Строим крипто-трейдинг платформу

Олег Докука

Обо мне

Обо мне

- Инженер в компании Levi9

Обо мне

- Инженер в компании Levi9
- Активный докладчик

Обо мне

- Инженер в компании Levi9
- Активный докладчик
- Контрибьютор Reactor 3

Обо мне

- Инженер в компании Levi9
- Активный докладчик
- Контрибьютор Reactor 3

/OlehDokuka /oleh.dokuka /OlegDokuka

Oleh Dokuka, Igor Lozynskyi

Reactive Programming in Spring 5.0

Build Reactive Systems with Spring and Reactor

Packt>

Что будет?

А будет

А будет

- Как построить крипто-трейдинг платформу

А будет

- Как построить крипто-трейдинг платформу
- Spring +

А будет

- Как построить крипто-трейдинг платформу
- Spring +
- Разбор подходов и особенностей с Reactor 3

А будет

- Как построить крипто-трейдинг платформу
- Spring +
- **Разбор подходов и особенностей с Reactor 3**

Чего НЕ будет?

А не будет

А не будет

- Blockchain
- Kotlin
- Тестирования

Какие требования?

Функциональные

Функциональные

- RealTime стоимость Битка

Функциональные

- RealTime стоимость Битка
- Графики, лента продаж и прочее

Функциональные

- RealTime стоимость Битка
- Графики, лента продаж и прочее
- Пользовательский кошелек

Функциональные

- RealTime стоимость Битка
- Графики, лента продаж и прочее
- Пользовательский кошелек
- Торговля биткоином

Нефункциональные

Нефункциональные

- Высокая пропускная способность и быстрота ответа

Нефункциональные

- Высокая пропускная способность и быстрота ответа
- Эффективная утилизация железа

Нефункциональные

- Высокая пропускная способность и быстрота ответа
- Эффективная утилизация железа
- Отказоустойчивость

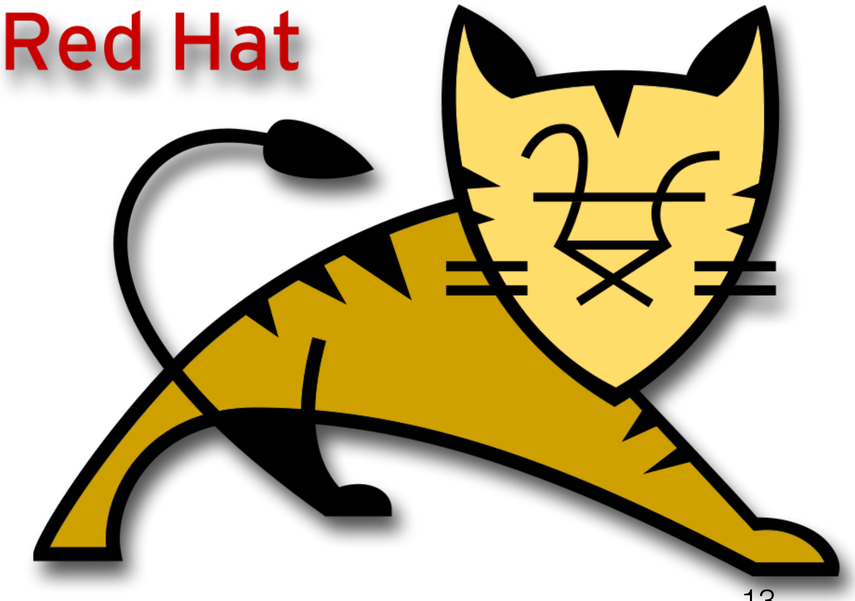
Нефункциональные

- Высокая пропускная способность и быстрота ответа
- Эффективная утилизация железа
- Отказоустойчивость
- Java + Spring Stack

Какой сервер?


jetty://


JBoss®
by Red Hat




undertow


Netty


Netty

Асинхронный, event-driven фреймворк


Netty

- Асинхронные, Не блокирующие операции

- Асинхронные, Не блокирующие операции
- Event-Loop


Netty

- Высокая пропускная способность

- Высокая пропускная способность
- Эффективная утилизация ресурсов

1 Netty ≈ 3 Tomcat


Netty


VERT.X


play


Netty


Ratpack
ratpack.io


VERT.X


play


Netty

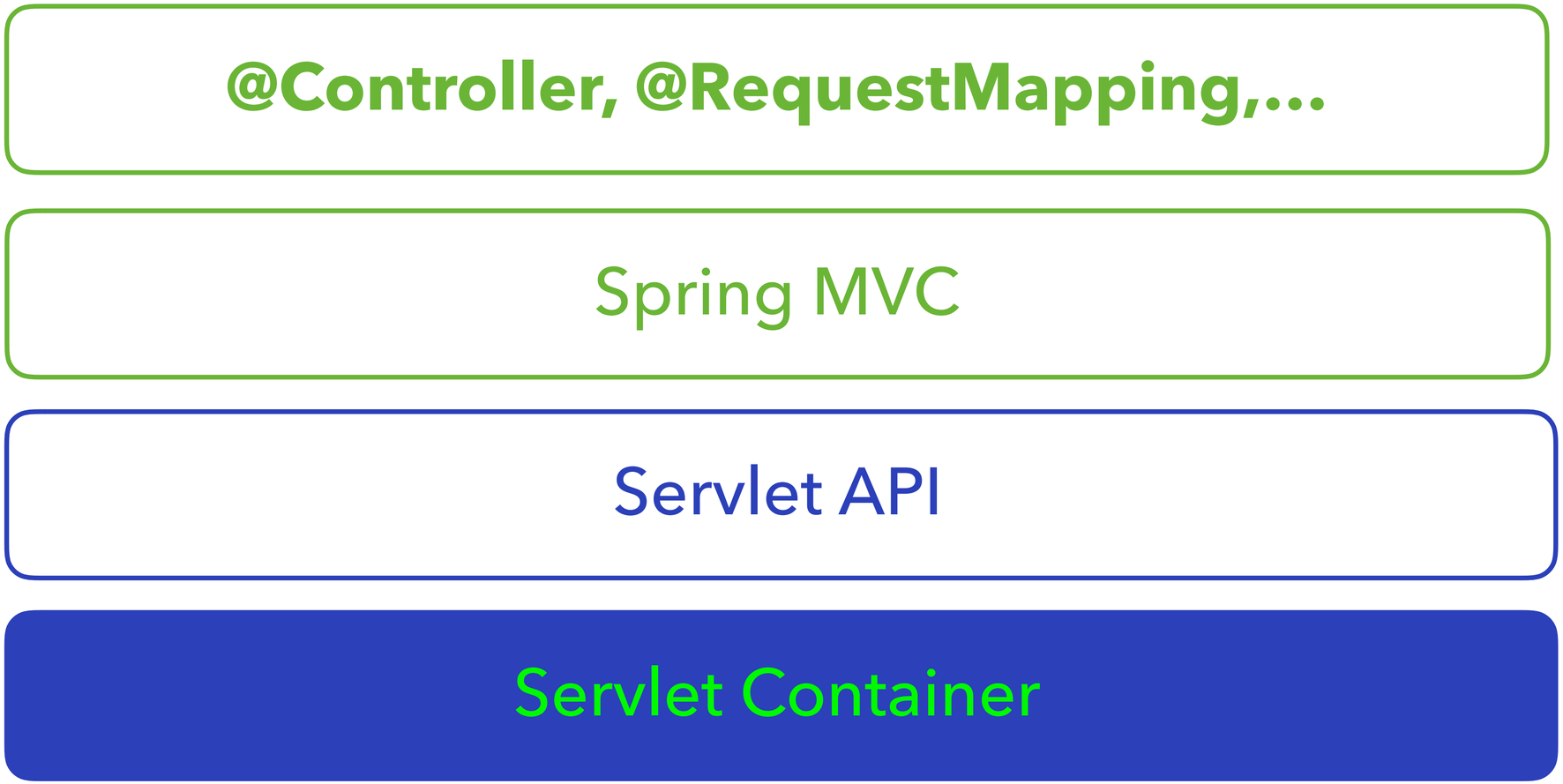

Ratpack
ratpack.io


spring
by Pivotal™

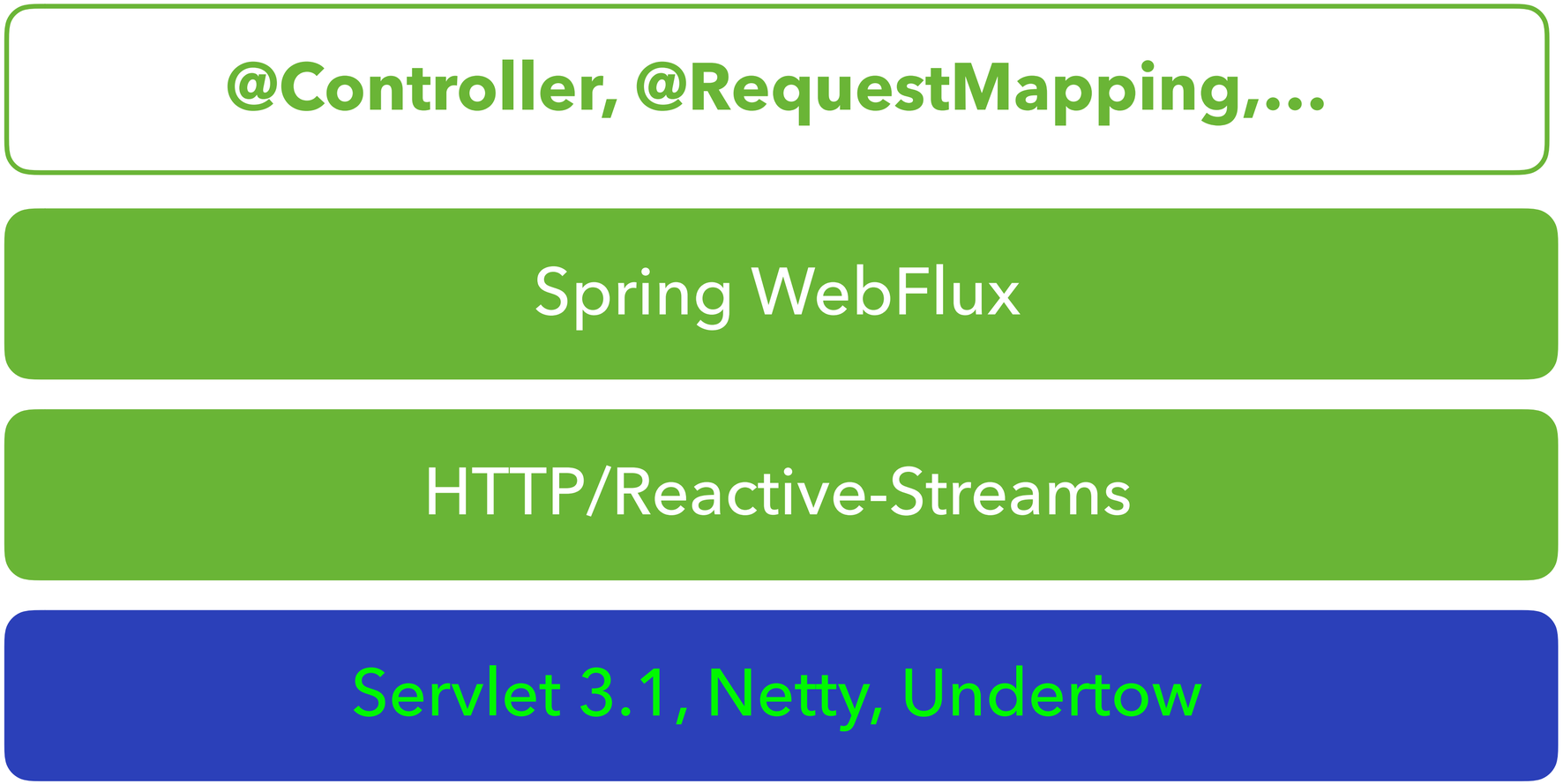
WebFlux

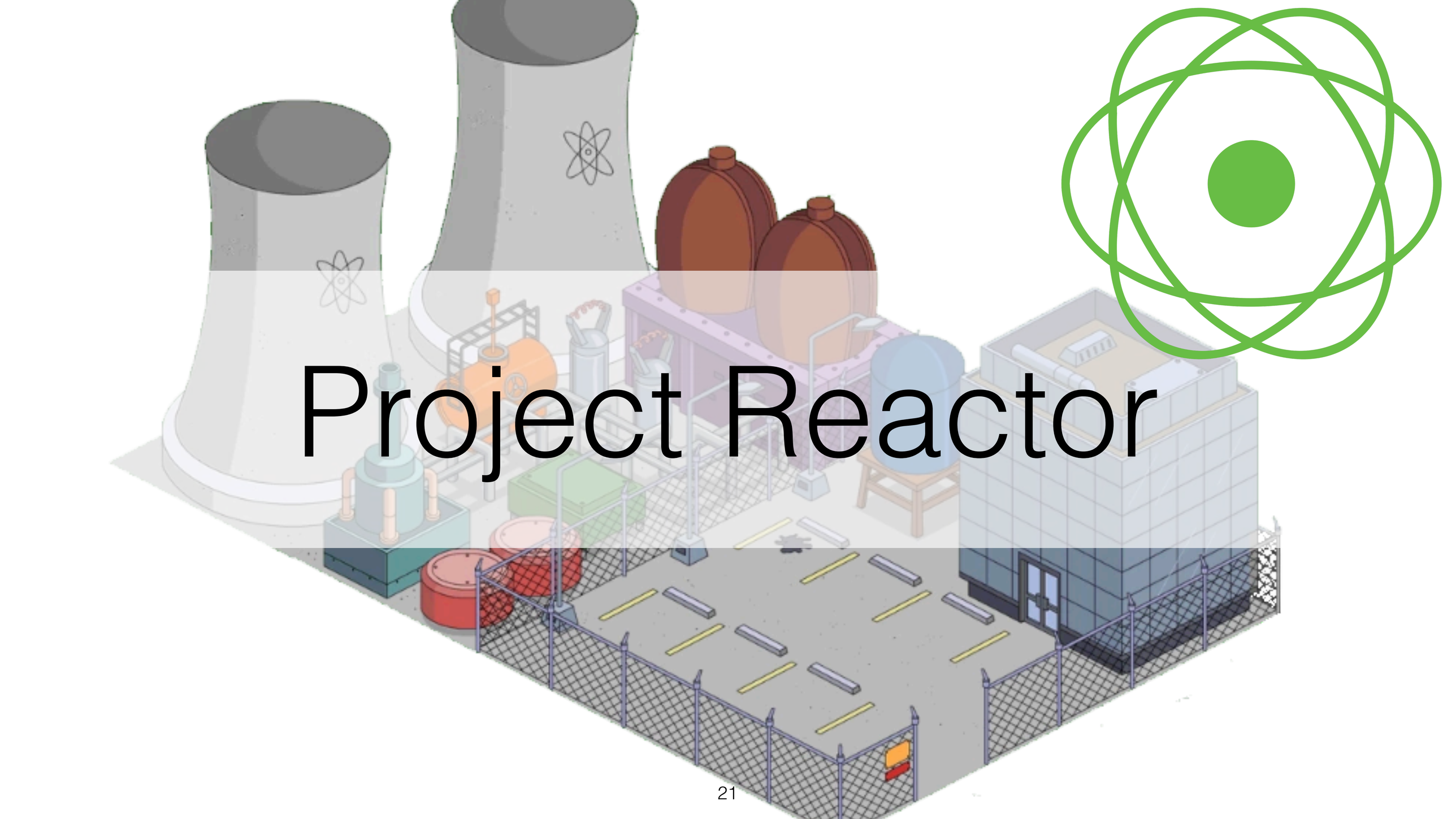
WebMVC на реактивных стероидах

Default Stack (WebMVC)



Reactive Stack (WebFlux)





Project Reactor

```
Flux.just(1, 2, 3, 4)
        .map(mapFunction())
        .filter(filterFunction())
        .map(mapFunction())
        .doOnNext(someAction())
        .subscribe()
```

Reactive Types

Reactive Types

- `Mono<T>`

Reactive Types

- `Mono<T>`

Reactive Types

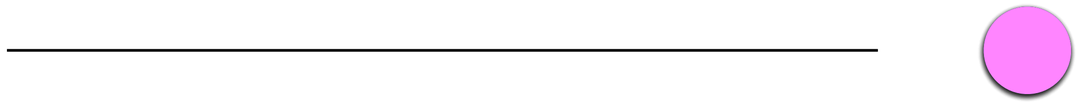
- `Mono<T>`

Reactive Types

- `Mono<T>`
- `Flux<T>`

Reactive Types

- `Mono<T>`
- `Flux<T>`



Reactive Types

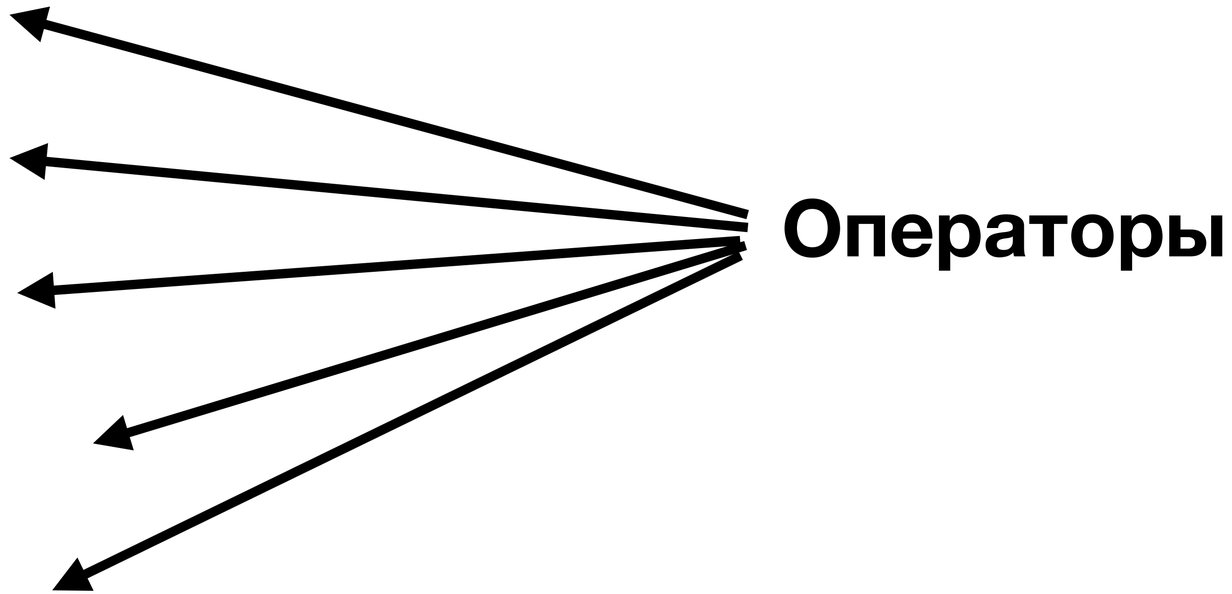
```
abstract class Flux<T>
                implements Publisher<T> {
    ...
}
```

Reactive Types

```
abstract class Flux<T>
                implements Publisher<T> {
   ...
}
```

```
Flux.just(1, 2, 3, 4)
      .map(…)
      .filter(…)
      .map(…)
      .doOnNext(…)
      .subscribe()
```

```
Flux.just(1, 2, 3, 4)
    .map(…)
    .filter(…)
    .map(…)
    .doOnNext(…)
    .subscribe()
```



То есть

То есть

- Netty

То есть

- Netty
- Spring WebFlux

То есть

- Netty
- Spring WebFlux
- Project Reactor 3

То есть

- Netty
- Spring WebFlux
- Project Reactor 3
- Spring Boot 2

Начнем с API

Что нужно?

Что нужно?

- Вернуть HTML `trade.io/`

Что нужно?

- Вернуть HTML `trade.io/`
- WebSocket `trade.io/stream`

Для WebSocket

Для WebSocket

- Реализовать `WebSocketHandler`

```
interface WebSocketHandler {
 Mono<Void> handle(WebSocketSession session);
}
```

```
interface WebSocketHandler {
 Mono<Void> handle(WebSocketSession session);
}
```

```
interface WebSocketHandler {
 Mono<Void> handle(WebSocketSession session);
}
```

```
interface WebSocketSession {
 Flux<WebSocketMessage> receive();
 Mono<Void> send(Publisher<WebSocketMessage> messages);
}
```

```
interface WebSocketSession {
 Flux<WebSocketMessage> receive();
 Mono<Void> send(Publisher<WebSocketMessage> messages);
}
```

```
interface WebSocketSession {
 Flux<WebSocketMessage> receive();
 Mono<Void> send(Publisher<WebSocketMessage> messages);
}
```

```
interface WebSocketSession {
 Flux<WebSocketMessage> receive();
 Mono<Void> send(Publisher<WebSocketMessage> messages);
}
```

```
class WebSocketMessage {
 DataBuffer getPayload()
 String getPayloadAsText()
 WebSocketMessage retain()
 void release()
}
```

```
class WebSocketMessage {
  DataBuffer getPayload()
  String getPayloadAsText()
  WebSocketMessage retain()
  void release()
}
```

```
class WebSocketMessage {
 DataBuffer getPayload()
 String getPayloadAsText()
 WebSocketMessage retain()
 void release()
}
```

Для WebSocket

- Реализовать `WebSocketHandler`
- Добавить конфигурацию

Для WebSocket

- Реализовать `WebSocketHandler`
- Добавить конфигурацию
- Протокол - **JSON**

Что создадим?

Что создадим?

- `IndexController.java`

Что создадим?

- `IndexController.java`
- `CryptoChannel.java`

Что создадим?

- `IndexController.java`
- `CryptoChannel.java`
- `WebSocketConfiguration.java`

Что создадим?

- IndexController.java
- CryptoChannel.java
- WebSocketConfiguration.java
- Message.java

Что создадим?

- IndexController.java
- CryptoChannel.java
- WebSocketConfiguration.java
- Message.java
- WebSocketMessageMapper.java

Что создадим?

- IndexController.java
- **CryptoChannel.java**
- WebSocketConfiguration.java
- Message.java
- WebSocketMessageMapper.java

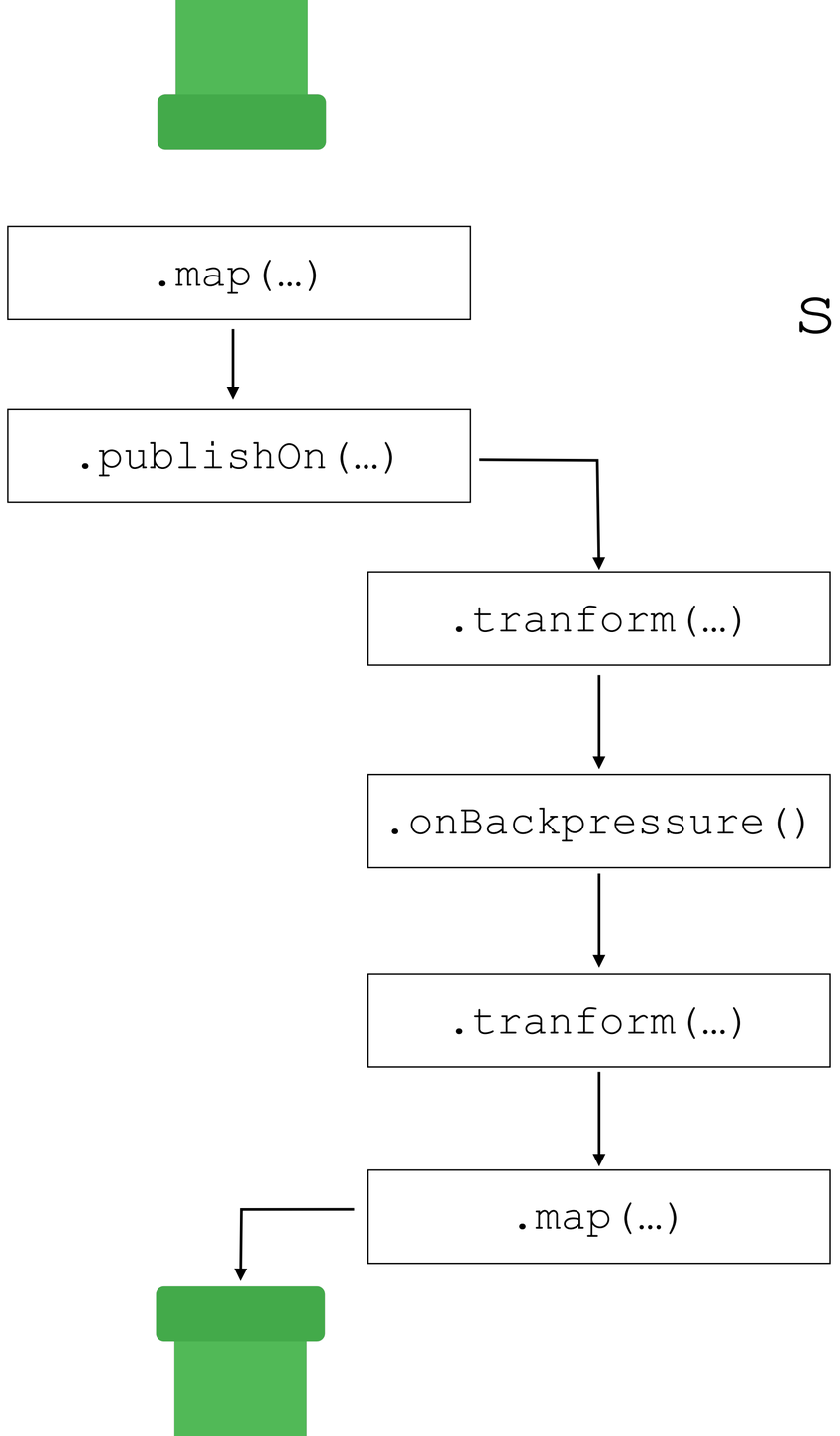
Talk is cheap. Show me the code.

- Linus Torvalds

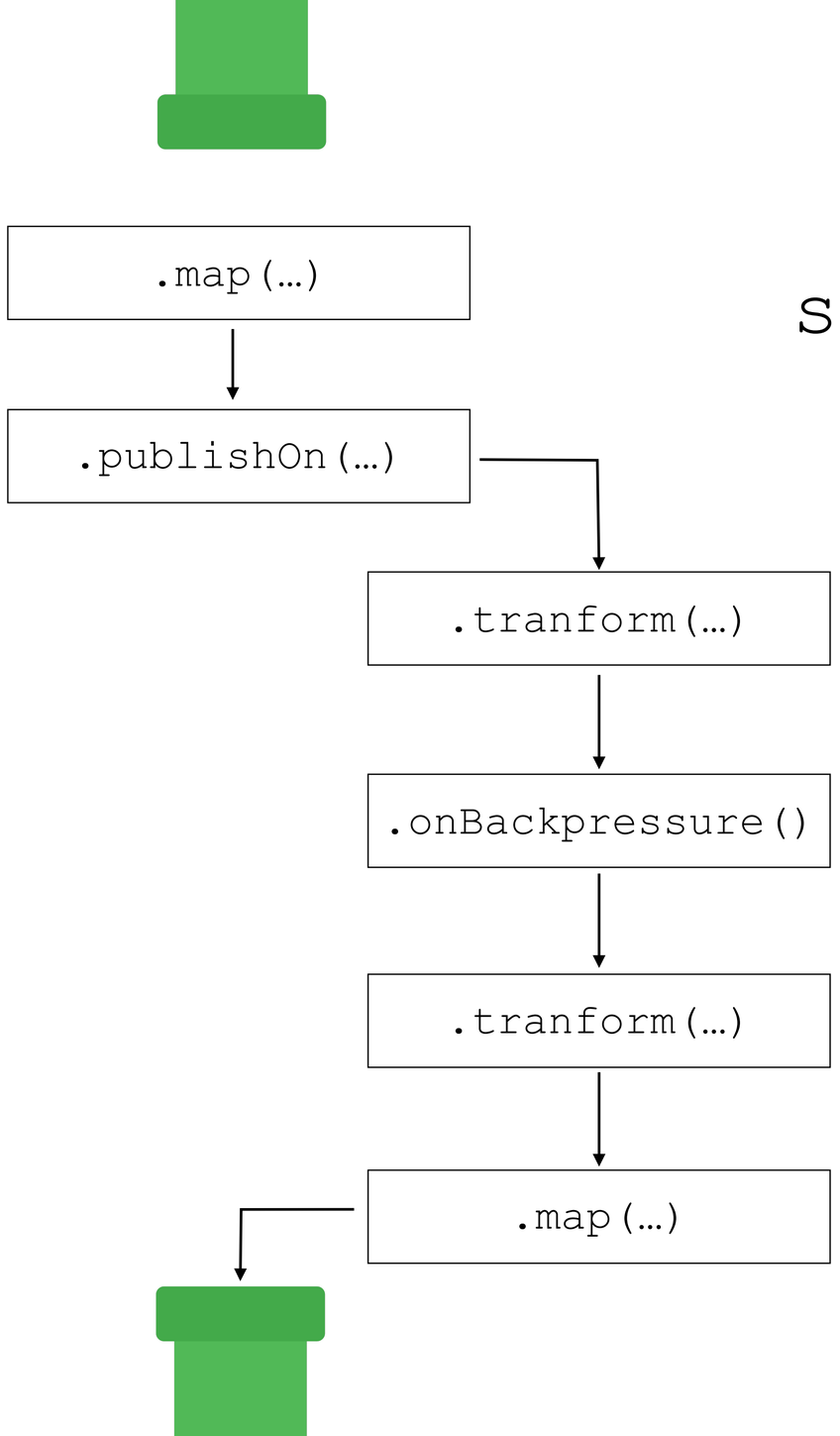
```
s.receive()
  .map(WebSocketMessage::retain)
  .map(WebSocketMessage::getPayload)
  .publishOn(Schedulers.parallel())
  .transform(mapper::decode)
  .transform(this::doHandle)
  .onBackpressureBuffer()
  .transform(s -> mapper.encode(…))
  .map(db -> new WebSocketMessage(…))
  .as(session::send);
```

```
s.receive()
  .map(WebSocketMessage::retain)
  .map(WebSocketMessage::getPayload)
  .publishOn(Schedulers.parallel())
  .transform(mapper::decode)
  .transform(this::doHandle)
  .onBackpressureBuffer()
  .transform(s -> mapper.encode(…))
  .map(db -> new WebSocketMessage(…))
  .as(session::send);
```

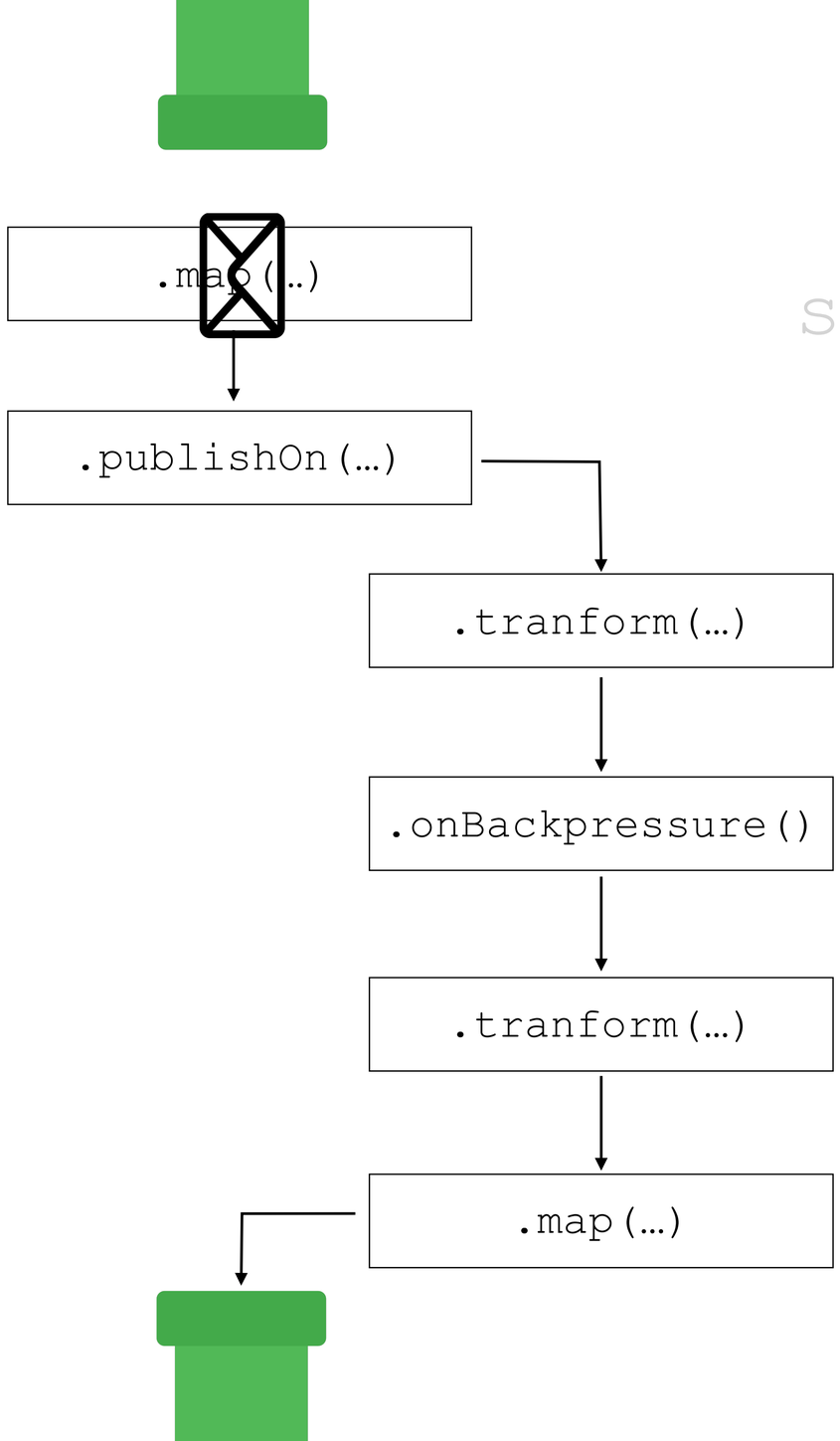
```
s.receive()
  .map(WebSocketMessage::retain)
  .map(WebSocketMessage::getPayload)
  .publishOn(Schedulers.parallel())
  .transform(mapper::decode)
  .transform(this::doHandle)
  .onBackpressureBuffer()
  .transform(s -> mapper.encode(…))
  .map(db -> new WebSocketMessage(…))
  .as(session::send);
```



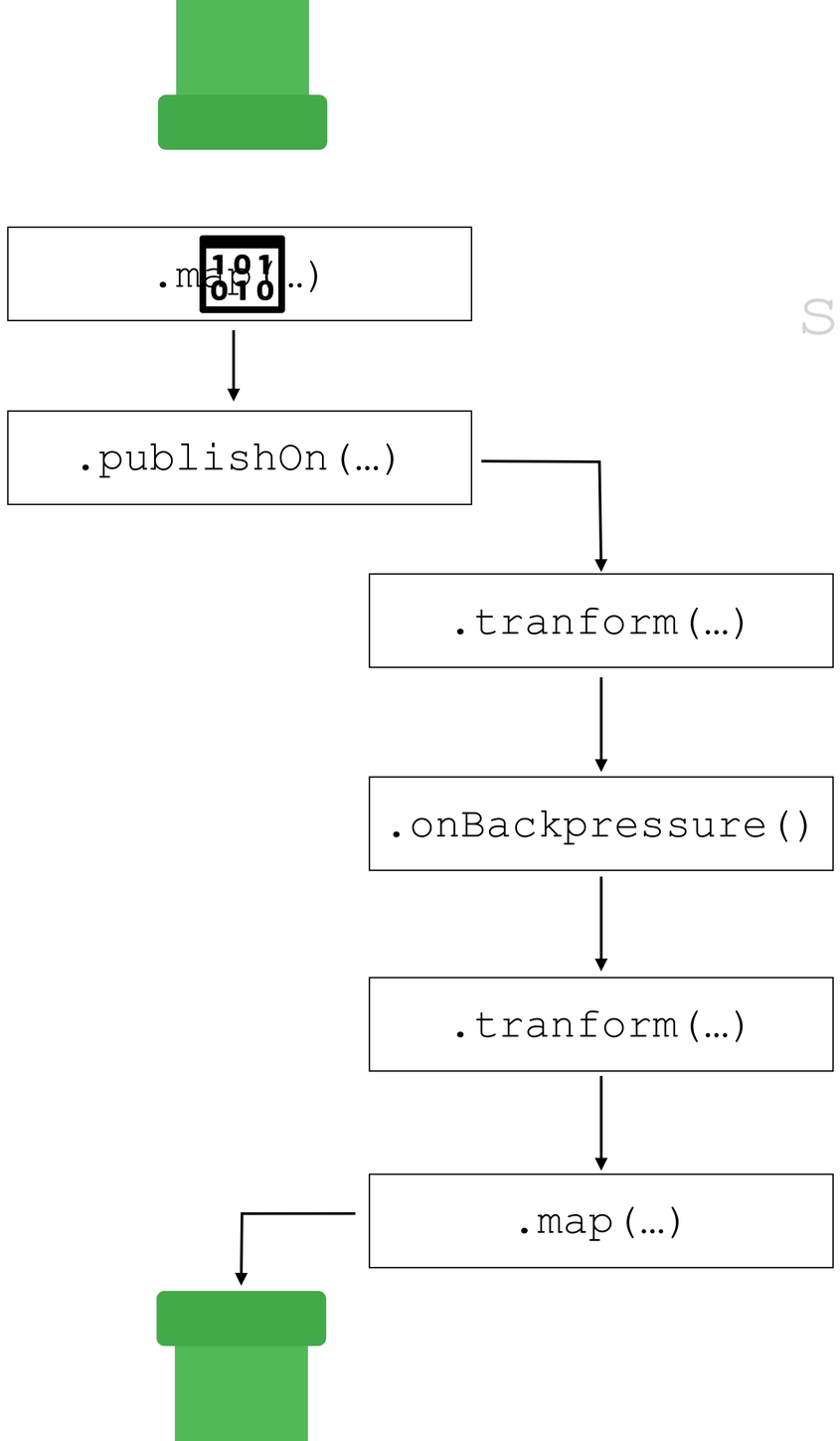

```
s.receive()
 .map(WebSocketMessage::retain)
 .map(WebSocketMessage::getPayload)
 .publishOn(Schedulers.parallel())
 .transform(mapper::decode)
 .transform(this::doHandle)
 .onBackpressureBuffer()
 .transform(s -> mapper.encode(…))
 .map(db -> new WebSocketMessage(…))
 .as(session::send);
```



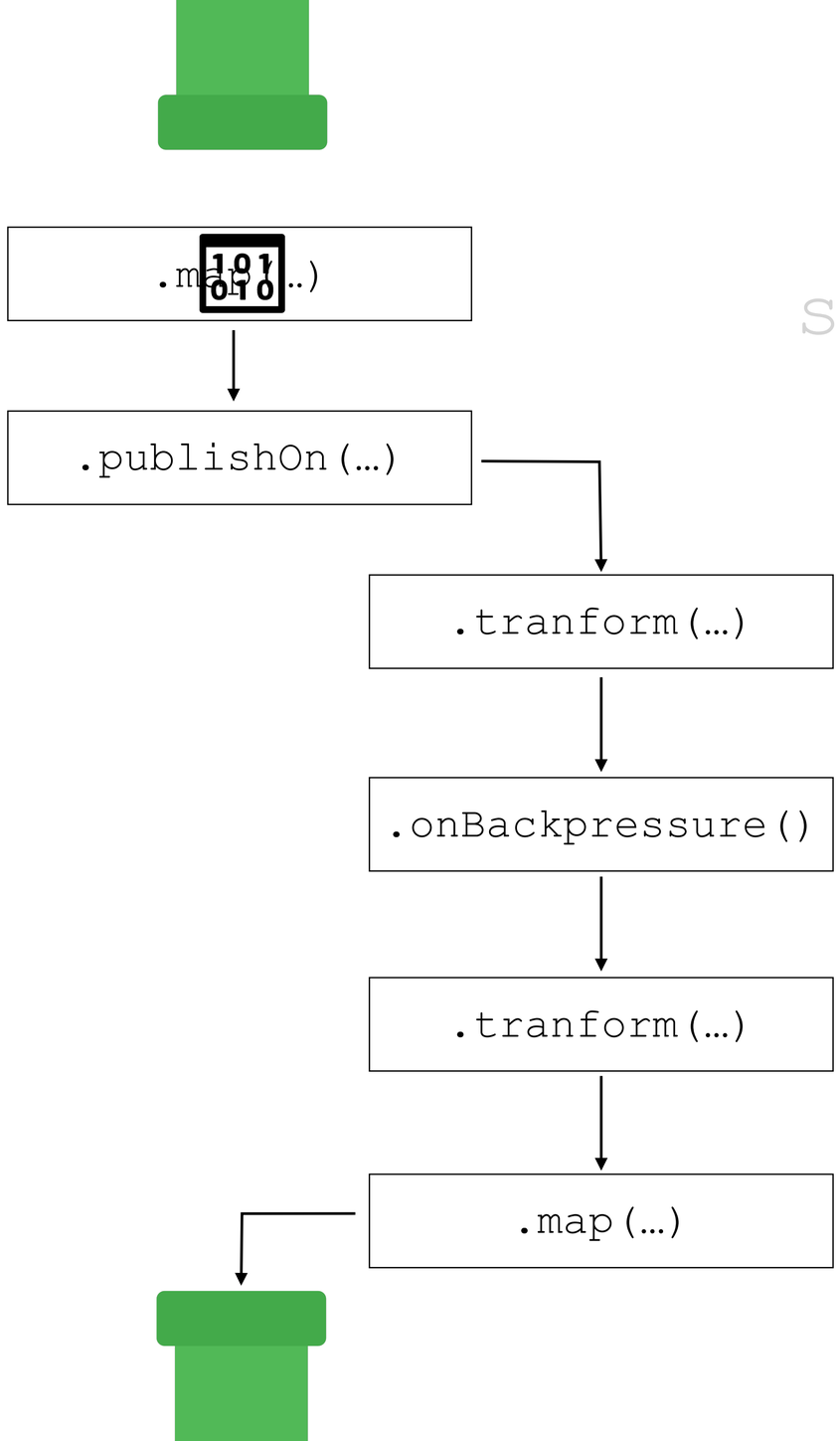

```
s.receive()
 .map(WebSocketMessage::retain)
 .map(WebSocketMessage::getPayload)
 .publishOn(Schedulers.parallel())
 .transform(mapper::decode)
 .transform(this::doHandle)
 .onBackpressureBuffer()
 .transform(s -> mapper.encode(…))
 .map(db -> new WebSocketMessage(…))
 .as(session::send);
```



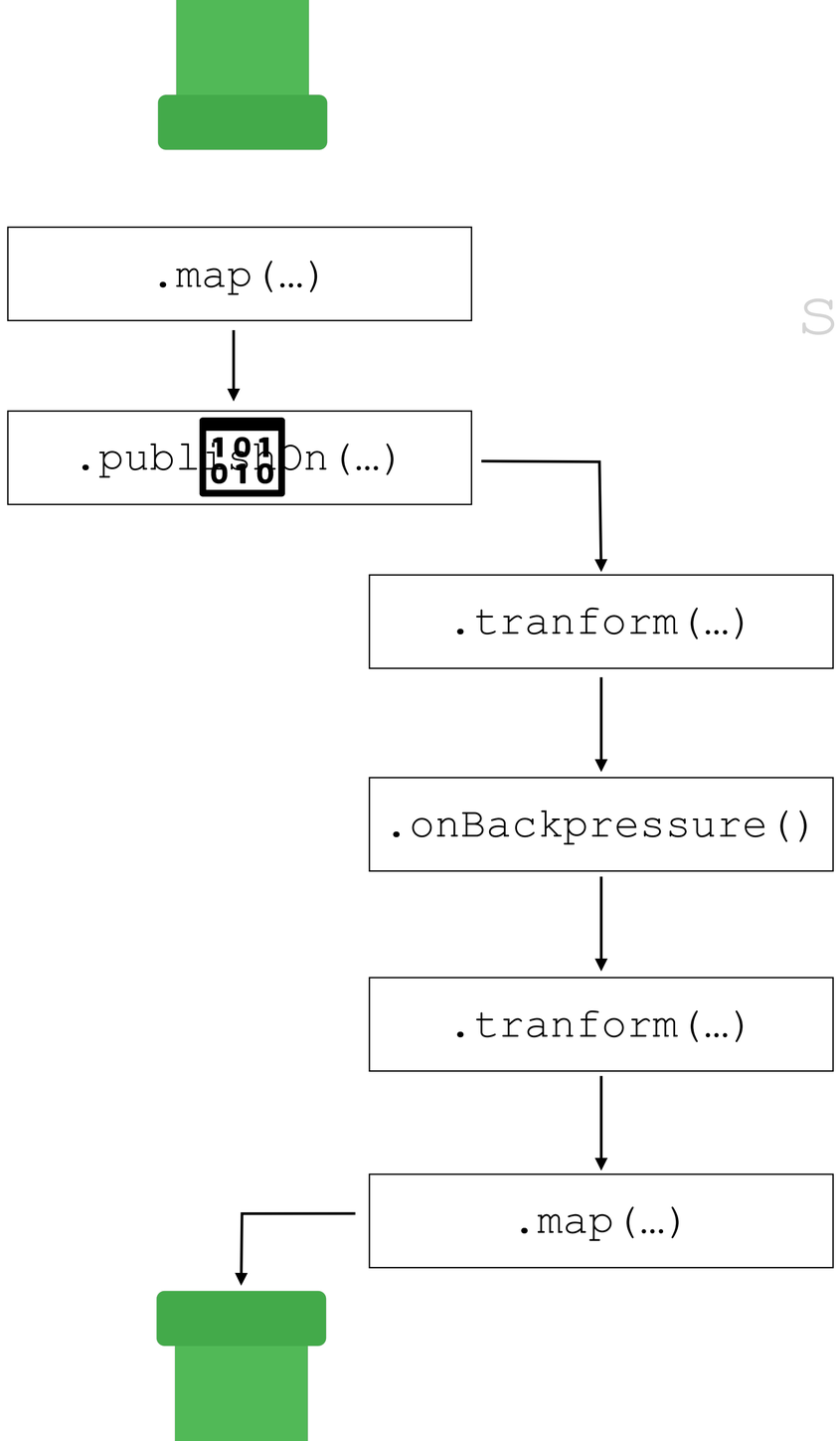

```
s.receive()
 .map(WebSocketMessage::retain)
 .map(WebSocketMessage::getPayload)
 .publishOn(Schedulers.parallel())
 .transform(mapper::decode)
 .transform(this::doHandle)
 .onBackpressureBuffer()
 .transform(s -> mapper.encode(…))
 .map(db -> new WebSocketMessage(…))
 .as(session::send);
```



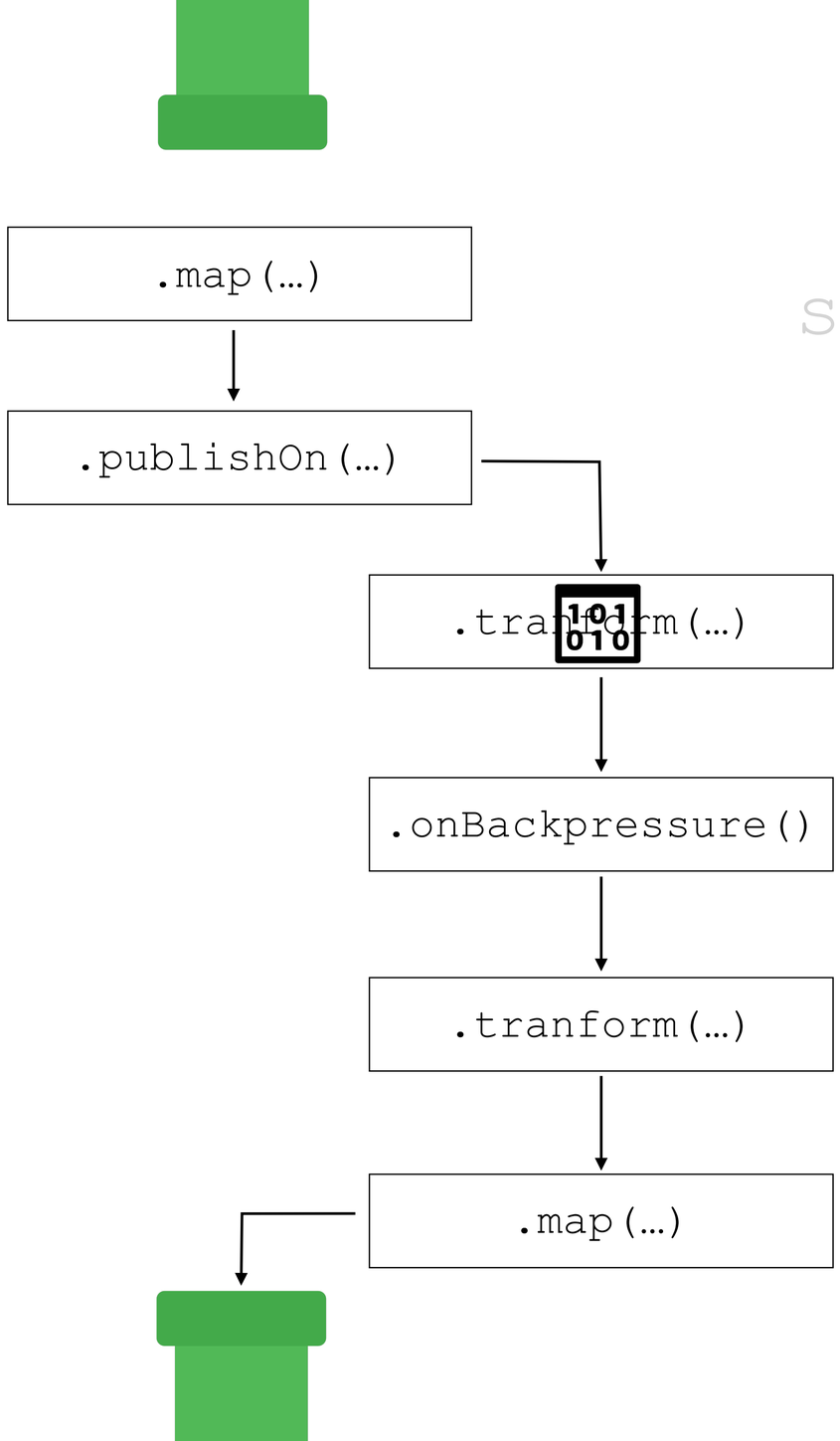

```
s.receive()
 .map(WebSocketMessage::retain)
 .map(WebSocketMessage::getPayload)
 .publishOn(Schedulers.parallel())
 .transform(mapper::decode)
 .transform(this::doHandle)
 .onBackpressureBuffer()
 .transform(s -> mapper.encode(…))
 .map(db -> new WebSocketMessage(…))
 .as(session::send);
```



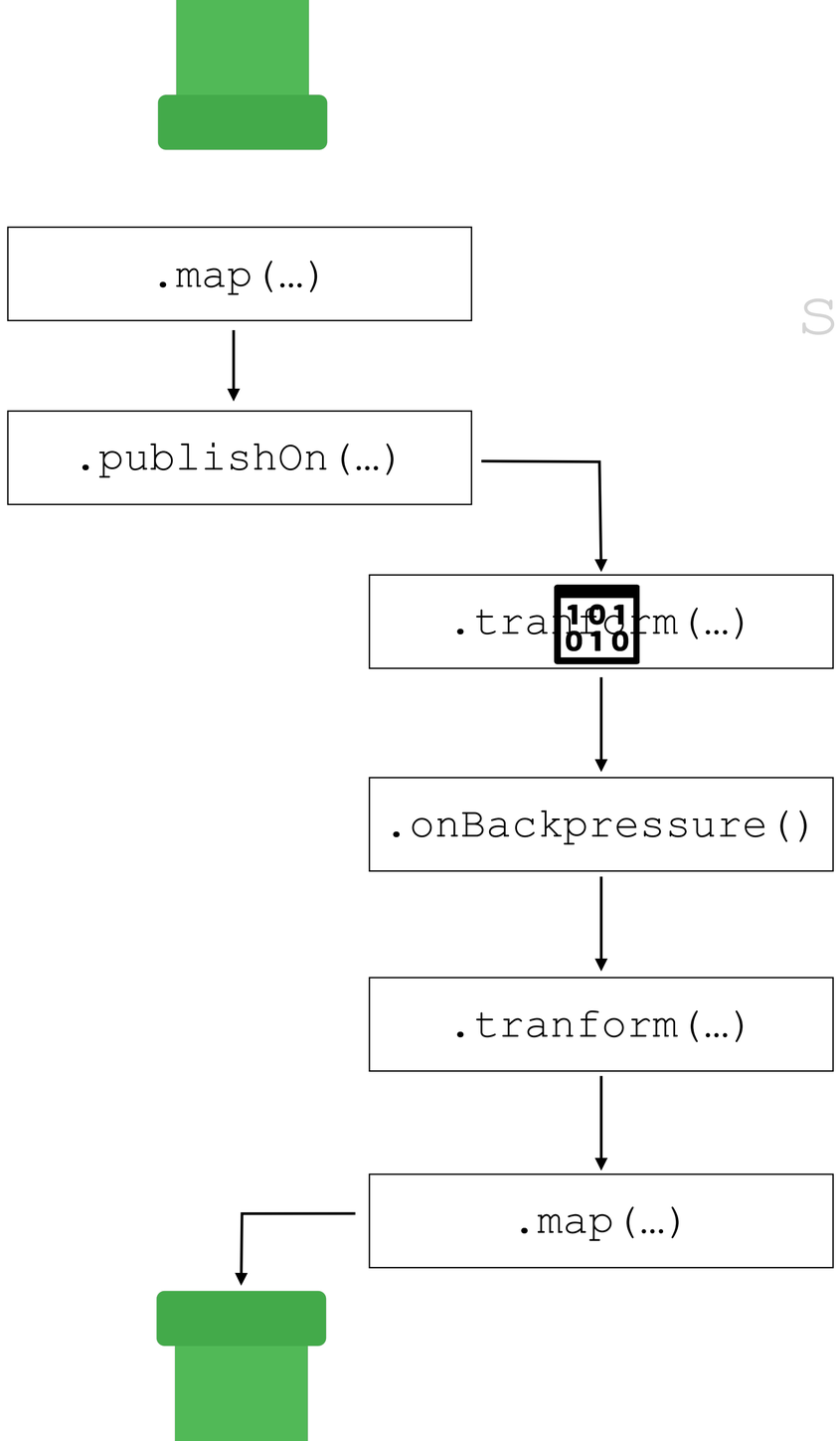

```
s.receive()
 .map(WebSocketMessage::retain)
 .map(WebSocketMessage::getPayload)
 .publishOn(Schedulers.parallel())
 .transform(mapper::decode)
 .transform(this::doHandle)
 .onBackpressureBuffer()
 .transform(s -> mapper.encode(…))
 .map(db -> new WebSocketMessage(…))
 .as(session::send);
```



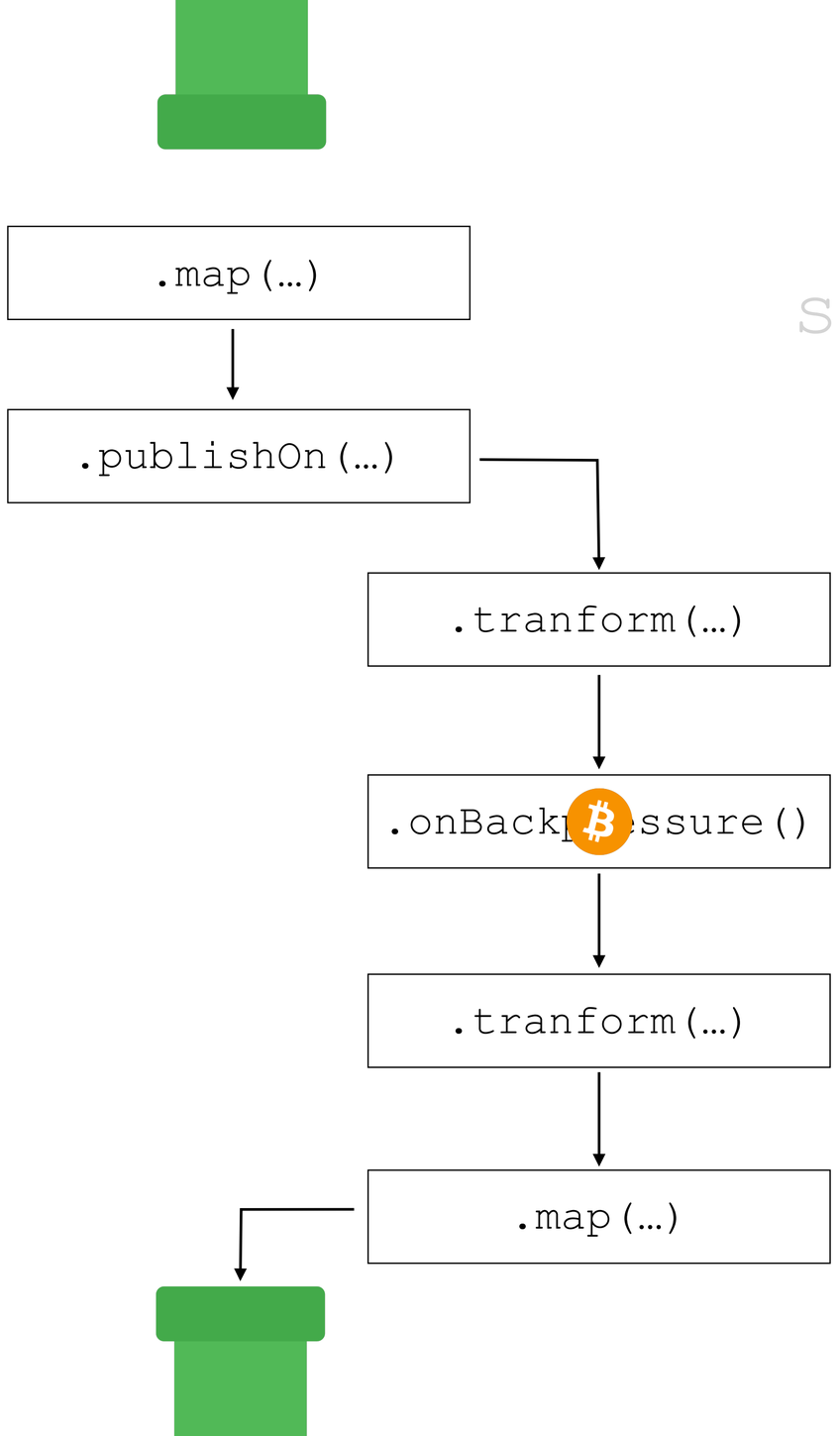

```
s.receive()
 .map(WebSocketMessage::retain)
 .map(WebSocketMessage::getPayload)
 .publishOn(Schedulers.parallel())
 .transform(mapper::decode)
 .transform(this::doHandle)
 .onBackpressureBuffer()
 .transform(s -> mapper.encode(…))
 .map(db -> new WebSocketMessage(…))
 .as(session::send);
```



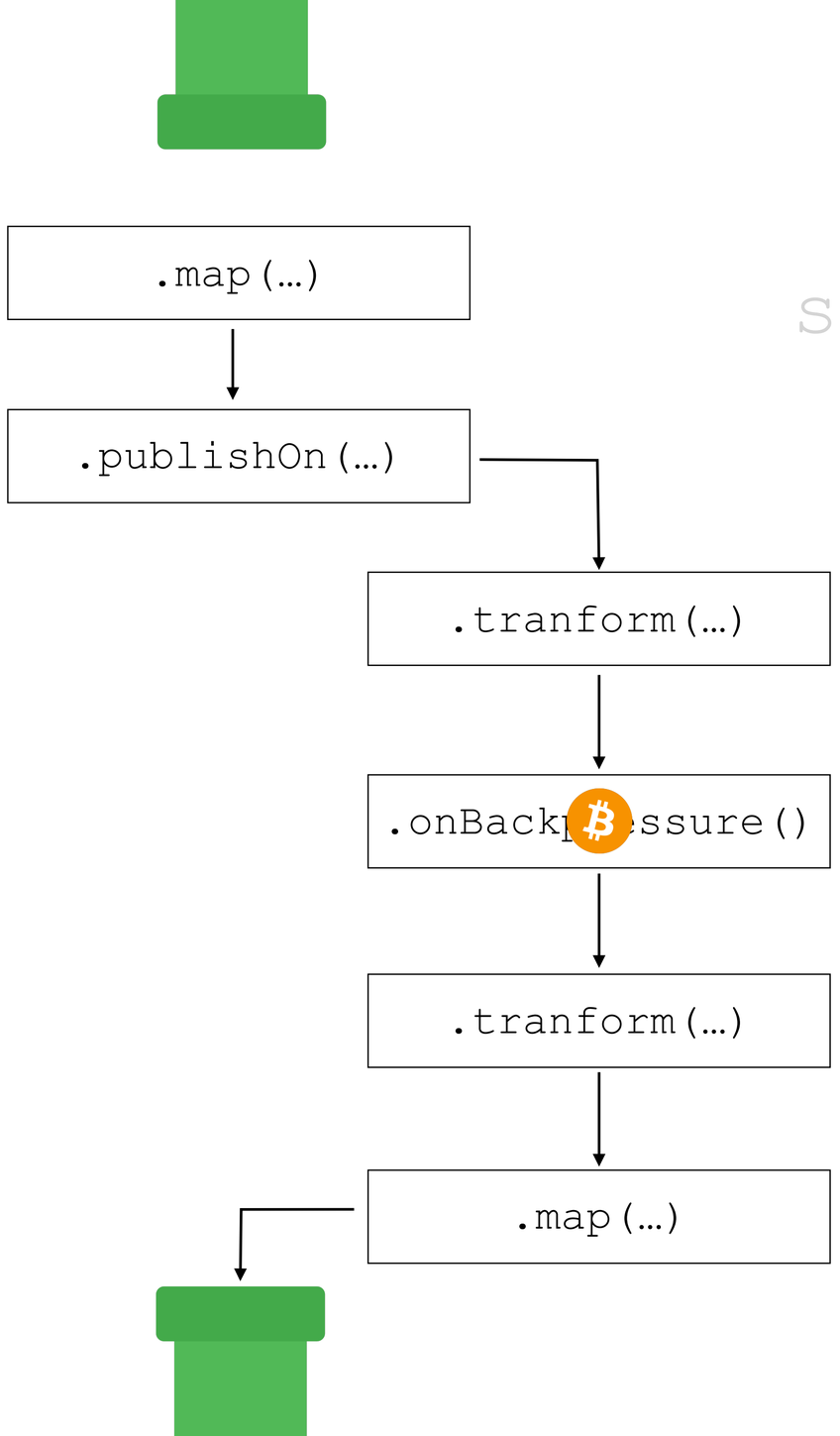

```
s.receive()
 .map(WebSocketMessage::retain)
 .map(WebSocketMessage::getPayload)
 .publishOn(Schedulers.parallel())
 .transform(mapper::decode)
 .transform(this::doHandle)
 .onBackpressureBuffer()
 .transform(s -> mapper.encode(…))
 .map(db -> new WebSocketMessage(…))
 .as(session::send);
```



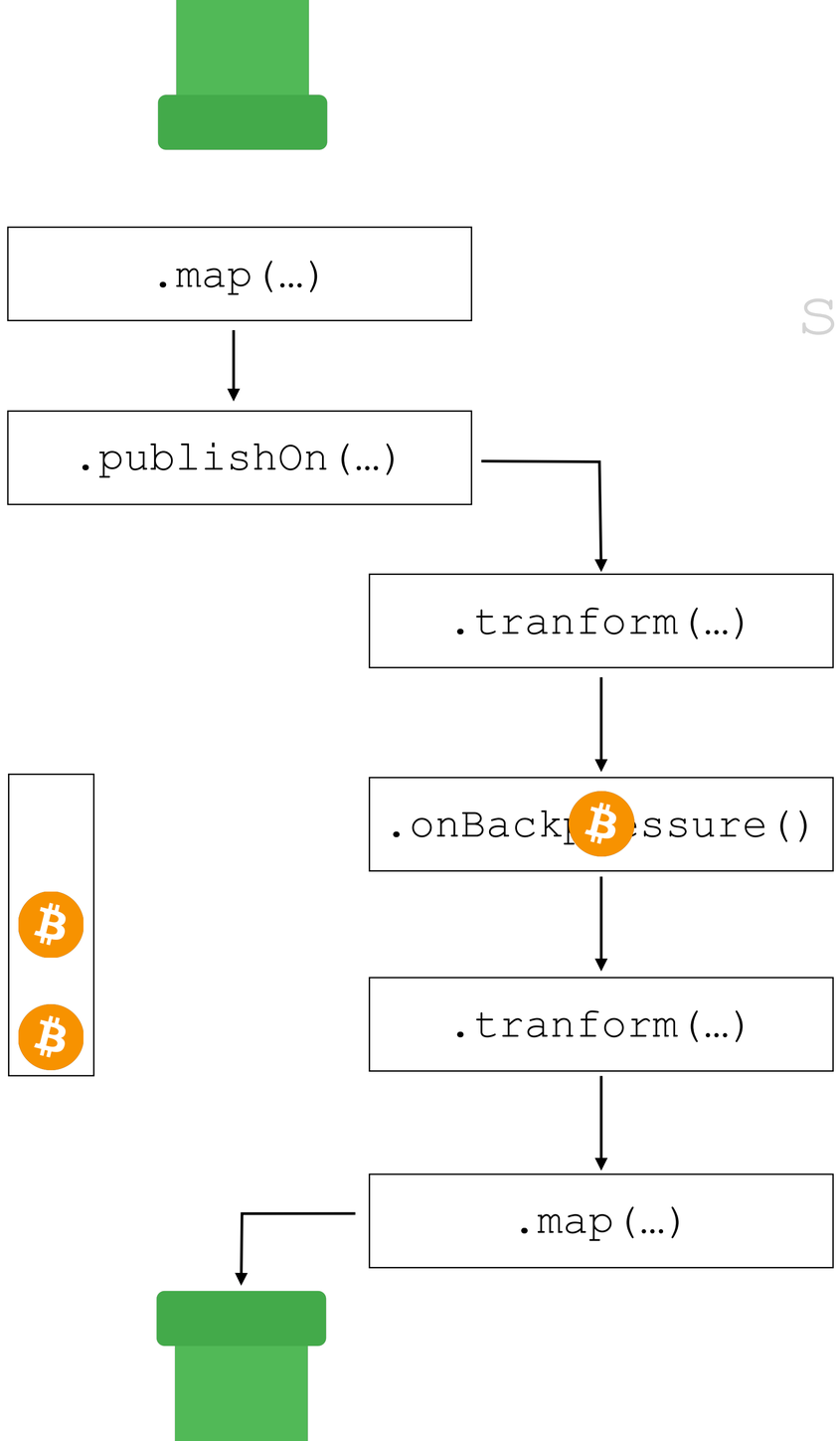

```
s.receive()
 .map(WebSocketMessage::retain)
 .map(WebSocketMessage::getPayload)
 .publishOn(Schedulers.parallel())
 .transform(mapper::decode)
 .transform(this::doHandle)
 .onBackpressureBuffer()
 .transform(s -> mapper.encode(…))
 .map(db -> new WebSocketMessage(…))
 .as(session::send);
```



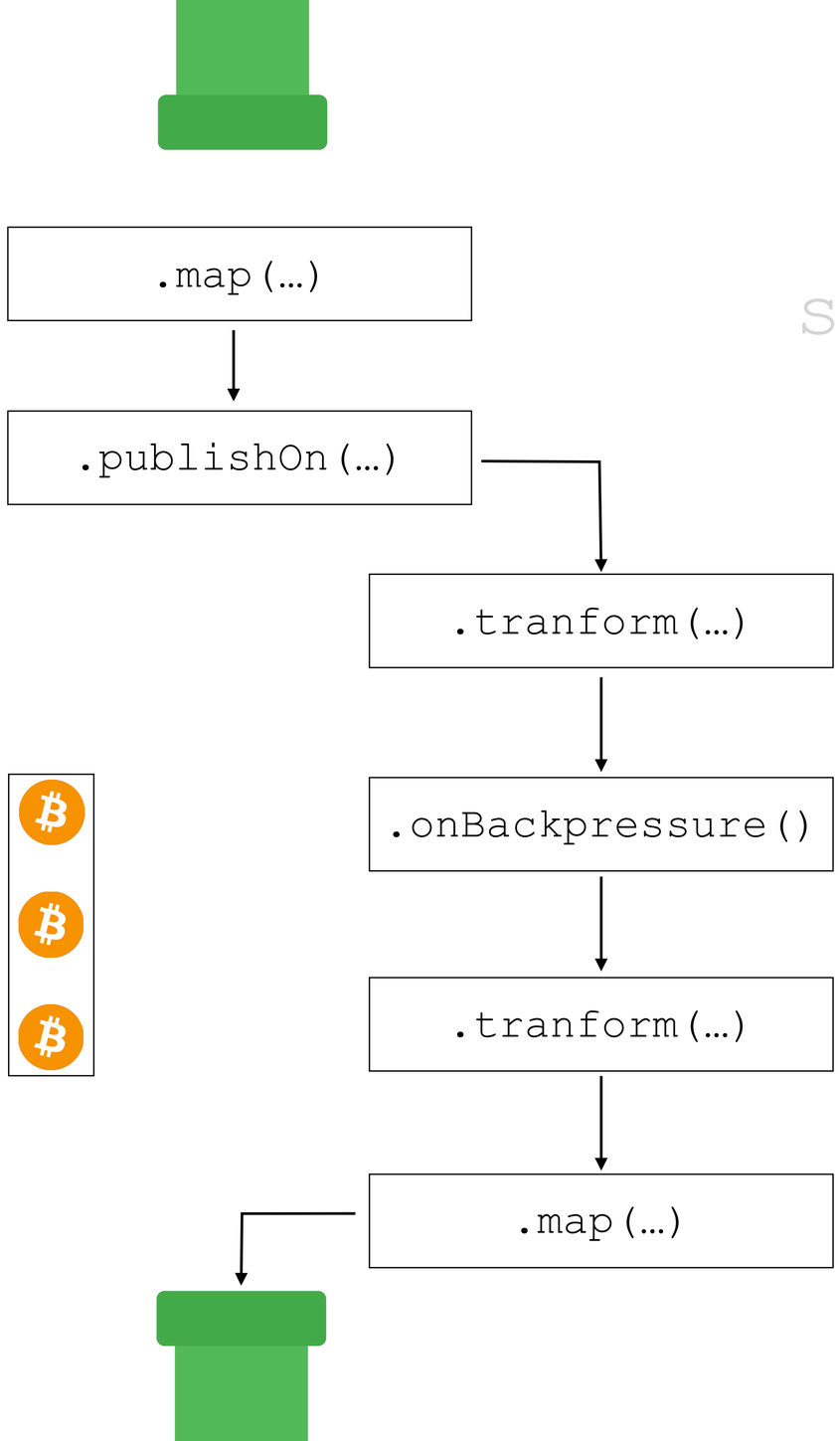

```
s.receive()
 .map(WebSocketMessage::retain)
 .map(WebSocketMessage::getPayload)
 .publishOn(Schedulers.parallel())
 .transform(mapper::decode)
 .transform(this::doHandle)
 .onBackpressureBuffer()
 .transform(s -> mapper.encode(…))
 .map(db -> new WebSocketMessage(…))
 .as(session::send);
```



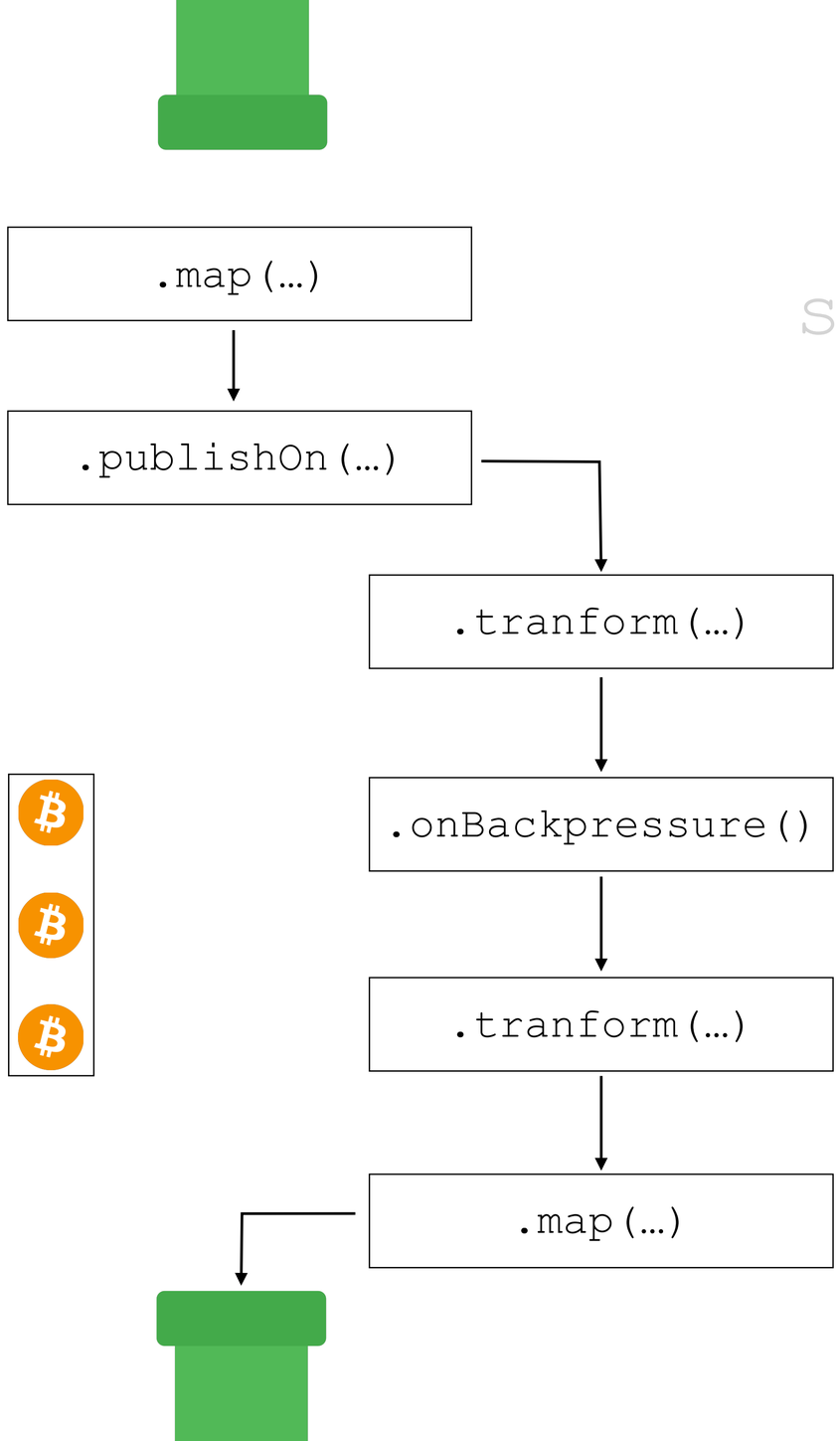

```
s.receive()
 .map(WebSocketMessage::retain)
 .map(WebSocketMessage::getPayload)
 .publishOn(Schedulers.parallel())
 .transform(mapper::decode)
 .transform(this::doHandle)
 .onBackpressureBuffer()
 .transform(s -> mapper.encode(…))
 .map(db -> new WebSocketMessage(…))
 .as(session::send);
```



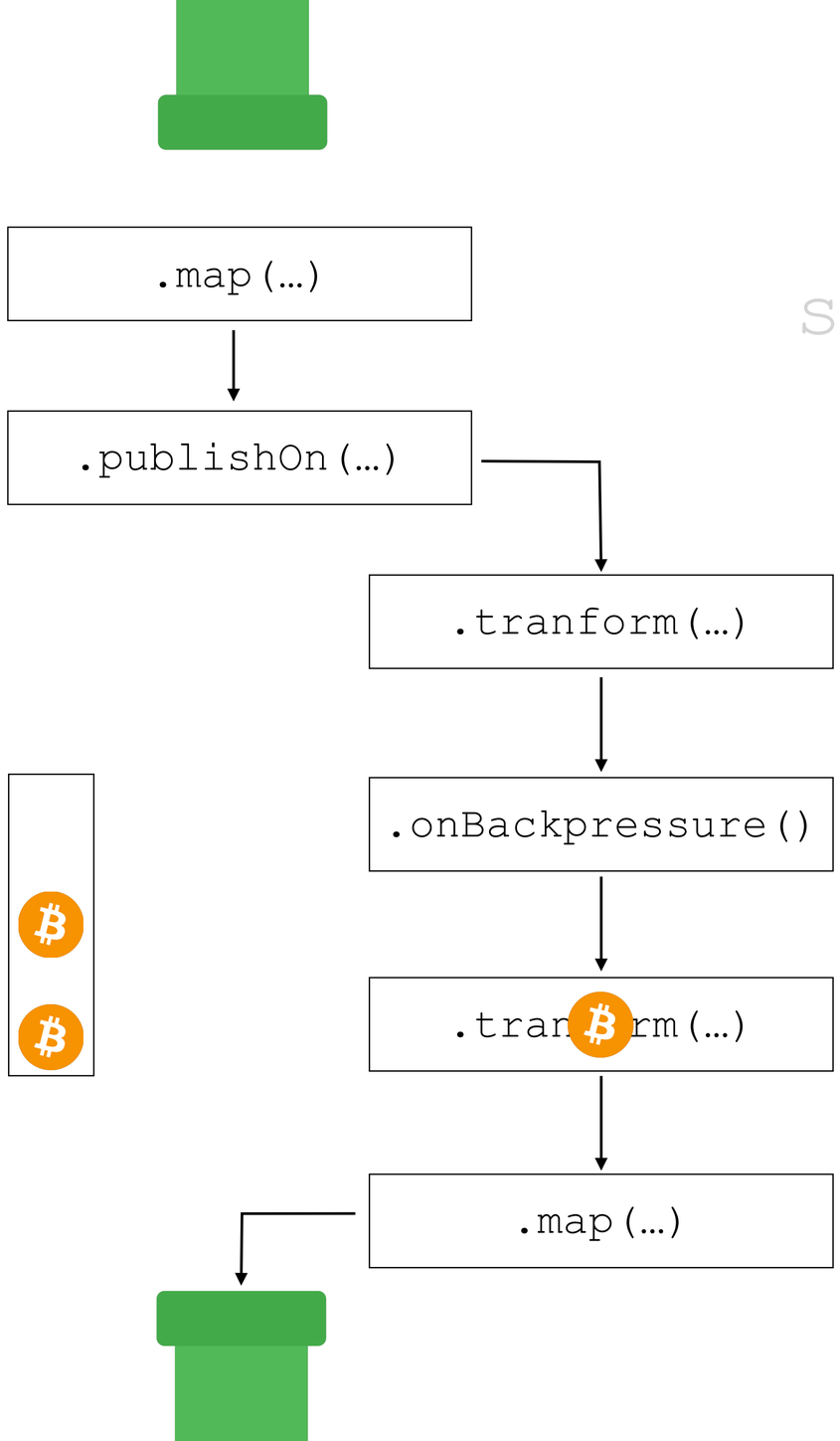

```
s.receive()
 .map(WebSocketMessage::retain)
 .map(WebSocketMessage::getPayload)
 .publishOn(Schedulers.parallel())
 .transform(mapper::decode)
 .transform(this::doHandle)
 .onBackpressureBuffer()
 .transform(s -> mapper.encode(…))
 .map(db -> new WebSocketMessage(…))
 .as(session::send);
```



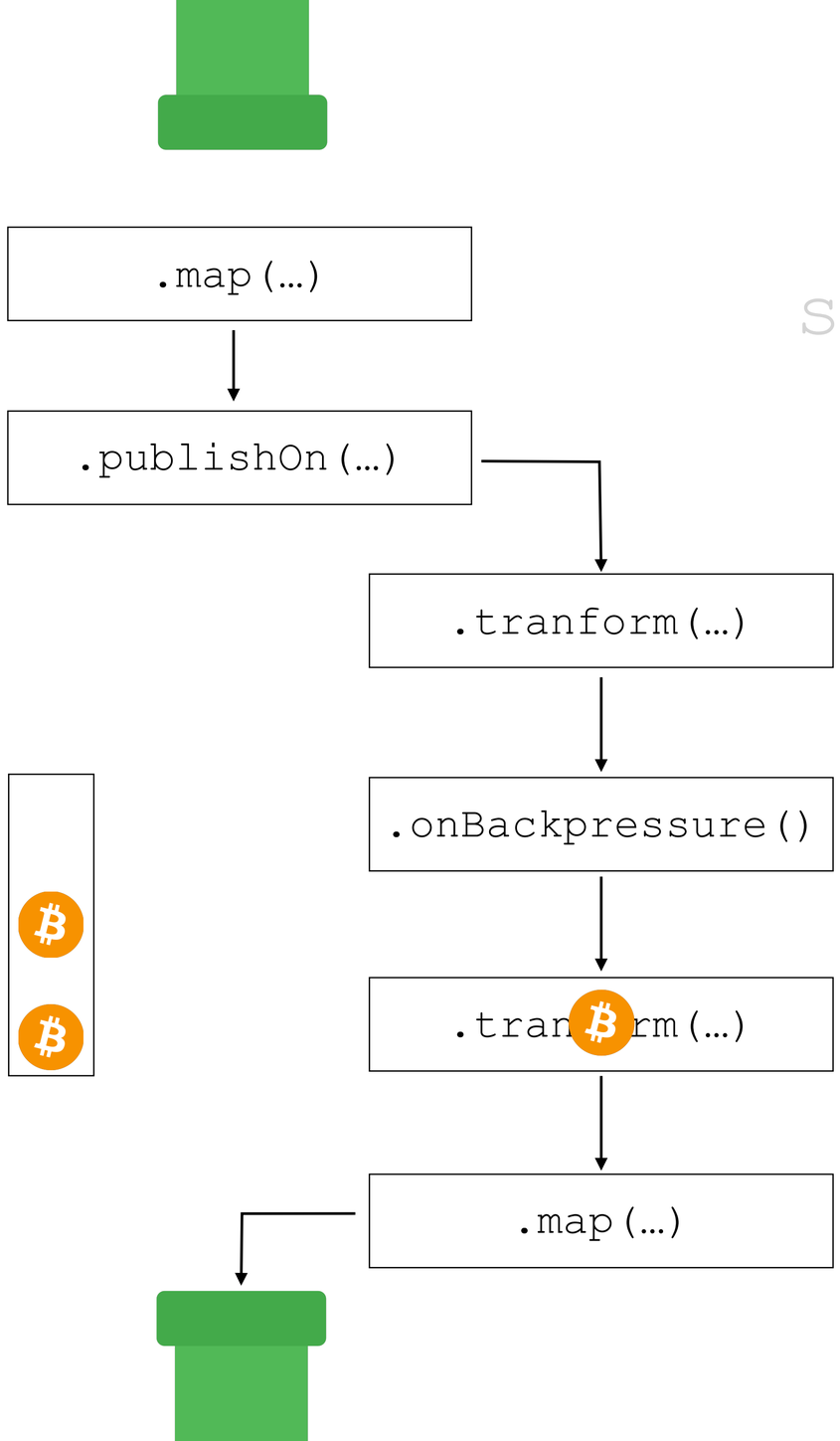

```
s.receive()
 .map(WebSocketMessage::retain)
 .map(WebSocketMessage::getPayload)
 .publishOn(Schedulers.parallel())
 .transform(mapper::decode)
 .transform(this::doHandle)
 .onBackpressureBuffer()
 .transform(s -> mapper.encode(…))
 .map(db -> new WebSocketMessage(…))
 .as(session::send);
```



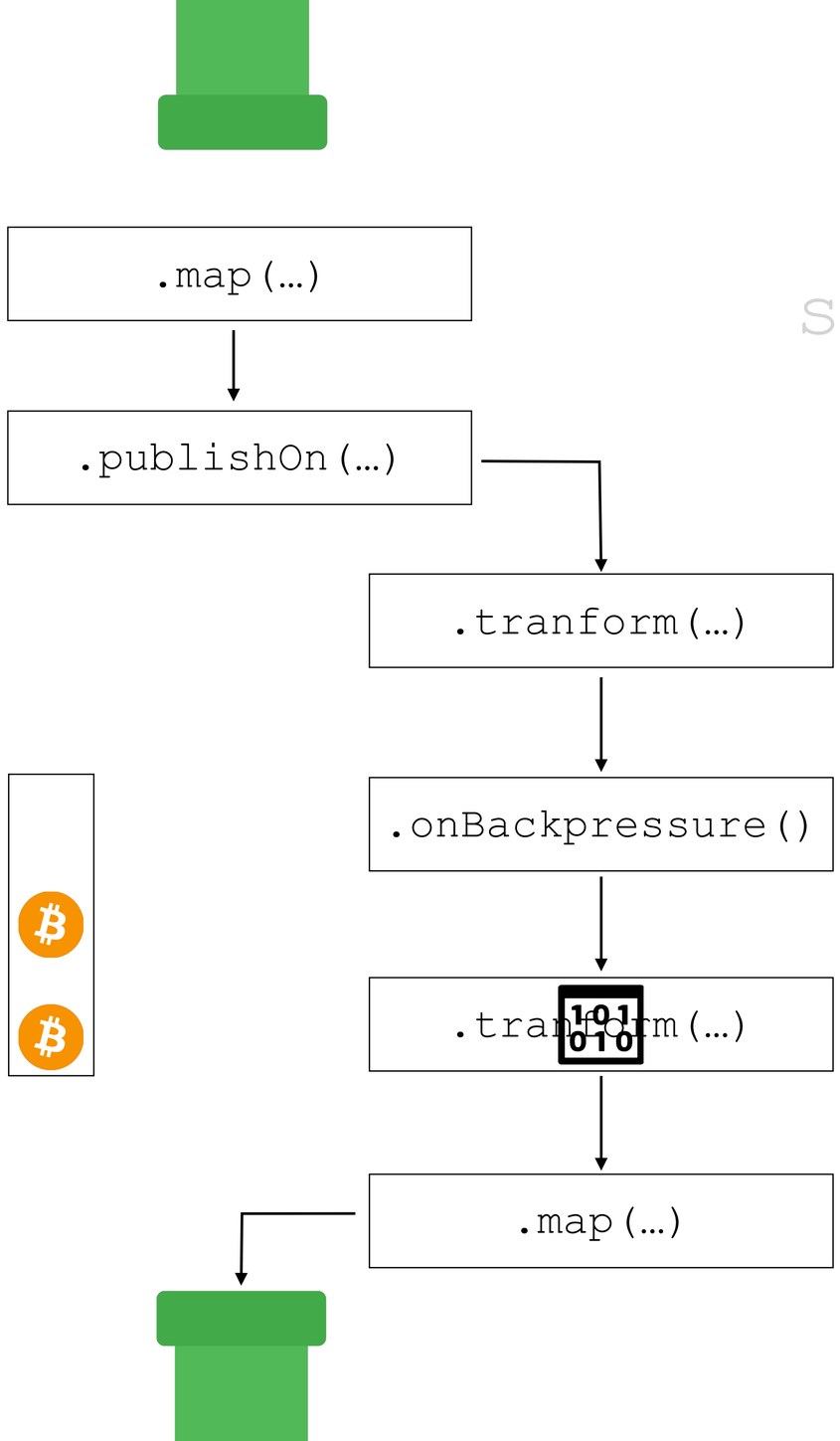

```
s.receive()
 .map(WebSocketMessage::retain)
 .map(WebSocketMessage::getPayload)
 .publishOn(Schedulers.parallel())
 .transform(mapper::decode)
 .transform(this::doHandle)
 .onBackpressureBuffer()
 .transform(s -> mapper.encode(…))
 .map(db -> new WebSocketMessage(…))
 .as(session::send);
```



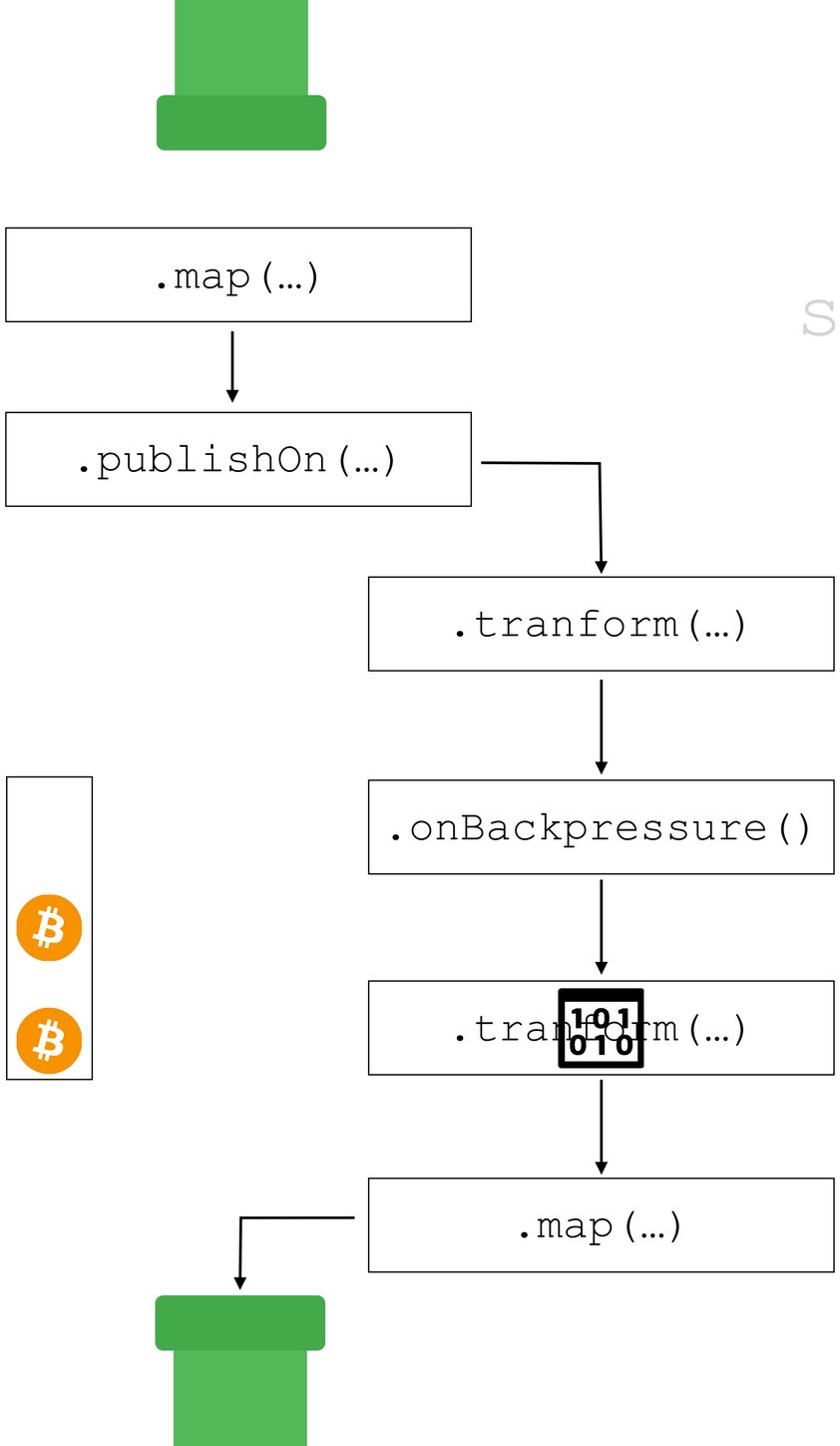

```
s.receive()
 .map(WebSocketMessage::retain)
 .map(WebSocketMessage::getPayload)
 .publishOn(Schedulers.parallel())
 .transform(mapper::decode)
 .transform(this::doHandle)
 .onBackpressureBuffer()
 .transform(s -> mapper.encode(…))
 .map(db -> new WebSocketMessage(…))
 .as(session::send);
```



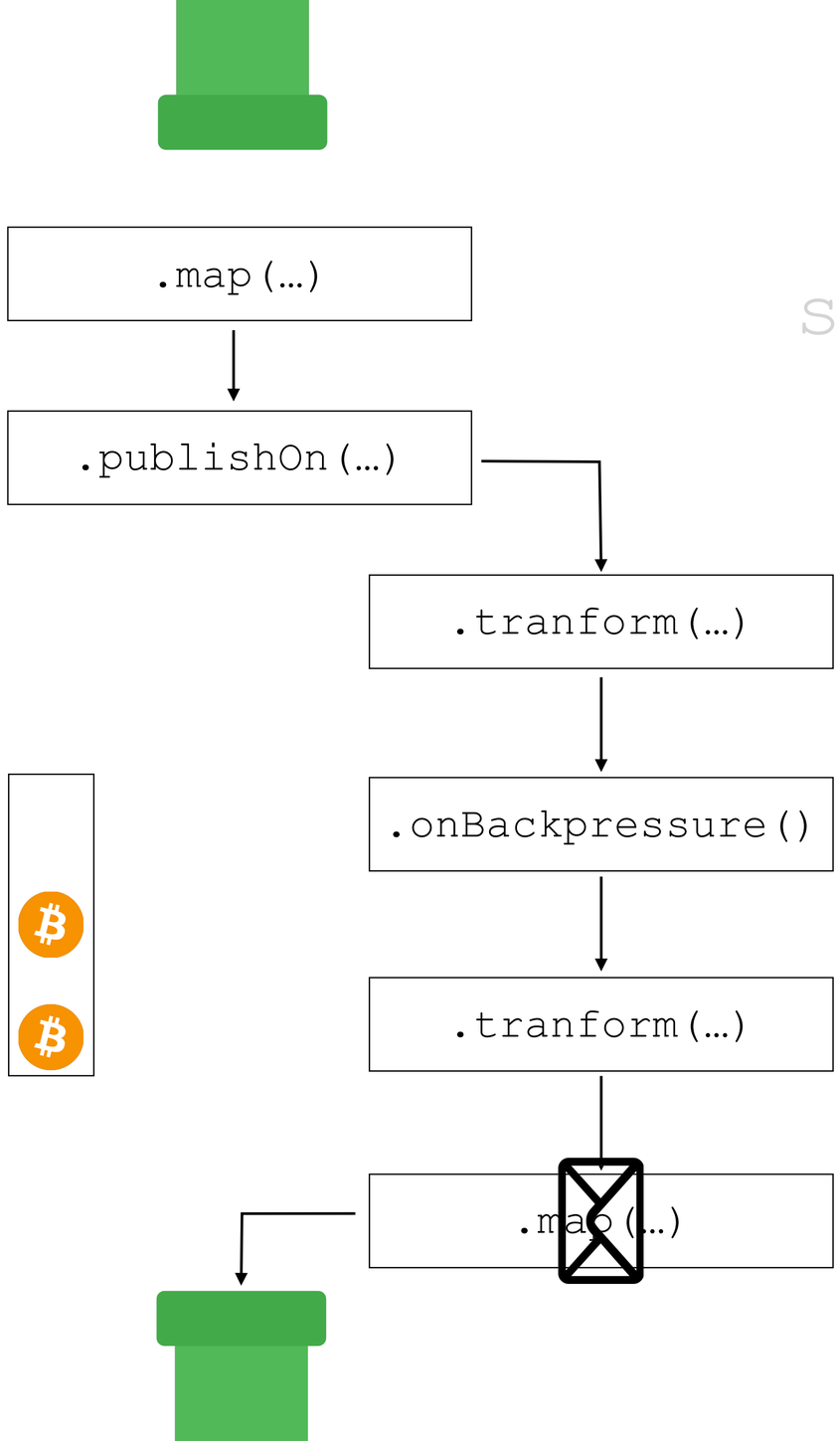

```
s.receive()
 .map(WebSocketMessage::retain)
 .map(WebSocketMessage::getPayload)
 .publishOn(Schedulers.parallel())
 .transform(mapper::decode)
 .transform(this::doHandle)
 .onBackpressureBuffer()
 .transform(s -> mapper.encode(…))
 .map(db -> new WebSocketMessage(…))
 .as(session::send);
```



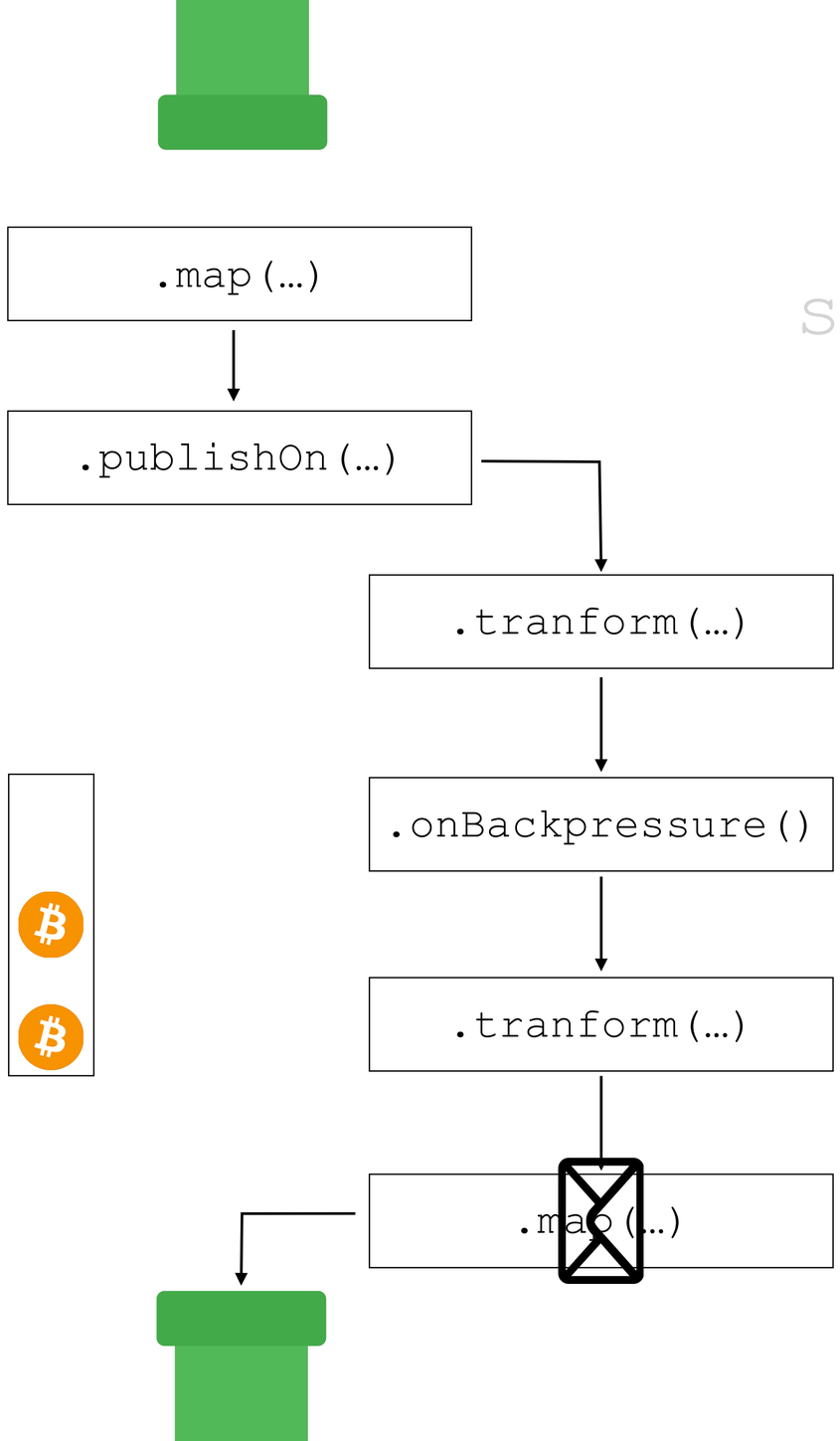

```
s.receive()
 .map(WebSocketMessage::retain)
 .map(WebSocketMessage::getPayload)
 .publishOn(Schedulers.parallel())
 .transform(mapper::decode)
 .transform(this::doHandle)
 .onBackpressureBuffer()
 .transform(s -> mapper.encode(…))
 .map(db -> new WebSocketMessage(…))
 .as(session::send);
```



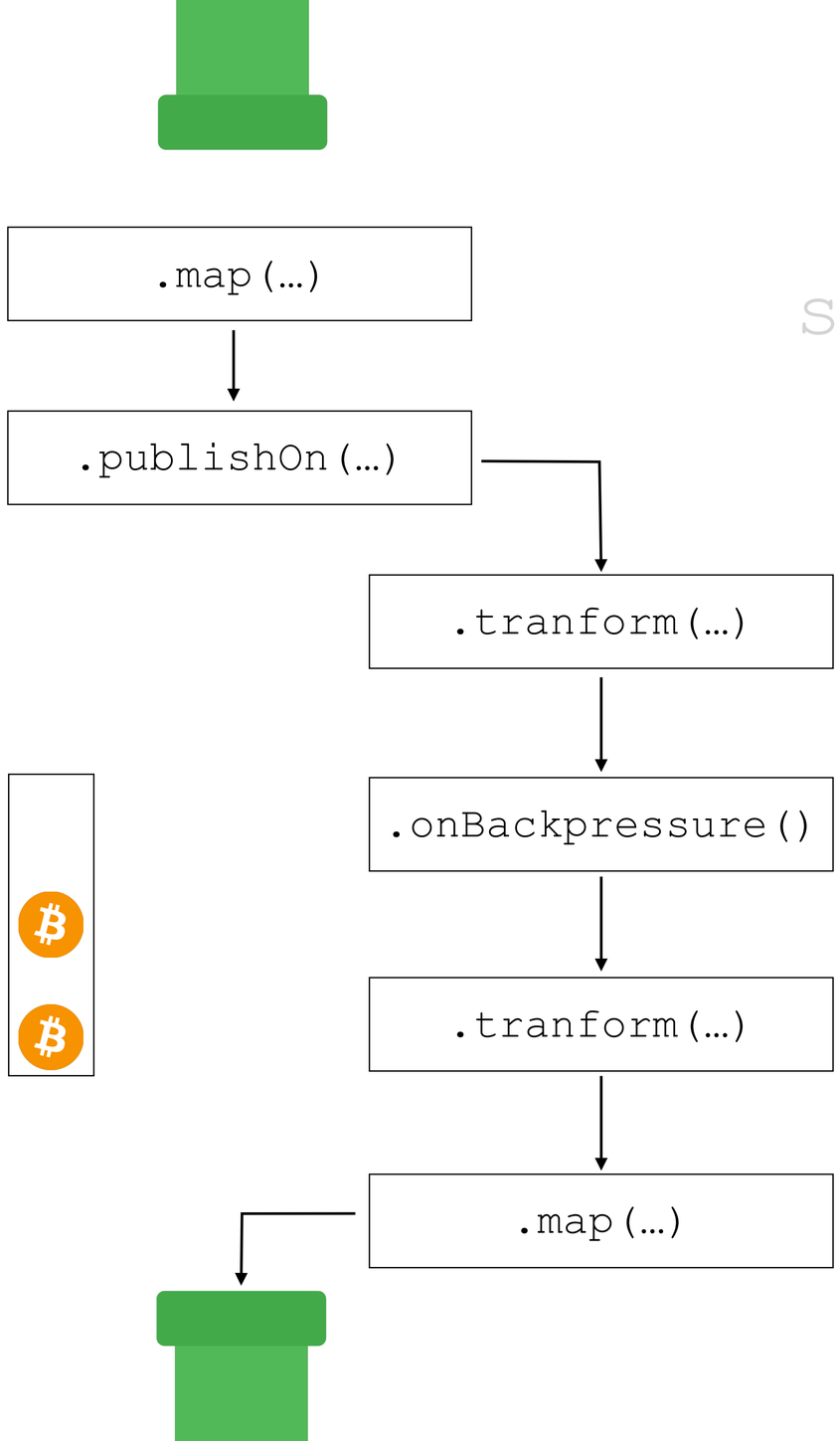

```
s.receive()
 .map(WebSocketMessage::retain)
 .map(WebSocketMessage::getPayload)
 .publishOn(Schedulers.parallel())
 .transform(mapper::decode)
 .transform(this::doHandle)
 .onBackpressureBuffer()
 .transform(s -> mapper.encode(…))
 .map(db -> new WebSocketMessage(…))
 .as(session::send);
```




```
s.receive()
 .map(WebSocketMessage::retain)
 .map(WebSocketMessage::getPayload)
 .publishOn(Schedulers.parallel())
 .transform(mapper::decode)
 .transform(this::doHandle)
 .onBackpressureBuffer()
 .transform(s -> mapper.encode(…))
 .map(db -> new WebSocketMessage(…))
 .as(session::send);
```




```
s.receive()
 .map(WebSocketMessage::retain)
 .map(WebSocketMessage::getPayload)
 .publishOn(Schedulers.parallel())
 .transform(mapper::decode)
 .transform(this::doHandle)
 .onBackpressureBuffer()
 .transform(s -> mapper.encode(…))
 .map(db -> new WebSocketMessage(…))
 .as(session::send);
```




```
s.receive()
 .map(WebSocketMessage::retain)
 .map(WebSocketMessage::getPayload)
 .publishOn(Schedulers.parallel())
 .transform(mapper::decode)
 .transform(this::doHandle)
 .onBackpressureBuffer()
 .transform(s -> mapper.encode(…))
 .map(db -> new WebSocketMessage(…))
 .as(session::send);
```

Что запомнить!?

Что запомнить!?

- `WebSocketMessage.retain/release` для подсчёта ссылок

Что запомнить!?

- `WebSocketMessage.retain/release` для подсчёта ссылок
- `.publishOn()` для переноса работу с Event-Loop

Помогает

Помогает


Работать с Netty

Помогает

 Работать с Netty

 Построить чистый асинхронный код

Помогает

 Работать с Netty

 Построить чистый асинхронный код

 Управлять потоками

Помогает

 Работать с Netty

 Построить чистый асинхронный код

 Управлять потоками

 Управлять Backpressure

Не очень

Не очень

Конфигурировать WebSocket API

Не очень

👎 Конфигурировать WebSocket API

👎 Конвертировать `bytes <-> java`

Не очень

👎 Конфигурировать WebSocket API

👎 Конвертировать `bytes <-> java`

👎 Управлять ссылками

Первые данные

Откуда брать?

Откуда брать?

- Bitmex - WebSocket API

Откуда брать?

- Bitmex - WebSocket API
- Bitfinex - WebSocket API

Что нужно?

Что нужно?

- WebSocket Client

```
interface WebSocketClient {
 Mono<Void> execute(URI url,
                    WebSocketHandler handler);
}
```

```
interface WebSocketClient {
 Mono<Void> execute(URI url,
                    WebSocketHandler handler);
}
```

```
interface WebSocketClient {
 Mono<Void> execute(URI url,
                    WebSocketHandler handler);
}
```

Что создадим?

Что создадим?

- `CryptoService.java`

Что создадим?

- `CryptoService.java`
- `BitmexMessage.java`

Что создадим?

- `CryptoService.java`
- `BitmexMessage.java`
- `BitmexMessageMapper.java`

Что создадим?

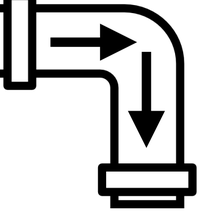
- CryptoService.java
- BitmexMessage.java
- BitmexMessageMapper.java
- BitmexService.java

Что создадим?

- CryptoService.java
- BitmexMessage.java
- BitmexMessageMapper.java
- **BitmexService.java**

Talk is cheap. Show me the code.

- Linus Torvalds

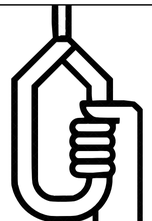


map(…)

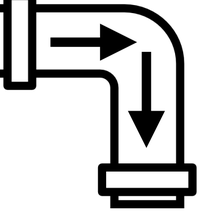
publishOn(…)

flatMapIterable(…)

then()



```
Flux.create(sink ->
...
 s -> s.receive()
        .skip(6)
        .map(WSM::getPayloadAsText)
        .publishOn(…)
        .flatMapIterable(…)
        .doOnNext(sink::next)
        .then();
...
);
```

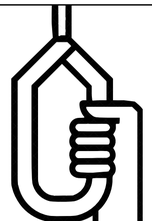


map(…)

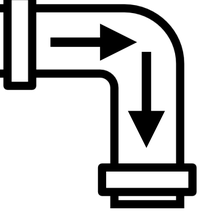
publishOn(…)

flatMapIterable(…)

then()



```
Flux.create(sink ->
...
 s -> s.receive()
       .skip(6)
       .map(WSM::getPayloadAsText)
       .publishOn(…)
       .flatMapIterable(…)
       .doOnNext(sink::next)
       .then();
...
);
```

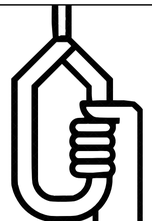


map(…)

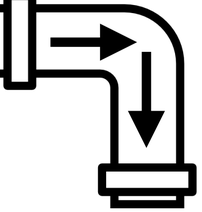
publishOn(…)

flatMapIterable(…)

then()



```
Flux.create(sink ->
...
 s -> s.receive()
       .skip(6)
       .map(WSM::getPayloadAsText)
       .publishOn(…)
       .flatMapIterable(…)
       .doOnNext(sink::next)
       .then();
...
);
```

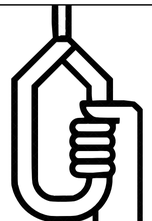


map(…)

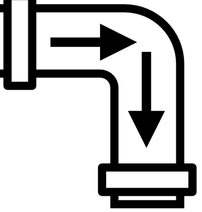
publishOn(…)

flatMapIterable(…)

then()



```
Flux.create(sink ->
...
 s -> s.receive()
       .skip(6)
       .map(WSM::getPayloadAsText)
       .publishOn(…)
       .flatMapIterable(…)
       .doOnNext(sink::next)
       .then();
...
);
```



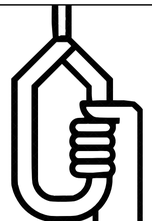
map(…)

publishOn(…)

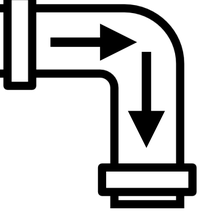
flatMapIterable(…)

doOnNext(sink::next)

then()



```
Flux.create(sink ->
...
 s -> s.receive()
       .skip(6)
       .map(WSM::getPayloadAsText)
       .publishOn(…)
       .flatMapIterable(…)
       .doOnNext(sink::next)
       .then();
...
);
```



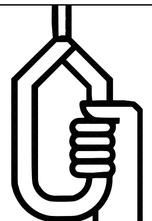
map(…)

publishOn(…)

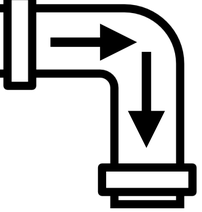
flatMapIterable(…)

doOnNext(sink::next)

then()



```
Flux.create(sink ->
...
 s -> s.receive()
       .skip(6)
       .map(WSM::getPayloadAsText)
       .publishOn(…)
       .flatMapIterable(…)
       .doOnNext(sink::next)
       .then();
...
);
```



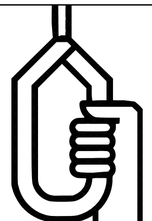
map(…)

publishOn(…)

flatMapIterable(…)

doOnNext(sink::next)

then()



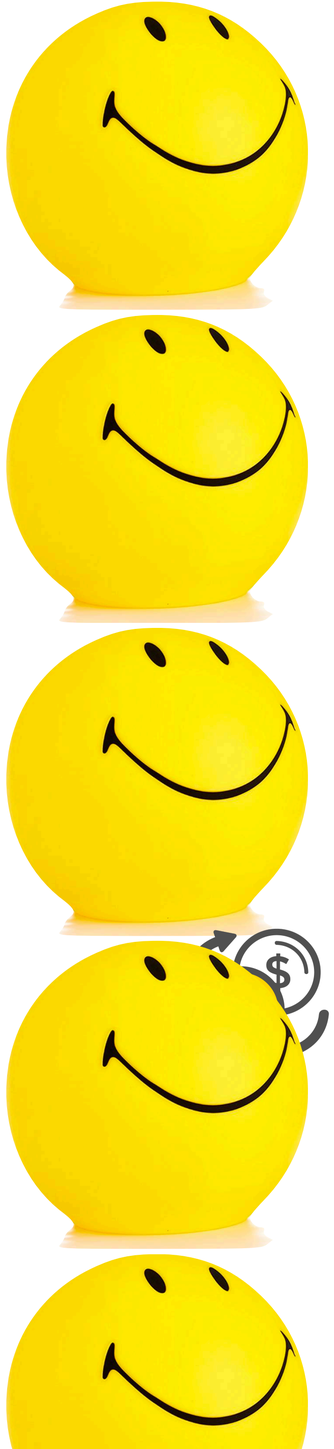
```
Flux.create(sink ->
...
 s -> s.receive()
       .skip(6)
       .map(WSM::getPayloadAsText)
       .publishOn(…)
       .flatMapIterable(…)
       .doOnNext(sink::next)
       .then();
...
);
```

ClientWebSocketConnection

BitmexWebSocketConnection

ClientWebSocketConnection

BitmexWebSocketConnection



ClientWebSocketConnection

ClientWebSocketConnection

ClientWebSocketConnection

ClientWebSocketConnection

ClientWebSocketConnection

BitmexWebSocketConnection

BitmexWebSocketConnection

BitmexWebSocketConnection

BitmexWebSocketConnection

BitmexWebSocketConnection

Talk is cheap. Show me the code.

- Linus Torvalds

ClientWebSocketConnection

ClientWebSocketConnection

ClientWebSocketConnection

ClientWebSocketConnection

ClientWebSocketConnection

BitmexWebSocketConnection

BitmexWebSocketConnection

BitmexWebSocketConnection

BitmexWebSocketConnection

BitmexWebSocketConnection

ClientWebSocketConnection

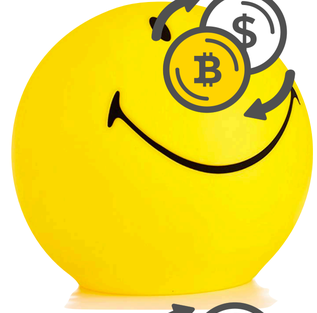
ClientWebSocketConnection

ClientWebSocketConnection

BitmexWebSocketConnection

ClientWebSocketConnection

ClientWebSocketConnection



ClientWebSocketConnection

ClientWebSocketConnection

ClientWebSocketConnection

BitmexWebSocketConnection

ClientWebSocketConnection

ClientWebSocketConnection

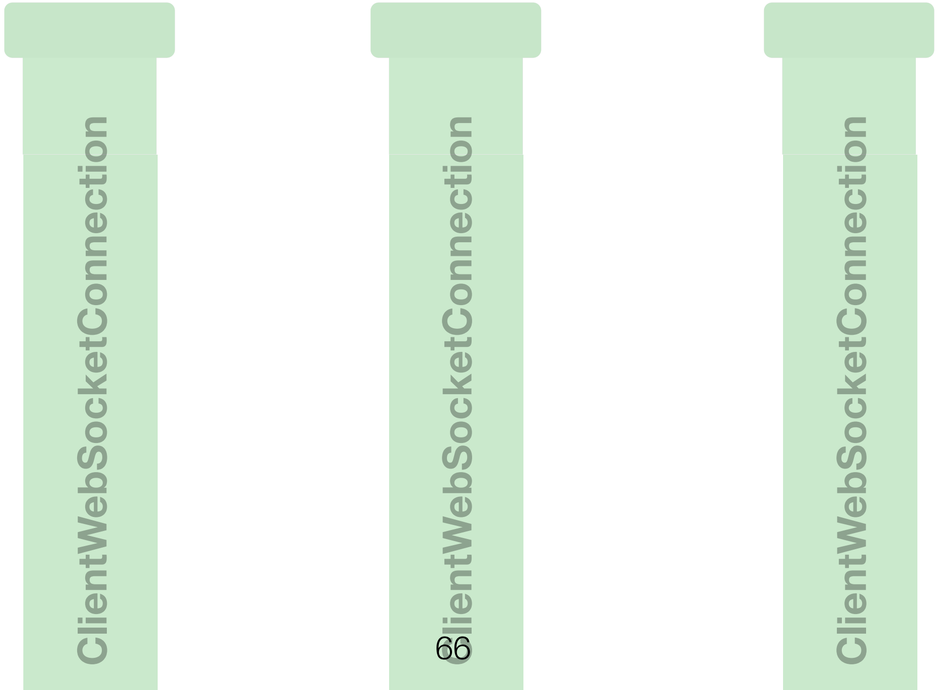
А как же Bitfinex?

Talk is cheap. Show me the code.

- Linus Torvalds

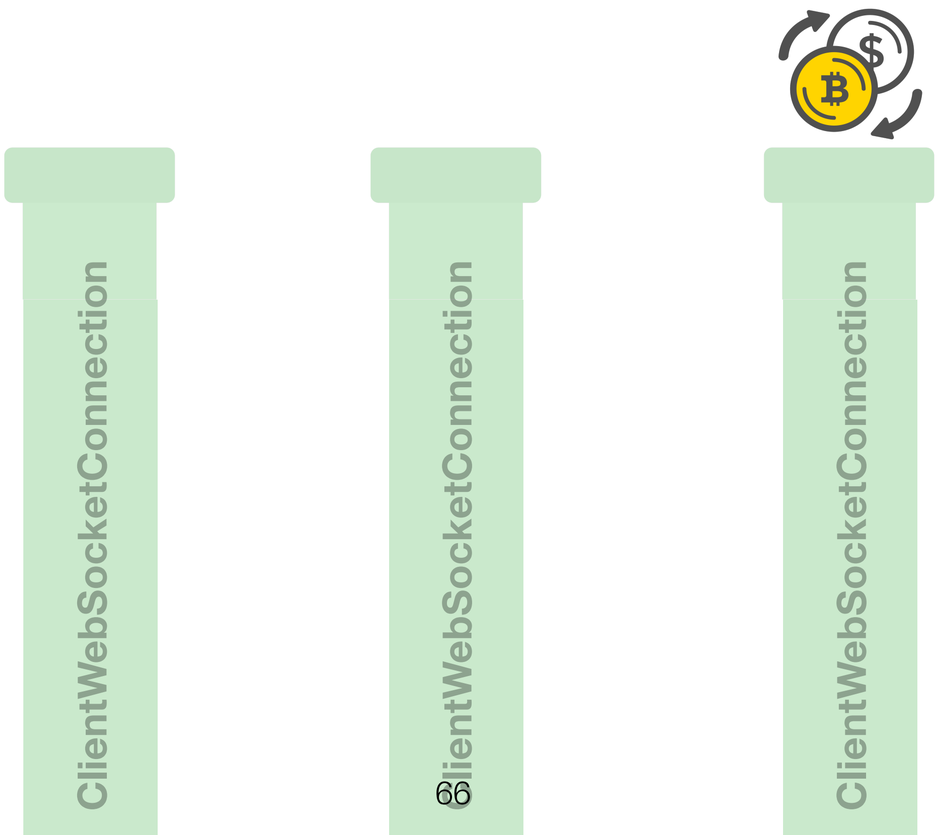
.publish()

DirectProcessor



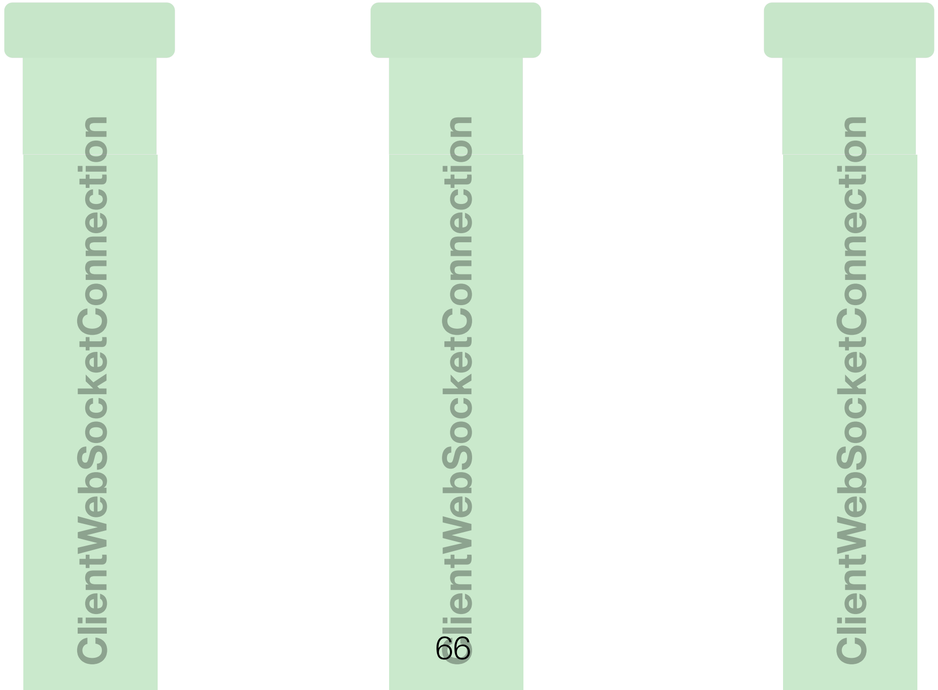
.publish()

DirectProcessor



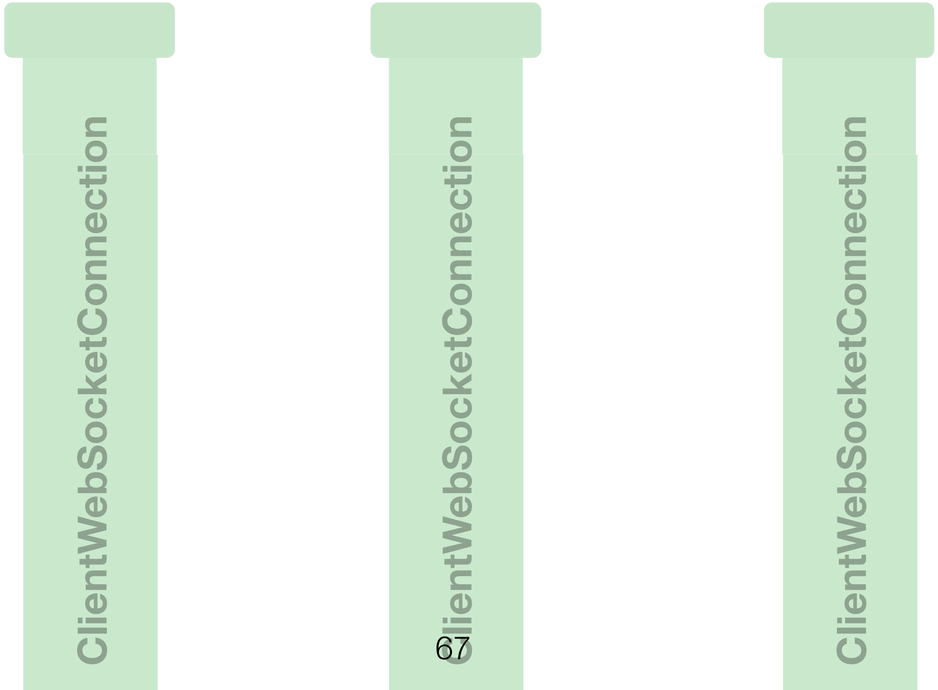
.publish()

DirectProcessor



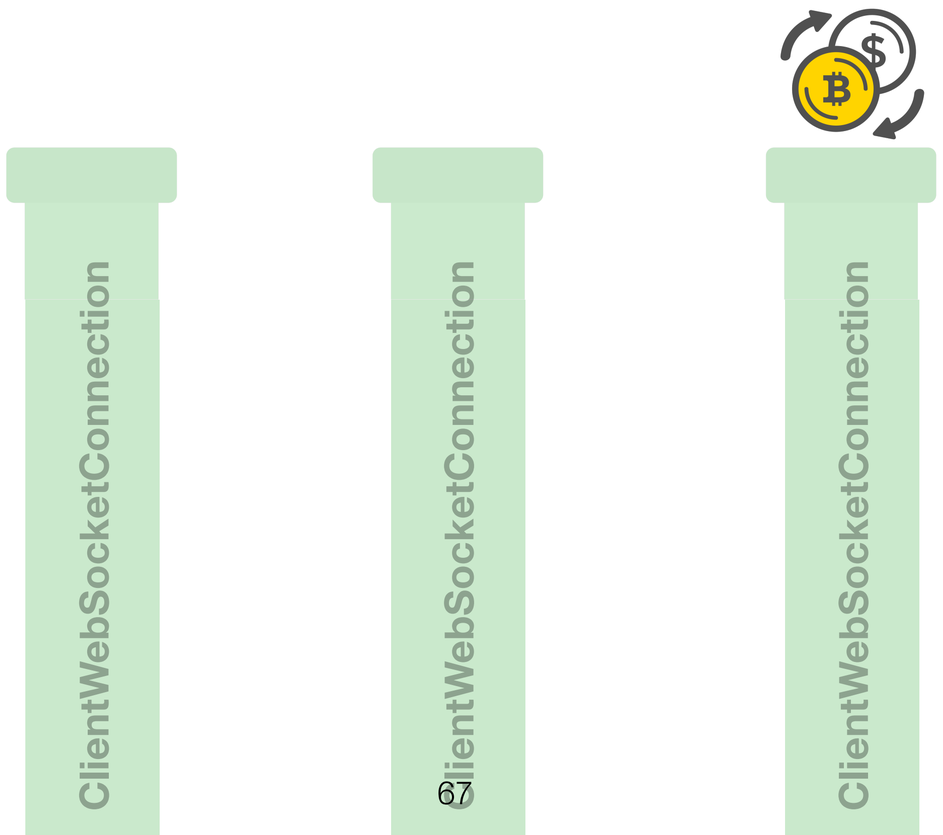
.publish()

DirectProcessor



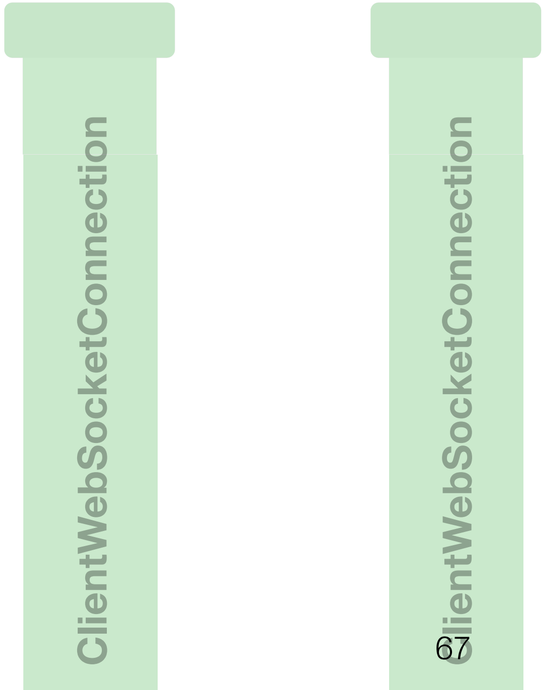
.publish()

DirectProcessor



.publish()

DirectProcessor



Что запомнить!?

Что запомнить!?

- `FluxSink` для перенаправления

Что запомнить!?

- `FluxSink` для перенаправления
- `.publish()` для уравниловки

Что запомнить!?

- `FluxSink` для перенаправления
- `.publish()` для уравниловки
- `DirectProcessor` для push

Помогает

Помогает


Работать с WebSocket

Помогает

 Работать с WebSocket

 Перенаправлять данные

Помогает

 Работать с WebSocket

 Перенаправлять данные

 Мультикастить данные

Помогает

 Работать с WebSocket

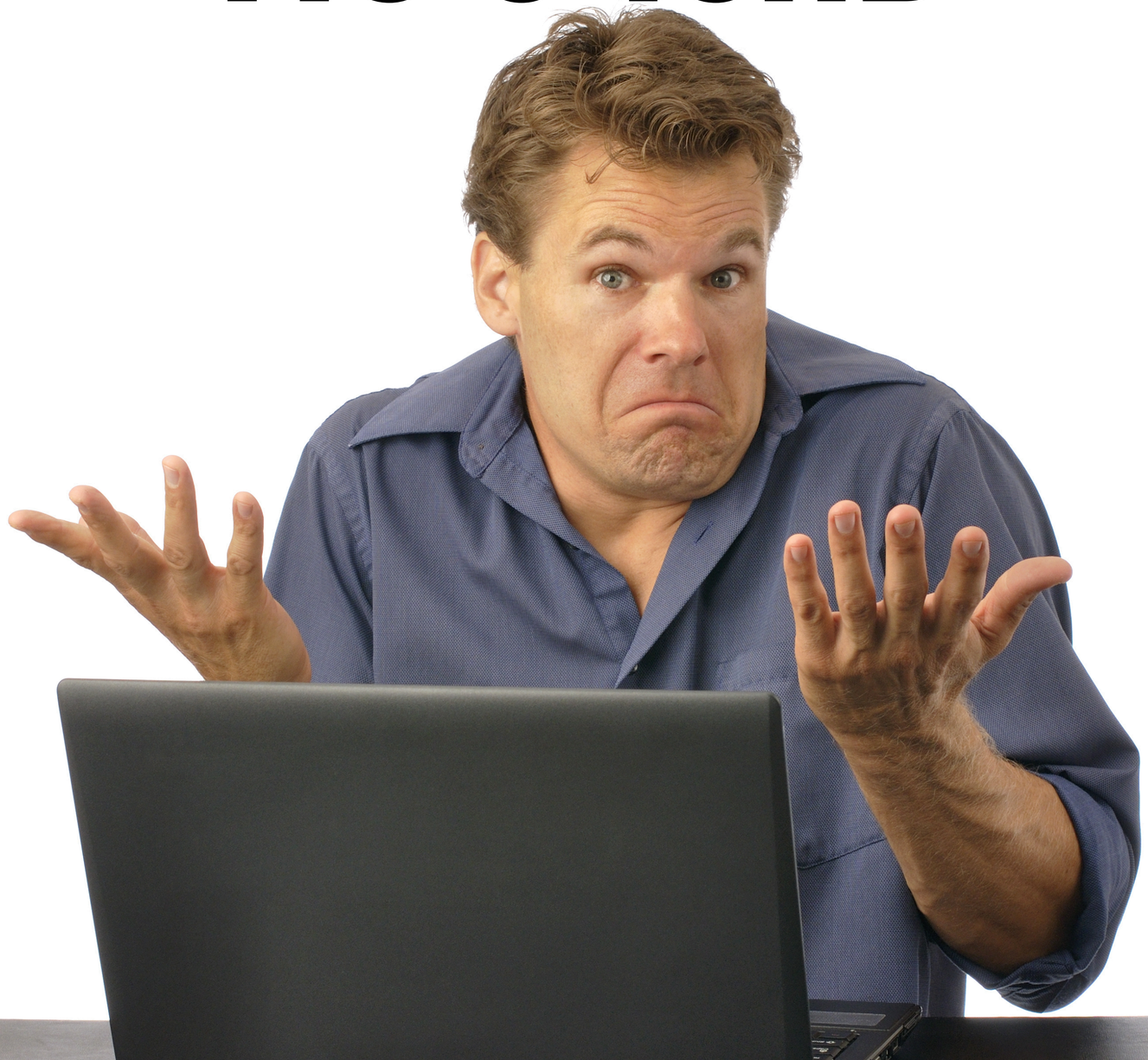
 Перенаправлять данные

 Мультикастить данные

 Объединять данные

Не очень

Не очень



Строим Кошелёк

Что нужно?

Что нужно?

- MongoDB - реактивная база данных

Что нужно?

- MongoDB - реактивная база данных
- Spring Data Mongo Reactive

```
interface ReactiveCrudRepository<T, ID> extends Repository<T, ID>
    <S extends T> Mono<S> save(S entity);
    <S extends T> Flux<S> saveAll(Iterable<S> entities);
    <S extends T> Flux<S> saveAll(Publisher<S> entityStream);
    Mono<T> findById(ID id);
    Mono<T> findById(Mono<ID> id);
    Mono<Boolean> existsById(ID id);
    Mono<Boolean> existsById(Mono<ID> id);
    Flux<T> findAll();
    Flux<T> findAllById(Iterable<ID> ids);
    Flux<T> findAllById(Publisher<ID> idStream);
    Mono<Long> count();
    Mono<Void> deleteById(ID id);
    Mono<Void> delete(T entity);
    Mono<Void> deleteAll(Iterable<? extends T> entities);
    Mono<Void> deleteAll(Publisher<? extends T> entityStream);
    Mono<Void> deleteAll();
}
```

```
interface ReactiveCrudRepository<T, ID> extends Repository<T, ID>
    <S extends T> Mono<S> save(S entity);
    <S extends T> Flux<S> saveAll(Iterable<S> entities);
    <S extends T> Flux<S> saveAll(Publisher<S> entityStream);
    Mono<T> findById(ID id);
    Mono<T> findById(Mono<ID> id);
    Mono<Boolean> existsById(ID id);
    Mono<Boolean> existsById(Mono<ID> id);
    Flux<T> findAll();
    Flux<T> findAllById(Iterable<ID> ids);
    Flux<T> findAllById(Publisher<ID> idStream);
    Mono<Long> count();
    Mono<Void> deleteById(ID id);
    Mono<Void> delete(T entity);
    Mono<Void> deleteAll(Iterable<? extends T> entities);
    Mono<Void> deleteAll(Publisher<? extends T> entityStream);
    Mono<Void> deleteAll();
}
```

Что нужно?

- MongoDB - реактивная база данных
- Spring Data Mongo Reactive
- Конфигурация SpringSecurity - для пользовательского доступа

```
class ReactiveSecurityContextHolder {
 static Mono<SecurityContext> getContext()
}
```

```
class ReactiveSecurityContextHolder {
 static Mono<SecurityContext> getContext()
}
```

Что создадим?

Что создадим?

- `Wallet.java`

Что создадим?

- `Wallet.java`
- `WalletRepository.java`

Что создадим?

- `Wallet.java`
- `WalletRepository.java`
- `WalletService.java`

Что создадим?

- `Wallet.java`
- `WalletRepository.java`
- `WalletService.java`
- `LocalWalletService.java`

Что создадим?

- Wallet.java
- WalletRepository.java
- WalletService.java
- LocalWalletService.java
- LocalMessageMapper.java

Что создадим?

- Wallet.java
- WalletRepository.java
- WalletService.java
- **LocalWalletService.java**
- LocalMessageMapper.java

Talk is cheap. Show me the code.

- Linus Torvalds

```
ReactiveSecurityContextHolder
    .getContext()
    .map(getAuthentication)
    .map(getName);
```

```
ReactiveSecurityContextHolder
    .getContext()
    .map(getAuthentication)
    .map(getName)
    .subscribe(out::println)
```

```
ReactiveSecurityContextHolder
    .getContext()
    .map(getAuthentication)
    .map(getName)
    .subscribe(out::println)
```

null

```
ReactiveSecurityContextHolder
    .getContext()
    .map(getAuthentication)
    .map(getName)
    .subscribe(out::println)
```

{emptyMap}

```
ReactiveSecurityContextHolder
    .getContext()                                    {emptyMap}
    .map(getAuthentication)
    .map(getName)
    .subscribe(out::println)
```

```
ReactiveSecurityContextHolder
    .getContext()
    .map(getAuthentication)
    .map(getName)
    .subscribe(out::println)
```

null

```
ReactiveSecurityContextHolder
    .getContext()
    .map(getAuthentication)
    .map(getName);
```

```
ReactiveSecurityContextHolder
    .getContext()
    .map(getAuthentication)
    .map(getName)
    .subscriberContext(security)
```

```
ReactiveSecurityContextHolder
    .getContext()
    .map(getAuthentication)
    .map(getName)
    .subscriberContext(security)
    .subscribe(out::println)
```

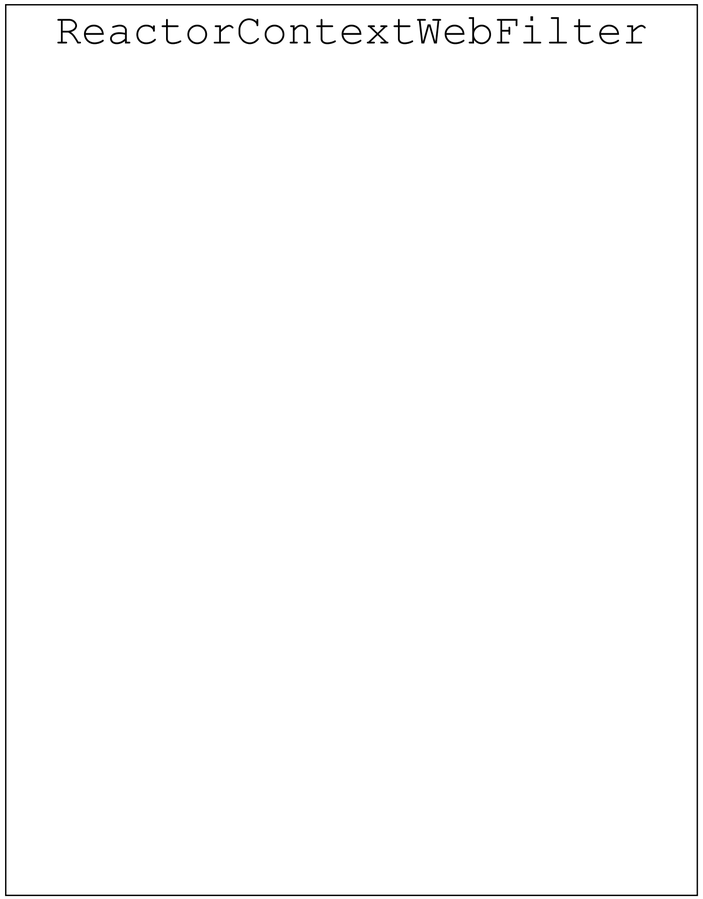
```
ReactiveSecurityContextHolder
    .getContext()
    .map(getAuthentication)
    .map(getName)
    .subscriberContext(security)
    .subscribe(out::println)                    {emptyMap}
```

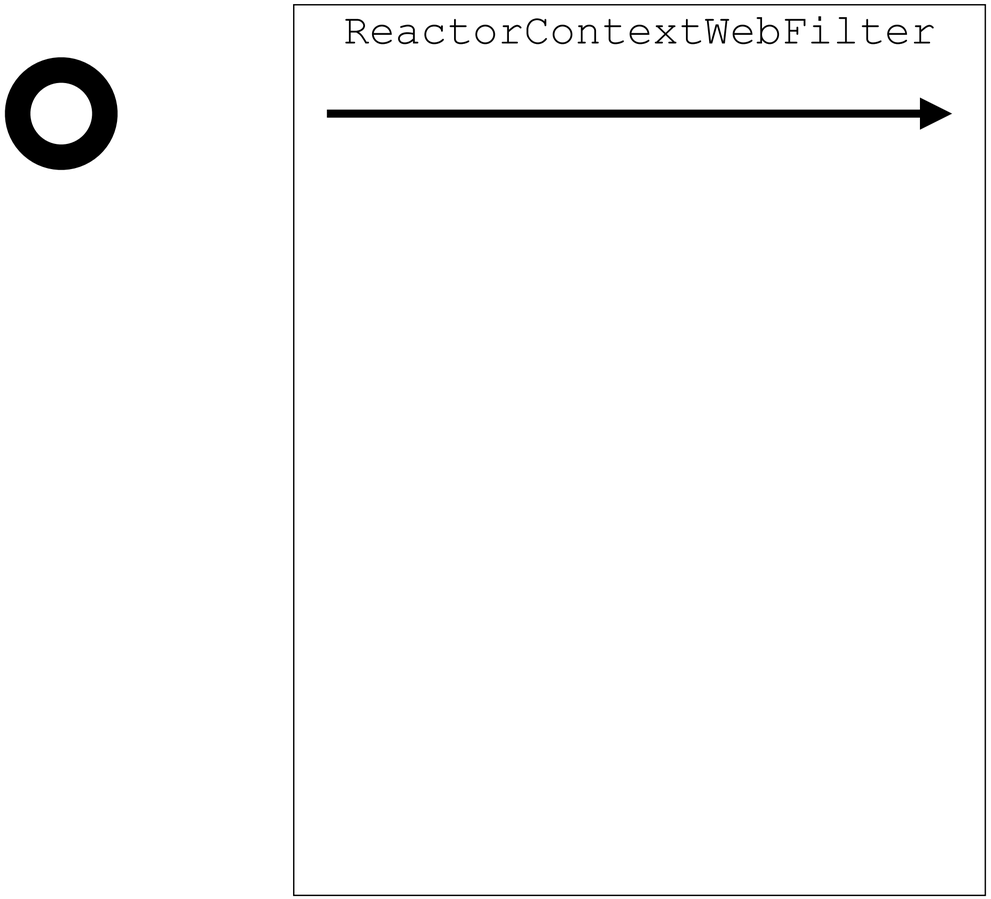
```
ReactiveSecurityContextHolder
    .getContext()
    .map(getAuthentication)
    .map(getName)
    .subscriberContext(security)  {emptyMap}
    .subscribe(out::println)
```

```
ReactiveSecurityContextHolder
    .getContext()
    .map(getAuthentication)
    .map(getName)
    .subscriberContext(security)  {security}
    .subscribe(out::println)
```

```
ReactiveSecurityContextHolder
    .getContext()                                   {security}
    .map(getAuthentication)
    .map(getName)
    .subscriberContext(security)
    .subscribe(out::println)
```

```
ReactiveSecurityContextHolder
    .getContext()
    .map(getAuthentication)
    .map(getName)
    .subscriberContext(security)
    .subscribe(out::println)
```

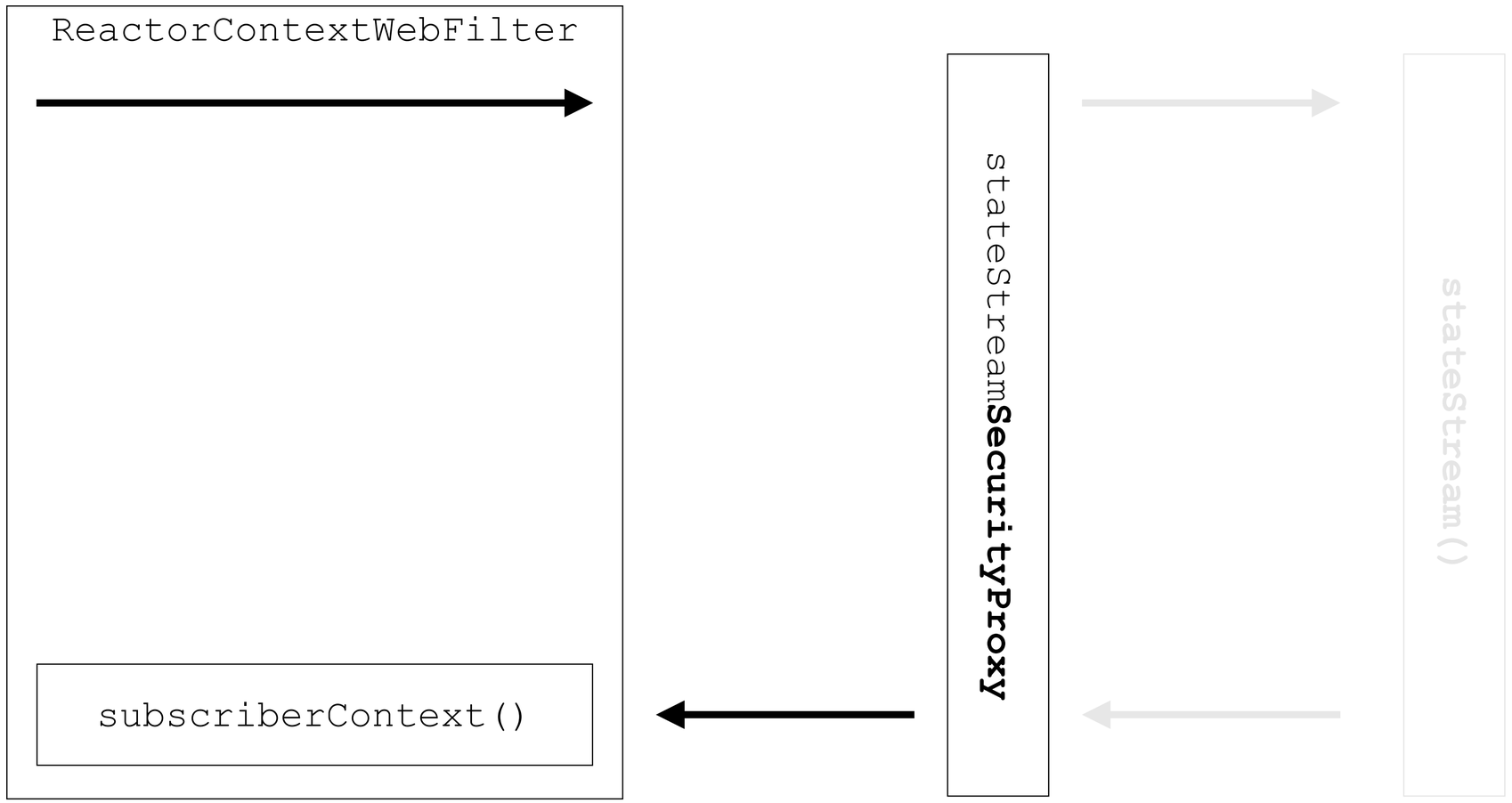

ReactorContextWebFilter


ReactorContextWebFilter

ReactorContextWebFilter

stateStream()
stateStreamSecurityProxy

ReactorContextWebFilter
stateStream**SecurityProxy**
stateStream()


ReactorContextWebFilter
subscriberContext()
stateStreamSecurityProxy
stateStream()

ReactorContextWebFilter
stateStream**SecurityProxy**
stateStream()
subscriberContext()

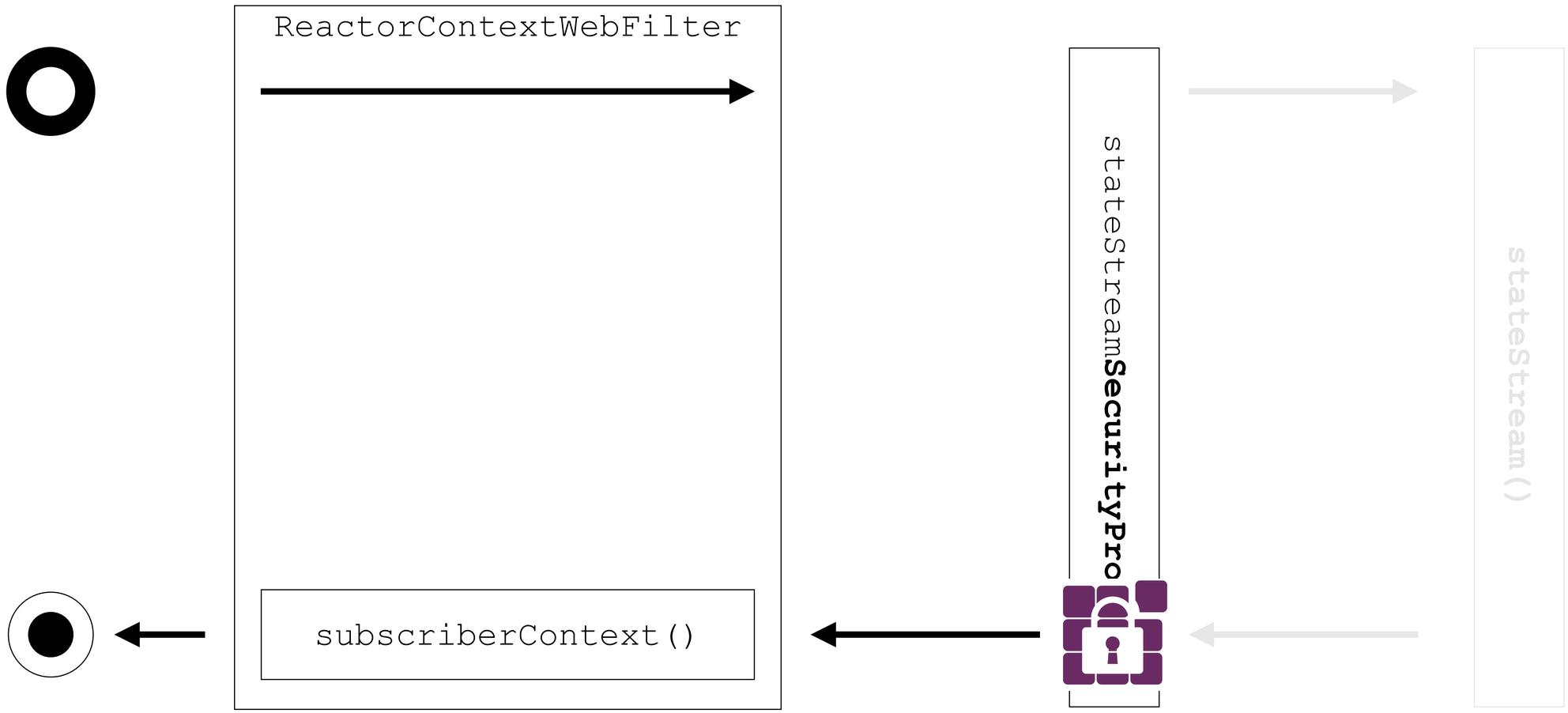
ReactorContextWebFilter
stateStream**SecurityProxy**
stateStream()
subscriberContext()

ReactorContextWebFilter
stateStream**SecurityProxy**
stateStream()
subscriberContext()

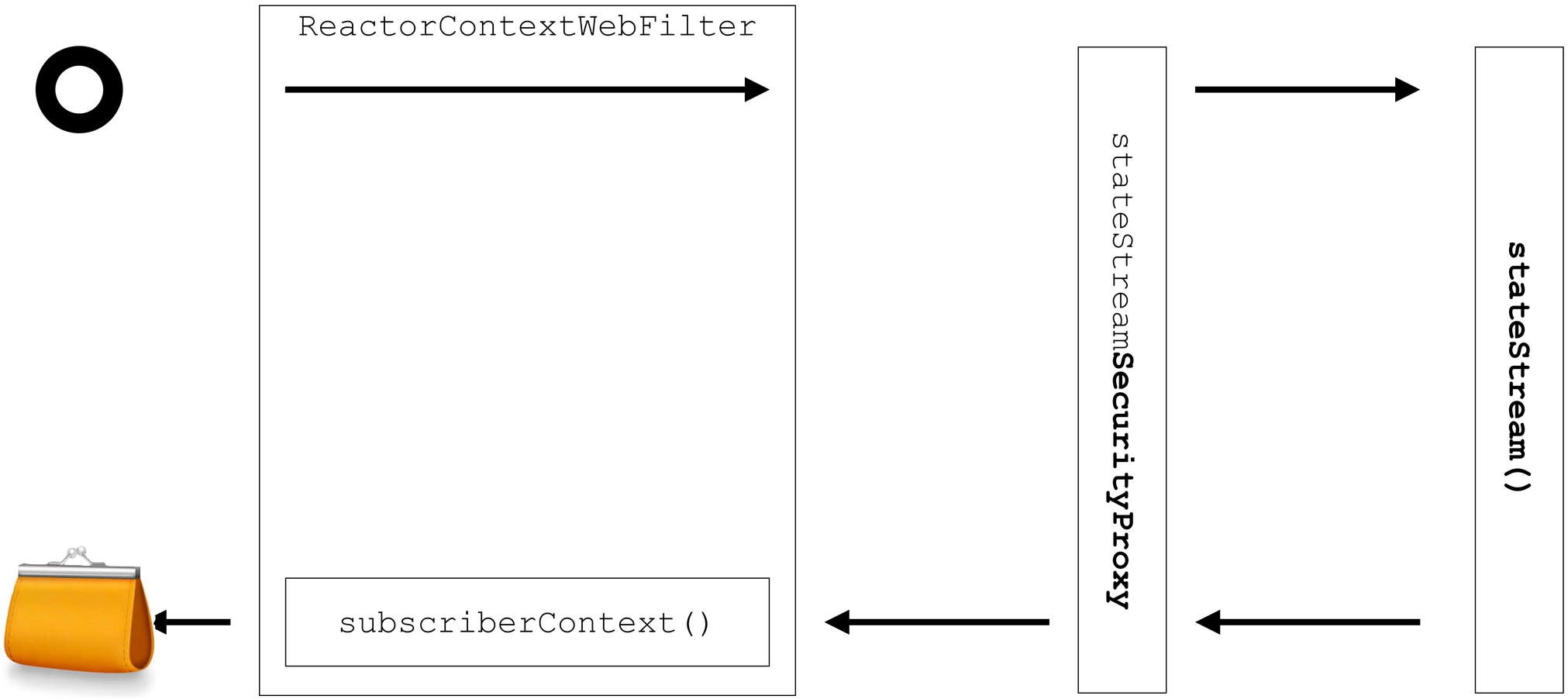
ReactorContextWebFilter
stateStream
stateStream()
subscriberContext()

ReactorContextWebFilter
stateStream**SecurityProxy**
stateStream()
subscriberContext()

ReactorContextWebFilter
stateStream**SecurityProxy**
stateStream()
subscriberContext()


ReactorContextWebFilter
subscriberContext()
stateStreamSecurityPro
stateStream()

ReactorContextWebFilter
stateStreamSecurityProxy
stateStream()
subscriberContext()


ReactorContextWebFilter
stateStreamSecurityProxy
stateStream()
subscriberContext()

```
ReactiveSecurityContextHolder
    .getContext()
    .map(getAuthentication)
    .map(getName);
```

```
ReactiveSecurityContextHolder
    .getContext()
    .map(getAuthentication)
    .map(getName);
```

...

ReactorContextWebFilter…

admin

```
ReactiveSecurityContextHolder
   .getContext()
   .map(getAuthentication)
   .map(getName);
```

...

ReactorContextWebFilter…

Что запомнить!?

Что запомнить!?

- Замыкаем что бы получить `Context`

Что запомнить!?

- Замыкаем что бы получить `Context`
- `Context` задом наперед

Что запомнить!?

- Замыкаем что бы получить `Context`
- `Context` задом наперед
- `SecurityContext` живет в `Context`

Помогает

Помогает

 Работать реактивно с MongoDB

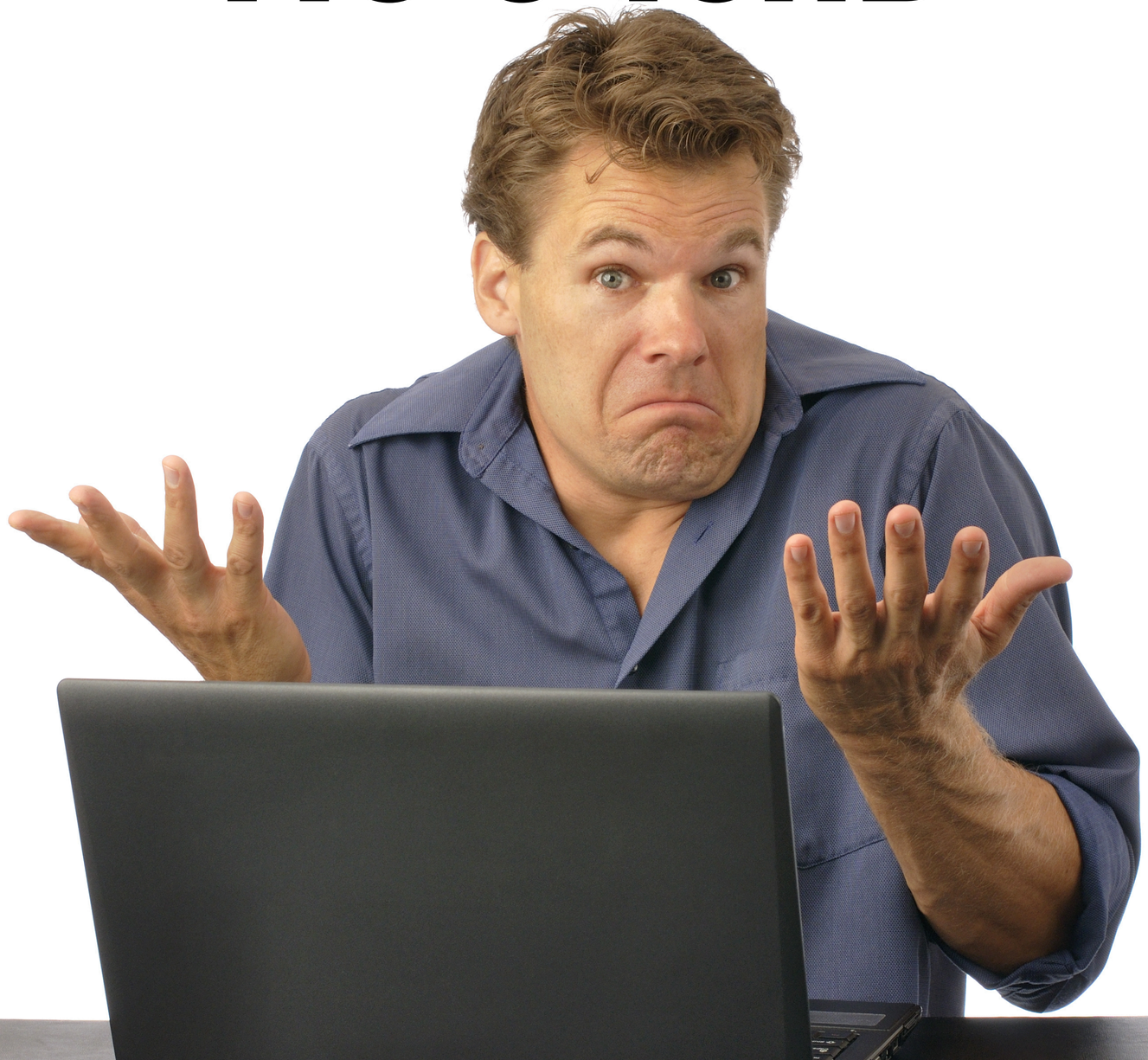
Помогает

 Работать реактивно с MongoDB

 Защищать API просто

Не очень

Не очень



Механика торговли

Что нужно?

- Сохранять результаты торгов в базу

Бизнес механизм

- Снять деньги с кошелька
- Провести торги
- В случае успеха - добавить деньги в кошелек
- В случае неудачных торгов - вернуть деньги
- В случае нехватки средств - ничего не делать

Что измениться?

- `Wallet.java`

```
class Wallet {
  ...
  Wallet withdraw(float amount)
  Wallet adjust(float amount)
}
```

Что измениться?

- `Wallet.java`
- `CryptoService.java`

```
interface CryptoService {
  Flux<Message<?>> stream();
  default Mono<Void> trade(…)
}
```

Что измениться?

- `Wallet.java`
- `CryptoService.java`
- `WalletService.java`

```
interface WalletService {
 Flux<Message<Float>> changesStream();
 Mono<Void> withdraw(Message<Message.Trade> trade);
 Mono<Void> adjust(Message<Message.Trade> trade);
 Mono<Void> rollback(Message<Message.Trade> trade);
}
```

Что создадим?

Что создадим?

- `Trade.java`

Что создадим?

- `Trade.java`
- `TradeRepository.java`

Что создадим?

- `Trade.java`
- `TradeRepository.java`
- `LocalCryptoService.java`

Что создадим?

- Trade.java
- TradeRepository.java
- **LocalCryptoService.java**

Talk is cheap. Show me the code.

- Linus Torvalds

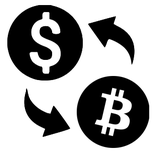
```
tradeOffer
 .onBackpressureBuffer()
 .flatMap(trade ->
        ws.withdraw(trade)
          .then(doTrade(trade))
          .then(ws.adjust(trade))
          .then(doStoreTrade(trade))
          .onErrorResume(
            NotEnoughMoneyException.class,
            t -> Mono.empty()
          )
          .onErrorResume(t ->
            ws.rollback(trade)
              .then(Mono.empty())
          )
 )
 .map(LocalMessageMapper::tradeToMessage)
 .doOnNext(stream.sink()::next)
 .then();
```

```
tradeOffer
  .onBackpressureBuffer()
  .flatMap(trade ->
        ws.withdraw(trade)
          .then(doTrade(trade))
          .then(ws.adjust(trade))
          .then(doStoreTrade(trade))
          .onErrorResume(
            NotEnoughMoneyException.class,
            t -> Mono.empty()
          )
          .onErrorResume(t ->
            ws.rollback(trade)
              .then(Mono.empty())
          )
  )
  .map(LocalMessageMapper::tradeToMessage)
  .doOnNext(stream.sink()::next)
  .then();
```

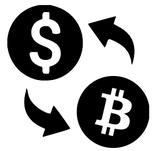
```
tradeOffer
 .onBackpressureBuffer()
 .flatMap(trade ->
        ws.withdraw(trade)
          .then(doTrade(trade))
          .then(ws.adjust(trade))
          .then(doStoreTrade(trade))
          .onErrorResume(
            NotEnoughMoneyException.class,
            t -> Mono.empty()
          )
          .onErrorResume(t ->
            ws.rollback(trade)
              .then(Mono.empty())
          )
 )
 .map(LocalMessageMapper::tradeToMessage)
 .doOnNext(stream.sink()::next)
 .then();
```

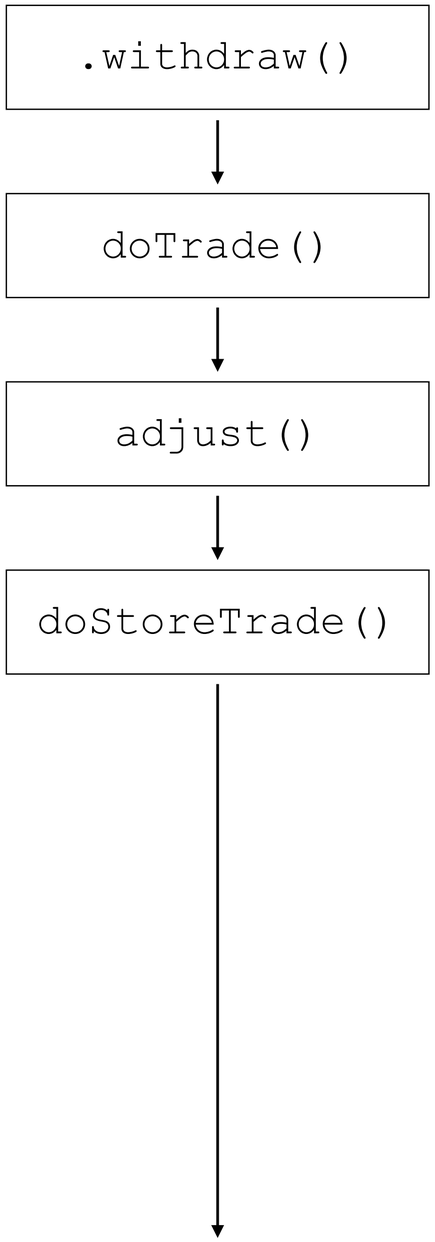
```
tradeOffer
 .onBackpressureBuffer()
 .flatMap(trade ->
        ws.withdraw(trade)
          .then(doTrade(trade))
          .then(ws.adjust(trade))
          .then(doStoreTrade(trade))
          .onErrorResume(
            NotEnoughMoneyException.class,
            t -> Mono.empty()
          )
          .onErrorResume(t ->
            ws.rollback(trade)
              .then(Mono.empty())
          )
 )
 .map(LocalMessageMapper::tradeToMessage)
 .doOnNext(stream.sink()::next)
 .then();
```

```
        .flatMap(trade ->
                ws.withdraw(trade)
                  .then(doTrade(trade))
                  .then(ws.adjust(trade))
                  .then(doStoreTrade(trade))
                  .onErrorResume(
                    NotEnoughMoneyException.class,
                    t -> Mono.empty()
                  )
                  .onErrorResume(t ->
                    ws.rollback(trade)
                      .then(Mono.empty())
                  )
    }
```



```
.flatMap(trade ->
        ws.withdraw(trade)
          .then(doTrade(trade))
          .then(ws.adjust(trade))
          .then(doStoreTrade(trade))
          .onErrorResume(
            NotEnoughMoneyException.class,
            t -> Mono.empty()
          )
          .onErrorResume(t ->
            ws.rollback(trade)
              .then(Mono.empty())
          )
}
```

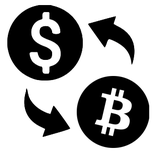





```
.flatMap(trade ->
      ws.withdraw(trade)
        .then(doTrade(trade))
        .then(ws.adjust(trade))
        .then(doStoreTrade(trade))
        .onErrorResume(
          NotEnoughMoneyException.class,
          t -> Mono.empty()
        )
        .onErrorResume(t ->
          ws.rollback(trade)
            .then(Mono.empty())
        )
```

onErrorResume(NEME)

onErrorResume(rollback())



.withdraw()

doTrade()

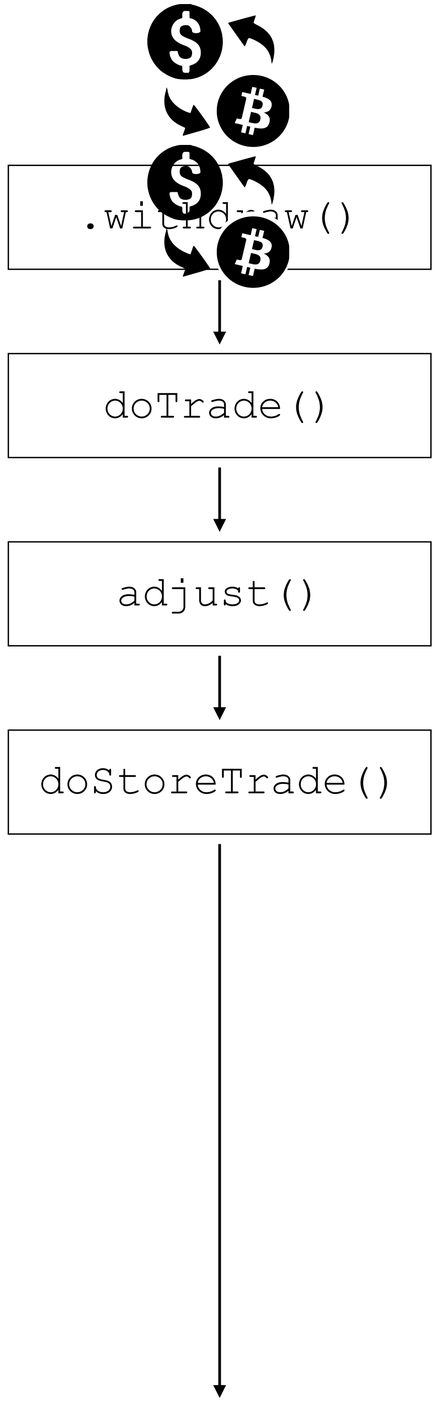
adjust()

doStoreTrade()

onErrorResume(NEME)

onErrorResume(rollback())

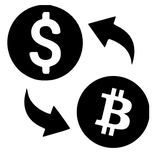
```
.flatMap(trade ->
    ws.withdraw(trade)
      .then(doTrade(trade))
      .then(ws.adjust(trade))
      .then(doStoreTrade(trade))
      .onErrorResume(
        NotEnoughMoneyException.class,
        t -> Mono.empty()
      )
      .onErrorResume(t ->
        ws.rollback(trade)
          .then(Mono.empty())
      )
```




```
.flatMap(trade ->
    ws.withdraw(trade)
      .then(doTrade(trade))
      .then(ws.adjust(trade))
      .then(doStoreTrade(trade))
      .onErrorResume(
        NotEnoughMoneyException.class,
        t -> Mono.empty()
      )
      .onErrorResume(t ->
        ws.rollback(trade)
          .then(Mono.empty())
      )
```

onErrorResume(NEME)

onErrorResume(rollback())



.withdraw()

doTrade()

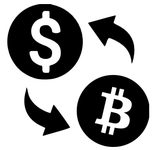
adjust()

doStoreTrade()

onErrorResume(NEME)

onErrorResume(rollback())

```
.flatMap(trade ->
    ws.withdraw(trade)
      .then(doTrade(trade))
      .then(ws.adjust(trade))
      .then(doStoreTrade(trade))
      .onErrorResume(
        NotEnoughMoneyException.class,
        t -> Mono.empty()
      )
      .onErrorResume(t ->
        ws.rollback(trade)
          .then(Mono.empty())
      )
```



.withdraw()

doTrade()

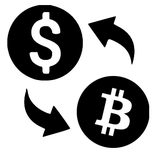
adjust()

doStoreTrade()

onErrorResume(NEME)

onErrorResume(rollback())

```
.flatMap(trade ->
    ws.withdraw(trade)
      .then(doTrade(trade))
      .then(ws.adjust(trade))
      .then(doStoreTrade(trade))
      .onErrorResume(
        NotEnoughMoneyException.class,
        t -> Mono.empty()
      )
      .onErrorResume(t ->
        ws.rollback(trade)
          .then(Mono.empty())
      )
```



.withdraw()

doTrade()

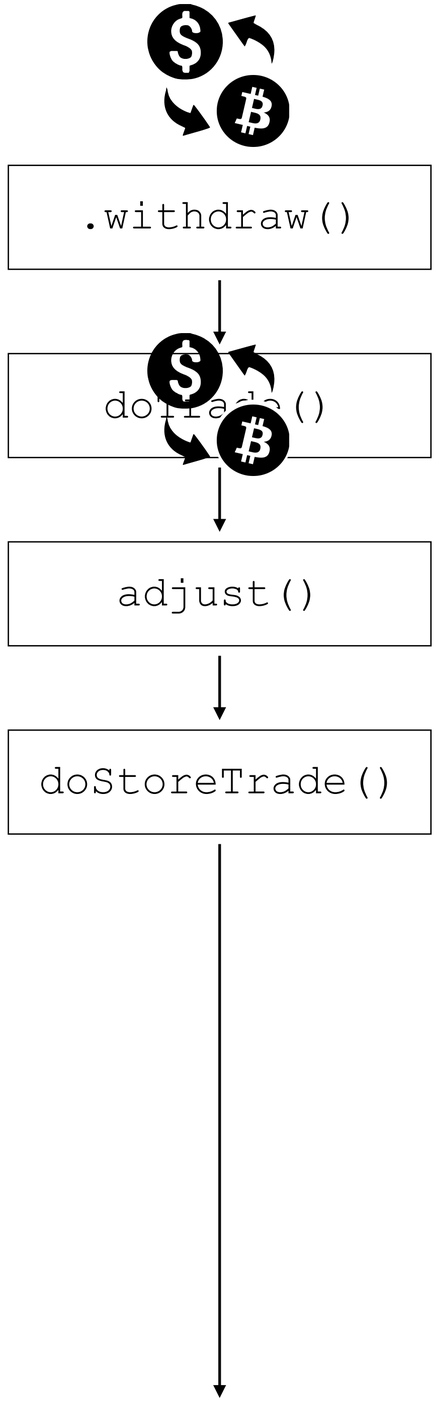
adjust()

doStoreTrade()

onErrorResume(NEME)

onErrorResume(rollback())

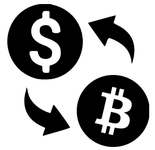
```
.flatMap(trade ->
    ws.withdraw(trade)
      .then(doTrade(trade))
      .then(ws.adjust(trade))
      .then(doStoreTrade(trade))
      .onErrorResume(
        NotEnoughMoneyException.class,
        t -> Mono.empty()
      )
      .onErrorResume(t ->
        ws.rollback(trade)
          .then(Mono.empty())
      )
```




```
.flatMap(trade ->
    ws.withdraw(trade)
      .then(doTrade(trade))
      .then(ws.adjust(trade))
      .then(doStoreTrade(trade))
      .onErrorResume(
        NotEnoughMoneyException.class,
        t -> Mono.empty()
      )
      .onErrorResume(t ->
        ws.rollback(trade)
          .then(Mono.empty())
      )
```

onErrorResume(NEME)

onErrorResume(rollback())



.withdraw()

doTrade()

adjust()

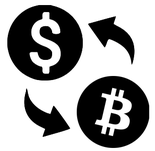
doStoreTrade()

onErrorResume(NEME)

onErrorResume(rollback())

```
.flatMap(trade ->
    ws.withdraw(trade)
      .then(doTrade(trade))
      .then(ws.adjust(trade))
      .then(doStoreTrade(trade))
      .onErrorResume(
        NotEnoughMoneyException.class,
        t -> Mono.empty()
      )
      .onErrorResume(t ->
        ws.rollback(trade)
          .then(Mono.empty())
      )
```


bellecs



.withdraw()

doTrade()

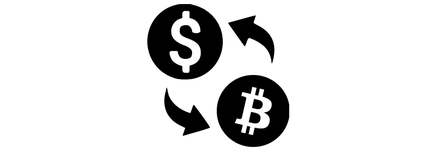
adjust()

doStoreTrade()

onErrorResume(NEME)

onErrorResume(rollback())

```
.flatMap(trade ->
    ws.withdraw(trade)
      .then(doTrade(trade))
      .then(ws.adjust(trade))
      .then(doStoreTrade(trade))
      .onErrorResume(
        NotEnoughMoneyException.class,
        t -> Mono.empty()
      )
      .onErrorResume(t ->
        ws.rollback(trade)
          .then(Mono.empty())
      )
```



.withdraw()

doTrade()

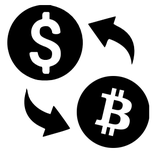
adjust()

doStoreTrade()

onErrorResume(NEME)

onErrorResume(rollback())

```
.flatMap(trade ->
    ws.withdraw(trade)
      .then(doTrade(trade))
      .then(ws.adjust(trade))
      .then(doStoreTrade(trade))
      .onErrorResume(
        NotEnoughMoneyException.class,
        t -> Mono.empty()
      )
      .onErrorResume(t ->
        ws.rollback(trade)
          .then(Mono.empty())
      )
```



.withdraw()

doTrade()

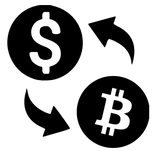
adjust()

doStoreTrade()

onErrorResume(NEME)

onErrorResume(rollback())

```
.flatMap(trade ->
    ws.withdraw(trade)
      .then(doTrade(trade))
      .then(ws.adjust(trade))
      .then(doStoreTrade(trade))
      .onErrorResume(
        NotEnoughMoneyException.class,
        t -> Mono.empty()
      )
      .onErrorResume(t ->
        ws.rollback(trade)
          .then(Mono.empty())
      )
```



.withdraw()

doTrade()

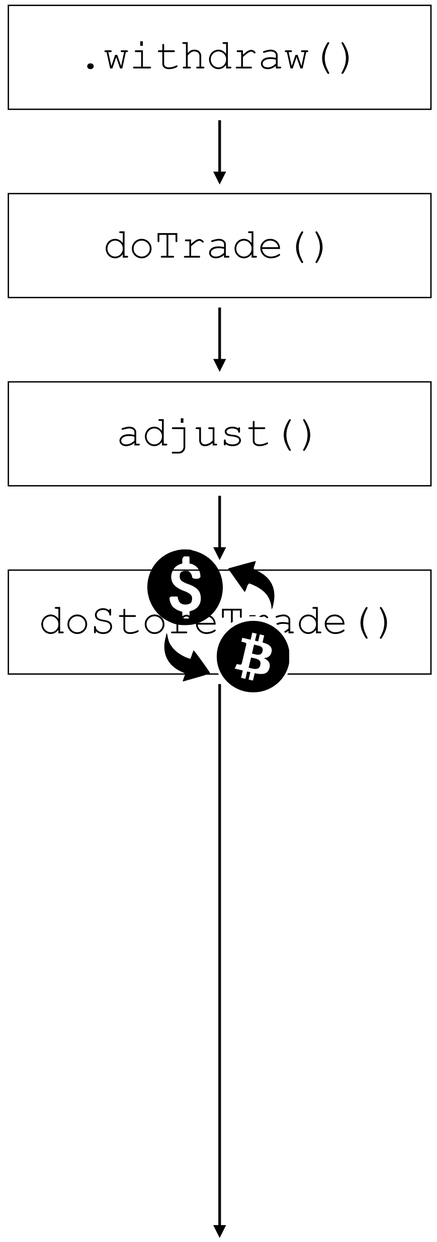
adjust()

doStoreTrade()

onErrorResume(NEME)

onErrorResume(rollback())

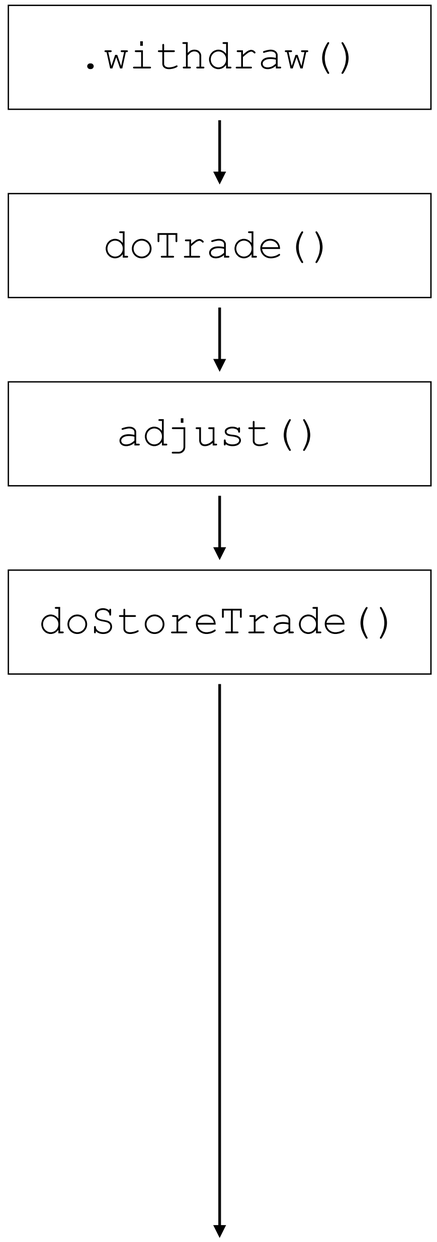
```
.flatMap(trade ->
    ws.withdraw(trade)
      .then(doTrade(trade))
      .then(ws.adjust(trade))
      .then(doStoreTrade(trade))
      .onErrorResume(
        NotEnoughMoneyException.class,
        t -> Mono.empty()
      )
      .onErrorResume(t ->
        ws.rollback(trade)
          .then(Mono.empty())
      )
```




```
.flatMap(trade ->
    ws.withdraw(trade)
      .then(doTrade(trade))
      .then(ws.adjust(trade))
      .then(doStoreTrade(trade))
      .onErrorResume(
        NotEnoughMoneyException.class,
        t -> Mono.empty()
      )
      .onErrorResume(t ->
        ws.rollback(trade)
          .then(Mono.empty())
      )
```

onErrorResume(NEME)

onErrorResume(rollback())

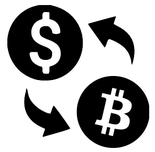



```
.flatMap(trade ->
      ws.withdraw(trade)
        .then(doTrade(trade))
        .then(ws.adjust(trade))
        .then(doStoreTrade(trade))
        .onErrorResume(
          NotEnoughMoneyException.class,
          t -> Mono.empty()
        )
        .onErrorResume(t ->
          ws.rollback(trade)
            .then(Mono.empty())
        )
```

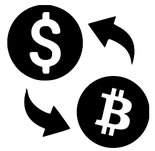
onErrorResume(NEME)

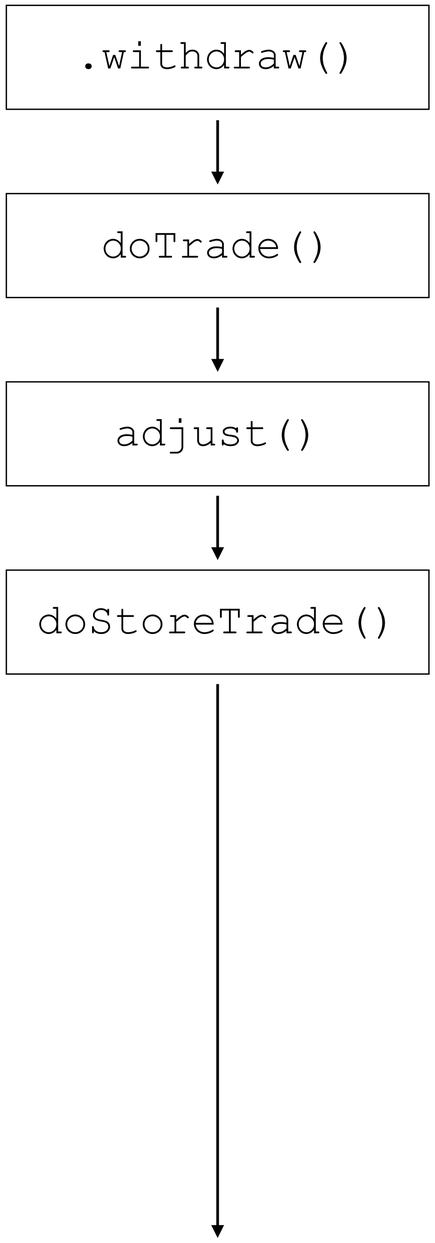
onErrorResume(rollback())

```
        .flatMap(trade ->
                ws.withdraw(trade)
                  .then(doTrade(trade))
                  .then(ws.adjust(trade))
                  .then(doStoreTrade(trade))
                  .onErrorResume(
                    NotEnoughMoneyException.class,
                    t -> Mono.empty()
                  )
                  .onErrorResume(t ->
                    ws.rollback(trade)
                      .then(Mono.empty())
                  )
    }
```



```
.flatMap(trade ->
        ws.withdraw(trade)
          .then(doTrade(trade))
          .then(ws.adjust(trade))
          .then(doStoreTrade(trade))
          .onErrorResume(
            NotEnoughMoneyException.class,
            t -> Mono.empty()
          )
          .onErrorResume(t ->
            ws.rollback(trade)
              .then(Mono.empty())
          )
}
```

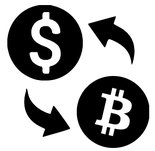


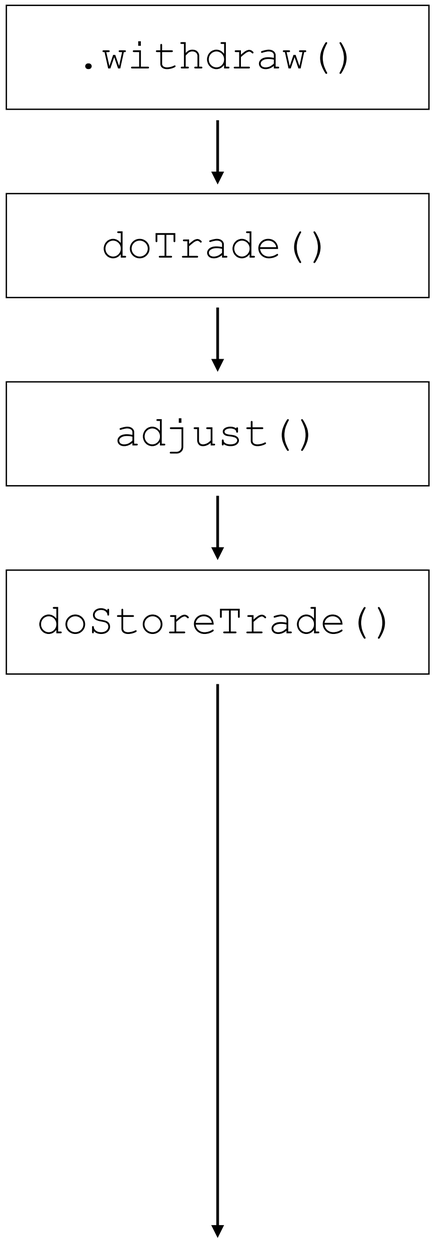



```
.flatMap(trade ->
      ws.withdraw(trade)
        .then(doTrade(trade))
        .then(ws.adjust(trade))
        .then(doStoreTrade(trade))
        .onErrorResume(
          NotEnoughMoneyException.class,
          t -> Mono.empty()
        )
        .onErrorResume(t ->
          ws.rollback(trade)
            .then(Mono.empty())
        )
```

onErrorResume(NEME)

onErrorResume(rollback())

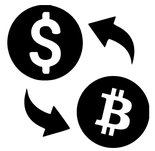





```
.flatMap(trade ->
      ws.withdraw(trade)
        .then(doTrade(trade))
        .then(ws.adjust(trade))
        .then(doStoreTrade(trade))
        .onErrorResume(
          NotEnoughMoneyException.class,
          t -> Mono.empty()
        )
        .onErrorResume(t ->
          ws.rollback(trade)
            .then(Mono.empty())
        )
```

onErrorResume(NEME)

onErrorResume(rollback())



.withdraw()

doTrade()

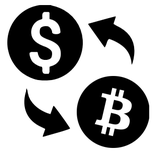
adjust()

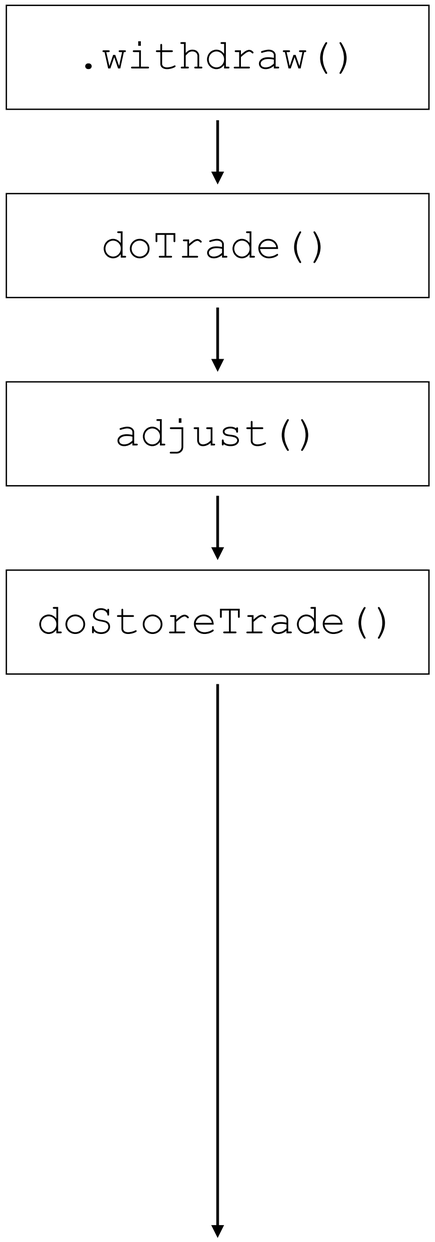
doStoreTrade()

onErrorResume(NEME)

onErrorResume(rollback())

```
.flatMap(trade ->
    ws.withdraw(trade)
      .then(doTrade(trade))
      .then(ws.adjust(trade))
      .then(doStoreTrade(trade))
      .onErrorResume(
        NotEnoughMoneyException.class,
        t -> Mono.empty()
      )
      .onErrorResume(t ->
        ws.rollback(trade)
          .then(Mono.empty())
      )
```

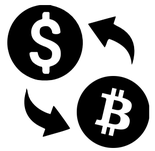





```
.flatMap(trade ->
    ws.withdraw(trade)
      .then(doTrade(trade))
      .then(ws.adjust(trade))
      .then(doStoreTrade(trade))
      .onErrorResume(
        NotEnoughMoneyException.class,
        t -> Mono.empty()
      )
      .onErrorResume(t ->
        ws.rollback(trade)
          .then(Mono.empty())
      )
```

onErrorResume(NEME)

onErrorResume(rollback())



.withdraw()

doTrade()

adjust()

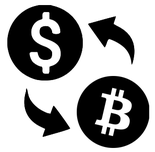
doStoreTrade()

onError e(NEME)

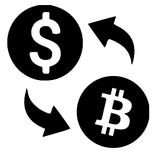
onErrorResume(rollback())

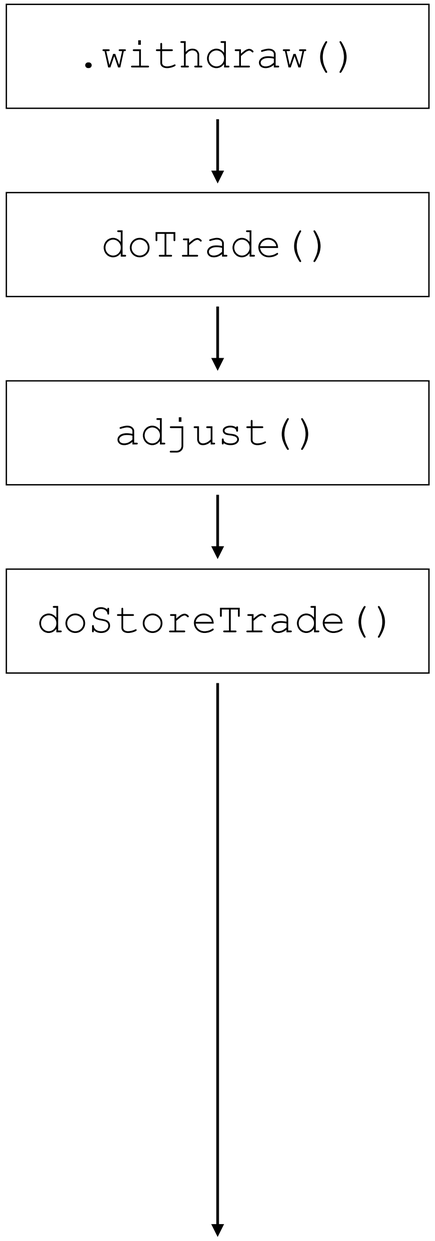
```
.flatMap(trade ->
    ws.withdraw(trade)
      .then(doTrade(trade))
      .then(ws.adjust(trade))
      .then(doStoreTrade(trade))
      .onErrorResume(
        NotEnoughMoneyException.class,
        t -> Mono.empty()
      )
      .onErrorResume(t ->
        ws.rollback(trade)
          .then(Mono.empty())
      )
```

```
        .flatMap(trade ->
                ws.withdraw(trade)
                  .then(doTrade(trade))
                  .then(ws.adjust(trade))
                  .then(doStoreTrade(trade))
                  .onErrorResume(
                    NotEnoughMoneyException.class,
                    t -> Mono.empty()
                  )
                  .onErrorResume(t ->
                    ws.rollback(trade)
                      .then(Mono.empty())
                  )
    }
```



```
.flatMap(trade ->
        ws.withdraw(trade)
          .then(doTrade(trade))
          .then(ws.adjust(trade))
          .then(doStoreTrade(trade))
          .onErrorResume(
            NotEnoughMoneyException.class,
            t -> Mono.empty()
          )
          .onErrorResume(t ->
            ws.rollback(trade)
              .then(Mono.empty())
          )
}
```

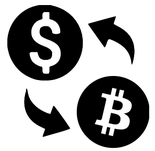





```
.flatMap(trade ->
      ws.withdraw(trade)
        .then(doTrade(trade))
        .then(ws.adjust(trade))
        .then(doStoreTrade(trade))
        .onErrorResume(
          NotEnoughMoneyException.class,
          t -> Mono.empty()
        )
        .onErrorResume(t ->
          ws.rollback(trade)
            .then(Mono.empty())
        )
```

onErrorResume(NEME)

onErrorResume(rollback())



.withdraw()

doTrade()

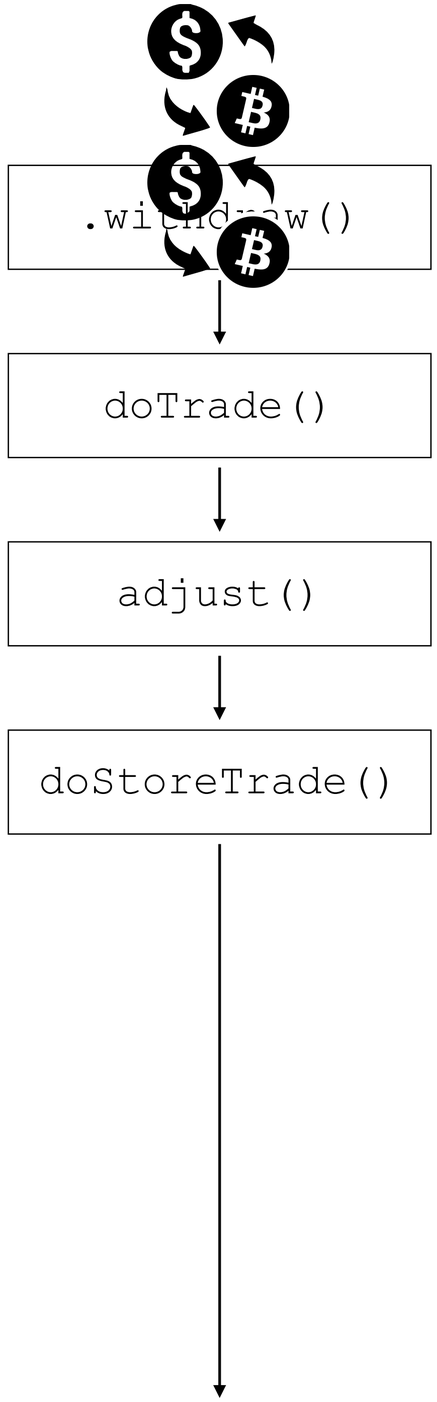
adjust()

doStoreTrade()

onErrorResume(NEME)

onErrorResume(rollback())

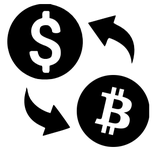
```
.flatMap(trade ->
      ws.withdraw(trade)
        .then(doTrade(trade))
        .then(ws.adjust(trade))
        .then(doStoreTrade(trade))
        .onErrorResume(
          NotEnoughMoneyException.class,
          t -> Mono.empty()
        )
        .onErrorResume(t ->
          ws.rollback(trade)
            .then(Mono.empty())
        )
```




```
.flatMap(trade ->
    ws.withdraw(trade)
      .then(doTrade(trade))
      .then(ws.adjust(trade))
      .then(doStoreTrade(trade))
      .onErrorResume(
        NotEnoughMoneyException.class,
        t -> Mono.empty()
      )
      .onErrorResume(t ->
        ws.rollback(trade)
          .then(Mono.empty())
      )
```

onErrorResume(NEME)

onErrorResume(rollback())



.withdraw()

doTrade()

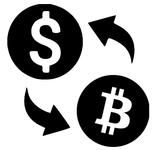
adjust()

doStoreTrade()

onErrorResume(NEME)

onErrorResume(rollback())

```
.flatMap(trade ->
    ws.withdraw(trade)
      .then(doTrade(trade))
      .then(ws.adjust(trade))
      .then(doStoreTrade(trade))
      .onErrorResume(
        NotEnoughMoneyException.class,
        t -> Mono.empty()
      )
      .onErrorResume(t ->
        ws.rollback(trade)
          .then(Mono.empty())
      )
```



.withdraw()

doTrade()

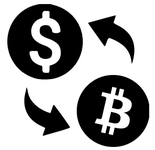
adjust()

doStoreTrade()

onErrorResume(NEME)

onErrorResume(rollback())

```
.flatMap(trade ->
    ws.withdraw(trade)
      .then(doTrade(trade))
      .then(ws.adjust(trade))
      .then(doStoreTrade(trade))
      .onErrorResume(
        NotEnoughMoneyException.class,
        t -> Mono.empty()
      )
      .onErrorResume(t ->
        ws.rollback(trade)
          .then(Mono.empty())
      )
```



.withdraw()

doTrade()

adjust()

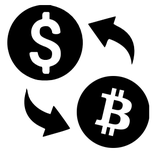
doStoreTrade()

onErrorResume(NEME)

onErrorResume(rollback())

```
.flatMap(trade ->
    ws.withdraw(trade)
      .then(doTrade(trade))
      .then(ws.adjust(trade))
      .then(doStoreTrade(trade))
      .onErrorResume(
        NotEnoughMoneyException.class,
        t -> Mono.empty()
      )
      .onErrorResume(t ->
        ws.rollback(trade)
          .then(Mono.empty())
      )
```


You get nothing!



.withdraw()

doTrade()

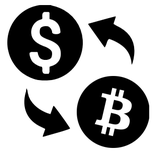
adjust()

doStoreTrade()

onErrorResume(NEME)

onErrorResume(rollback())

```
.flatMap(trade ->
    ws.withdraw(trade)
      .then(doTrade(trade))
      .then(ws.adjust(trade))
      .then(doStoreTrade(trade))
      .onErrorResume(
        NotEnoughMoneyException.class,
        t -> Mono.empty()
      )
      .onErrorResume(t ->
        ws.rollback(trade)
          .then(Mono.empty())
      )
```



.withdraw()

doTrade()

adjust()

doStoreTrade()

onErrorResume(NEME)

onErrorRes ollback())

```
.flatMap(trade ->
    ws.withdraw(trade)
      .then(doTrade(trade))
      .then(ws.adjust(trade))
      .then(doStoreTrade(trade))
      .onErrorResume(
        NotEnoughMoneyException.class,
        t -> Mono.empty()
      )
      .onErrorResume(t ->
        ws.rollback(trade)
          .then(Mono.empty())
      )
```

Что запомнить!?

Что запомнить!?

- `.onErrorXXX` для ошибок

Что запомнить!?

- `.onErrorXXX` для ошибок
- `.then` для последовательности операций

Что запомнить!?

- `.onErrorXXX` для ошибок
- `.then` для последовательности операций
- `Request/Session scope` - своими руками

Помогает

Помогает


Работать с ошибками

Помогает


Работать с ошибками


Строить сложное - просто

Не очень

Не очень

Конфигурировать `Request/Session scope`

А как же отказоустойчивость?

Что нужно?

Что нужно?

- Стабильно работать в случае отказа внешних сервисов

Что нужно?

- Стабильно работать в случае отказа внешних сервисов
- Стабильно работать в случае не авторизованного доступа

Talk is cheap. Show me the code.

- Linus Torvalds

Что запомнить!?

Что запомнить!?

- Изолируем отдельные компоненты

Что запомнить!?

- Изолируем отдельные компоненты
- `.timeout` – чтобы долго не ждать

Что запомнить!?

- Изолируем отдельные компоненты
- `.timeout` – чтобы долго не ждать
- `.retryWhen` – для повторных попыток

Помогает

Помогает


Изолировать компоненты

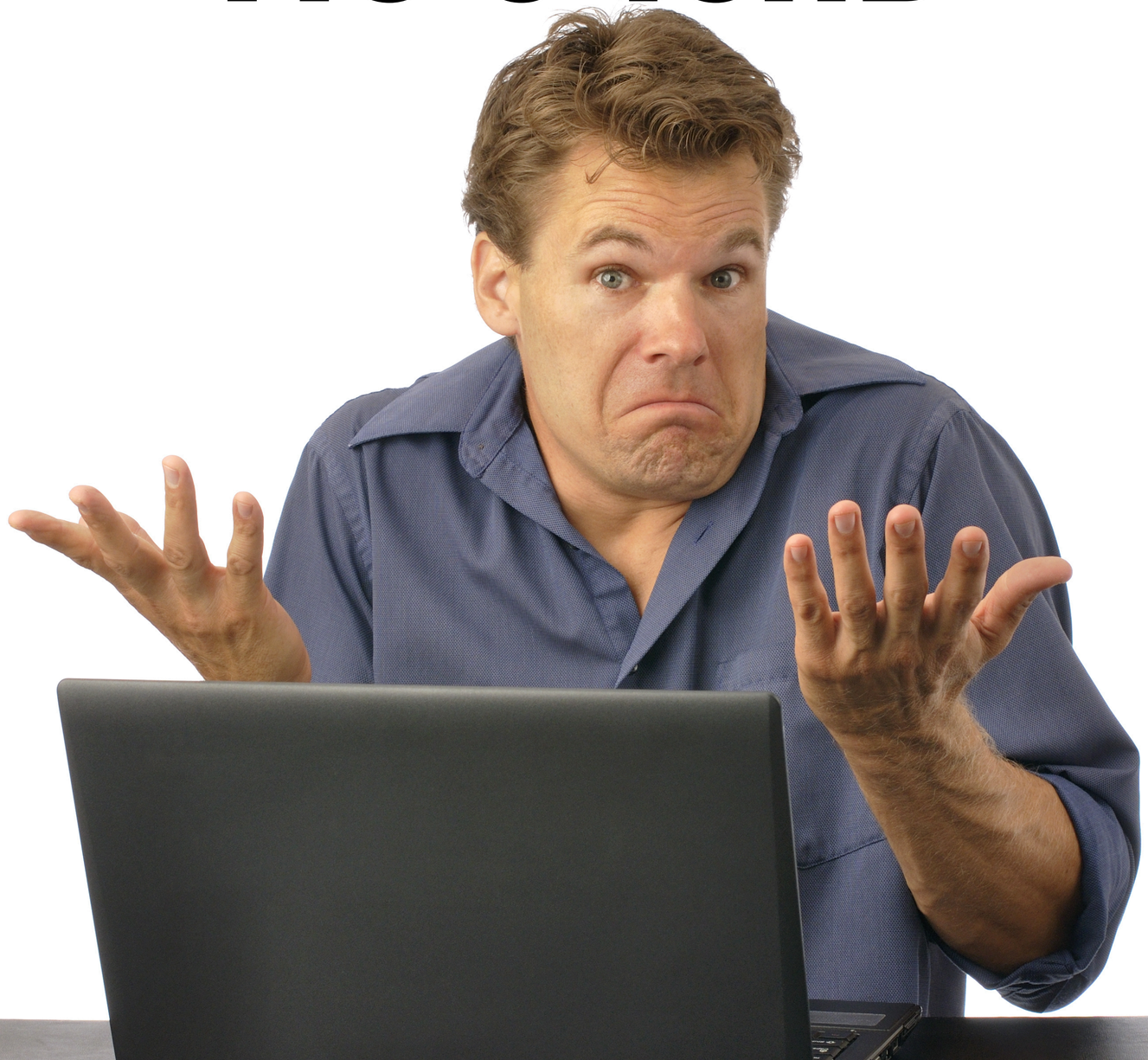
Помогает


Изолировать компоненты

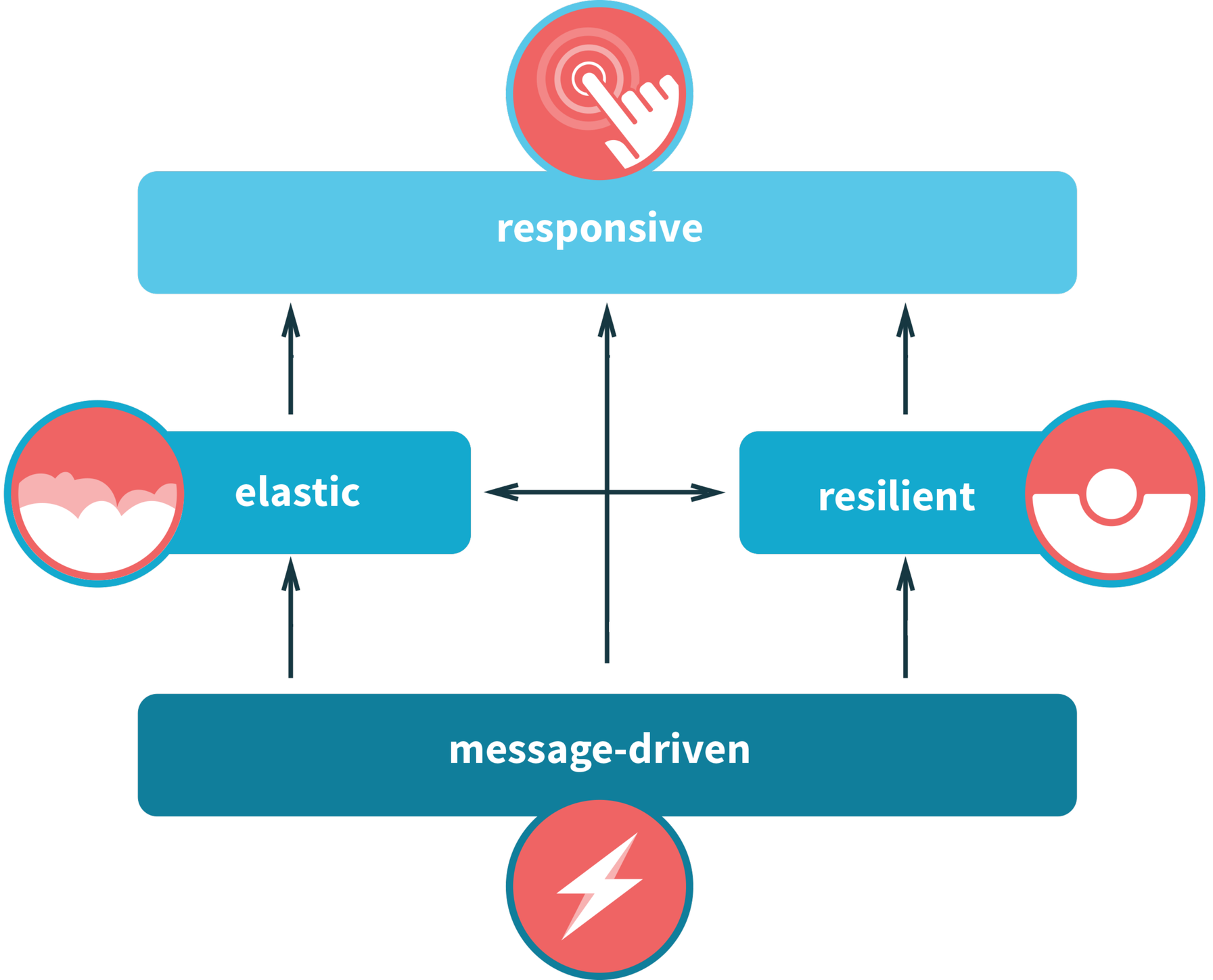

Работать с переподключением

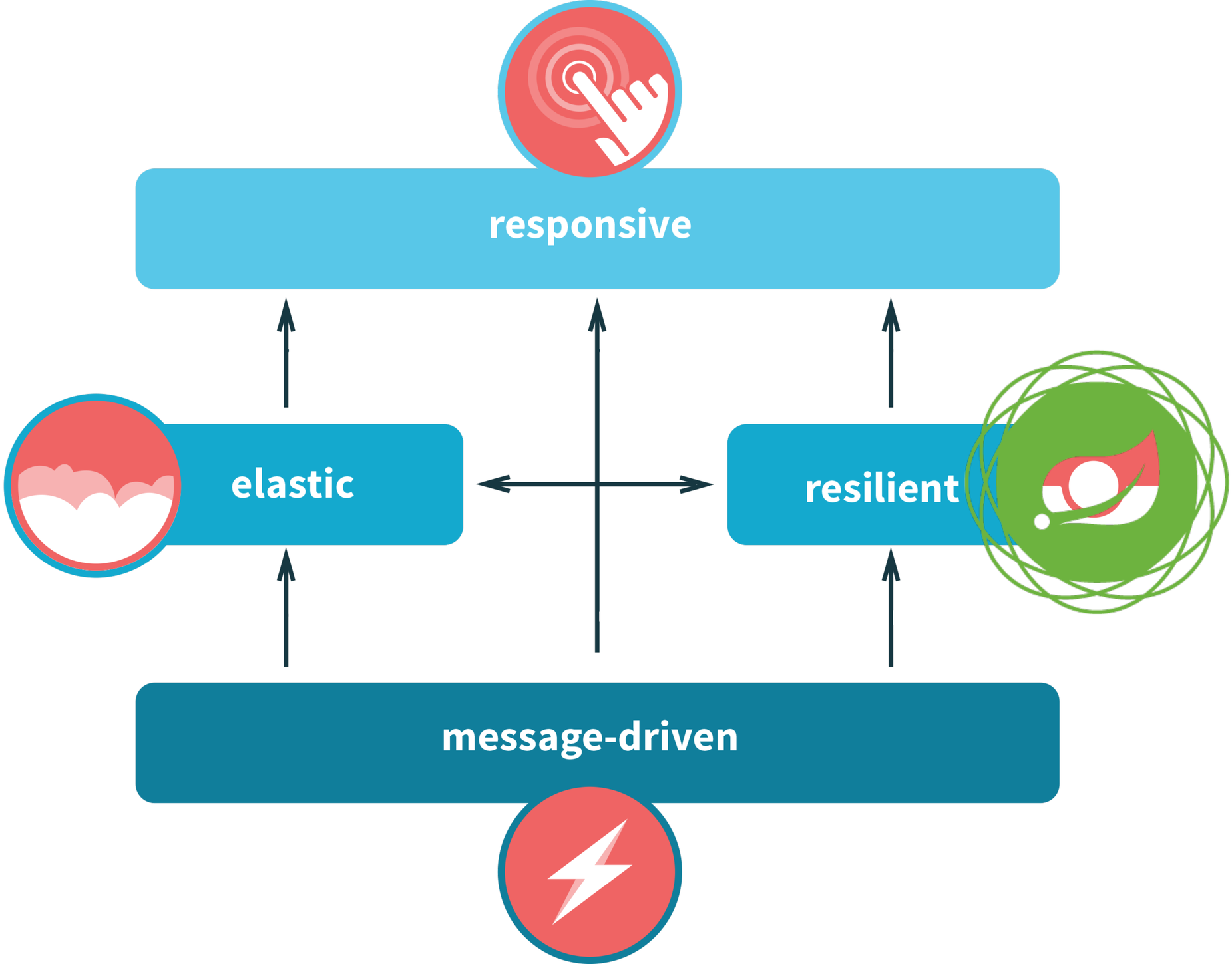
Не очень

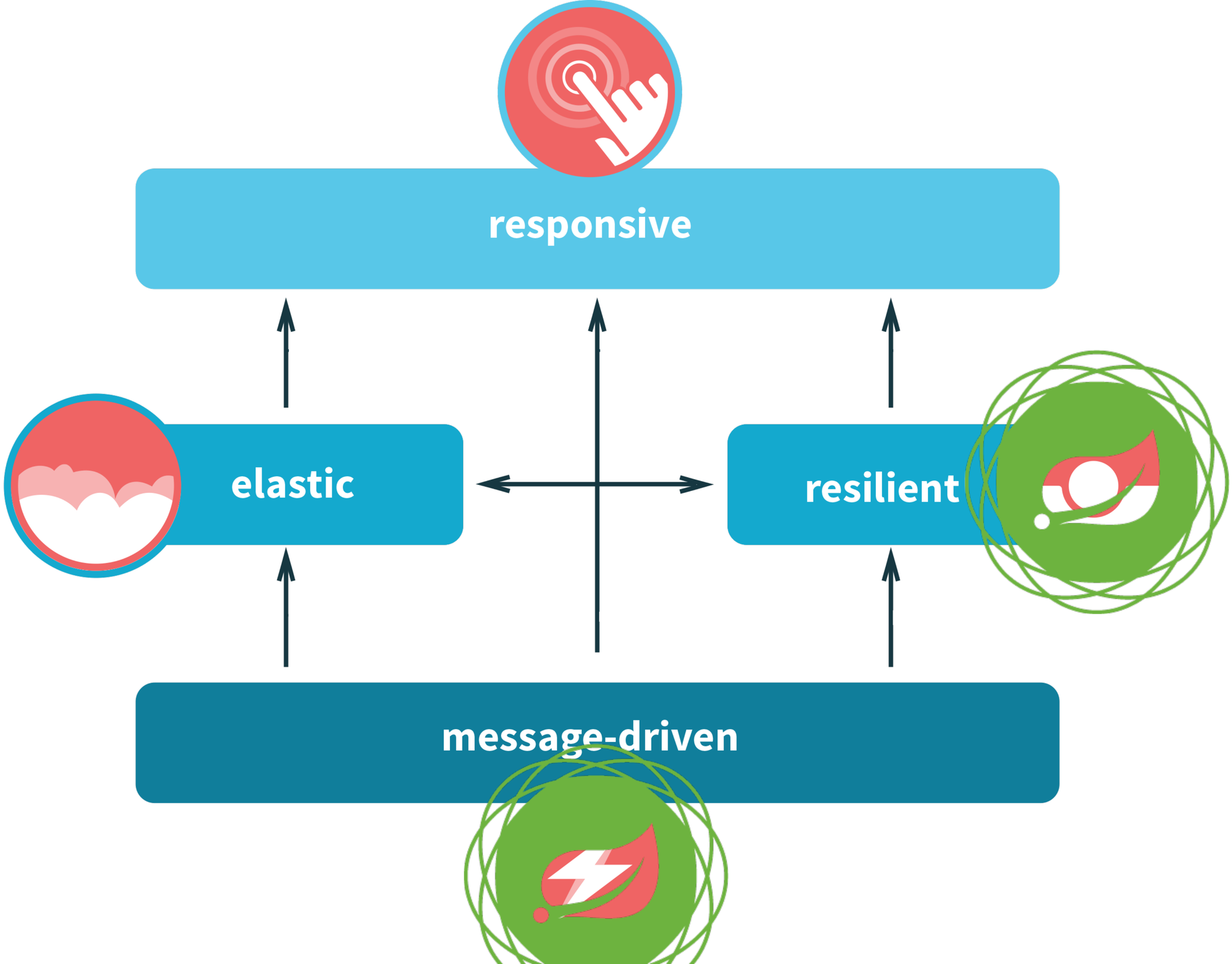
Не очень

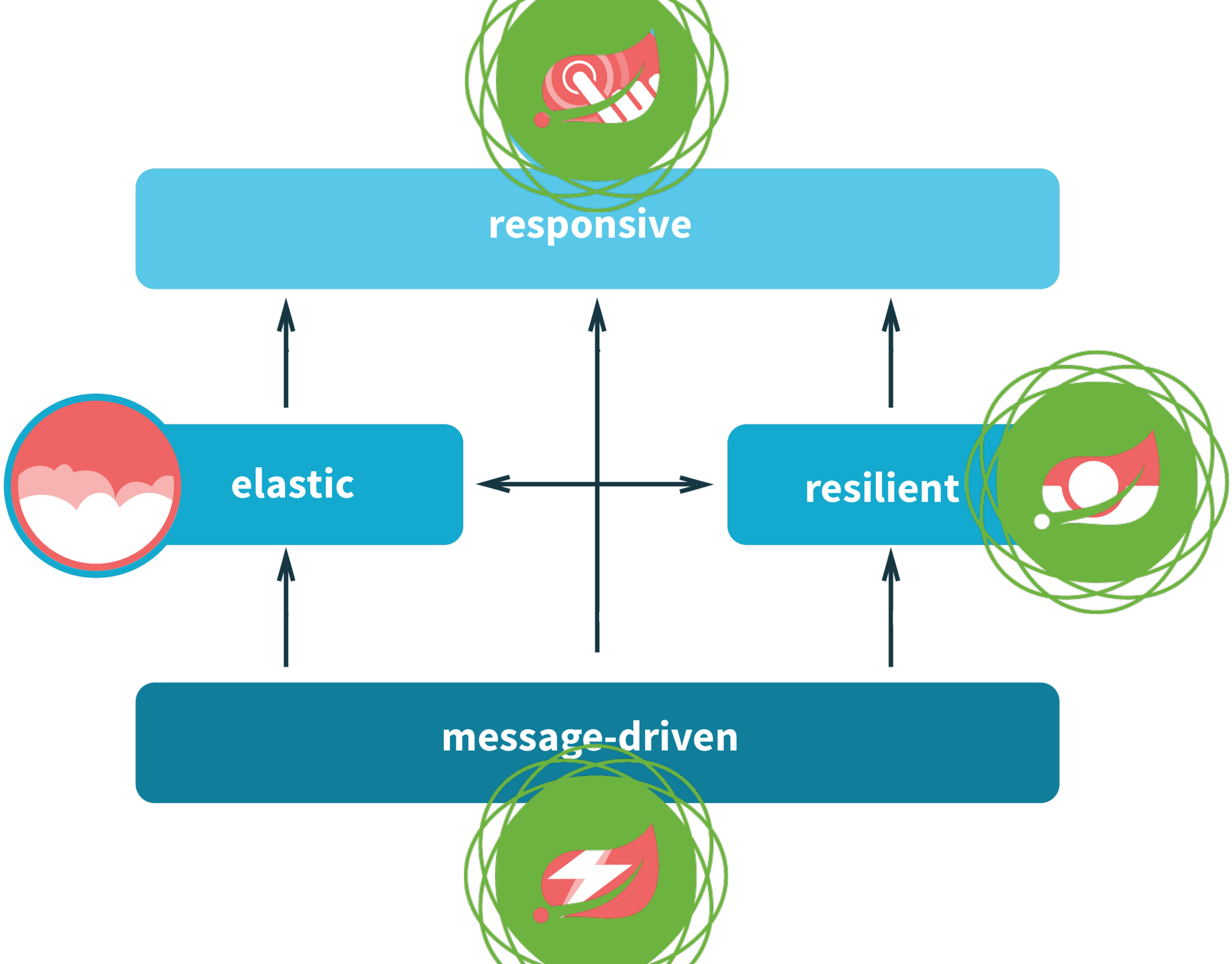


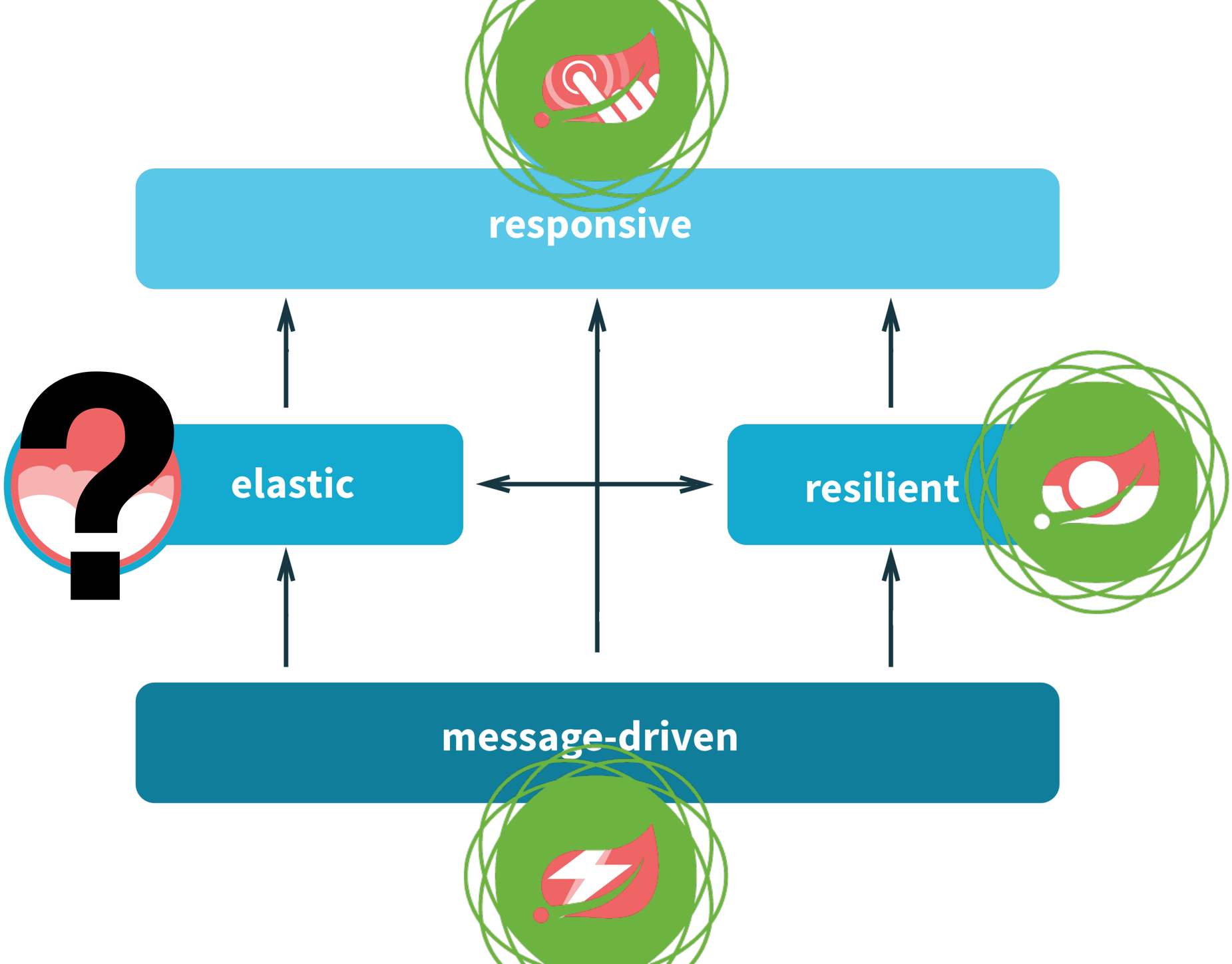
Итожим


responsive
elastic
resilient
message-driven


responsive
elastic
resilient
message-driven


responsive
elastic
resilient
message-driven

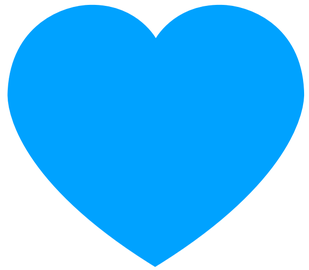

responsive
elastic
resilient
message-driven


responsive
elastic
resilient
message-driven

С чем уйти?

WebFlux - тот же WebMVC но работает с Netty

WebFlux + Reactor



Чистый асинхронный код

Ивентам - **Event-Loop**

Бизнес логике - **отдельный Thread Pool**

Экономь сокеты

- `.publish()` данные

Реактивная интеграция с Базой с - **Spring Data Reactive**

Замыкаем потоки для общего Context

Грустим (пока что) изза Request/Session Scopes

Отлавливаем ошибки с -

.onErrorXXX

Остаемся непоколебимыми с -

`.retryWhen`

Экономим время с -

.timeout

Что еще упустили?

Что еще упустили?

- Дебажить сложно - но **просто тестить** с `StepVerifier`

Что еще упустили?

- Дебажить сложно - но **просто тестить** с `StepVerifier`
- Нет реактивного JDBC (пока что)

Что еще упустили?

- Дебажить сложно - но **просто тестить** с `StepVerifier`
- Нет реактивного JDBC (пока что)
- Императивный код никто не отменял

Где применять?

Где применять?

- Системы с высокой нагрузкой

Где применять?

- Системы с высокой нагрузкой
- Там где нужно экономить на железе

Где применять?

- Системы с высокой нагрузкой
- Там где нужно экономить на железе
- Microservices orchestration

Код живет тут

https://goo.gl/JZHhrq

На посмотреть

- **WebFlux** воркшоп - *https://bclozel.github.io/webflux-workshop/*
- Играем с **Reactor 3** - *https://tech.io/playgrounds/929/reactive-programming-with-reactor-3/Intro*
- **JDBC**? Смотреть тут - *https://www.youtube.com/watch?v=0iXu05WU7zI*
- Все еще не уверены? - **Netflix** вещает - *https://goo.gl/dMBUC7*

Вопросы

Олег Докука

 /OlehDokuka  /oleh.dokuka  /OlegDokuka