Моделювання бізнес процесів засобами реактивного програмування

Студент:
Проскурня Дмитро

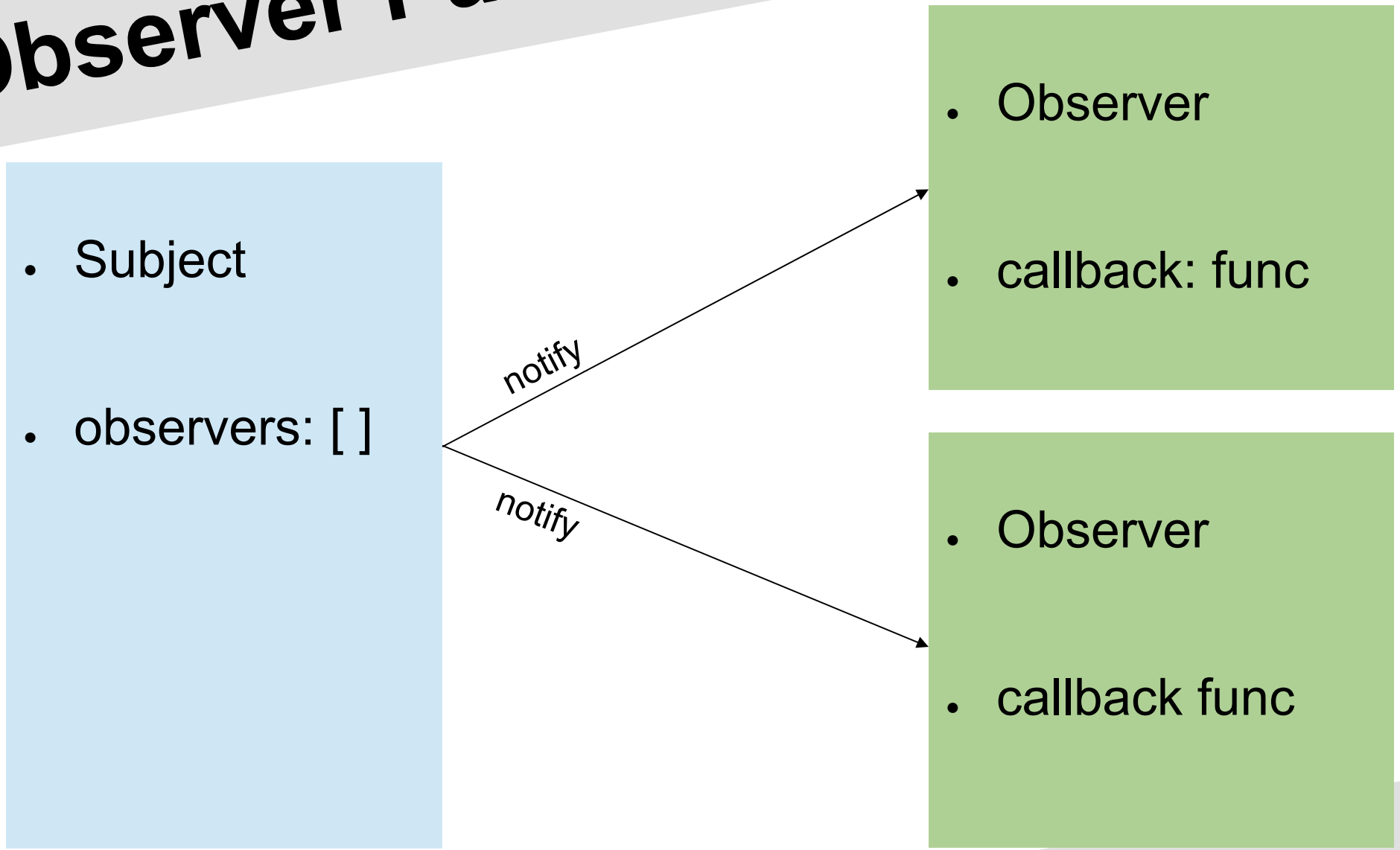
Науковий керівник:
Жежерун О.П.

pull/push стратегії

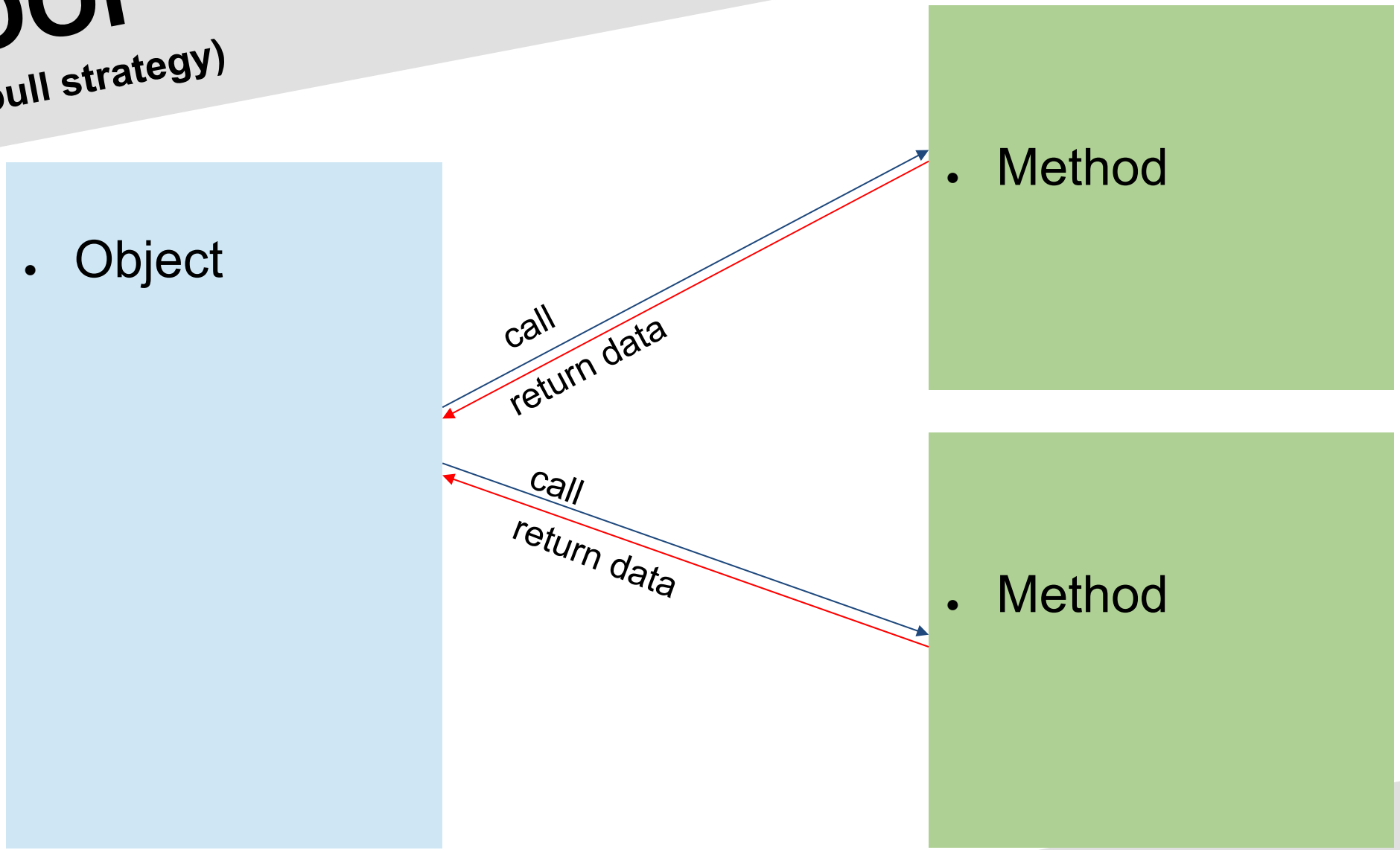
Rx Extensions

Visual Reactor

Observer Pattern

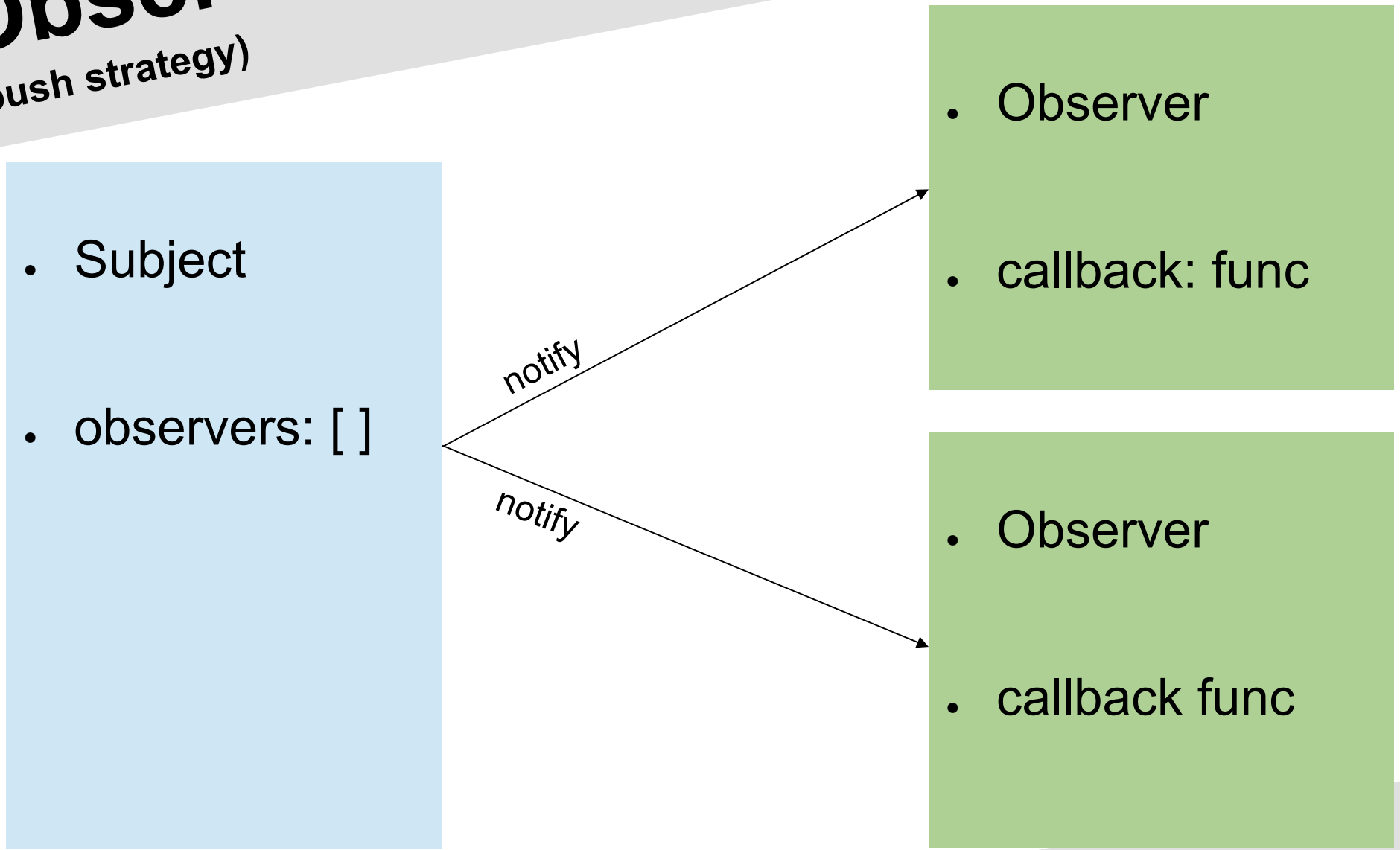


OOP
(pull strategy)



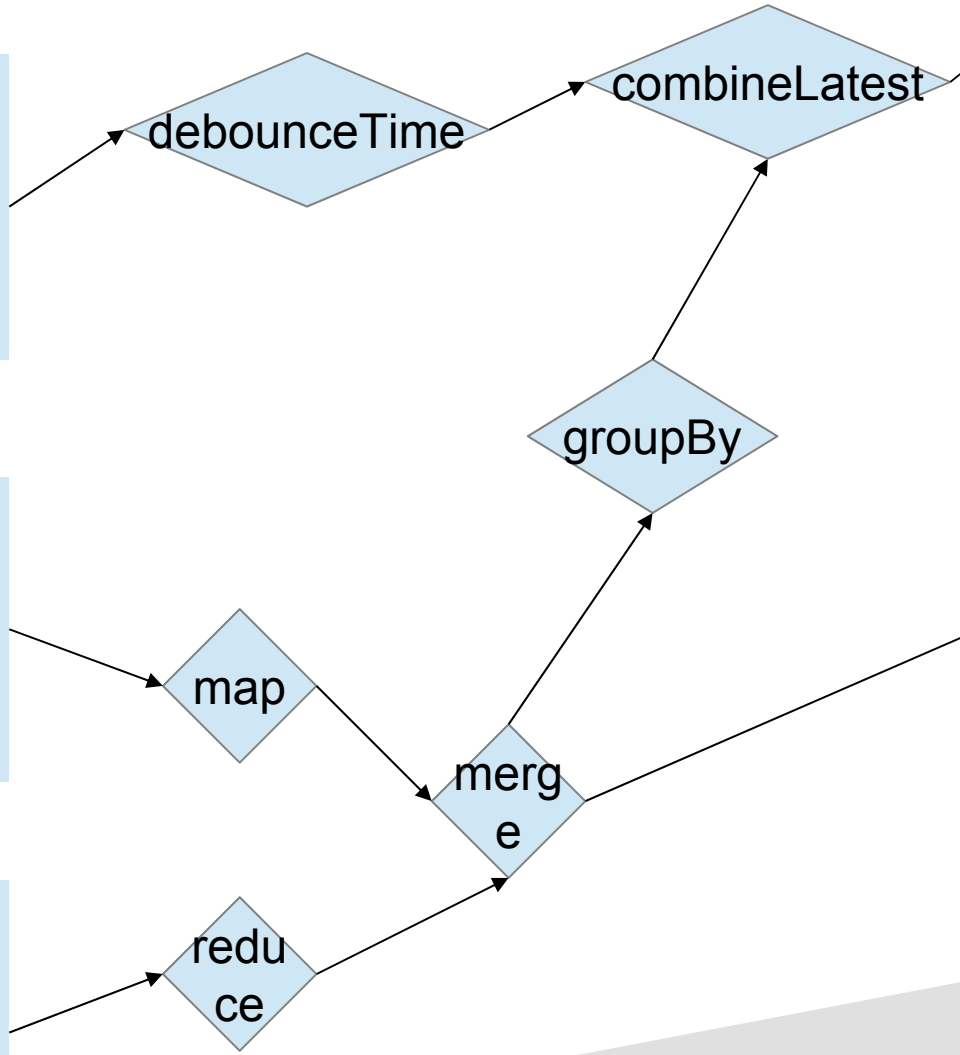
Observer Pattern

(push strategy)



Rx Extensions

- Subject
- Behavior Subject
- Replay Subject



- Observer
  - next: func
  - error: func
  - complete: func

- Observer
  - next: func
  - error: func
  - complete: func

Reactive Streams

- Інтерфейс асинхронних потоків даних з контролем завантаженості підписників.

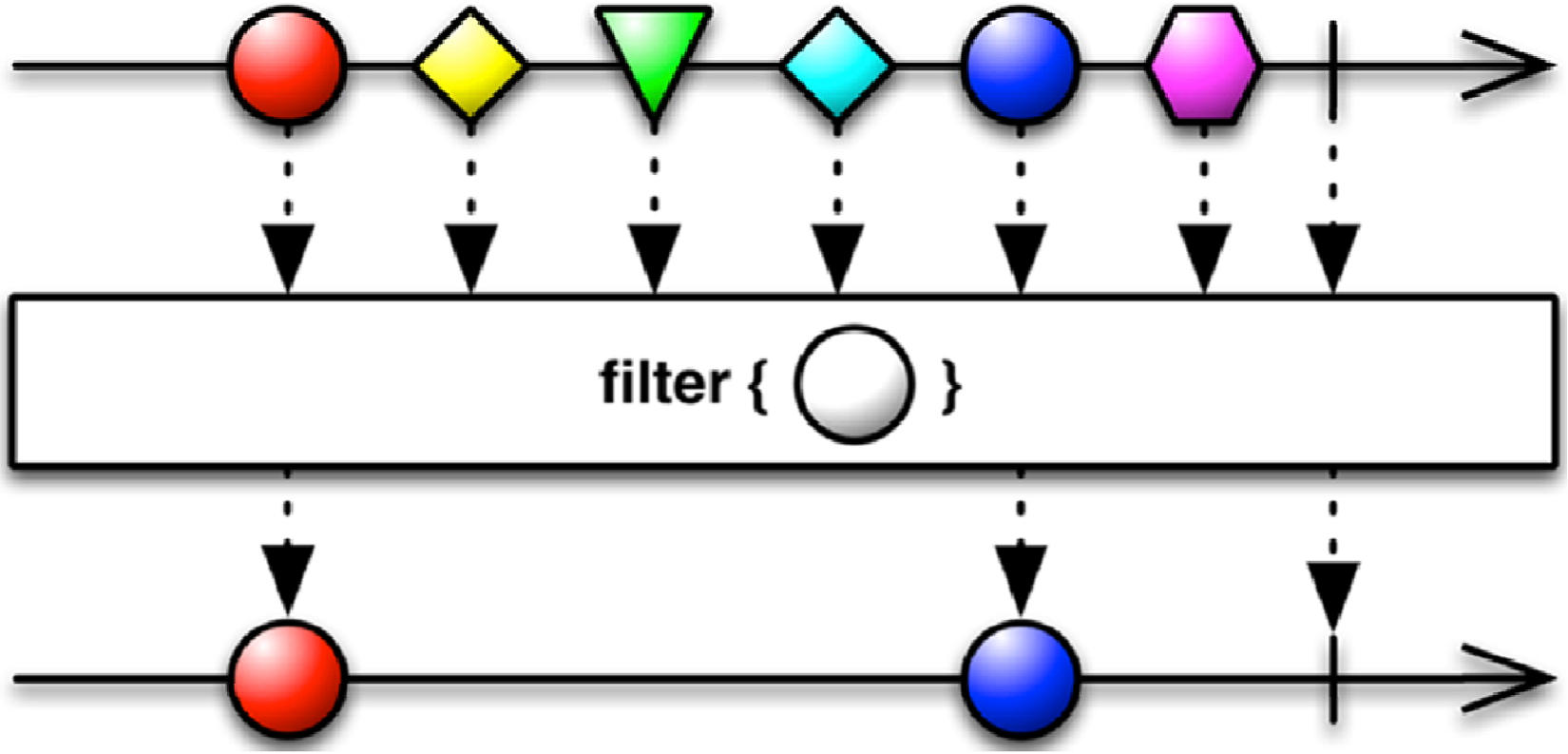
- Імплементації: Akka Streams, MongoDB, Ratpack, Reactive Rabbit – driver for RabbitMQ/AMQP, Spring and Pivotal Project Reactor, Netflix RxJava, Slick 3.0, Vert.x 3.0, Cassandra, Elasticsearch, Apache Kafka, Parallel Universe Quasar, Play Framework, Armeria...

Rx Operators

100+ операторів в RxJS

- оператори **створення** використовуються для ініціалізації потоків даних
- **трансформаційні** оператори змінюють тип елементів потоку
- оператори **фільтрації** відкидають з потоку непотрібні елементи
- **комбінаційні** операції поєднують окремі потоки в один
- оператори **обробки помилок**
- **умовні та булеві** оператори є широким класом операторів, які зокрема здійнсюють умовну фільтрацію елементів та аналіз відповідності потоку певному критерію
- **математичні та агрегаційні** оператори агрегують потік у єдине значення
- оператори **поширення** дозволяють розділяти один і той же потік між кількома підписниками
- оператори **контролю зворотнього тиску** дозволяють встановлювати стратегії для зменшення частоти виникнення подій потоку
- **утилітні** оператори темпорально керують потоком, встановлюючи затримки чи використовуючи планувальники

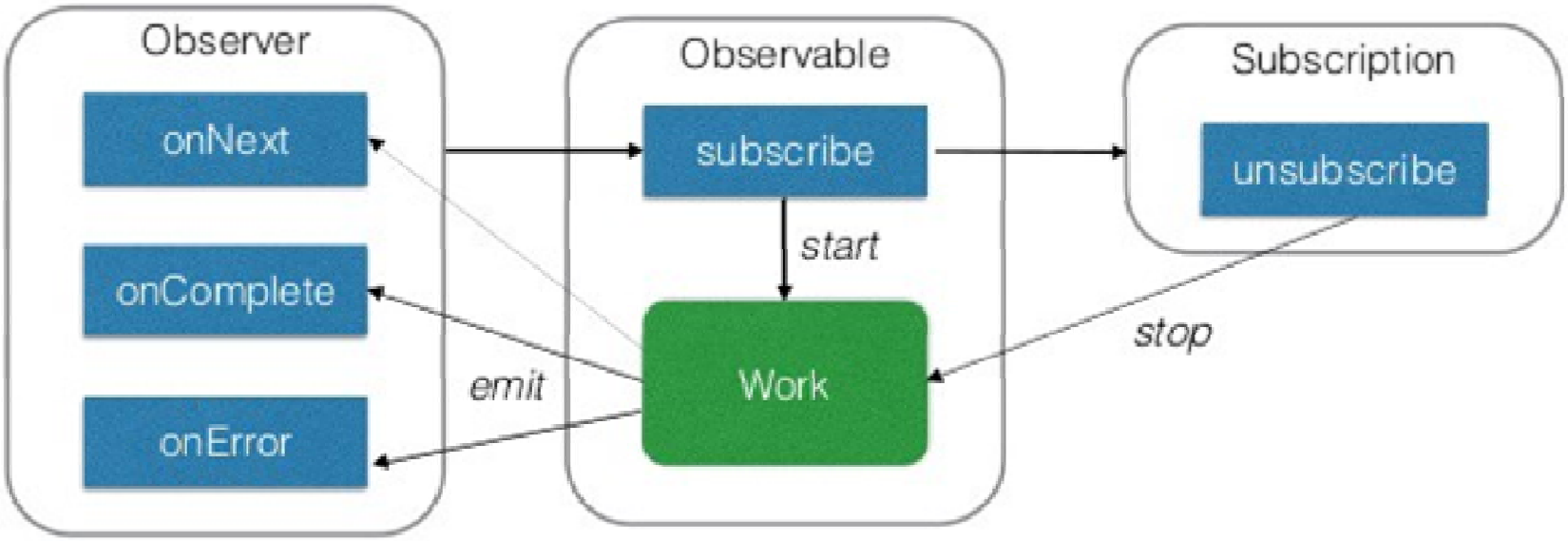
filter(predicate)



operators

- audit
- auditTime
- buffer
- bufferCount
- bufferTime
- bufferToggle
- bufferWhen
- catchError
- combineAll
- ~~combineLatest (deprecated)~~
- ~~concat (deprecated)~~
- concatAll
- concatMap
- concatMapTo
- count
- debounce
- debounceTime
- defaultIfEmpty
- delay
- delayWhen
- dematerialize
- distinct
- distinctUntilChanged
- distinctUntilKeyChanged
- elementAt
- endWith
- every
- exhaust
- exhaustMap
- expand
- filter
- finalize
- find
- findIndex
- first
- flatMap
- groupBy
- ignoreElements
- isEmpty
- last
- map
- mapTo
- materialize
- max
- ~~merge (deprecated)~~
- mergeAll
- mergeMap
- mergeMapTo
- mergeScan
- min
- multicast
- observeOn
- onErrorResumeNext
- pairwise
- ~~partition (deprecated)~~
- pluck
- publish
- publishBehavior
- publishLast
- publishReplay
- ~~race (deprecated)~~
- reduce
- refCount
- repeat
- repeatWhen
- retry
- retryWhen
- sample
- sampleTime
- scan
- sequenceEqual
- share
- shareReplay
- single
- skip
- skipLast
- skipUntil
- skipWhile
- startWith
- subscribeOn
- switchAll
- switchMap
- switchMapTo
- take
- takeLast
- takeUntil
- takeWhile
- tap
- throttle
- throttleTime
- throwIfEmpty
- timeInterval
- timeout
- timeoutWith
- timestamp
- toArray
- window
- windowCount
- windowTime
- windowToggle
- windowWhen
- withLatestFrom
- ~~zip (deprecated)~~
- zipAll

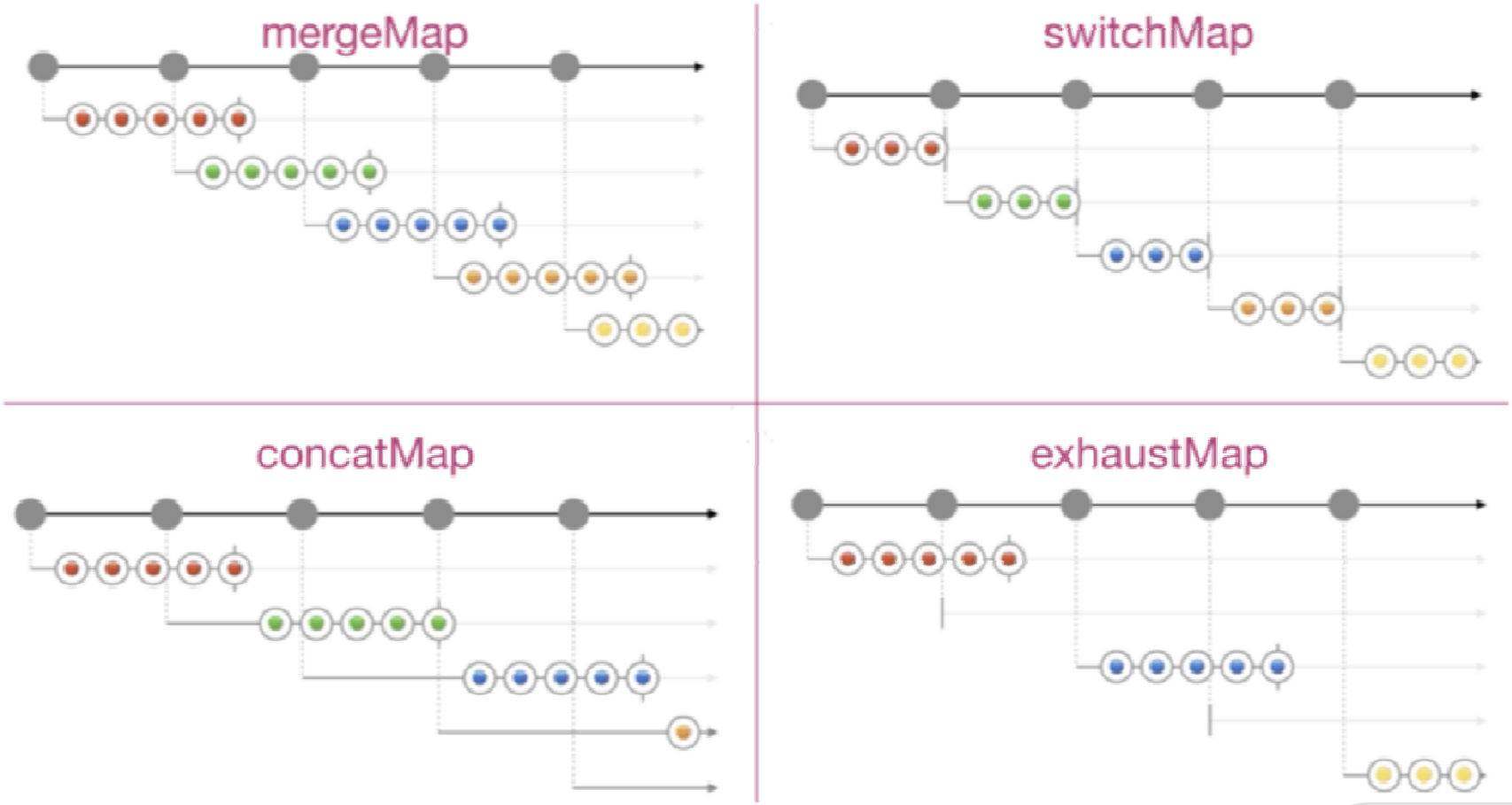
subscribe
(onNext, onError, onComplete)



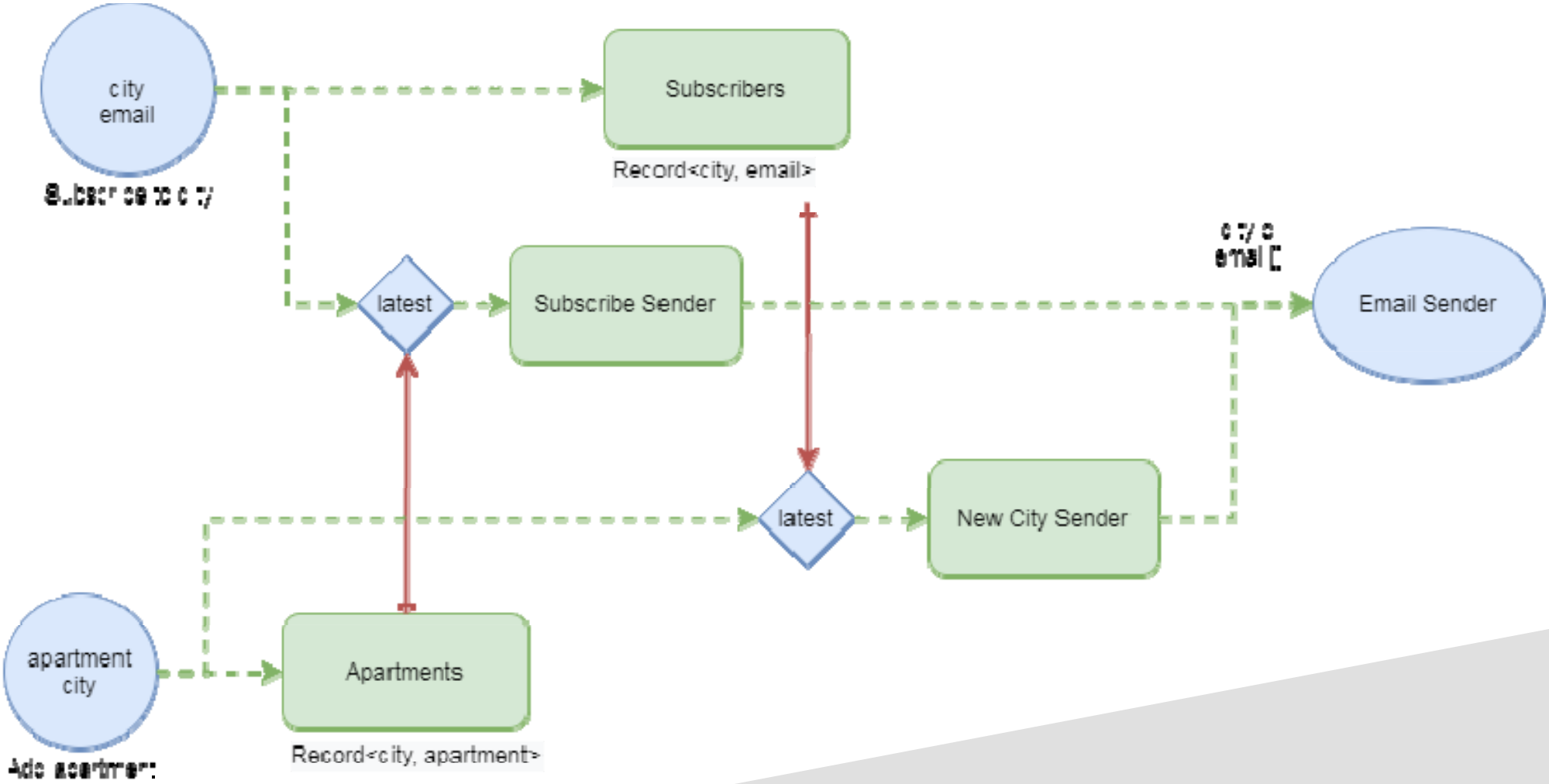
Переваги ReactiveX

- єдиний інтерфейс для синхронного і асинхронного виконання коду
- можливість отримання (розподіленого у часі) вектору даних, а не лише одиничних значень
- ініціативність замість виконання наказу
- вбудована підтримка протяжних у часі процесів
- дані незалежні від методів їх обробки
- безпечний рефакторинг

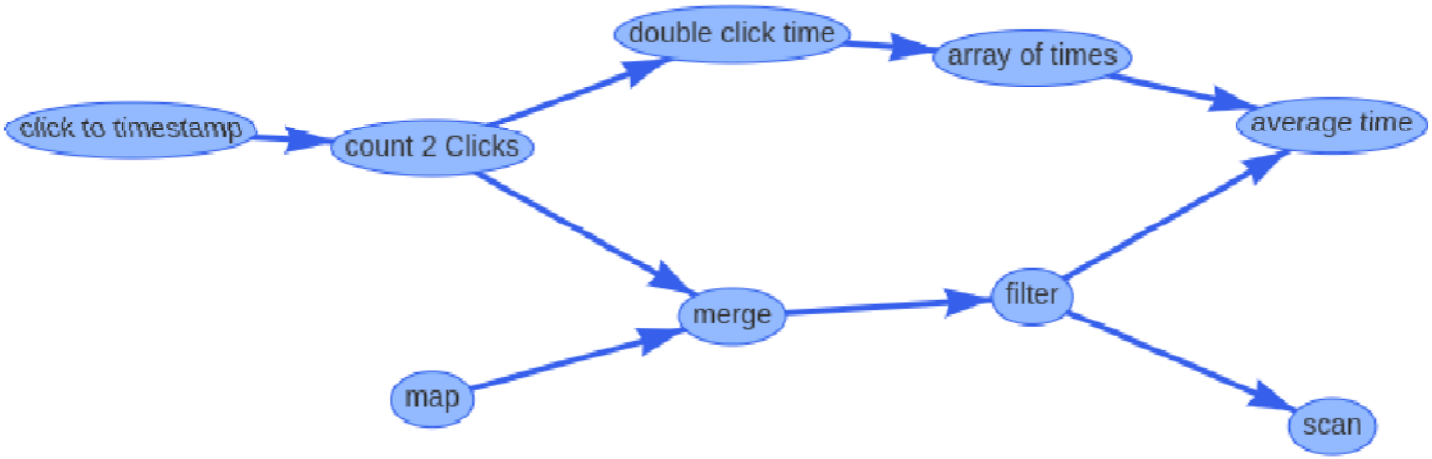
складно думати потоками



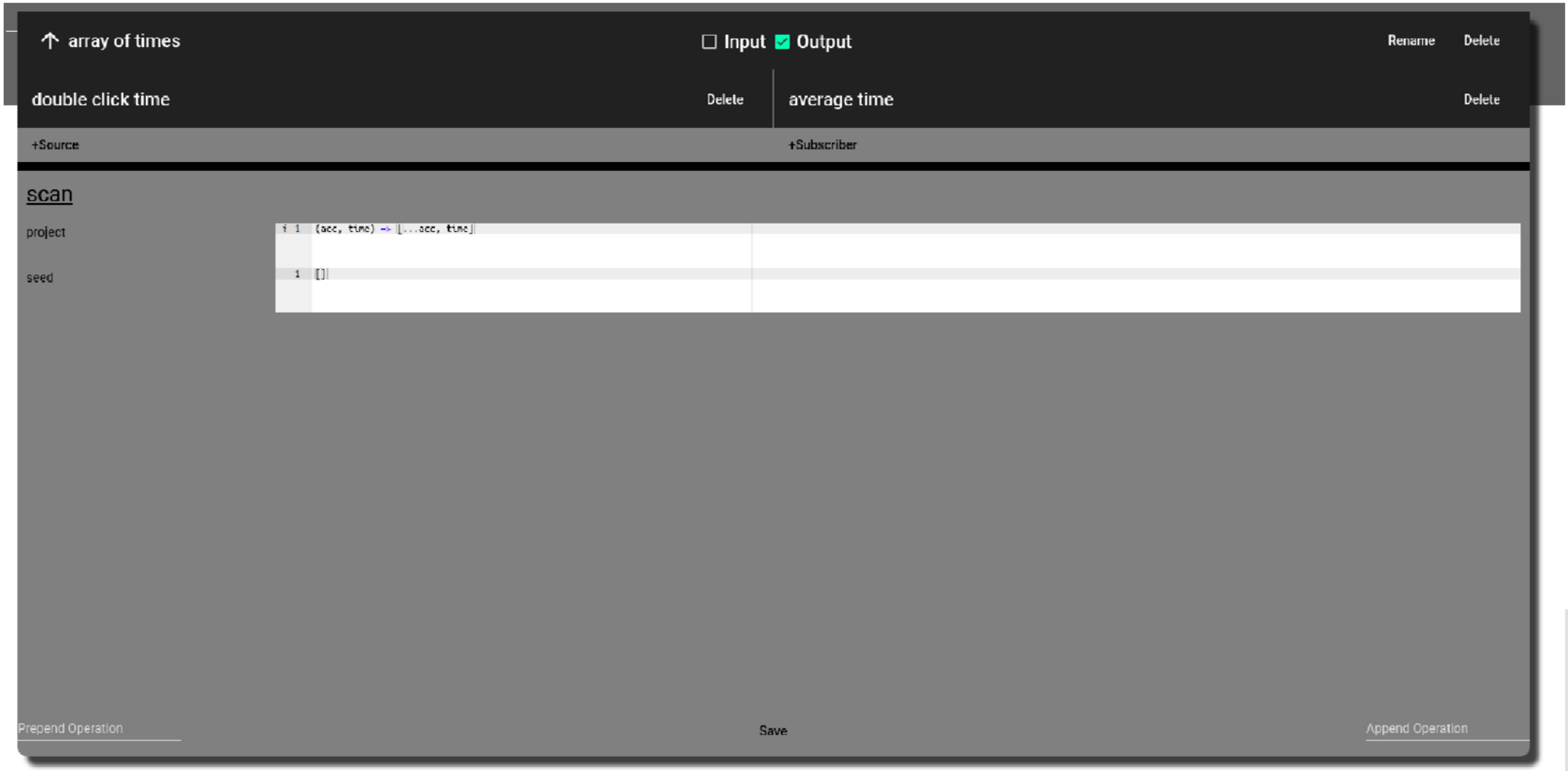
Visual Reactor



Visual Reactor



Visual Reactor



Serverless Computing

AWS Lambda

since 2014

Google Cloud Functions

since 2016

IBM Cloud Functions

since 2016

Azure Functions

since 2016