<http://rxmarbles.com/>

Operators

* map
* mapTo
* filter
* take
* first
* takeUntil

### catch / retry / retryWhen / repeat

* catch
* retry 发生错误后重试
* retryWhen
* repeat 不发生错误重试

### distinct / distinctUntilChanged

* distinct
* distinctUntilChanged 只和最后一个值比较

### debounce / throttle

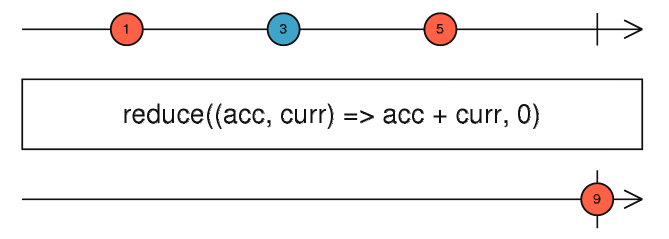
* debounce
* throttle
* forkJoin
* skip
* takeLast
* last
* concat
* stratWith
* merge

### delay

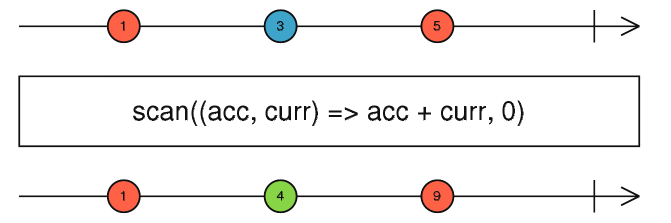
* delay
* delayWhen

### reduce / scan

* reduce reduce 只会发出一个值， 并且是当源 Observable 完成时才发出。它等价于使用 [scan](https://cn.rx.js.org/class/es6/Observable.js~Observable.html#instance-method-scan) 操作符后面再跟 [last](https://cn.rx.js.org/class/es6/Observable.js~Observable.html#instance-method-last) 操作符

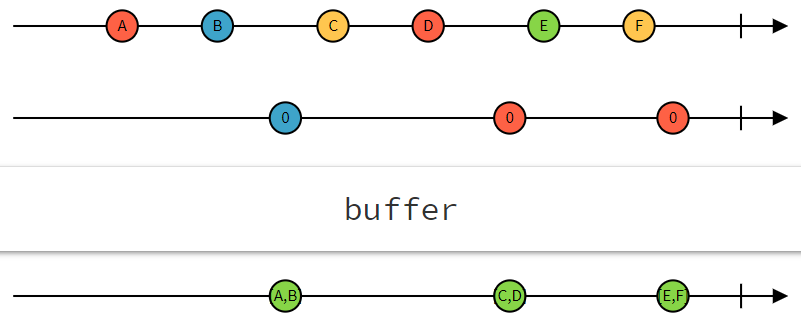


* scan 这就像是[reduce](https://cn.rx.js.org/class/es6/Observable.js~Observable.html#instance-method-reduce)， 但是会发出中间的累加值



### buffer

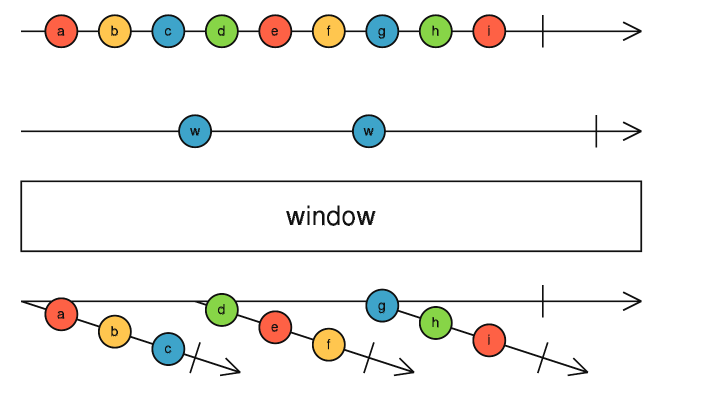
* buffer 把 source 存起来，等到newest 再发射



* bufferCount
* bufferTime
* bufferToggle
* bufferWhen

### window / windowToggle / groupBy

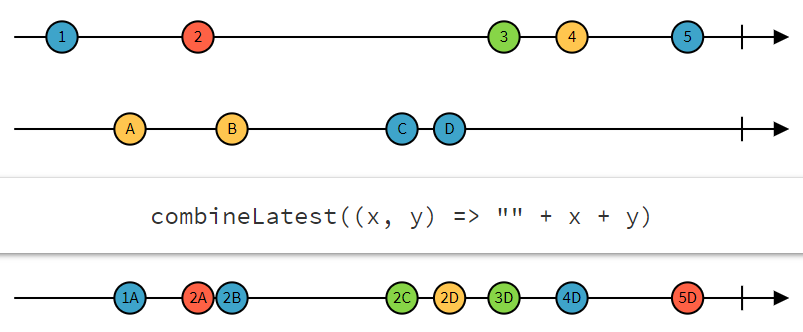
* window 就像是 [buffer](https://cn.rx.js.org/class/es6/Observable.js~Observable.html#instance-method-buffer), 但发出的是嵌套的 Observable ，而不是数组。



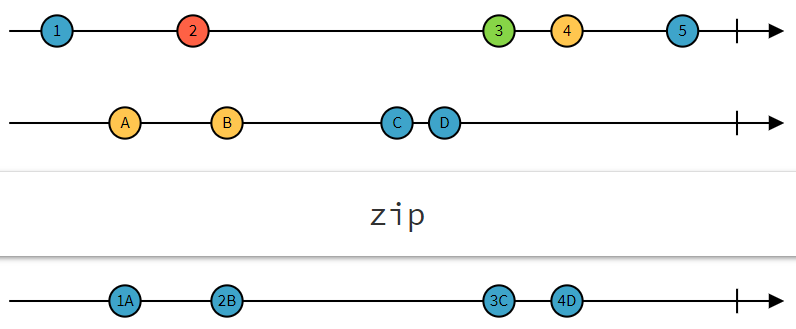
* windowToggle
* groupBy

### combineLatest / zip / withLatestFrom

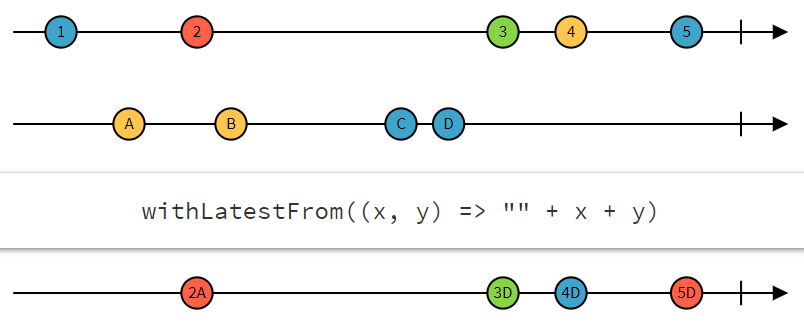
* combineLatest 不管是source 还是newest，只要任意一个发送新值，就会callback



* zip位置一一对应



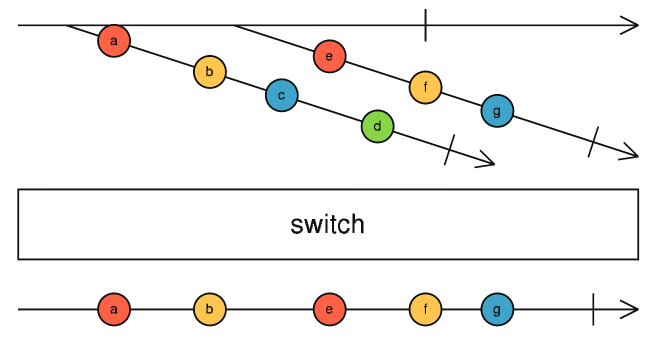
* withLatestFrom source 发送新值就callback



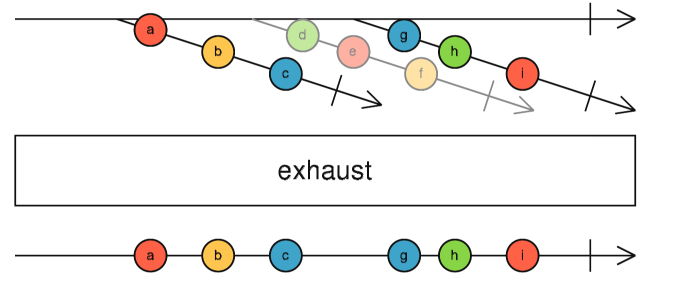
Higher Order Observable

### switch / concatAll / mergeAll / exhaust

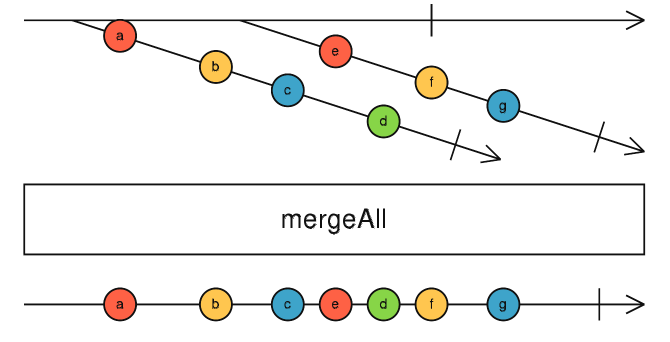
* switch 一旦有新的内部 Observable 出现，通过丢弃前一个，将 高级 Observable 打平



* concatAll
* exhaust 在当前内部 Observable 仍在执行的情况下，通过丢弃 接下来的内部 Observable 将高阶 Observable 打平



* mergeAll 将高阶 Observable 转换成一阶 Observable ，一阶 Observable 会同时发出在内部 Observables 上发出的所有值。



### switchMap / concatMap / mergeMap / exhaustMap

* switchMap map + switch
* concatMap map + concatAll

将每个值映射为 Observable, 然后使用[concatAll](https://cn.rx.js.org/class/es6/Observable.js~Observable.html#instance-method-concatAll)将所有的 内部 Observables 打平。

* mergeMap map + mergeAll
* exhaustMap map + exhaust

## 漸進式取值

## Subject, BehaviorSubject, ReplaySubject, AsyncSubject

## BehaviorSubject

* 创建的时候就要赋值
* 订阅者会立刻收到最新的值

## ReplaySubject

* 无论何时订阅，都会将所有历史订阅内容全部发出
* 添加参数指派会放最后几个元素

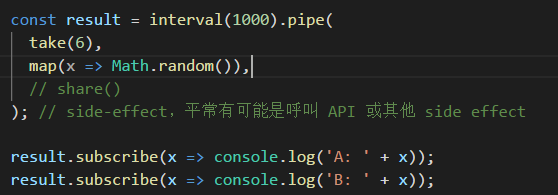
## AsyncSubject

* 只在complete的时候发送最后的一个值

# multicast, refCount, publish, share

Subject 与 Observable的差异

* Subject有一份 observer  清单
* side-effect 一定要用subject



* 訂閱就是執行一個 funciton
* 訂閱接收的物件具備 next, error, complete 三個方法
* 訂閱會返回一個可退訂(unsubscribe)的物件