Learning Reactive Programming with Reactor

Esteban Herrera
JAVA ARCHITECT

@eh3rrera www.eherrera.net

Overview

Reactive Streams

- Interfaces

Project Reactor

- Flux
- Mono

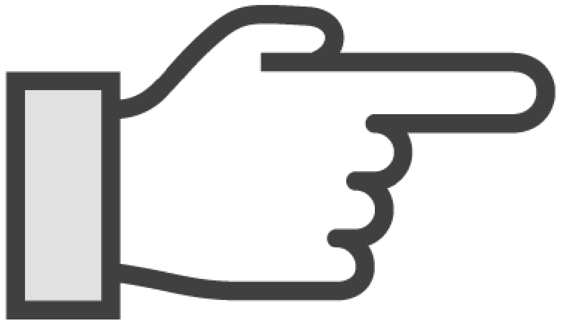
Setting up the demo

Demo: Flux and Mono

Demo: Operators

Reactive Streams

Reactive Streams



Contract

- Asynchronous
- Non-blocking
- Backpressure

Standard

- Lightbend, Netflix, Twitter, and others

Only concerned with the stream of data

- No transformation (operators)

Reactive Streams

API

Technology Compatibility Kit (TCK)

Core Interfaces

Publisher

Subscriber

Subscription

Processor

Publisher

```
public interface Publisher<T> {
    public void subscribe(Subscriber<? super T> s);
}
```

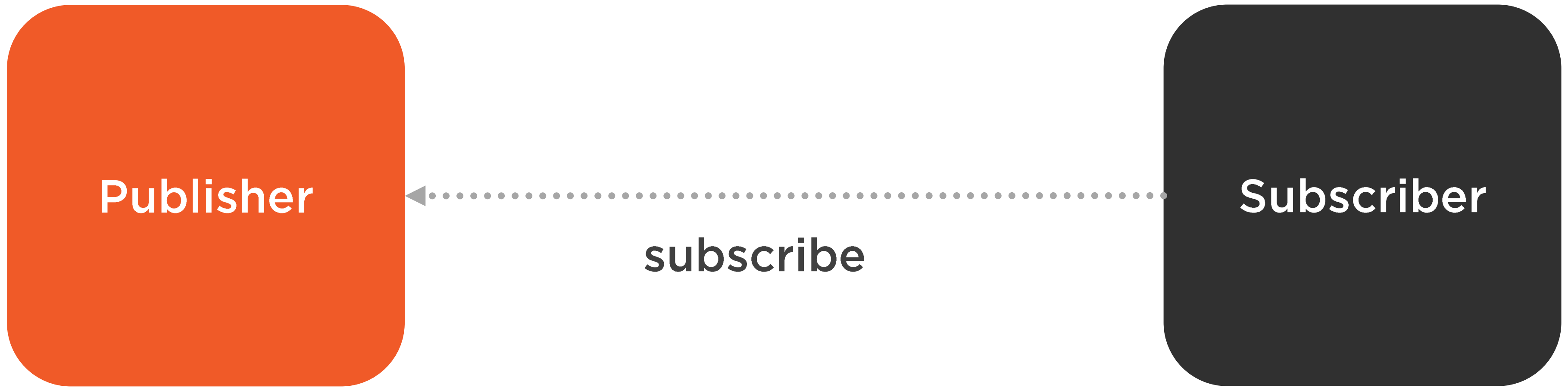
Subscriber

```java
public interface Subscriber<T> {
    public void onSubscribe(Subscription s);
    public void onNext(T t);
    public void onError(Throwable t);
    public void onComplete();
}
```

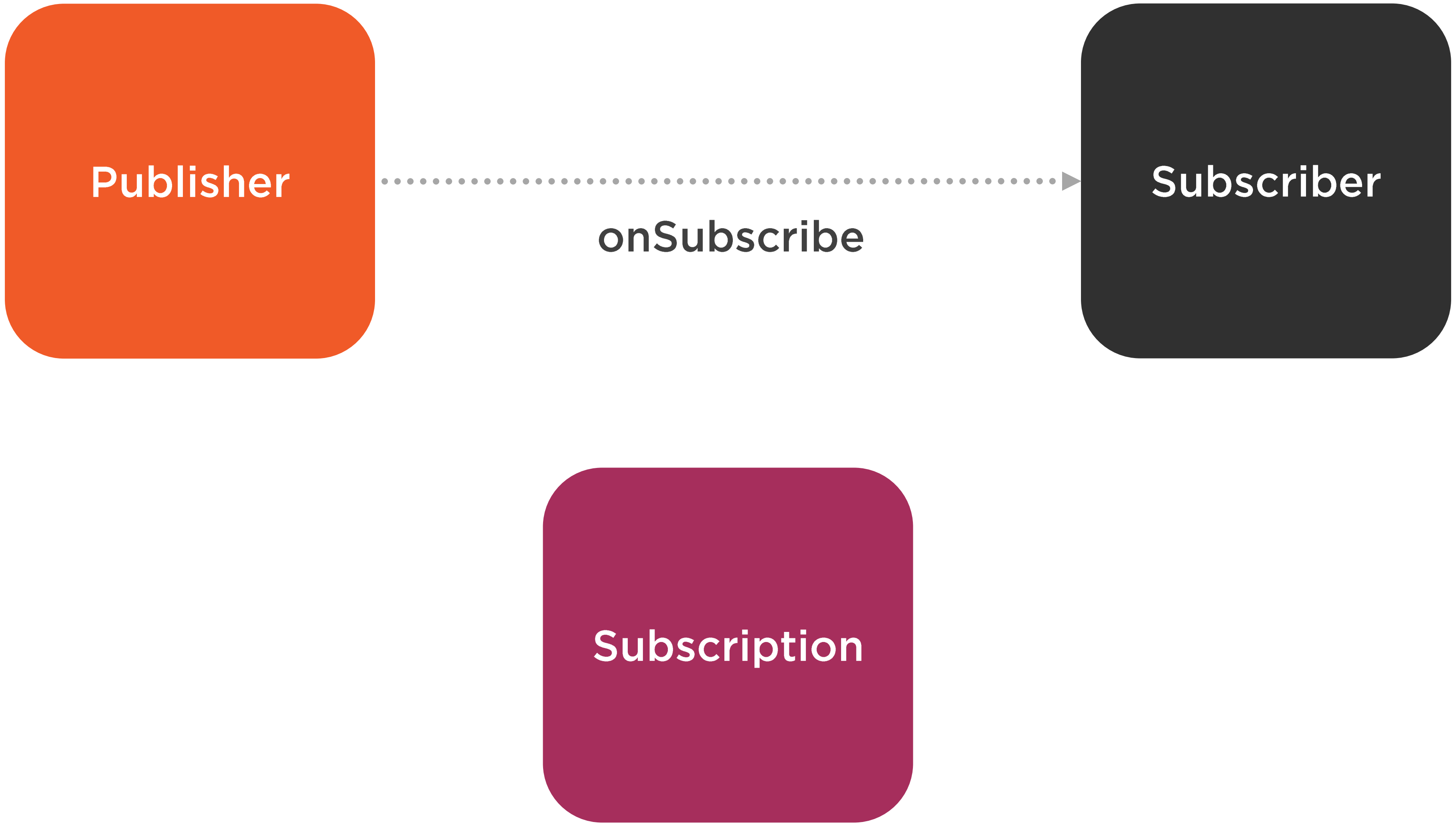
Subscription

```
public interface Subscription {
    public void request(long n);
    public void cancel();
}
```

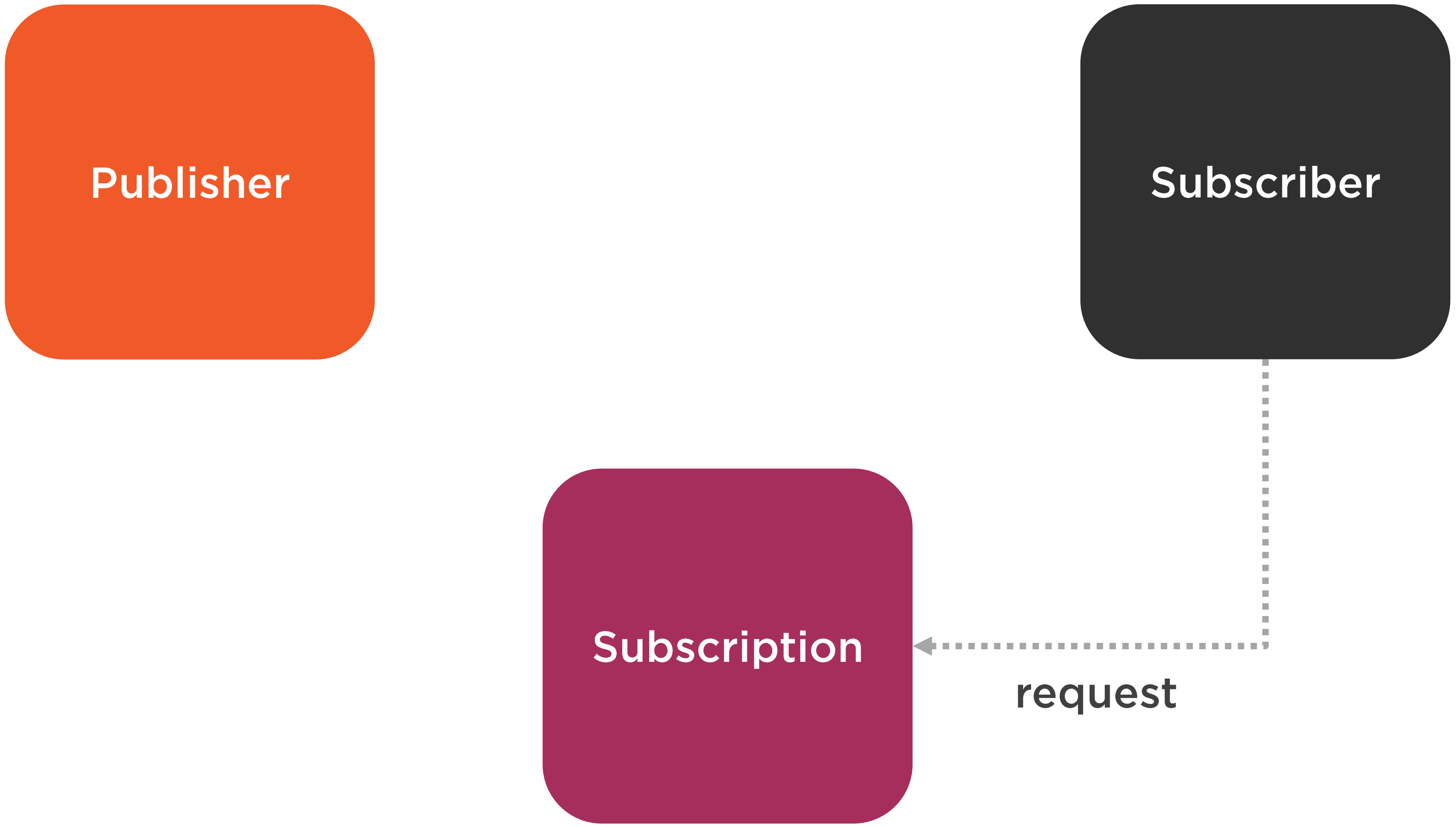
Data Flow



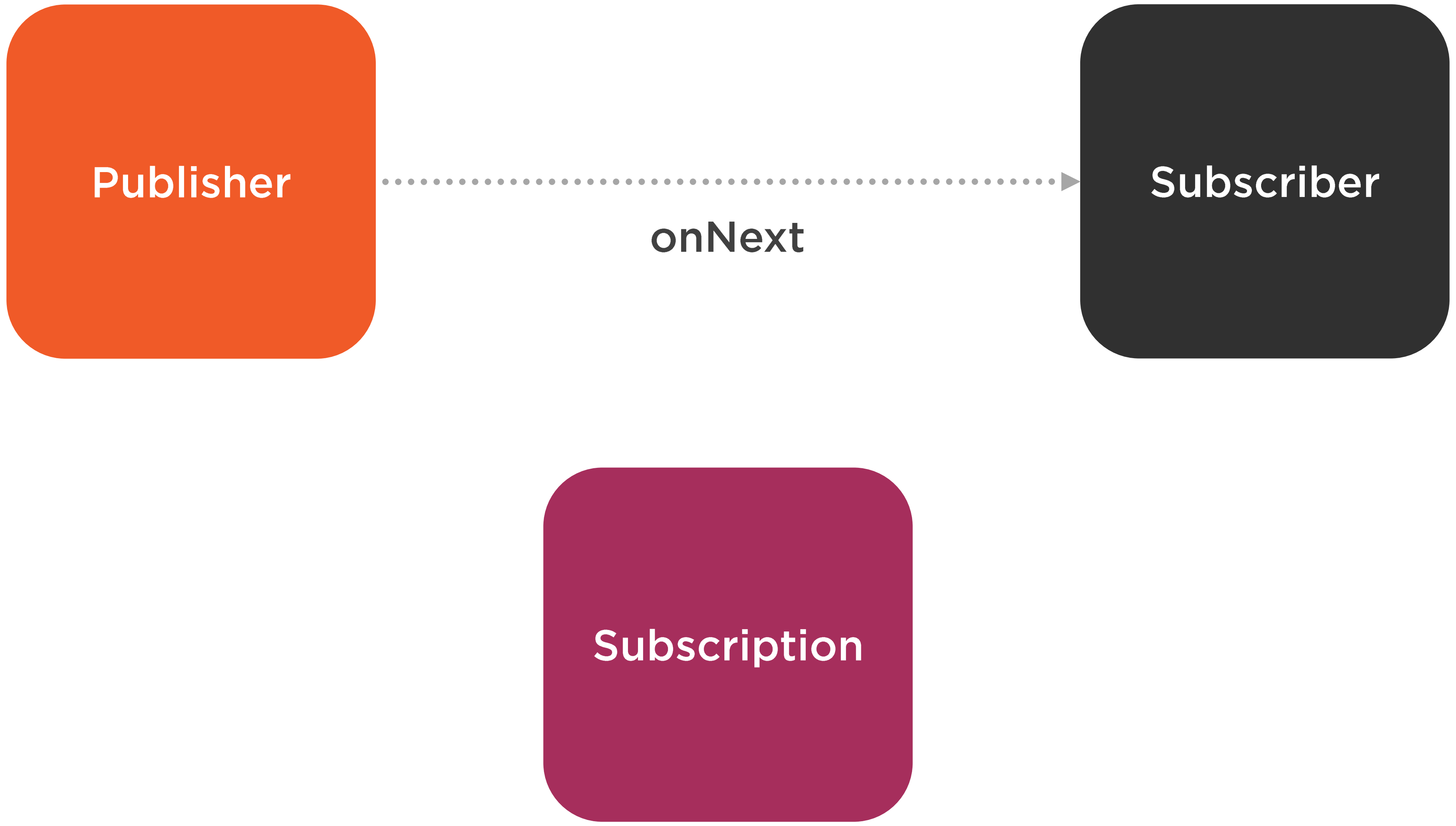
Data Flow



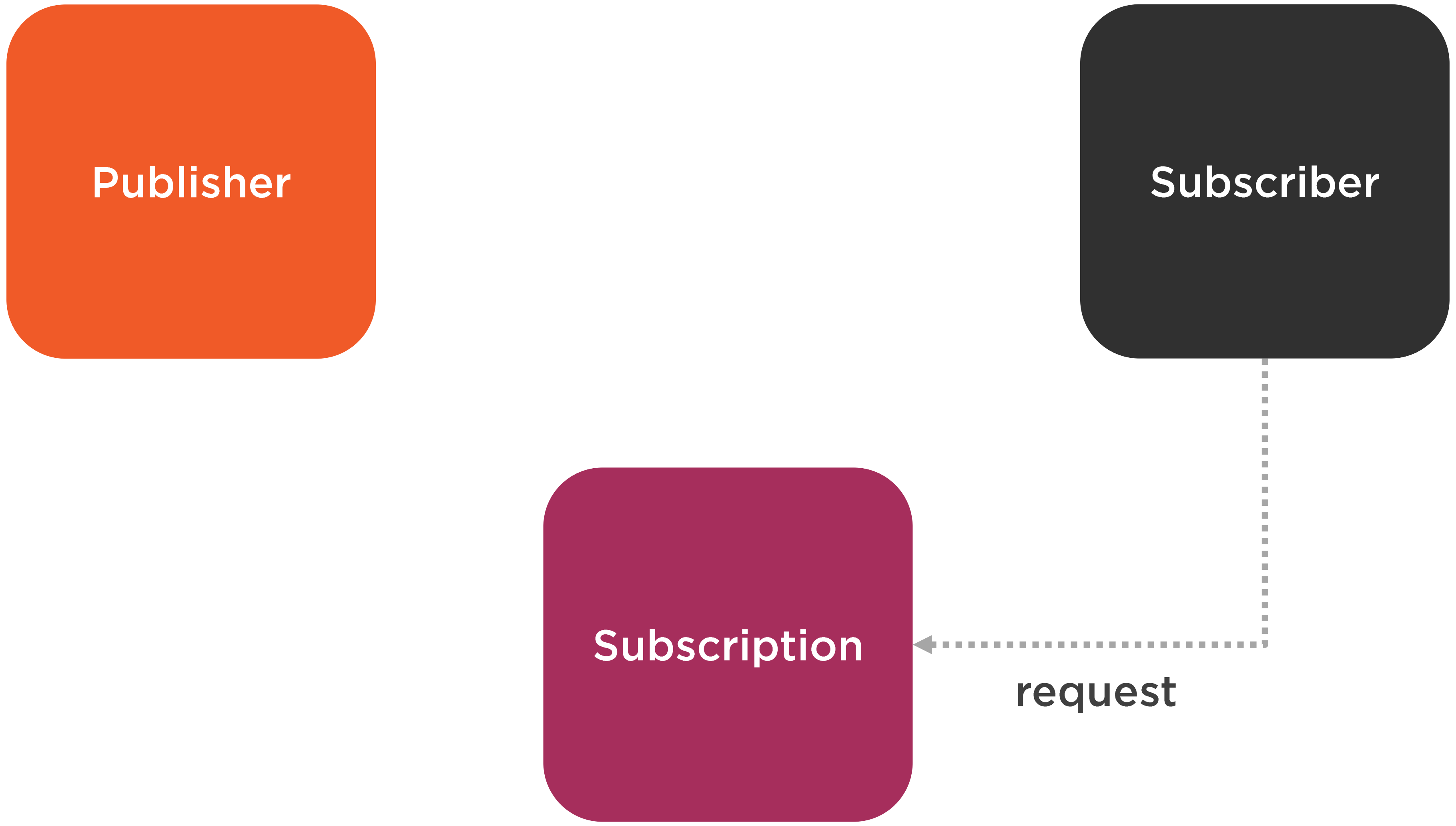
Data Flow



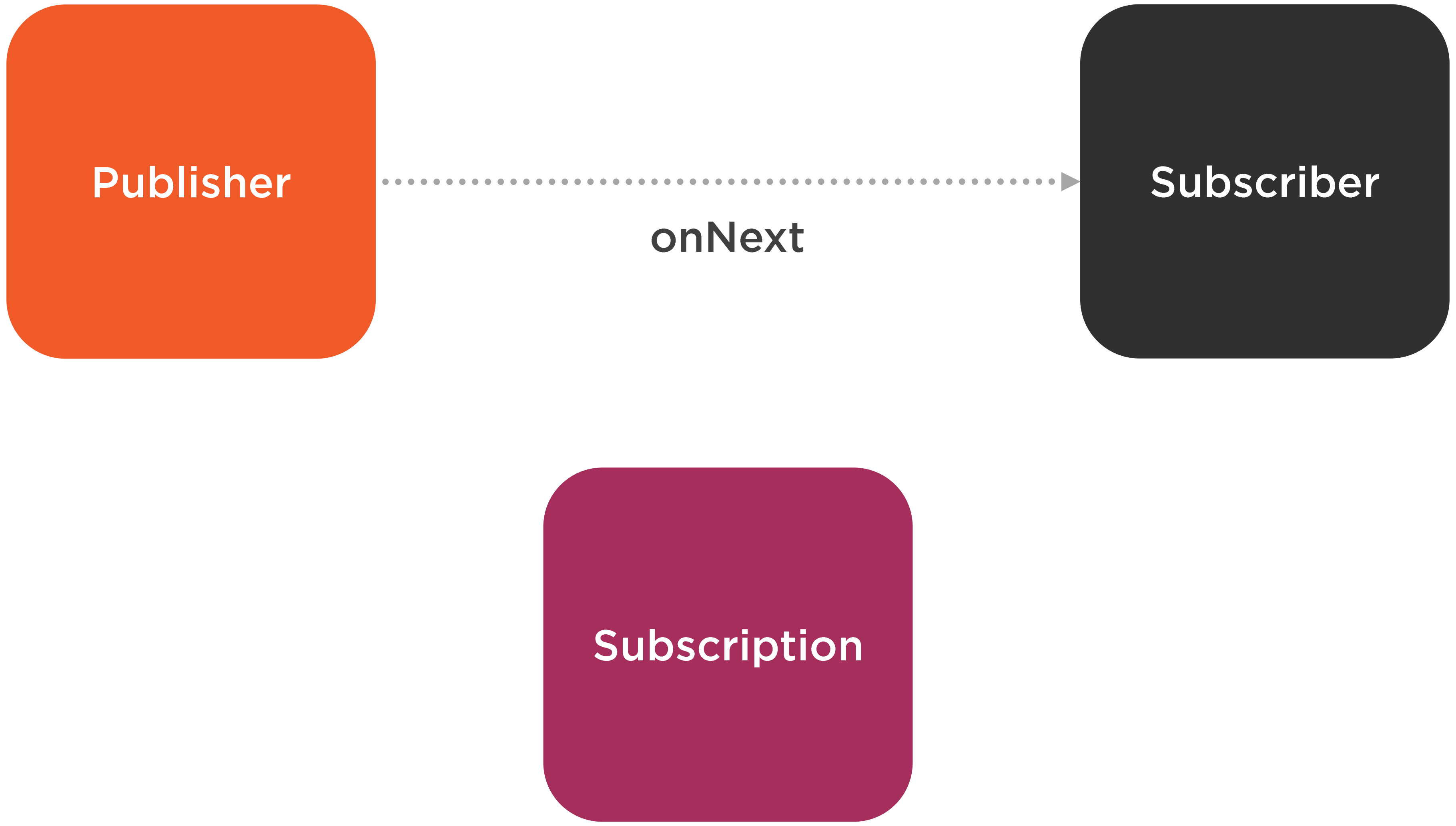
Data Flow



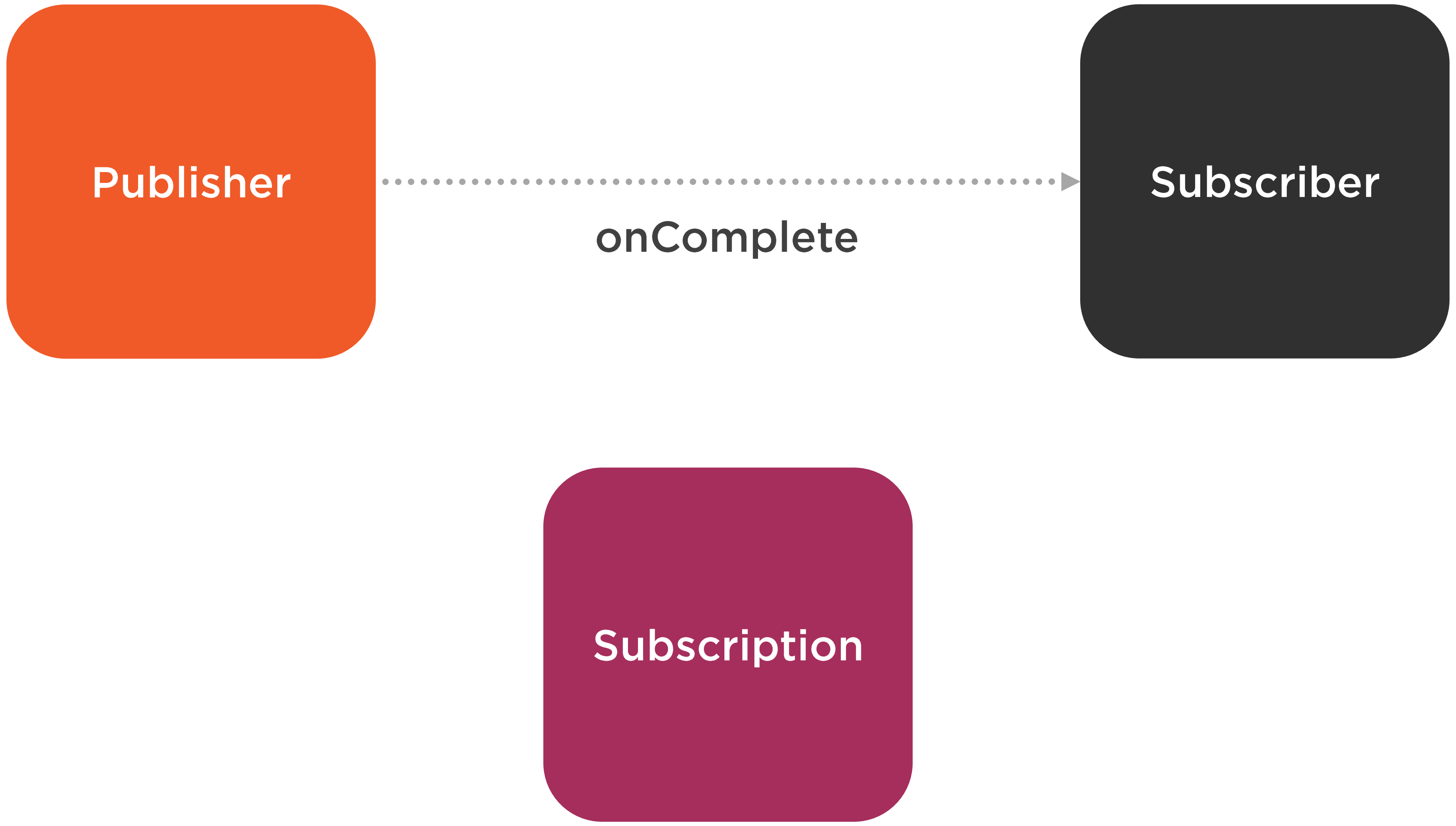
Data Flow



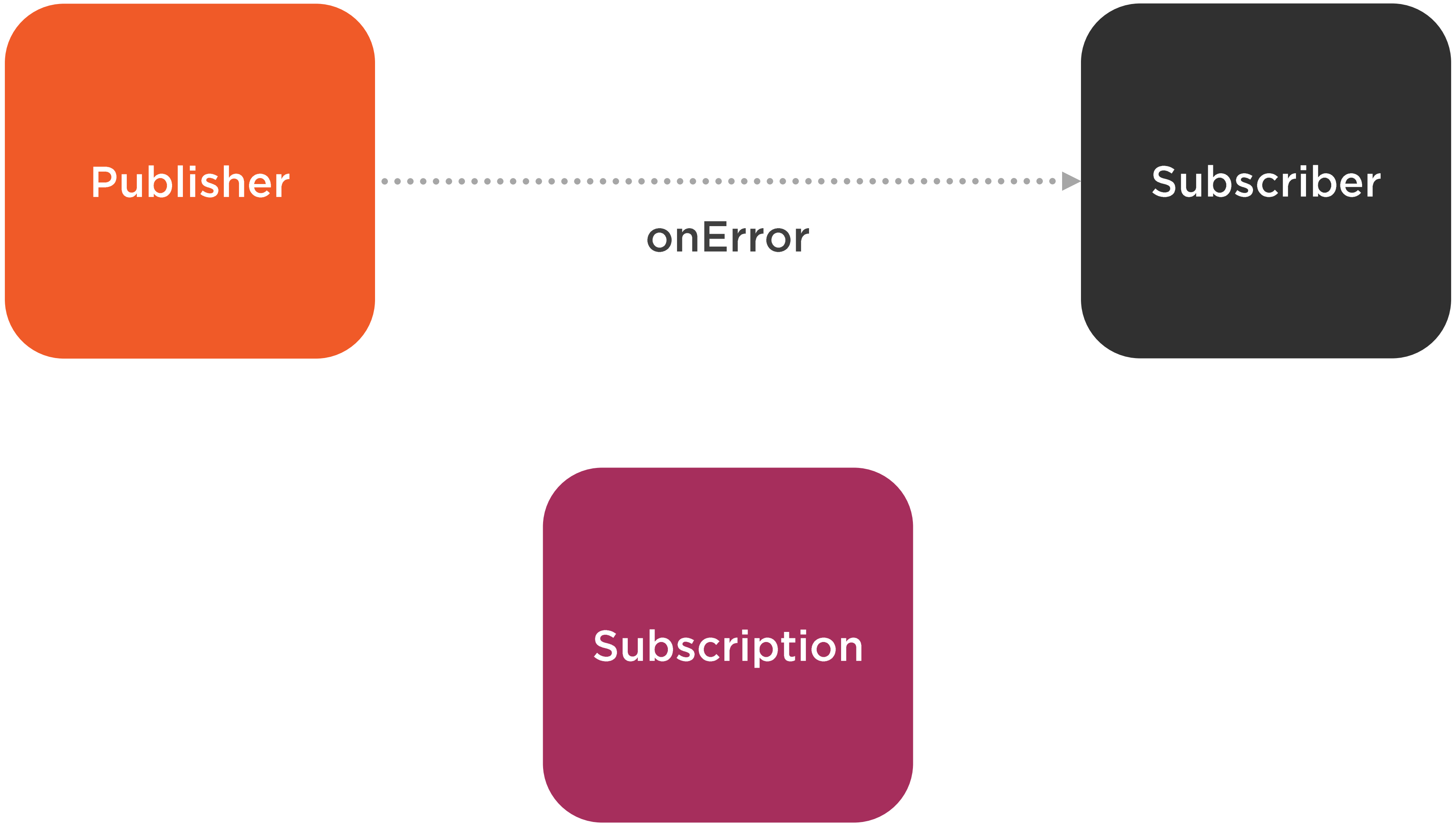
Data Flow



Data Flow



Data Flow



Project Reactor

Reactor Publishers

[0, 1]

Mono

[0 … n]

Flux

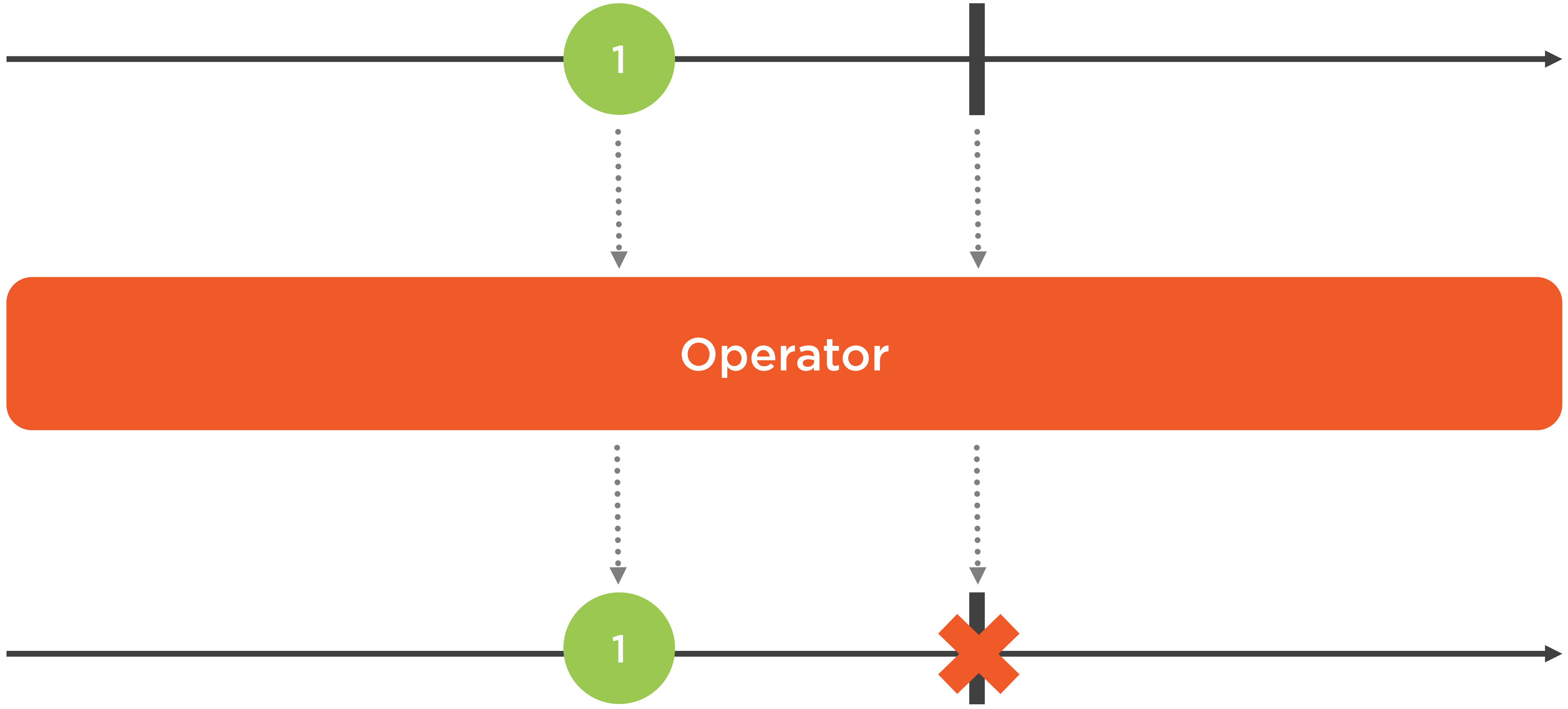
Reactor Publishers

Mono is like the Optional type

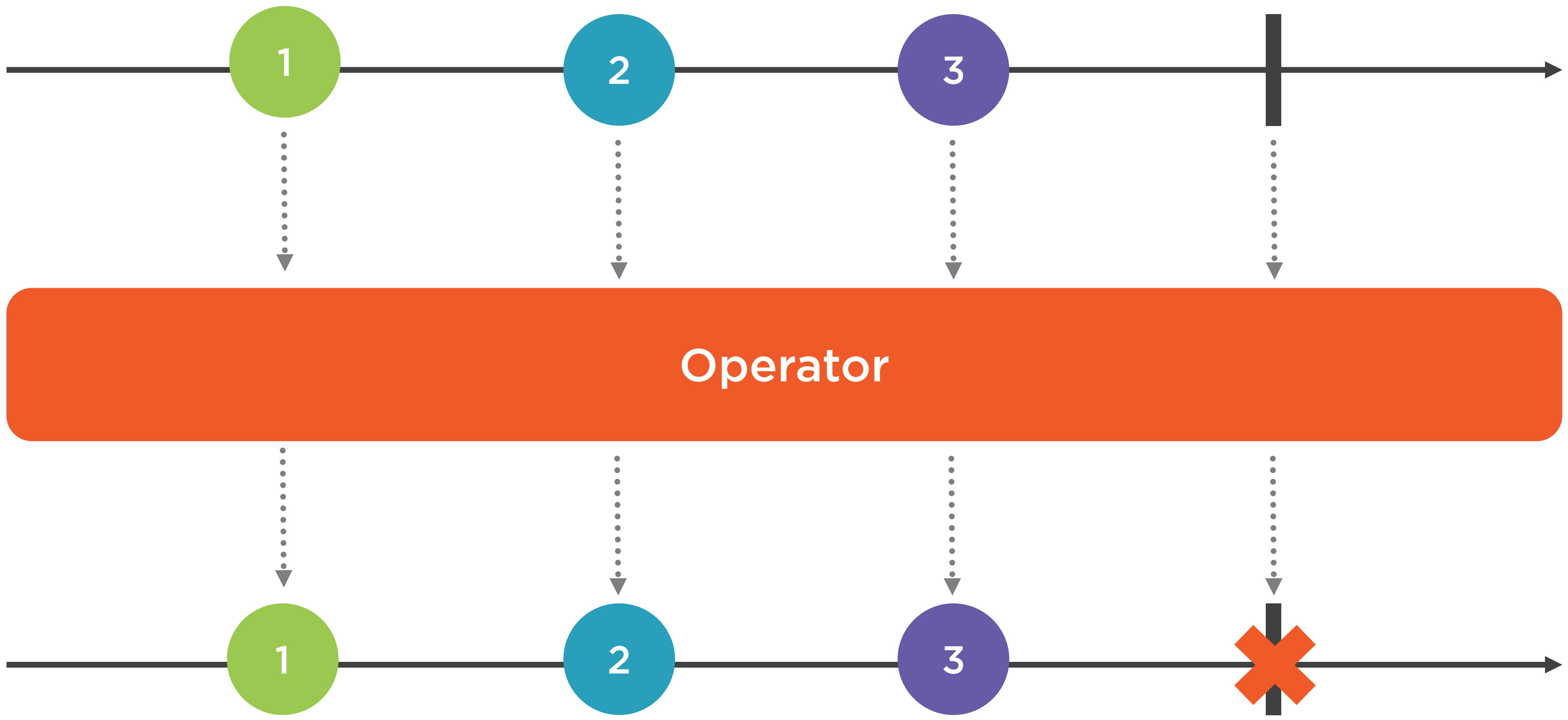
What to return?

- Flux for lists
- Mono for single objects or void

Mono



Flux



The Subscribe Method

```
subscribe()

subscribe(Consumer<? super T> consumer)

subscribe(Consumer<? super T> consumer,
          Consumer<? super Throwable> errorConsumer)

subscribe(Consumer<? super T> consumer,
          Consumer<? super Throwable> errorConsumer,
          Runnable completeConsumer)

subscribe(Consumer<? super T> consumer,
          Consumer<? super Throwable> errorConsumer,
          Runnable completeConsumer,
          Consumer<? super Subscription> subscriptionConsumer)
```

Demo

Setting up the project

Demo

Mono

- Creation
- Subscription
- Exceptions

Demo

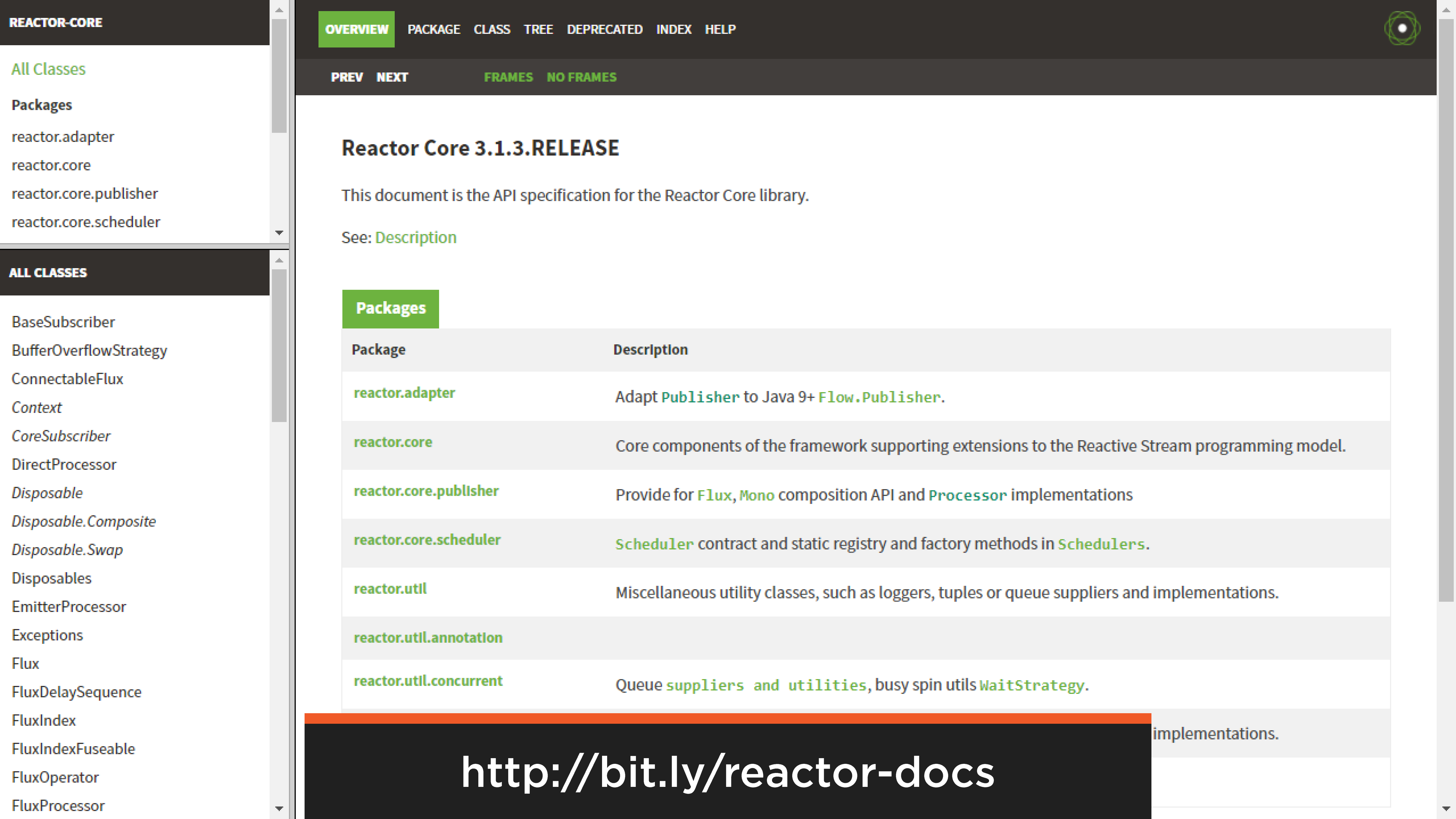
Flux

- Creation
- Subscription
- Request

Demo

Operators

- map
- flatMap
- concat and merge
- zip



REACTOR-CORE

All Classes

Packages

reactor.adapter
reactor.core
reactor.core.publisher
reactor.core.scheduler

ALL CLASSES

BaseSubscriber
BufferOverflowStrategy
ConnectableFlux
Context
CoreSubscriber
DirectProcessor
Disposable
Disposable.Composite
Disposable.Swap
Disposables
EmitterProcessor
Exceptions
Flux
FluxDelaySequence
FluxIndex
FluxIndexFuseable
FluxOperator
FluxProcessor

OVERVIEW PACKAGE CLASS TREE DEPRECATED INDEX HELP
PREV NEXT FRAMES NO FRAMES

Reactor Core 3.1.3.RELEASE

This document is the API specification for the Reactor Core library.

See: Description

Packages

Package	Description
reactor.adapter	Adapt `Publisher` to Java 9+ `Flow.Publisher`.
reactor.core	Core components of the framework supporting extensions to the Reactive Stream programming model.
reactor.core.publisher	Provide for `Flux`, `Mono` composition API and `Processor` implementations
reactor.core.scheduler	`Scheduler` contract and static registry and factory methods in `Schedulers`.
reactor.util	Miscellaneous utility classes, such as loggers, tuples or queue suppliers and implementations.
reactor.util.annotation	
reactor.util.concurrent	Queue `suppliers and utilities`, busy spin utils `WaitStrategy`.
	implementations.

http://bit.ly/reactor-docs

Things to Remember

Reactive Streams

- Publisher
- Subscriber
- Subscription
- Processor

Project Reactor

- Flux
- Mono
- Subscription
- Operators