What and Whatnot
Why?
The Reactive Manifesto

Published on September 16 2014. (v2.0)

Organisations working in disparate domains are independently discovering patterns for building software that look the same. These systems are more robust, more resilient, more flexible and better positioned to meet modern demands.

These changes are happening because application requirements have changed dramatically in recent years. Only a few years ago a large application had tens of servers, seconds of response time, hours of offline maintenance and gigabytes of data. Today applications are deployed on everything from mobile devices to cloud-based clusters running thousands of multi-core processors. Users expect millisecond response times and 100% uptime. Data is measured in Petabytes. Today’s demands are simply not met by yesterday’s software architectures.

We believe that a coherent approach to systems architecture is needed, and we believe that all necessary aspects are already recognised individually: we want systems that are Responsive, Resilient, Elastic and Message Driven. We call these Reactive Systems.

Systems built as Reactive Systems are more flexible, loosely-coupled and scalable. This makes them easier to develop and amenable to change. They are significantly more tolerant of failure and when failure does occur they meet it with elegance rather than disaster. Reactive Systems are highly responsive, giving users effective interactive feedback.
The Manifesto

- Responsive
- Resilient
- Elastic
- Message Driven
Asynchronous Programming

...is hard
Applications require it

- Desktop
- Mobile
- Web
- Server-Side
Attempts to ease it

- Threading
- Async/Await
- Futures
- Callbacks
- Promises
Issues...

- Combining async operations
- Different techniques for different languages
Reactive Extensions

An API for asynchronous programming with observable streams

Choose your platform
Reactive Extensions

An API for asynchronous programming with observable streams

Choose your platform
Everything. Is. A. Stream
And. It’s. Observable.
DOGS MUST KEEP OWNERS ON A LEASH
Let’s take a step back…
What is Programming?
An Art.
We are Crafts People
Actually we just process data
Data Processing

In Stages
Data comes in various forms...

- **Scalars**: Date of Birth, Sales Amount, Names
- **Collections**: List of Countries, Collection of Customers
Demo

Processing Data Functionally
public trait Iterable<out T> {
    public fun iterator(): Iterator<T>
}

public trait Iterator<out T> {
    /**
     * Returns the next element in the iteration.
     */
    public fun next(): T

    /**
     * Returns `true` if the iteration has more elements.
     */
    public fun hasNext(): Boolean
}
fun main(args: Array<String>) {
    val numbers = listOf(1, 2, 3, 4, 5, 6, 7, 8)

    numbers.forEach { number -> println(number) }
}

Blocking Call
Some long running process

```java
val customers = Database.getCustomersByCountry("Spain")

customers.filter { it.profession == "Agriculture" }
    .getGrowthFigures(it.city)
    .displayGrowth(it)
```

Blocking Calls
We shouldn’t block!
We’re waiting for data…
Why not just get notified?
i.e. onDataReady event
“What’s the difference between an array and events?”

–Erik Meijer
Observer Pattern
public trait Iterable<out T> {
    public fun iterator(): Iterator<T>
}

public trait Iterator<out T> {
    /**
     * Returns the next element in the iteration.
     */
    public fun next(): T

    /**
     * Returns `true` if the iteration has more elements.
     */
    public fun hasNext(): Boolean
}
public trait Observable<out T> {
    public fun Subscribe(observer: Observer<T>)
}

public trait Observer<in T> {
    public fun onNext(t: T)
    public fun onError(e: Exception)
    public fun onComplete()
}
Duality

In mathematics, a duality, generally speaking, translates concepts, theorems or mathematical structures into other concepts, theorems or structures, in a one-to-one fashion, often (but not always) by means of an involution* operation: if the dual of A is B, then the dual of B is

Involution Function: \( f(f(x)) = x \)
Law of De Morgan

\(! (a \land b) = !a \lor !b\)

\(! (a \lor b) = !a \land !b\)
Reactive Extensions is

- Extension of Observer Pattern
- Operators to work with sequences of data and their composition
- Abstracts low-level threading, etc.
Reactive Extensions

- Languages
  - Java: RxJava
  - JavaScript: RxJS
  - C#: Rx.NET
  - C#(Unity): UniRx
  - Scala: RxScala
  - Clojure: RxClojure
  - C++: RxCpp
  - Ruby: Rx.rb
  - Python: RxPY
  - Groovy: RxGroovy
  - JRuby: RxJRuby
  - Kotlin: RxKotlin

- Frameworks
  - RxNetty
  - RxReact
  - RxAndroid
  - RxUI
Now it’s all just Demos
Is this the Silver Bullet?
Where Rx helps

- Asynchronous code
- Combining results
- Events
- Errors
Most importantly…
It adds a layer of abstraction
Some issues…

- Learning Curve
- Thinking in Functions
- Thinking in Streams
- Can still cause havoc in your code
- Dealing with Schedulers
- Doesn’t automatically give you parallelism
Learning more

Introductions

- *Introduction to Rx*: a free, on-line book by Lee Campbell
- *Your Mouse is a Database* by Erik Meijer
- *The introduction to Reactive Programming you’ve been missing* by André Staltz
- *Mastering Observables* from the Couchbase documentation
- *33rd Degree Reactive Java* by Tomasz Kowalczewski
- *2 minute introduction to Rx* by André Staltz
- Wikipedia: *Reactive Programming* and *Functional Reactive Programming*
- *Be Reactive* by ColinTheShots
- *Collection Pipeline* by Martin Fowler
- *Rx for .NET and RxJava for Android* by Olli Salonen

Tutorials

- *Learn RxJS* by Jafar Husain
- *Learn RxJava* by Jafar Husain
- *RxJS Koans* by Matt Podwysocki
- *Rx Workshop*
- *Reactive Programming and MVC*
RxMarbles

Interactive diagrams of Rx Observables

- delay
- delayWithSelector
- findIndex
- map
- scan
- debounce
- debounceWithSelector

COMBINING OPERATORS
- combineLatest
- concat
- merge
- sample
- startWith
- withLatestFrom
- zip
Thank you!
Credits


- Call Center [http://upload.wikimedia.org/wikipedia/commons/7/7c/Callcentre.jpg](http://upload.wikimedia.org/wikipedia/commons/7/7c/Callcentre.jpg)

- Open Office Layout [http://upload.wikimedia.org/wikipedia/commons/a/a4/TradeMe_offices.jpg](http://upload.wikimedia.org/wikipedia/commons/a/a4/TradeMe_offices.jpg)