GitHub 앱 만들기

1 인트로 (완성앱 & 구현 기능 소개)

명령형과 선언형

“어떻게”

명령형

“무엇을”

선언형

RxSwift

Bindings

```
Observable
    .combineLatest(
        firstName.rx.text,
        lastName.rx.text
    ) { $0 + " " + $1 }
    .map { "Greetings, \($0)" }
    .bind(to: greetingLabel.rx.text)
    .disposed(by: disposeBag)
```

코드 출처: https://github.com/ReactiveX/RxSwift/blob/main/Documentation/Why.md#bindings

재시도

```
func doSomethingIncredible(forWho: String) throws ->
IncredibleThing
```

코드 출처: https://github.com/ReactiveX/RxSwift/blob/main/Documentation/Why.md#retries

재시도

```
doSomethingIncredible(“me")
    .retry(3)
```

코드 출처: https://github.com/ReactiveX/RxSwift/blob/main/Documentation/Why.md#retries

Delegate

```
public func scrollViewDidScroll(scrollView: UIScrollView) { [weak self]
    self?.leftPositionConstraint.constant = scrollView.contentOffset.x
}
```

Delegate

```
self.resultsTableView.rx.contentOffset
    .map { $0.x }
    .bind(to: self.leftPositionConstraint.rx.constant)
```

비동기 API들

- Notification Center
- The delegate pattern
- Grand Central Dispatch(GCD)
- Closures

비동기 API들

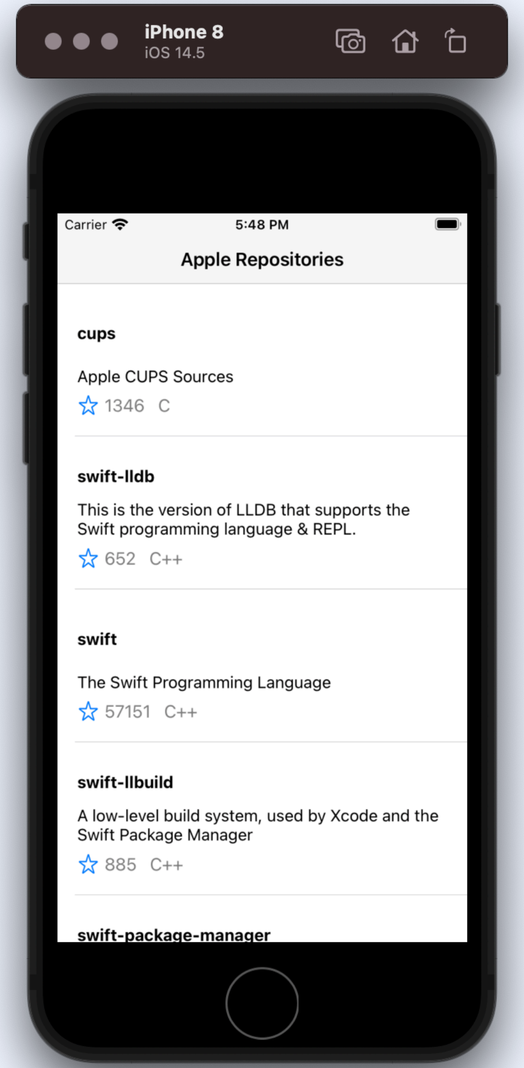
```
override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)

        makeUI()
        connectUIControls()
        fetchData()
        checkForChanges()
}
```

Benefits

- Composable
- Reusable
- Declarative
- Understandable and concise
- Stable
- Less stateful
- Without leaks

GitHub Repository App



GitHub 앱 만들기

2 RxSwift 알아보기

RxSwift

기본 개념

Every `Observable` instance is just a sequence

구성 요소

- Observable
- Operator
- Scheduler

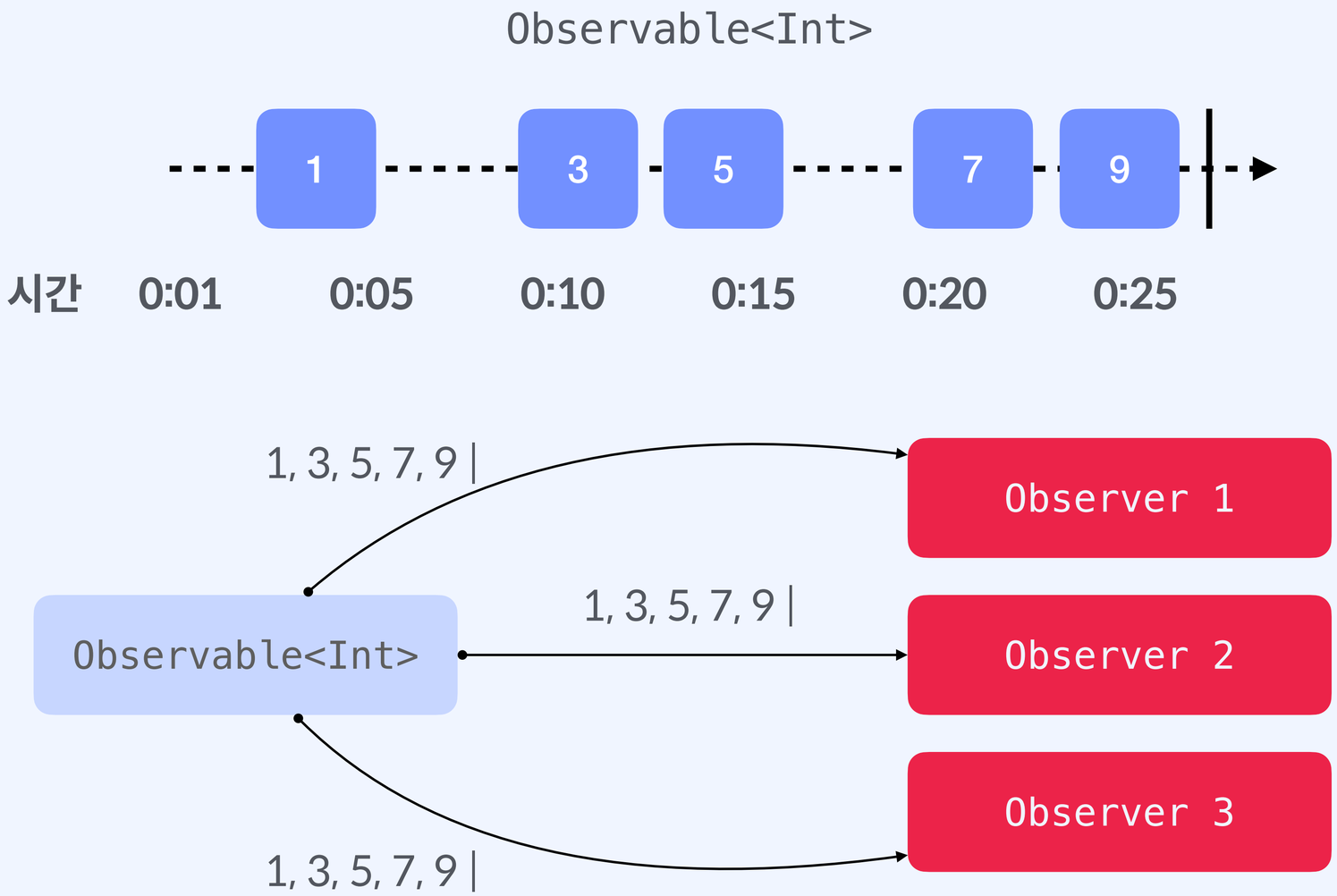
Observable

`Observable<T>`

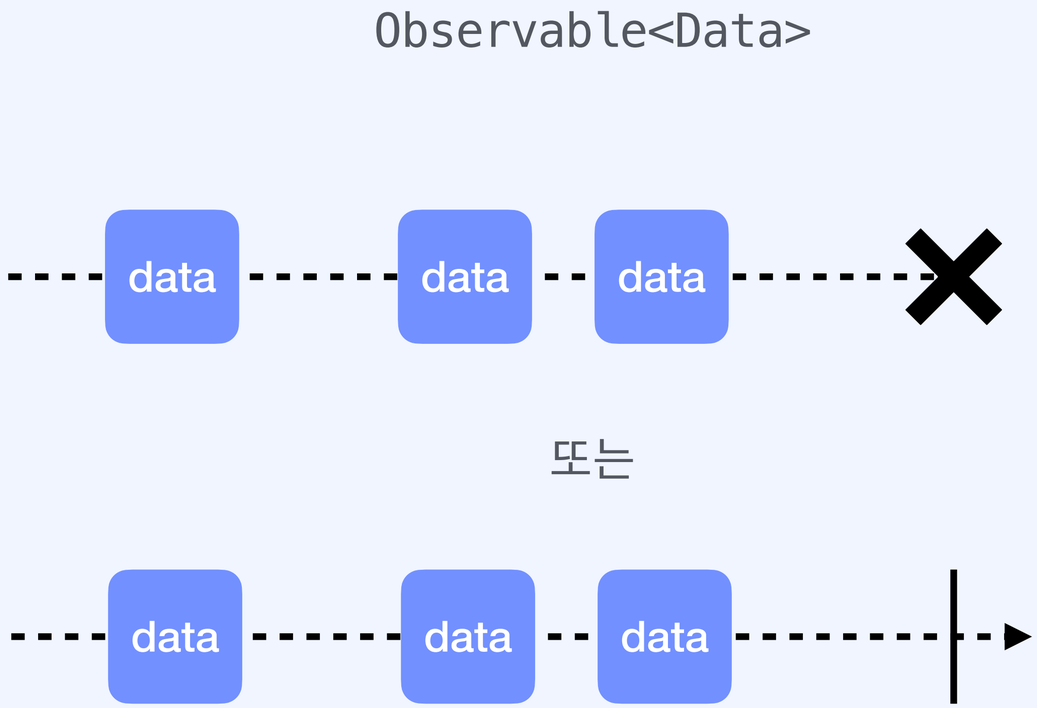
- Rx 코드의 기반
- T 형태의 데이터 snapshot을 '전달' 할 수 있는 일련의 이벤트를 비동기적으로 생성하는 기능
- 하나 이상의 observers가 실시간으로 어떤 이벤트에 반응
- 세 가지 유형의 이벤트만 방출

```
enum Event<Element>  {
    case next(Element)       // next element of a sequence
    case error(Swift.Error) // sequence failed with error
    case completed           // sequence terminated successfully
}
```

Observable



Finite Observable



Finite Observable

```
Network.download(file: "https://www...")
    .subscribe(onNext: { data in
        //임시 파일에 데이터 추가
    },
    onError: { error in
        //사용자에게 에러 표현
    },
    onCompleted: {
        //다운로드 된 파일 사용
    })
```

Infinite Observable

```
UIDevice.rx.orientation
    .subscribe(onNext: { current in
        switch current {
        case .landscape:
            // 가로모드 배치
        case .portrait:
            // 세로모드 배치
        }
    })
```

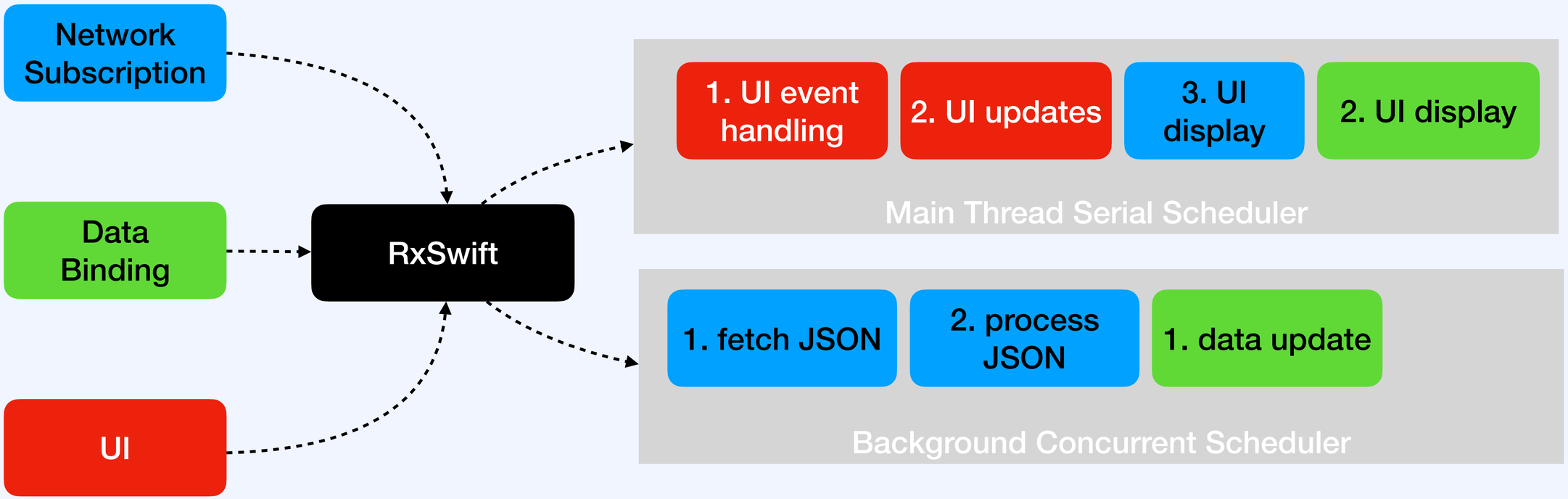
Operator

```
( 2 + 5 ) * 10 - 8
```

Operator

```
UIDevice.rx.orientation
    .filter { value in
        return value != .landscape
    }
    .map { _ in
        return "세로로만 볼거예요!"
    }
    .subscribe(onNext: { string in
        showAlert(text: string)
    })
```

Scheduler



GitHub 앱 만들기

3 RxSwift 설치하기

CocoaPods

```
pod 'RxSwift', '6.2.0'
```

Swift Package Manager

```
https://github.com/ReactiveX/RxSwift.git
```

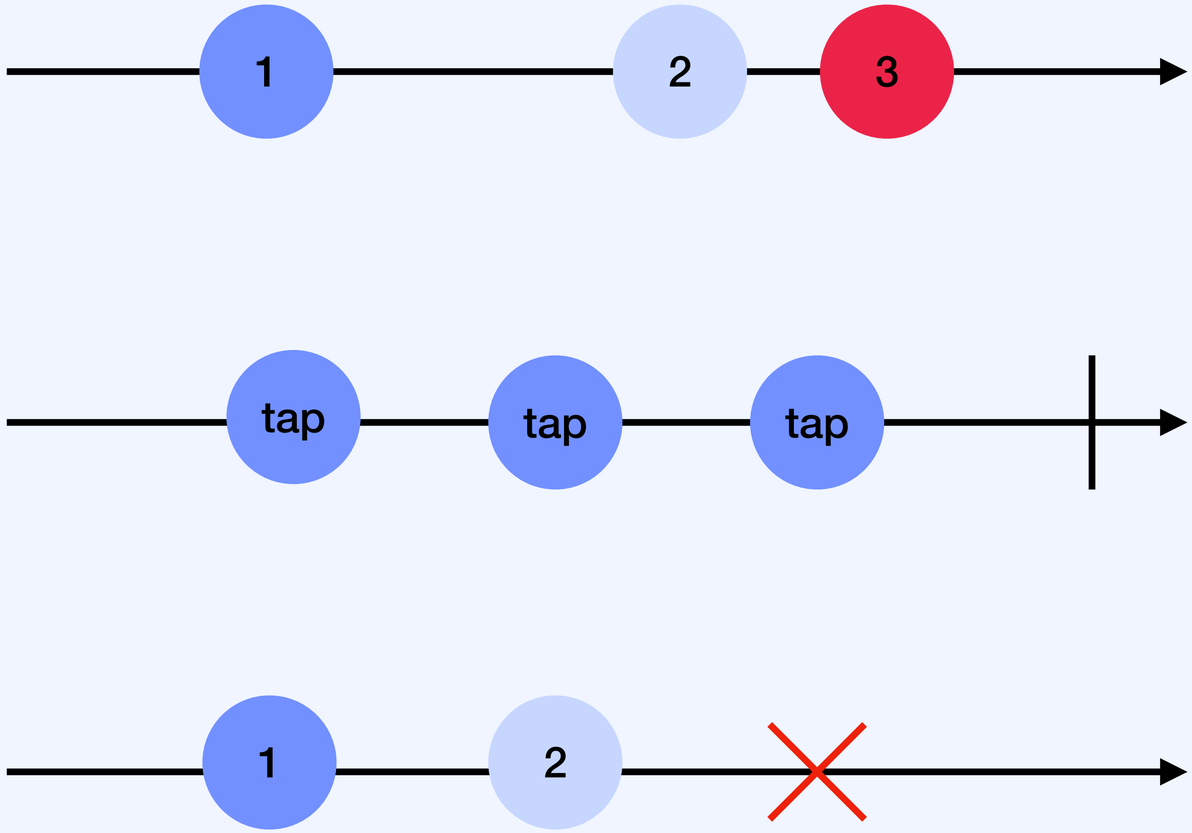
GitHub 앱 만들기

4 Observable 알아보기

Observable

- Rx의 심장
- Observable = Observable Sequence = Sequence
- 비동기적(asynchronous)
- Observable 들은 일정 기간 동안 계속해서 이벤트를 생성 (emit)
- marble diagram: 시간의 흐름에 따라서 값을 표시하는 방식
- 참고하면 좋을 사이트: RxMarbles

Observable 생명주기



Observable 생명주기

- Observable은 어떤 구성요소를 가지는 next 이벤트를 계속해서 방출할 수 있다.
- Observable은 error 이벤트를 방출하여 완전 종료될 수 있다.
- Observable은 complete 이벤트를 방출하여 완전 종료 될 수 있다.

Observable 생명주기

```
/// Represents a sequence event.
///
/// Sequence grammar:
/// **next\* (error | completed)**
@frozen public enum Event<Element> {
    /// Next element is produced.
    case next(Element)

    /// Sequence terminated with an error.
    case error(Swift.Error)

    /// Sequence completed successfully.
    case completed
}
```

GitHub 앱 만들기

5 Subject 알아보기

Subject

Observable이자 Observer

Subject의 종류

- **PublishSubject**

  빈 상태로 시작하여 새로운 값만을 subscriber에 방출한다.

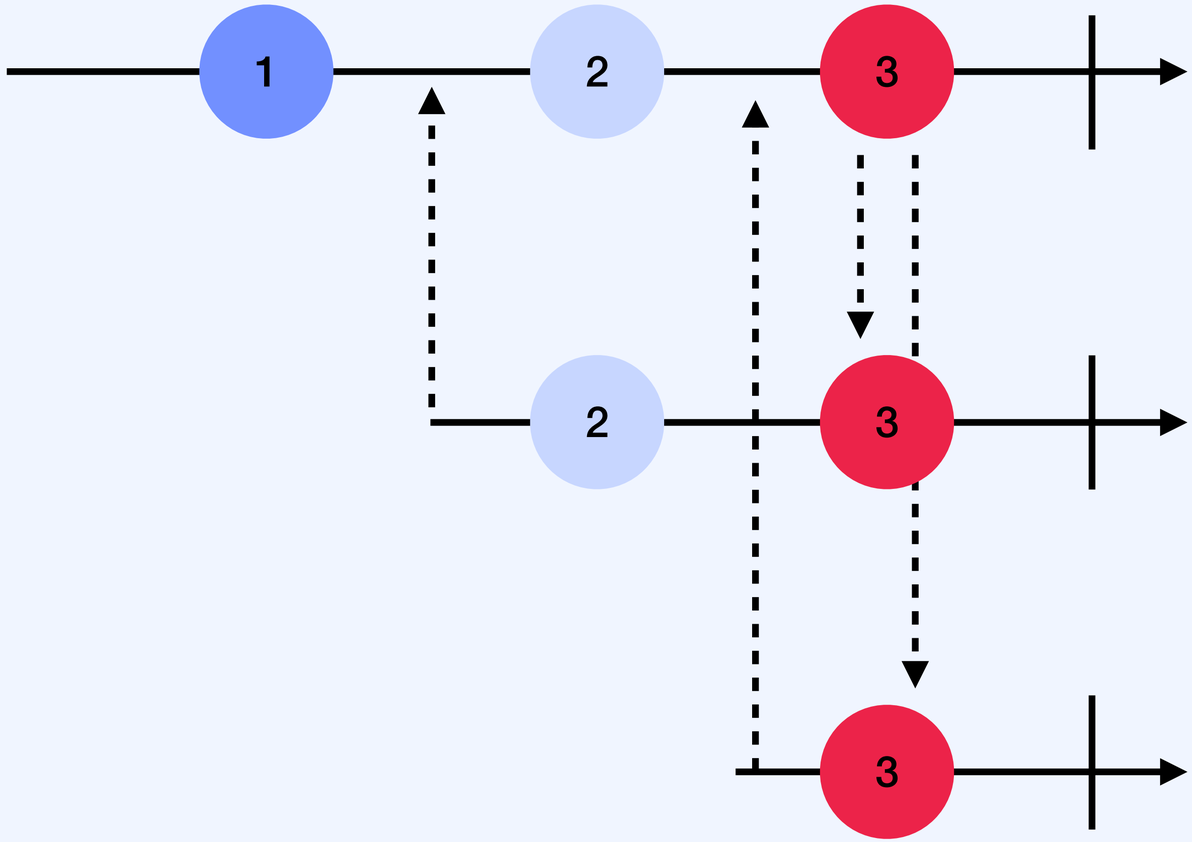
- **BehaviorSubject**

  하나의 초기값을 가진 상태로 시작하여, 새로운 subscriber에게 초기값 또는 최신값을 방출한다.

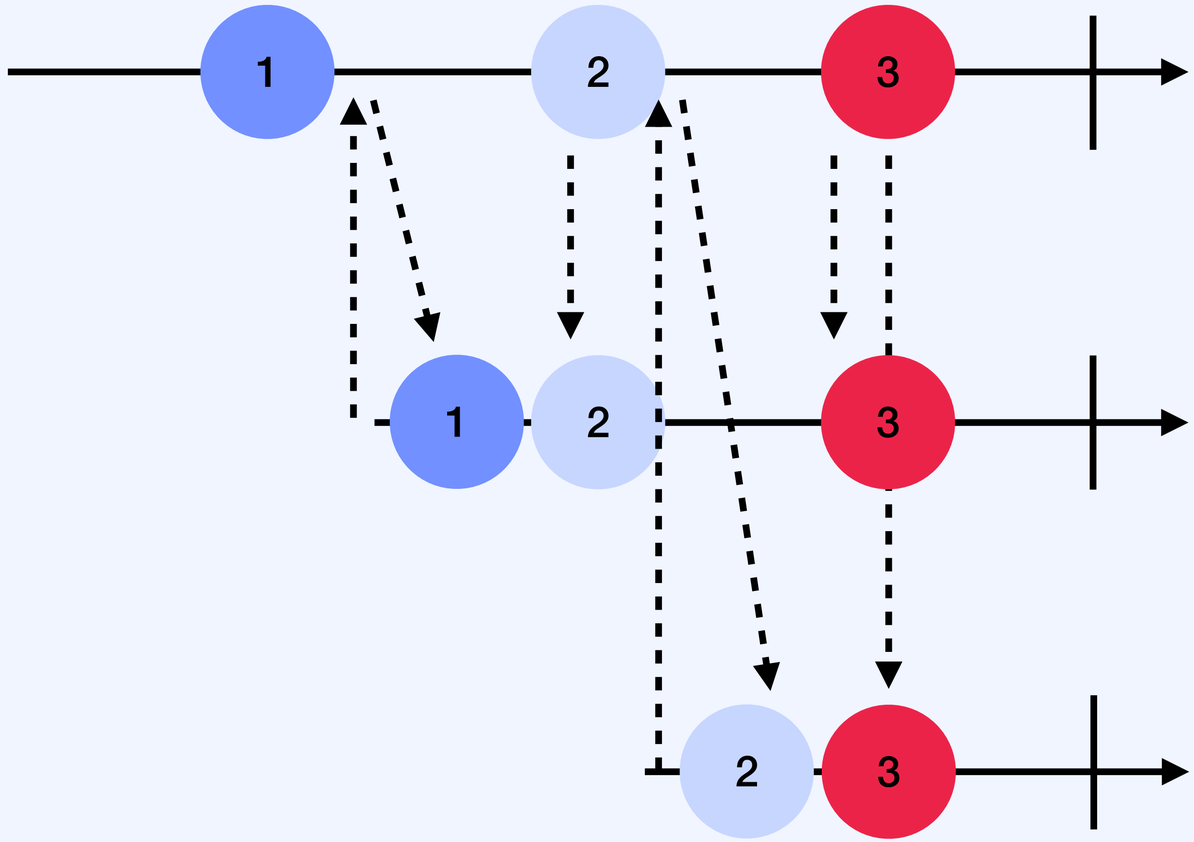
- **ReplaySubject**

  버퍼를 두고 초기화하며, 버퍼 사이즈 만큼의 값들을 유지하면서 새로운 subscriber에게 방출한다.

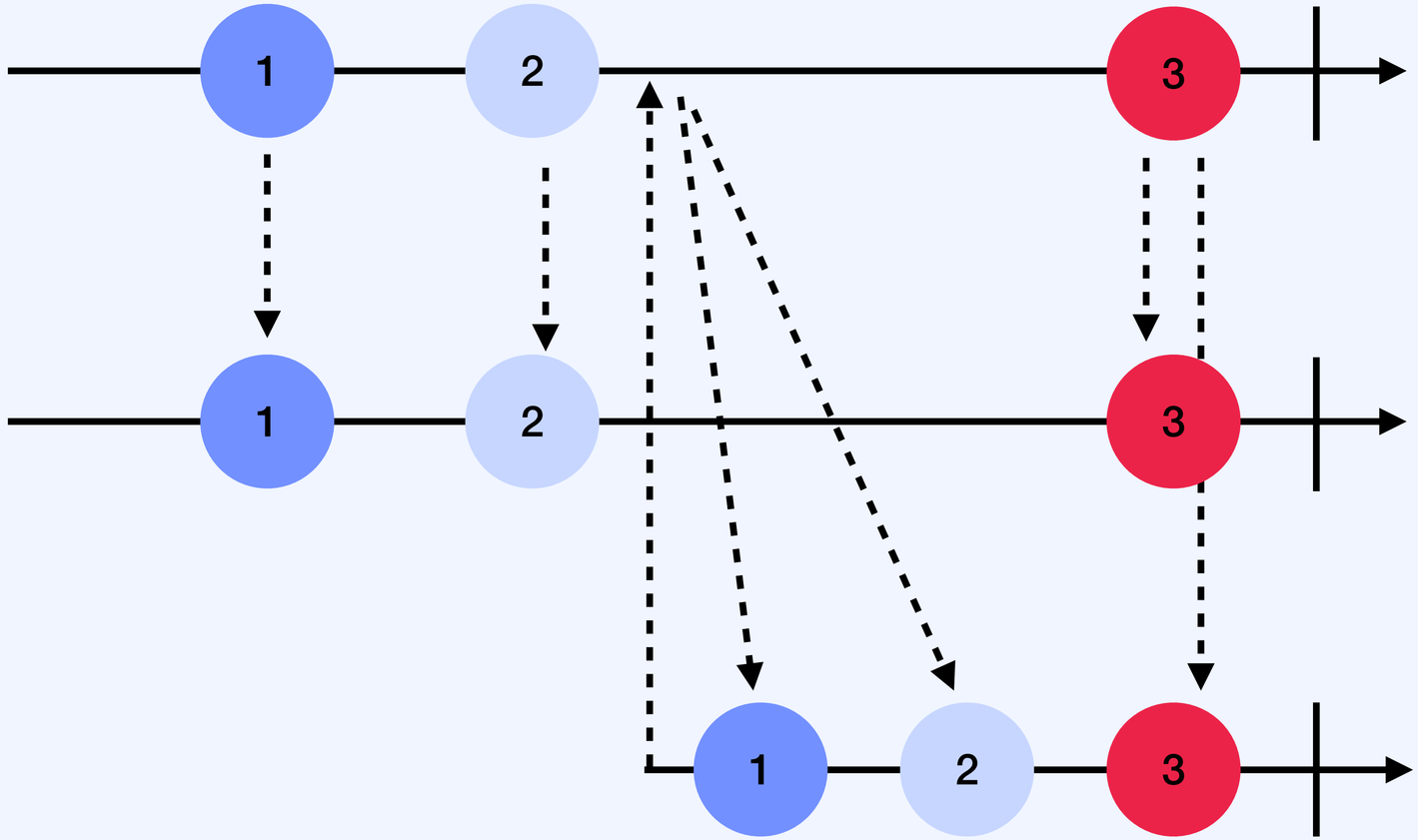
PublishSubject



BehaviorSubject



ReplaySubject



GitHub 앱 만들기

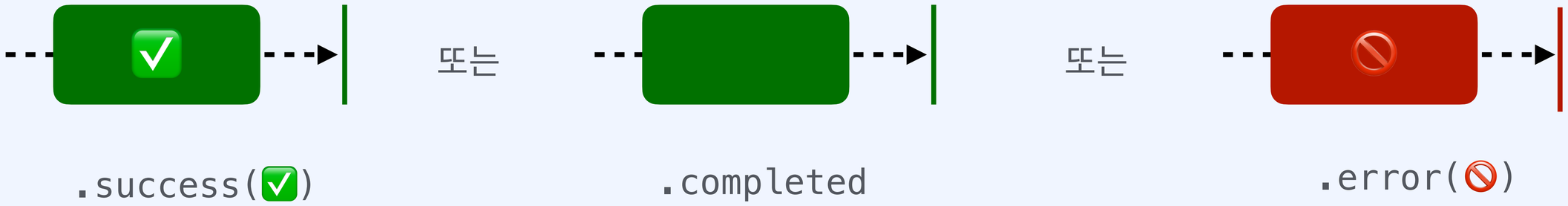
6 Single, Maybe, Completable 알아보기

Single

.success(✅)

.error(🚫)

Maybe



Completable

Single

.success(✅)

.error(🚫)

GitHub 앱 만들기

7 Filtering Operator

GitHub 앱 만들기

8 Transforming Operator

GitHub 앱 만들기

9 기본 UI 그리기

GitHub 앱 만들기

10 GitHub API 연결하기

GitHub 앱 만들기

11 아웃트로 (정리, 현업 예시)

정리

정리

가독성 높은 코드

선언형

RxKotlin, RxJava..

비동기

반응형

다양한 Operators

콜백지옥탈출

함수형