Add some FUN to your Functional programming with RXJS

About Me

Senior UI Engineer at Netflix

We do a lot of RX stuff with...

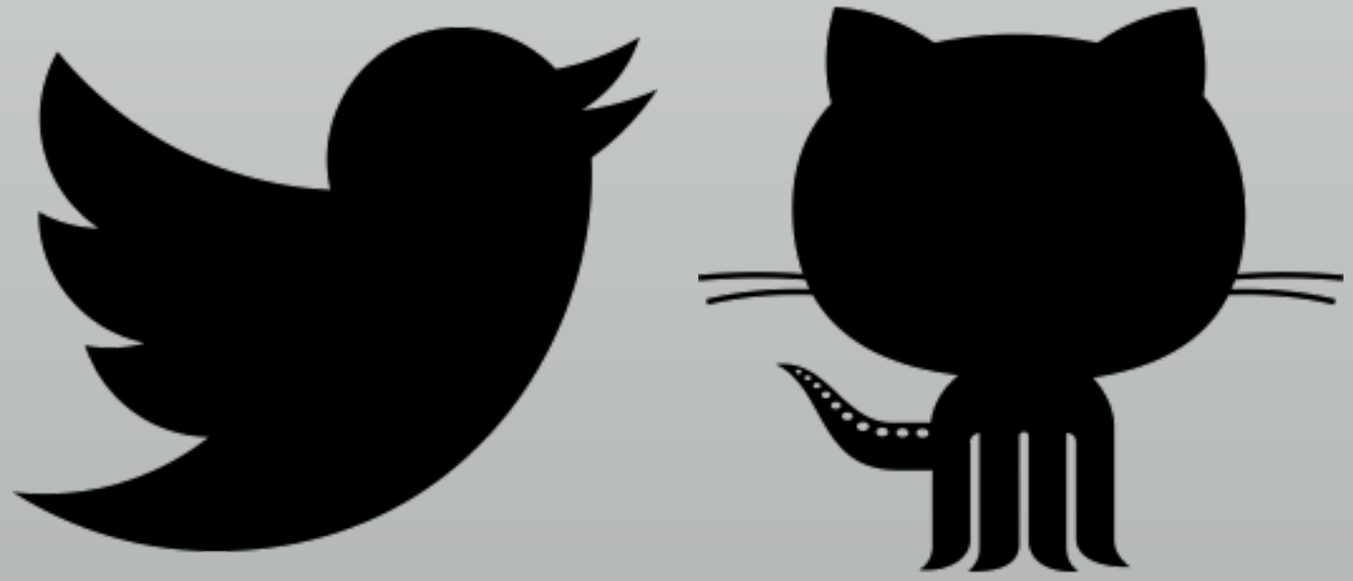
Java...

Groovy...

and JavaScript

About Me

bittersweetryan



Before We Begin

Before We Begin

Be KIND to those of us with OCD and...

Before We Begin

Change those Dropbox icons to B&W!

Functional Review

(With an RX twist)

Functional Review

Map

Filter

Reduce

Zip

Functional Review

Map - Transforms data

```
var searchResultsSets =
 keyups.
   map( function( key ){
      return Observable.getJSON('/search?' +
        input.value)
});
```

Functional Review

Filter - Narrows Collections

```
var searchResultsSets = keyups.
    filter( function ( key ){
        return input.value.length > 1;
    }).
    map( function( key ){
        return Observable.getJSON('/search?' +
          input.value)
    );
```

Functional Review

Reduce - Turns a collection into a single value

```
var html =
  searchResultsSets.
    reduce(function( prev,curr ) {
      return prev + '<li>' + curr.name + '</li>';
  },'' );
```

Functional Review

Zip - Combines two collections

```
var movies = [ 'Super Troopers', 'Pulp Fiction',
  'Fargo' ] ;
var boxArts =[ '/cdn/23212/120x80',
  '/cdn/73212/120x80','/cdn/99212/120x80' ] ;

var withArt = Observable.
  zip(movies, boxarts, function(movie, boxart){
    return {title : movie, boxart : boxart};
  });

//[{title : 'Super Troo…', boxart : '/cdn…'},
// {title : 'Pulp Fict…', boxart : '/cdn…' },
// {title : 'Fargo', boxart : 'cdn…' } ]
```

Functional Review

Observable Data Streams Are Like Crazy Straws

forEach

reduce

distinctUntilChanged

map

filter

throttle

keyPresses Observable

Thinking Functionally

Thinking Functionally

Replace loops with map, reduce, and filter.

```
searchResults = Observable.
  getJSON('/people?' + input.value);

searchResults.
  forEach( function( reps ){
    var names = [];
    for( var i = 1; i < resp; i++ ){
      var name = data[i].fName + ' ' +
        data[i].lName;
      names.push( name );
    }
});
```

Thinking Functionally

Replace loops with map and reduce.

```
searchResults = Observable.
  getJSON('/people?' +  input.value).
  map( function( data ){
    return data.fName + data.lName;
  } );

searchResults.forEach( function( resp ){
  //resp will now be the names array
})
```

Thinking Functionally

Replace if's with filters.

```
var keyPresses = O.fromEvent( el, 'keyup' )

keyPresses.forEach( function( e ){
  if( e.which === keys.enter ){ //=> no!
    //do something
  }
});
```

Thinking Functionally

Replace if's with filters.

```
var enterPresses = O.fromEvent( el, 'keyup' )
   .filter( function( e ){
      return e.which && e.which === keys.enter;
   });

enterPresses.forEach( function( e ){
   //do something
});
```

Thinking Functionally

Don't put too much in a single stream.

Smaller streams are OK.

```
var submits = O.fromEvent(input,'keypresses').
  throttle().
  map( function( e ){
    return e.which
  } ).
  filter( function( key ){
    return key === keys.enter || keys.escape;
  }).
  map().
  reduce().
  ...
```

Thinking Functionally

Don't put too much in a single stream.

Smaller streams are OK.

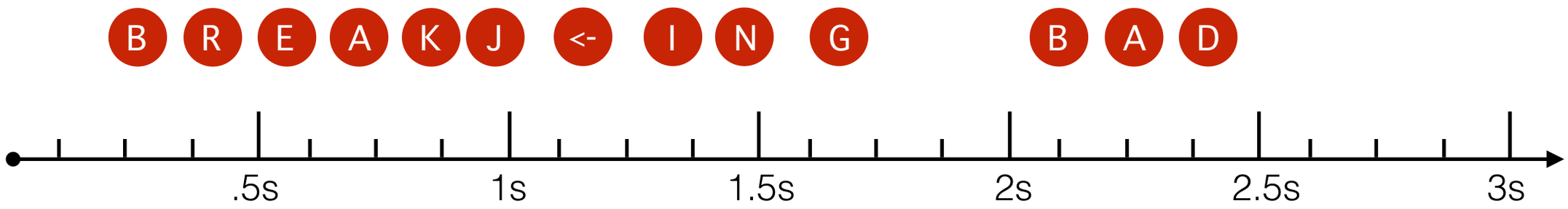
```
var submits =
Observable.fromEvent(input,'keypresses').
  throttle().
  map( function( e ){
     return e.which
  } );

var enters = submits.filter( function( key ) ){
  return e.which === keys.enter;
}

var escapes = submits.filter( function( key ) ){
```

Flattening Patterns

managing concurrency

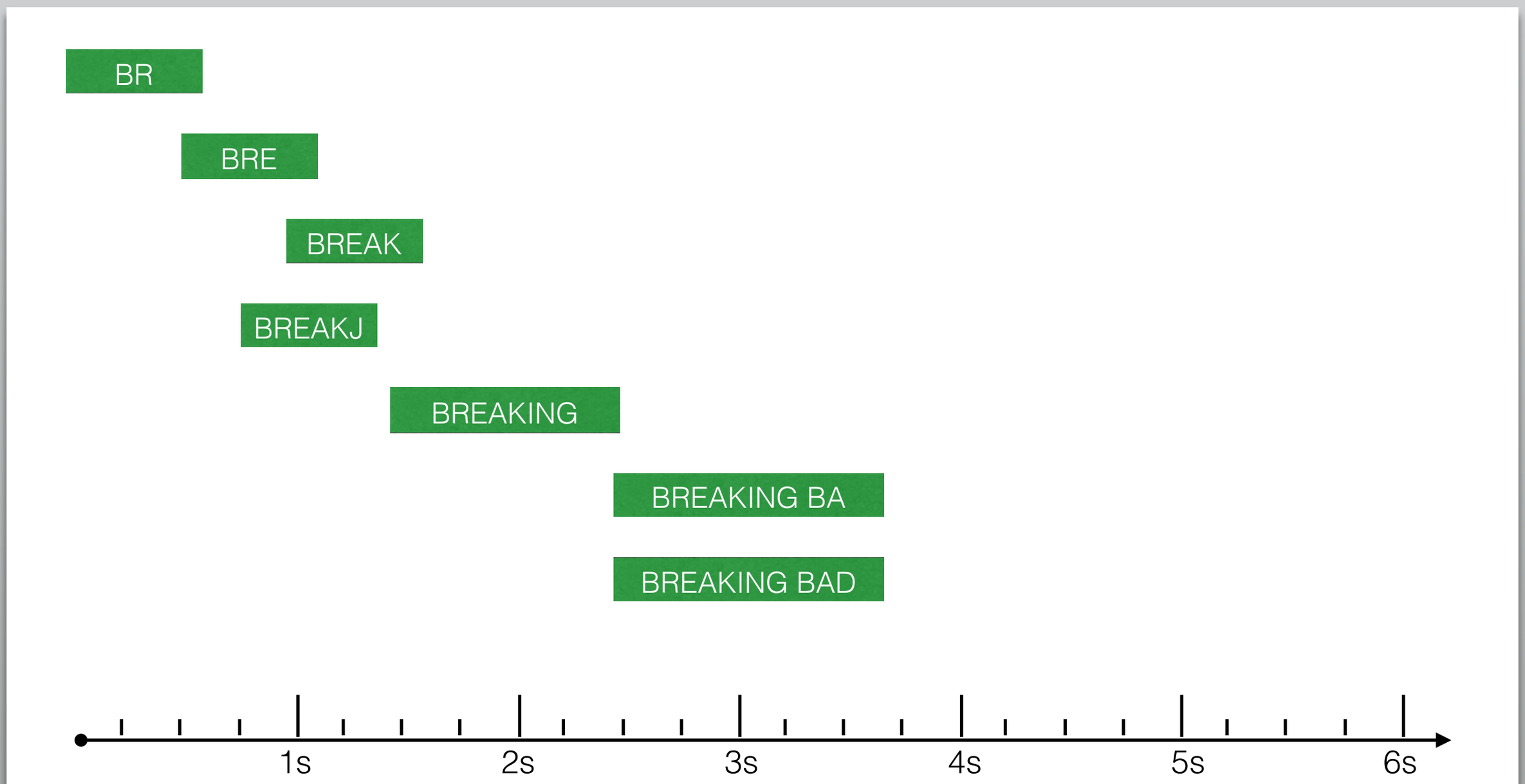
Observables = Events Over Time

Key presses over time…



Observables = Events Over Time

Ajax requests over time…



The Three Musketeers

Starring

Goofy as **merge**

Donald as **concat**

Mickey as **switchLatest**

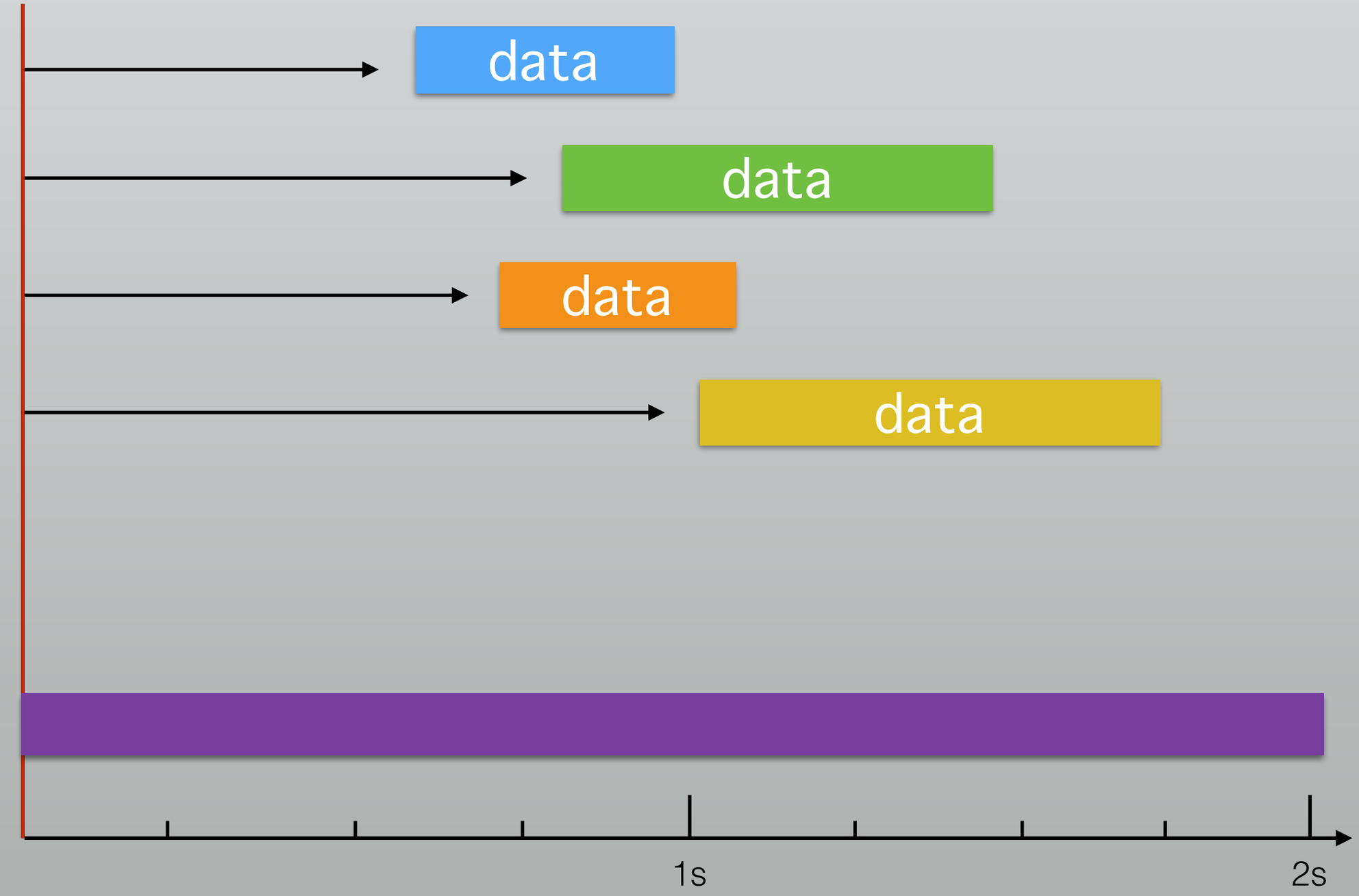
Flattening Patterns

merge - combines **items** in a collection as each item arrives

concat - combines **collections** in the order they arrived

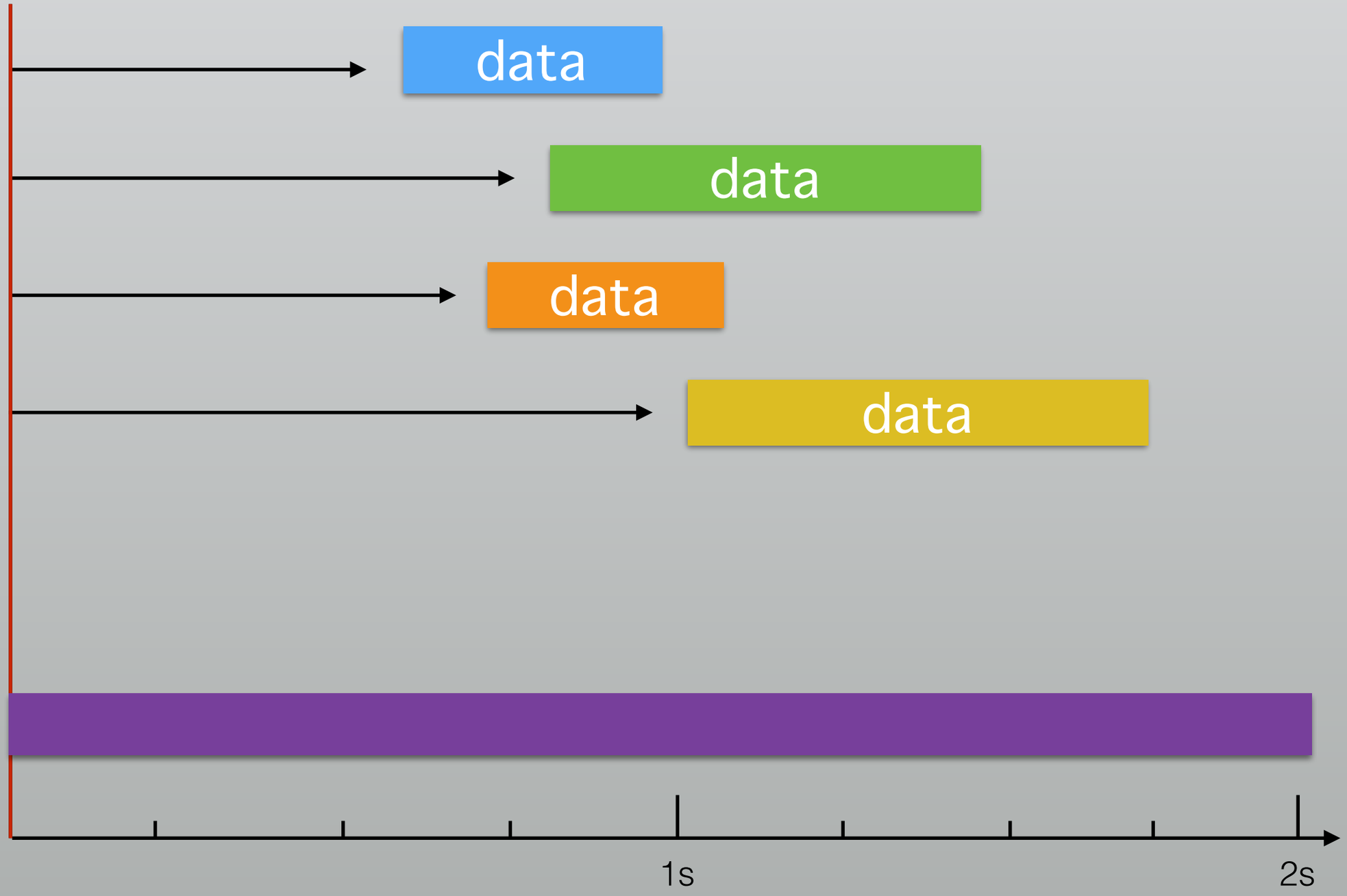
switchLatest - switches to the latest **collection** that arrives

Merge



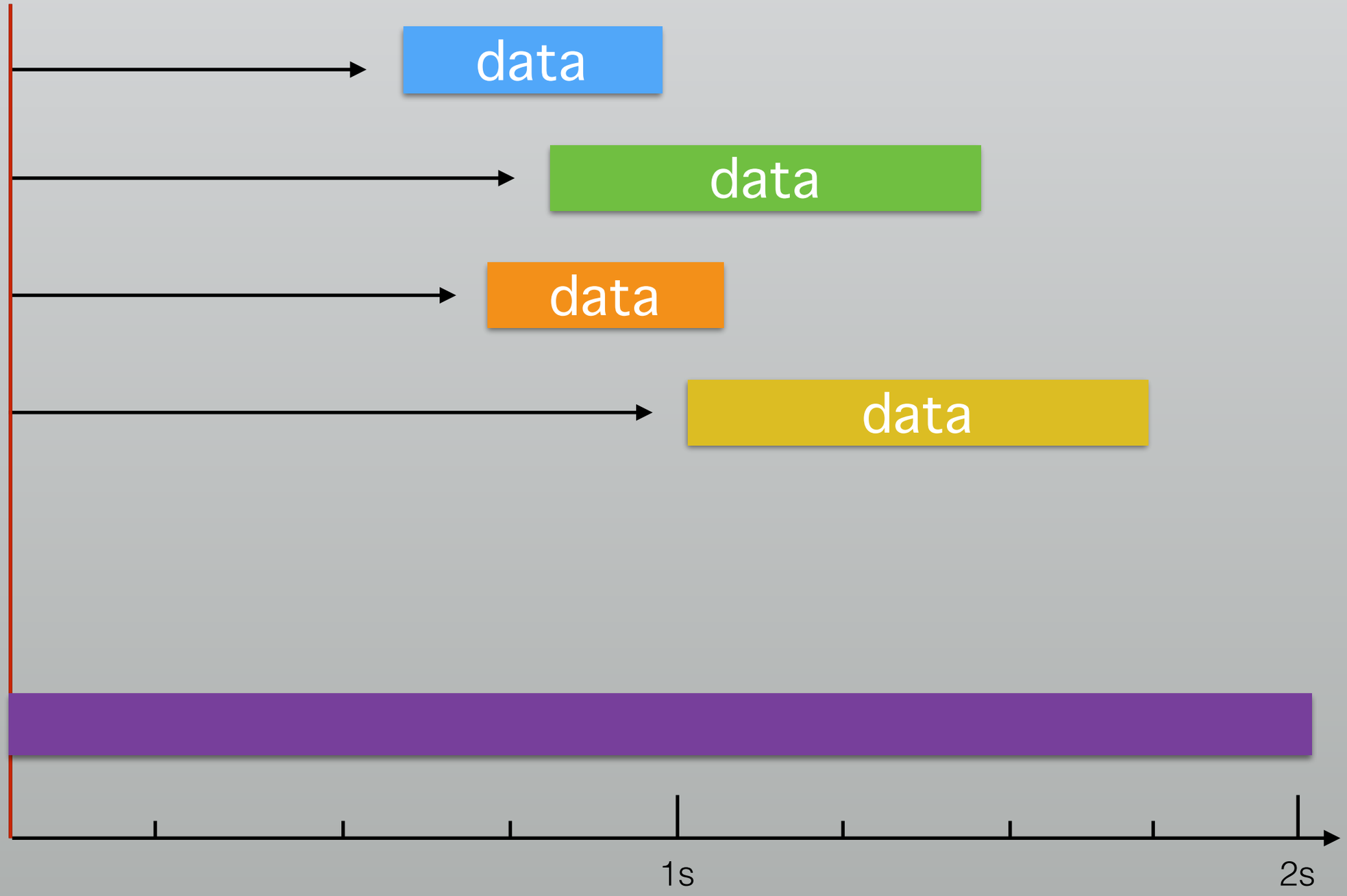
http://jsbin.com/wehusi/13/edit

Concat



http://jsbin.com/fejod/4/edit

SwitchLatest



Building Animated AutoComplete

Putting Everything Together

Animated Autocomplete

Break

Observables = Events Over Time

Simple Widget, High Complexity

- Respond to key presses
- Send off Ajax requests
- Animate out when search results become invalid
- Animate in when new search results come in
  - Don’t show old results
  - Make sure one animation is finished before starting another

Animated Autocomplete

```
var keyups =
  Observable.
    fromEvent( searchInput, 'keypress');
```

Animated Autocomplete

```
var searchResultsSets =
  keyups.
    filter( function ( e ){
      return input.value.length > 1;
    }).
    map( function( e ){
      return Observable.
        getJSON('/search?' + input.value);
    }).
    switchLatest();
```

Animated Autocomplete

```
var animateOuts =
  keyups.
    map(function( resultSet ){
      return animateOut(resultsDiv);
    });
```

```
var animateIns =
  searchResultsSets.
    map( function( resultSet ){
      return Observable.
        of(resultsSet).
        concat(animateIn(resultsDiv));
    });
```

Animated Autocomplete

```
var resultSets =
  animateOuts.
    merge(animateIns).
    concatAll();

resultSets.
  forEach( function( resultSet ){
    if (resultSet.length === 0) {
      $('.search-results').addClass('hidden');
    }
    else {
      resultsDiv.innerHTML = toHTML(resultSet );
    }
  } );
```

Animated Autocomplete

```
var keyups =
      Observable.
            fromEvent( searchInput, 'keypress');

var searchResultsSets =
      keyups.
            filter( function ( e ){
                  return input.value.length > 1;
            }).
            map( function( e ){
                  return Observable.
                        getJSON('/search?' + input.value);
            }).
            switchLatest();
var animateOuts =
      keyups.
            map(function( resultSet ){
                  return animateOut(resultsDiv);
            });

var animateIns =
      searchResultsSets.
            map( function( resultSet ){
                  return Observable.
                        of(resultsSet).
                        concat(animateIn(resultsDiv));
            });

var resultSets =
      animateOuts.
      merge(animateIns).
            concatAll();

resultSets.
      forEach( function( resultSet ){
            if (resultSet.length === 0) {
                  $('.search-results').addClass('hidden');
            }
            else {
                  resultsDiv.innerHTML = toHTML(resultSet );
            }
  } );
```

Thank You.

ranklam@netflix.com
@bittersweetryan