Scheduler

Study-Rx

Scheduler

어떤 프로그램의 세부 일정을 주관하는 관리자

Scheduler

- 스케줄러는 Rx 코드를 어느 스레드에서 실행할지 지정할 수 있다.
- subscribeOn() 함수와 observeOn() 함수를 모두 지정하면 Observable에서 데이터 흐름이 발생하는 스레드와 처리된 결과를 구독자에게 발행하는 스레드를 분리할 수 있다.
- subscribeOn() 함수만 호출하면 Observable의 모든 흐름이 동일한 스레드에서 실행된다.
- 스케줄러를 별도로 지정하지 않으면 현재(main) 스레드에서 동작을 실행한다.

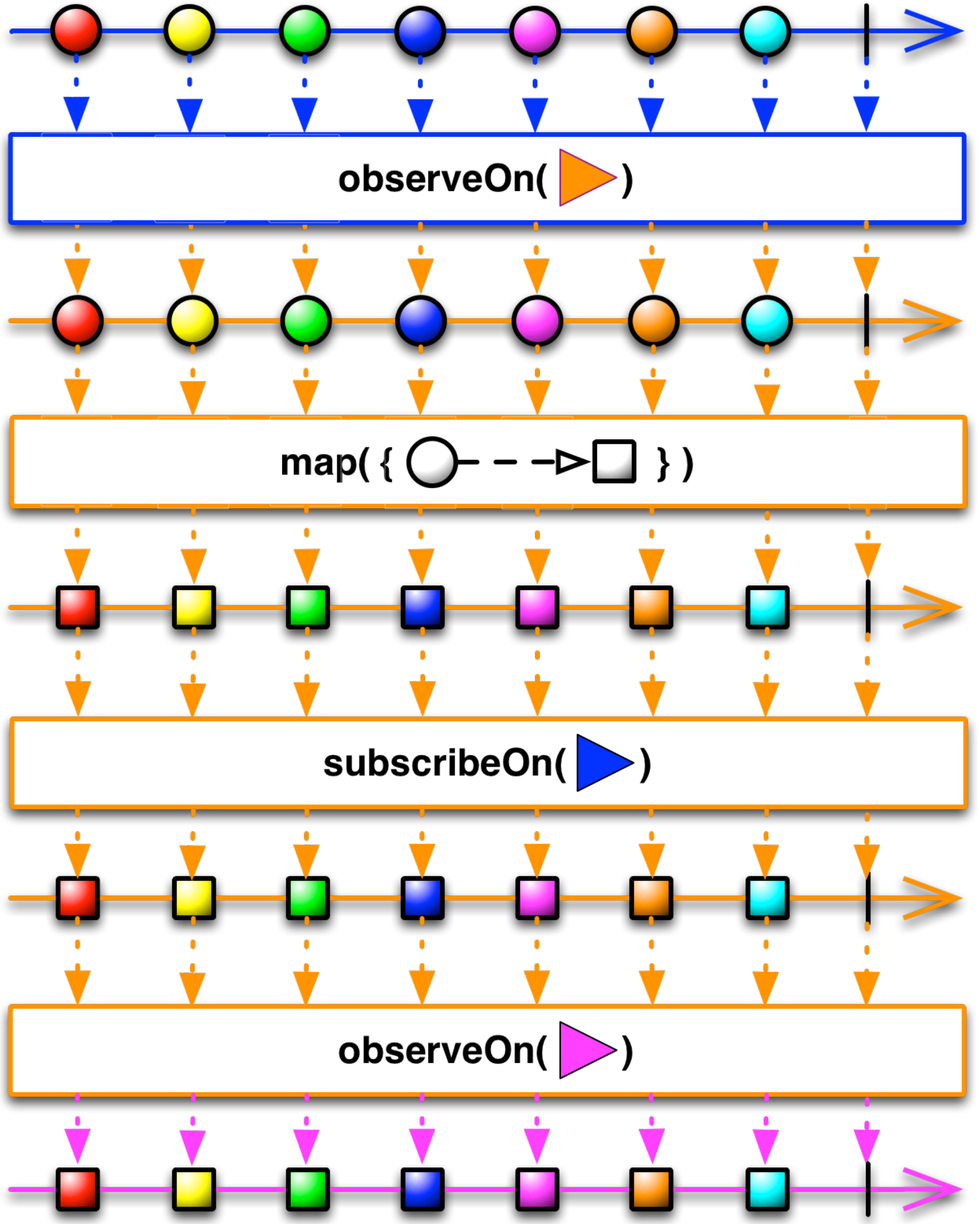
Marble Diagrams

subscribeOn() 함수

- 데이터 흐름을 발행하는 스레드를 지정
- 처리된 결과를 구독자에게 전달하는 스레드를 지정
- 처음 지정한 스레드를 고정시키므로 재호출 불가

observeOn() 함수

- 여러 번 호출할 수 있다
- 동작하는 스레드를 바꿀 수 있다.



RxJava

스케줄러	용도
`Schedulers.computation( )`	이벤트-루프와 콜백 처리 같은 연산 중심적인 작업을 위해 사용된다; 그렇기 때문에 I/O를 위한 용도로는 사용하지 말아야 한다(대신 `Schedulers.io( )`를 사용); 기본적으로 스레드의 수는 프로세서의 수와 같다
`Schedulers.from(executor)`	명시한 `Executor`를 스케줄러로 사용한다
`Schedulers.immediate( )`	현재 스레드에서 즉시 실행할 작업을 스케줄링 한다
`Schedulers.io( )`	블러킹 I/O의 비동기 연산 같은 I/O 바운드 작업을 처리한다. 이 스케줄러는 필요한 만큼 증가하는 스레드-풀을 통해 실행된다; 일반적인 연산이 필요한 작업은 `Schedulers.computation( )`를 사용하면 된다; 기본적으로 `Schedulers.io( )`이며 `CachedThreadScheduler`로, `CachedThreadScheduler`는 스레드 캐싱을 사용하는 새로운 스레드 스케줄러라고 생각하면 된다
`Schedulers.newThread( )`	각각의 단위 작업을 위한 새로운 스레드를 생성한다
`Schedulers.trampoline( )`	대기 중인 큐를 처리한 후에 현재 스레드에서 실행 될 작업 큐를 만든다

RxSwift

- CurrentThreadScheduler (Serial) : 현재 스레드에 있는 작업의 단위들을 스케줄 해준다.
- MainScheduler (Serial) : 메인 스레드에서 실행되어야 하는 스케줄러로 UI 작업에 쓰인다.
- SerialDispatchQueueScheduler (Serial) : 특정 DispatchQueue에서 실행되어야 하는 추상적인 작업에서 사용한다.
- ConcurrentDispatchQueueScheduler (Concurrent) :
  - 특정 DispatchQueue에서 실행되어야 하는 스케줄러로 추상적인 작업에서 사용하며 시리얼 디스패치 큐에도 보낼 수 있다
  - 백그라운드 작업에 적합하다.
- OperationQueueScheduler (Concurrent) :
  - NSOperationQueue에서 실행되어야 하는 스케줄러로 추상적인 작업에서 사용한다.
  - maxConcurrentOperationCount를 이용하여 컨커런트 처리과정을 미세 조정하고 싶을 때 적합하다.