making allocators async
is a bad great idea

```
pub unsafe trait GlobalAlloc {
    unsafe fn alloc(
        &self,
        layout: Layout,
    ) -> *mut u8;

    unsafe fn dealloc(
        &self,
        ptr: *mut u8,
        layout: Layout,
    );
    // ...
}
```

```rust
pub unsafe trait Allocator {
    fn allocate(
        &self,
        layout: Layout,
    ) -> Result<NonNull<[u8]>, AllocError>;

    unsafe fn deallocate(
        &self,
        ptr: NonNull<u8>,
        layout: Layout,
    );
    // ...
}
```

what does it mean to be
"out of memory"?

```rust
pub async fn alloc(layout: Layout) -> NonNull<u8>;
pub unsafe fn dealloc(ptr: *mut u8, layout: Layout);
```

handling reality in "userspace"

challenge one:
`drop` isn't `async`

challenge two:

`alloc::collections`

isn't `async`

challenge three:
you can't "turn off"
non-`async` allocations