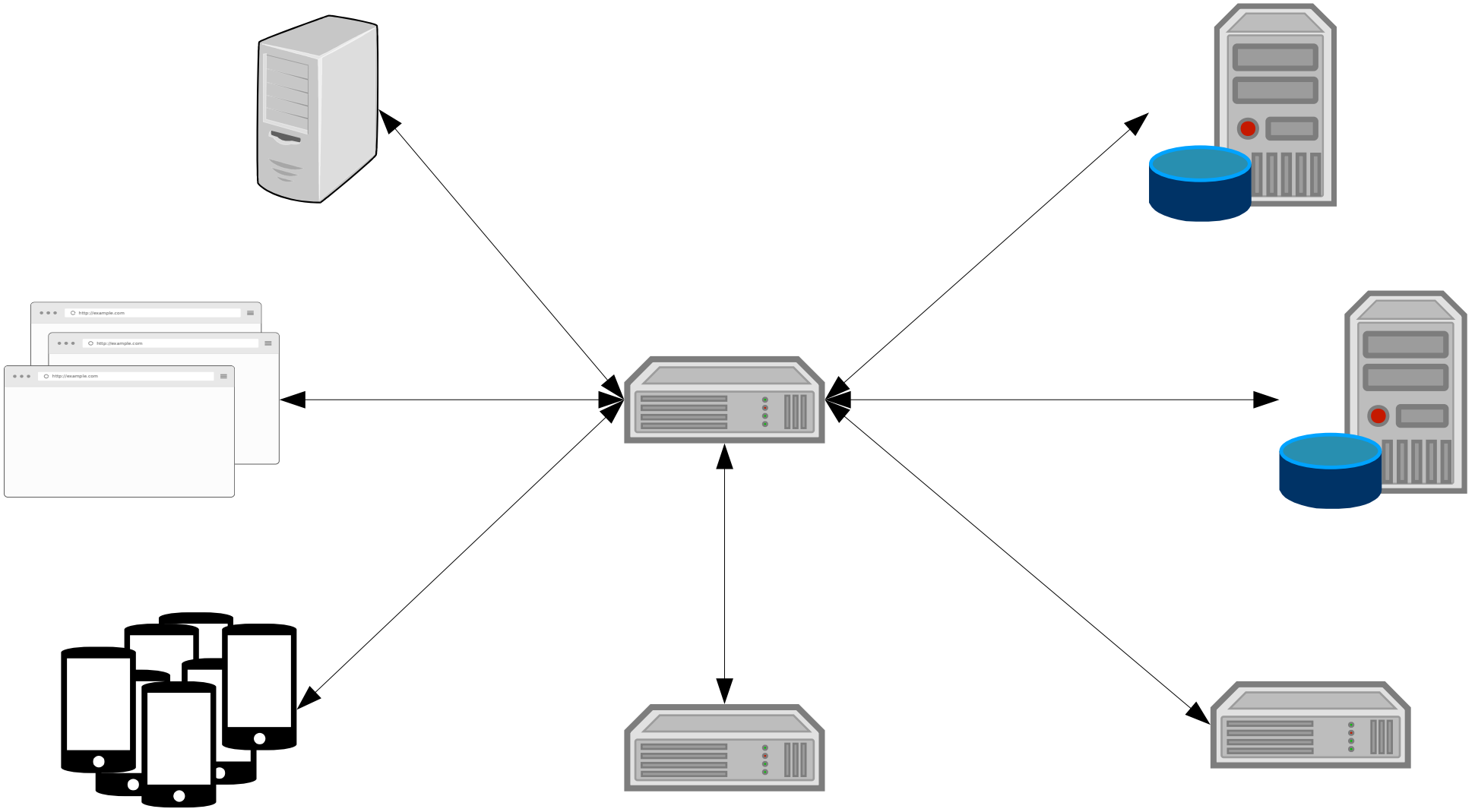
# Modern app programming with RxJava & Eclipse Vert.x

# Who am I?

- Thomas Segismont
- *tsegismont* on GitHub, Twitter, Gmail, Freenode...
- Vert.x core team since August 2016
- At Red Hat since November 2012 (RHQ and Hawkular)

# Expectations

- Goals
  - Why should you consider learning Vert.x and RxJava?
  - What does RxJava look like? Vert.x?
  - How can I combine them to build scalable, efficient and resilient apps?
- Non goals
  - In-depth study of RxJava or Vert.x

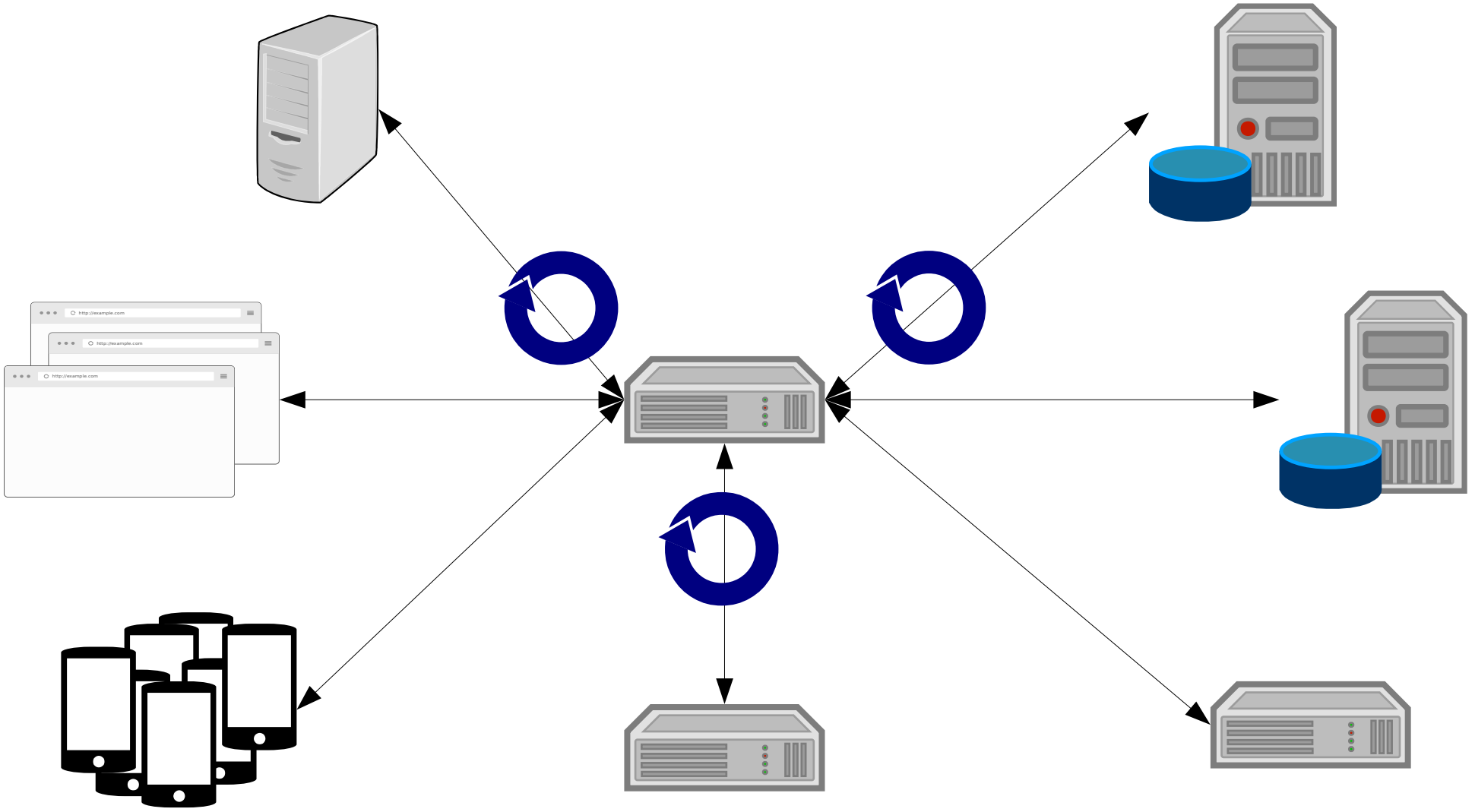

http://example.com
http://example.com
http://example.com


Alexandre Duret-Lutz (CC BY-SA 2.0)

# ExecutorService to the rescue

# Asynchronous

- ExecutorService
- Future




**Thread pools, thread pools EVERYWHERE!**


The C10K problem

# Non-blocking

- Stop wasting CPU for waiting
  - NIO
  - CompletableFuture

# Improvements in modern app servers

- Non blocking I/O in connectors and dispatch to worker threads
- Continuations (suspend/resume)

# The cost of contention and delay

- Neil J. Gunther, Universal Law of Computational Scalability

# Mechanical sympathy

# The pursuit of answers

- Benchmark by Netflix
  - Tomcat vs Netty for real web apps
  - http://bit.ly/2cJMBsG
- Under load
  - Less thread migrations
  - More instructions per cycle
  - Severe latency degradation on Tomcat (unpredictable)
  - Higher throughput on Netty

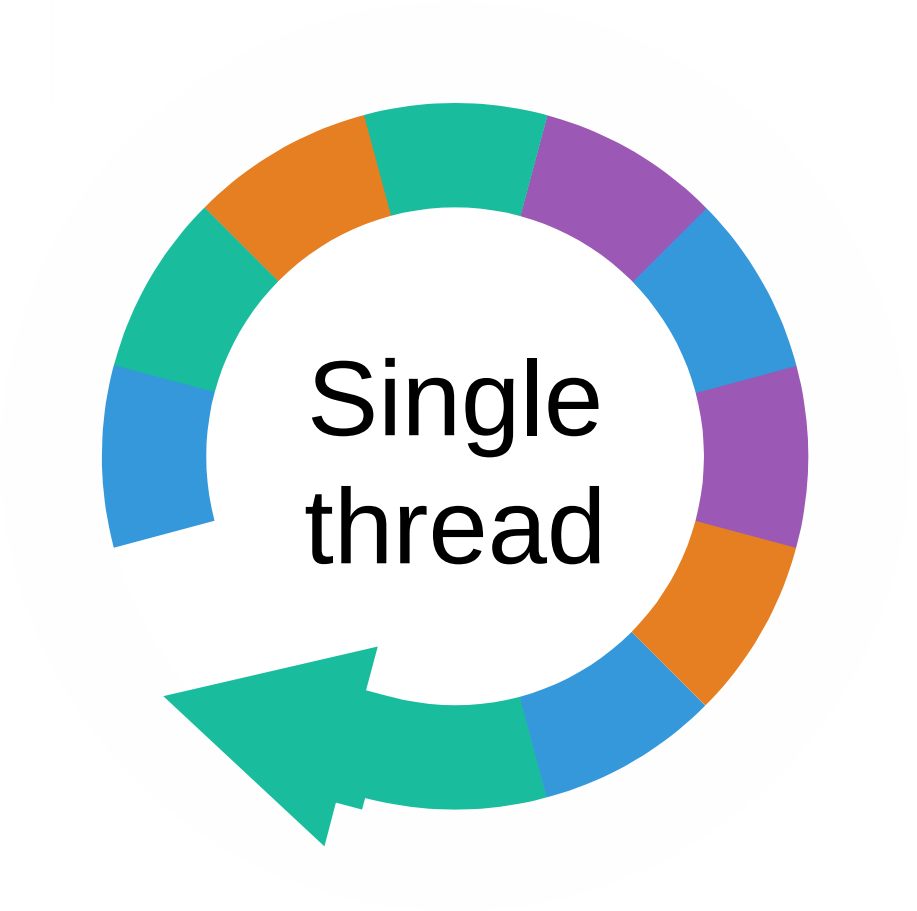
# Event loops


recycle
it's easy!

# New stuff?

- Browser (AJAX, `#setTimeout`)
- Swing
- Node.js

# Reactor pattern



# Event loop benefits

- Mechanical sympathy
- Simple concurrency model
- Easier to scale

# Vert.x

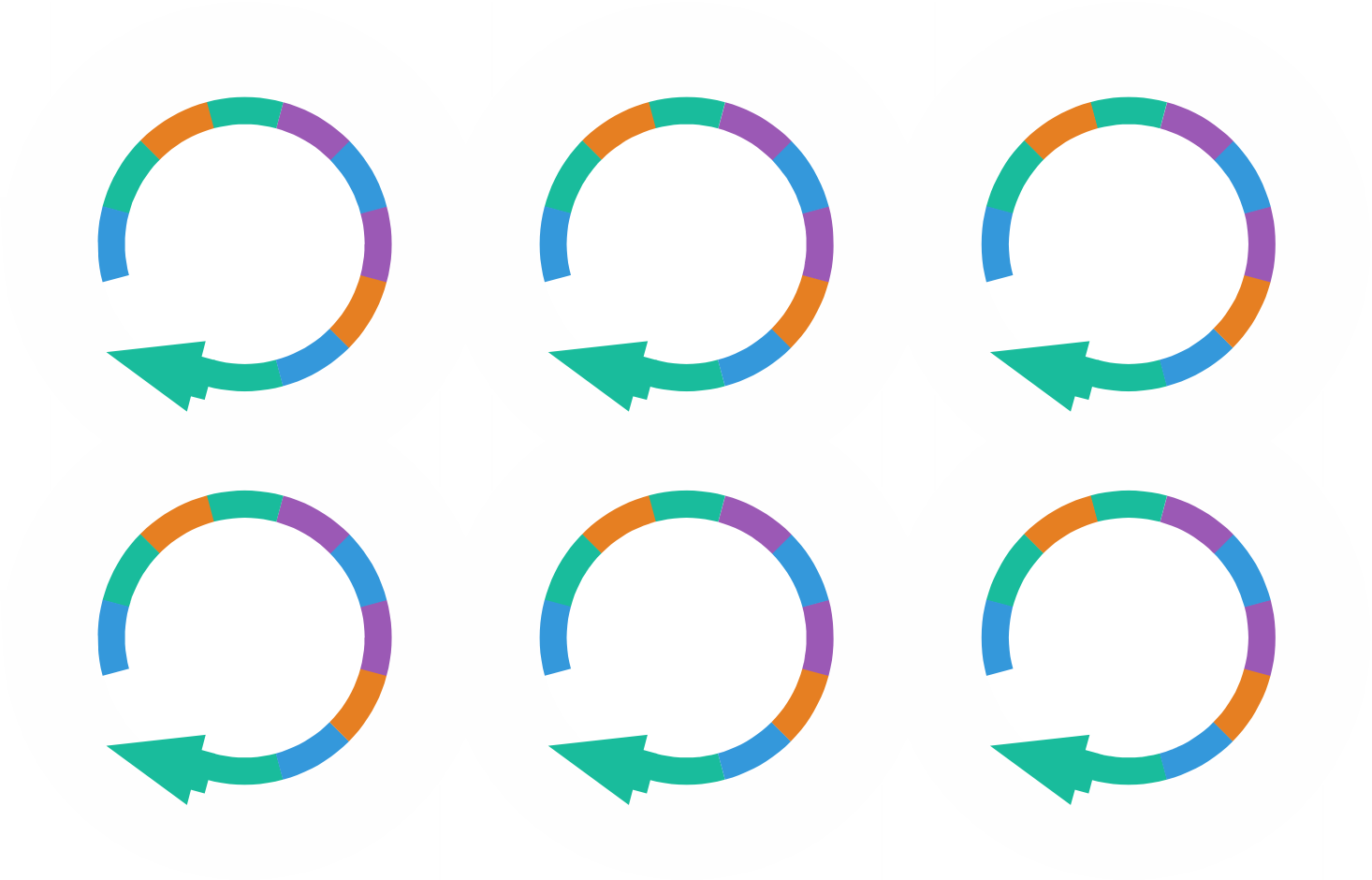
# What is Vert.x

- Toolkit (lib)
- Polyglot (on the JVM)
- Reactive

# Events

- Http server request
- Buffer read from a file
- Server bound to port
- Message from the bus

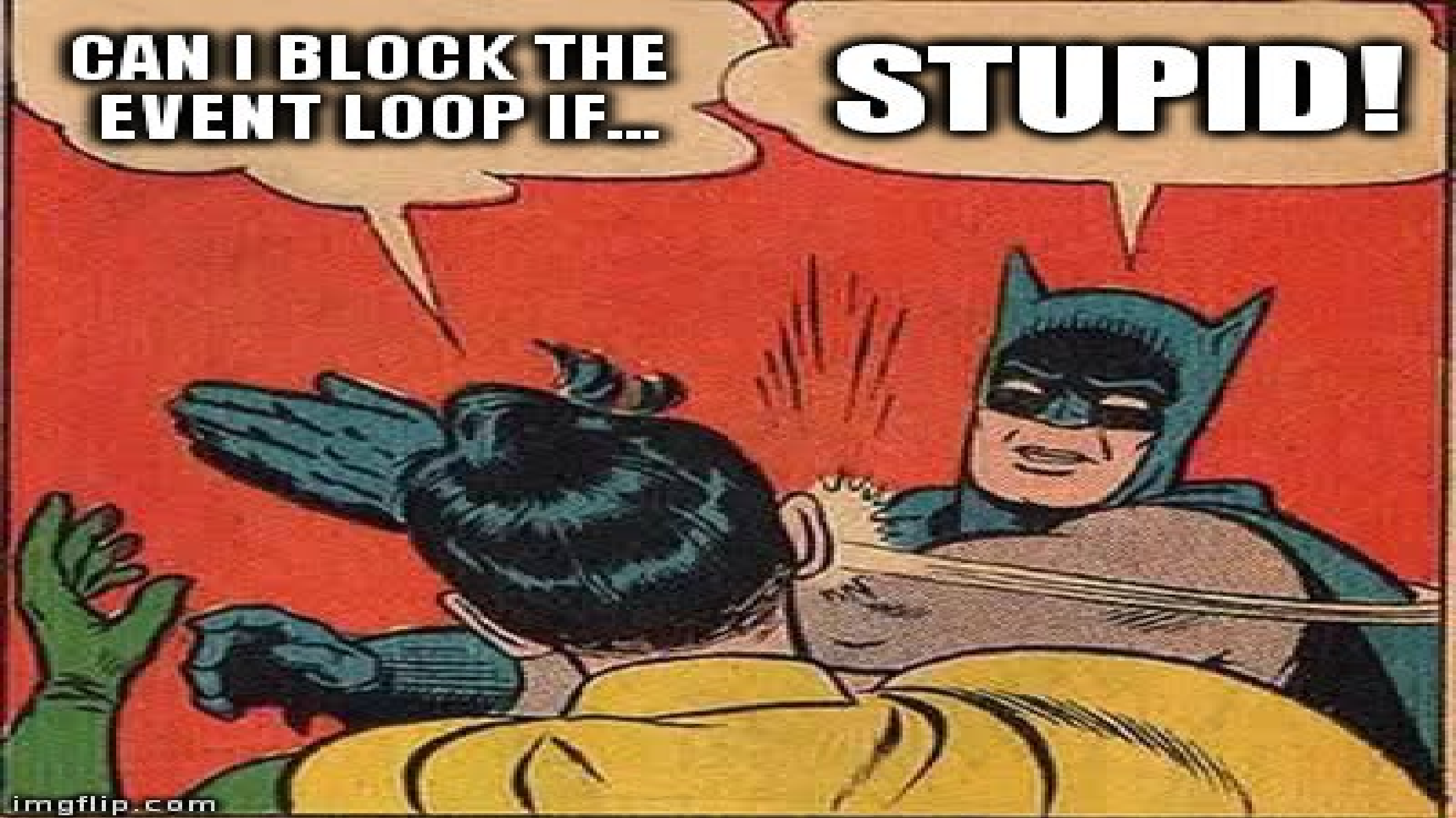
# Multi-Reactor



# Event Bus

- Nervous system of Vert.x
- Message passing style
- Different messaging paradigms:
  - Point to point
  - Publish/Subscribe
  - Request/Reply


MQTT.ORG
RabbitMQ
JWT
hazelcast
mongoDB
AMQP
kafka
redis
kubernetes
docker
Infinispan
apache ignite
RxJava
MySQL
Apache Camel
OPENSHIFT
PostgreSQL
spring

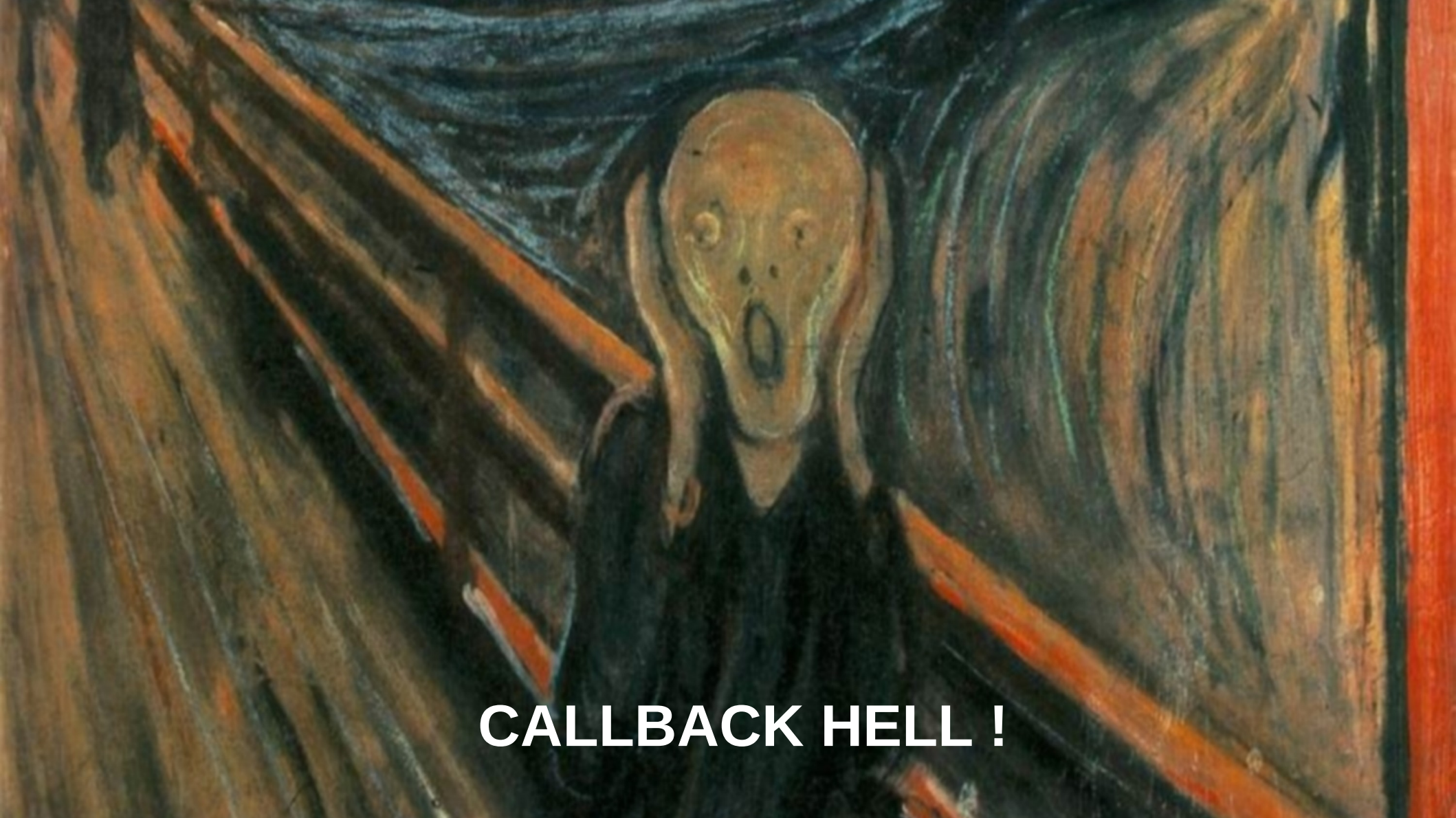

CAN I BLOCK THE EVENT LOOP IF...
STUPID!
imgflip.com

# Running blocking code

- In the real world, most JVM libraries have blocking APIs
  - JDBC
  - io.File
  - Your legacy libs
  - Heavy computation
- With Vert.x you can interact with blocking parts easily
- **You won’t throw away your code assets!**

# Callback based (reactive imperative)

```
void dumpToFile(ReadStream<Buffer> bufferSource, Handler<AsyncResult<Void>> handler) {
  bufferSource.pause();
  Future<Void> future = Future.<Void>future().setHandler(handler);
  FileSystem fileSystem = vertx.fileSystem();
  fileSystem.exists("/path/to/file", existsAr -> {
    if (existsAr.failed()) {
      System.out.println("No file...");
      future.fail(existsAr.cause());
    } else {
      fileSystem.truncate("/path/to/file", 0, truncateAr -> {
        if (truncateAr.failed()) {
          System.out.println("Truncate failure...");
          future.fail(truncateAr.cause());
        } else {
          fileSystem.open("/path/to/file", new OpenOptions(), openAr -> {
            if (openAr.failed()) {
              System.out.println("Open failure...");
              future.fail(truncateAr.cause());
            } else {
              AsyncFile asyncFile = openAr.result();
              bufferSource.handler(asyncFile::write);
              bufferSource.endHandler(v -> {
                asyncFile.close();
                future.complete();
              });
              bufferSource.resume();
            }
          });
        }
      });
    }
  });
}
```


CALLBACK HELL !

# RxJava

**A library for composing asynchronous and event-based programs using observable sequences for the Java VM.**

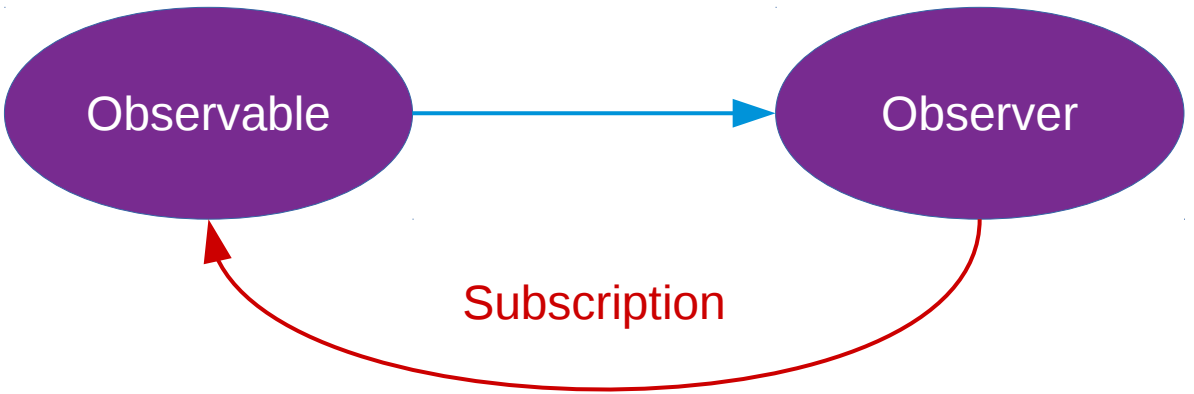
# Functional programming influence

- Favors composition
- Avoids global state
- Avoids side effects

# Data and events flows

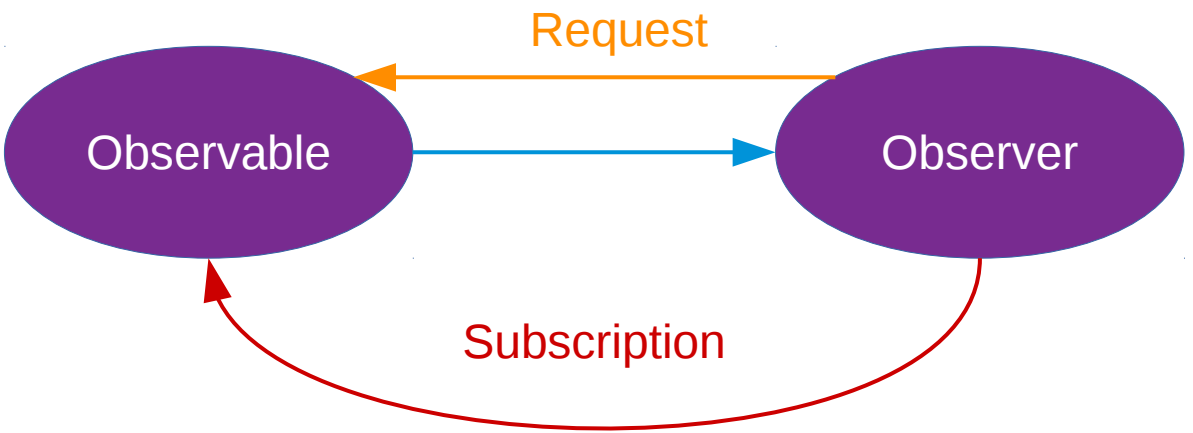
- It is great at organizing transformation of data and coordination of events
- It makes most sense when many sources of events are involved (our web apps!)

# Push based




- `OnNext` 0..∞
- Terminal event 0..1
  - `OnComplete`
  - `OnError`

# Reactive pull backpressure



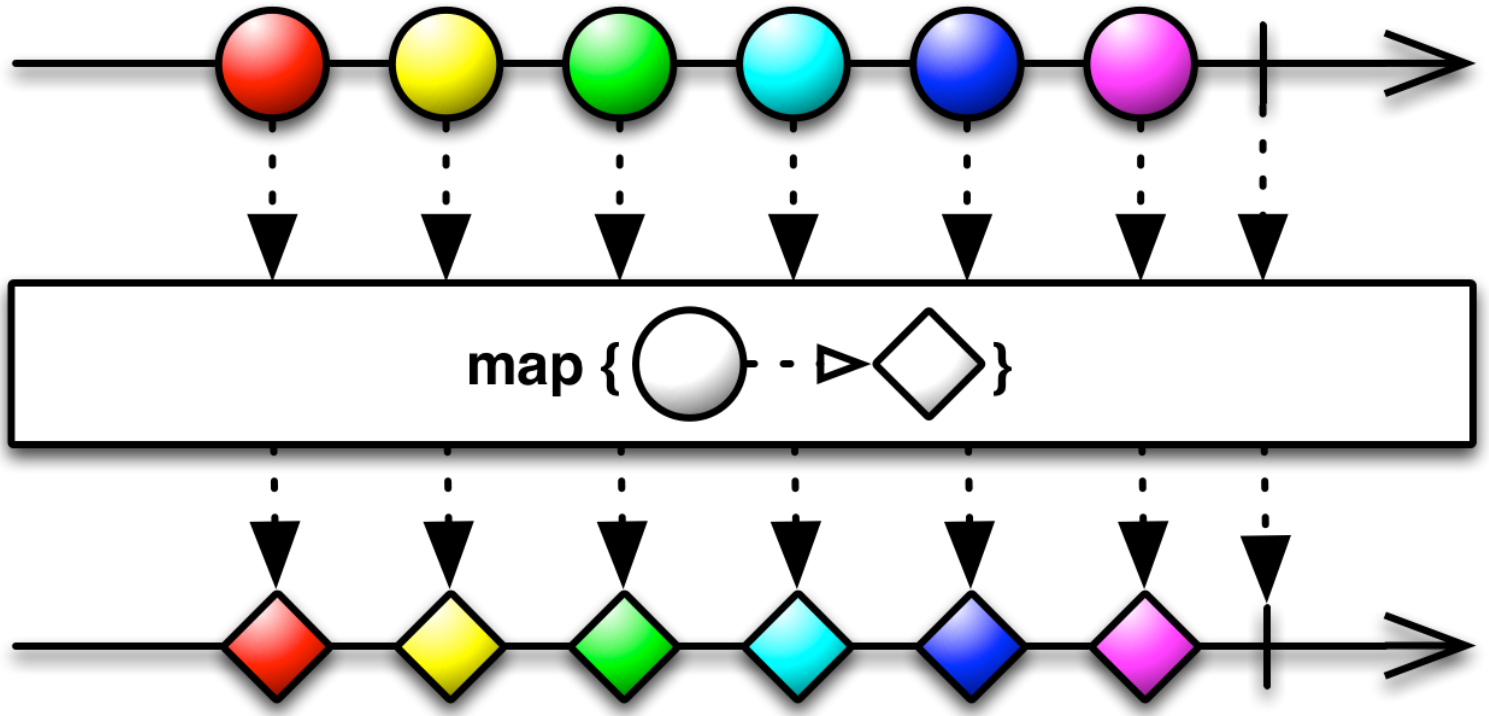
# Concurrency

- No concurrent emission of events
  - Easier to understand
  - Some operators need the guarantee (`reduce`)

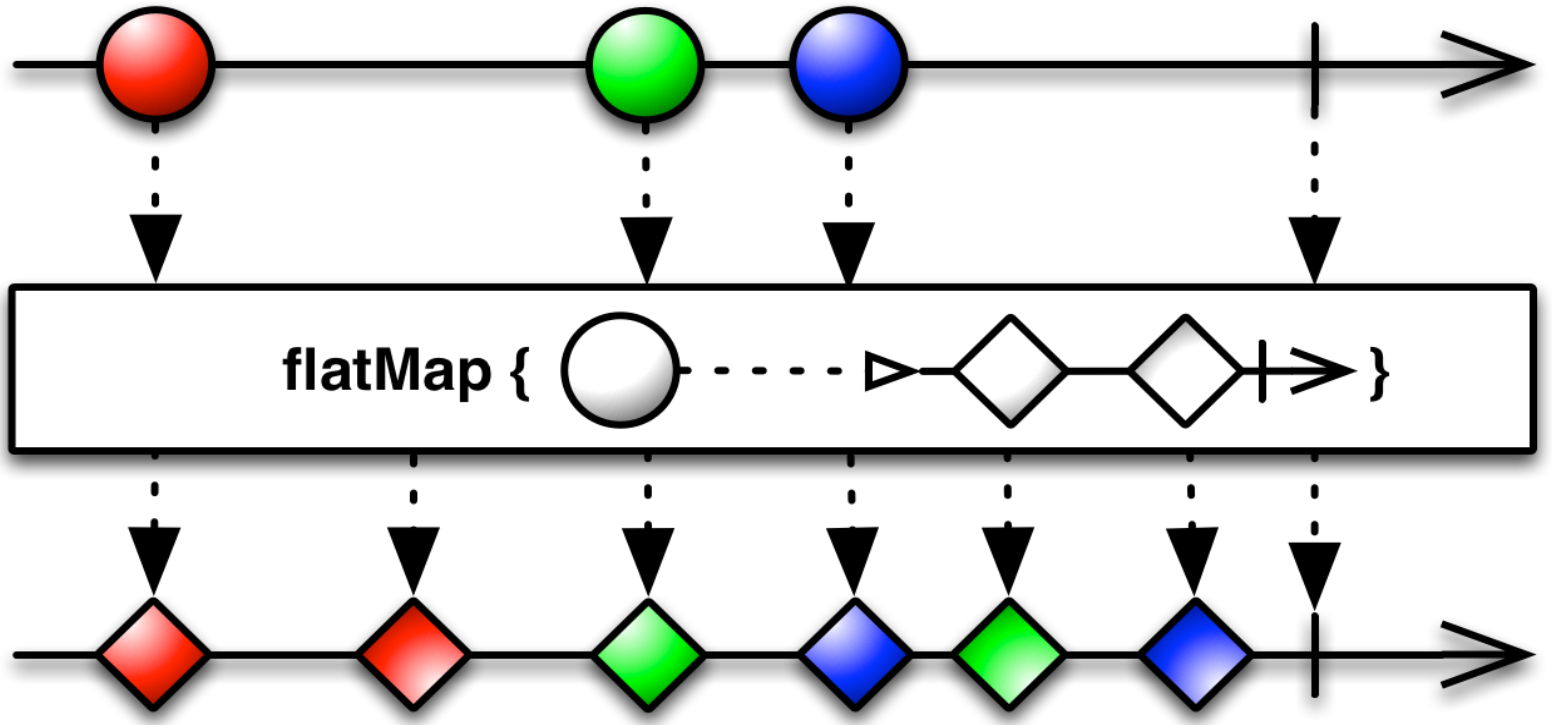
# Observable / Single / Completable

| | Zero | One | Many |
|---|---|---|---|
| Sync | void | T | Iterable<T> |
| Async | Completable | Single<T> | Observable<T> |

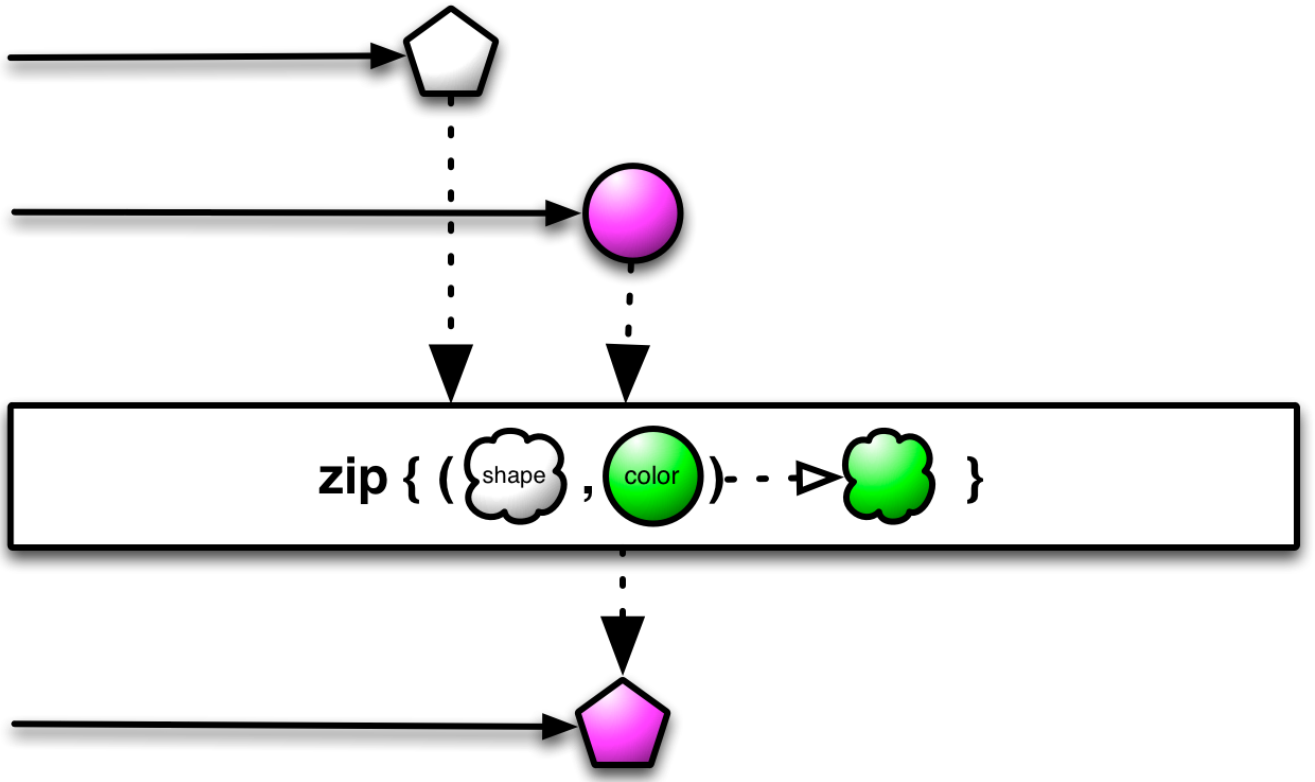
# Observable#map



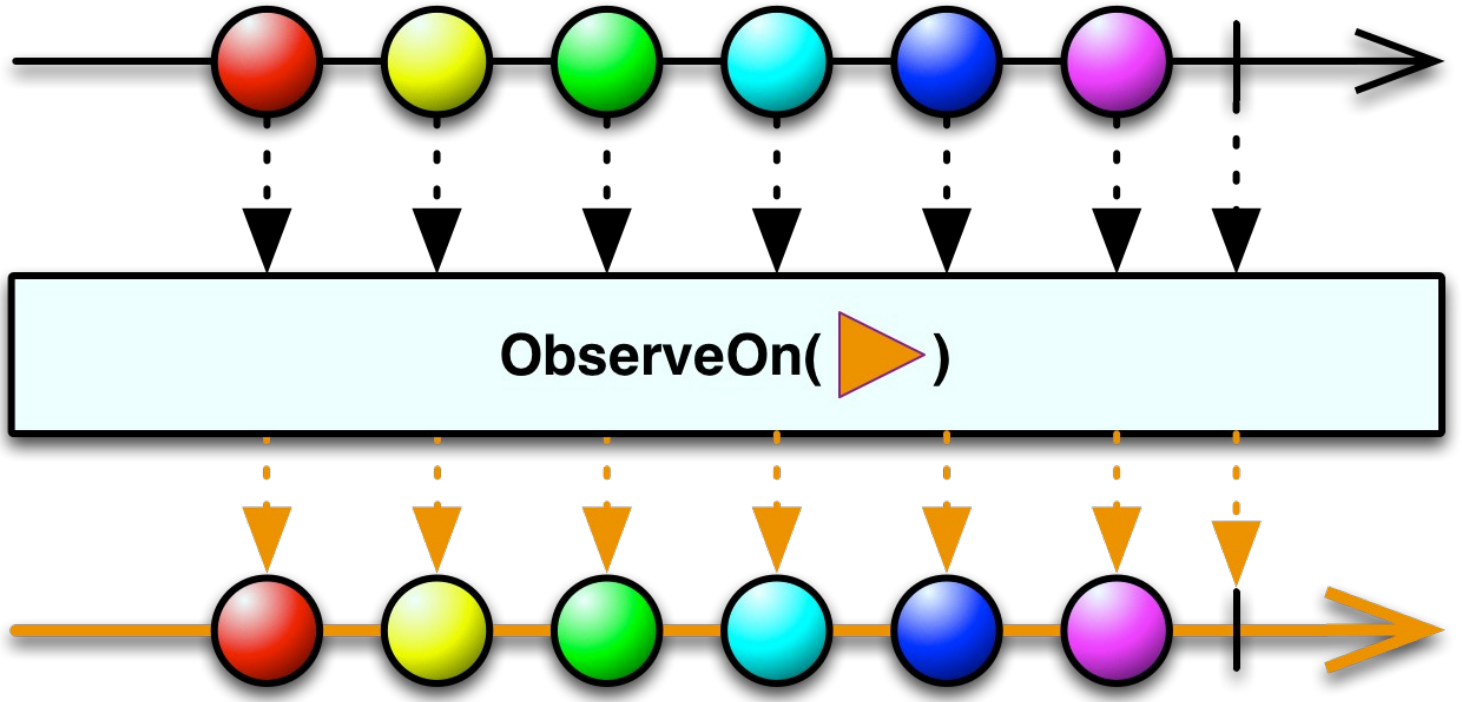
# Observable#flatMap



# Single#zip

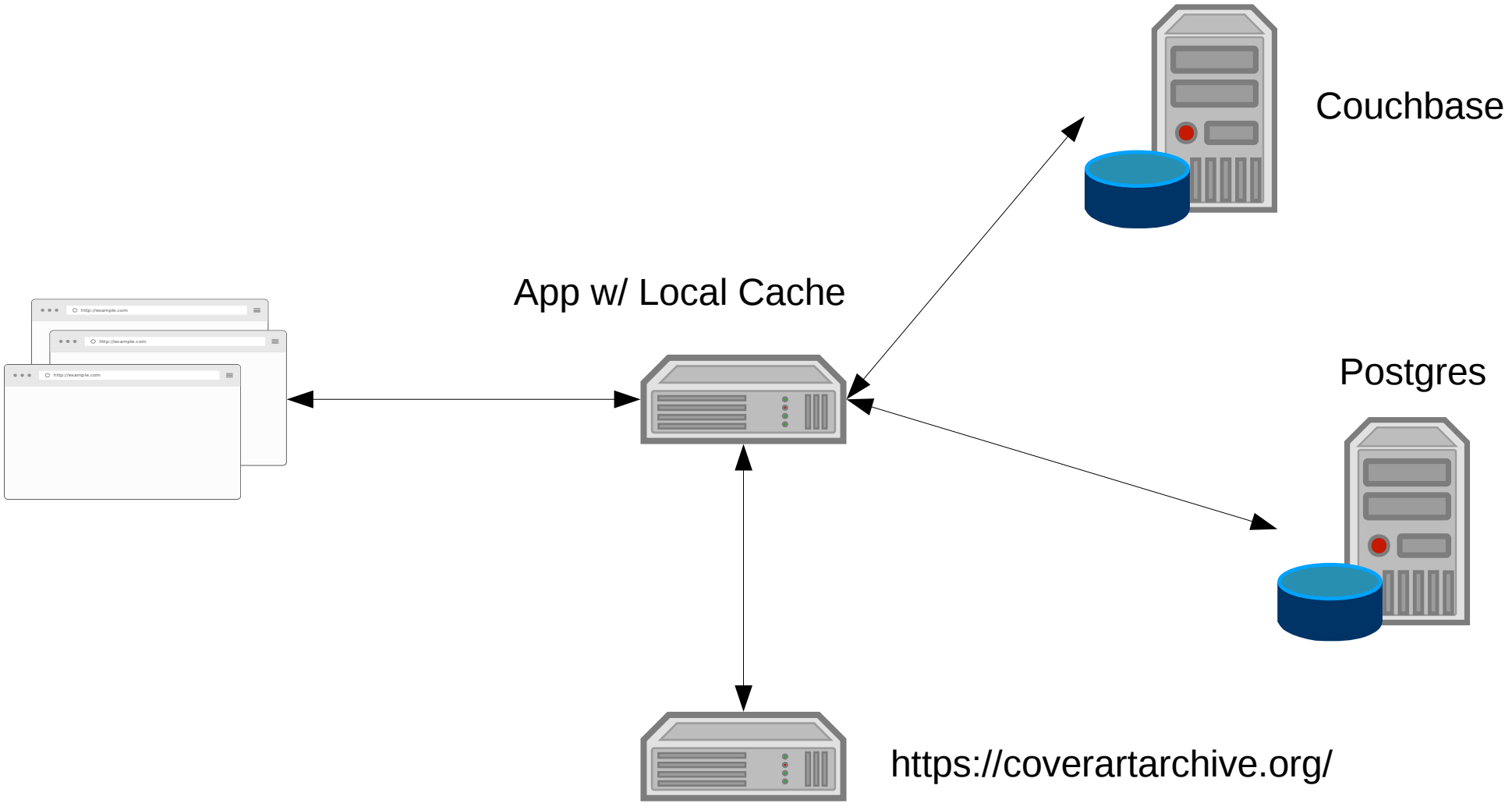


# Observable#observeOn



# Music Store Demo

https://github.com/tsegismont/vertx-musicstore


Couchbase
App w/ Local Cache
Postgres
https://coverartarchive.org/

# Thank you!

http://vertx.io