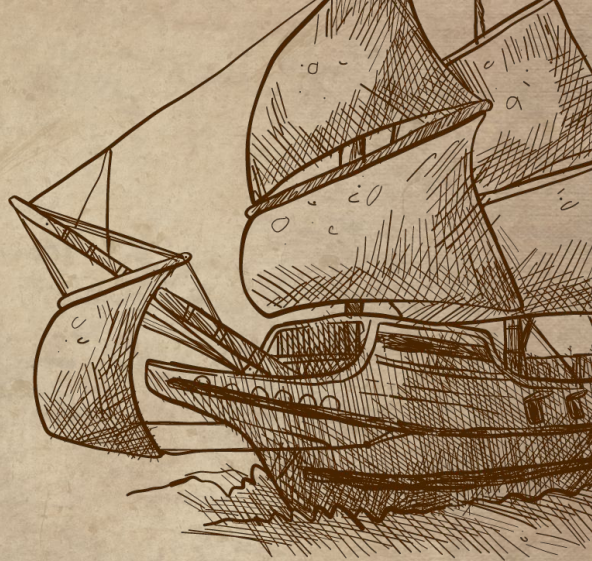
# FUN FACTS FOR EARLY BIRDS

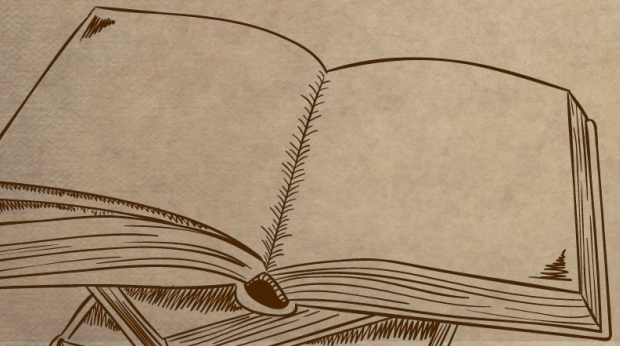
Did you know that:

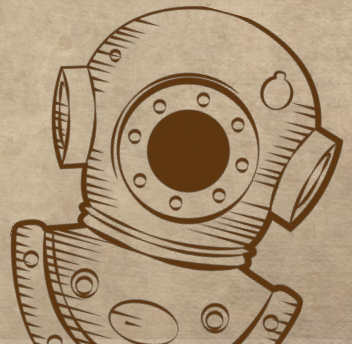
1. The word "**algorithm**" comes from the name of the Persian mathematician Muhammad ibn Musa **al-Khwarizmi**, who lived in the 9th century. His book on decimal numbers was translated into Latin under the title **Algoritmi de Numero Indorum** (meaning "Al-Khwarizmi on Indian numbers"). The title was misinterpreted to mean "algorithms on Indian numbers".
2. The word **Chess** (as well as **check**) come from the Persian word **shah** which means "king". **Checkmate** means "the king is dead".
3. Not all infinities are the same. There are more real numbers that natural numbers, but natural numbers and rational numbers have the same cardinality.
4. The word **tulip** comes from the Persian word **dulband** meaning "turban", a headwrap worn by men.
5. The word **magic** comes **magh**, a Persian word for the priests of Zoroastrianism, an Iranian religion practiced since 16th century BCE.
6. Speaking of Zoroastrianism, the word **Paradise** comes from Old Iranian word **paridaiza**, meaning "enclosed garden". This word was loaned in Greek as **paradeisos** and was used by Xenophon and others for an orchard or royal hunting park in Persia, and it was taken in Septuagint to mean "the garden of Eden," and in New Testament translations of Luke xxiii.43 to mean "the Christian heaven, place where the souls of the righteous departed await resurrection".
7. Al-Khwarizmi has another book called **al-mukhtasar fi hisab al-jabr wa al-muqabala** meaning "the compendium on calculation by restoring and balancing". **Al-jabr** "restoring" became **algebra** in Europe.

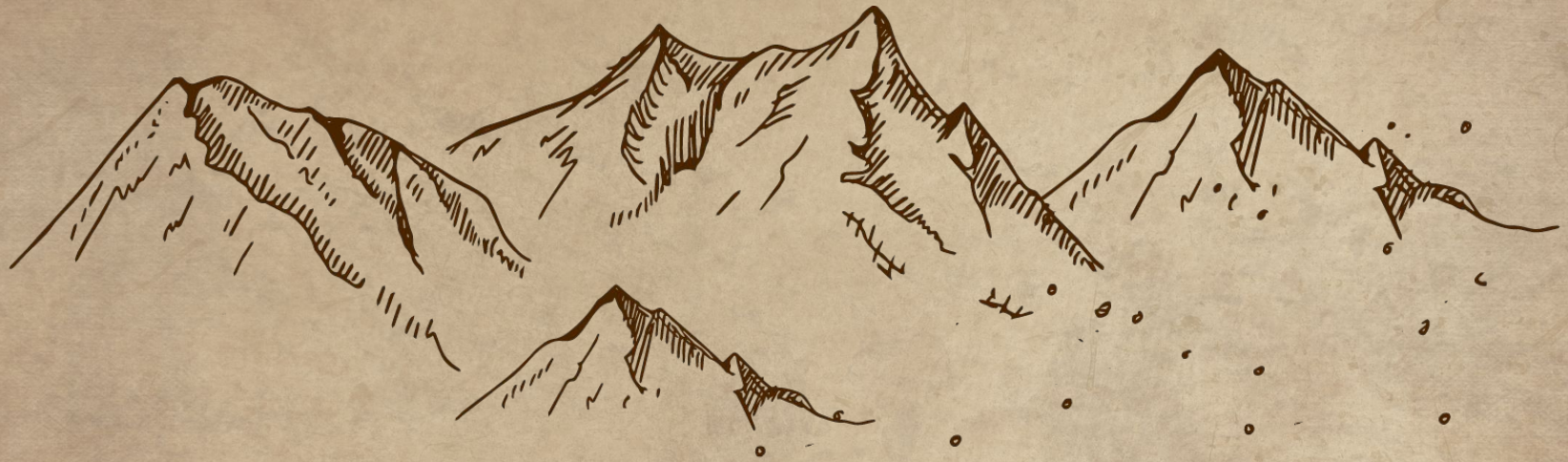


# How to Create RxJS Operators

Mohammad-Ali A'râbi



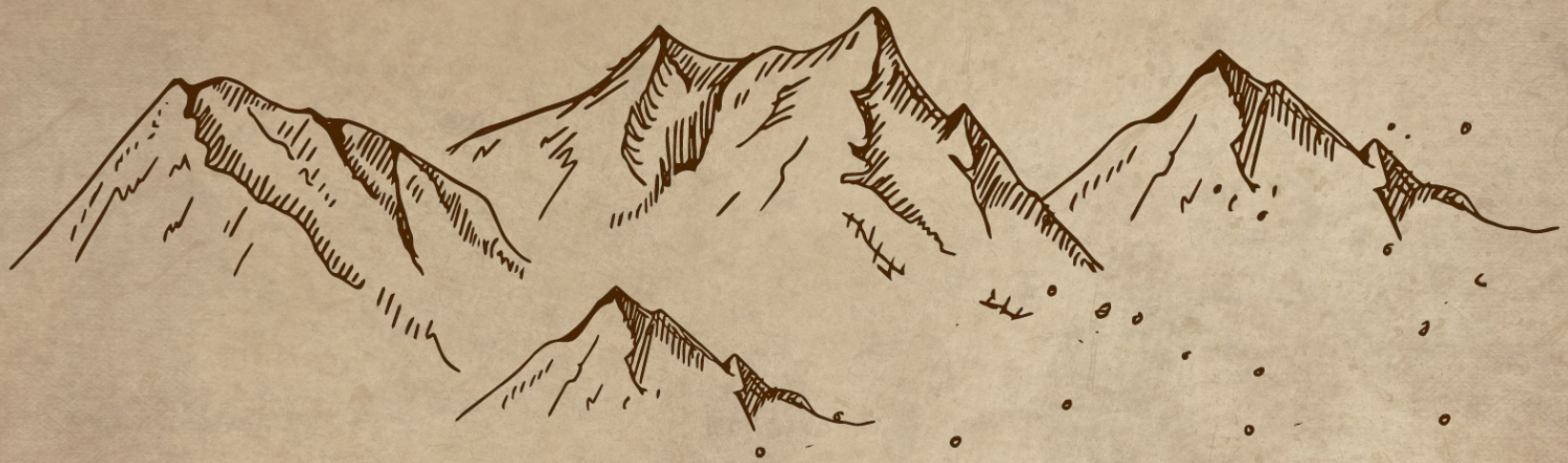




01 Rx

02 RxJS

03 RxJSx



01 Rx

02 RxJS

03 RxJSx

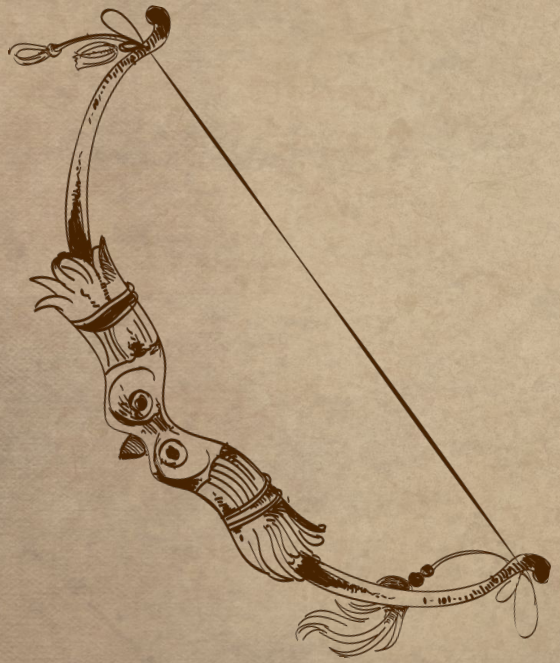
04 RxJSxJxJSxxS

# Mohammad-Ali A'rabi

Software Engineer @ Jodel
Docker Captain

Based in Freiburg, Germany

**aerabi** @ GitHub, Medium,
LinkedIn, Telegram, GoodReads
**aerabi.com** @ BlueSky
**MohammadAliEN** @ Twitter



01

# ReactiveX

A brief history

# COMPLEXITY KILLS

# A Brief History of ReactiveX

## 2007

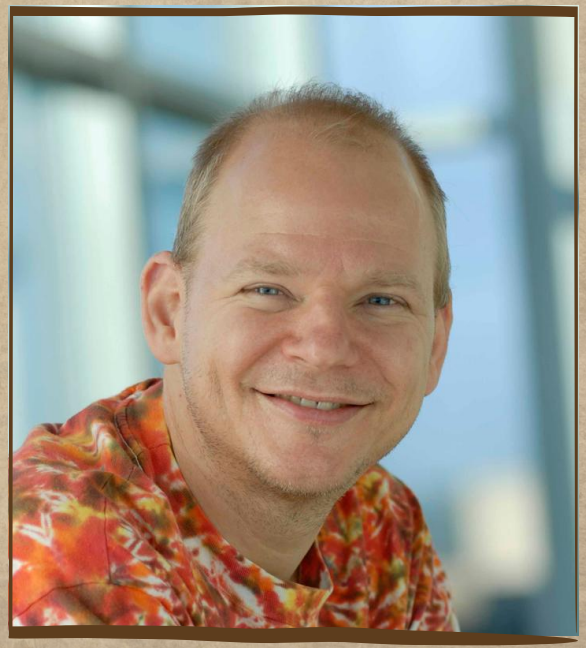
Complexity Kills @ Microsoft
Erik Meijer started ReactiveX

## 2012

Microsoft open-sourced RxJS

## 2013

Erik Meijer moved to Netflix
and started RxJava



# 2018

Amsterdam, Reactive hype

# Lovely summer day in Holland

# It's cold outside... and inside

**RxJava / Reactor**

```
numbers
  .map(number -> number * 2)
  .reduce((accumulator, current) -> accumulator + current)
  .subscribe(result -> {
    System.out.println("Reactor Result: " + result);
  });
```

**Akka Streams**

```
source
  .map(number -> number * 2)
  .runWith(Sink.fold(0, (xs, x) -> xs + x), materializer)
  .thenAccept(result -> {
    System.out.println("Akka Streams Result: " + result);
    system.terminate();
  });
```

FRP

**RxJS 5**

```
numbers
  .map(number => number * 2)
  .reduce((accumulator, current) => accumulator + current)
  .subscribe(result => {
    console.log(`RxJS 5 Result: ${result}`);
  });
```

Observable methods

independent functions

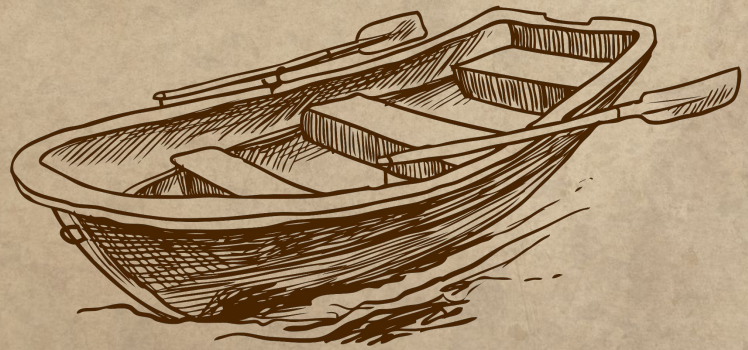
**RxJS 5.5 (2017)**

```
source.pipe(
  map(number => number * 2),
  reduce((accumulator, current) => accumulator + current),
).subscribe(result => {
  console.log(`RxJS 5 Result: ${result}`);
});
```

02

# DOING RXJS IN BACKEND

Pipe the shit
out of the code




Node + TypeScript = happiness

# There are two types of people

**Type I**

```
source
  .pipe(map(number => number * 2))
  .pipe(reduce((accumulator, current) => accumulator + current))
  .subscribe(result => {
    console.log(`RxJS 5 Result: ${result}`);
  });
```

**Type II**

```
source.pipe(
  map(number => number * 2),
  reduce((accumulator, current) => accumulator + current),
).subscribe(result => {
  console.log(`RxJS 5 Result: ${result}`);
});
```

“Our code is pure pipes.”

—SOMEONE FAMOUS

# WE ARE ALL PLUMBERS!

**Hard-to-read**

```
repository.getAll(user)
  .pipe(reduce((acc, v) => acc.concat([v]), [] as DocLink[]))
  .pipe(tap(list => expect(list.length).toEqual(10)))
  .subscribe(done, fail);
```


HALFWAY
THERE

**Readable**

```
const accumulateAsArray = <T>() =>
  reduce((acc, v: T) => acc.concat([v]), [] as T[]);

repository.getAll(user)
  .pipe(accumulateAsArray<DocLink>())
  .pipe(tap(list => expect(list.length).toEqual(10)))
  .subscribe(done, fail);
```

# SIDE-EFFECT KILLS!

# BLOG

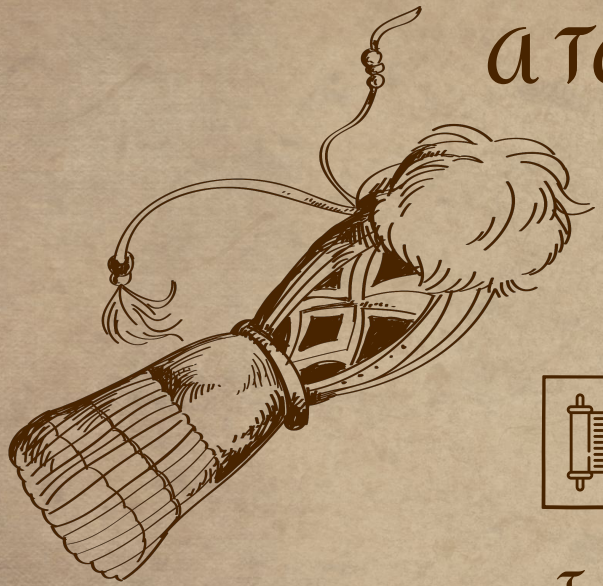
**Keeping Original Value When Transforming in RxJS**

**aerabi.medium.com**

## Keep original

```
repository.getAll(user)
  .pipe(flatMap(user => service.getPosts(user)))
  .pipe(flatMap(posts => // lost access to users));
```

# A TALE of TWO WEAPONS

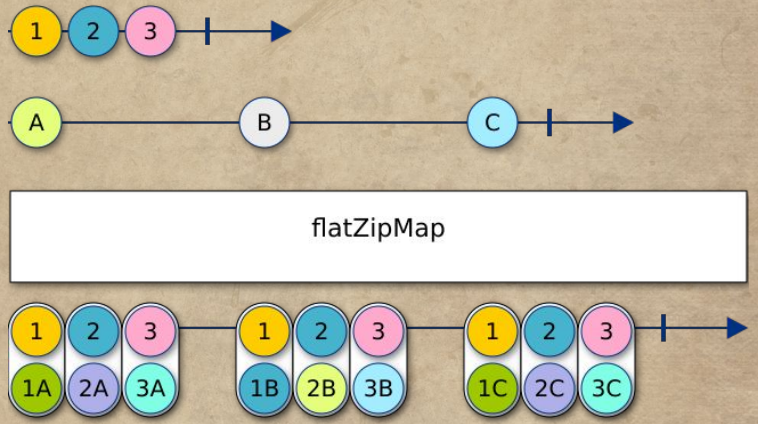


## Tuples

TypeScript makes it possible to have typing for tuples – that are otherwise arrays

## Operators

With introduction of new operators, we can deal with tuples and arrays




## Keep original

```
repository.getAll(user)
  .pipe(flatZipMap(user => service.getPosts(user)))
  .pipe(flatMap(([user, posts]) => // access to users));
```

# 03

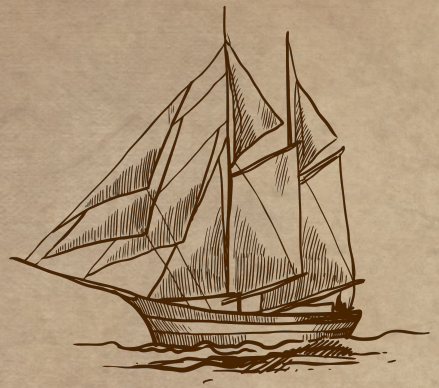
# RxJSx

RxJS Extensions
(i.e. extensions for reactive extensions)

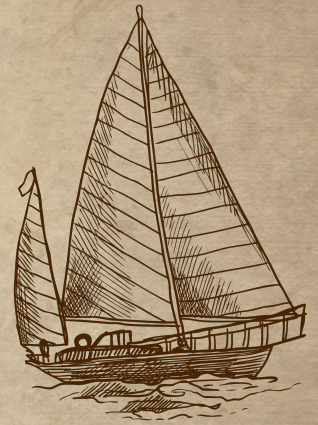
# BLOG

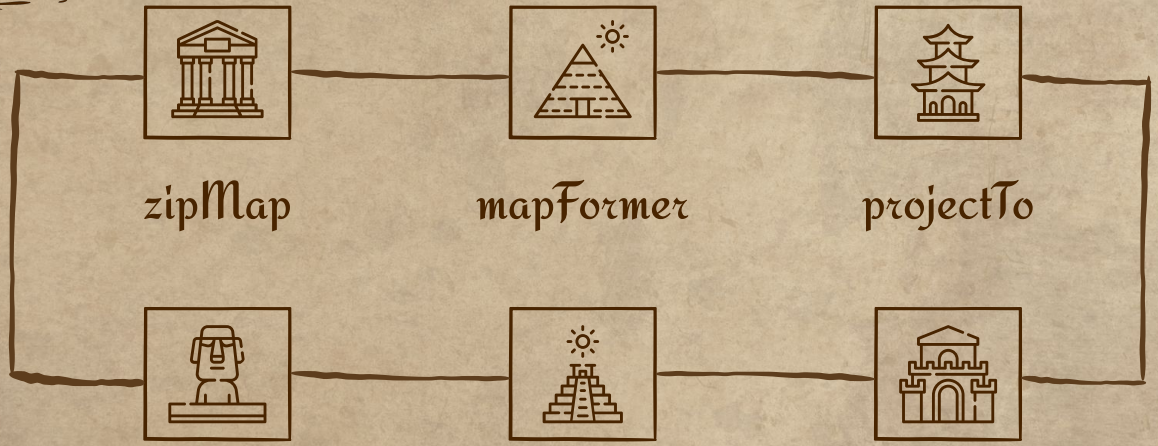
**13 Handy RxJS Operators**
to Deal with Arrays and Tuples

**aerabi.medium.com**



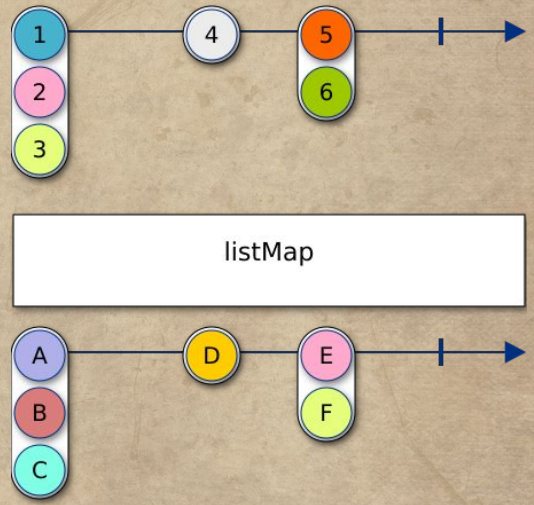
# Extensions for Reactive Extensions for JS

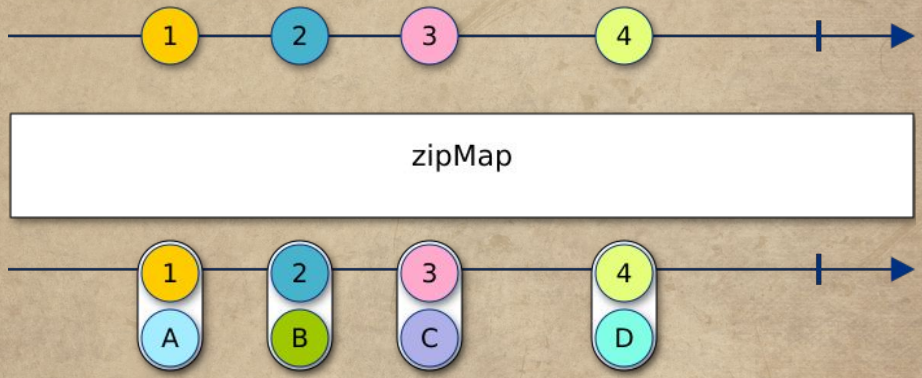




## listMap

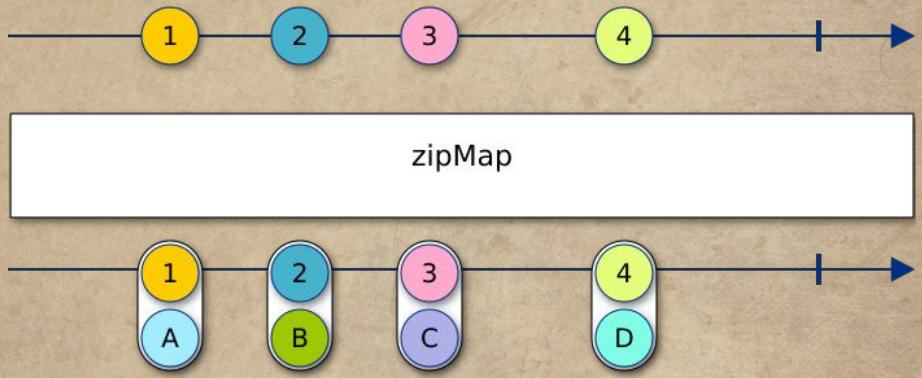
```
export function listMap<T, R>(
  project: (v1: T) => R
): OperatorFunction<T[], R[]> {
  return map(list => list.map(project));
}
```






## zipMap

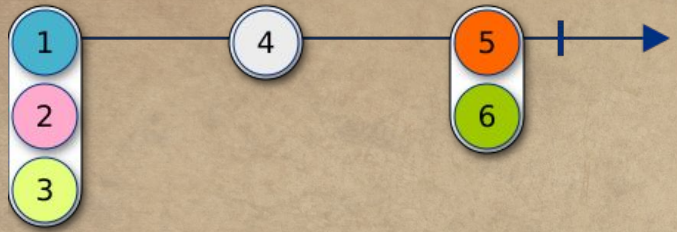
```
export function zipMap<T, R>(
  project: (v1: T) => R
): OperatorFunction<T, [T, R]> {
  return map(val => [val, project(val)]);
}
```

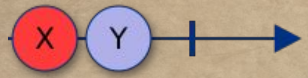



## flatZipMap

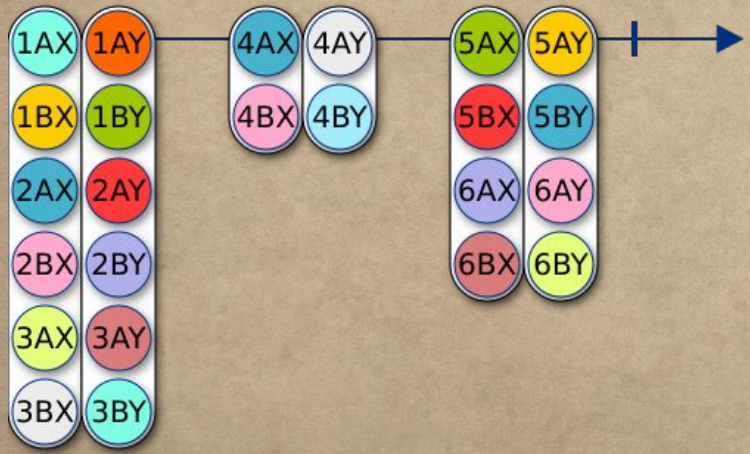
```
export function flatZipMap<T, O extends ObservableInput<any>>(
  project: (v1: T) => O
): OperatorFunction<T, [T, ObservedValueOf<O>]> {
  return flatMap(val => zip(of(val), project(val)));
}
```

# PURE MADNESS



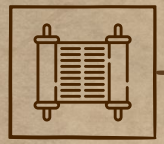

flatListFlatMap




**flatListFlatMap**

```
export function flatZipMap<T, R, O extends ObservableInput<R[]>>(
  project: (v1: T) => O
): OperatorFunction<T[], R[]> {
  return flatMap(list =>
    zip(...list.flatMap(value => project(value)))
      .pipe(map(list => list.flatMap(x => x))),
  );
}
```

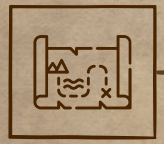
# FINAL LIST



GITHUB — github.com/**rxjsx**/rxjsx

NPM — **@rxjsx**/rxjsx

HACKTOBER — Contribute as part of **Hacktoberfest**



MEDIUM — Learn about it @ aerabi.medium.com

# Conclusions

1. No wet pipes

   Make the pipes DRY

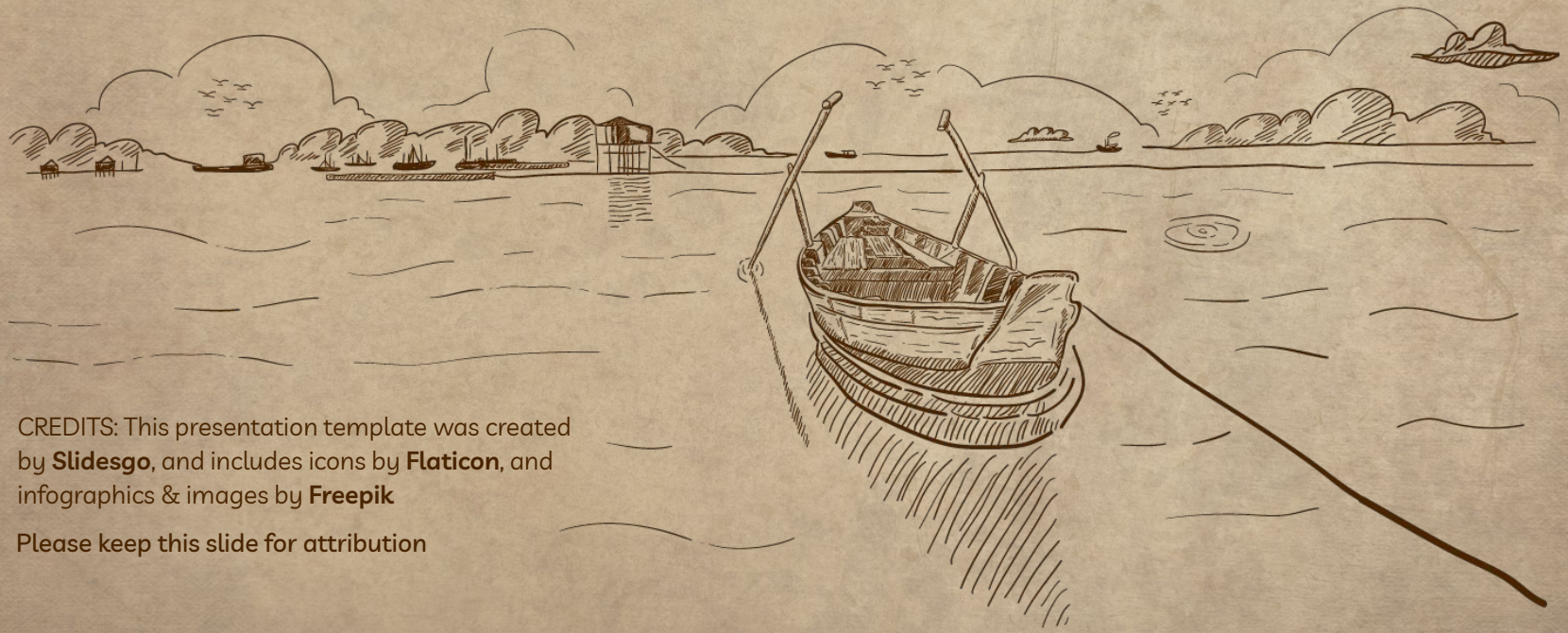
2. TypeScript is good

   Use tuples, don't mutate

# Thanks

Please don't forget to like and subscribe

@MohammadAliEN
aerabi.medium.com



# DO YOU HAVE ANY QUESTIONS?

The slides are here:
**github.com/aerabi/talks**