@simonbasle

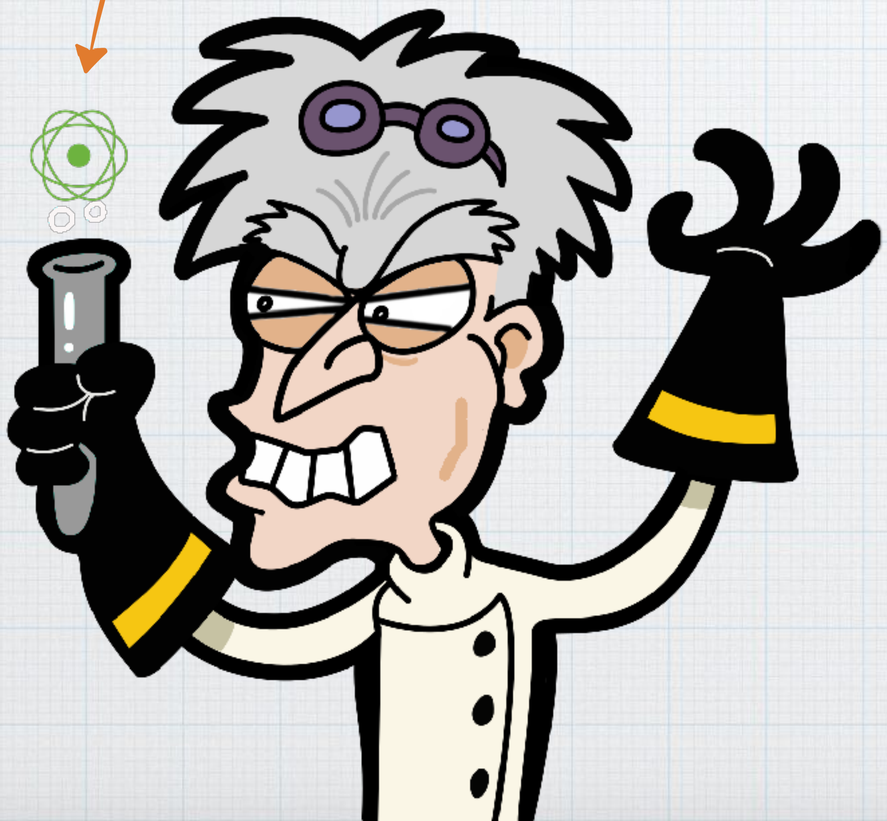
reactive MythBusters

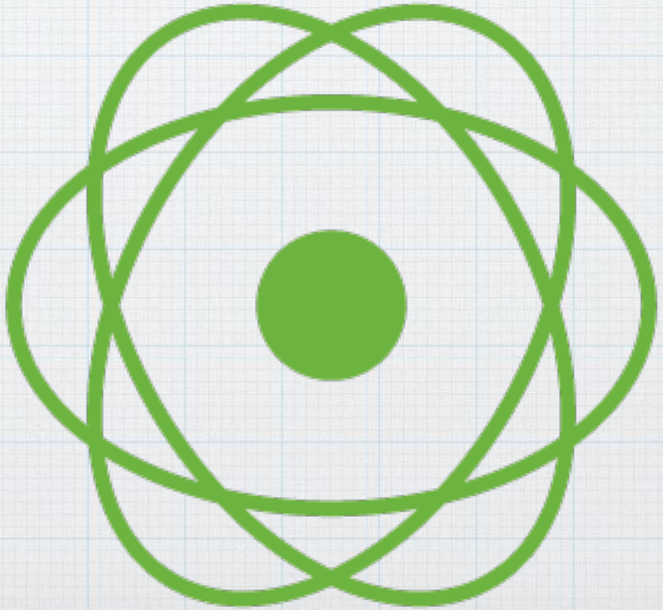
reactive MythBusters

testing 9 propositions
about reactive programming
for science (and glory)

This mad scientist is making mad claims

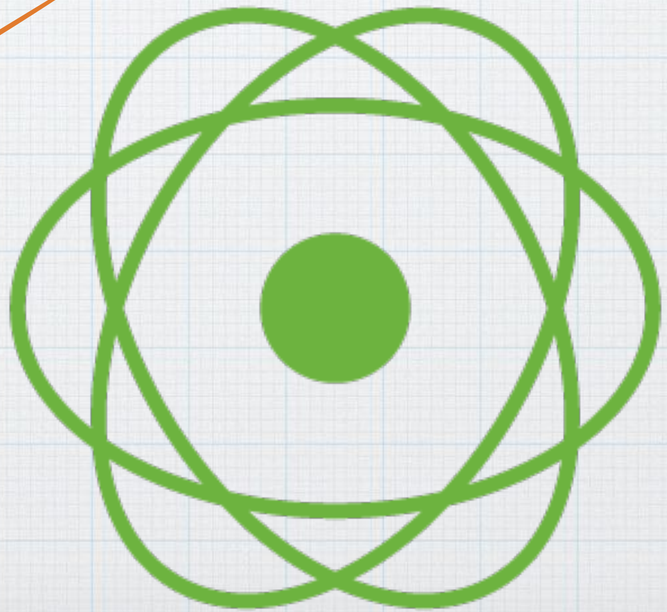
But he's still a scientist,
so he'll put them to the test

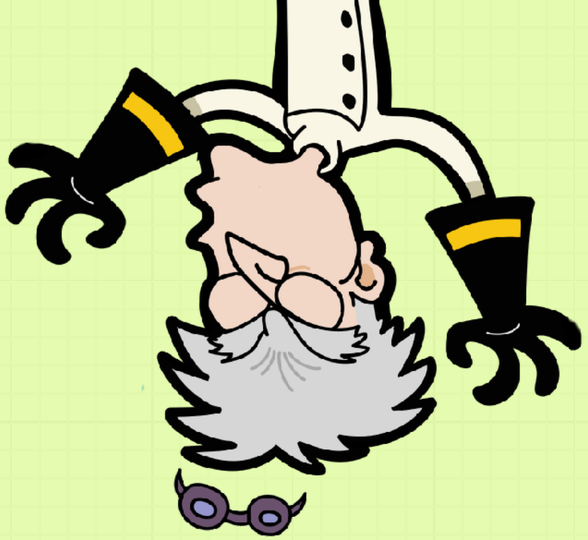




and sometimes he'll be

proven wrong





BUSTED

PLAUSIBLE

CONFIRMED

reactive MythBusters

with Reactor

the 9 propositions

There's a Learning curve

It's inherently complex

It's limited by a set of operators

It's only good for GUIs

Implementing Flow is easy

This is hard to Test

This is hard to Debug

Blocking can make things worse

You have to be a concurrent programming expert

1. There's a Learning curve

1. There's a Learning curve

LiveExperiment

!

1. There's a Learning curve

CONFIRMED

Read the Docs ;)

http://projectreactor.io/docs/

Reactor Core

A Reactive Streams foundation for Java 8

Version	Release Train
3.1.1.RELEASE	BISMUTH-SR3

Github

Javadoc

Release | Snapshot | Kotlin Doc

Reference

Release | Snapshot

Choosing an Operator

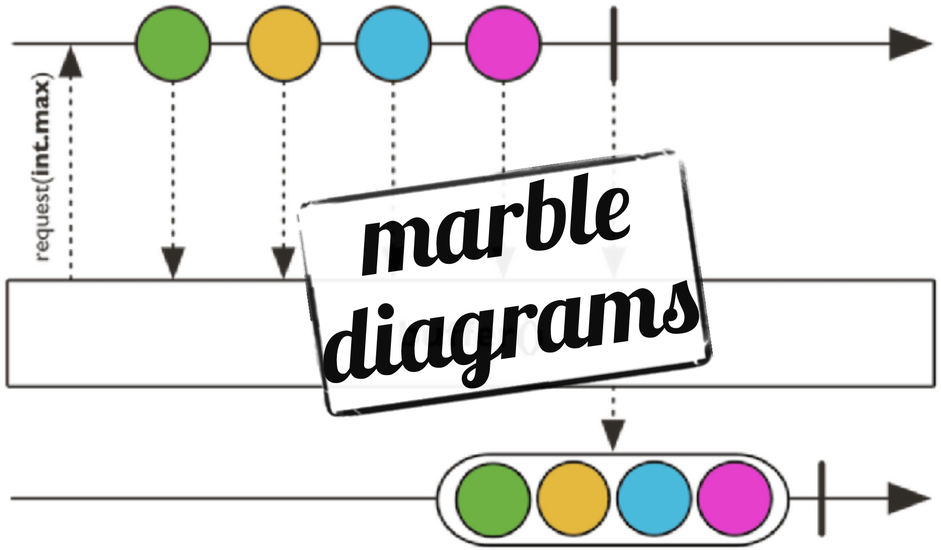
Read the Docs ;)

javadoc

buffer

```
public final Flux<List<T>> buffer()
```

Collect all incoming values into a single `List` buffer that will be emitted by the returned `Flux` once this Flux completes.



Returns:

a buffered `Flux` of at most one `List`

Read the Docs ;)

reference documentation

4. Reactor Core Features

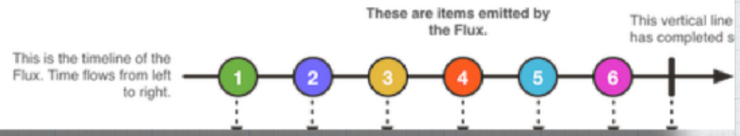
The Reactor project main artifact is `reactor-core`, a reactive libra
and targets Java 8.

Reactor introduces composable reactive types that implement `Pub`
most notably `Flux` and `Mono`. A `Flux` object represents a reactive s
a single-value-or-empty (0..1) result.

This distinction carries a bit of semantic information into the type, i
processing. For instance, an HTTP request produces only one resp
operation. Expressing the result of such an HTTP call as a `Mono<H`
it as a `Flux<HttpResponse>`, as it offers only operators that are re

Operators that change the maximum cardinality of the processing
operator exists in `Flux`, but it returns a `Mono<Long>`.

4.1. Flux, an Asynchronous Sequenc



Read the Docs ;)

Appendix A: Which operator do I need?

> In this section, if an operator is specific to `Flux` or `Mono` it is prefixed accordingly. Common operators have no prefix. When a specific use case is covered by a combination of operators, it is presented as a method call, with leading dot and parameters in parentheses, like this: `.methodCall(parameter)`.

I want to deal with:

- Creating a New Sequence…
- Transforming an Existing Sequence
- Peeking into a Sequence
- Handling Errors
- Working with Time
- Splitting a `Flux`
- Going Back to the Synchronous World

A.1. Creating a New Sequence…

- that emits a `T`, and I already have: `just`
  - …from an `Optional<T>`: `Mono#justOrEmpty(Optional<T>)`
  - …from a potentially `null` T: `Mono#justOrEmpty(T)`

operator choice matrix

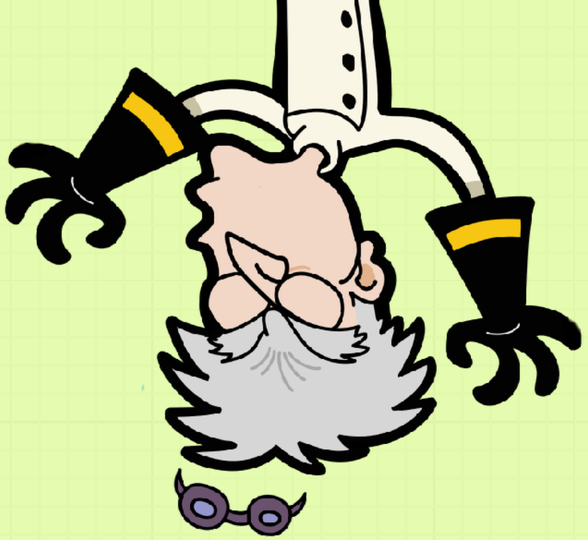
2. It's inherently complex

2. It's inherently complex

Live Experiment

!

2. It's inherently complex



BUSTED

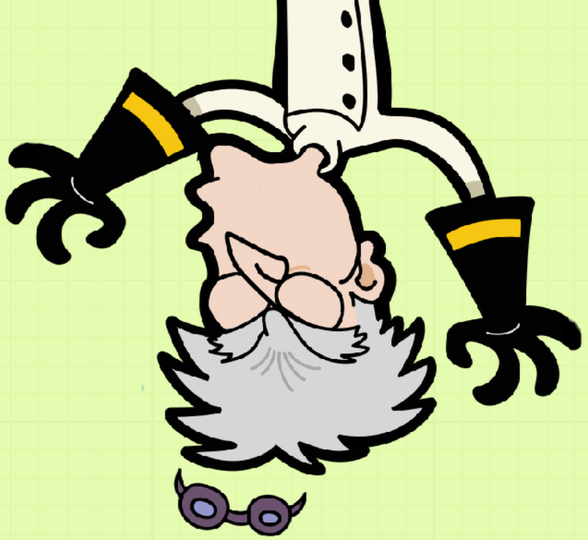
3. It's limited by a set of operators

3. It's limited by a set of operators

LiveExperiment

!

3. It's limited by a set of operators



BUSTED

4. It's only good for GUIs

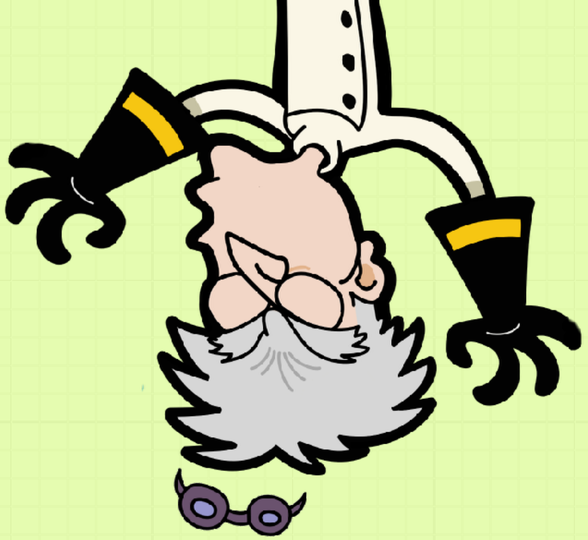
4. It's only good for GUIs

LiveExperiment

!

4. It's only good for GUIs

4. It's only good for GUIs



BUSTED

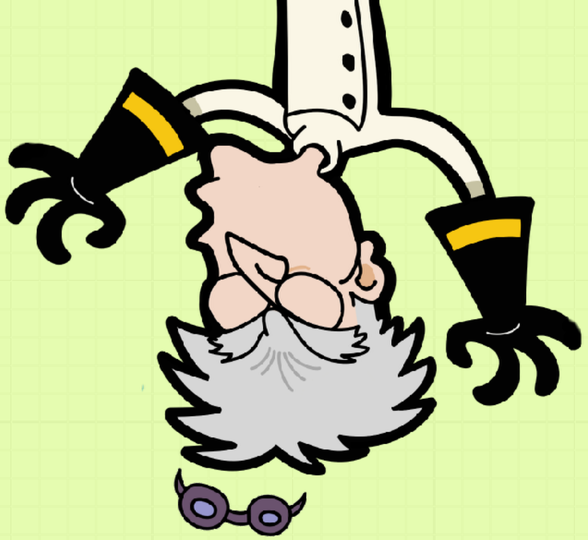
5. Implementing Flow is easy

5. Implementing Flow is easy

CodeExperiment

!

5. Implementing Flow is easy



BUSTED

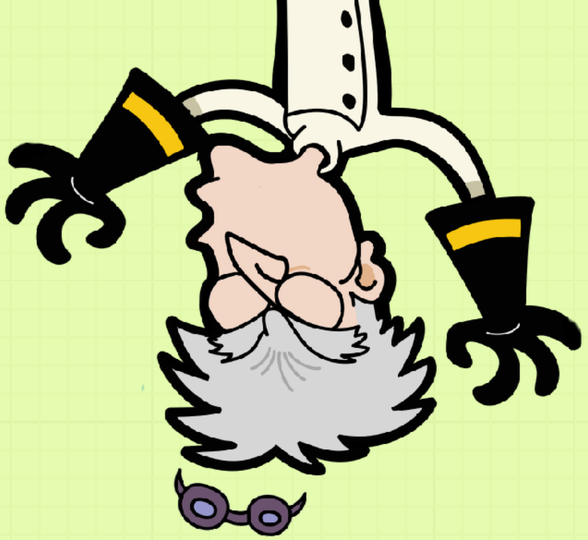
6. This is hard to Test

6. This is hard to Test

Live Experiment

!

6. This is hard to Test



BUSTED

7. This is hard to Debug

7. This is hard to Debug

LiveExperiment

!

7. This is hard to Debug

PLAUSIBLE

8. Blocking can make things worse

8. Blocking can make things worse

Live Experiment !

8. Blocking can make things worse

CONFIRMED

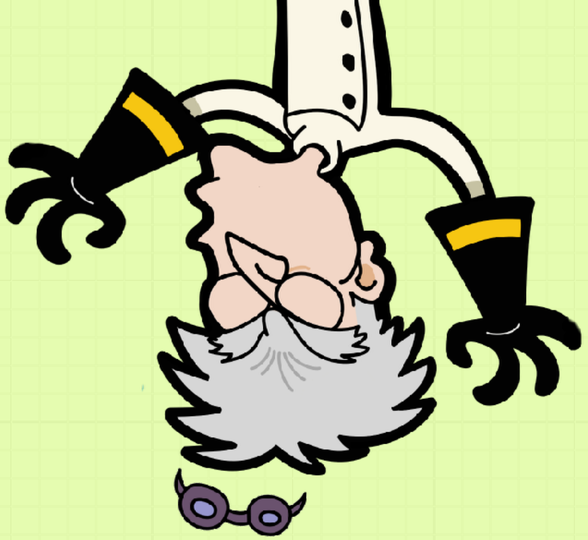
9. You have to be a concurrent programming expert

9. You have to be a concurrent programming expert

LiveExperiment

!

9. You have to be a concurrent programming expert



BUSTED

There's a Learning curve

It's inherently complex

It's limited by a set of operators

It's only good for GUIs

Implementing Flow is easy

This is hard to Test

This is hard to Debug

Blocking can make things worse

You have to be a concurrent programming expert

that's it

that's it

Questions?!

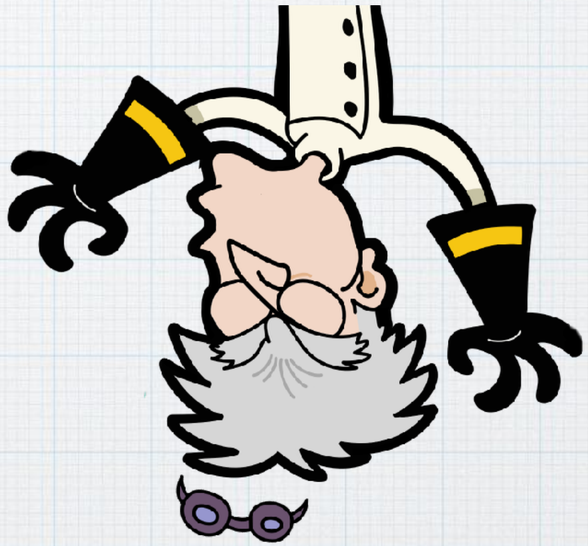
@simonbasle

that's it

Questions?!

@simonbasle

The END



The END

Thank You !

Mad Scientist CC-By-SA via Wikimedia https://commons.wikimedia.org/wiki/File:Mad_scientist_transparent_background.svg
modified by Simon Baslé