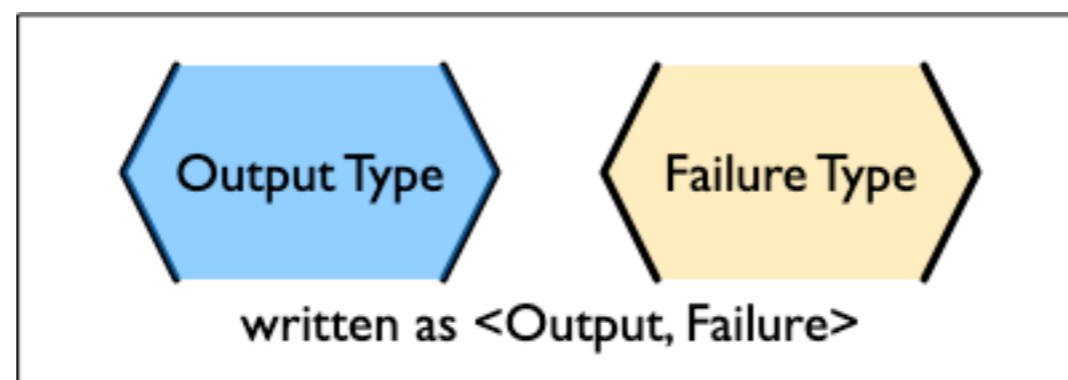


Combine

Professor Larry Heimann
Carnegie Mellon University
67-443: Mobile App Development

Publishers

- Provides data when available and upon request.
- A publisher that has not had any subscription requests will not provide any data.
- Describe publishers with two associated types: one for Output and one for Failure.



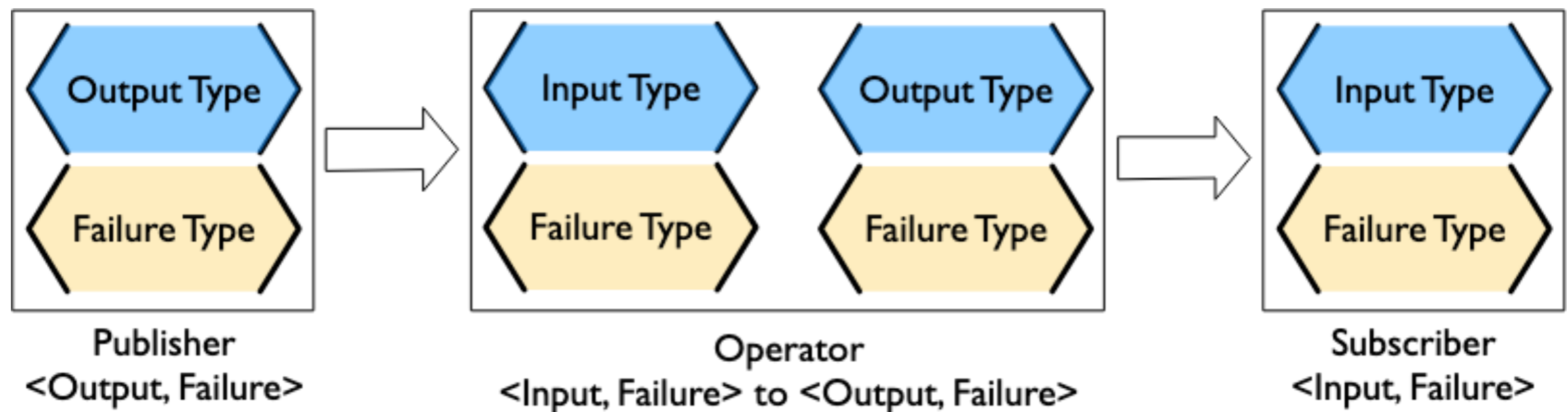
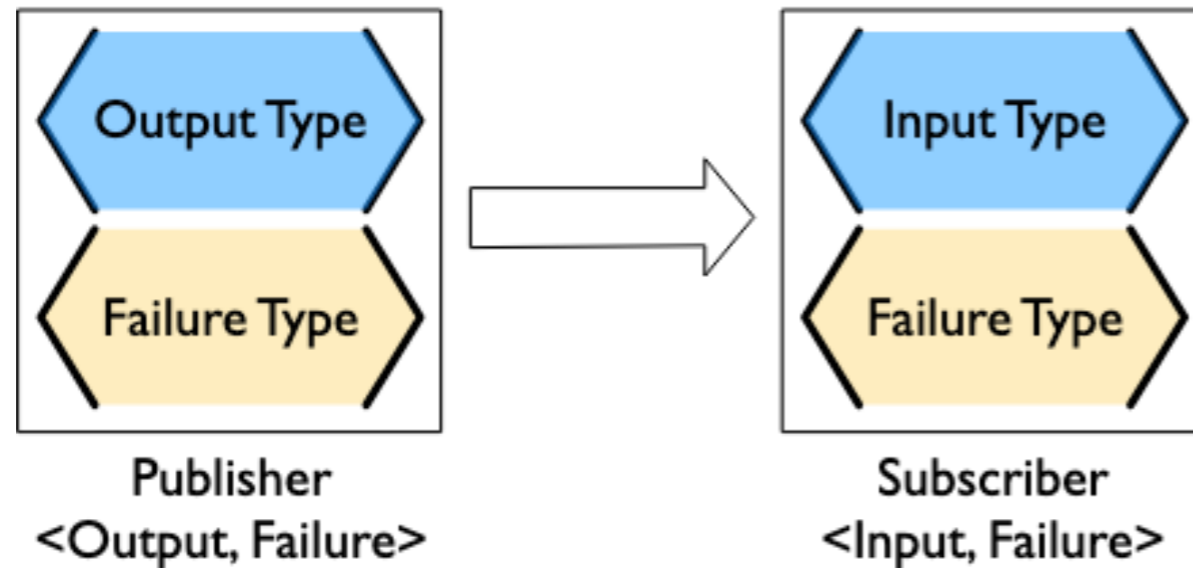
Subscribers

- Subscribers are responsible for requesting data and accepting the data (and possible failures) provided by a publisher.
- A subscriber is described with two associated types, one for Input and one for Failure.
- The subscriber initiates the request for data, and controls the amount of data it receives.
- It can be thought of as "driving the action" within Combine, as without a subscriber, the other components stay idle.

The Sink Subscriber

- Most common way to create a subscriber in Combine is with the `sink()` method.
- If the error type is `Never` then the sink method will provide one simple closure to handle emitted values.
- In cases where the publisher can fail or complete, sink provides another closure called `receiveCompletion`. It allows you to process completion and failure events.
- Of course, you also have to handle the actual processing of values with another closure `receiveValue`.

Pipelines & Operators



Apple Foundation Publishers

- There is a generic Publisher protocol so you can create customized publishers
- Apple provides 3 popular Publishers in Foundation
 - `DataTaskPublisher`
 - `NotificationCenter.Publisher`
 - `Timer.Publisher`

DataTaskPublisher

- The `URLSession` class provides the convenient `dataTaskPublisher(for:)` method, which returns a `DataTaskPublisher` based on a URL. This is most useful for basic API calls.
- The `dataTaskPublisher` emits a tuple of `(data: Data, response: URLResponse)`, much like the existing closure-based handler. The error is handled by the publisher's failure event.
- In a typical API call that returns some JSON, we map to the received data, decode to a matching type and handle errors.

DataTaskPublisher

```
let url = URL(string: "https://  
jsonplaceholder.typicode.com/posts/1")!
```

```
let cancellable = URLSession.shared  
    .dataTaskPublisher(for: url)  
    .retry(1)  
    .map(\.data)  
    .decode(type: Post.self, decoder: JSONDecoder())  
    .replaceError(with: .empty)  
    .sink { it in  
        print(it)  
    }
```