RxJS

Реактивное программирование

Реактивное программирование — парадигма программирования, ориентированная на потоки данных и распространение изменений

Observer pattern

Iterator pattern

Functional programming

RxJS

Rx - Reactive Extension

.NET

Java Script

Java

https://github.com/Reactive-Extensions/RxJS/tree/master/dist

Arrays

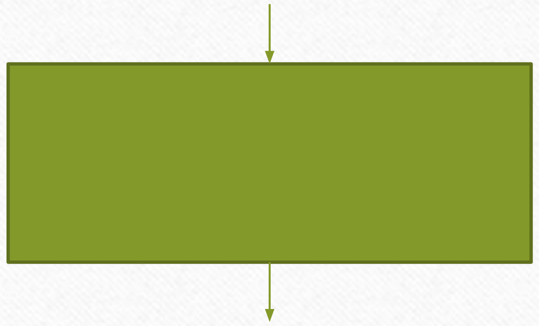
```
let array = [1, 2, 3]
```

```
for (let i = 0; i < array.length; i++) {
   console.log(array[i]);
}
```

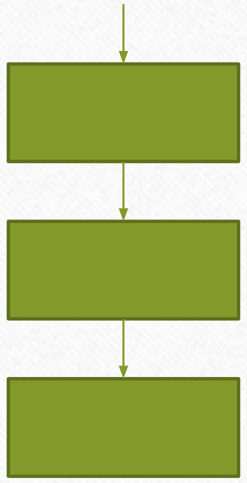
```
array.forEach ( x => console.log(x));
```

JS FLOW

```
for (let i = 0; i < array.length; i++) {
   console.log(array[i]);
}
```

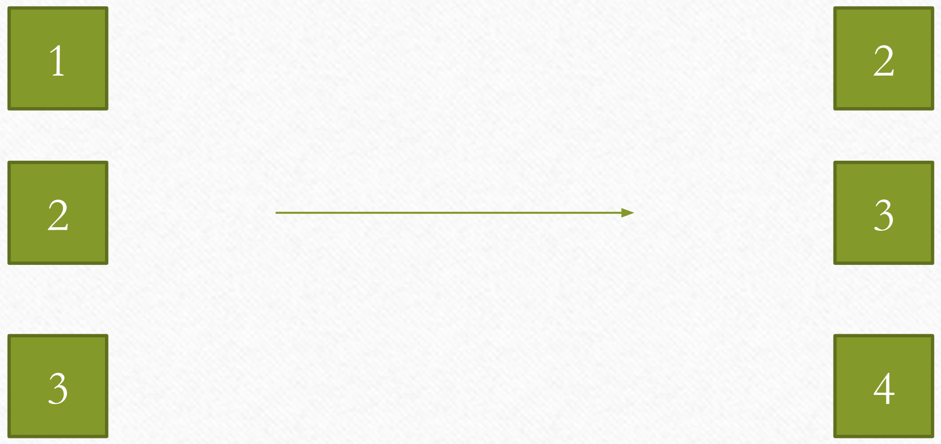


```
array.forEach ( x => console.log(x));
```



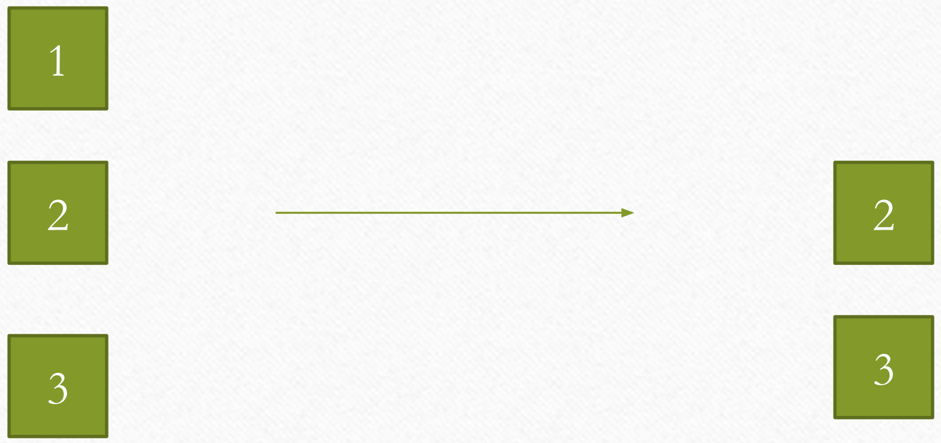
ARRAY METHODS: MAP

```
let newArr = arr.map( x => x + 1)
```



ARRAY METHODS: FILTER

```
let newArr = arr.filter( x => x > 1)
```



АСИНХРОННОЕ ПРОГРАММИРОВАНИЕ

```
function func2(callback) {
  callback('Done!');
}
```

```
function func1(message) {
  console.log(message);
}
```

```
func2 (func1);
```

АСИНХРОННОЕ ПРОГРАММИРОВАНИЕ

Callback hell

Big Ball of Mud

Callbacks can run more than once

Callbacks change error semantics.

Concurrency gets increasingly complicated.

promises

Улучшение над callback

Вырабатывают только одно значение

↓

Необходимо создавать отдельный промис на каждый запрос

Event emitter

Events force side effects.

Events are not first-class values.

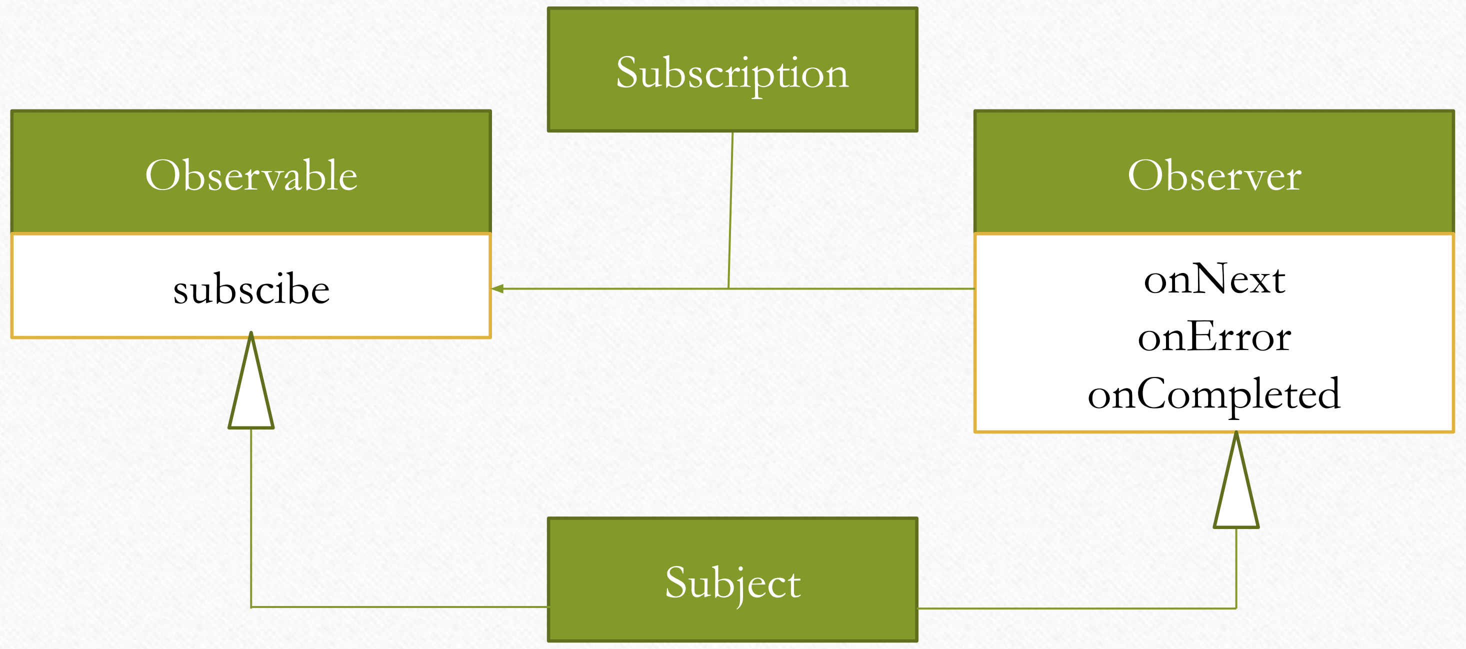
It is easy to miss events if we start listening too late.

PUSH vs PULL

	PUSH	PULL	IoC
one value	functions	promises	
values	generators, iterators	observables	

RX PATTERN

RX PATTERN = Observer Pattern + Iterator Pattern



Observable

```
Rx.Observable
      .from([‘Ivan’, ‘Petr’, ‘Sergey’])
      .subscribe(
           x =>{ console.log('Next: ' + x); },
          err => { console.log('Error:', err); }
          ( ) => { console.log('Completed'); }
      );
```

Observer

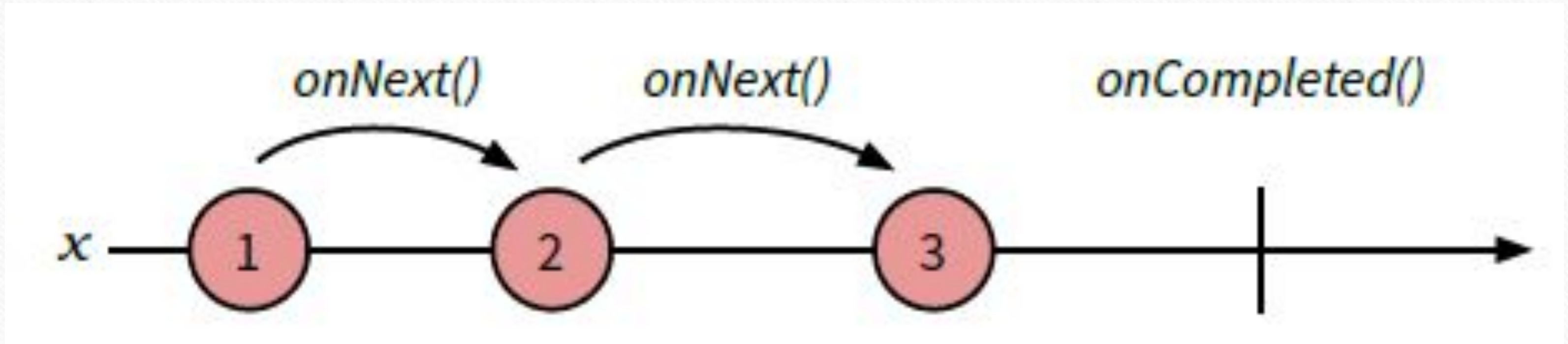
```
var observer = Rx.Observer.create(
        x =>{ console.log('Next: ' + x); },
        err => { console.log('Error: ' + err); },
        ( ) { console.log('Completed'); }
);
```

From Event

```
var allMoves = Rx.Observable.fromEvent(document, 'mousemove');

allMoves.subscribe(e => { console.log(e.clientX, e.clientY); });

allMoves
  .map(e => e.clientX)
  .filter(x => x  < window.innerWidth / 2 )
  .subscribe(e => console.log(‘mouse on the left’);
```

Marbel diagrams

Rx.Observable.range(1, 3);



interval

```
var a = Rx.Observable.interval(200).map(function(i) {
  return 'A' + i;
});

var b = Rx.Observable.interval(100).map(function(i) {
  return 'B' + i;
});
```

subscription

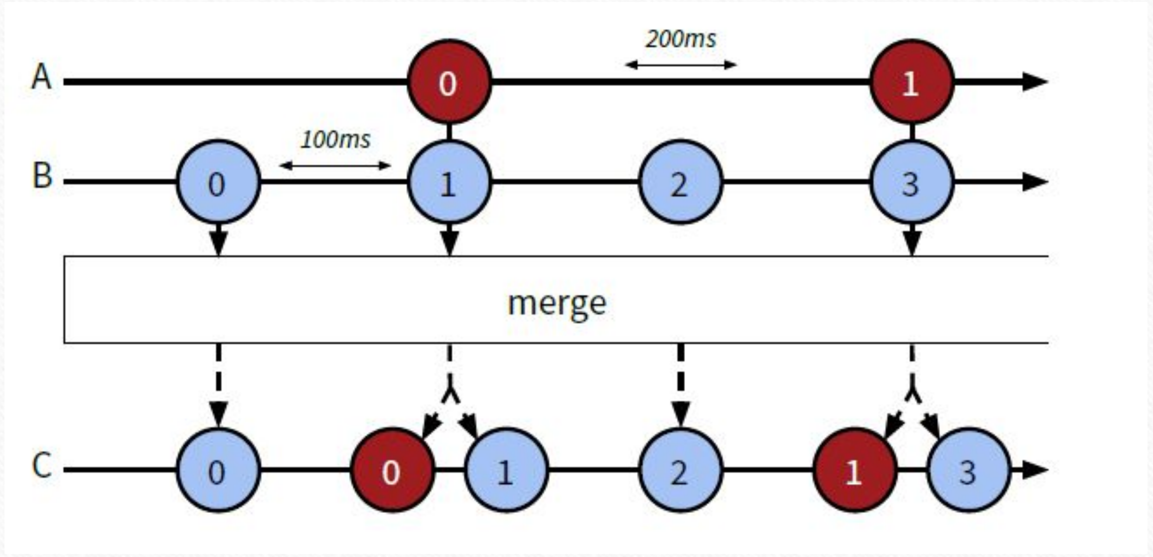
```
let observable = Rx.Observable.interval(1000);
let subscription = observable.subscribe(x => console.log(x));

subscription.dispose( );
```

take

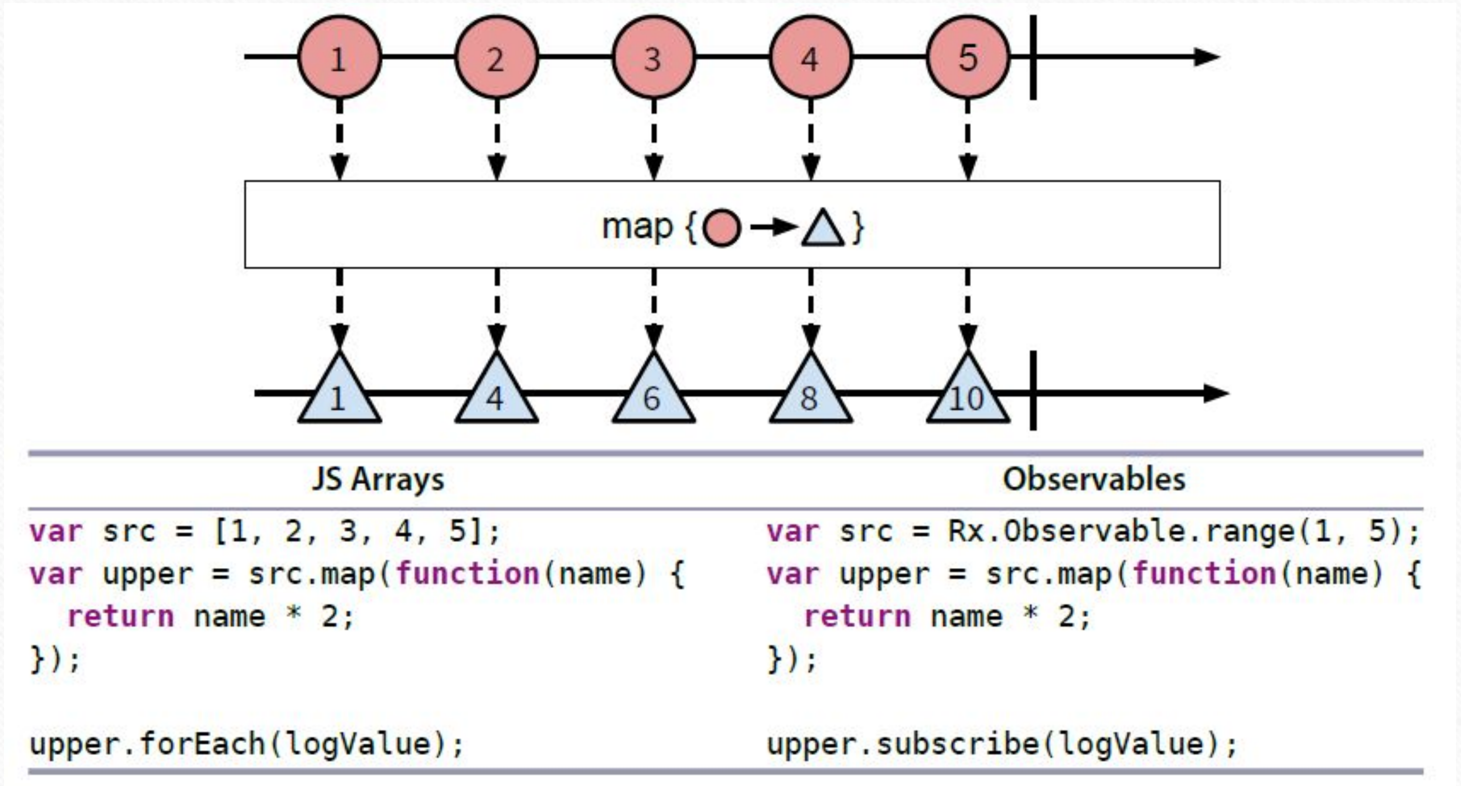
```
let observable = Rx.Observable.interval(1000)
        .take(5);

observable.subscribe(x => console.log(x));
```

merge

```
var a = Rx.Observable.interval(200).map(i => 'A' + i);
var b = Rx.Observable.interval(100).map(i => ‘B' + i);
Rx.Observable.merge(a, b).subscribe(x => console.log(x));
```



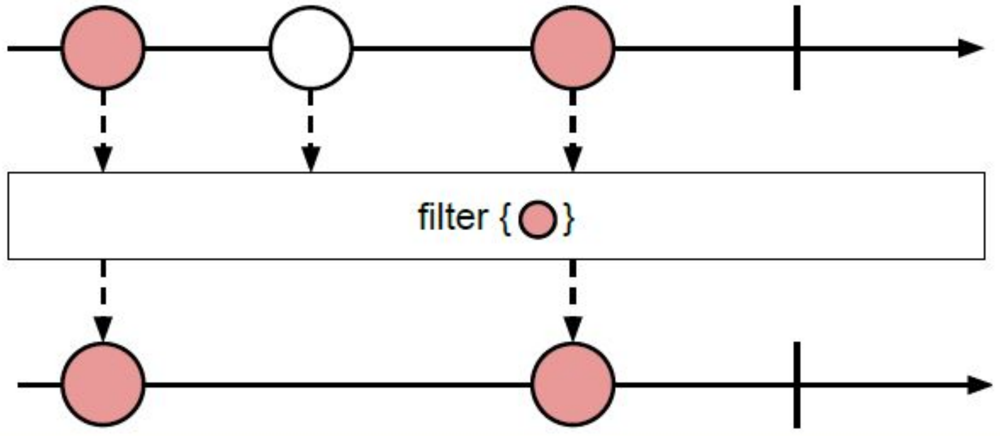
map



JS Arrays	Observables

`var src = [1, 2, 3, 4, 5];`
`var upper = src.map(function(name) {`
`  return name * 2;`
`});`
 	
`upper.forEach(logValue);`
	`var src = Rx.Observable.range(1, 5);`
	`var upper = src.map(function(name) {`
	`  return name * 2;`
	`});`
	
	`upper.subscribe(logValue);`

filter

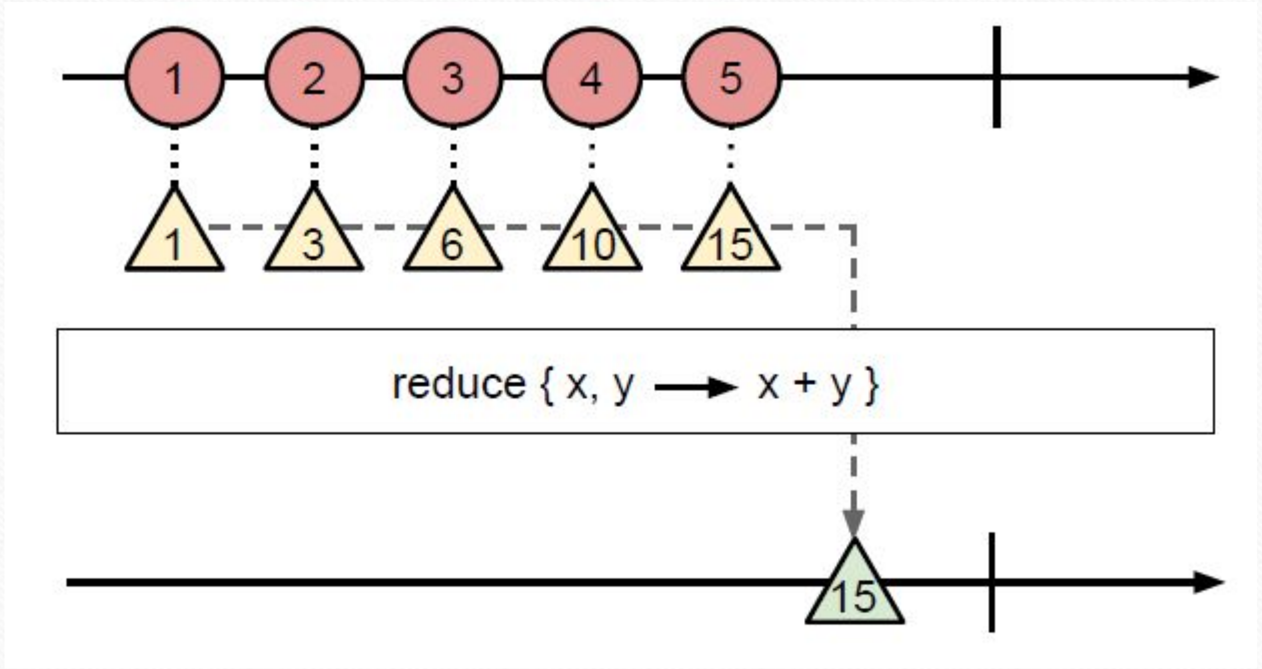


JS Arrays	Observables
`var isEven = (function(val) { return val % 2 !== 0; });`	

`var src = [1, 2, 3, 4, 5];`
`var even = src.filter(isEven);`
 	
`even.forEach(logValue);`
	`var src = Rx.Observable.range(1, 5);`
	`var even = src.filter(isEven);`
	
	`even.subscribe(logValue);`

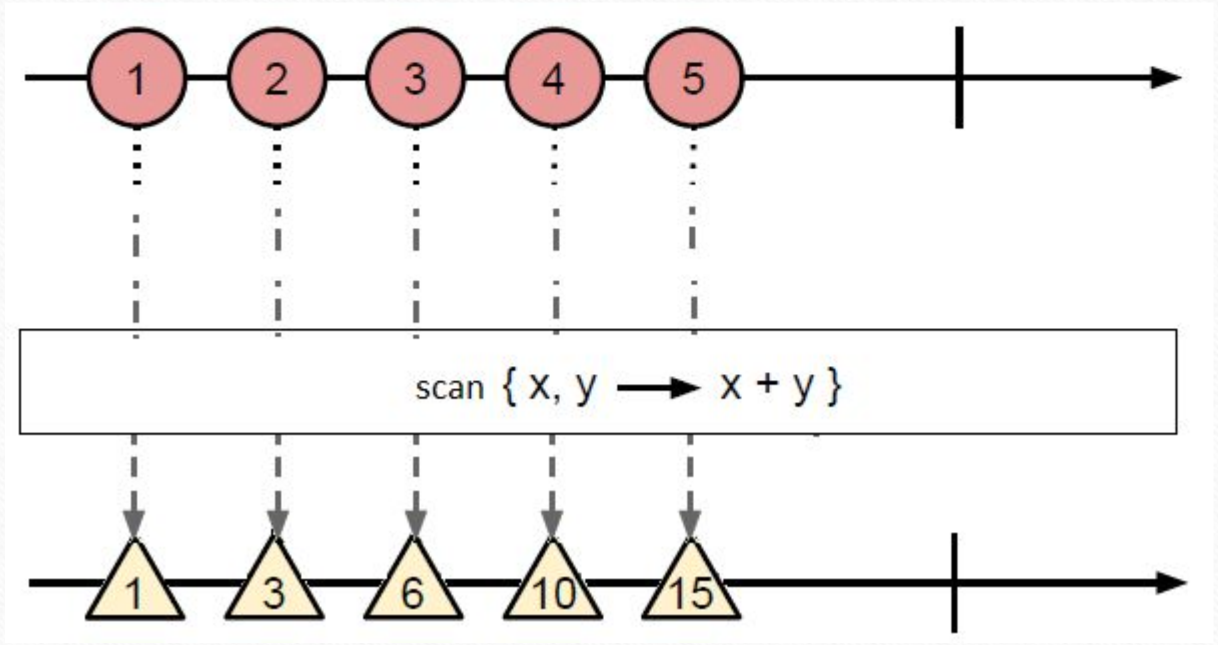
reduce

```
var src = Rx.Observable.range(1, 5);
var sum = src.reduce( (acc, x) => acc + x);
sum.subscribe(logValue);
```



scan

```
var src = Rx.Observable.range(1, 5);
var sum = src.scan( (acc, x) => acc + x);
sum.subscribe(logValue);
```

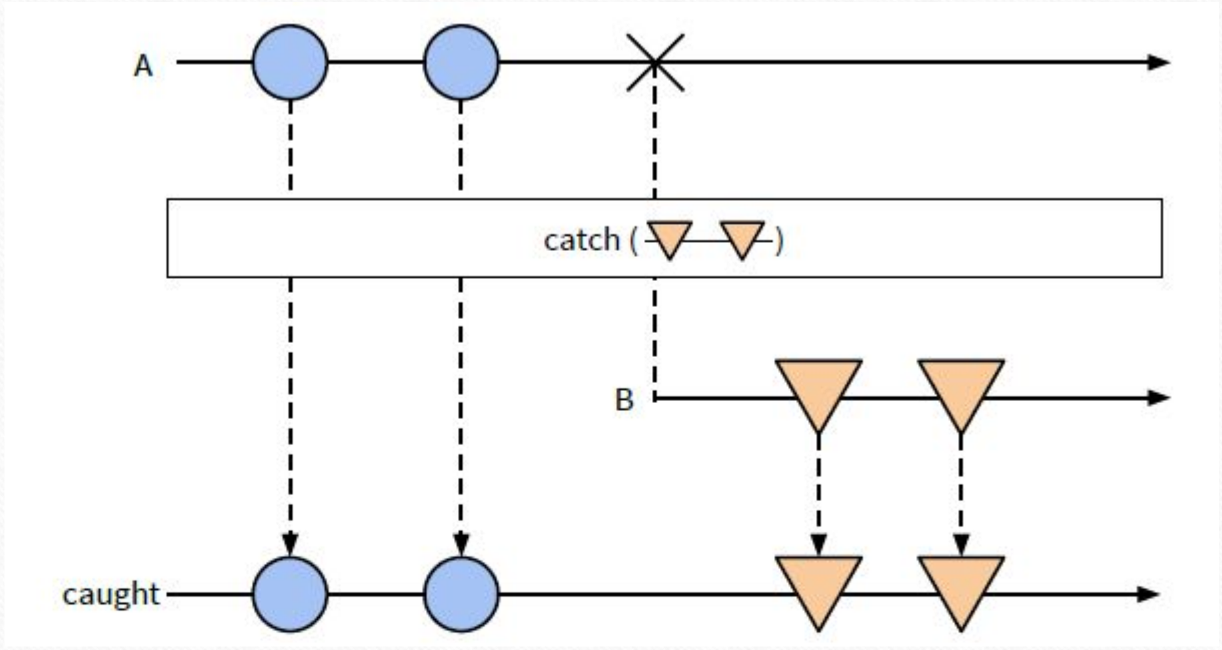


Custom Observable

```
Rx.Observable.create( (observer) => {
    observer.onNext (someValue);
    …
    observer.onError(new Error('some error'));
    …
     observer.onCompleted( );
});
```

Handling Error

```
let caught = observable.catch(
   Rx.Observable.return({
      error: “Some details”
    })
);
```



Handling Error

```
Rx.DOM
   .get('/products')
   .retry(5)
   .map(xhr => xhr.result)
   .subscribe(
         result => console.log(result),
         err => console.error('ERROR: ', err)
   );
```

Handling Error

```
var observable = Rx.Observable
   .fromEvent (button, ‘click’)
   .throttle(500)
   .flatMap( ( ) => Rx.DOM.get(‘products’))
   .retry(5)
   .map(xhr => xhr.result);

observable.subscribe(
          result => console.log(result),
          err => console.error('ERROR: ', err)
    );
```

Hot and Cold Observables

Работают сразу же

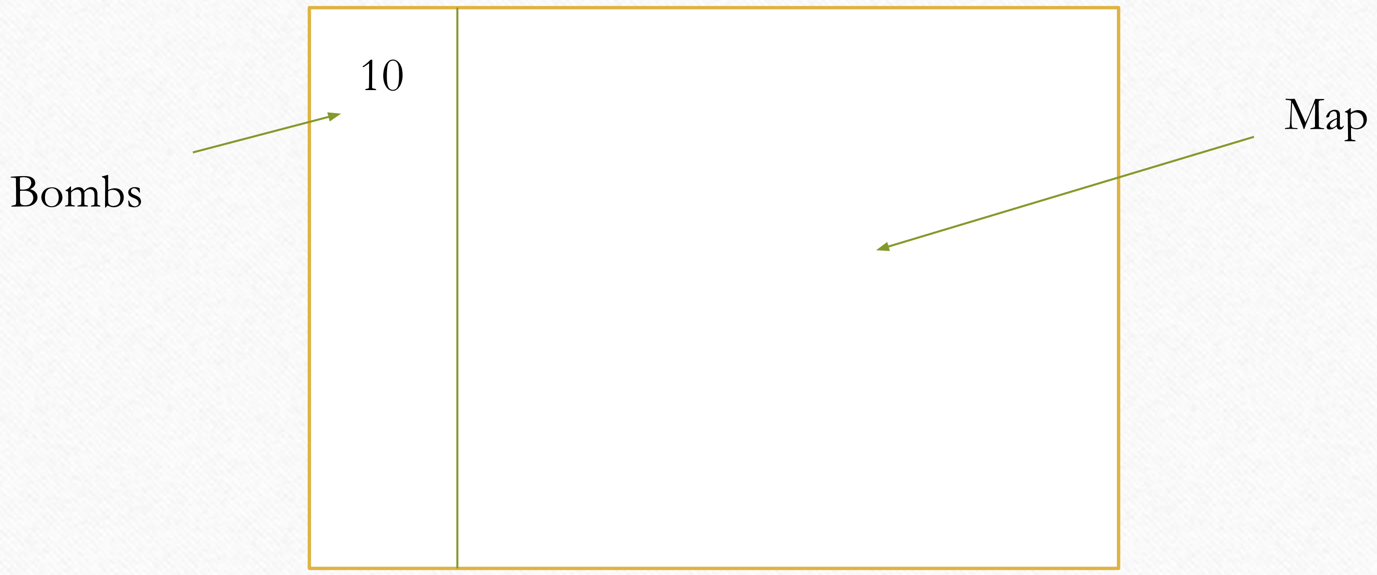
Нужен подписчик

Для каждого подписчика – своя последовательность

Cold -> Hot

```
var hotObservable = coldObservable.publish( );

// код

hotObservable.connect( );
```

The Game


10
Bombs
Map