Rust Ownership

Daniel Sánchez Domínguez

Lifetime

Let people take pointers to random data on the stack, and the result is a pervasive unmanageable un-safety. It allows to holding a pointer to something that:

- went out of scope
- got mutated away

A lifetime is the name of a region (~block/scope) of code somewhere in a program.

```
// Only one reference in input, so the output must be derived
// from that input
fn foo(&A) -> &B; // sugar for:
fn foo<'a>(&'a A) -> &'a B;

// Many inputs, assume they're all independent
fn foo(&A, &B, &C); // sugar for:
fn foo<'a, 'b, 'c>(&'a A, &'b B, &'c C);

// Methods, assume all output lifetimes are derived from `self`
fn foo(&self, &B, &C) -> &D; // sugar for:
fn foo<'a, 'b, 'c>(&'a self, &'b B, &'c C) -> &'a D;
```

In practical terms, `fn foo<'a>(&'a A) -> &'a B` means that the input must live at least as long as the output. If you keep the output around for a long time, this will expand the region that the input must be valid for.