Using Observable Streams in Angular 2

Yakov Fain, Farata Systems

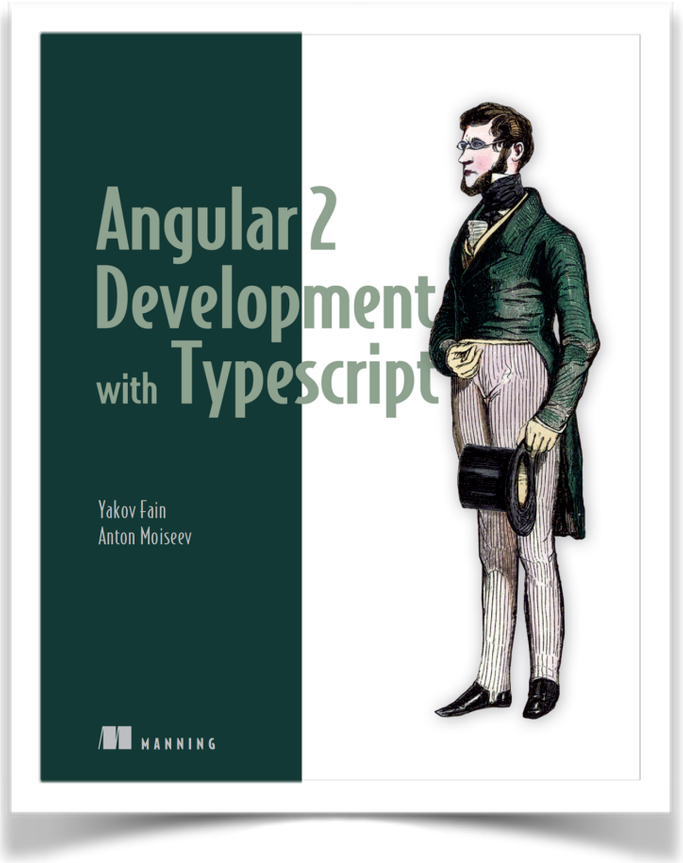
@yfain

About myself

- Co-founder of two companies:
  - Farata Systems
  - SuranceBay

- Java Champion

- Recently co-authored the book Angular Development with TypeScript



The Agenda

- Functional 101
- Pull vs Push
- Intro to RxJS 5
- Observables in Angular 2

A pure function

- Produces no side effects
- The same input always results in the same output
- Doesn’t modify the input
- Doesn’t rely on the external state

A higher-order function can:

- Take one or more functions as argument(s)
- Return a function

Composing functions with chaining

```
let beers = [
    {name: "Stella", country: "Belgium", price: 9.50},
    {name: "Sam Adams", country: "USA", price: 8.50},
    {name: "Bud Light", country: "USA", price: 6.50},
    {name: "Brooklyn Lager", country: "USA", price: 8.00},
    {name: "Sapporo", country: "Japan", price: 7.50}
];

beers
    .filter(beer => beer.price < 8)
    .map(beer => beer.name + ":" + beer.price)
    .forEach(result => console.log("Cheap: " + result));
```

source: es6/function_chaining.html

Pull vs Push in JavaScript

- Pull
  - Looping through an array
  - Looping through ES6 iterables
  - Using ES6 generator functions

- Push
  - ES6 promises
  - ES7 observables

McDonald's

STAMAR ΜΟΝΟΠΡΟΣΩΠΗ ΕΠΕ
ΕΚΜΕΤΑΛΕΥΣΗ ΕΣΤΙΑΤΟΡΙΩΝ
ΠΛ. ΜΟΝΑΣΤΗΡΑΚΙΟΥ 12 ΑΘΗΝΑ
ΑΦΜ:095452825 ΤΗΛ:010 3310866
ΔΟΥ:ΣΤ ΑΘΗΝΩΝ

ΕΥΧΑΡΙΣΤΟΥΜΕ

** ΝΟΜΙΜΗ ΑΠΟΔΕΙΞΗ **

03-10-03 12-17
#000001 ΧΕΙΡΙΣ.1 01
GREEK CHICKEN MG B €4.55
ΜΕ COLA B €0.00
ΜΕΡΙΚΟ €4.55
ΣΥΝΟΛΟ **€4.55**
ΕΥΡΩ
δ1551
ΜΕΤΡΗΤΑ €5.00
ΡΕΣΤΑ €0.45
δ153
ΙΣΟΤ. 1€=340.75δ

ΝΟΜΙΜΕΣ ΑΠΟΔΕΙΞΕΙΣ 000034
ΠΚ 99002033

Async calls with ES6 promises

Limitations of promises

- Can return only one value
- Guaranteed response, i.e. can’t be canceled

```
function getCustomers(){

  let promise = new Promise(
    function (resolve, reject){

      console.log("Getting customers");
       // Emulate an async server call here
       setTimeout(function(){
          let success = true;
          if (success){
          resolve( "John Smith");
        }else{
            reject("Can't get customers");
        }
       },1000);
     });

    return promise;
}
```

```
function getOrders(customer){

  let promise =  new Promise(
      function (resolve, reject){

      // Emulate an async server call here
      setTimeout(function(){
        let success = true;
        if (success){
          resolve( `Found the order 123 for ${customer}`);
        }else{
            reject("Can't get orders");
        }
       },1000);

     });

    return promise;
}
```

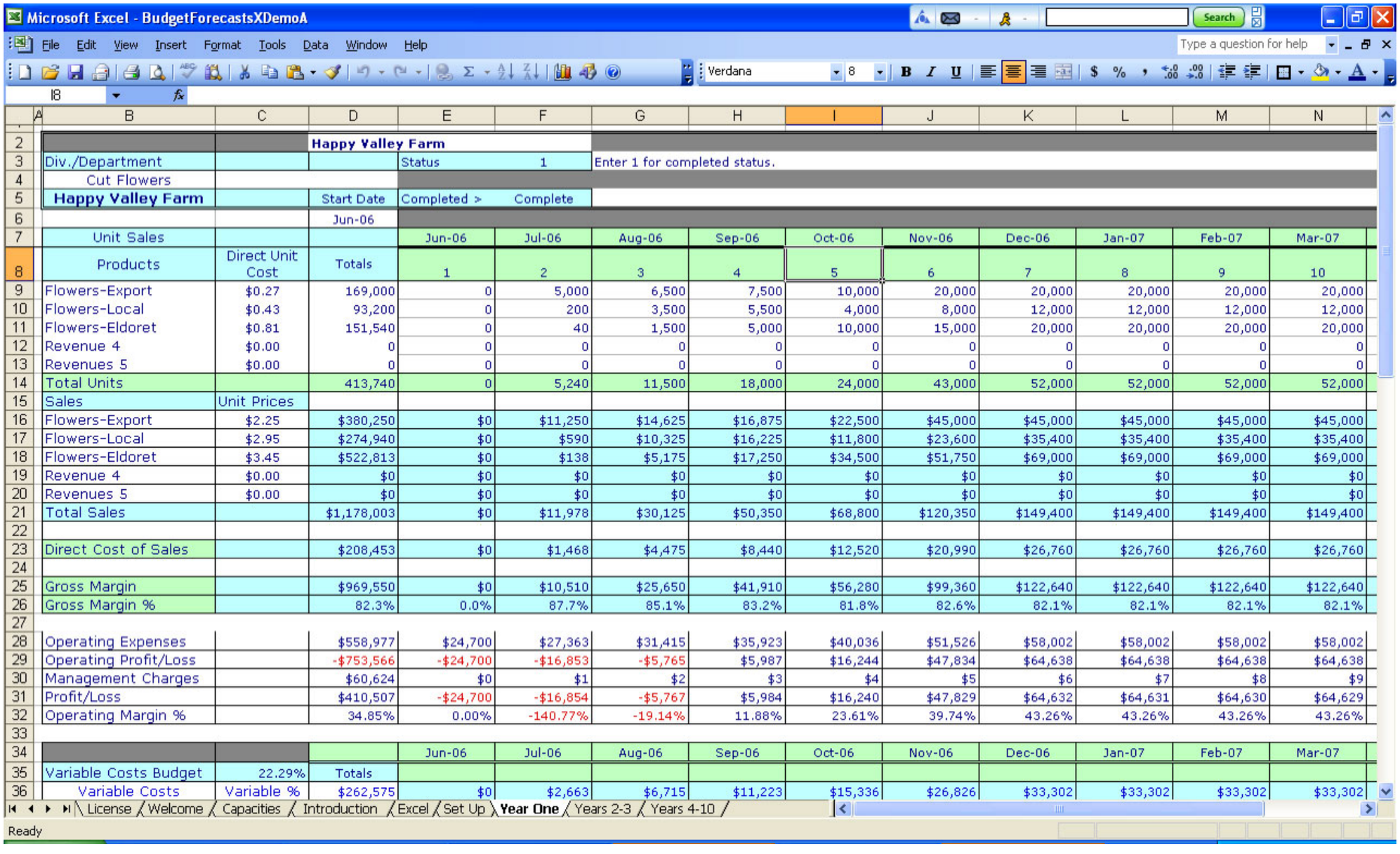
Chaining resolved promises

```
getCustomers()
        .then(cust => {console.log(cust);return cust;})
        .then(cust => getOrders(cust))
        .then(order => console.log(order))
        .catch(err => console.error(err));
```

source: es6/promises.html

Functional reactive programming

Reactive UI


Microsoft Excel - BudgetForecastsXDemoA
File Edit View Insert Format Tools Data Window Help
Verdana 8
I8
Happy Valley Farm
Div./Department
Status 1
Enter 1 for completed status.
Cut Flowers
Happy Valley Farm
Start Date
Completed > Complete
Jun-06
Unit Sales
Products
Direct Unit Cost
Totals
Jun-06 Jul-06 Aug-06 Sep-06 Oct-06 Nov-06 Dec-06 Jan-07 Feb-07 Mar-07
1 2 3 4 5 6 7 8 9 10
Flowers-Export $0.27 169,000 0 5,000 6,500 7,500 10,000 20,000 20,000 20,000 20,000 20,000
Flowers-Local $0.43 93,200 0 200 3,500 5,500 4,000 8,000 12,000 12,000 12,000 12,000
Flowers-Eldoret $0.81 151,540 0 40 1,500 5,000 10,000 15,000 20,000 20,000 20,000 20,000
Revenue 4 $0.00 0 0 0 0 0 0 0 0 0 0 0
Revenues 5 $0.00 0 0 0 0 0 0 0 0 0 0 0
Total Units 413,740 0 5,240 11,500 18,000 24,000 43,000 52,000 52,000 52,000 52,000
Sales Unit Prices
Flowers-Export $2.25 $380,250 $0 $11,250 $14,625 $16,875 $22,500 $45,000 $45,000 $45,000 $45,000 $45,000
Flowers-Local $2.95 $274,940 $0 $590 $10,325 $16,225 $11,800 $23,600 $35,400 $35,400 $35,400 $35,400
Flowers-Eldoret $3.45 $522,813 $0 $138 $5,175 $17,250 $34,500 $51,750 $69,000 $69,000 $69,000 $69,000
Revenue 4 $0.00 $0 $0 $0 $0 $0 $0 $0 $0 $0 $0 $0
Revenues 5 $0.00 $0 $0 $0 $0 $0 $0 $0 $0 $0 $0 $0
Total Sales $1,178,003 $0 $11,978 $30,125 $50,350 $68,800 $120,350 $149,400 $149,400 $149,400 $149,400
Direct Cost of Sales $208,453 $0 $1,468 $4,475 $8,440 $12,520 $20,990 $26,760 $26,760 $26,760 $26,760
Gross Margin $969,550 $0 $10,510 $25,650 $41,910 $56,280 $99,360 $122,640 $122,640 $122,640 $122,640
Gross Margin % 82.3% 0.0% 87.7% 85.1% 83.2% 81.8% 82.6% 82.1% 82.1% 82.1% 82.1%
Operating Expenses $558,977 $24,700 $27,363 $31,415 $35,923 $40,036 $51,526 $58,002 $58,002 $58,002 $58,002
Operating Profit/Loss -$753,566 -$24,700 -$16,853 -$5,765 $5,987 $16,244 $47,834 $64,638 $64,638 $64,638 $64,638
Management Charges $60,624 $0 $1 $2 $3 $4 $5 $6 $7 $8 $9
Profit/Loss $410,507 -$24,700 -$16,854 -$5,767 $5,984 $16,240 $47,829 $64,632 $64,631 $64,630 $64,629
Operating Margin % 34.85% 0.00% -140.77% -19.14% 11.88% 23.61% 39.74% 43.26% 43.26% 43.26% 43.26%
Jun-06 Jul-06 Aug-06 Sep-06 Oct-06 Nov-06 Dec-06 Jan-07 Feb-07 Mar-07
Variable Costs Budget 22.29% Totals
Variable Costs Variable % $262,575 $0 $2,663 $6,715 $11,223 $15,336 $26,826 $33,302 $33,302 $33,302 $33,302
License Welcome Capacities Introduction Excel Set Up Year One Years 2-3 Years 4-10
Ready

Some open-source Rx libraries

- **Rx.NET** - for .Net
- **RxCpp** - for C and C++
- **RxJS** - for JavaScript
- **Rx.rb** - for Ruby
- **Rx.py** - for Python
- **RxJava** - for Java

http://reactivex.io

Libraries with Reactive extensions

http://reactivex.io

- **RxJS**

- Rx.NET, RxCpp, RxJS, Rx.rb, Rx.py, RxSwift, RxScala, RxPHP, RxJava, RxAndroid, RxJavaFX, RxSwing …

RxJS 5

- Github repo:
  https://github.com/ReactiveX/rxjs

- CDN:
  https://npmcdn.com/@reactivex/rxjs@5.0.0-beta.7/dist/global/Rx.umd.js

- Installing locally:

  npm init -y
  npm install rxjs-es —save

- Documentation:
  http://reactivex.io/rxjs

Main RxJS players

- **Observable** - data stream that pushes data over time
- **Observer** - consumer of observable
- **Subscriber** - connects observer with observable
- **Operator** - en-route data transformation

TS39 Observable proposal https://github.com/zenparsing/es-observable

Learning the terms

Data Flow

Data Source

Data Flow

Observable

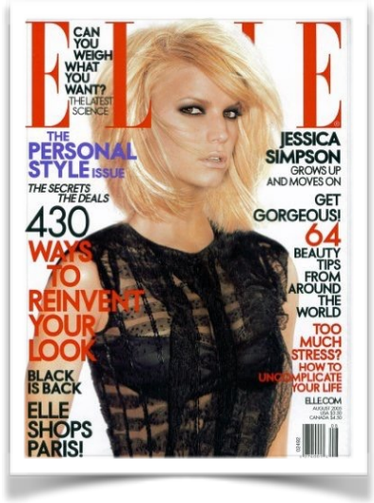
Data Source

Data Flow

Observable

Data Source

next



Data Flow

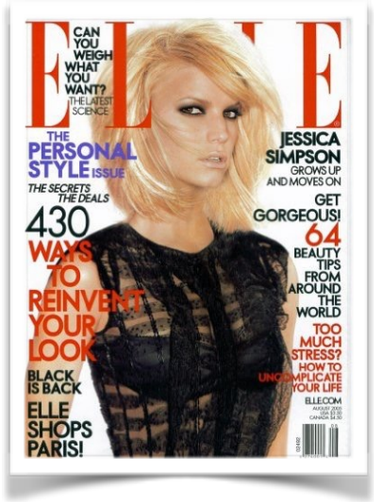
Observer

Subscriber

Observable

Data Source

next



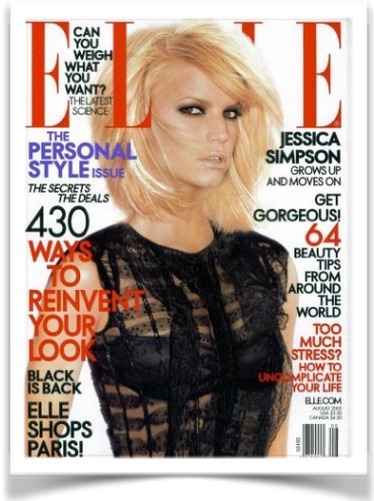
Data Flow

Observer

Subscriber

Observable

Data Source

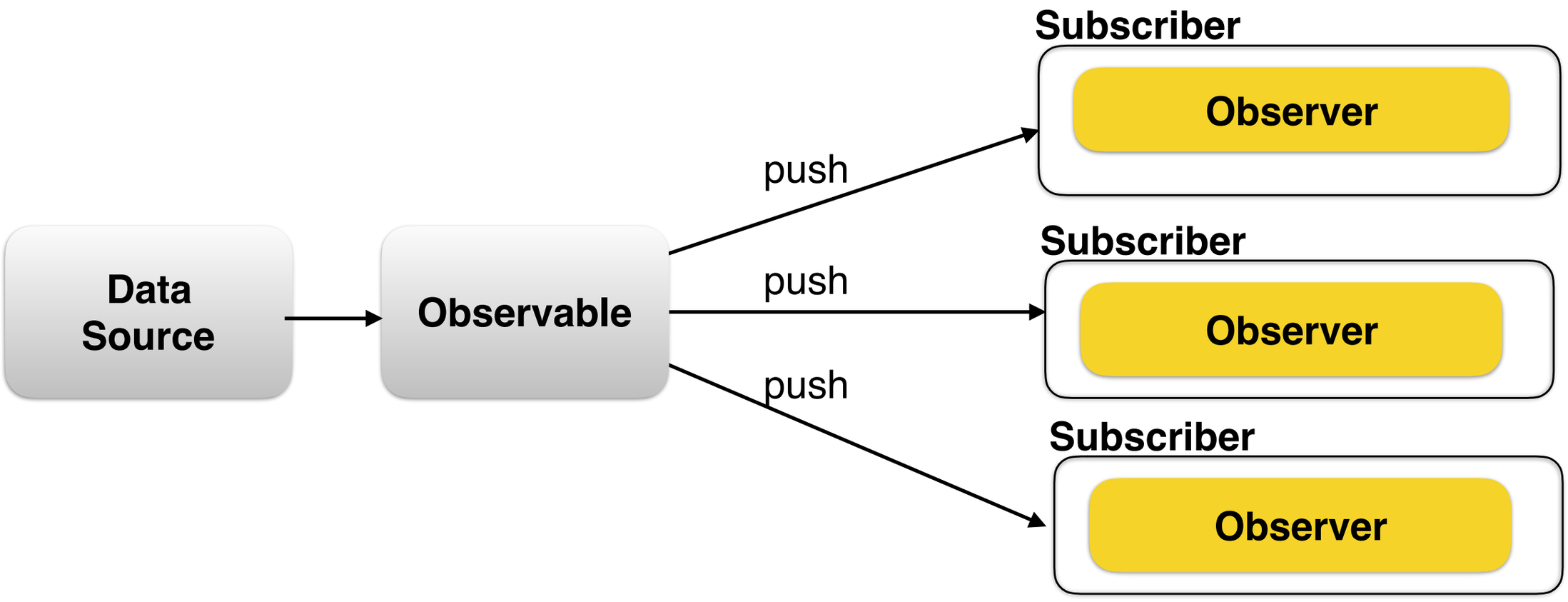


next()

error()

complete()

Event-driven push



Subscribe to messages from Observable and handle them by Observer

The Push Model with Observable Streams

An observable stream

- Can push any number of values over time
- **Cold** observable won't emit any data until someone subscribes
- **Hot** observable emits the data even if there are no subscribers

Observable allows:

- Subscribe/unsubscribe to its data stream
- Emit the next element to the observer
- Notify the observer about errors
- Inform the observer about the stream completion
- Transform the data with operators

Three main functions of an Observable

- Emit the next element
- Throw an error
- Send a signal that the stream is over

An Observer provides

- A function to handle the next object from the stream
- A function to handle errors
- A function to handle end-of-stream

Observables vs Promises

- Observable can return multiple values
- Observables can be canceled
- Observables signal the end-of-stream

Reading RxJS 5 sources

Subscriber.ts

```
export class Subscriber<T>
       extends Subscription implements Observer<T>{
    …
}
```

Observer.ts

```
export interface Observer<T> {
  isUnsubscribed?: boolean;
  next: (value: T) => void;
  error: (err: any) => void;
  complete: () => void;
}
```

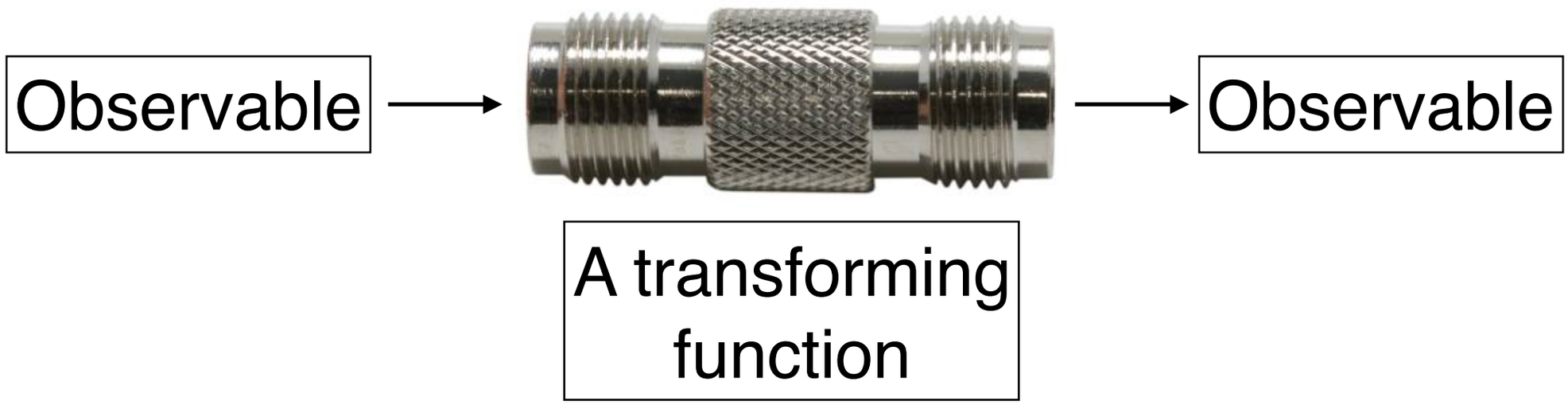
Observable.ts

```
subscribe(observerOrNext?: PartialObserver<T> | ((value: T) => void),
          error?: (error: any) => void,
          complete?: () => void): Subscription { … }
```

Your code:

```
let mySubscription = myObservable.subscribe(myObserver)
…
mySubscribtion.unsubscribe()
```

An Operator



An Operator

Observable →  → Observable

A transforming function

pure function

```
observableBeers
        .filter(beer => beer.price < 8))
```

higher-order function

STAMAR ΜΟΝΟΠΡΟΣΩΠΗ ΕΠΕ
ΕΚΜΕΤΑΛΕΥΣΗ ΕΣΤΙΑΤΟΡΙΩΝ
ΠΛ.ΜΟΝΑΣΤΗΡΑΚΙΟΥ 12 ΑΘΗΝΑ
ΑΦΜ:095452825 ΤΗΛ:010 3310866
ΔΟΥ:ΣΤ ΑΘΗΝΩΝ
ΕΥΧΑΡΙΣΤΟΥΜΕ

** ΝΟΜΙΜΗ ΑΠΟΔΕΙΞΗ **
03-10-03 12-17
#000001 ΧΕΙΡΙΣ.1 01
GREEK CHICKEN MG B €4.55
ΜΕ COLA B €0.00
ΜΕΡΙΚΟ €4.55
ΣΥΝΟΛΟ €4.55
ΕΥΡΩ
δ1551
ΜΕΤΡΗΤΑ €5.00
ΡΕΣΤΑ €0.45
δ153
ΙΣΟΤ. 1€=340.75δ
ΝΟΜΙΜΕΣ ΑΠΟΔΕΙΞΕΙΣ 000034
ΠΚ 99002033

VS

Observable Beer

```
observableBeers = Rx.Observable.from(beers)
    .filter(beer => beer.price < 8)
    .map(beer => beer.name + ": $" + beer.price);

observableBeers
    .subscribe(
        beer => console.log(beer),
        err  => console.error(err),
        ()   => console.log("The sream is over")
    );
```

Observable Beer

No streaming

Streaming begins

```
observableBeers = Rx.Observable.from(beers)
    .filter(beer => beer.price < 8)
    .map(beer => beer.name + ": $" + beer.price);

observableBeers
    .subscribe(
        beer => console.log(beer),
        err  => console.error(err),
        ()   => console.log("The sream is over")
    );
```

Observable Beer

Creating Observable

Operators

```
observableBeers = Rx.Observable.from(beers)
    .filter(beer => beer.price < 8)
    .map(beer => beer.name + ": $" + beer.price);

observableBeers
    .subscribe(
      beer => console.log(beer),
      err => console.error(err),
      () => console.log("The sream is over")
    );
```

Observer

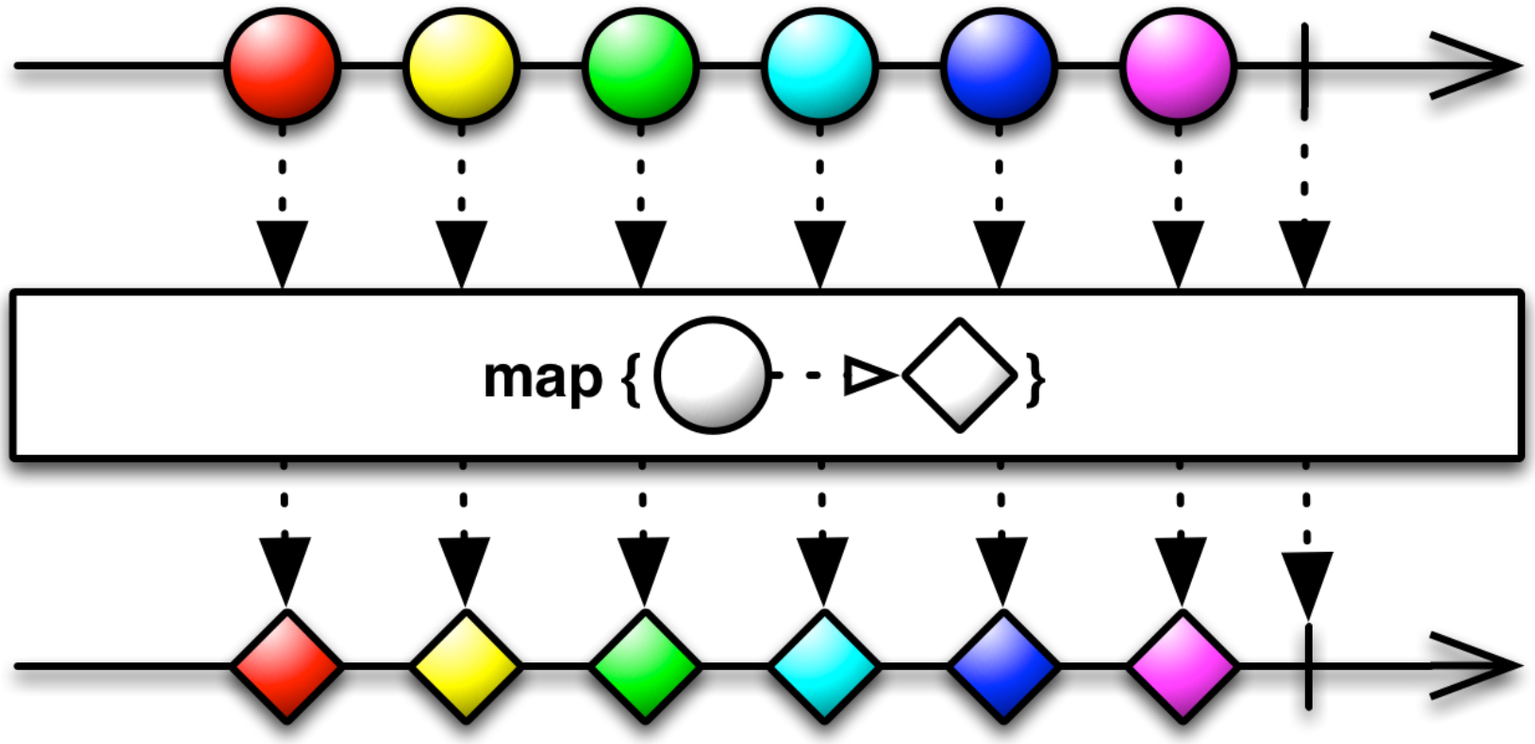
Demo

observable_beer

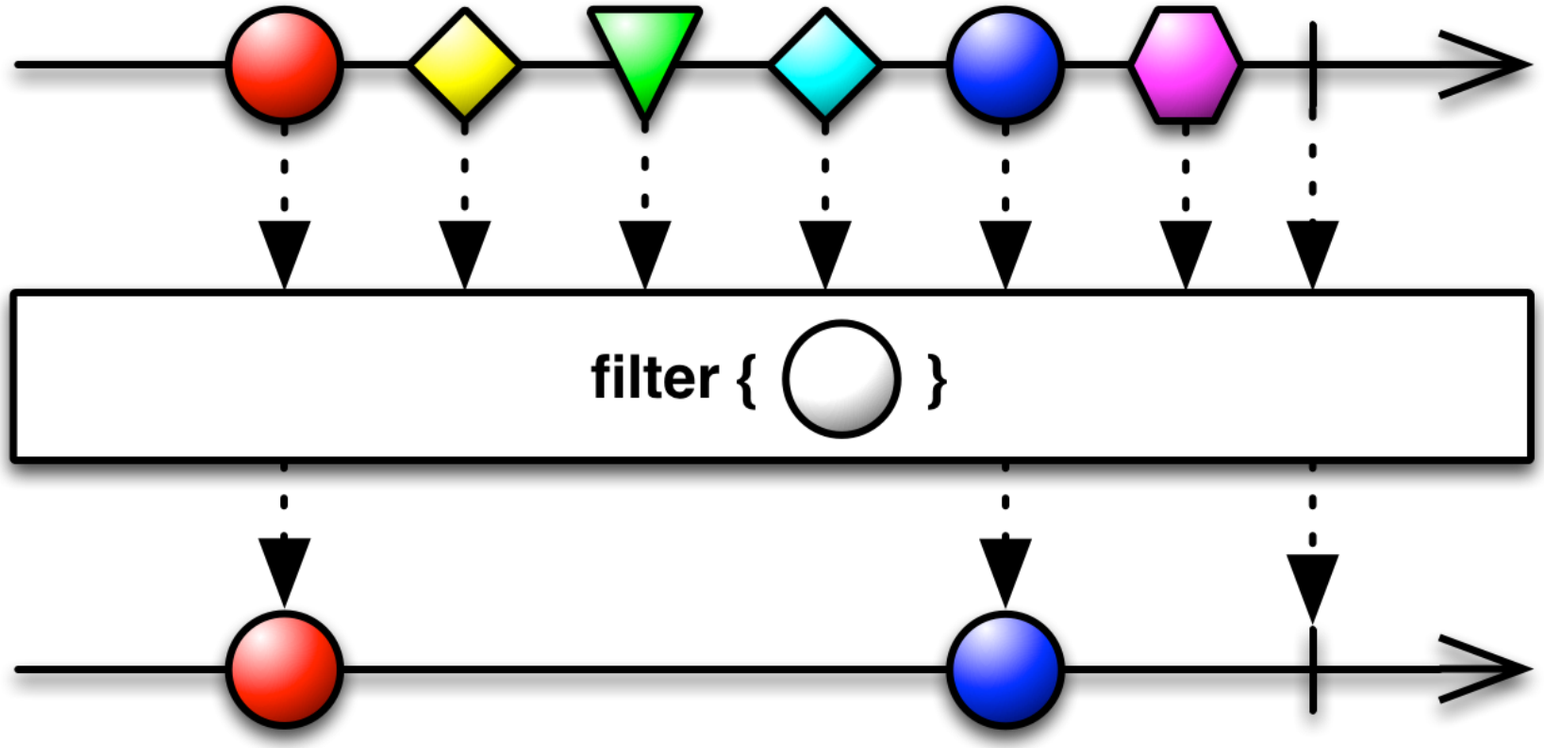
Marble Diagrams

http://rxmarbles.com

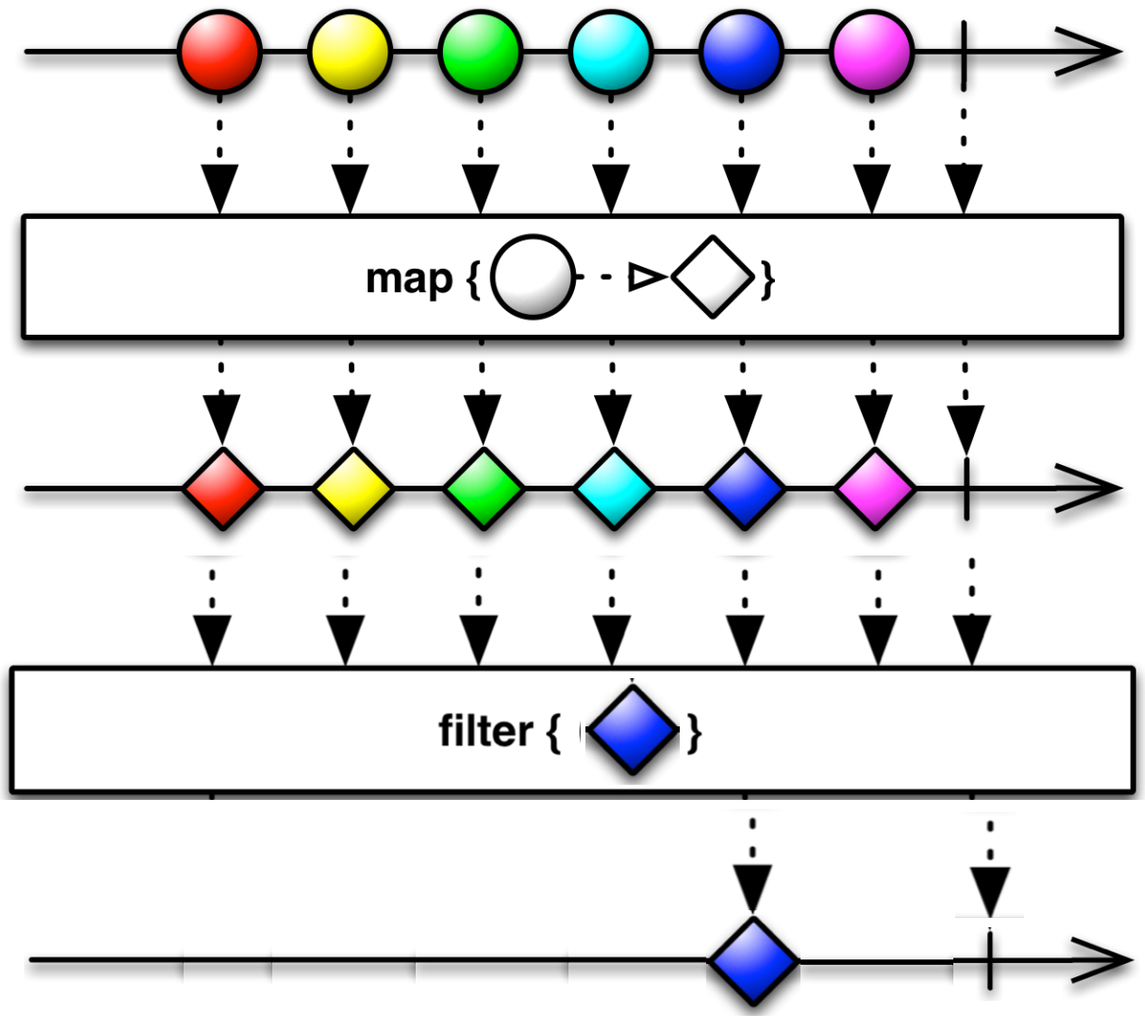
Observable map(function){}



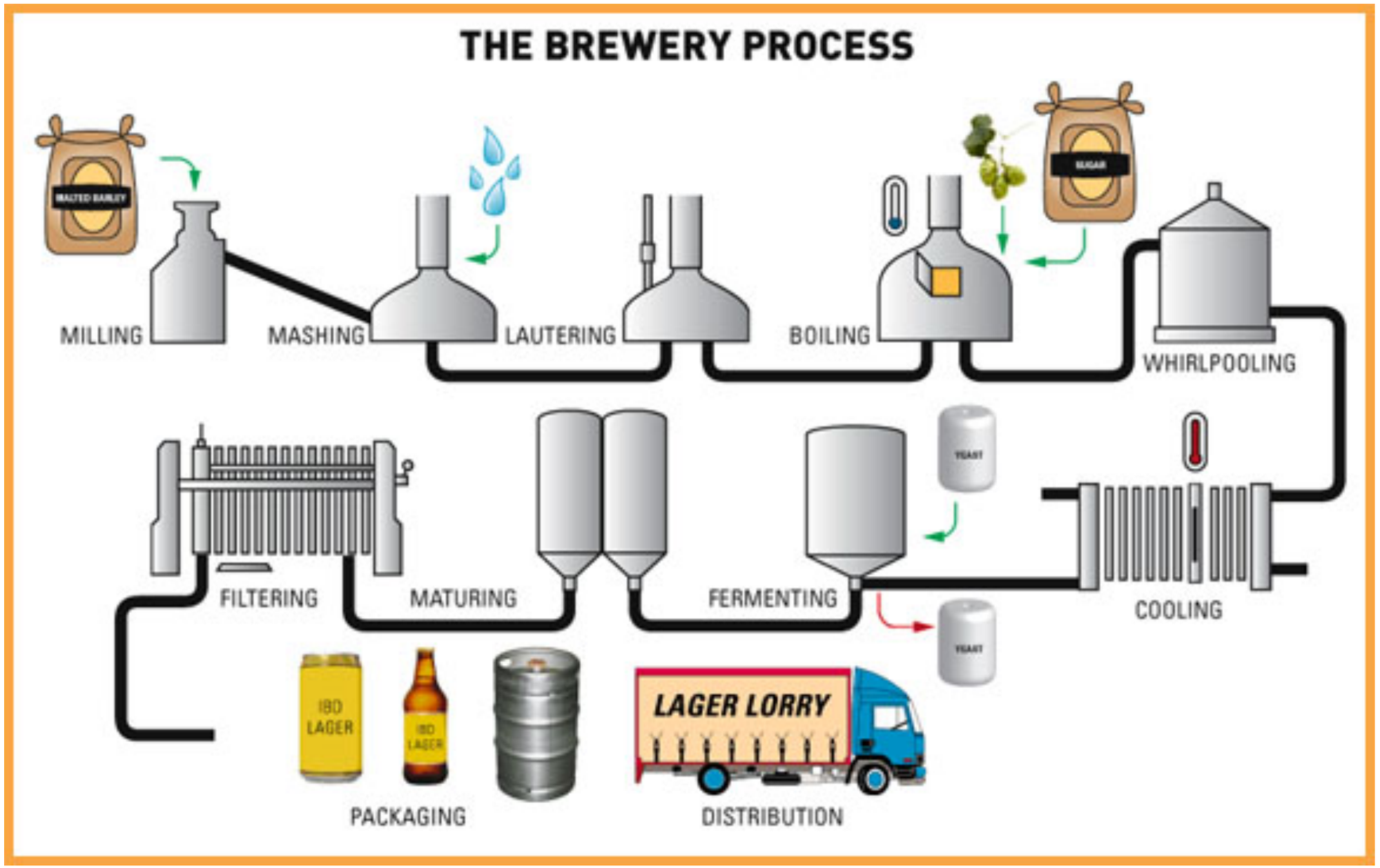
Observable filter(function){}



Operator chaining: map and filter



RX: the data moves across your algorithm



Operator

Operator

Operator

Operator

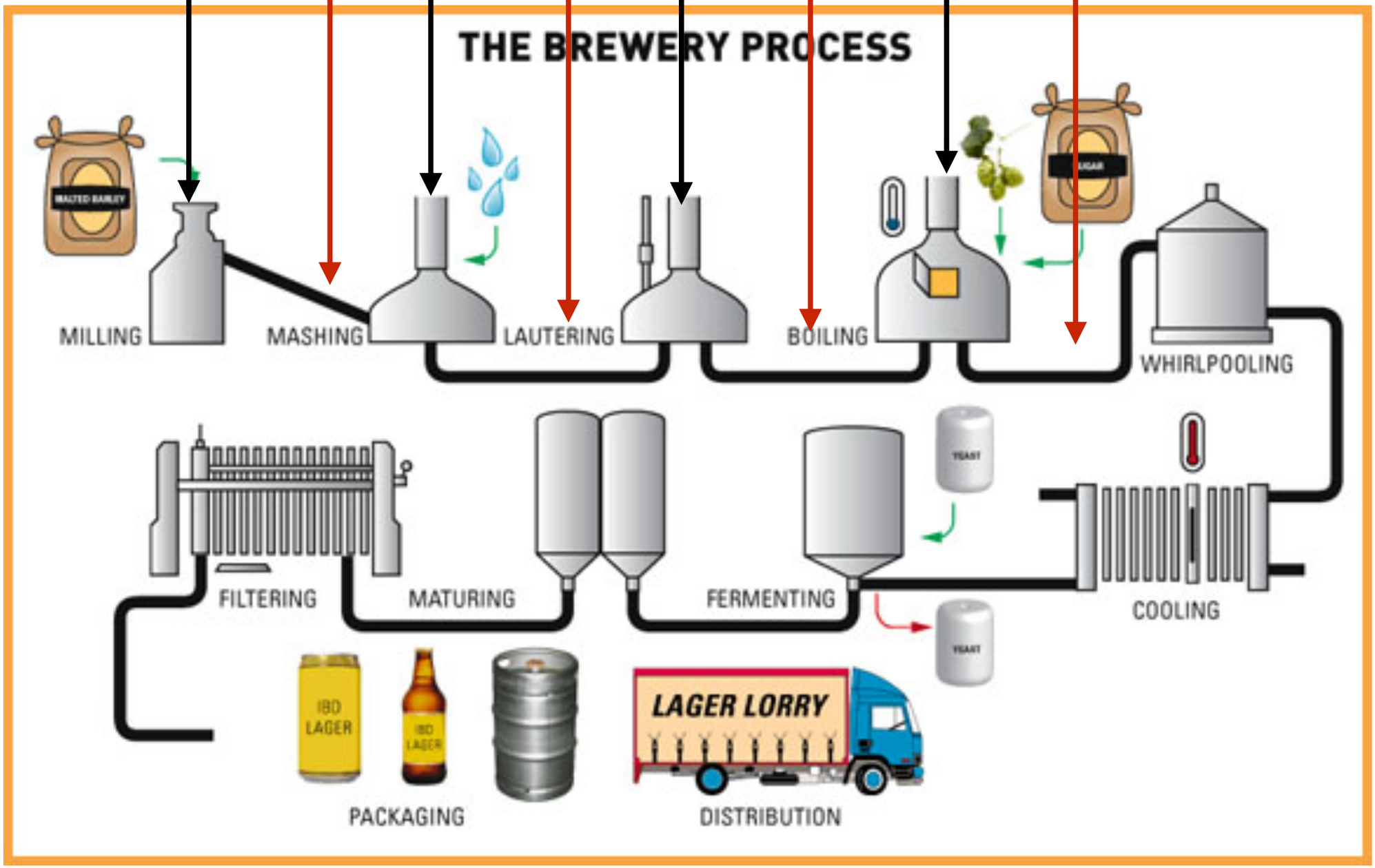
Observable

Observable

Observable

Observable

Observable



Creating an Observable

- Observable.create() - returns Observable that can invoke methods on Observer
- Observable.from() - converts an array or Iterable into Observable
- Observable.fromEvent() - converts events into Observable
- Observable.fromPromise() - converts a Promise into Observable
- Observable.range() - returns a sequence of integers in the specified range
- Observable.just() - converts up to 10 items into Observable

Demo

observable_create

A side-effect operator do()

- **A function with side effects** returns a value and affects the environment

- Examples: I/O, changing state, logging

- **do()** performs a side effect action on each emission

- Doesn’t affect the Observable’s items

- Returns an Observable identical to the source

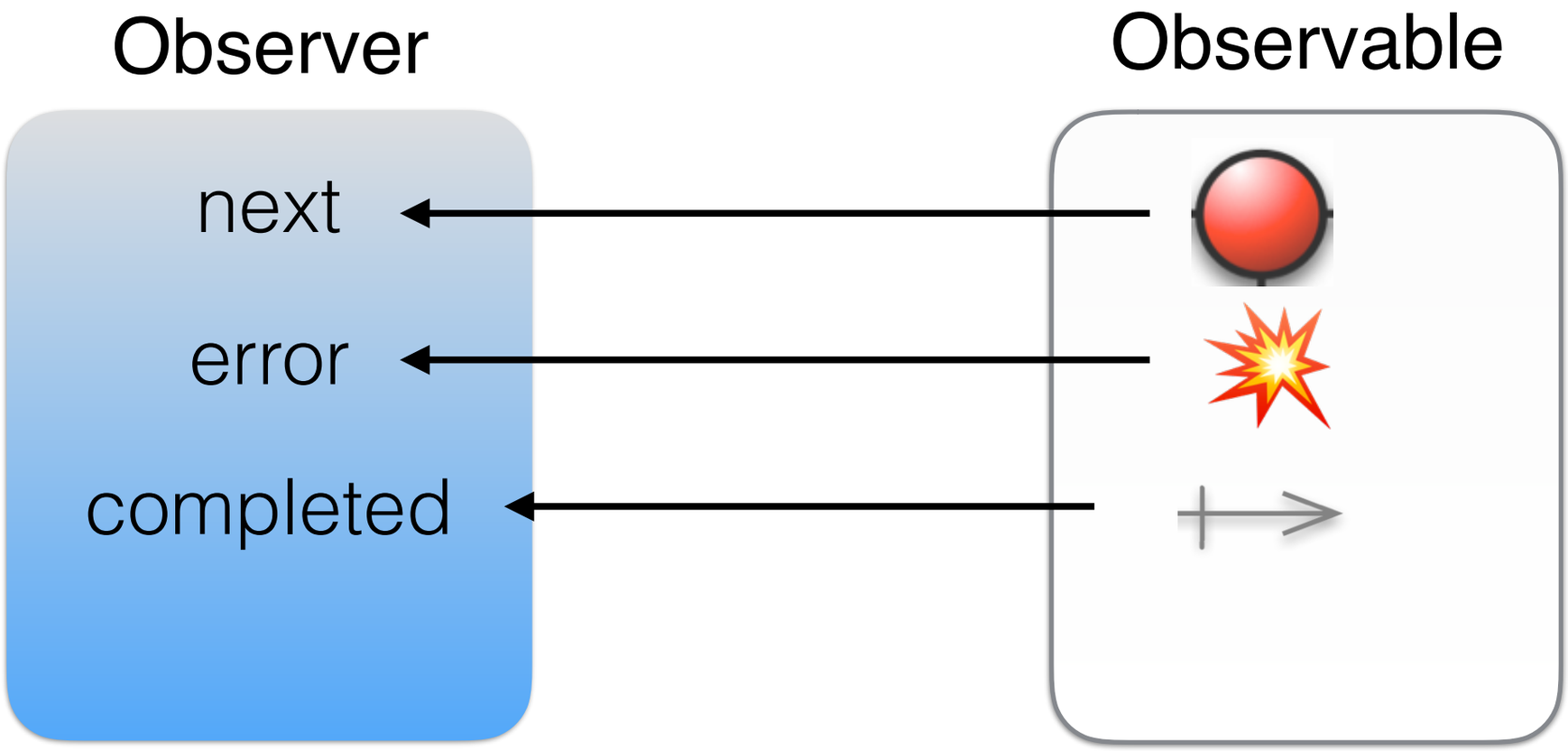
Applying do()

```
Rx.Observable.from(beers)
    .do(beer => console.log("Before filter " + beer.name + " price: " + beer.price))
    .filter(beer => beer.price < 8)
    .do(beer => console.log("After filter " + beer.name + " price: " + beer.price))
    .map(beer => beer.name + ": $" + beer.price)
    .do(beer => console.log("  After map " + beer))
    .subscribe(
        beer => console.log(beer),
        err => console.error(err),
        () => console.log("The sream is over")
);
```

Look ma, no map!

```
Before filter Stella price: 9.5
Before filter Sam Adams price: 8.5
Before filter Bud Light price: 6.5
After filter Bud Light price: 6.5
  After map Bud Light: $6.5
Bud Light: $6.5
Before filter Brooklyn Lager price: 8
Before filter Sapporo price: 7.5
After filter Sapporo price: 7.5
  After map Sapporo: $7.5
Sapporo: $7.5
The sream is over
```

Error Handling



Error-handling operators

- Observer.error() kills the subscription

- retry(n) - Immediately resubscribe (up to n times)

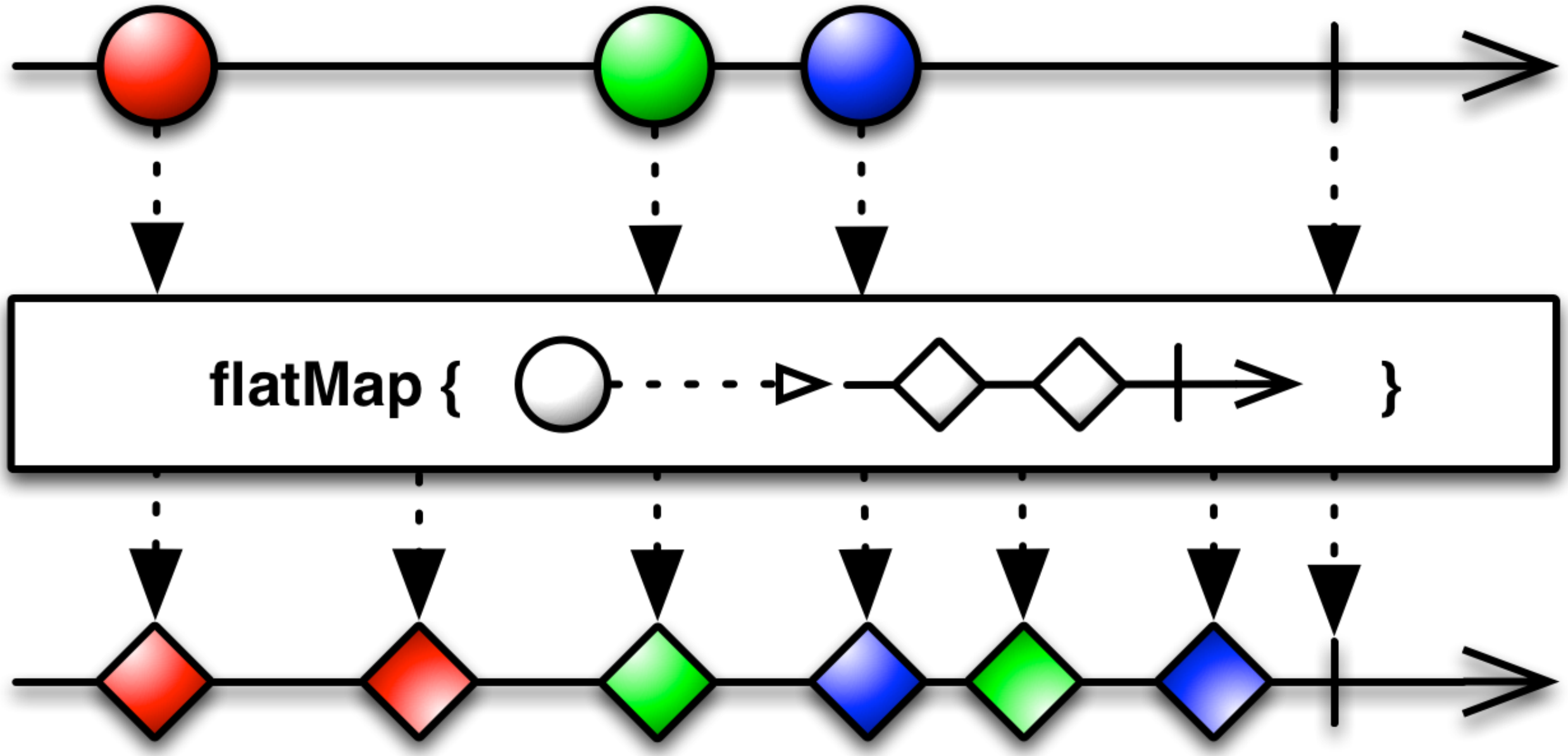
- retryWhen() - intercept, analyze the error, resubscribe
  - use Observable.timer() to delay resubscription

- catch() - used for failover to another Observable

Demo

observable_error
observable error failover

The flatMap() operator



Observable.flatMap()

Producer

```
function getDrinks() {

    let beers = Rx.Observable.from([
        {name: "Stella", country: "Belgium", price: 9.50},
        {name: "Sam Adams", country: "USA", price: 8.50},
        {name: "Bud Light", country: "USA", price: 6.50}
    ]);

    let softDrinks = Rx.Observable.from([
        {name: "Coca Cola", country: "USA", price: 1.50},
        {name: "Fanta", country: "USA", price: 1.50},
        {name: "Lemonade", country: "France", price: 2.50}
    ]);

    return Rx.Observable.create( observer => {
            observer.next(beers);
            observer.next(softDrinks);
            observer.complete();
        }
    );
}
```

Consumer

```
getDrinks()
    .flatMap(drinks => drinks)
    .subscribe(
        drink => console.log("Subscriber got " + drink.name + ": " + drink.price ),
        error => console.err(error),
        () => console.log("The stream of observables is over")
     );
```

Demo

drinks_flat

Observables in Angular

Observables in Angular

- Angular comes with RxJS 5
- Use cases for observable streams:

  - UI events, form controls
  - EventEmitter
  - Handling HTTP responses
  - Data pushed via websockets
  - Your own app functions

Observable Events in Angular

```
@Component({
    selector: "app",
    template: `
      <h2>Observable events demo</h2>
      <input type="text" placeholder="Enter stock" [ngFormControl]="searchInput">
    `
})
class AppComponent {

    searchInput: Control;

    constructor(){
        this.searchInput = new Control('');

        this.searchInput.valueChanges
            .debounceTime(500)
            .subscribe(stock => this.getStockQuoteFromServer(stock));
    }

    getStockQuoteFromServer(stock) {

        console.log(`The price of ${stock} is ${100*Math.random().toFixed(4)}`);
    }
}
```

Observable

Demo

observable_events

Subscribing to EventEmitter

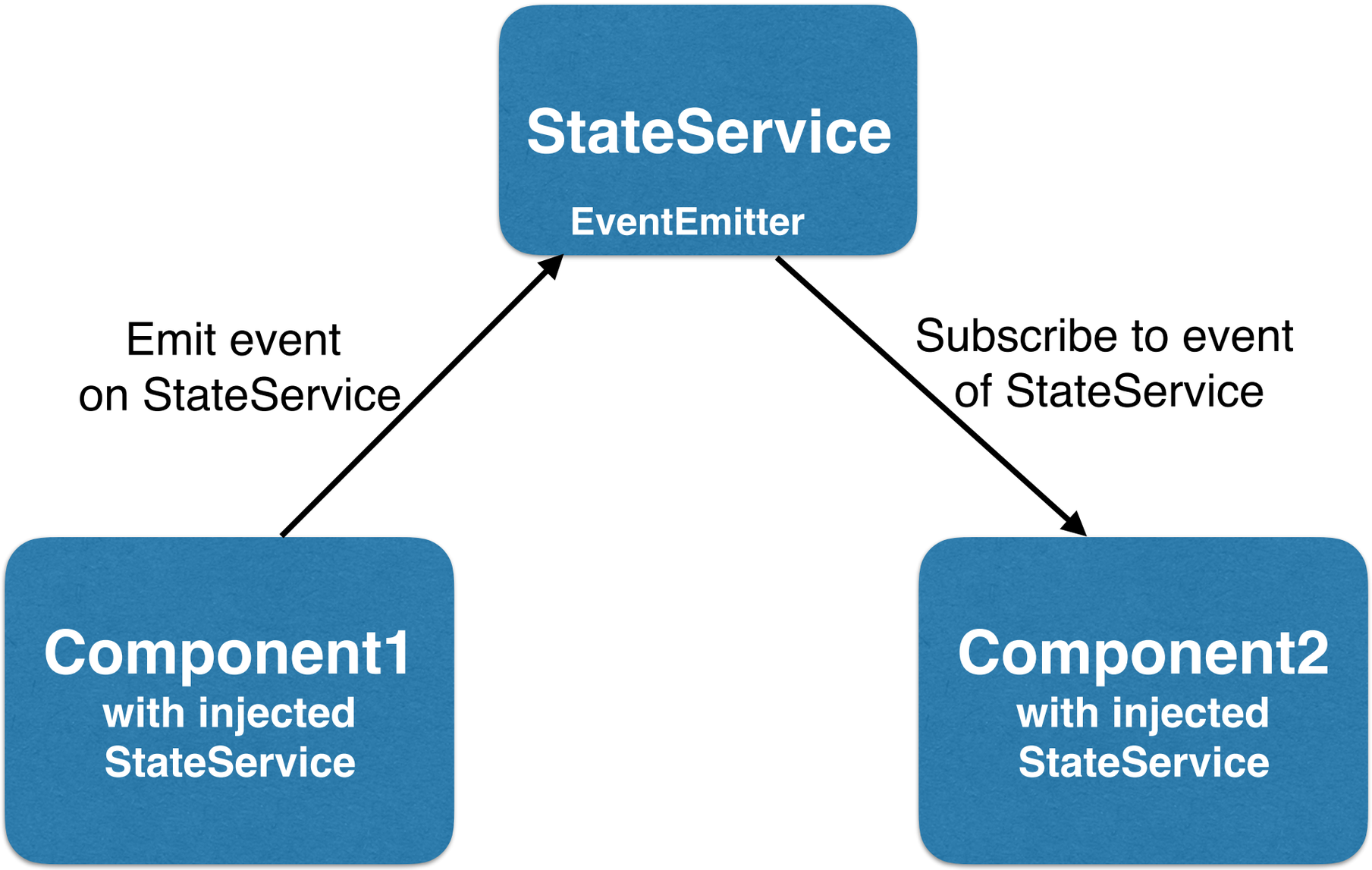
Angular:

```
export declare class EventEmitter<T> extends Subject<T> {}
```

Your app:

```
myEvent: EventEmitter<string> = new EventEmitter();

myEvent.emit(“Hello World”);

…

myEvent.subscribe(event => console.log(”Received " + event);
```

Mediator, DI, Events, and Observables



An alternative impl of Mediator: http://bit.ly/1ZMqa6q

Demo

1.systemjsconfig: main_search.ts

2.npm start

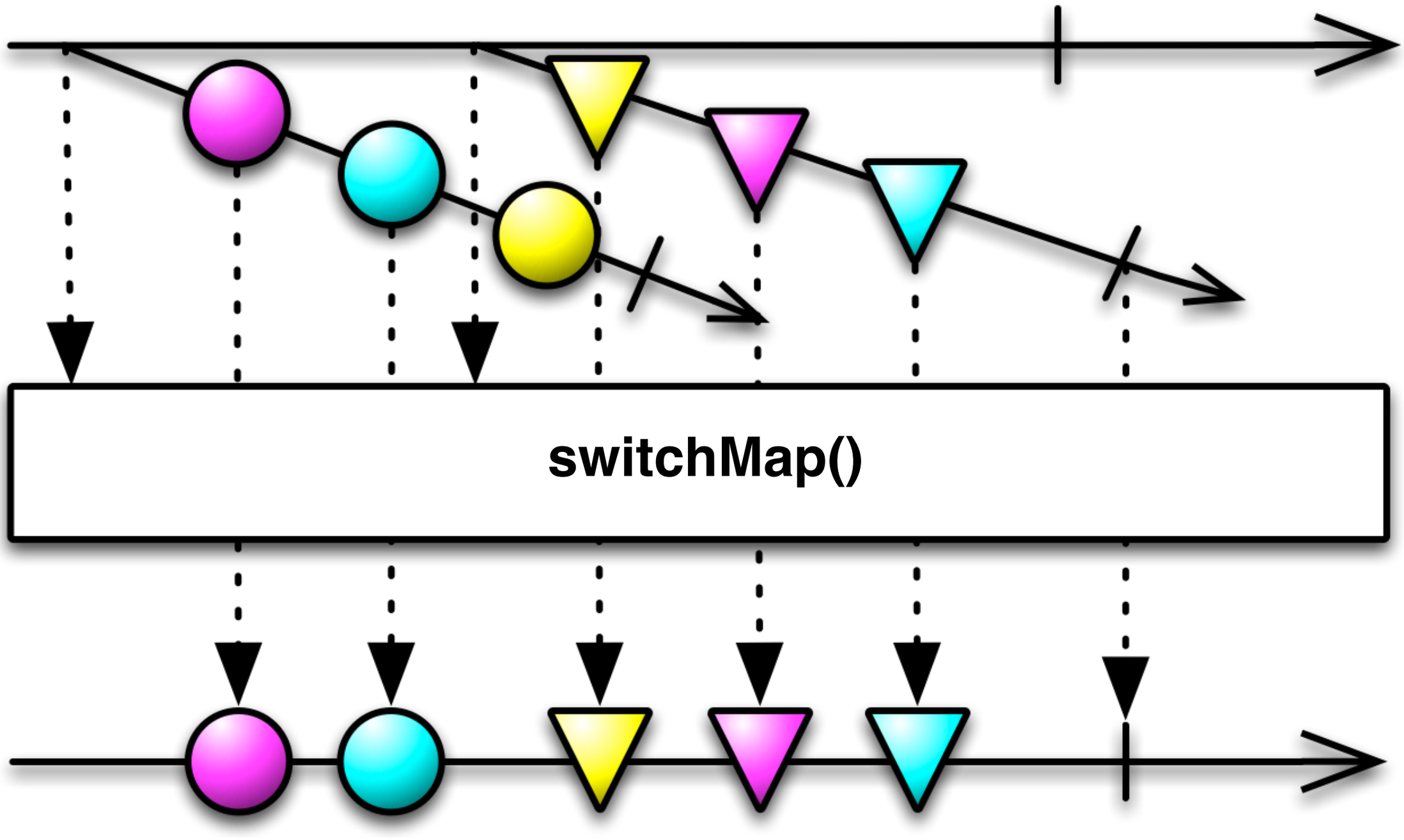
Http and Observables

```
class AppComponent {

  products: Array<string> = [];

  constructor(private http: Http) {

    this.http.get('http://localhost:8080/products')
        .map(res => res.json())
        .subscribe(
          data => {
            this.products=data;
        },
          err =>
            console.log("Can't get products. Error code: %s, URL: %s ",
                                                  err.status, err.url),
          () => console.log('Product(s) are retrieved')
        );
  }
}
```

DI

Observer

The switchMap() operator



Switching over from one observable to another

Canceling Http with switchMap

Observable 1

Observable 2

```
this.searchInput.valueChanges
        .debounceTime(200)
        .switchMap(city => this.getWeather(city))
        .subscribe(
            res => {
              if (res['cod'] === '404') return;
              if (!res.list[0]) {
                 this.temperature ='City is not found';
              } else {
                this.temperature =
                    `Current temperature is  ${res.list[0].main.temp}F, ` +
                    `humidity: ${res.list[0].main.humidity}%`;
              }
            },
            err => console.log(`Can't get weather. Error code: %s, URL: %s`, err.message, err.url),
            () => console.log(`Weather is retrieved`)
        );
}

getWeather(city): Observable<Array> {
  return this.http.get(this.baseWeatherURL + city + this.urlSuffix)
    .map(res => res.json());
}
```

Demo

observable_events_http

AsyncPipe

- AsyncPipe subscribes and unwraps the Promise or Observable
- In templates use the pipe symbol with **async**
- Can’t handle errors

async in a template

```
@Component({
  selector: 'http-client',
  template: `<h1>All Products</h1>
  <ul>
    <li *ngFor="let product of products | async">
       {{product.title}}
    </li>

  </ul>
  `})
class AppComponent {

  products: Observable<Array<String>>;

  constructor(private http: Http) {

   this.products = this.http.get('/products')
        .map(res => res.json());
  }
}
```

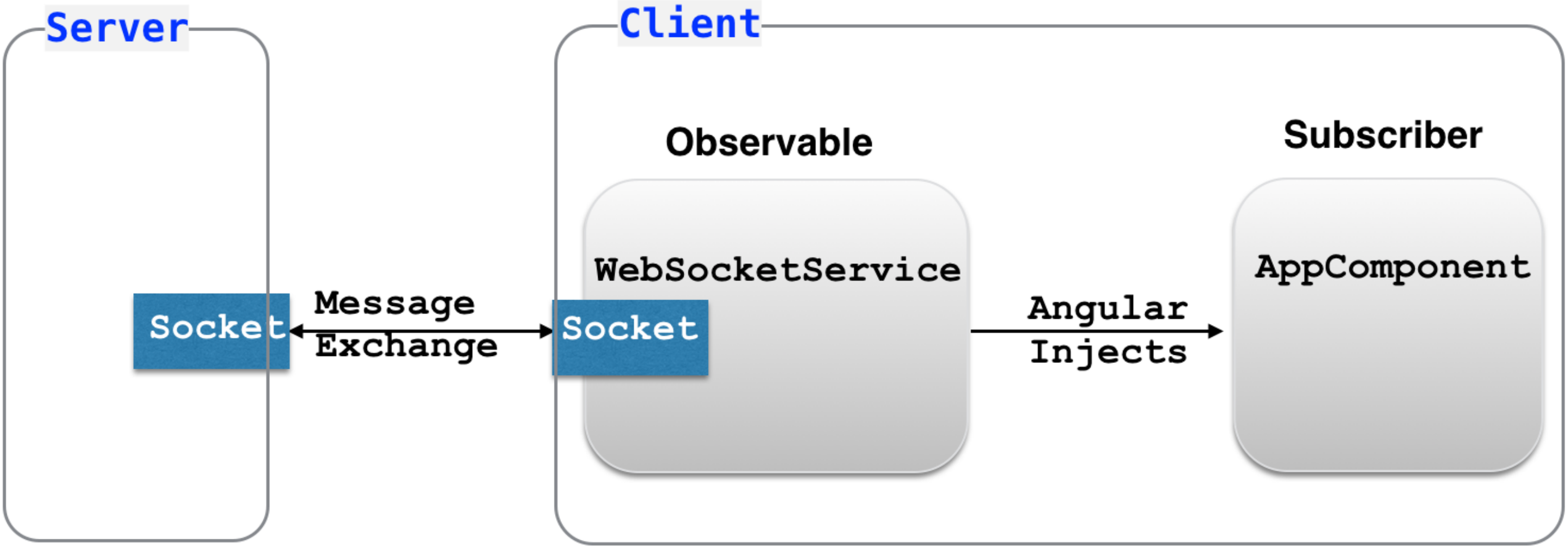
Demo

1.systemjs.config: main_asyncpipe.ts

2.npm run restserver

3.http://localhost:8000

WebSocket and Observables



Wrapping WebSocket into Observable

```
import {Observable } from 'rxjs/Rx';

export class BidService{

    ws: WebSocket;

    createObservableSocket(url:string): Observable{

        this.ws = new WebSocket(url);

        return new Observable(
                        observer => {

                          this.ws.onmessage = (event) => observer.next(event.data);

                          this.ws.onerror   = (event) => observer.error(event);

                           this.ws.onclose  = (event) => observer.complete();
                        });
    }

}
```

Subscribing to Websocket's messages

```
@Component({ … })
class BidComponent {
  newBid: Bid;

  constructor(private wsService: BidService) {

      this.wsService.createObservableSocket("ws://localhost:8085")
        .map(res => JSON.parse(res))
        .subscribe(
            data => {

              this.newBid = data;
              this.newBid.bidTime= Date.parse(data.bidTime);
                console.log(this.newBid);
            },
            err => console.log( err),
            () =>  console.log( 'The bid stream is complete')
        );
  }

}

bootstrap(BidComponent);
```

Demo

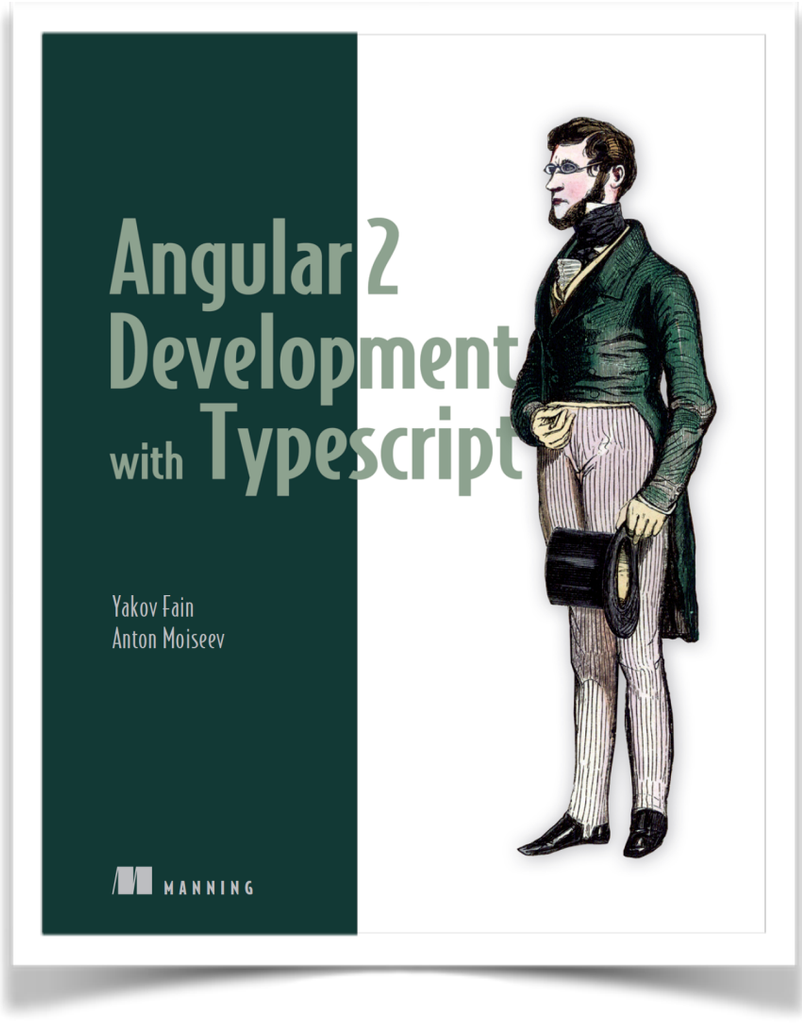
1.systemjs.config: bids/bid_component.ts

2.npm run bidserver

3.http://localhost:8000

Thank you!

- Code samples:
  https://github.com/yfain/observables
- Our company: faratasystems.com
- Blog: yakovfain.com
- Twitter:@yfain



discount code: faindz