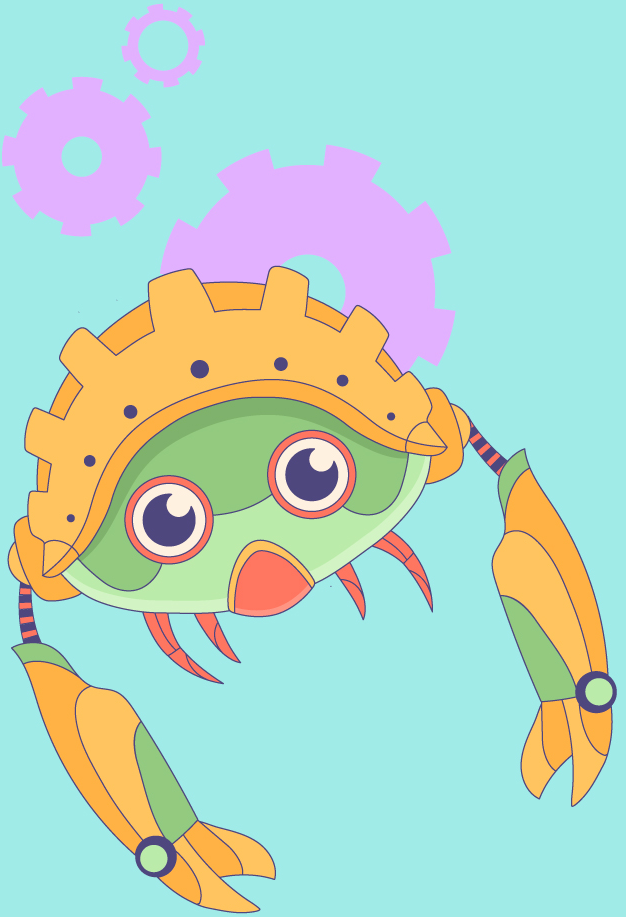
Alice Ryhl

Software Engineer at Google

Making Rust less special



Who am I?

- Working on the Android Rust team at Google.
- Rust in the Linux Kernel.
- Maintainer of Tokio.

Rust Project goals for 2024

Aug. 12, 2024 · Niko Matsakis on behalf of Leadership Council

With the merging of RFC #3672, the Rust project has selected a **slate of 26 Project Goals** for the second half of 2024 (2024H2). This is our first time running an experimental new roadmapping process; assuming all goes well, we expect to be running the process roughly every six months. Of these goals, we have designated three of them as our **flagship goals**, representing our most ambitious and most impactful efforts: (1) finalize preparations for the Rust 2024 edition; (2) bring the Async Rust experience closer to parity with sync Rust; and (3) resolve the biggest blockers to the Linux kernel building on stable Rust. As the year progresses we'll be posting regular updates on these 3 flagship goals along with the 23 others.

What does it mean to be special?



What does it mean to be special?

- You are visitor number X

What if two people visit your website at exactly the same time?



You are visitor number X

```
public class CounterServlet extends HttpServlet {
    private int count = 0;

    @Override
    protected void doGet(HttpServletRequest req,
                         HttpServletResponse resp)
    throws IOException
    {
        count++;
        resp.getWriter().println("You are visitor number: " + count);
    }
}
```

You are visitor number X

```
public class CounterServlet extends HttpServlet {
    private int count = 0;

    @Override
    protected void doGet(HttpServletRequest req,
                         HttpServletResponse resp)
    throws IOException
    {
        int currCounter = count;
        count = currCount + 1;
        resp.getWriter().println("You are visitor number: " + count);
    }
}
```

What does it mean to be special?

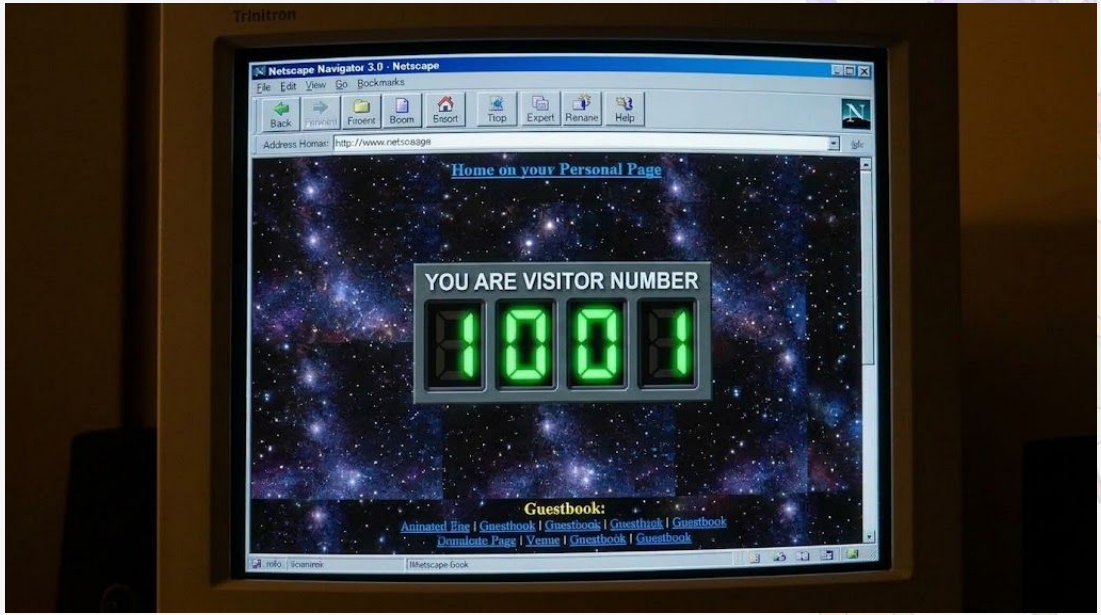
- You are visitor number 1000





What does it mean to be special?

- You are visitor number 1002



What does it mean to be special?

You are visitor number X

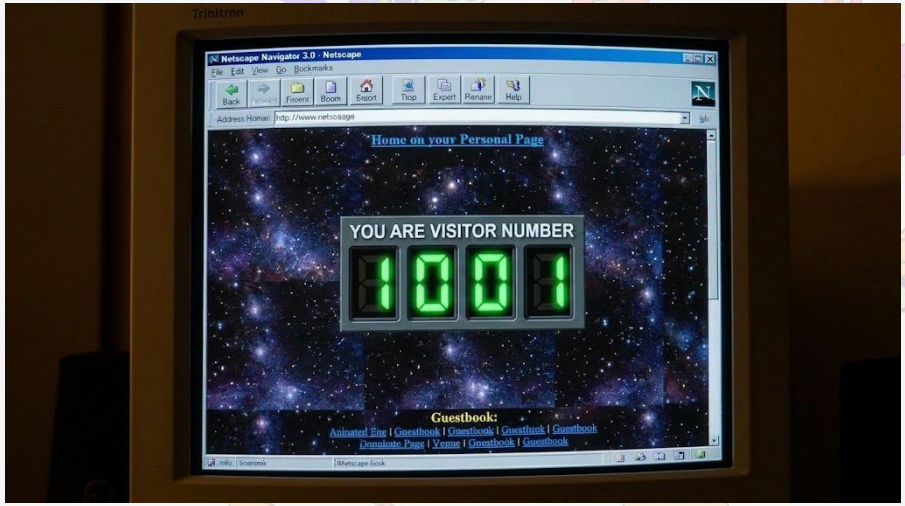
```
public class CounterServlet extends HttpServlet {
    private int count = 0;

    @Override
    protected synchronized void doGet(HttpServletRequest req,
                                      HttpServletResponse resp)
    throws IOException
    {
        count++;
        resp.getWriter().println("You are visitor number: " + count);
    }
}
```

What does it mean to be special?

- You are visitor number 1000





Java's `synchronized` is special



What does it mean to be special?

- You are visitor number X

What about the F5 masher?

New visitor:

- Read and write counter.

Returning visitor:

- Read counter

What does it mean to be special?

- You are visitor number X

What about the F5 masher?

New visitor:

- Read and write counter.

Returning visitor:

- Read counter
- Returning visitors can run in parallel!

What about the F5 masher?

```
public class VisitorCounter {
    private int count = 0;
    private ReadWriteLock rwLock = new ReentrantReadWriteLock();

    // 1. Returning Visitor (has 'visited=true' cookie): Read-Only
    // Multiple returning visitors can execute this simultaneously.
    public int getCountForReturningVisitor() {
        rwLock.readLock().lock();
        try {
            return count;
        } finally {
            rwLock.readLock().unlock();
        }
    }
```

What about the F5 masher?

```
    // 2. New Visitor (no cookie): Read + Write
    // Exclusive access: blocks all other new AND returning visitors
    public int incrementAndGetForNewVisitor() {
        rwLock.writeLock().lock();
        try {
            count++;
            return count;
        } finally {
            rwLock.writeLock().unlock();
        }
    }
}
```

What does it mean to be special?

- Visitor counters in Rust

The Rust Mutex is not special.

```
struct SharedData {
    immutable: String,
    mutable: Mutex<MutableData>,
}

struct MutableData {
    counter: u32,
}
```

Visitor counters in Rust

```
// 2. New Visitor: Increments and returns the counter
async fn increment_count(State(state): State<Arc<SharedData>>) -> String {
    let count = {
        let mut guard = state.mutable.lock().unwrap();
        guard.counter += 1;
        guard.counter
    };

    if count == 1000 {
        format!("🎉 CONGRATULATIONS! You are the 1000th visitor! 🎉")
    } else {
        format!("{}: You are visitor number {}", state.immutable, count)
    }
}
```

Visitor counters in Rust

The Rust Mutex is not special. Therefore:

1. The Linux Kernel could write its own Mutex type, backed by the C ‘struct mutex’ type.
2. Your custom Mutex can have additional features: RwLock

The Rust standard library

core	alloc	std
Contains logic that needs no special support.	Contains logic that needs memory allocation.	Contains logic that needs an operating system.

The Rust standard library

core

Contains logic that needs no special support.

alloc

Contains logic that needs memory allocation.

std

Co…nat
nee…ing
Mutex
system.

The Rust standard library in the Linux Kernel

core

Contains logic that needs no special support.

alloc

Contains logic that needs memory allocation.



The Rust standard library in the Linux Kernel

core	alloc	kernel
Contains logic that needs no special support.	Contains logic that needs memory allocation.	Custom logic written for the Linux Kernel specifically.

The Rust standard library in the Linux Kernel

core

Contains logic that needs no special support.

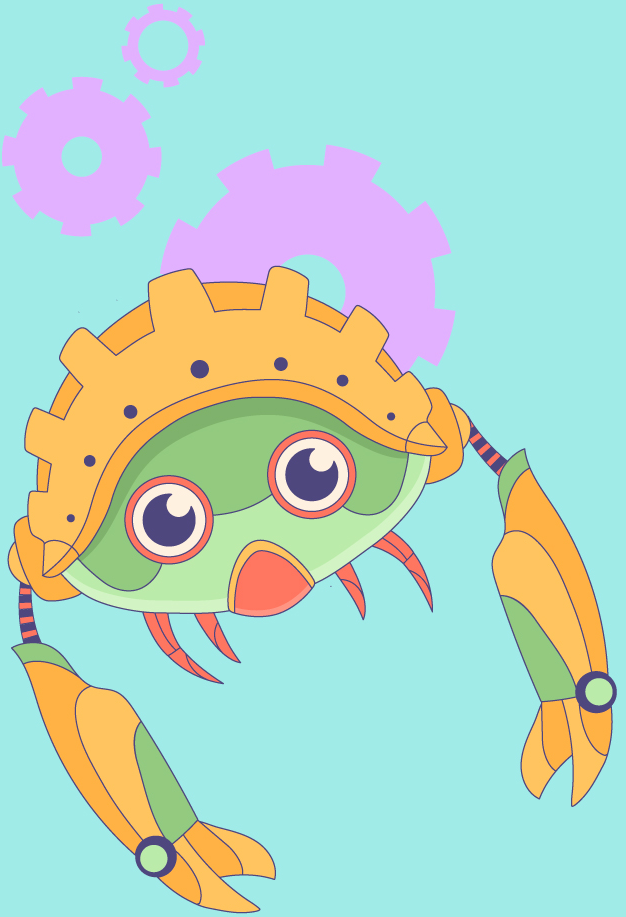
alloc

Contains logic that needs memory allocation.

kernel

writ Mutex nux Kernel specifically.

Unfinished Business



The Rust standard library in the Linux Kernel

core

Contains logic that needs no special support.

alloc

Contains logic that needs memory allocation.

~~Mutex~~

The Rust standard library in the Linux Kernel

core

Contains logic that needs no special support.

alloc

Co… …nat n… …y allocation.

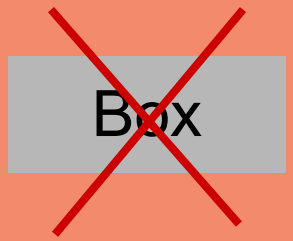
Box

Mutex

The Rust standard library in the Linux Kernel

core

Contains logic that needs no special support.





Box<T> pre-1.0

A history lesson.

&T

borrowed pointer

~T

owned pointer

@T

managed pointer

RFC: Remove `~` in favor of `box` and `Box` #59

Merged alexcrichton merged 1 commit into `rust-lang:master` from `pcwalton:remove-tilde` on May 2, 2014

Conversation 85 | Commits 1 | Checks 0 | Files changed 1



pcwalton commented on May 1, 2014 — Contributor

No description provided.

RFC: Remove `~` in favor of `box` and `Box` ff5f67a

Detailed design

The `~EXPR` production is removed from the language, and all such uses are converted into `box`.

Add a lang item, `box`. That lang item will be defined in `liballoc` (NB: not `libmetal` / `libmini`, for bare-metal programming) as follows:

```
#[lang="box"]
pub struct Box<T,A=Heap>(*T);
```

All parts of the compiler treat instances of `Box<T>` identically to the way it treats `~T` today.

The early days of Box<T>

Niko Matsakis (commit 35c261aea0d, March 30, 2015):

> "Add #[fundamental] annotations into libcore so that Sized and the Fn traits are considered fundamental, along with Box (though that is mostly for show; the real type is ~T in the compiler)."

Felix Klock (commit b4457fb8a21, PR #23859, March 31, 2015):

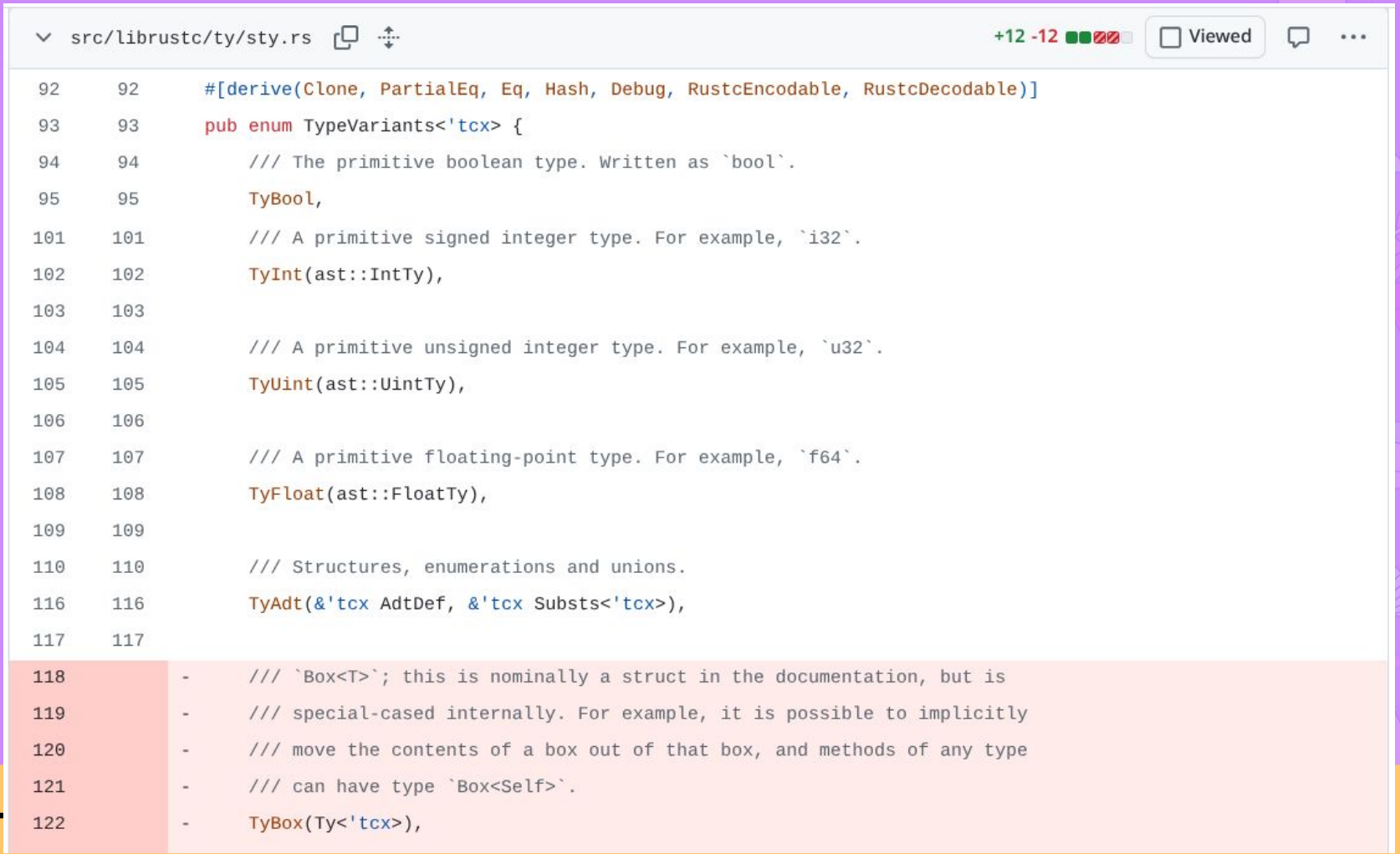
> "This is all of course a giant hack that will hopefully go away with an eventually removal of special treatment of Box<T> (aka ty_unique) from the compiler."

The early days of Box<T>

Since the compiler did not actually use the Box<T> struct, it used to be subtly wrong.

```
// December 2014
#[lang = "owned_box"]
pub struct Box<T>(Unique<T>);


// January 2015 (Rust 1.0)
#[lang = "owned_box"]
#[fundamental]
pub struct Box<T: ?Sized>(Unique<T>);
```


src/librustc/ty/sty.rs
+12 -12
Viewed
92 92 #[derive(Clone, PartialEq, Eq, Hash, Debug, RustcEncodable, RustcDecodable)]
93 93 pub enum TypeVariants<'tcx> {
94 94 /// The primitive boolean type. Written as `bool`.
95 95 TyBool,
101 101 /// A primitive signed integer type. For example, `i32`.
102 102 TyInt(ast::IntTy),
103 103
104 104 /// A primitive unsigned integer type. For example, `u32`.
105 105 TyUint(ast::UintTy),
106 106
107 107 /// A primitive floating-point type. For example, `f64`.
108 108 TyFloat(ast::FloatTy),
109 109
110 110 /// Structures, enumerations and unions.
116 116 TyAdt(&'tcx AdtDef, &'tcx Substs<'tcx>),
117 117
118 - /// `Box<T>`; this is nominally a struct in the documentation, but is
119 - /// special-cased internally. For example, it is possible to implicitly
120 - /// move the contents of a box out of that box, and methods of any type
121 - /// can have type `Box<Self>`.
122 - TyBox(Ty<'tcx>),

Unfinished Business

- Box<T> becomes a struct

However, other parts of Box<T> remain special.

```
unsafe impl<T: ?Sized> Drop for Box<T>
{
    fn drop(&mut self) {
        // FIXME: Do nothing, drop is
        // currently performed by
        // compiler.
    }
}
```

Unfinished Business

- Box<T> becomes a struct

However, other parts of Box<T> remain special.

```
impl<T> Box<T> {
    #[inline(always)]
    pub fn new(x: T) -> Box<T> {
        box x
    }
}
```

DerefMove

- Normal types: Deref and DerefMut
- You can “move out” of a Box<T>.
- Depends on Drop being compiler magic.

Moving out

Box has special capabilities to its fields: the ability to move out of them.

```
let mybox = Box::new(MyStruct {
    a: foo(),
    b: bar(),
});

let a = mybox.a; // move!

drop(mybox); // drops b, not a
```

Beyond the &

Metadata	
Short title	Beyond the `&`
What and why	Smart pointers (`Arc` , `Pin` , FFI wrappers) get the same ergonomics as `&` and `&mut` — reborrowing, field access, in-place init
Contact	Tyler Mandry
Task owners	Benno Lossin, Alice Ryhl, Aapo Alasuutari, Ding Xiang Fei, Niko Matsakis, Oli Scherer, Tyler Mandry, Yoshua Wuyts

Summary

Smart pointers that feel as natural as `&` and `&mut` .

Custom Allocators

If Box<T> is generic enough, maybe it doesn't matter that it's special?

As of Rust v1.100.0, the Box<T> type supports custom allocators.

Most custom Boxes can now be replaced with the stdlib Box, using a custom allocator.

Custom Allocators

```
let pool = MemoryPool::new(1000_000);

let mybox = Box::new_in(MyStruct {
    a: foo(),
    b: bar(),
}, &pool);

let a = mybox.a; // move!

drop(mybox); // drops b, not a
             // frees in MemoryPool
```

Box<dyn FnOnce>

Calling a function that implements FnOnce is a *move* operation.

Therefore, it doesn't work in a custom Box.

```
let value = ...;

let func = Box::new(|| {
    call_once(value);
});

func(); // <-- takes ownership
        // of the function.
```

- MyBox<dyn FnOnce>

With custom allocators, there's a trick that lets you call it anyway.

```
impl MyBox<T> {
  fn call<U>(self) -> U
  where
    T: FnOnce() -> U,
  {
    let fake_box = Box::from_raw(
      self.as_ptr(),
      NoopAllocator,
    );
    fake_box()
  }
}
```

- MyBox<dyn FnOnce>

With custom allocators, there's a trick that lets you call it anyway.

```
impl MyBox<T> {
  fn call<U>(self) -> U
  where
    T: FnOnce() -> U,
  {
    let fake_box = Box::from_raw(
      self.as_ptr(),
      NoopAllocator,
    );
    fake_box()
  }
}
```

- MyBox<dyn FnOnce>

With custom allocators, there's a trick that lets you call it anyway.

```
impl MyBox<T> {
  fn call<U>(self) -> U
  where
    T: FnOnce() -> U,
  {
    let fake_box = Box::from_raw(
      self.as_ptr(),
      NoopAllocator,
    );
    fake_box()
  }
}
```

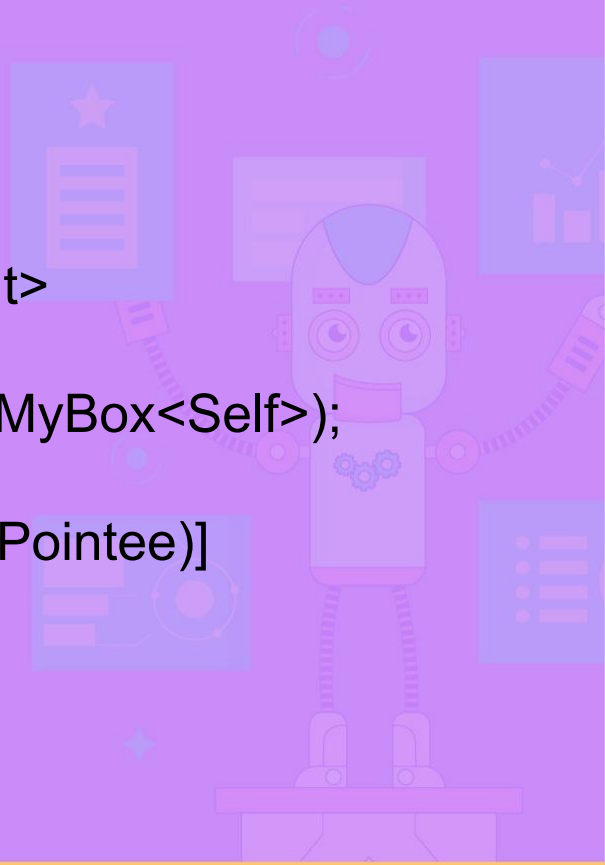
- MyBox<dyn FnOnce>

With custom allocators, there's a trick that lets you call it anyway.

```
impl MyBox<T> {
  fn call<U>(self) -> U
  where
    T: FnOnce() -> U,
  {
    let fake_box = Box::from_raw(
      self.as_ptr(),
      NoopAllocator,
    );
    fake_box()
  }
}
```

Trait objects and self types

- MyBox<dyn Trait>
- fn function(self: MyBox<Self>);
- #[derive(CoercePointee)]



Arbitrary Self Types

```
impl MyStruct {
    fn by_ref(&self);
    fn by_mut(&mut self);

    fn by_box(self: Box<Self>);

    fn by_custom_box(self: MyBox<Self>);
}
```

RFC: #[derive(SmartPointer)] #3621

Merged nikomatsakis merged 16 commits into `rust-lang:master` from `Darksonn:derive-smart-pointer` Jul 11, 2024

Conversation | Commits 16 | Checks | Files changed



Darksonn commented on May 2, 2024 • edited by ehuss Member ···

Use custom smart pointers with trait objects.

Rendered

Tracking issue: rust-lang/rust#123430

Co-authored by **@Darksonn** and **@Veykril**
Thank you to **@compiler-errors** for the original idea

☺ 👍 19 🎉 8 ❤️ 6

Trait objects with Box

```
#[derive(CoercePointee)]
#[repr(transparent)]
struct MyBox<T: ?Sized>(Box<T>);
```

Trait objects with Box

```
fn main() {
    let ptr: MyBox<i32> = MyBox(Box::new(4));

    let ptr: MyBox<dyn MyTrait> = ptr;
}
```

Trait objects with Box

```
trait MyTrait {
    fn func(self: MyBox<Self>);
}

fn call_func(value: MyBox<dyn MyTrait>) {
    value.func();
}
```

Memory safety in C?

C is trying to become more like Rust.

```
struct my_data {
    spinlock_t lock;
    int counter __guarded_by(&lock);
};
void increment(struct my_data *d)
{
    guard(spinlock)(&d->lock);
    d->counter++;
}
```

Memory safety in C?

C is trying to become more like Rust.

```
struct my_data {
    spinlock_t lock;
    int counter __guarded_by(&lock);
};
void increment(struct my_data *d)
{
    guard(spinlock)(&d->lock);
    d->counter++;
}
```

Memory safety in C?

C is trying to become more like Rust.

```
struct my_data {
    spinlock_t lock;
    int counter __guarded_by(&lock);
};
void increment(struct my_data *d)
{
    guard(spinlock)(&d->lock);
    d->counter++;
}
```

Memory safety in C?

On the lock struct: __ctx_lock_type(spinlock)

On the protected data: __guarded_by(&lock)

On lock/unlock functions: __acquires(&lock), __releases(&lock)

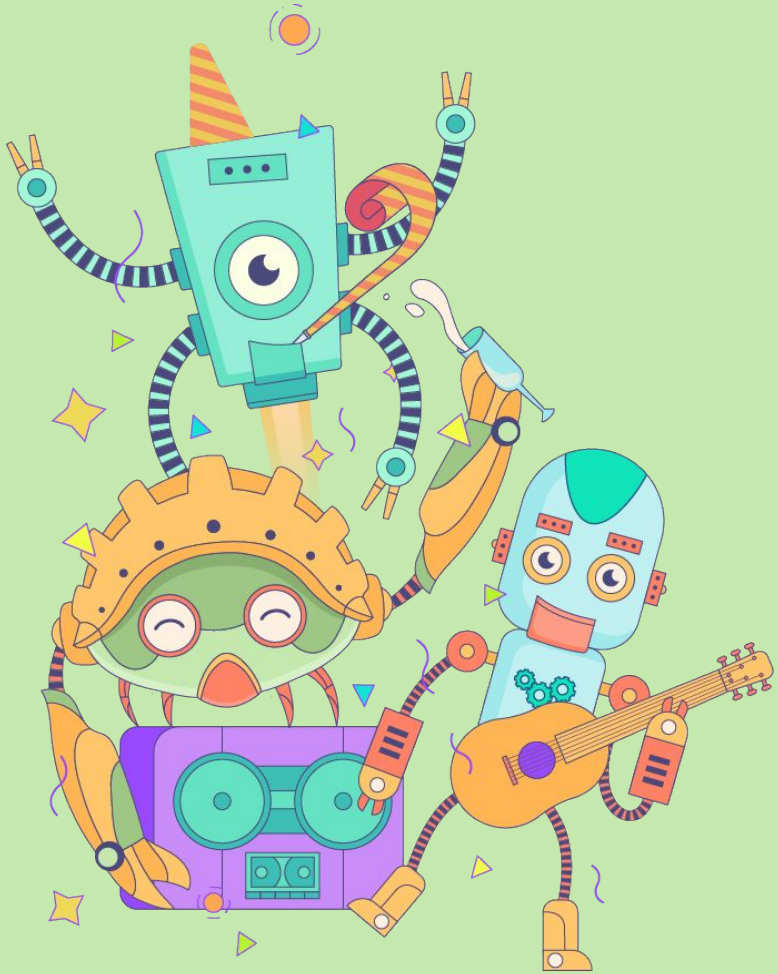
On caller/callee boundaries: __must_hold(&lock), __must_not_hold(&lock)

Escape hatches: context_unsafe(d->counter) and __context_unsafe("reason")

Is C becoming more like Rust?

- Other features too:
  - counted_by
  - __free
- Every limitation requires patching the compiler.
- Example: mutex data behind pointer
- Example: arrays of mutexes

Thank you for listening



Alice Ryhl

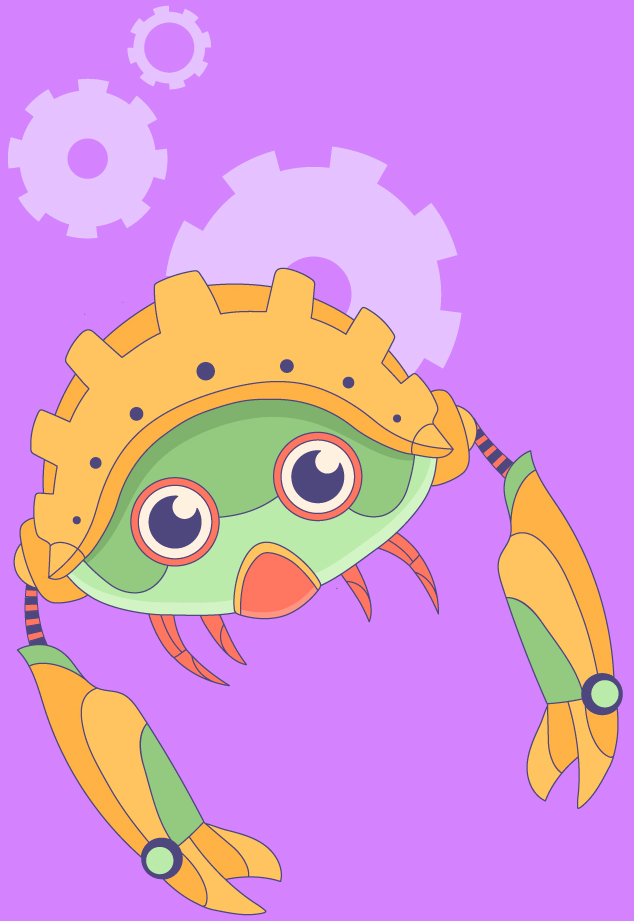
alice@ryhl.io



SPEAKER NAME

Professional Role

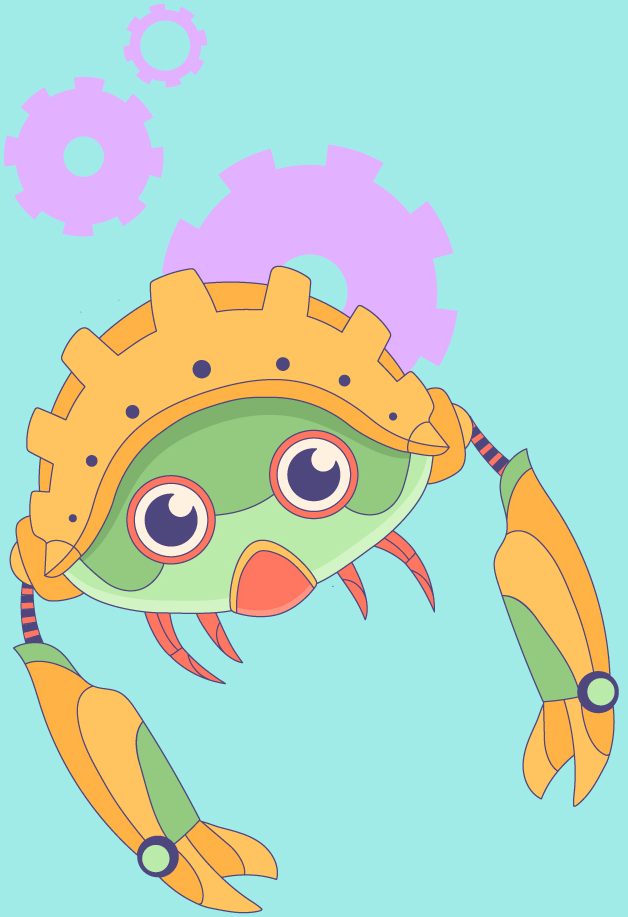
TALK NAME



SPEAKER NAME

Professional Role

TALK NAME



INTRODUCTION

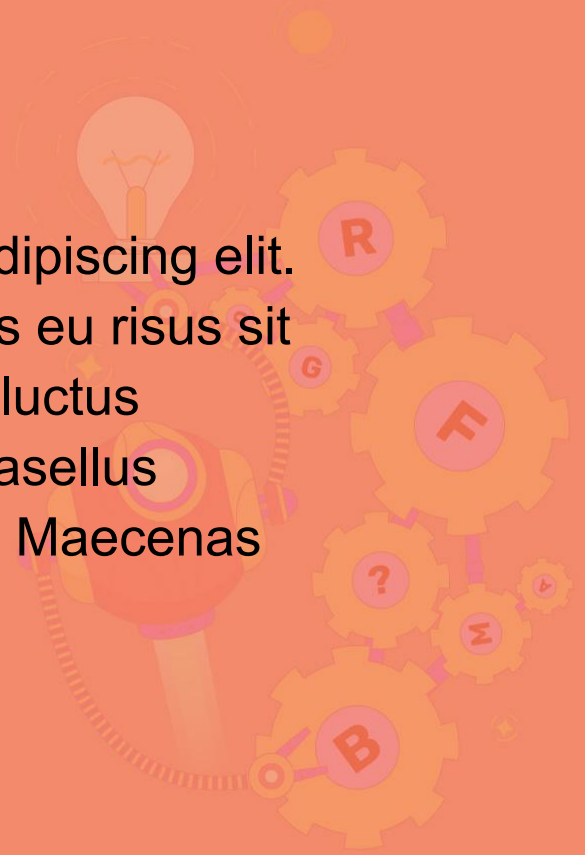
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur vitae turpis mi. Phasellus sodales eu risus sit amet dapibus. Donec semper purus in dui luctus dapibus. Nullam sed ullamcorper urna. Phasellus ullamcorper sodales sem nec ullamcorper. Maecenas blandit urna ac diam tempus viverra.

INTRODUCTION

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur vitae turpis mi. Phasellus sodales eu risus sit amet dapibus. Donec semper purus in dui luctus dapibus. Nullam sed ullamcorper urna. Phasellus ullamcorper sodales sem nec ullamcorper. Maecenas blandit urna ac diam tempus viverra.

INTRODUCTION

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur vitae turpis mi. Phasellus sodales eu risus sit amet dapibus. Donec semper purus in dui luctus dapibus. Nullam sed ullamcorper urna. Phasellus ullamcorper sodales sem nec ullamcorper. Maecenas blandit urna ac diam tempus viverra.



INTRODUCTION

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur vitae turpis mi. Phasellus sodales eu risus sit amet dapibus. Donec semper purus in dui luctus dapibus. Nullam sed ullamcorper urna. Phasellus ullamcorper sodales sem nec ullamcorper. Maecenas blandit urna ac diam tempus viverra.

SECTION TITLE

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

- Lorem ipsum dolor sit amet, consectetur adipiscing elit.
- Nam est ligula, faucibus nec urna sed, tristique volutpat massa.
- Phasellus congue leo nunc, eu vehicula enim interdum vitae.

SECTION TITLE

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

- Lorem ipsum dolor sit amet, consectetur adipiscing elit.
- Nam est ligula, faucibus nec urna sed, tristique volutpat massa.
- Phasellus congue leo nunc, eu vehicula enim interdum vitae.

SECTION TITLE

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

- Lorem ipsum dolor sit amet, consectetur adipiscing elit.
- Nam est ligula, faucibus nec urna sed, tristique volutpat massa.
- Phasellus congue leo nunc, eu vehicula enim interdum vitae.

SECTION TITLE

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

- Lorem ipsum dolor sit amet, consectetur adipiscing elit.
- Nam est ligula, faucibus nec urna sed, tristique volutpat massa.
- Phasellus congue leo nunc, eu vehicula enim interdum vitae.

SECTION TITLE

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

- Lorem ipsum dolor sit amet, consectetur adipiscing elit.
- Nam est ligula, faucibus nec urna sed, tristique volutpat massa.
- Phasellus congue leo nunc, eu vehicula enim interdum vitae.

SECTION TITLE

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

- Lorem ipsum dolor sit amet, consectetur adipiscing elit.
- Nam est ligula, faucibus nec urna sed, tristique volutpat massa.
- Phasellus congue leo nunc, eu vehicula enim interdum vitae.

Subsection Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam est ligula, faucibus nec urna sed, tristique volutpat massa. Phasellus congue leo nunc, eu vehicula enim interdum vitae. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Nam est ligula, faucibus nec urna sed, tristique volutpat massa. Phasellus congue leo nunc, eu vehicula enim interdum vitae.

Subsection Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam est ligula, faucibus nec urna sed, tristique volutpat massa. Phasellus congue leo nunc, eu vehicula enim interdum vitae. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Nam est ligula, faucibus nec urna sed, tristique volutpat massa. Phasellus congue leo nunc, eu vehicula enim interdum vitae.

Subsection Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam est ligula, faucibus nec urna sed, tristique volutpat massa. Phasellus congue leo nunc, eu vehicula enim interdum vitae. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Nam est ligula, faucibus nec urna sed, tristique volutpat massa. Phasellus congue leo nunc, eu vehicula enim interdum vitae.

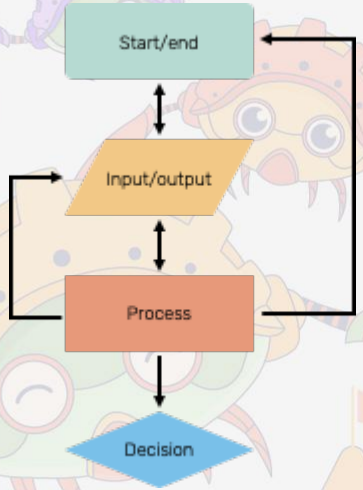
Subsection Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam est ligula, faucibus nec urna sed, tristique volutpat massa. Phasellus congue leo nunc, eu vehicula enim interdum vitae. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Nam est ligula, faucibus nec urna sed, tristique volutpat massa. Phasellus congue leo nunc, eu vehicula enim interdum vitae.

Subsection Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam est ligula, faucibus nec urna sed, tristique volutpat massa. Phasellus congue leo nunc, eu vehicula enim interdum vitae. Lorem ipsum dolor sit amet, consectetur adipiscing elit.



Section Title

Subsection Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Note: For use with the projector it is best to use a light theme in order to improve the accessibility and readability of the code.
You can paste the code from VScode that includes the formatting and optionally adapt the background with the "Fill color" button.

```
func hello() {
    fmt.Println("Hello world! 🌍")
}
```

Section Title

Subsection Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Note: For use with the projector it is best to use a light theme in order to improve the accessibility and readability of the code.
You can paste the code from VScode that includes the formatting and optionally adapt the background with the "Fill color" button.

```
func hello() {
    fmt.Println("Hello world! 🌍")
}
```

Section Title

Subsection Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Note: For use with the projector it is best to use a light theme in order to improve the accessibility and readability of the code.
You can paste the code from VScode that includes the formatting and optionally adapt the background with the "Fill color" button.

```
func hello() {
    fmt.Println("Hello world! 🌍")
}
```

Subsection Title

```
func hello() {
    fmt.Println("Hello world! 🌍")
}
```

Subsection Title

```
func hello() {
    fmt.Println("Hello world! 🌍")
}
```

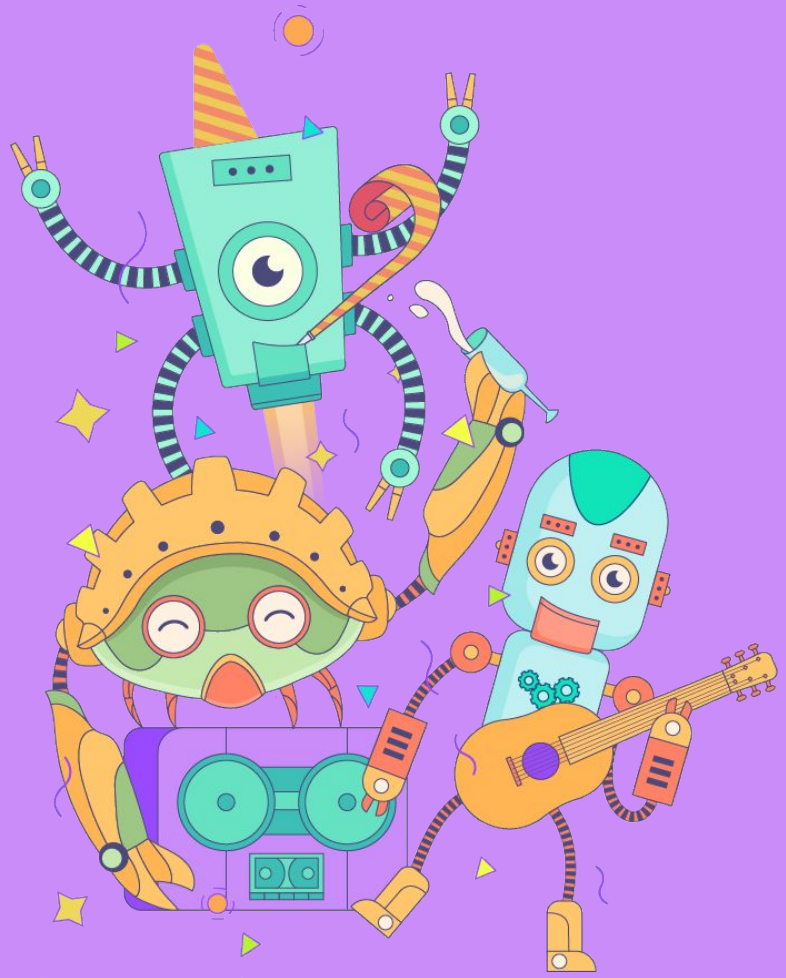
Subsection Title

```
func hello() {
    fmt.Println("Hello world! 🌍")
}
```

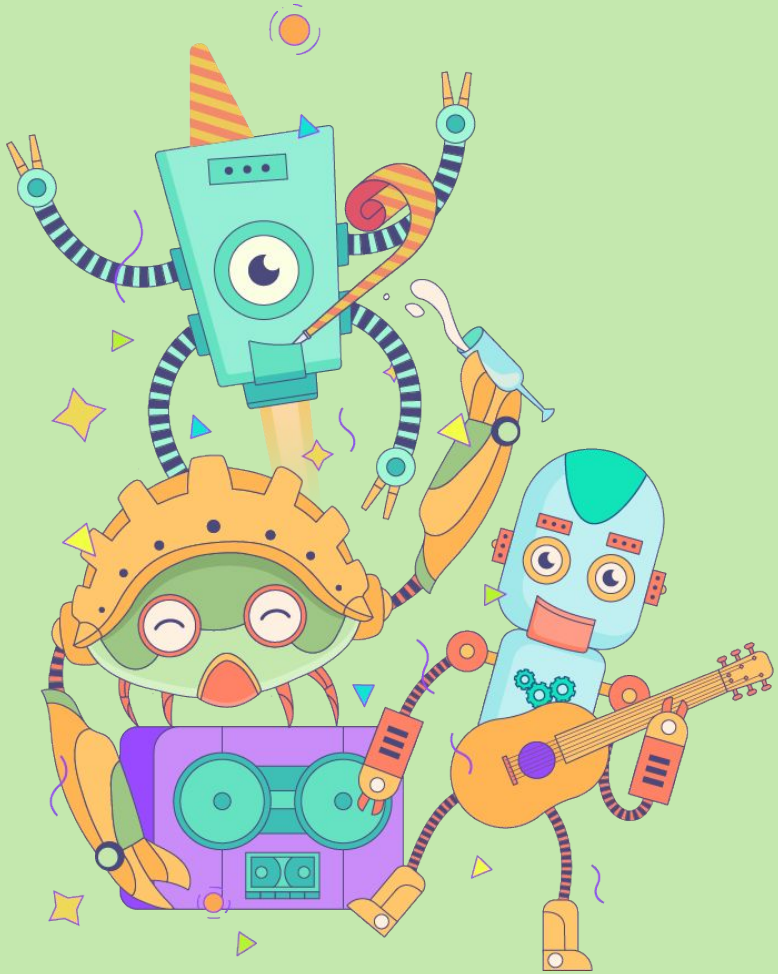
Subsection Title

```
func hello() {
    fmt.Println("Hello world! 🌍")
}
```

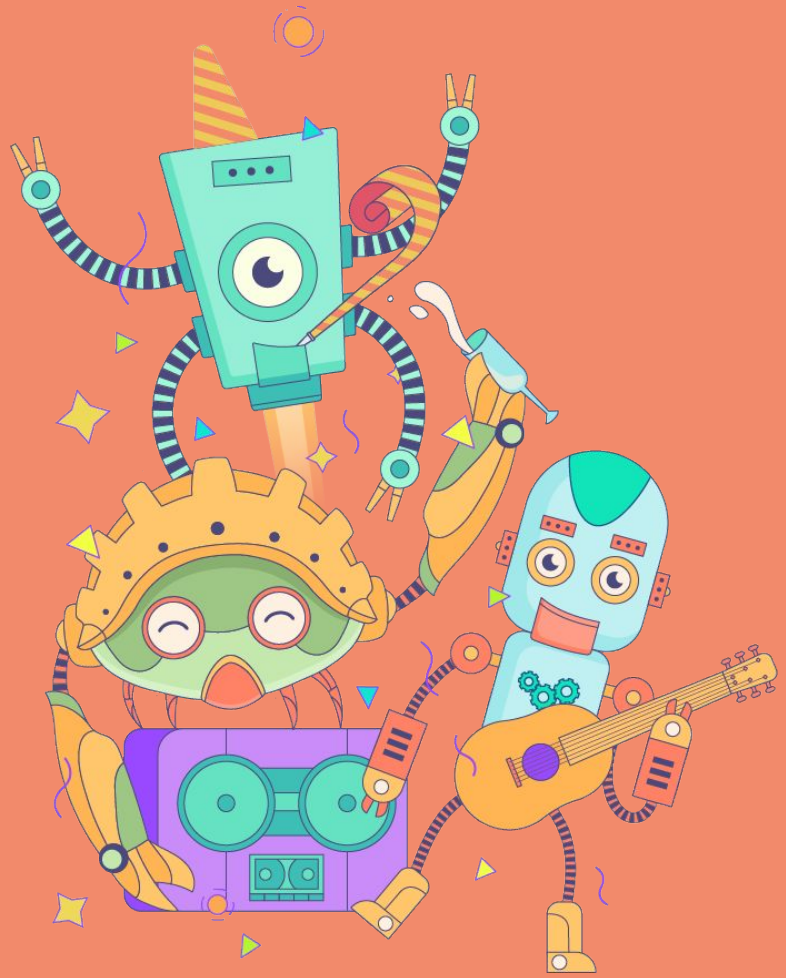
Greetings and conclusion



Greetings and conclusion



Greetings and conclusion



SPEAKER NAME

mail@speaker.com



SPEAKER NAME

mail@speaker.com



SPEAKER NAME

mail@speaker.com

