ReactiveCocoa 3

(some parts)

Motivation (kind of)

(from RAC readme)

Event streams unify all of Cocoa's common patterns for asynchrony and event handling, including:

- Delegate methods
- Callback blocks
- NSNotifications
- Control actions and responder chain events
- Futures and promises
- Key-value observing (KVO)

Events

```swift
/// Signals must conform to the grammar:
/// `Next* (Error | Completed | Interrupted)?`

enum Event<T, E : ErrorType> {
    case Next(Box.Box<T>)
    case Error(Box.Box<E>)
    case Completed
    case Interrupted
}
```

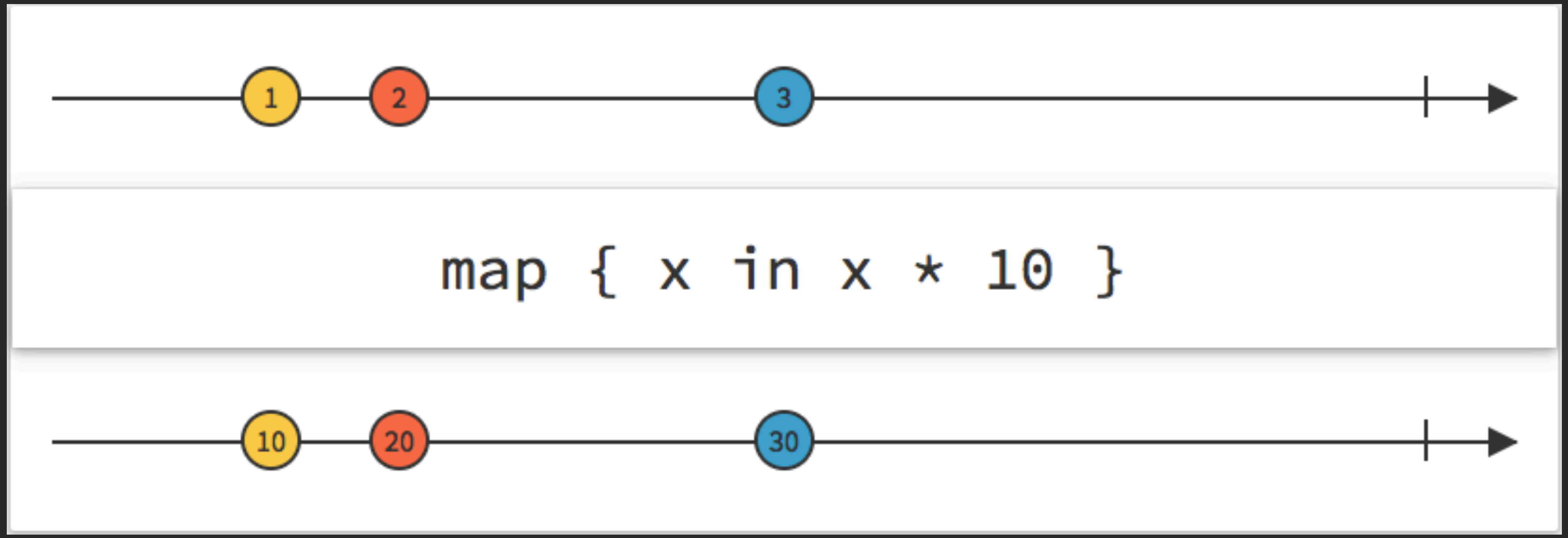
Signals

Demo

```
class Signal<T, E : ErrorType>

/// emits events after creation
/// (even without any observers)
/// good for streams of UI events
```

```swift
/// observe a signal with

func observe<T, E>(
    error: (E -> ())? = default,
    completed: (() -> ())? = default,
    interrupted: (() -> ())? = default,
    next: (T -> ())? = default
) ->
Signal<T, E> -> Disposable?
```

```
/// Maps each value in the signal to a new value.

func map<T, U, E>(transform: T -> U) ->
    Signal<T, E> -> Signal<U, E>
```



http://neilpa.me/rac-marbles/

```swift
/// pipe forward operator
/// Applies a Signal operator to a Signal.

func |><T, E, X>(
    signal: Signal<T, E>,
    transform: @noescape Signal<T, E> -> X)
    -> X
```

Signal Producers

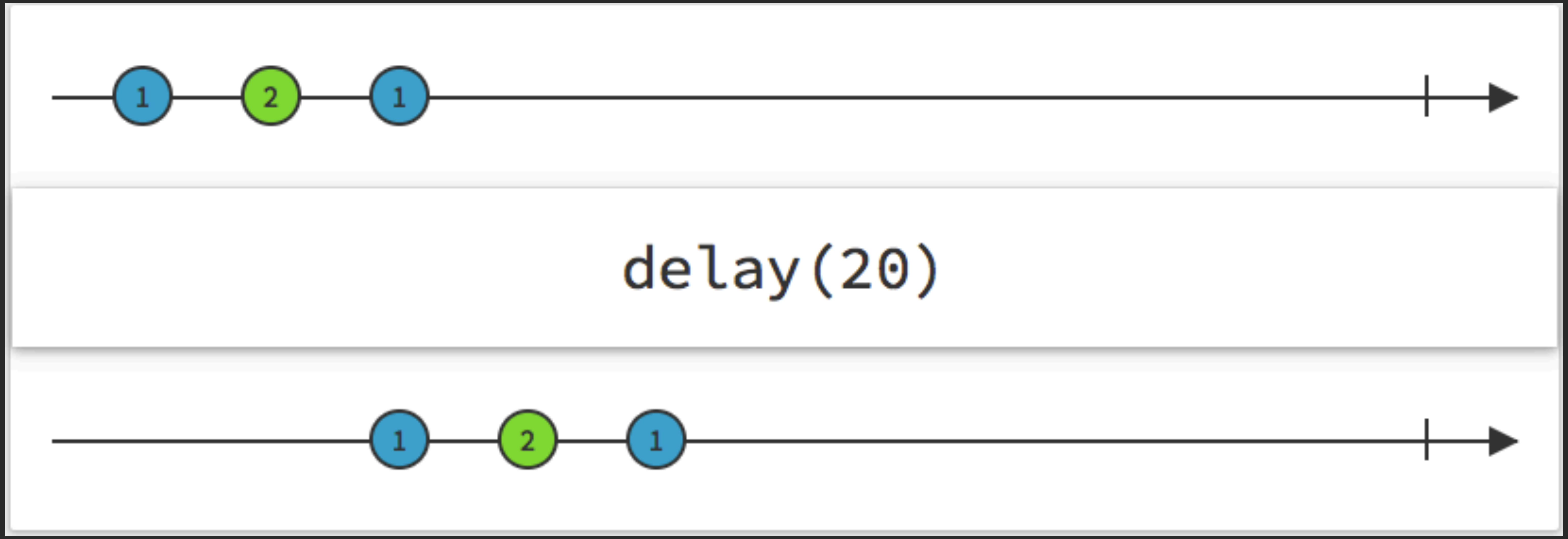
Demo

```swift
struct SignalProducer<T, E : ErrorType>

/// Signal producers are factories which create
signals when they are started.
/// good for network tasks
```

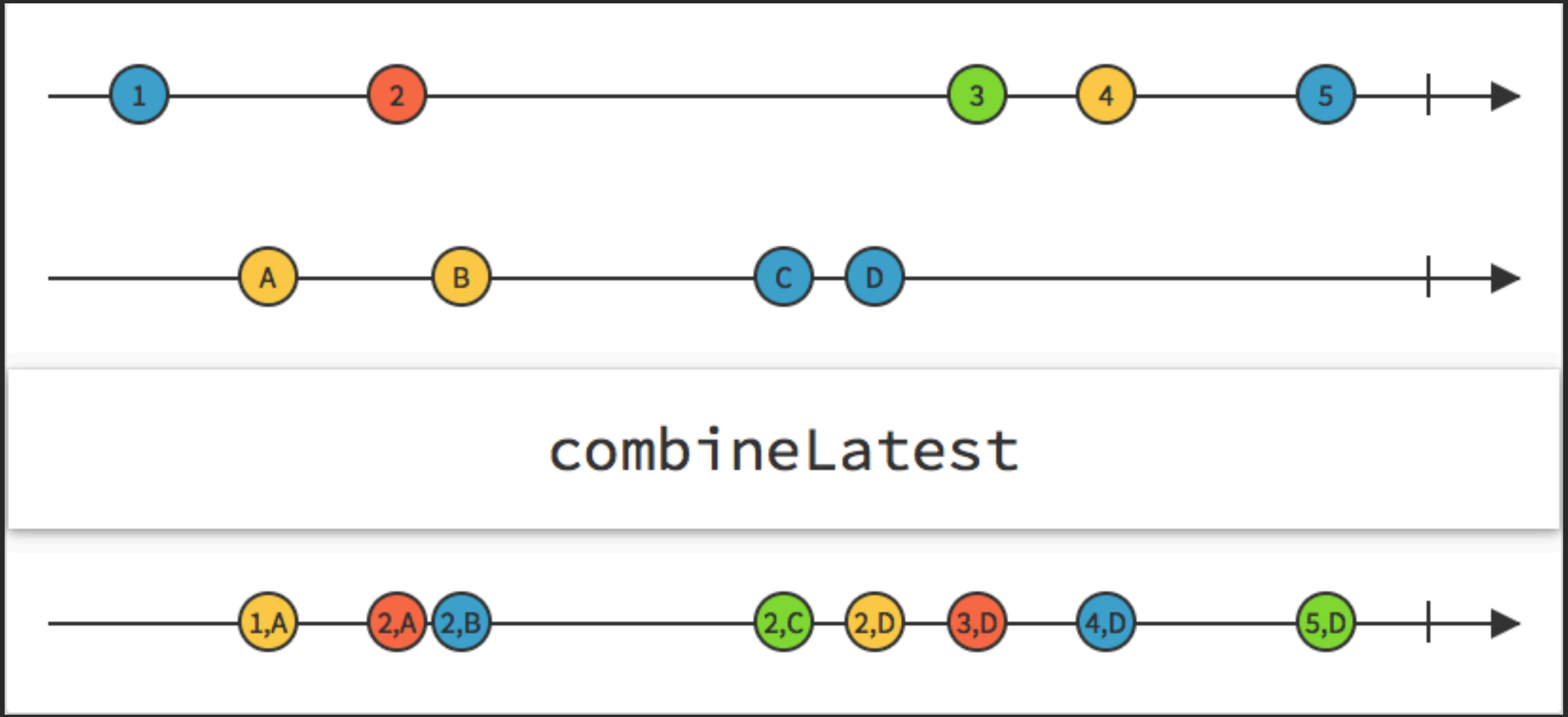
```swift
func start<T, E>(
    error: (E -> ())? = default,
    completed: (() -> ())? = default,
    interrupted: (() -> ())? = default,
    next: (T -> ())? = default
) ->
SignalProducer<T, E> -> Disposable
```

```swift
/// Ignores errors up to `count` times.
func retry<T, E>(count: Int)
-> SignalProducer<T, E> -> SignalProducer<T, E>
```

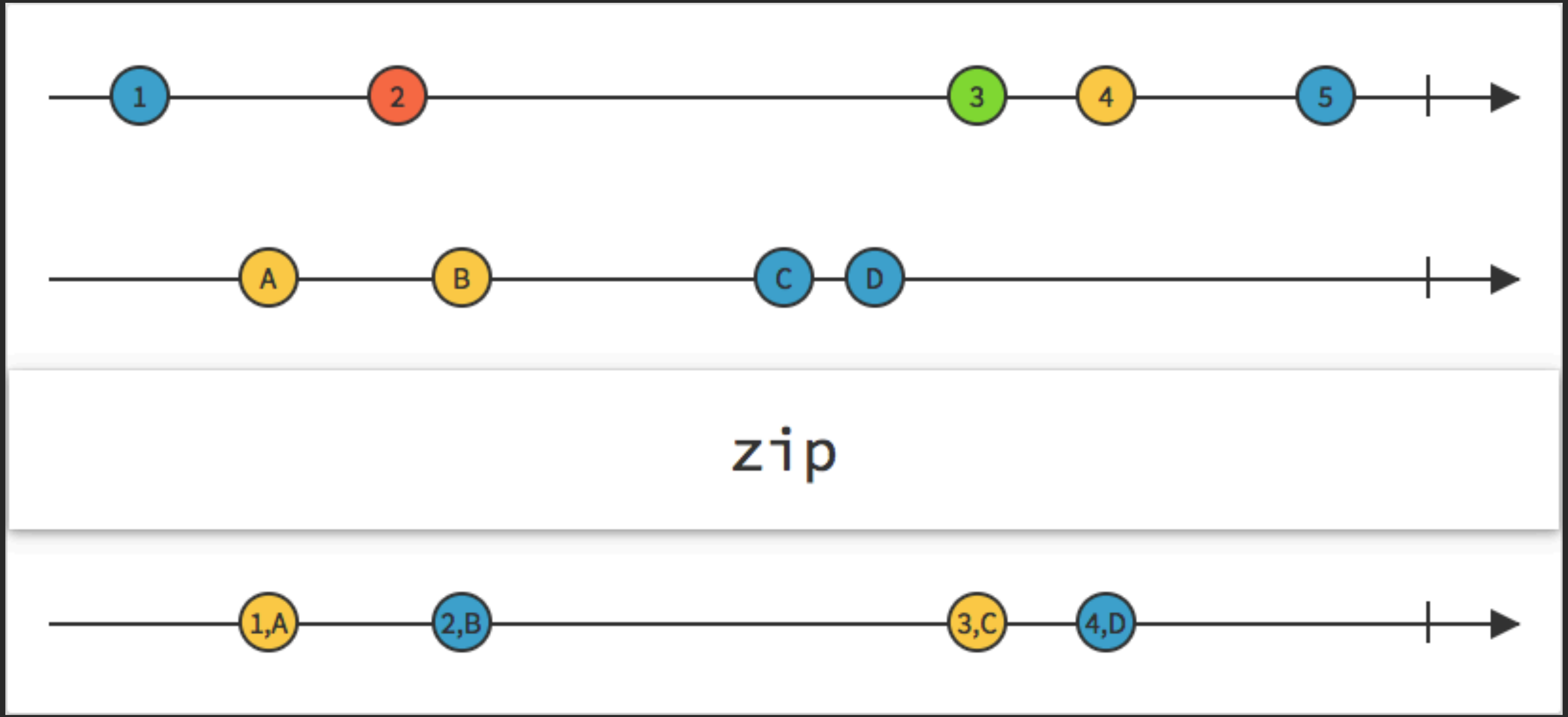
Operators



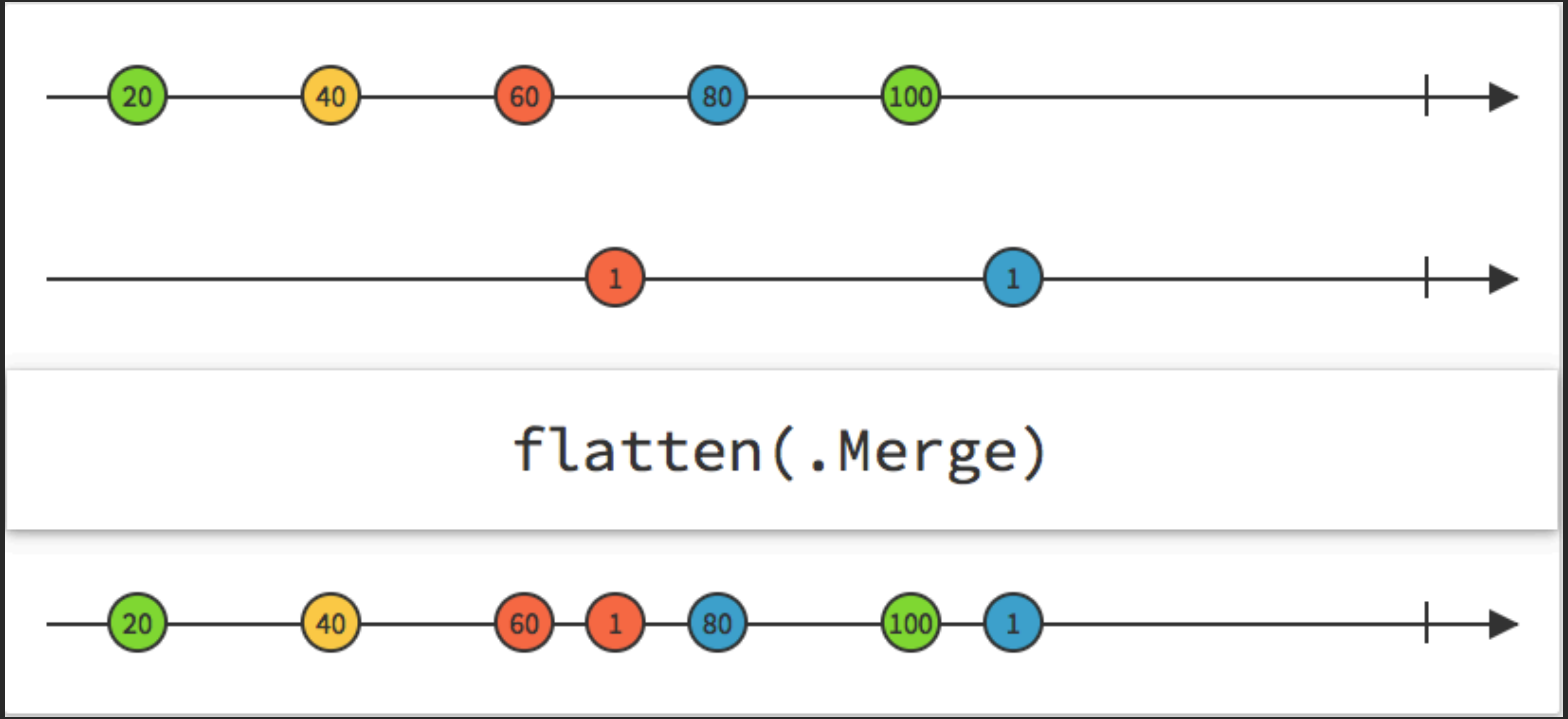
http://neilpa.me/rac-marbles/



http://neilpa.me/rac-marbles/



http://neilpa.me/rac-marbles/



http://neilpa.me/rac-marbles/

When to use it?

Links

https://github.com/ReactiveCocoa/ReactiveCocoa

https://github.com/ReactiveCocoa/ReactiveCocoa/tree/master/Documentation

https://skillsmatter.com/skillscasts/5906-reactivecocoa-and-swift-better-together
(highly recommended)

http://blog.scottlogic.com/2015/04/24/first-look-reactive-cocoa-3.html

http://blog.scottlogic.com/2015/04/28/reactive-cocoa-3-continued.html

http://blog.scottlogic.com/2015/05/15/mvvm-reactive-cocoa-3.html