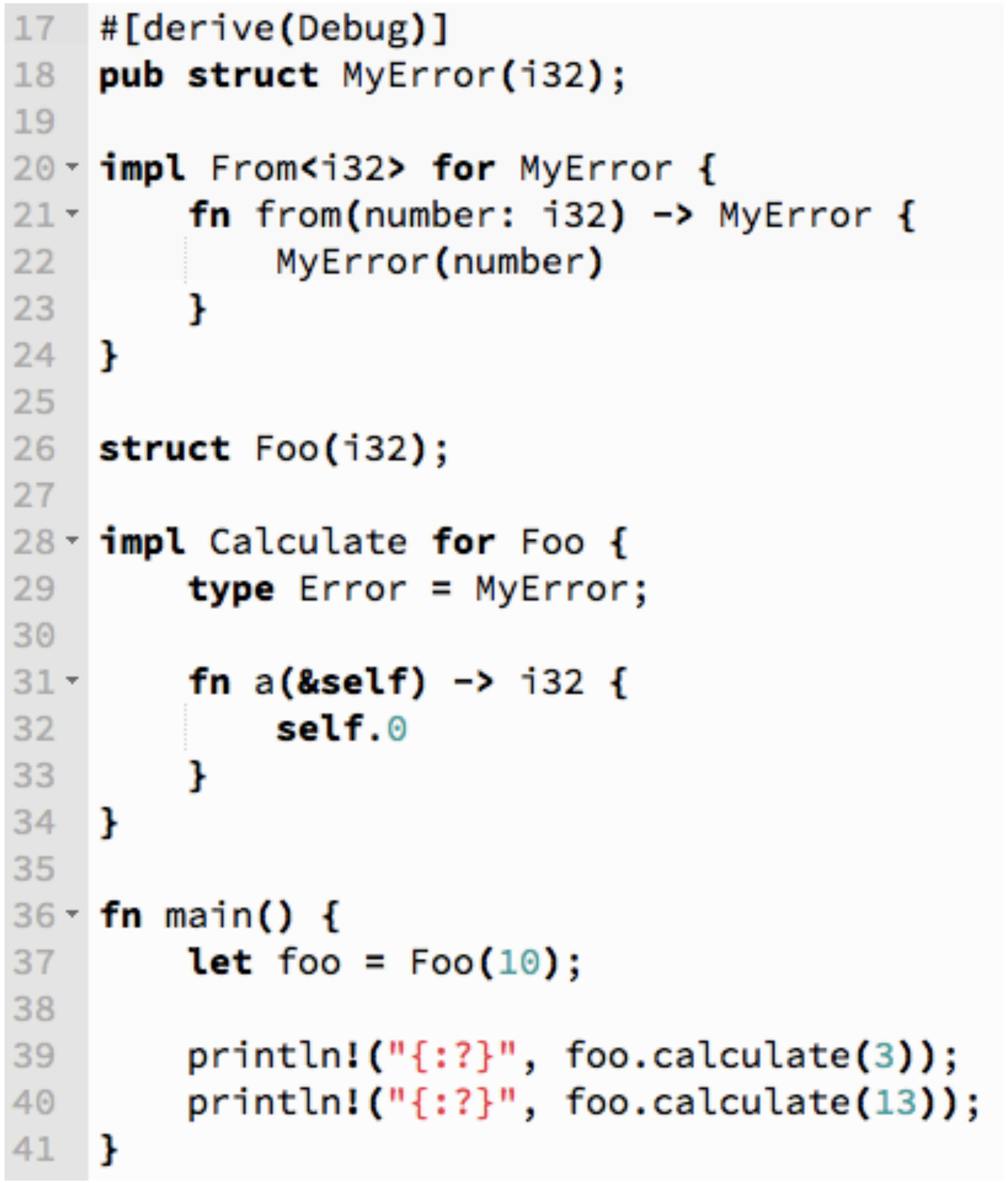
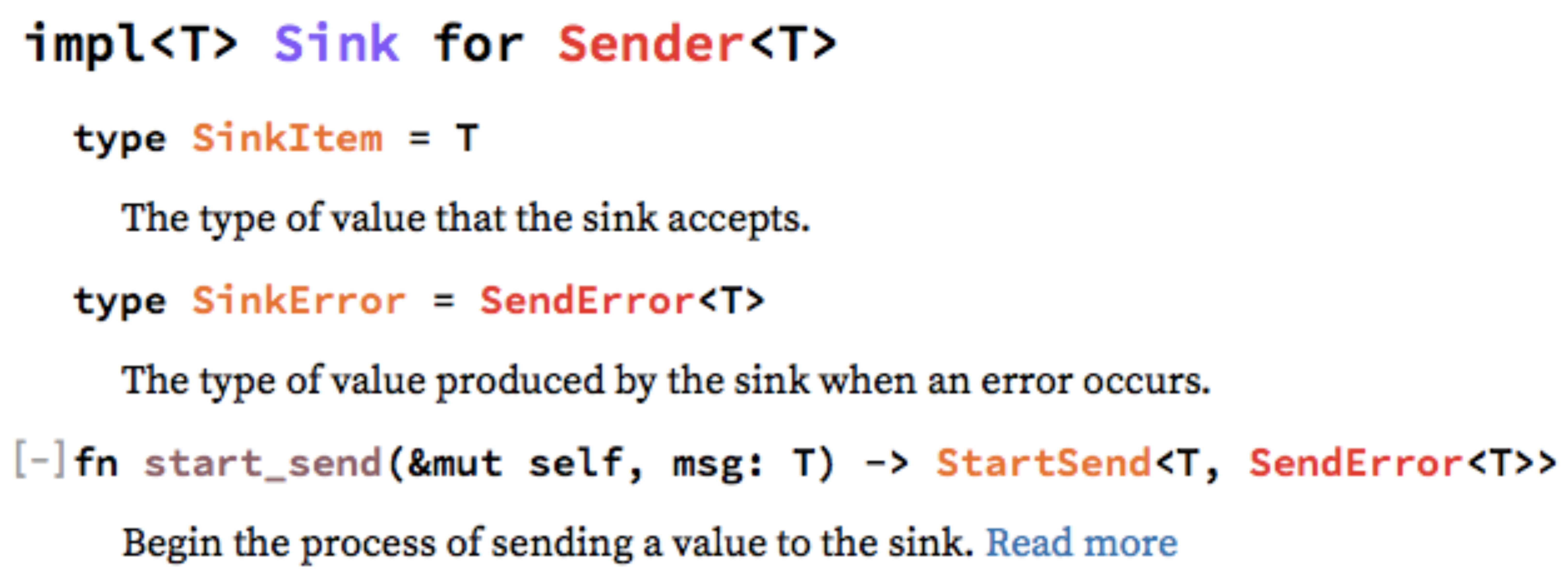


```rust
#[derive(Debug)]
pub struct MyError(i32);

impl From<i32> for MyError {
    fn from(number: i32) -> MyError {
        MyError(number)
    }
}

struct Foo(i32);

impl Calculate for Foo {
    type Error = MyError;

    fn a(&self) -> i32 {
        self.0
    }
}

fn main() {
    let foo = Foo(10);

    println!("{:?}", foo.calculate(3));
    println!("{:?}", foo.calculate(13));
}
```

```
Ok(7)
Err(MyError(23))
```

From bound in *futures* crate



```
impl<T> Sink for Sender<T>

  type SinkItem = T
```

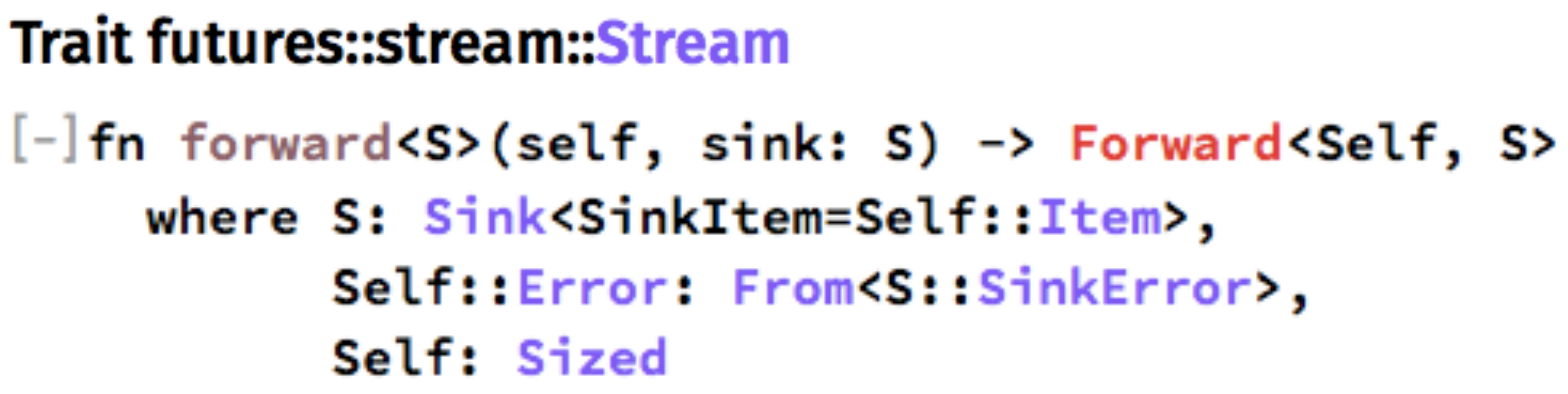
The type of value that the sink accepts.

```
  type SinkError = SendError<T>
```

The type of value produced by the sink when an error occurs.

```
[-]fn start_send(&mut self, msg: T) -> StartSend<T, SendError<T>>
```

Begin the process of sending a value to the sink. Read more



Trait futures::stream::Stream

```
[-]fn forward<S>(self, sink: S) -> Forward<Self, S>
     where S: Sink<SinkItem=Self::Item>,
           Self::Error: From<S::SinkError>,
           Self: Sized
```