ReactiveX

Thinking differently about events

Đinh Quang Trung
01 Mar 2016

Overview

1. What is ReactiveX (Rx)?
2. Idea of Rx, rethink about events
3. Get to know some basic Rx operators
4. How Rx solves popular async problems by examples

Discussion:

- When to use Rx? Where to use Rx?
- Rx on the “back”.
- How does it work with MVVM application?
- Testing in Rx?
- Reactive Programming?

ReactiveX

“An API for asynchronous programming with observable streams”

ReactiveX is now ported to:

Languages

- Java: RxJava
- JavaScript: RxJS
- C#: Rx.NET
- C#(Unity): UniRx
- Scala: RxScala
- Clojure: RxClojure
- C++: RxCpp
- Ruby: Rx.rb
- Python: RxPY
- Groovy: RxGroovy
- JRuby: RxJRuby
- Kotlin: RxKotlin
- Swift: RxSwift

ReactiveX for platforms and frameworks

- RxNetty
- RxAndroid
- RxCocoa

JavaScript 101

Function

```
function (x) {
   console.log(x);
}

function (x, y) {
   return x + y;
}
```

Function

```
function (x) { console.log(x) }

function (x, y) { return x + y }
```

Function

```
x => console.log(x)

(x, y)    =>    x + y
```

Function

```
x => console.log(x)

(x, y) => x + y
```

Quiz

```
var a = x => x;
var b = (y, z) =>
   y + z;

var c = a(1) + b(2, 3);

console.log(c)
```

Quiz

```
var a = x => x;
var b = (y, z) =>
   y + z;

var c = a(1) + b(2, 3);

console.log(c)
```

JavaScript 101

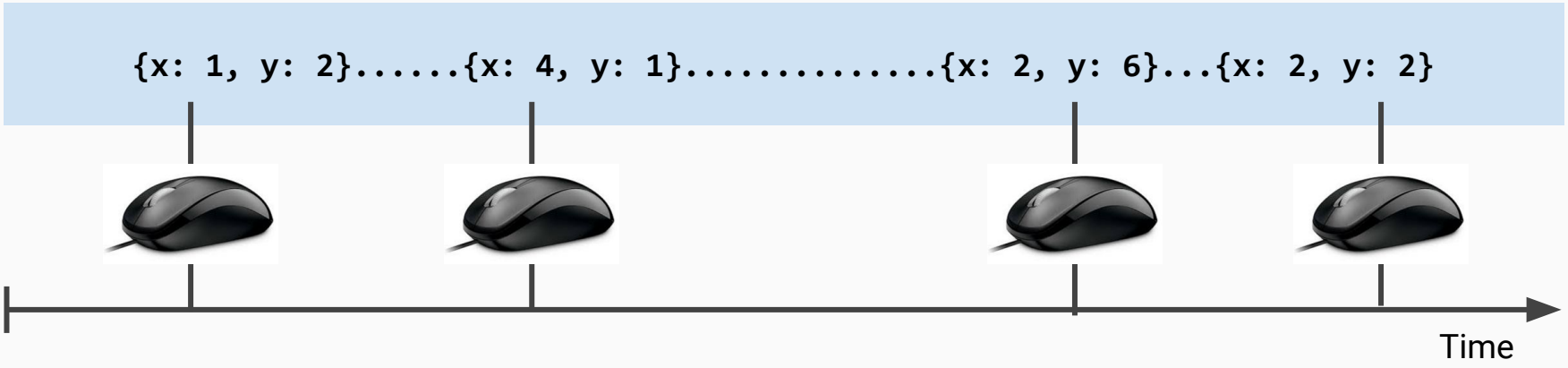
End.

Rethink about events

“What’s the difference between an Array...

```
[{x: 1, y: 2}, {x: 4, y: 1}, {x: 2, y: 6}, {x: 2, y: 2}]
```

...and an Event?”



This is an array

```
[1, 2, 3]
```

ForEach

```
> [1, 2, 3].forEach(x => console.log(x))
```

```
> 1
> 2
> 3
```

Map

```
> [1, 2, 3].map(x => x * 2)
```

```
> [2, 4, 6]
```

Filter

```
> [1, 2, 3].filter(x => x > 1)
```

```
> [2, 3]
```

ConcatAll

```
> [1, [2, 3], [], [4]].concatAll()
```

```
> [1, 2, 3, 4]
```

Map/Filter/ConcatAll

```
> [1, 2, 3].map(x => x * 2)
```

```
> [2, 4, 6]
```

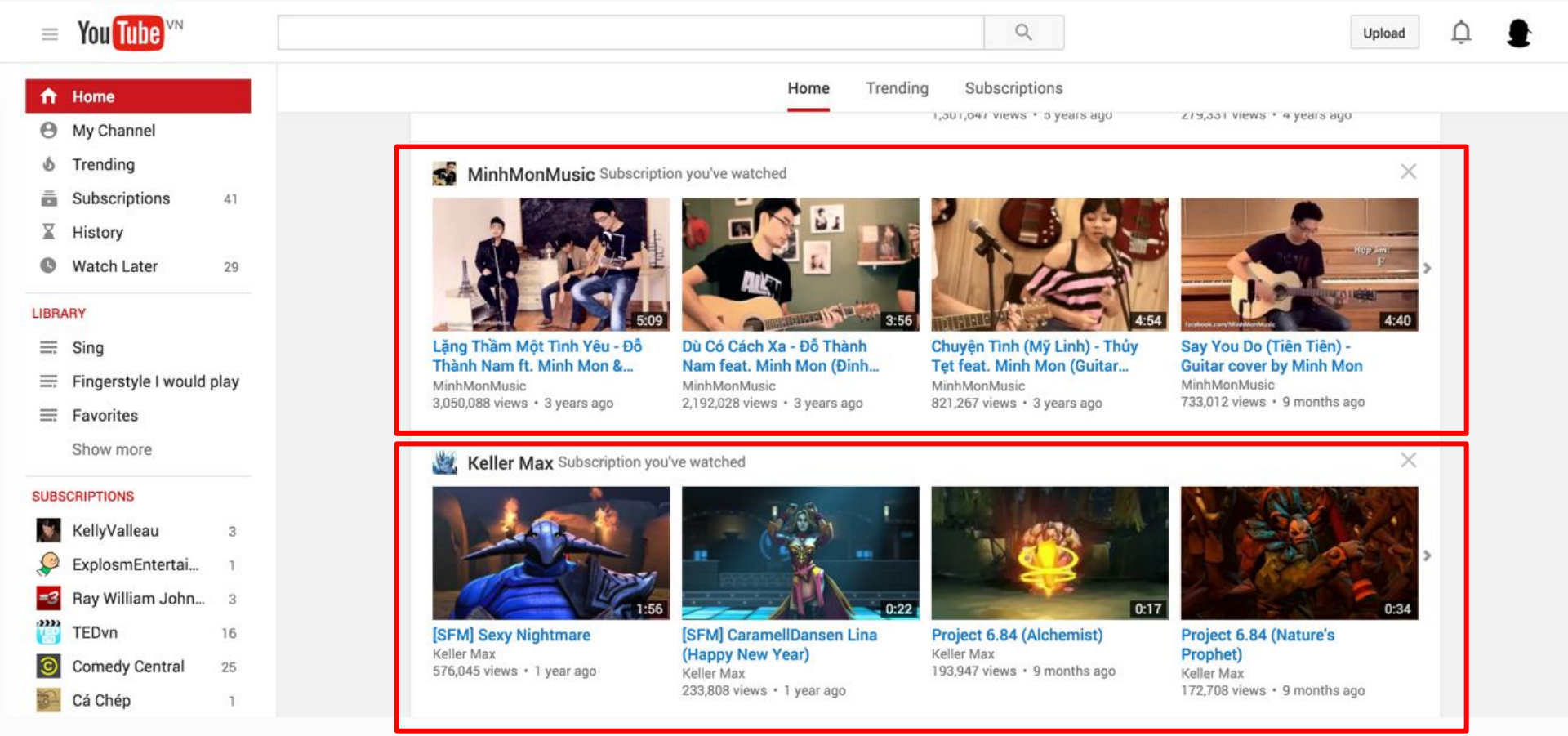
```
> [1, 2, 3].filter(x => x > 1)
```

```
> [2, 3]
```

```
> [1, [4, 9], [], [3]].concatAll()
```

```
> [1, 4, 9, 3]
```

Get user’s top rated videos



What we have

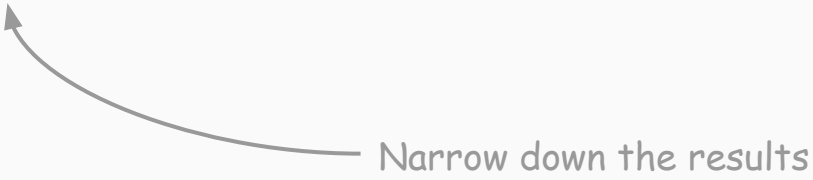

User
Playlist
Video
Rating: 1
Video
Rating: 2
Video
Rating: 1
Video
Rating: 2
Video
Rating: 1
Video
Rating: 7
Video
Rating: 7
Video
Rating: 3
Video
Rating: 8
Video
Rating: 6

What we have

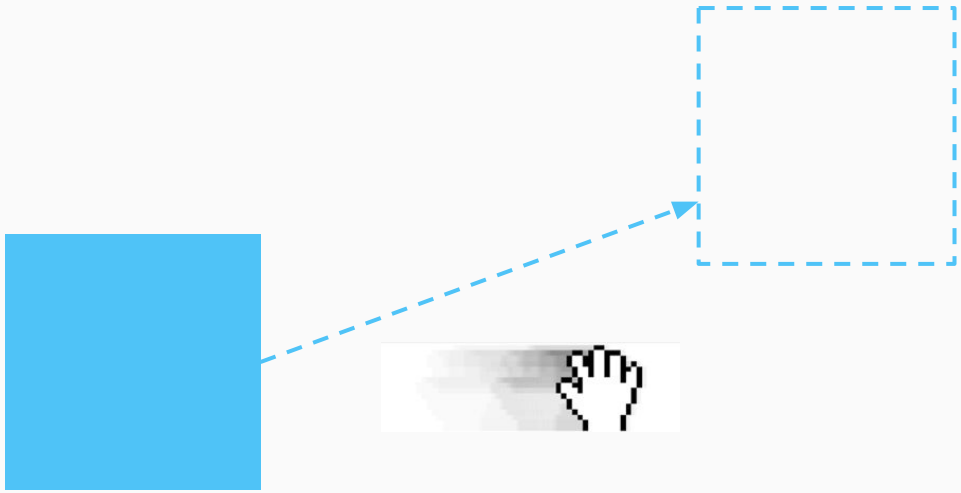
What we want

Get user’s top rated videos

```
var getTopRatedVideos = user =>
    user.playlists.map(playlist =>
        playlist.videos.filter(video => video.rating > 5)
    ).concatAll()

getTopRatedVideos(user)
    .forEach(video => console.log(video))
```



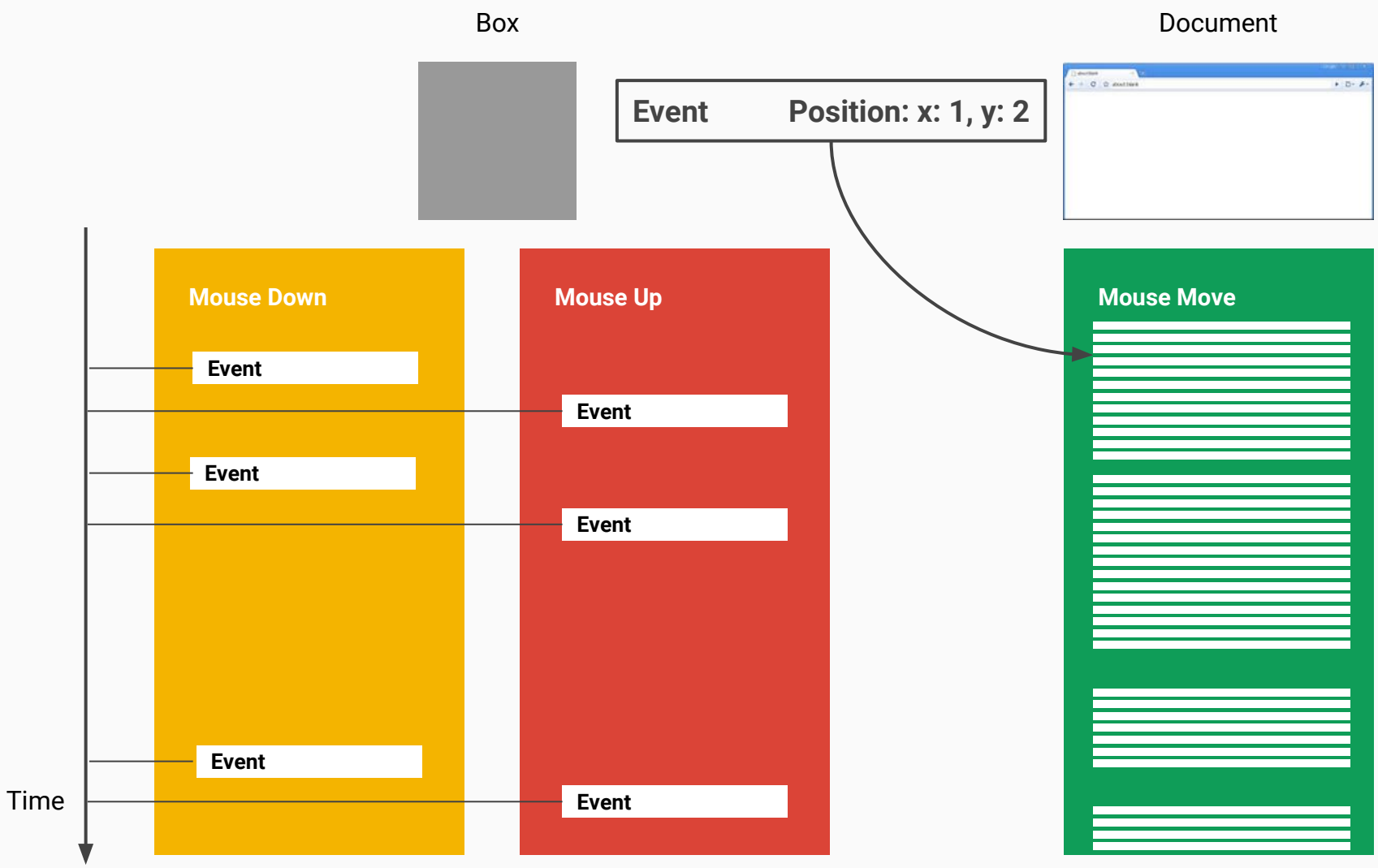
Handle drag-drop event



Explain the `mousedown`, `mouseup` and `mousemove` event:

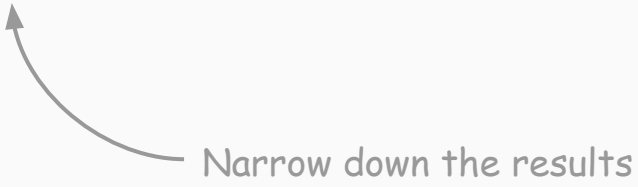
https://jsfiddle.net/trungdq88/p953tx0n/

Events as collections

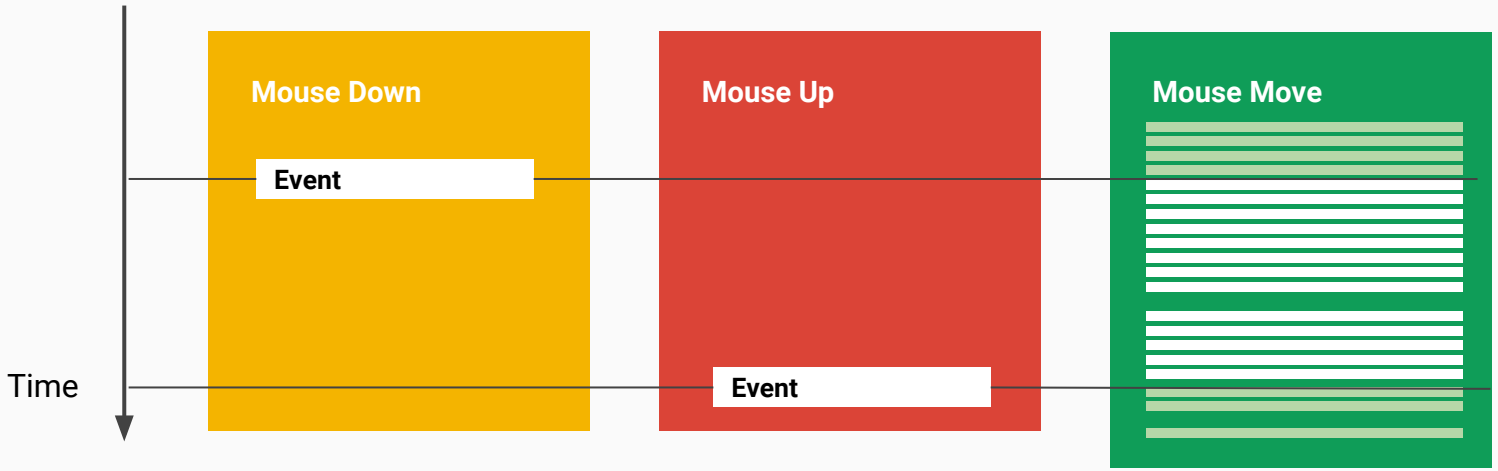


Handle drag-drop event

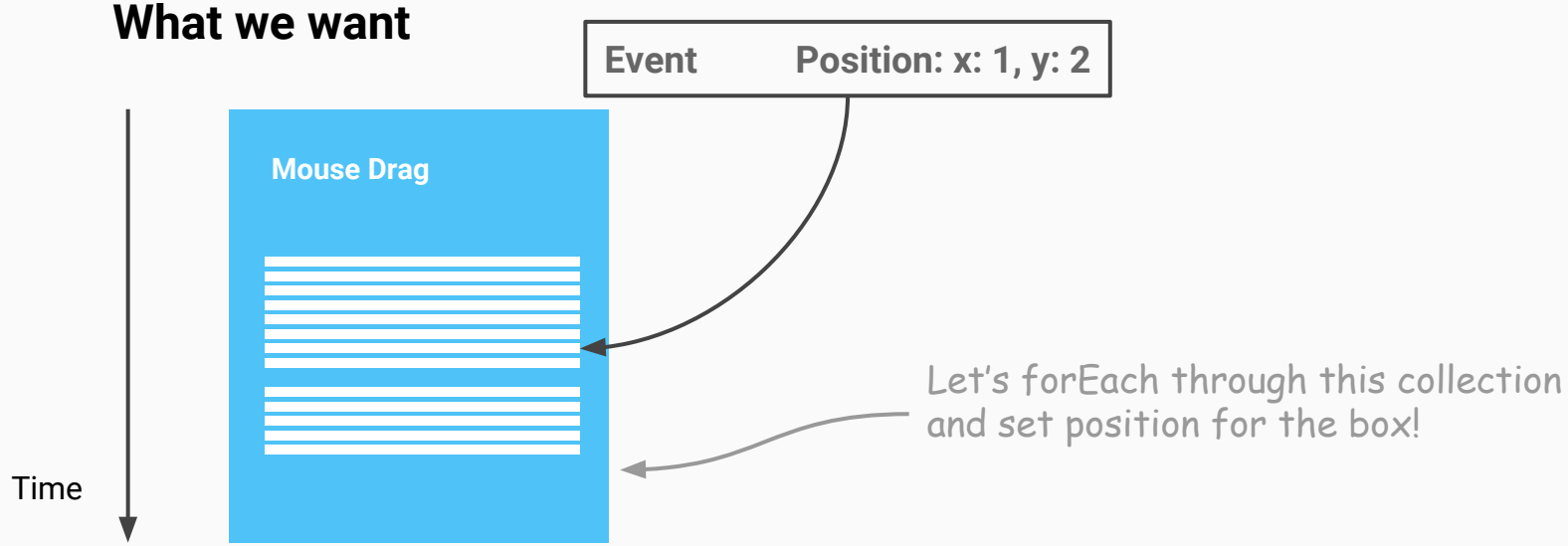
```
var getBoxDrags = box =>
    box.mouseDowns.map(mouseDown =>
        document.mouseMoves. filter takeUntil(document.mouseUps)
    ).concatAll()

getBoxDrags(box)
    .forEach(pos => box.position = pos)
```



What we have



What we want



Observable

by ReactiveX

Observable can handle...

- Events
- Data requests
- Animations

Events to Observables

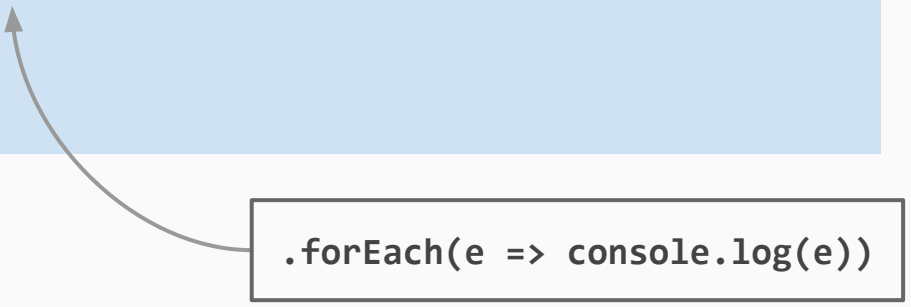
```
var mouseClicks = Observable.fromEvent(button, ‘click’);
```

Event handling the **old** way

```
// Create handler
var handler = e => console.log(e);

// Subscribe
document.addEventListener('click', handler);

// Ubsubscribe
document.removeEventListener('click', handler);
```

Event handling the **new** way

```
// Create handler
var handler = e => console.log(e);

// Create observable
var mouseClicks = Observable.fromEvent(button, ‘click’);

// Subscribe
var subscription = mouseClicks.forEach(handler);

// Ubsubscribe
subscription.dispose();
```



Event handling the **new** way, more complex

```
// Subscribe
var subscription = mouseClicks.forEach(
     // Next event data
     event => console.log(event),
     // Error
     error => console.log(error),
     // Completed
     () => console.log('Done')
);

// Ubsubscribe
subscription.dispose();
```

optional

Some observables lives and dies in peace.

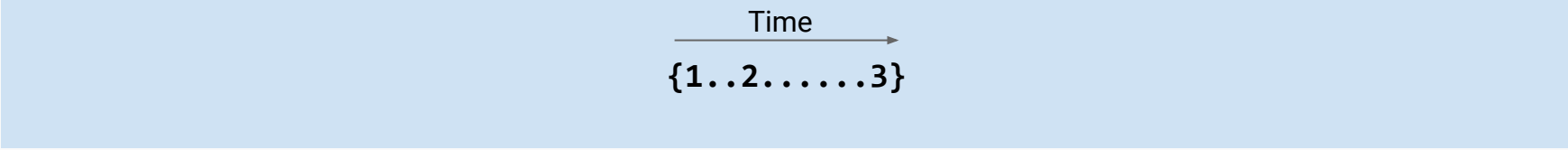
Some observables lives and dies in with exceptions.

Some observables never dies.

examples?

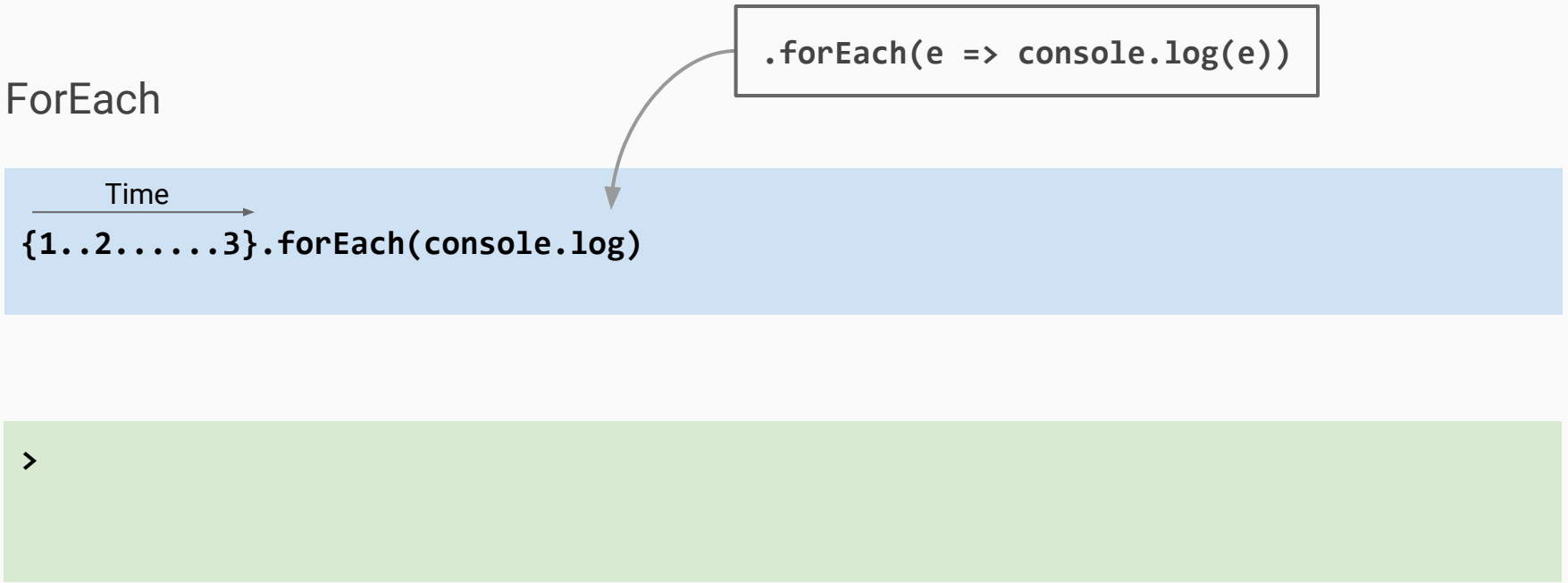
Get familiar with observables

This is an observable



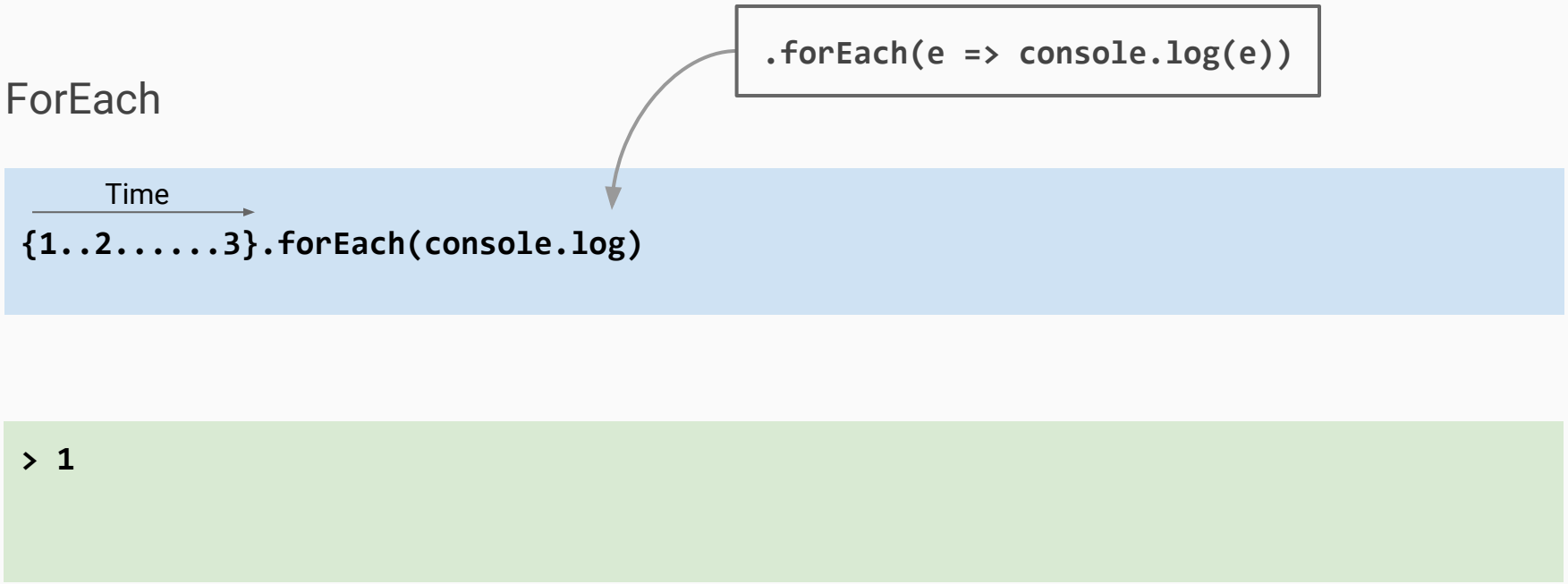
Get familiar with observables

ForEach



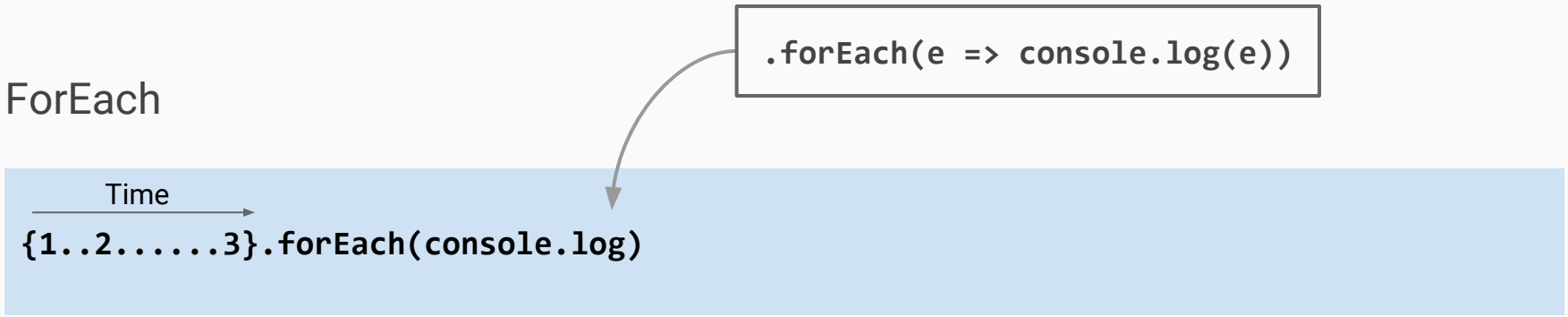
Get familiar with observables

ForEach



Get familiar with observables

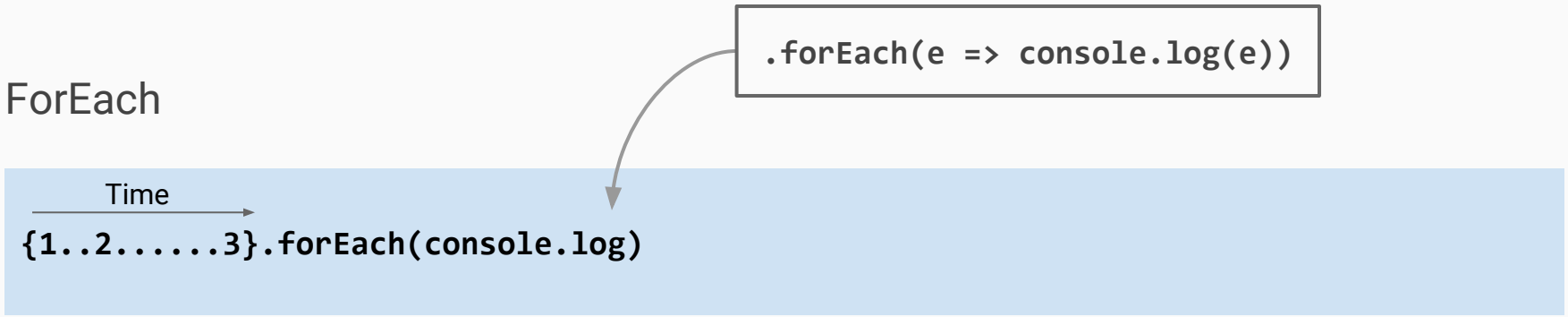
ForEach




```
> 1
> 2
```

Get familiar with observables

ForEach




```
> 1
> 2
> 3
```

Get familiar with observables

Map


```
{1..2.....3}.map(x => x * 2)
```

```
>
```

Get familiar with observables

Map


```
{1..2.....3}.map(x => x * 2)
```

```
> 2
```

Get familiar with observables

Map


```
{1..2.....3}.map(x => x * 2)
```

```
> 2
> 4
```

Get familiar with observables

Map


```
{1..2.....3}.map(x => x * 2)
```

```
> 2
> 4
> 6
```

Get familiar with observables

Filter


```
{1..2.....3}.filter(x => x > 1)
```

```
>
```

Get familiar with observables

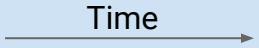
Filter


```
{1..2.....3}.filter(x => x > 1)
```

```
> 2
```

Get familiar with observables

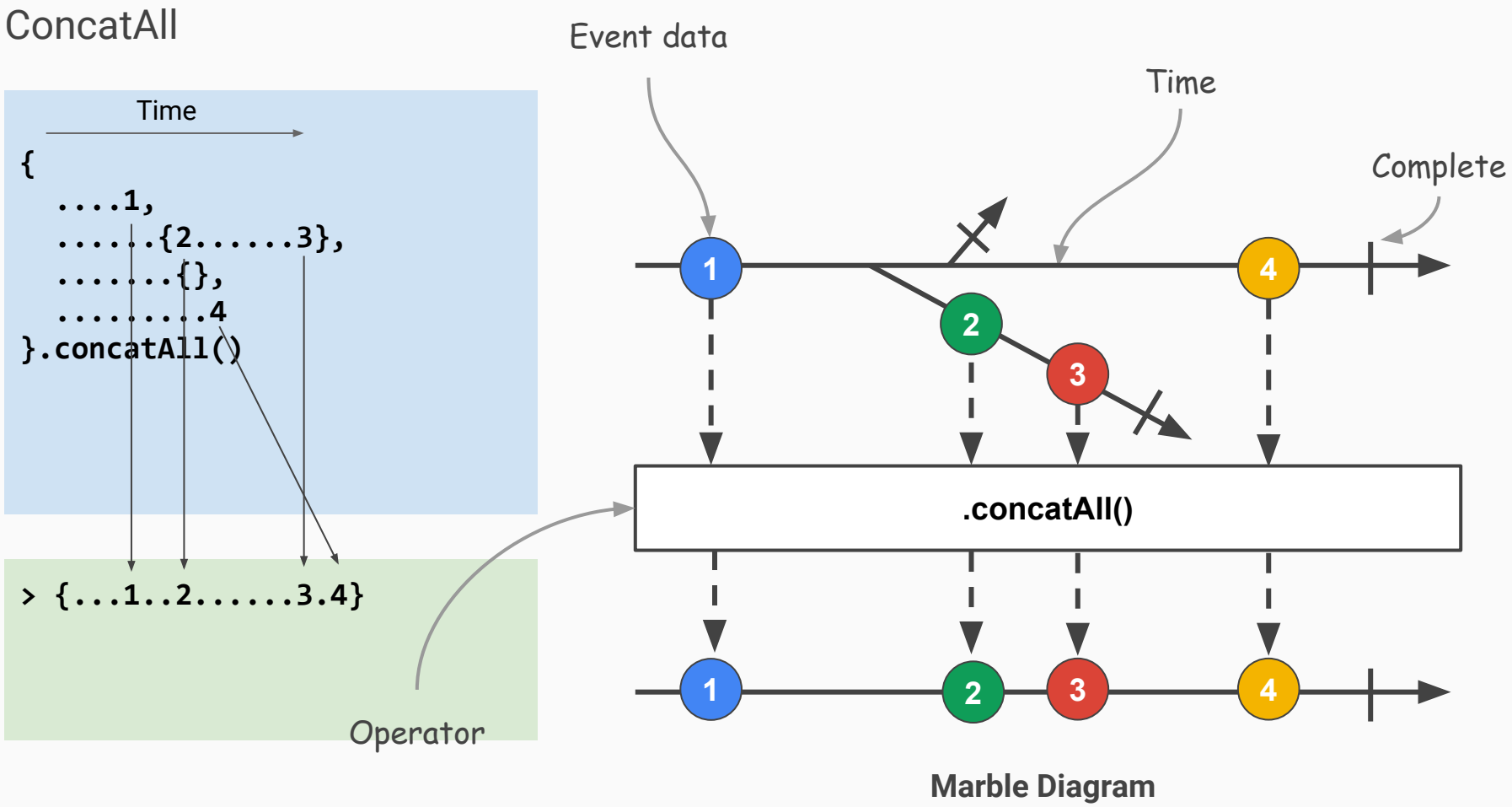
Filter




```
{1..2.....3}.filter(x => x > 1)
```

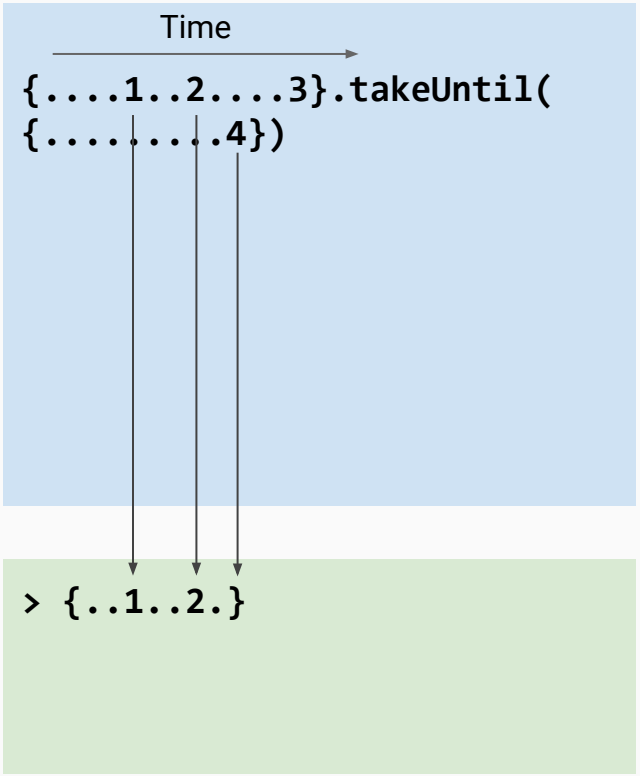
```
> 2
> 3
```

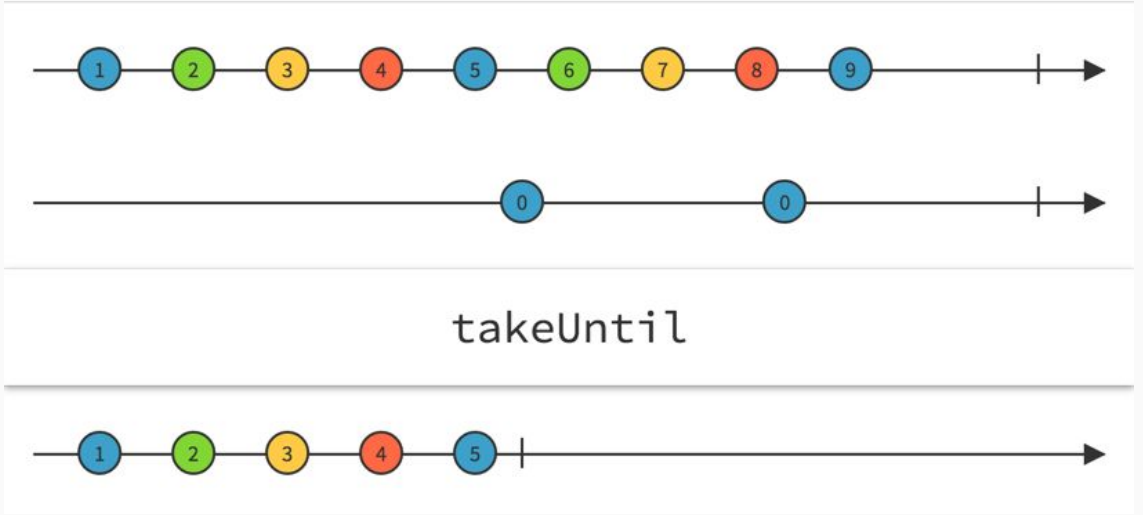
Get familiar with observables



Get familiar with observables

TakeUntil





Marble Diagram

Let’s see the drag-drop example again

```
var getBoxDrags = box =>
    box.mouseDowns.map(mouseDownEvent =>
        document.mouseMoves. filter takeUntil(document.mouseUps)
    ).concatAll()

getBoxDrags(box)
    .forEach(pos => box.position = pos)
```

ReactiveX operators

`takeUntil`

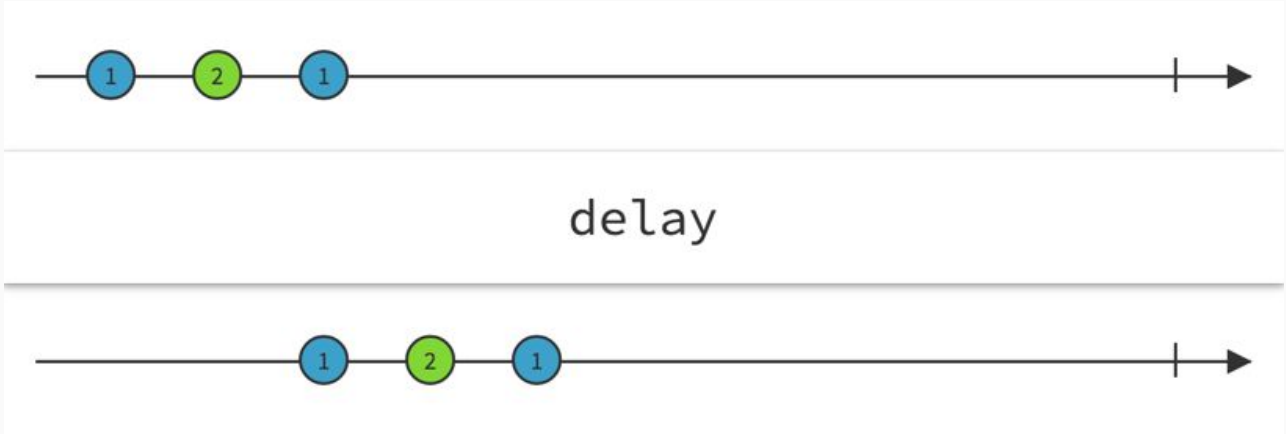
`concatAll`

ReactiveX operators

`takeUntil`

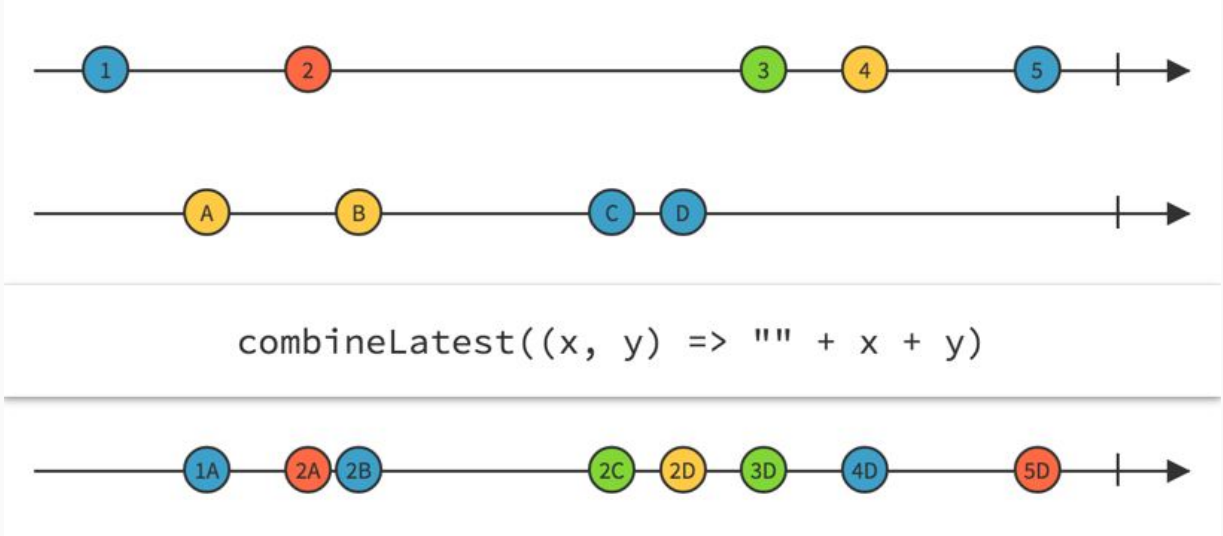
`concatAll`

`delay`



ReactiveX operators

- `takeUntil`
- `concatAll`
- `delay`
- `combineLatest`



ReactiveX operators

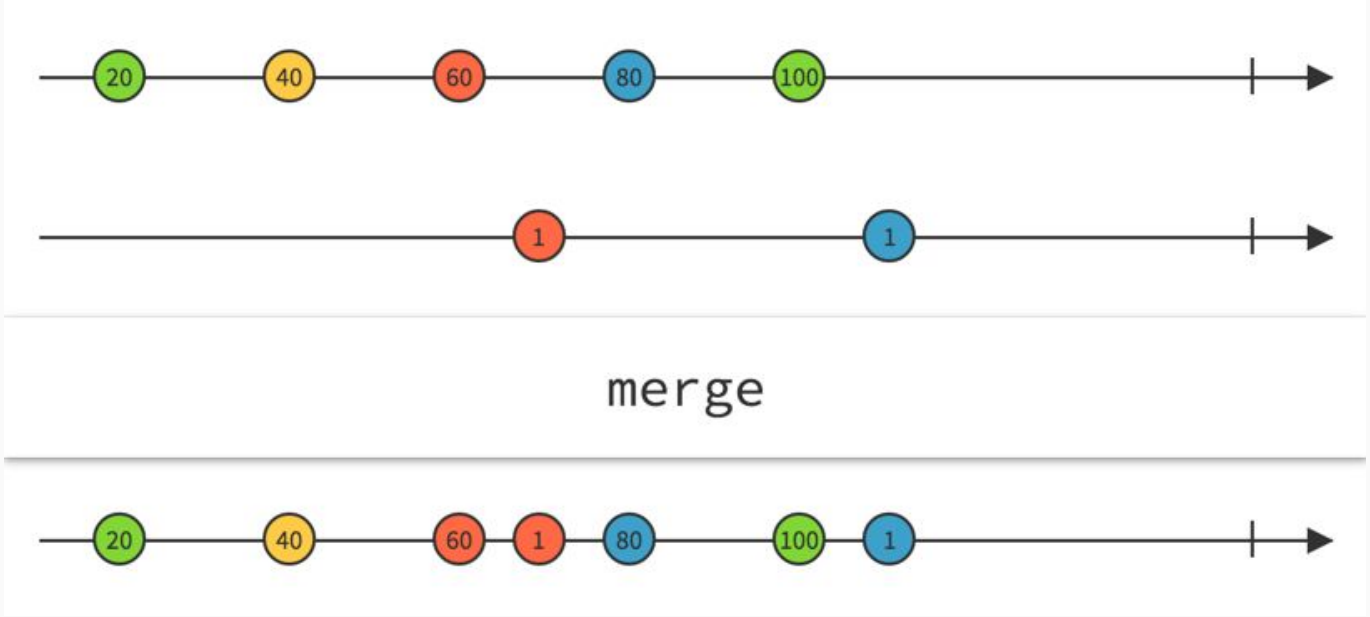
`takeUntil`

`concatAll`

`delay`

`combineLatest`

`merge`



ReactiveX operators

takeUntil
concatAll
delay
combineLatest
merge
sample
startWith
zip
take
takeUntil
while

find
sum
count
repeat
first
last
skip
skipLast
skipUntil
reduce
every

some
include
reduce
min
max
debounce
buffer
bufferWithTime
bufferWithCount
map
scan

Aggregate All Amb and_ And Any apply as_blocking AsObservable AssertEqual asyncAction asyncFunc Average averageDouble averageFloat averageInteger averageLong blocking Buffer bufferWithCount bufferWithTime bufferWithTimeOrCount byLine cache case Cast Catch catchException collect collect CombineLatest combineLatestWith Concat concat_all concatMap concatMapObserver concatAll concatWith Connect connect_forever cons Contains controlled Count countLong Create cycle Debounce decode DefaultIfEmpty Defer deferFuture Delay delaySubscription delayWithSelector Dematerialize Distinct DistinctUntilChanged Do doAction doOnCompleted doOnEach

MapCat mapCat map-indexed map_with_index Materialize Max MaxBy Merge mergeAll merge_concurrent mergeDelayError mergeObservable mergeWith Min MinBy MostRecent Multicast nest Never Next none nonEmpty nth ObserveOn ObserveOnDispatcher observeSingleOn of of_array ofArrayChanges of_enumerable of_enumerator ofObjectChanges OfType ofWithScheduler onBackpressureBlock onBackpressureBuffer onBackpressureDrop OnErrorResumeNext onErrorReturn onExceptionResumeNext orElse pairs pairwise partition partition-all pausable pausableBuffered pluck product Publish PublishLast publish_synchronized publishValue raise_error Range Reduce reductions

sumLong Switch switchCase switchIfEmpty switchLatest switchMap switchOnNext Synchronize Take take_with_time takeFirst TakeLast takeLastBuffer takeLastBufferWithTime takeLastWithTime takeRight TakeUntil takeUntilWithTime TakeWhile take_while_with_index tail tap tapOnCompleted tapOnError tapOnNext Then thenDo Throttle throttleFirst throttleLast throttleWithSelector throttleWithTimeout Throw throwError throwException TimeInterval Timeout timeoutWithSelector Timer Timestamp To to_a ToArray ToAsync toBlocking toBuffer to_dict

More ReactiveX operators:

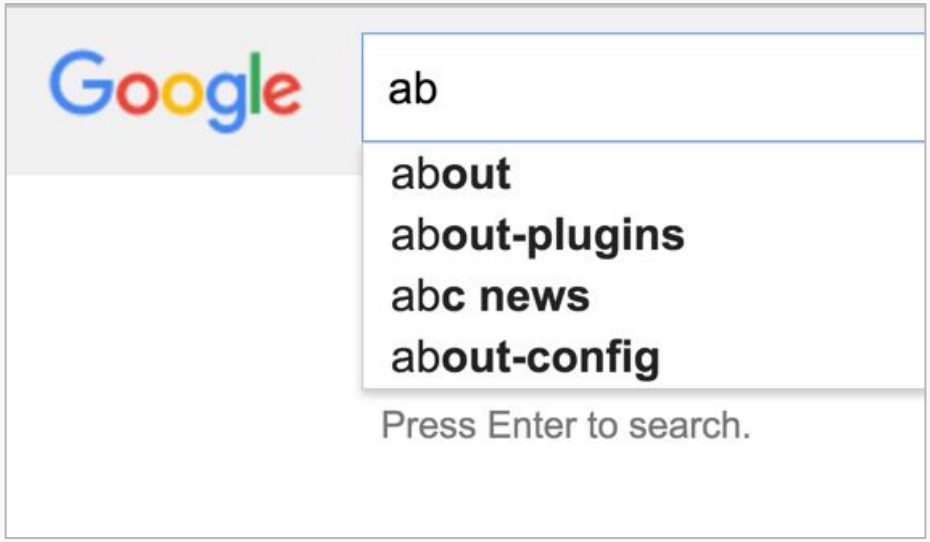
http://reactivex.io/documentation/operators.html

...with marble diagrams:

http://rxmarbles.com/

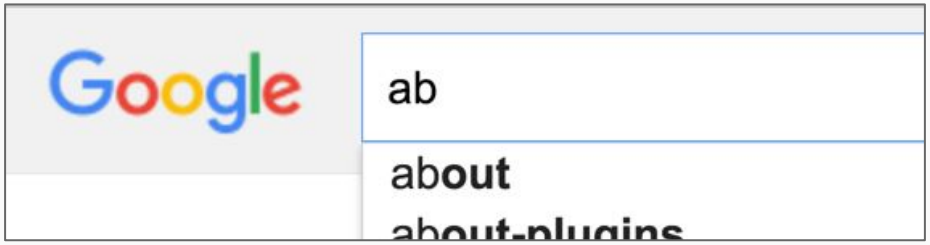
ReactiveX in practice

Auto Complete Box



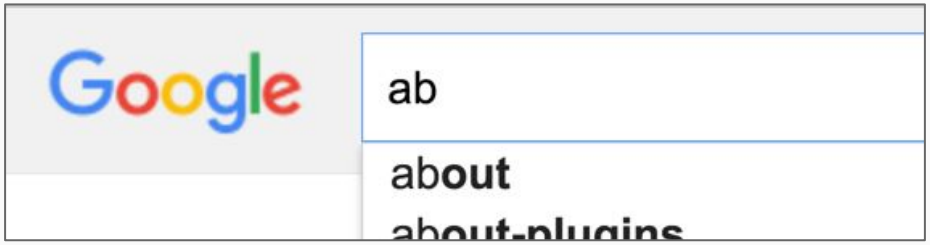
We are all have to deal with this. Web, Android and iOS developers.

Auto Complete Box



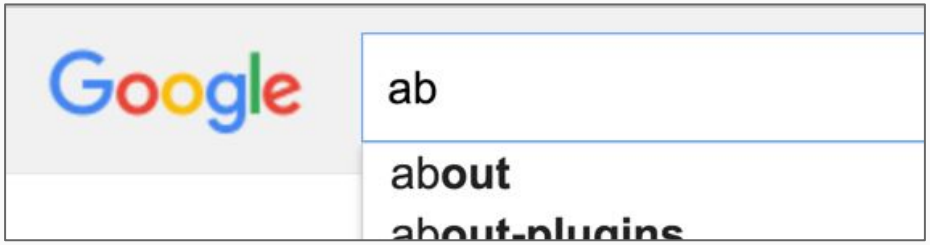

```
textBox.onKeyPress(value =>
     sendRequest('http://api.me/search?q=' + value).then(result =>
          showResult(result);
     );
);
```

Auto Complete Box



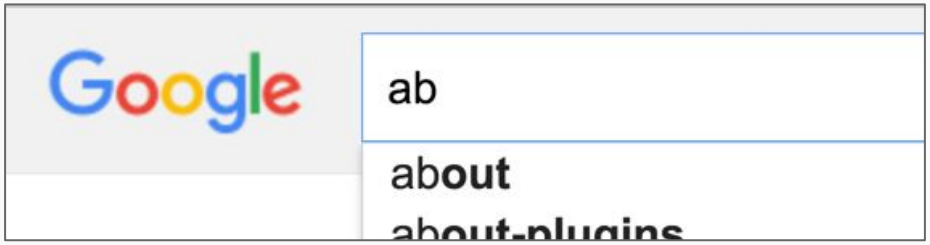

```
var t = timer;
textBox.onKeyPress(value =>
     clearTimeout(t);
     t = setTimeout(() => {
          sendRequest('http://api.me/search?q=' + value).then(result =>
               showResult(result);
          );
     }, 300);

);
```

Auto Complete Box



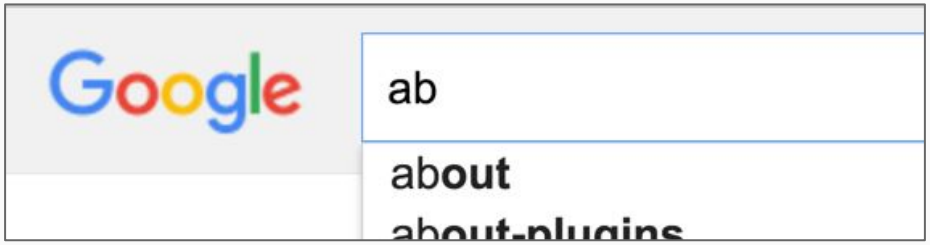

```
var t = timer;
var lastValue = ‘’;
textBox.onKeyPress(value =>
     lastValue = value;
     clearTimeout(t);
     t = setTimeout(() => {
          sendRequest(‘http://api.me/search?q=’ + value).then(result =>
               if (value === lastValue) {
                    showResult(result);
               }
          );
     }, 300);
);
```

Auto Complete Box




```
var t = timer;
var lastValue = ‘’;
textBox.onKeyPress(value =>
     lastValue = value;
     clearTimeout(t);
     t = setTimeout(() => {
          sendRequest(‘http://api.me/search?q=’ + value).then(result =>
               // Check if the value sent by last request have value
               // of the last changed value from textbox value to be
               // save as the value that we saved in the lastValue
               // variable blah balh blah...
               if (value === lastValue) {
                    showResult(result);
               }
          );
     }, 300);
);
```

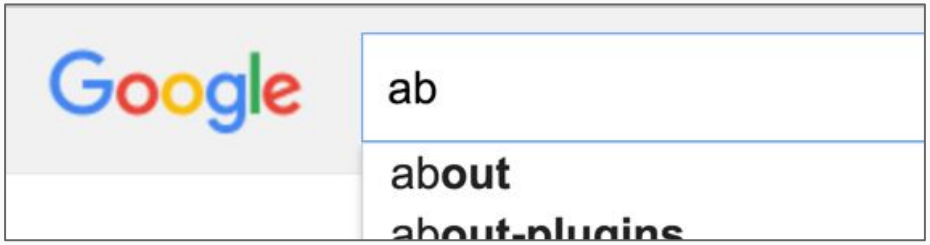
Auto Complete Box




```
var t = timer;
var lastValue = ‘’;
textBox.onKeyPress(value =>
     lastValue = value;
     clearTimeout(t);
     t = setTimeout(() => {
          sendRequest(‘http://api.me/search?q=’ + value).then(result =>
               // Check if the value sent by last request have value
               // of the last changed value from textbox value to be
               // save as the value that we saved in the lastValue
               // variable blah balh blah...
               if (value === lastValue) {
                    showResult(result);
               }
          );
     }, 300);
);
```

Android + iOS, Network on UI Thread?

Auto Complete Box

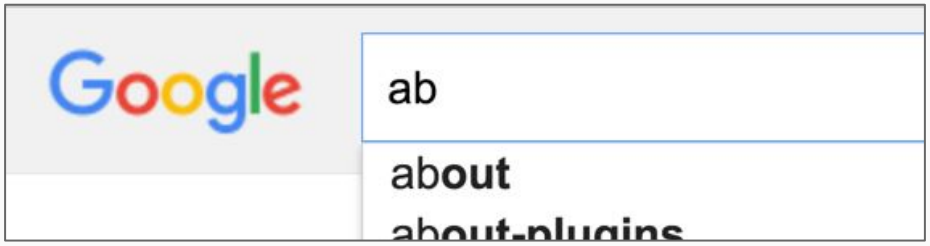



```
var t = timer;
var lastValue = ‘’;
textBox.onKeyPress(value =>
    lastValue = value;
    clearTimeout(t);
    t = setTimeout(() =
        sendRequest(‘h
            // Check
            // of the
            // save a
            // variab
            if (value
                showI
            }
        );
    }, 300);
);
```

Android + iOS, Network on UI Thread?

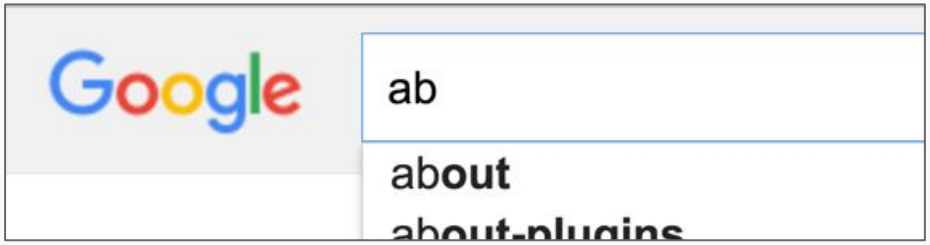
...and when they ask me to maintain this code

Auto Complete Box



...here comes the observable

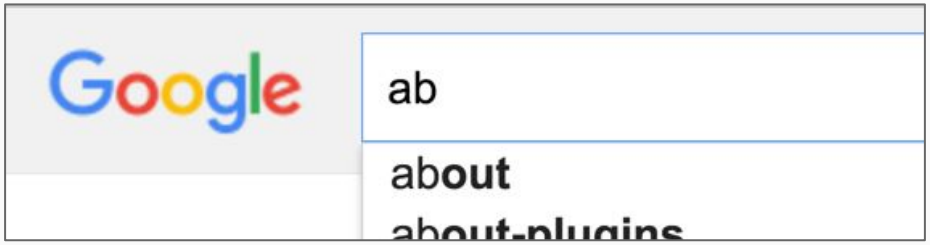
Auto Complete Box




```
var keyPresses = Observable.fromEvent(textBox, 'keypress');

var searchResults = keyPresses
     .debounce(300)
     .map(value => sendRequest('http://api.me/search?q=' + value)
          .takeUntil(keyPresses)
     )
     .concatAll()

searchResults.forEach(showResult);
```

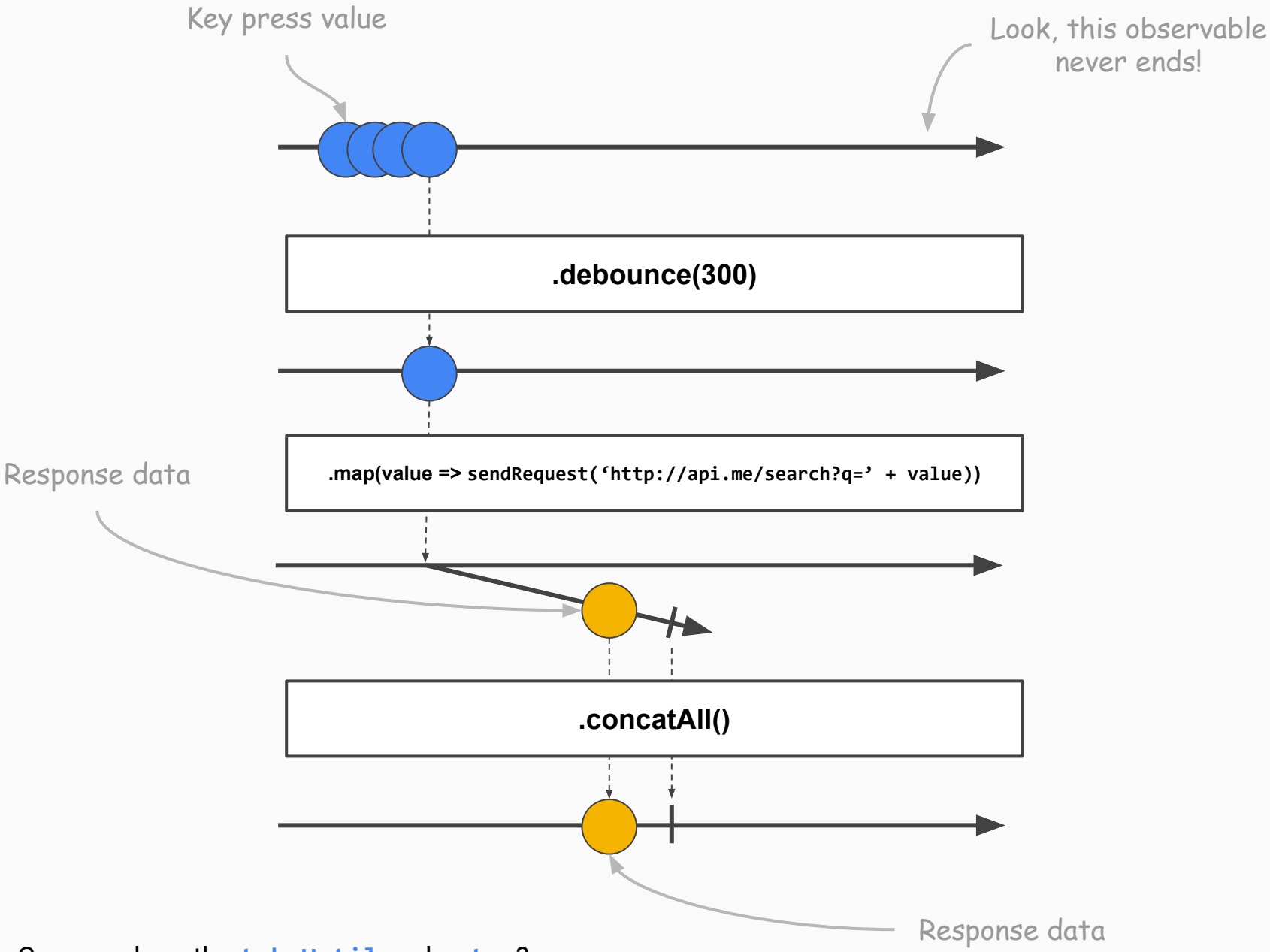
```
.forEach(result => showResult(result))
```

Auto Complete Box



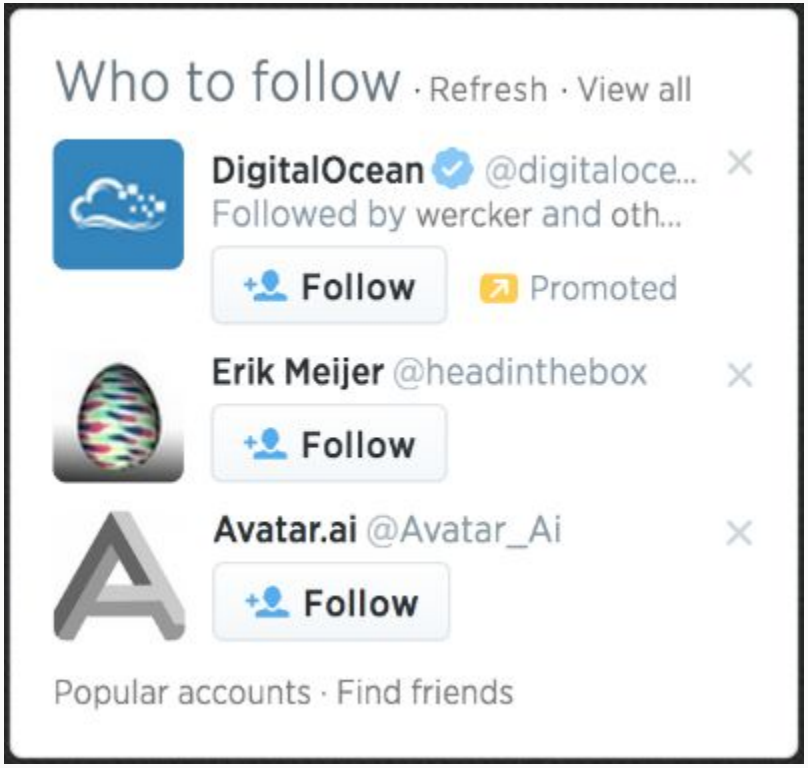

```
var keyPresses = Observable.fromEvent(textBox, 'keypress');

var searchResults = keyPresses
     .debounce(300)
     .map(value => sendRequest('http://api.me/search?q=' + value)
          .retry(3)
          .takeUntil(keyPresses)
     )
     .concatAll()

searchResults.forEach(
     result => showResult,
     error => showError
);
```

Bonus retry and error handling!



Can you draw the `takeUntil` and `retry`?

Twitter suggestion box



https://gist.github.com/staltz/868e7e9bc2a7b8c1f754

My “improved” version:
http://jsfiddle.net/trungdq88/qzsLzskz/4/

Who to follow Refresh

this will not be displayed x

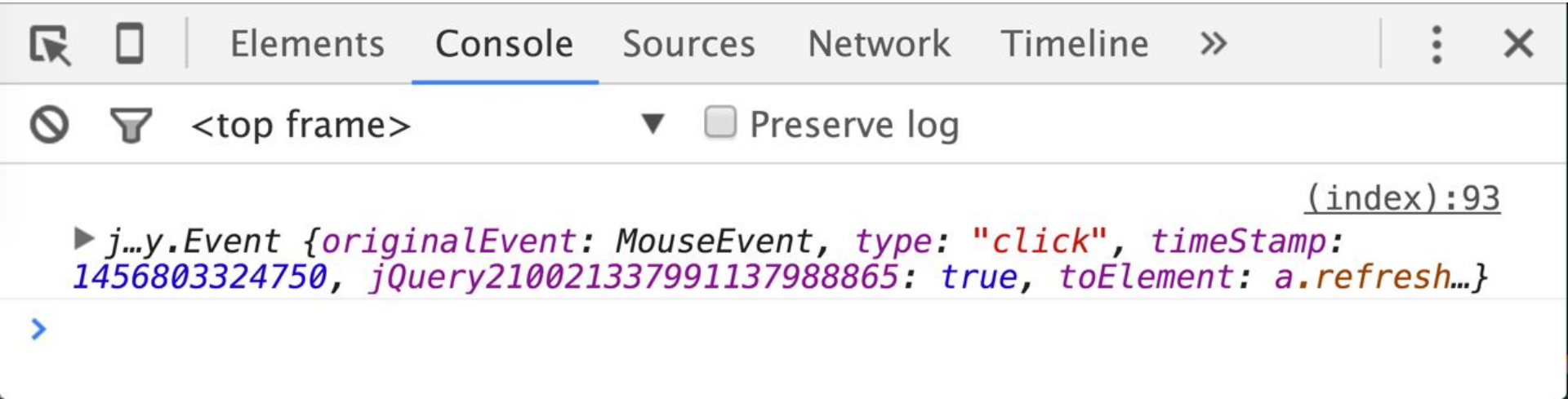
neither this x

nor this x

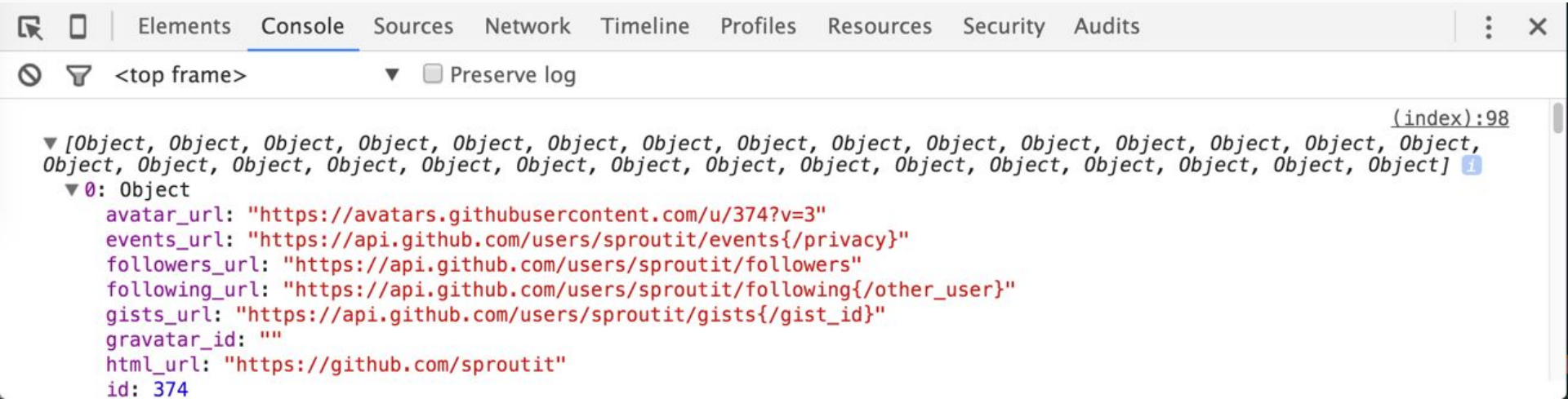
nor this x

```
var $refreshButton = $('.refresh');
```

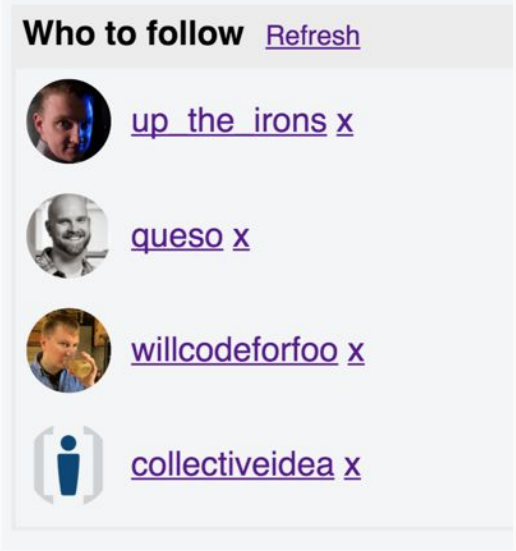
```
var $refreshButton = $('.refresh');

var refreshHandler =
     Rx.Observable.fromEvent($refreshButton[0], 'click')
       .subscribe(function (event) {
         console.log(event);
       });
```



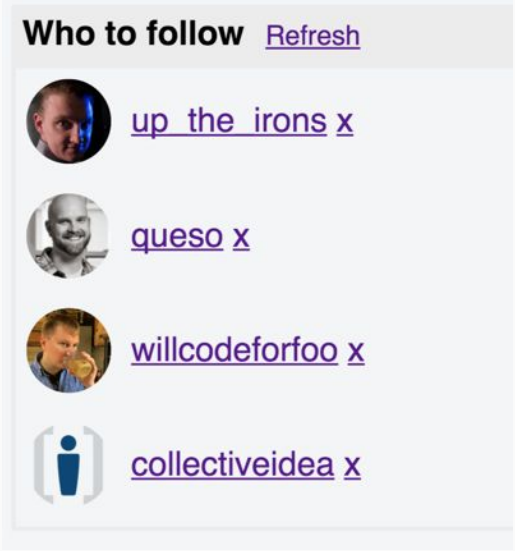
```
var $refreshButton = $('.refresh');

var refreshHandler =
     Rx.Observable.fromEvent($refreshButton[0], 'click')
       .flatMap(function () {
          return Rx.Observable.fromPromise(
            $.getJSON(getRandomUserUrl())
          );
       })
       .subscribe(function (response) {
          console.log(response);
       });
```



```
var suggestions = $('.suggestions');
var $refreshButton = $('.refresh');

var refreshHandler =
     Rx.Observable.fromEvent($refreshButton[0], 'click')
       .flatMap(function () {
          return Rx.Observable.fromPromise(
            $.getJSON(getRandomUserUrl())
          );
       })
       .subscribe(function (response) {
          var numberUser = $suggestions.length;
          var items = getRandomItemsFromArray(response, numberUser);

          for (var i = 0; i < numberUser; i++) {
            var $target = $($suggestions[i]);
            $target.find('.username').text(items[i].login);
            $target.find('img').attr('src', items[i].avatar_url);
          }
       });
```

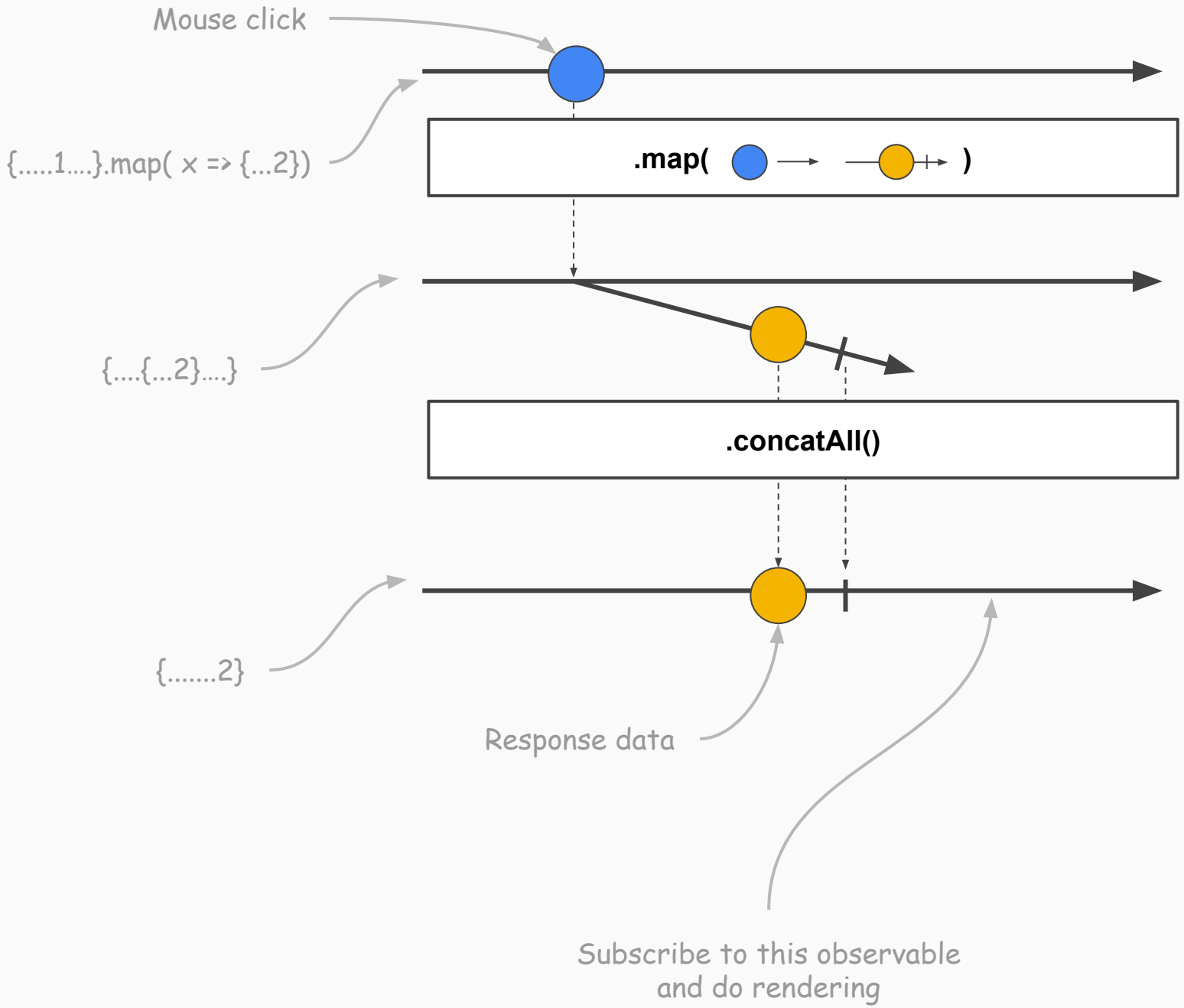


```
var suggestions = $('.suggestions');
var $refreshButton = $('.refresh');

var refreshHandler =
     Rx.Observable.fromEvent($refreshButton[0], 'click')
       .startWith('Initate click')
       .flatMap(function () {
          return Rx.Observable.fromPromise(
            $.getJSON(getRandomUserUrl())
          );
       })
       .subscribe(function (response) {
          var numberUser = $suggestions.length;
          var items = getRandomItemsFromArray(response, numberUser);

          for (var i = 0; i < numberUser; i++) {
            var $target = $($suggestions[i]);
            $target.find('.username').text(items[i].login);
            $target.find('img').attr('src', items[i].avatar_url);
          }
       });
```



```
// View references

var suggestions = $('.suggestions');
var $refreshButton = $('.refresh');

// Event Handler

var refreshHandler =
     Rx.Observable.fromEvent($refreshButton[0], 'click')
       .startWith('Initate click')
       .flatMap(function () {
          return Rx.Observable.fromPromise(
            $.getJSON(UserModel.getRandomUserUrl())
          );
       });

// View rendering

refreshHandler
  .subscribe(function (response) {
     var numberUser = $suggestions.length;
     var items = getRandomItemsFromArray(response, numberUser);
     for (var i = 0; i < numberUser; i++) {
       var $target = $($suggestions[i]);
       $target.find('.username').text(items[i].login);
       $target.find('img').attr('src', items[i].avatar_url);
     }
  });
```

```
// Model

...
```


Mouse click
{.....1....}.map(x => {...2})
.map( )
{....{...2}....}
.concatAll()
{.......2}
Response data
Subscribe to this observable
and do rendering

Twitter suggestion box

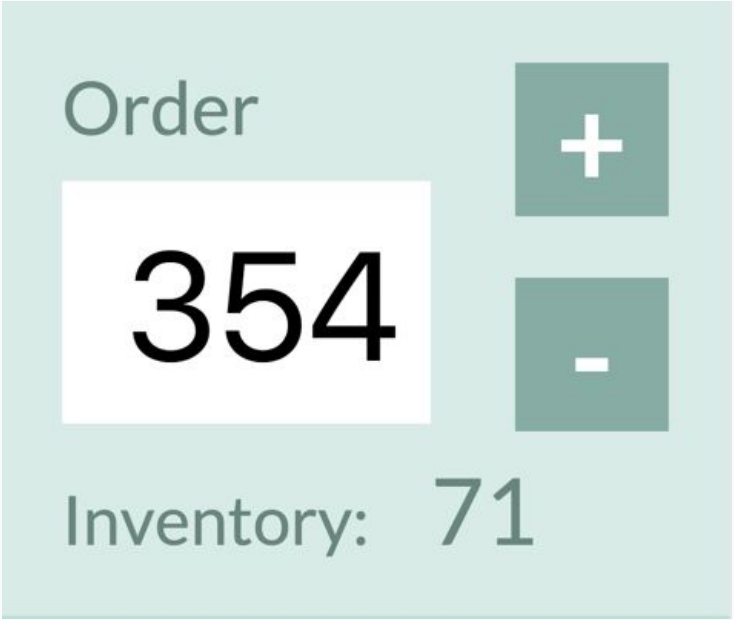
Full article (with source code - JavaScript)

https://gist.github.com/staltz/868e7e9bc2a7b8c1f754

My “improved” version:

http://jsfiddle.net/trungdq88/qzsLzskz/4/

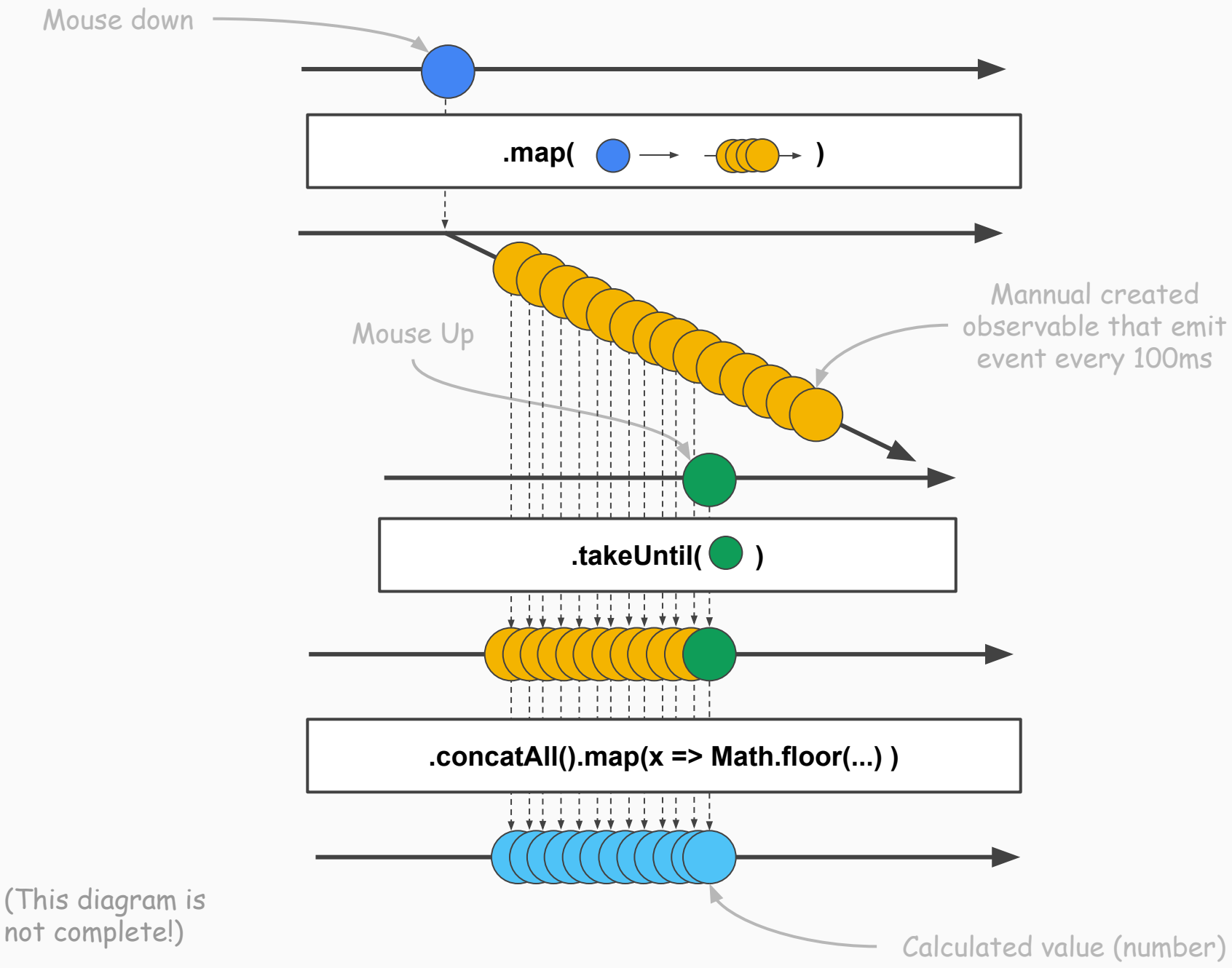
Accelerated number input



https://jsfiddle.net/trungdq88/0y0o2jtv/8/

```
var theNumber = 0;

var accelerateIncrease = function (btn) {
  return Rx.Observable.just(1)
    .concat(
      Rx.Observable.interval(100)
      .delay(300)
      .map(function (n) {
        return Math.floor(n / 10) * 10 + 1;
      })
      .takeUntil(Rx.Observable.fromEvent(btn, 'mouseup'))
    );
};

Rx.Observable.fromEvent(upBtn, 'mousedown')
  .flatMap(accelerateIncrease.bind(null, upBtn))
  .subscribe(function (value) {
    theNumber += value;
    number.innerText = theNumber;
  });

Rx.Observable.fromEvent(downBtn, 'mousedown')
  .flatMap(accelerateIncrease.bind(null, downBtn))
  .subscribe(function (value) {
    theNumber -= value;
    number.innerText = theNumber;
  });
```

https://jsfiddle.net/trungdq88/0y0o2jtv/8/


Mouse down
.map( )
Mannual created observable that emit event every 100ms
Mouse Up
.takeUntil( )
.concatAll().map(x => Math.floor(...))
(This diagram is not complete!)
Calculated value (number)

Recap

Overview

Overview

1. What is ReactiveX (Rx)?

Overview

1. What is ReactiveX (Rx)?

    “An API for asynchronous programming with observable streams”

Overview

1. What is ReactiveX (Rx)?

   “An API for asynchronous programming with observable streams”

2. Idea of Rx, rethink about events

Overview

1. What is ReactiveX (Rx)?

   “An API for asynchronous programming with observable streams”

2. Idea of Rx, rethink about events

   Events are just collections over time

Overview

1. What is ReactiveX (Rx)?

   “An API for asynchronous programming with observable streams”

2. Idea of Rx, rethink about events

   Events are just collections over time

3. Get to know some basic Rx operator

Overview

1. What is ReactiveX (Rx)?

   “An API for asynchronous programming with observable streams”

2. Idea of Rx, rethink about events

   Events are just collections over time

3. Get to know some basic Rx operator

   map, filter, forEach, takeUntil, concatAll

Overview

1. What is ReactiveX (Rx)?

   “An API for asynchronous programming with observable streams”

2. Idea of Rx, rethink about events

   Events are just collections over time

3. Get to know some basic Rx operator

   map, filter, forEach, takeUntil, concatAll

4. How Rx solves popular async problems by examples

Overview

1. What is ReactiveX (Rx)?

   “An API for asynchronous programming with observable streams”

2. Idea of Rx, rethink about events

   Events are just collections over time

3. Get to know some basic Rx operator

   map, filter, forEach, takeUntil, concatAll

4. How Rx solves popular async problems by examples

   *Can Rx solves **your** problems?*

References

This slide: http://j.mp/reactivex-sss

Documentations

Official documentation http://reactivex.io/documentation/
ReactiveX operator with marble diagrams http://rxmarbles.com
Model-view-viewmodel Wiki page https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel
Reactive Programming https://en.wikipedia.org/wiki/Reactive_programming
Testing your Rx application https://github.com/Reactive-Extensions/RxJS/blob/master/doc/gettingstarted/testing.md

Examples

Twitter suggestion box example https://gist.github.com/staltz/868e7e9bc2a7b8c1f754
Accelerated number input example https://jsfiddle.net/trungdq88/0y0o2jtv/8/
Twitter suggestion box modified by me http://jsfiddle.net/trungdq88/qzsLzskz/4/

Videos

Functional Reactive Programming with RxJava https://www.youtube.com/watch?v=_t06LRX0DV0
Async JavaScript with Reactive Extensions https://www.youtube.com/watch?v=XRYN2xt11Ek
Observable Pattern Implementations - Use of ReactiveX https://www.youtube.com/watch?v=Xj1Z701qdOM

Thank you!

Q/A
Discussion

Discussion

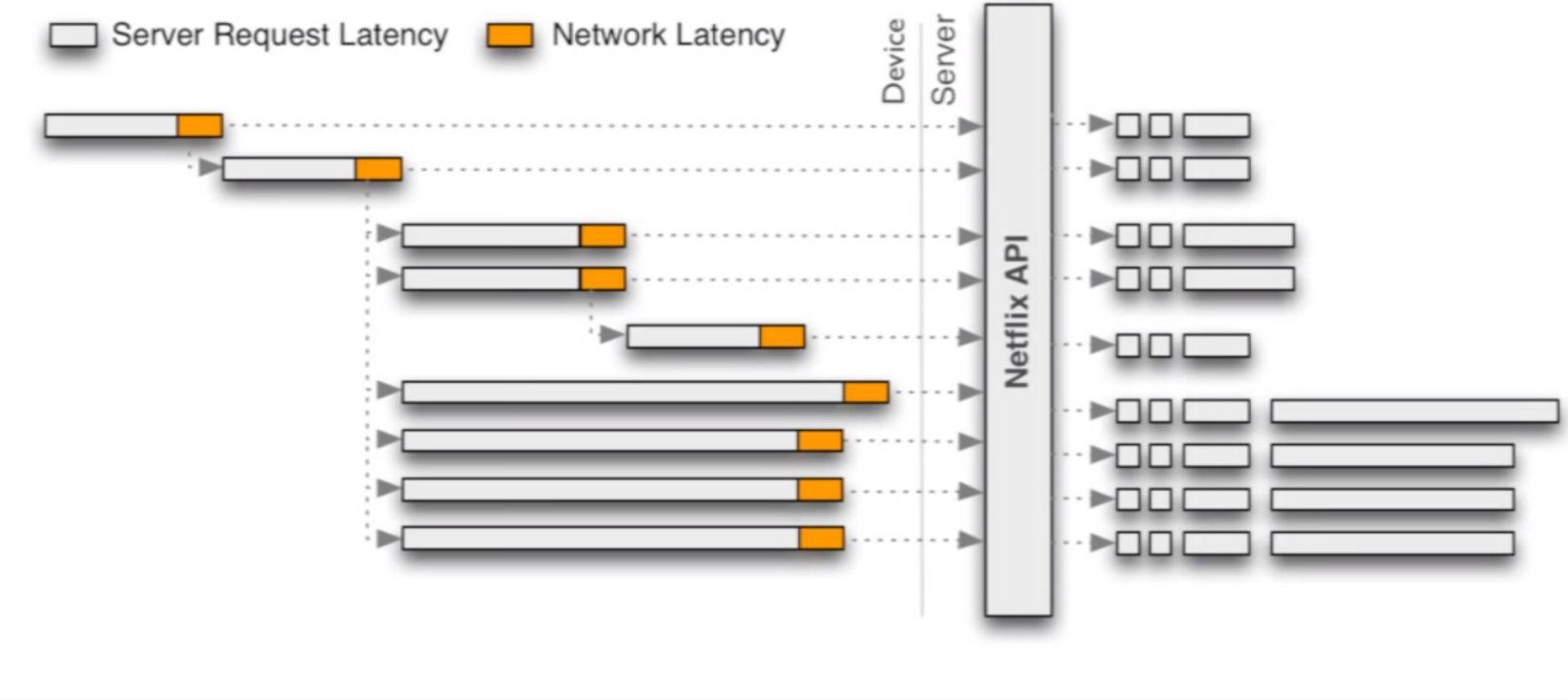
- When to use Rx? Where to use Rx?
- Rx on the “back”.
- How does it work with MVVM application?
- Testing in Rx?
- Reactive Programming?

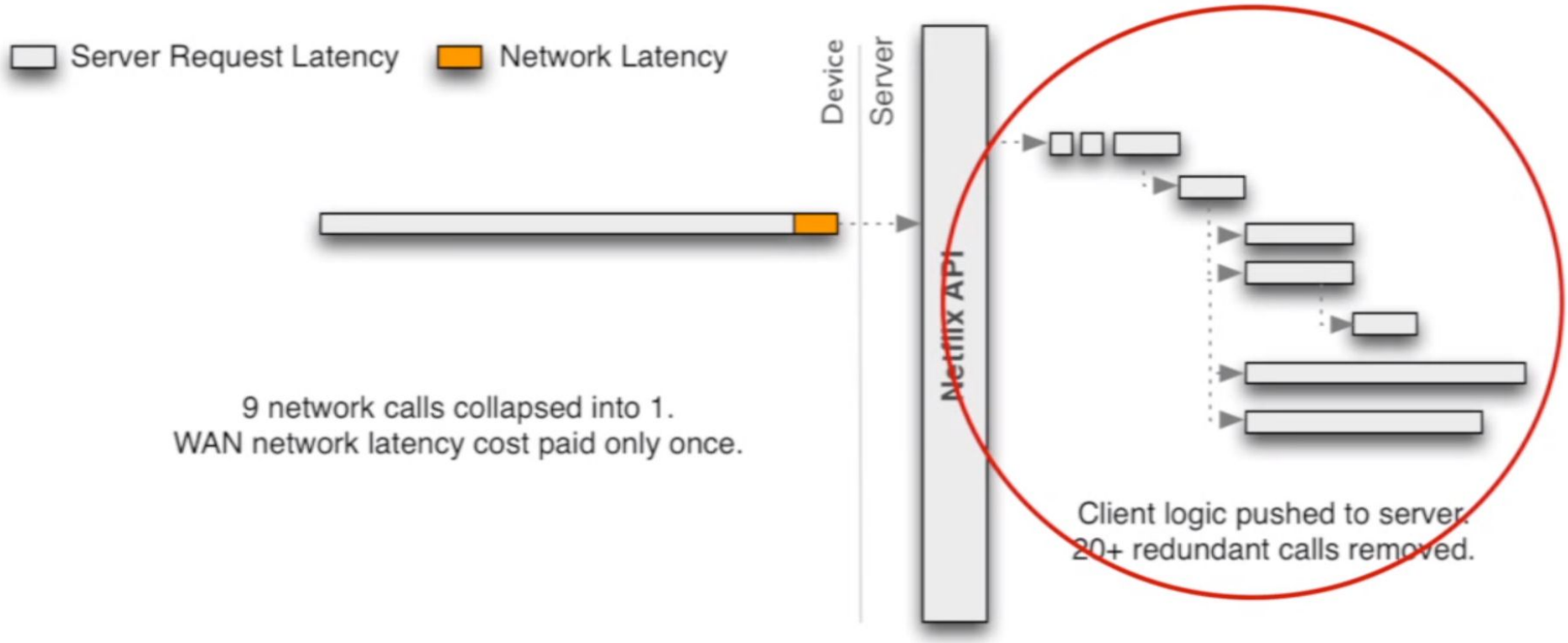
When and where to use ReactiveX

- Probably your Auto Complete search box in the future.
- Event based applications.
- Complicated custom UI behaviors, interactions, animations, events, hand gestures…
- Complex asynchronous data requests.
- View-Model binding in your MVVM applications?
- Your idea?

Rx on the “back”

Netflix uses RxJava to re-archiecture their backend


Server Request Latency
Network Latency
Device
Server
Netflix API



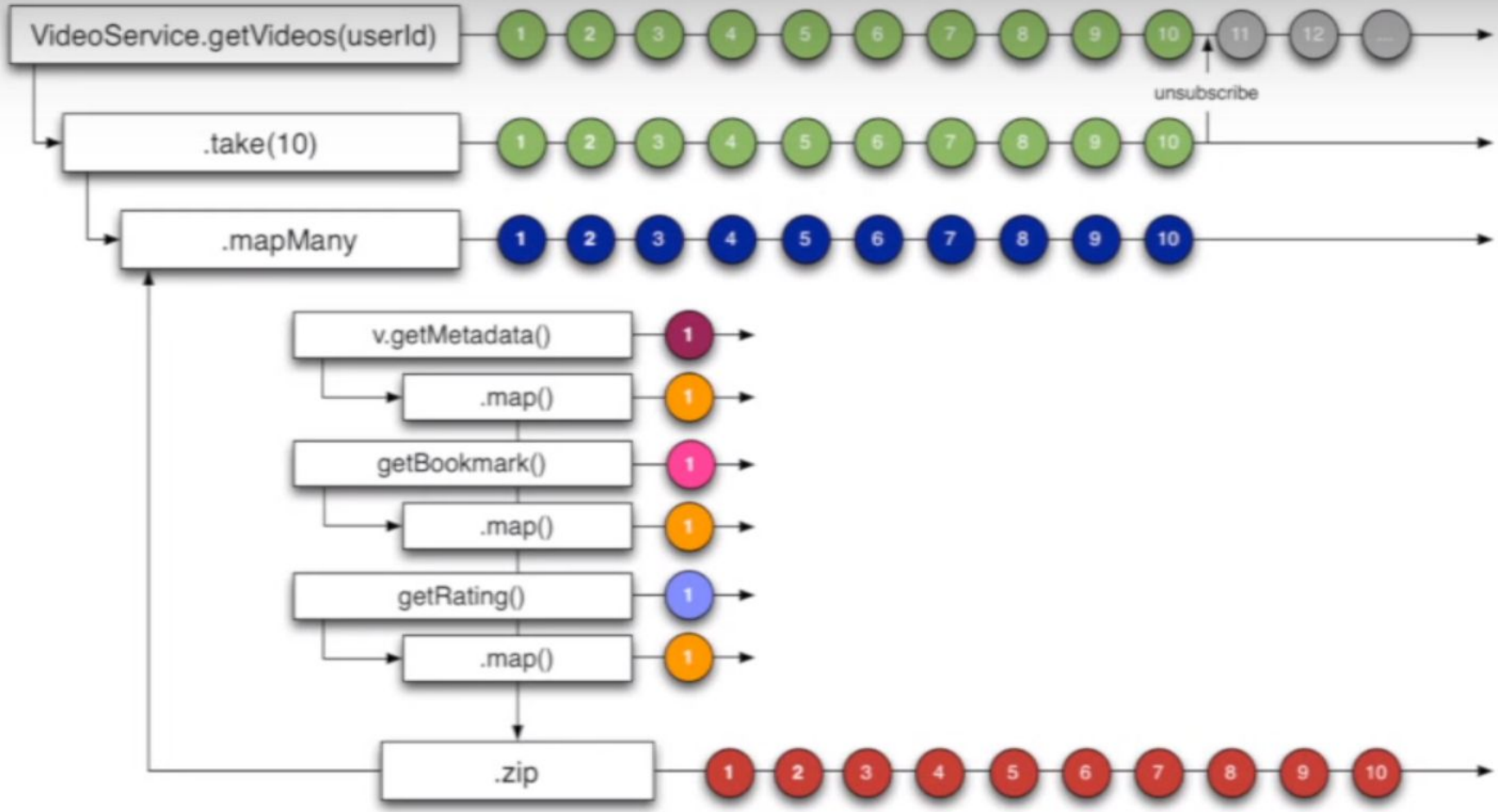
NESTED, CONDITIONAL, CONCURRENT EXECUTION

INSTEAD OF A BLOCKING API ...

```
class VideoService {
   def VideoList getPersonalizedListOfMovies(userId);
   def VideoBookmark getBookmark(userId, videoId);
   def VideoRating getRating(userId, videoId);
   def VideoMetadata getMetadata(videoId);
}
```

... CREATE AN OBSERVABLE API:

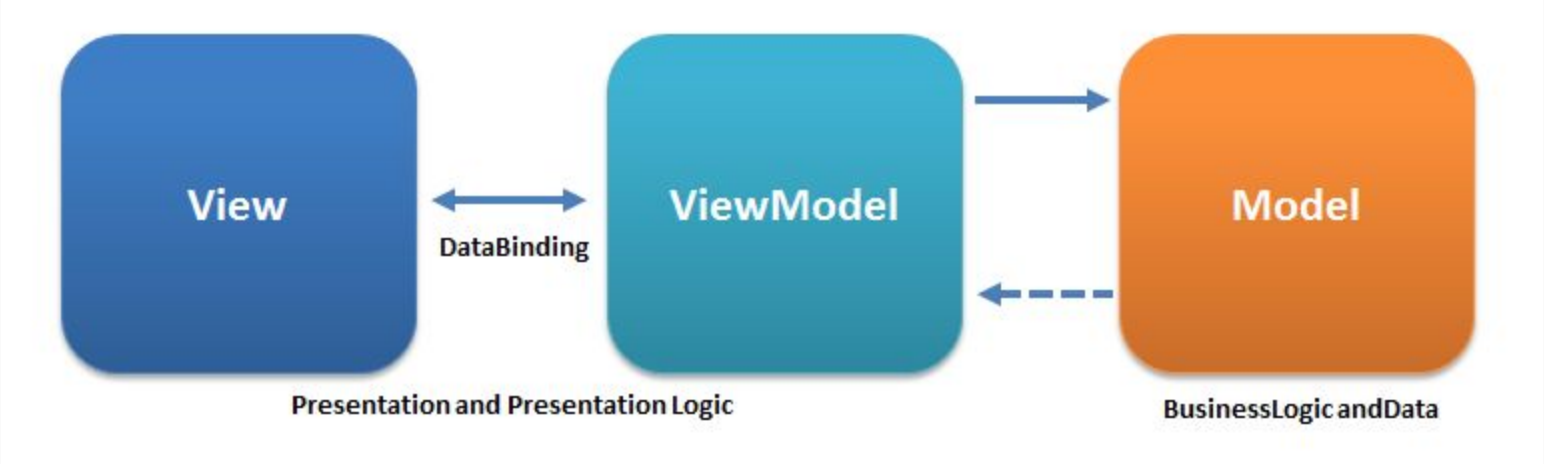
```
class VideoService {
   def Observable<VideoList> getPersonalizedListOfMovies(userId);
   def Observable<VideoBookmark> getBookmark(userId, videoId);
   def Observable<VideoRating> getRating(userId, videoId);
   def Observable<VideoMetadata> getMetadata(videoId);
}
```



[id:1000, title:video-1000-title, length:5428, bookmark:0, rating:[actual:4, average:3, predicted:0]]

Video: https://www.youtube.com/watch?v=_t06LRX0DV0

MVVM


View
DataBinding
ViewModel
Model
Presentation and Presentation Logic
BusinessLogic andData

Testing in Rx

https://github.com/Reactive-Extensions/RxJS/blob/master/doc/gettingstarted/testing.md