Please add this text at the end of the Function Pointers section, just before the Returning Closures section starts, so at the end of page 447 and before page 448.

We have another useful pattern that exploits an implementation detail of tuple structs and tuple-struct enum variants. These types use () as initializer syntax, which looks like a function call. The initializers are actually implemented as functions returning an instance that’s constructed from their arguments. We can use these initializer functions as function pointers that implement the closure traits, which means we can specify the initializer functions as arguments for methods that take closures, like so:

enum Status {

Value(u32),

Stop,

}

let list\_of\_statuses: Vec<Status> =

(0u32..20)

.map(Status::Value)

.collect();

Here we create Status::Value instances using each u32 value in the range that map is called on by using the initializer function of Status::Value. Some people prefer this style, and some people prefer to use closures. They compile to the same code, so use whichever style is clearer to you.