn from_origin(x: float, y: float)
    -> float {
  let x0 = 0.0,
      y0 = 0.0;

  dist(x, y, x0, y0)
}
```

```
struct Point {
    x: float,
    y: float
}
```

```
fn from_origin(p: Point) -> float {
  let origin = Point {
    x: 0.0,
    y: 0.0
  };
  dist(p, origin)
}
```

```
fn print_point(p: &Point) {
   match *p {
      Point {x,y} =>
         println(fmt!("(%f, %f)", x, y))
   }
}
```

```
fn print_point(p: &Point) {
   match *p {
      Point {x,y} =>
         println(fmt!("(%f, %f)", x, y))
   }
}
```

```
fn f() {
   let p = Point { ... };
   print_point(&p);
}
```

```
let p = @Point { ... };
print_point(p);
```

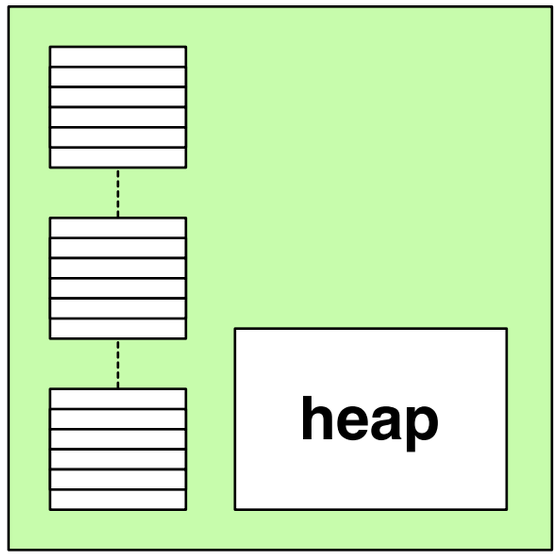
```
let p = ~Point { ... };

print_point(p);
```

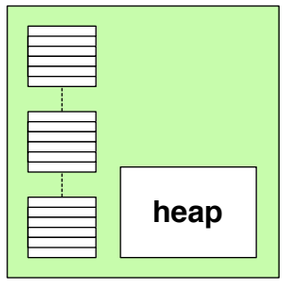
```
let p = ~Point { ... };
box.topLeft = move p; // deinitialized

print_point(p);       // error
```

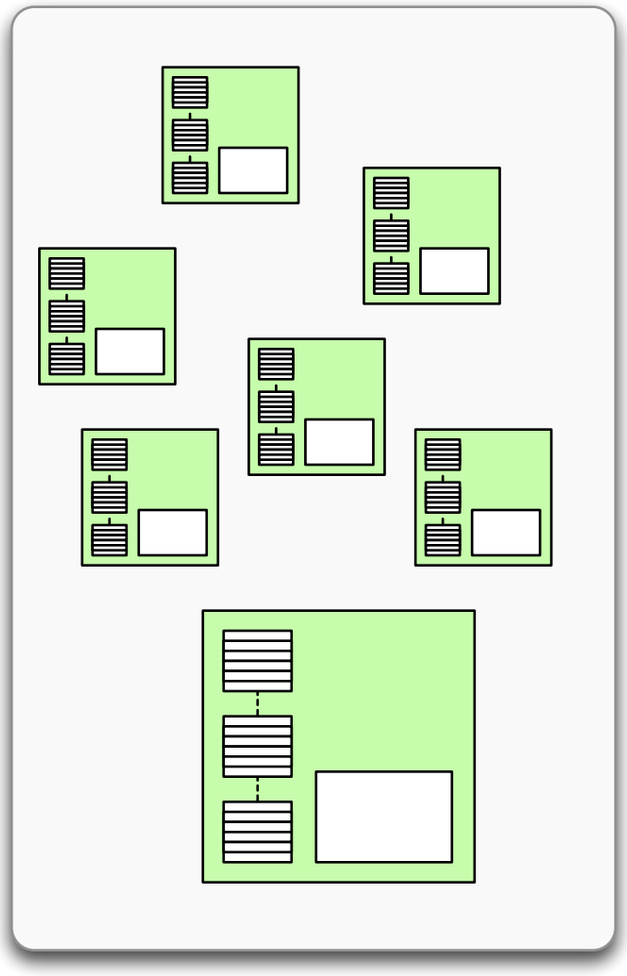
```
let p;                  // uninitialized
print_point(p); // error

p = ~Point { ... };
```

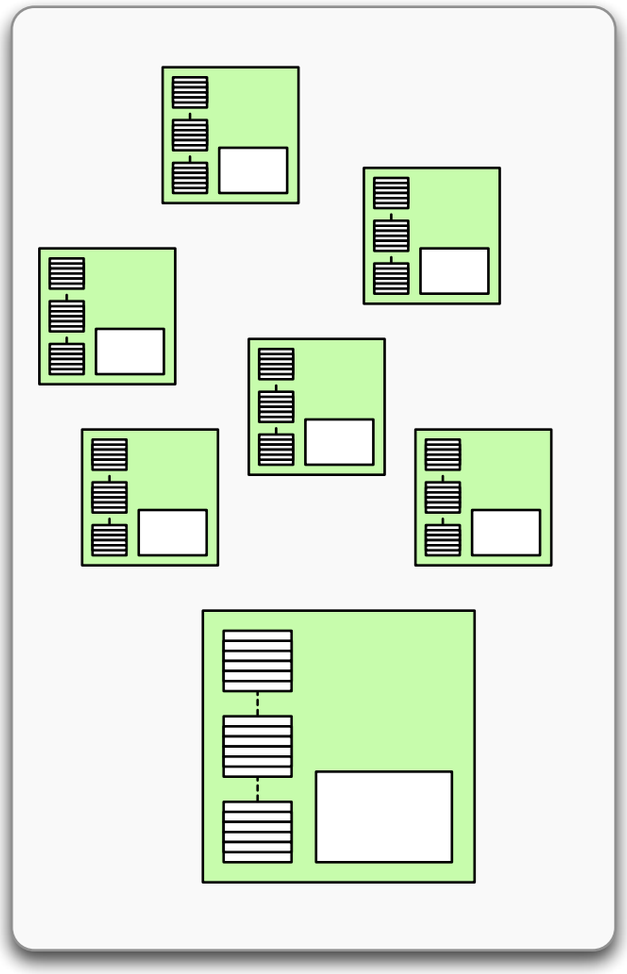
Rust	C++
&T	T&
@T	shared_ptr<T>
~T	unique_ptr<T>

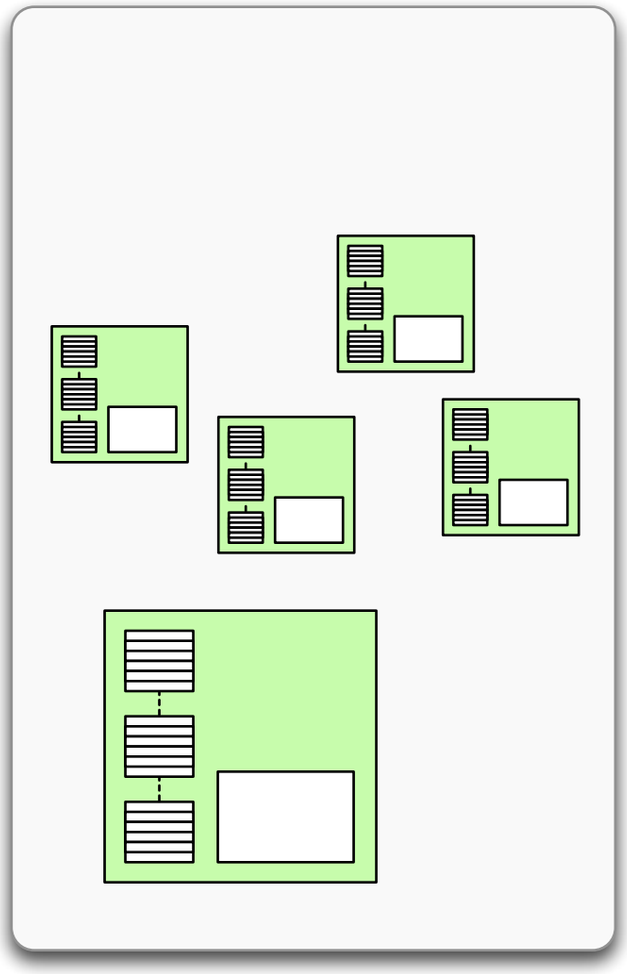
CONCURRENT

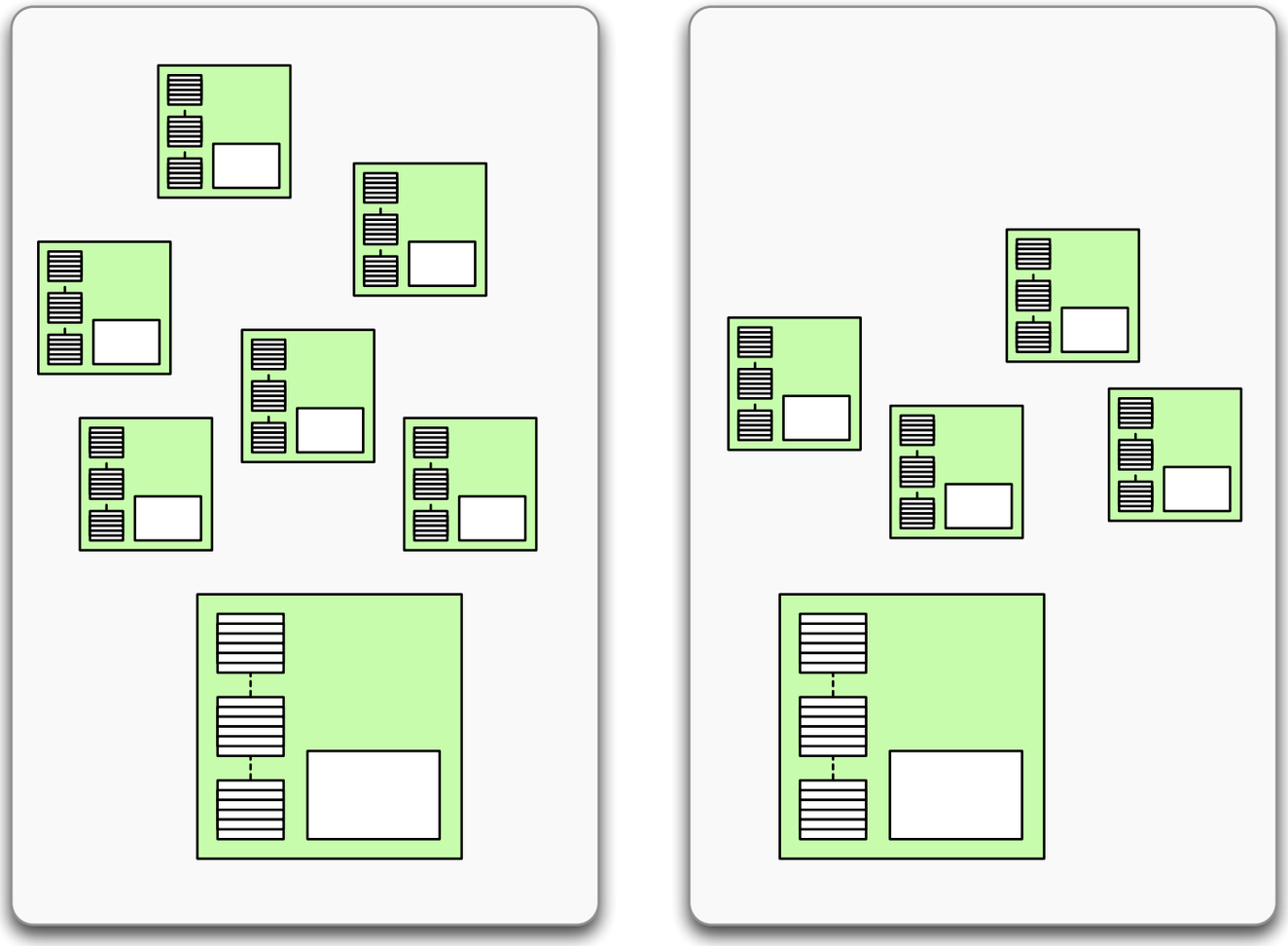

heap


heap

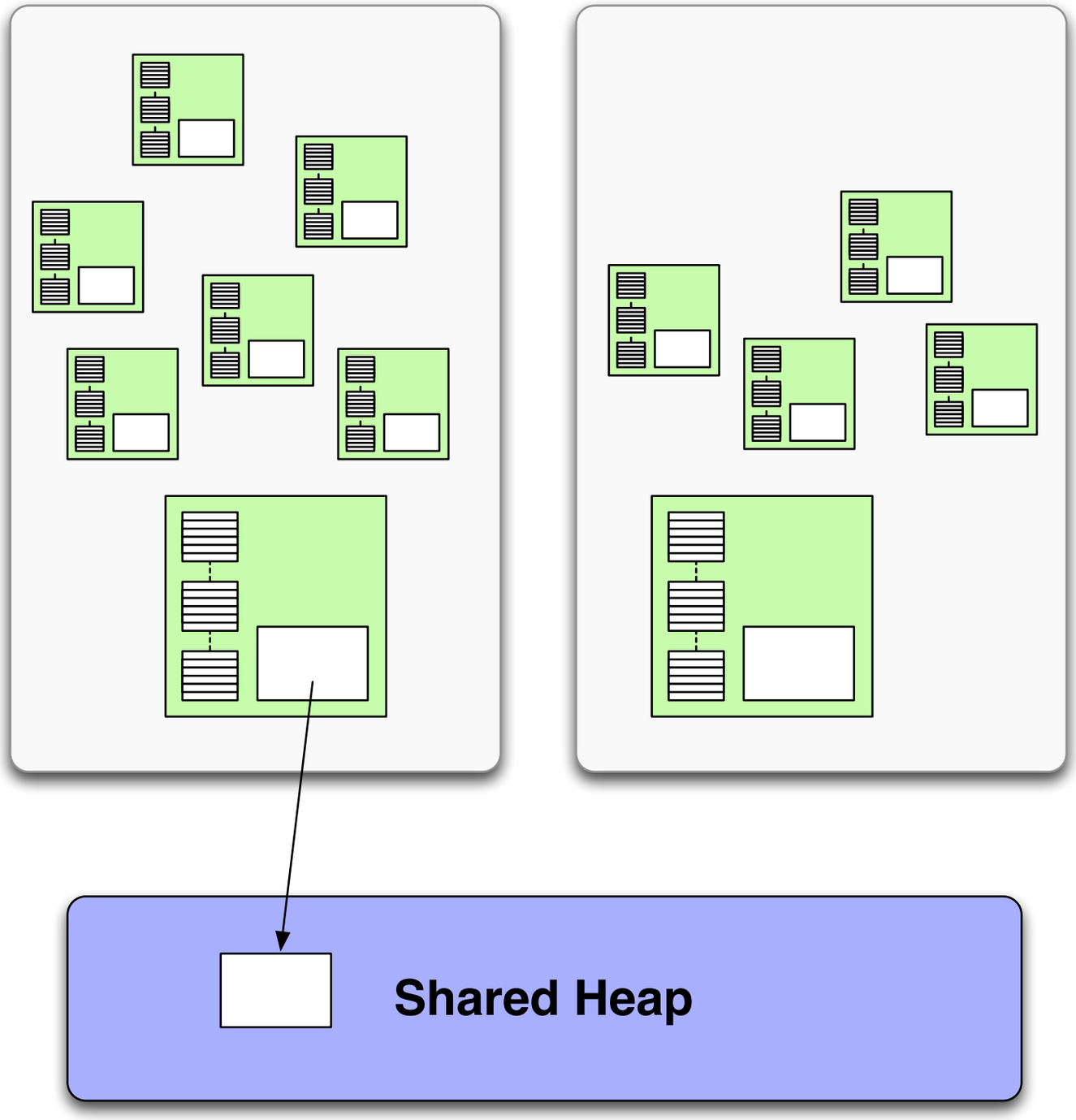


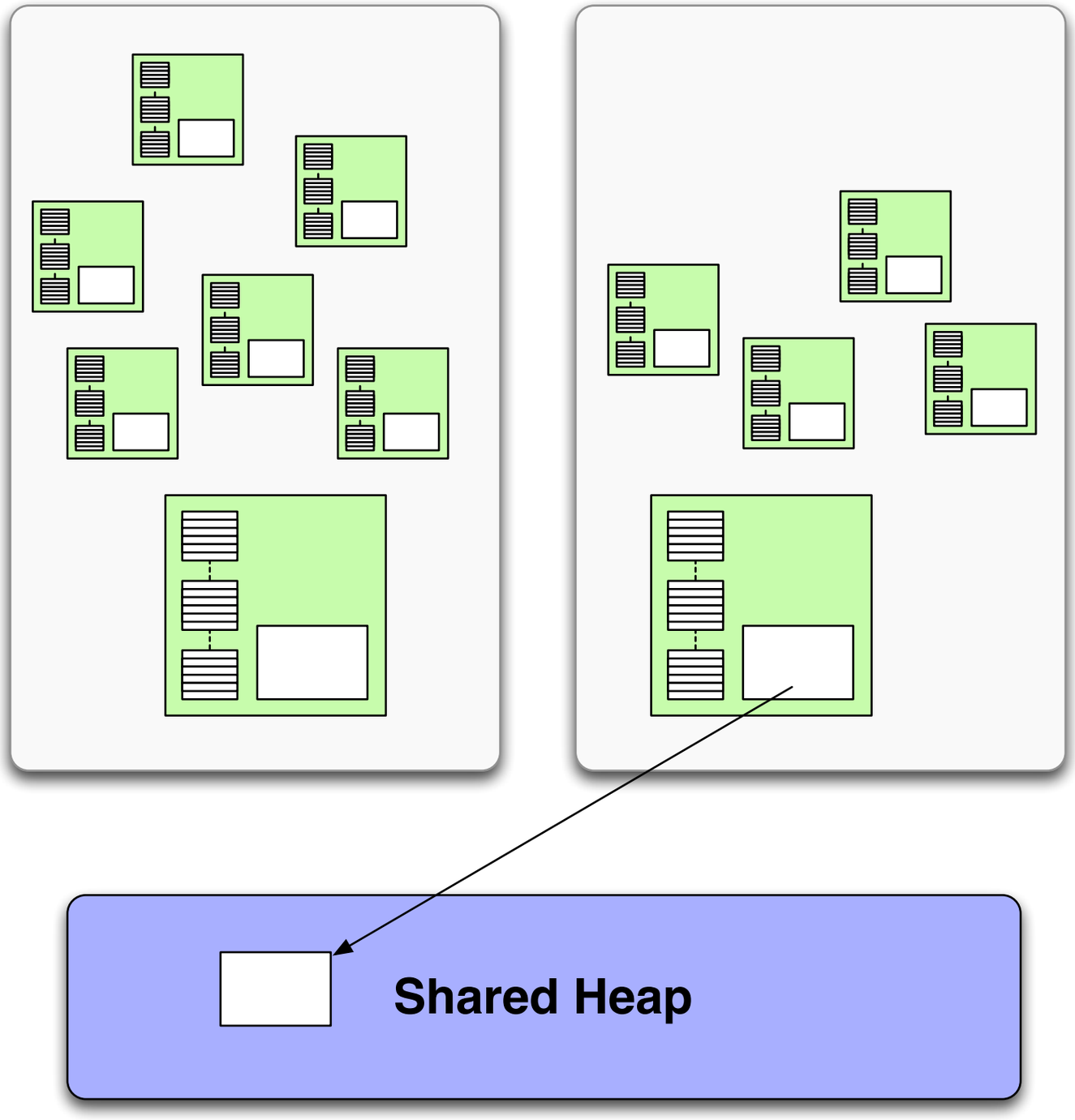






Shared Heap


Shared Heap


Shared Heap

```
let (ch, port) = pipes::stream();

do spawn |move ch| {
  let s = ~Point { x: 1.0, y: 2.0 };
  ch.send(s);
}

let s = port.recv();
assert s.x == 1.0;
assert s.y == 2.0;
```


SAFE

```rust
fn is_empty<T>(v: &[T]) -> bool {
    v.len() == 0
}
```

```
struct Monster {
  name: &str,
  mut health: int
}
```

```
struct Player {
  mut health: int
}
```

```
impl Monster {
  fn attack(&self, player: &Player) {
    player.health -= 10;
  }
}

monster.attack(&player);
```

```
trait ToJSON {
    fn to_json(&self) -> ~str;
}
```

```
impl Monster : ToJSON {
   fn to_json(&self) -> ~str {
      fmt!(...)
   }
}
```

```
impl Player : ToJSON {
   fn to_json(&self) -> ~str {
      fmt!(...)
   }
}
```

```
fn save<T:ToJSON>(x: &T, file: &str) {
  let writer = file_writer(...).get();
  writer.write(x.to_json());
}
```

```
save(&player, "p.json");
save(&monster, "m.json");
```

```
struct<T:Send> Chan<T> { ... }

impl<T:Send> Chan<T> {
  fn send(&self, x: T) { ... }
}
```

```
struct ARC<T:Const Send> { ... }
```

```
let img: Image = load_image();
let handle0: ARC<Image> = ARC(move img);

for N.times {
    let handle = handle0.clone();
    do spawn |move handle| {
        display_image(handle);
    }
}
```

Modula-3

```
unsafe module
```

Java

```
native
sun.misc.Unsafe
```

Ocaml

```
Obj.magic
```

Haskell

```
unsafePerformIO
```

Rust

```
unsafe { ... }
```

regions, region subtyping & polymorphism, arenas

traits as existentials

mutability tracking, purity, and borrow checking

freezing/thawing data structures

task-local storage

linked failure

one-shot closures

macros

Good reads

smallcultfollowing.com/babysteps

pcwalton.github.com

Join us

rust-lang.org

rust-dev@mozilla.org

irc.mozilla.org :: #rust

Thank you.

Image credits

Sean Martell
http://blog.seanmartell.com

Martyn
http://www.flickr.com/photos/martyn/438111857/in/set-102261

Ian Kobylanski
http://www.flickr.com/photos/iankobylanski/6151659680

Sudhir Naik
http://www.flickr.com/photos/sudhirnaik/4890017884/

Thomas Gibbard
http://www.flickr.com/photos/22305783@N06/3947478553