Erwin de Gier

Reactive Java: The state of the world

github.com/erwindeg

@erwindeg

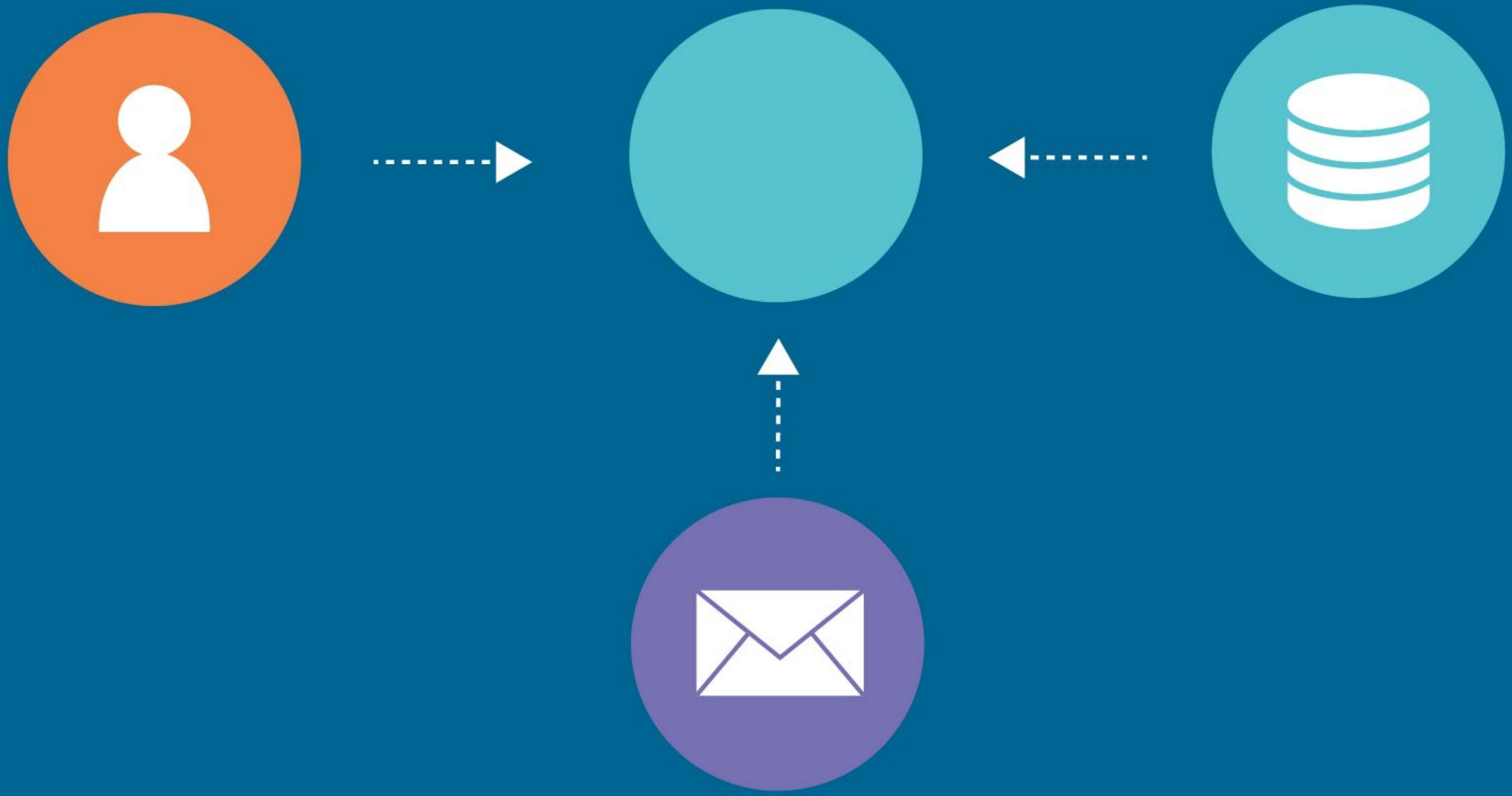
edegier.nl

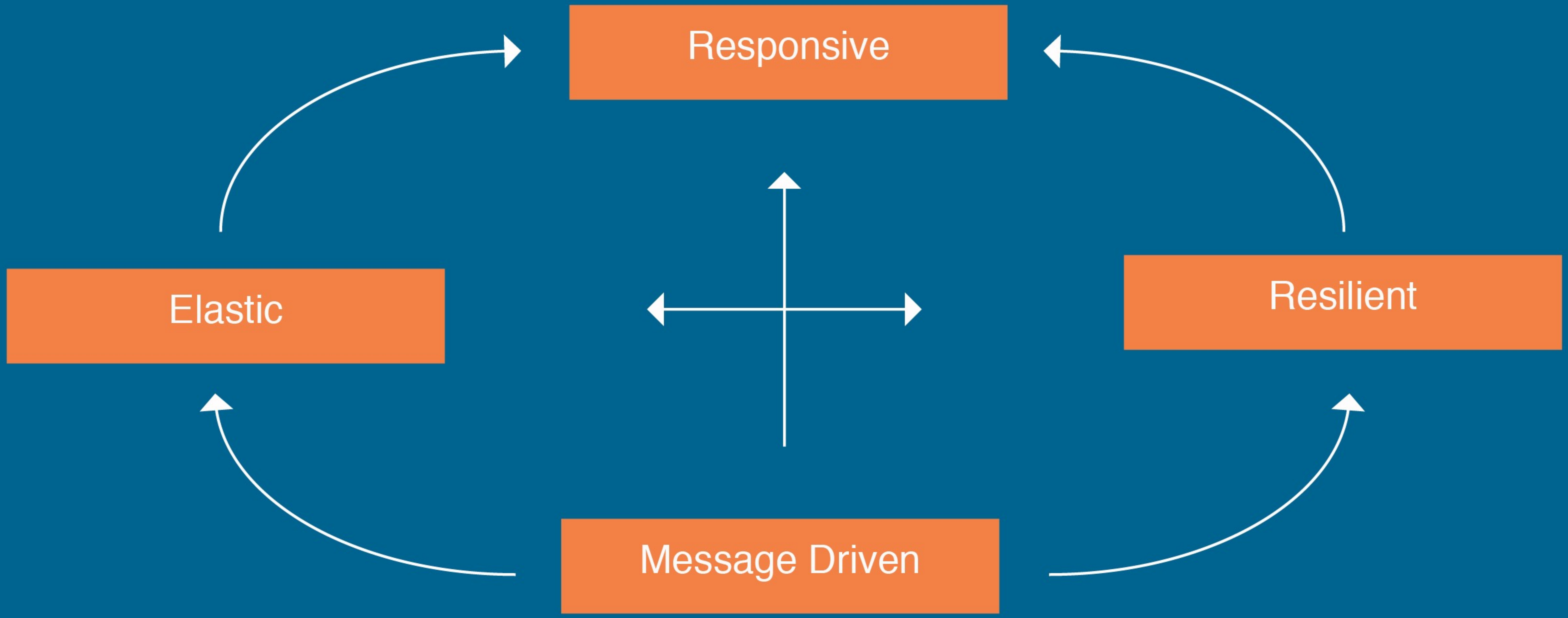
SOGETI

“Unless you can model your entire system synchronously, a single asynchronous source breaks imperative programming”

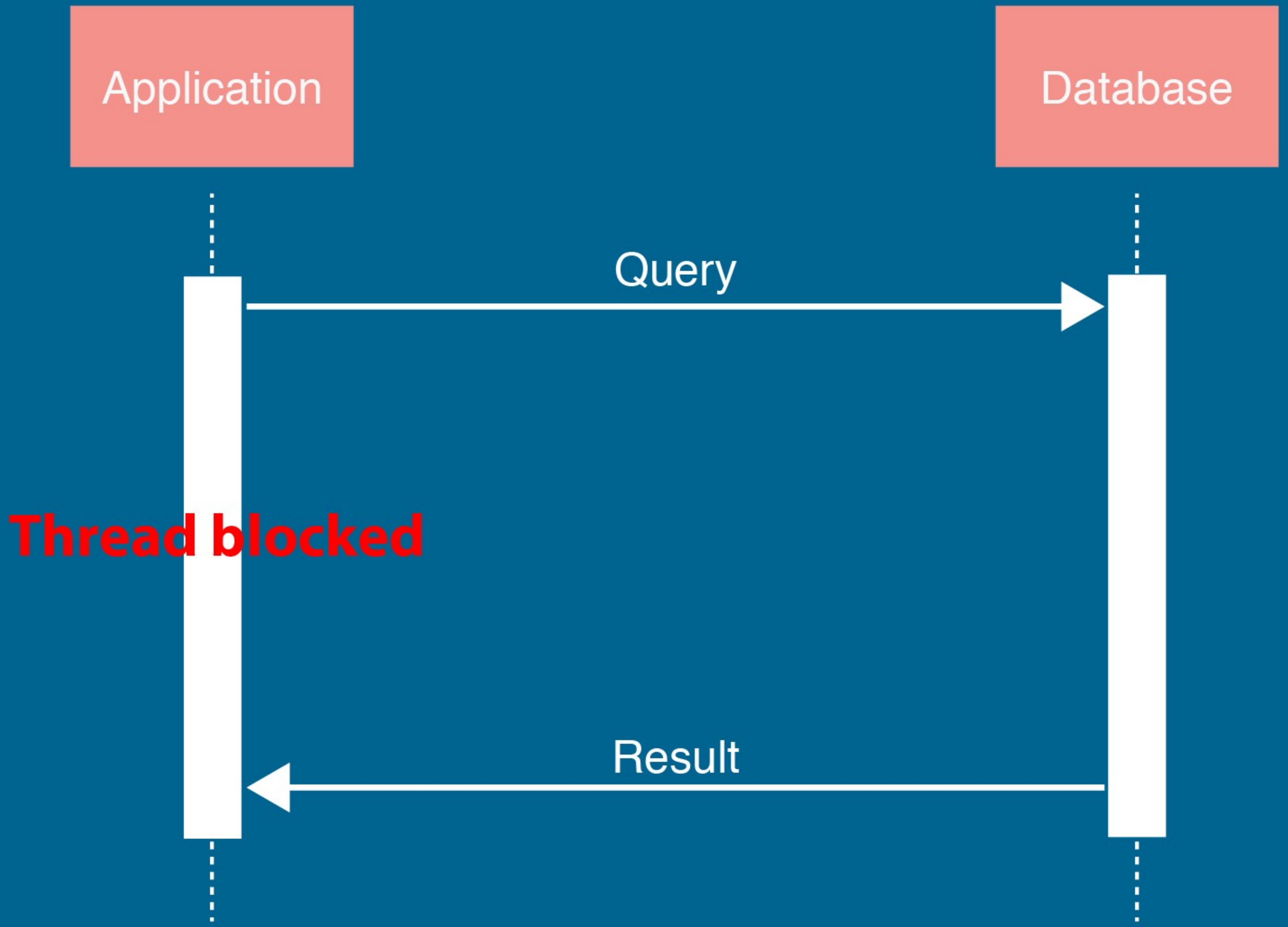
Jake Wharton

Why Reactive?

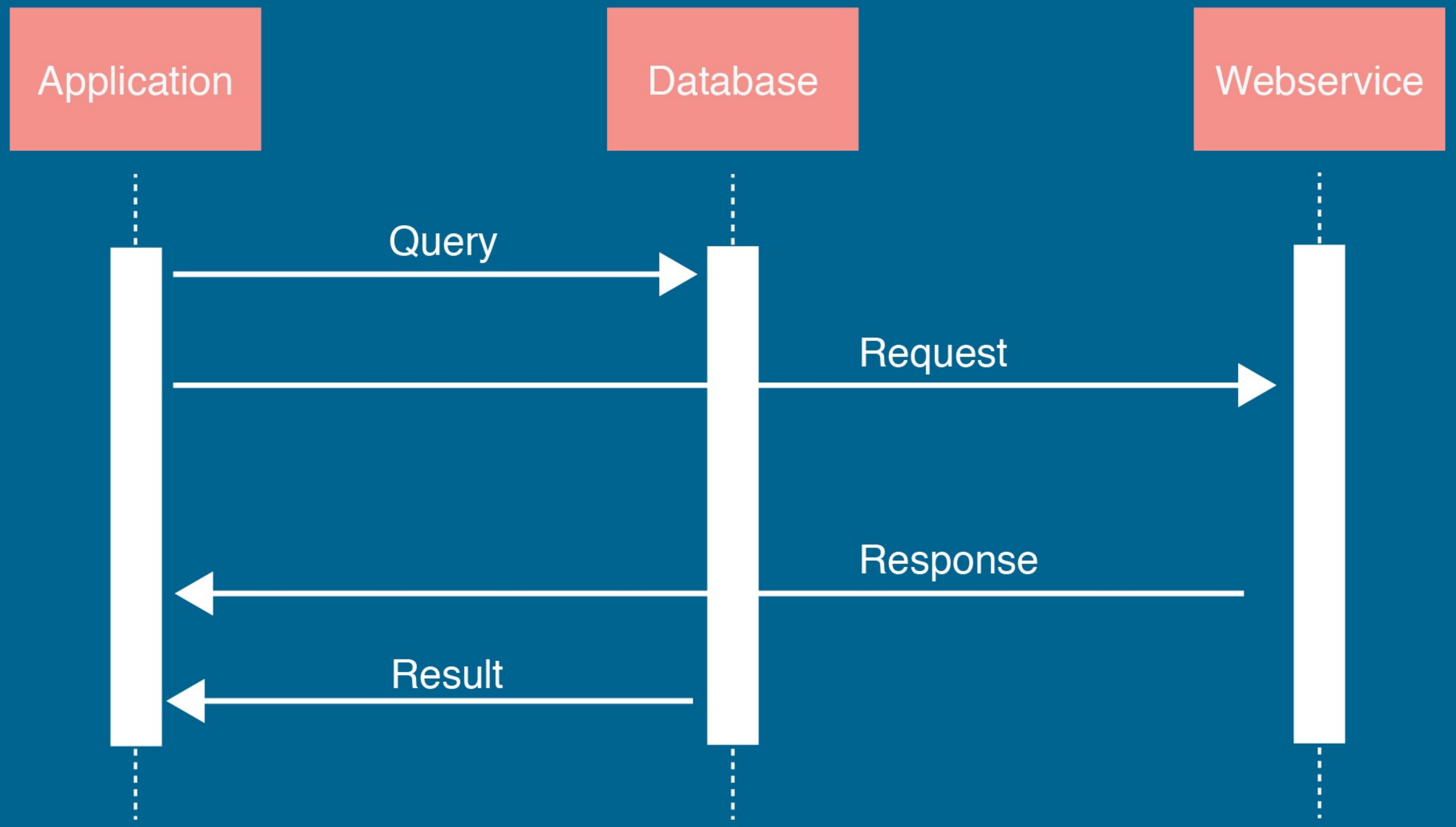



Responsive
Elastic
Resilient
Message Driven

Synchronous



Asynchronous

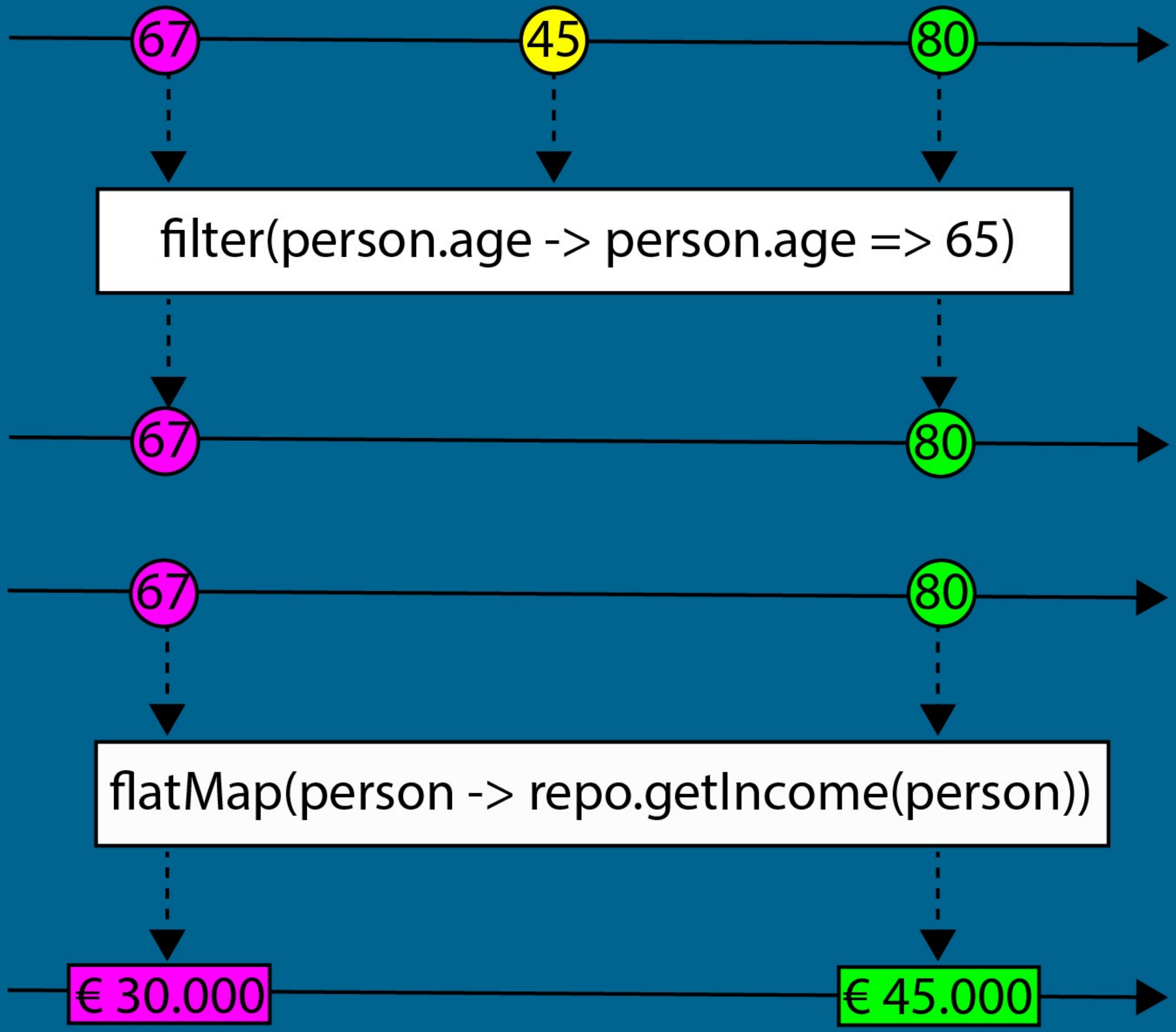


```java
//PersonRepository sync
public List<Person> findByName(String name);

public BigDecimal getIncome(String name);


//PersonRepository async
public void findByName(String name, Callback<List<Person>> persons);

public void getIncome(String name, Callback<BigDecimal> income);
```

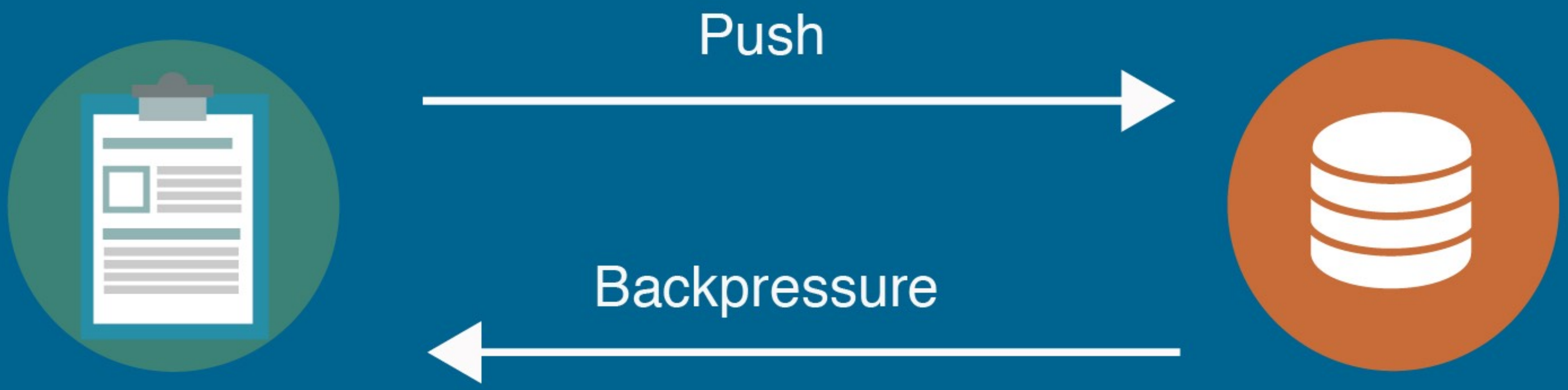
```
//Client call
repository.findByName("Erwin",
    persons -> {
      persons.stream().filter(person -> person.getAge() >= 65)
                      .forEach(person -> {
                         repository.getIncome(person, income ->
                         totalIncome = totalIncome.add(income));
                       });
          }
);
```

```java
//PersonRepository
public Observable<Person> findByName(String name);
public Observable<BigDecimal> getIncome(Person person);

//Client call
repository.findByName("Erwin")
    .filter(person -> person.getAge() >= 65)
    .flatMap(person -> repository.getIncome(person))
    .subscribe(income -> totalIncome = totalIncome.add(income));
```

RxJava

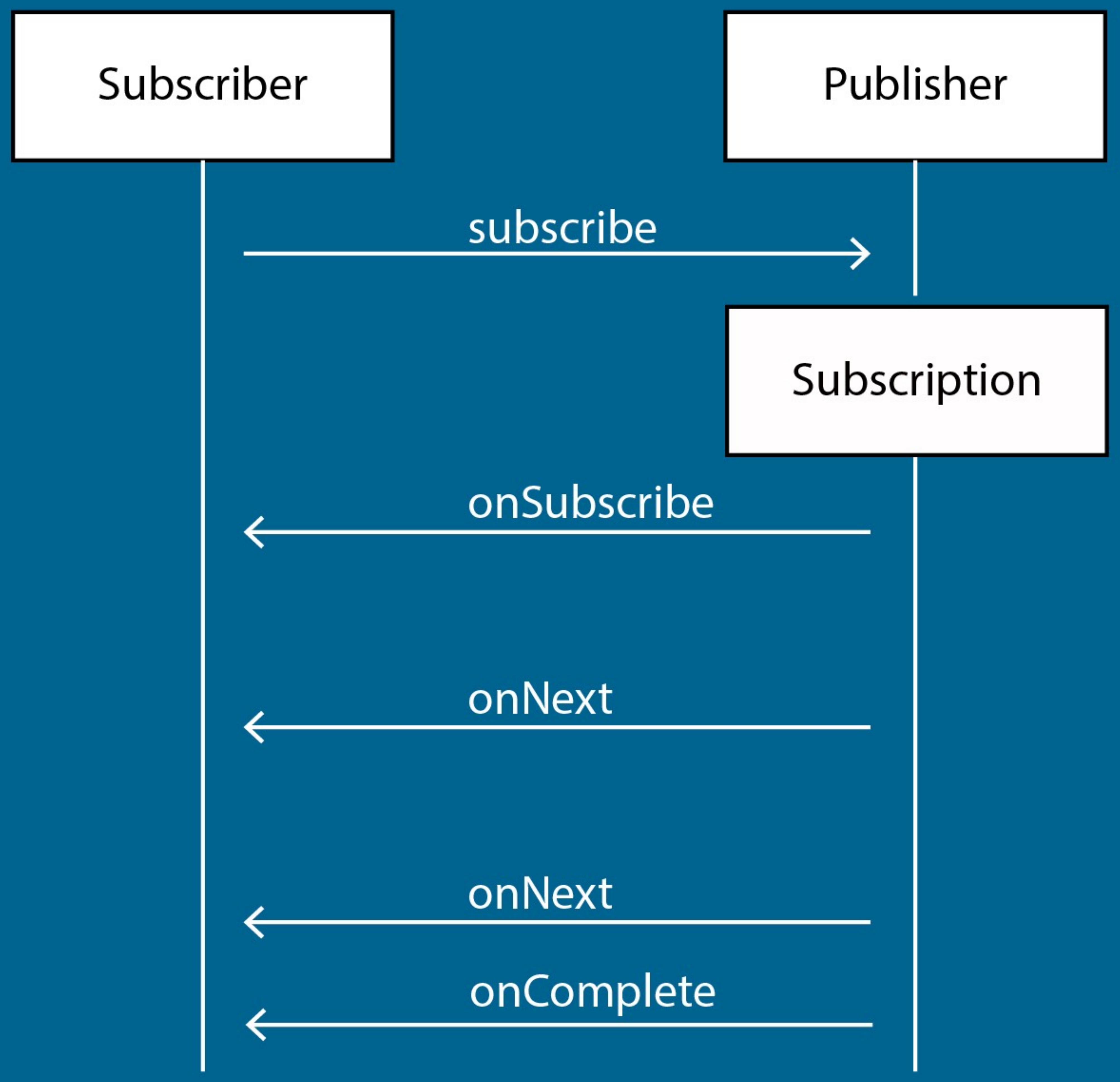

67
45
80
filter(person.age -> person.age => 65)
67
80
67
80
flatMap(person -> repo.getIncome(person))
€ 30.000
€ 45.000

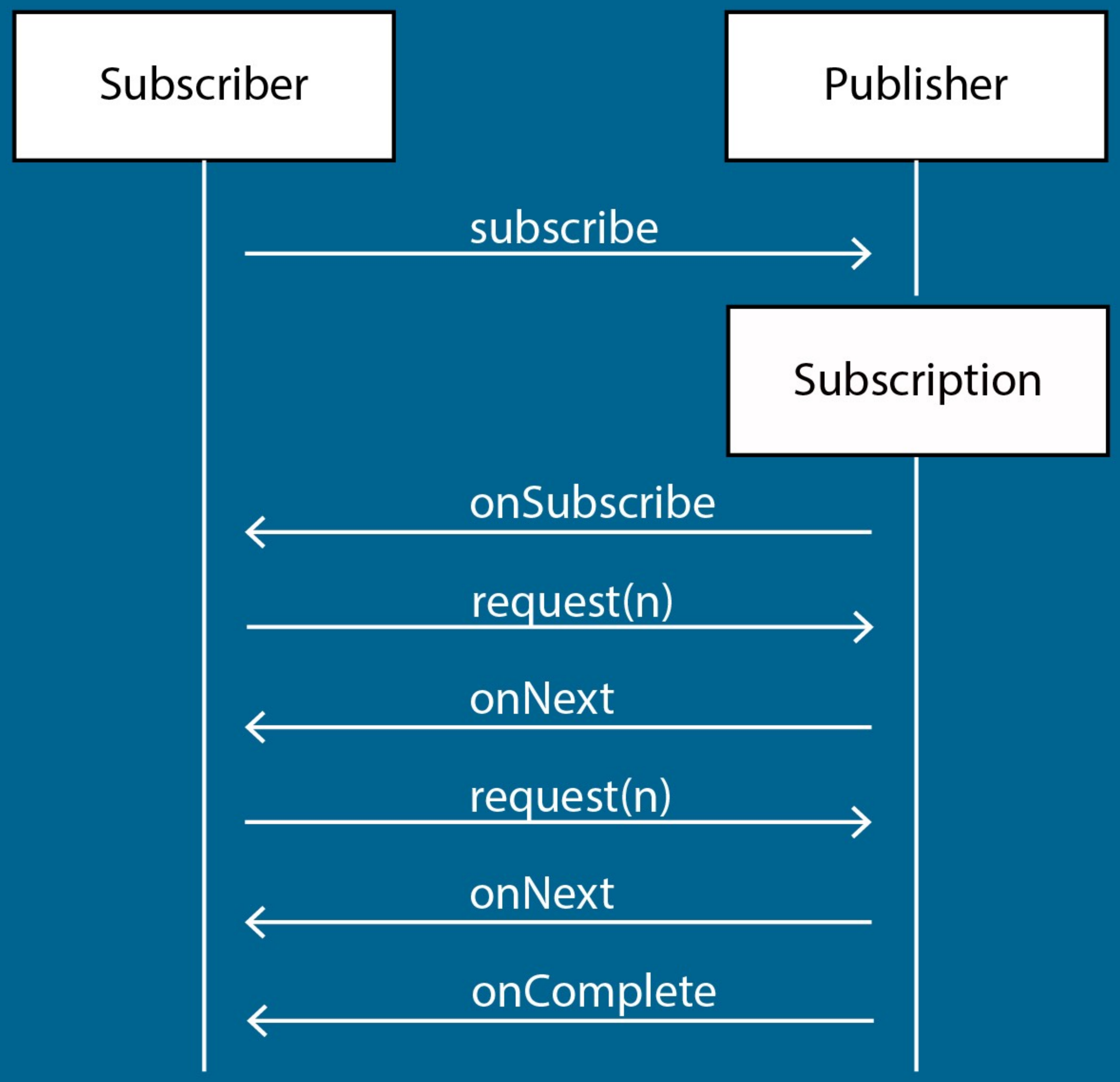
RxJava 2



http://www.reactive-streams.org

Backpressure



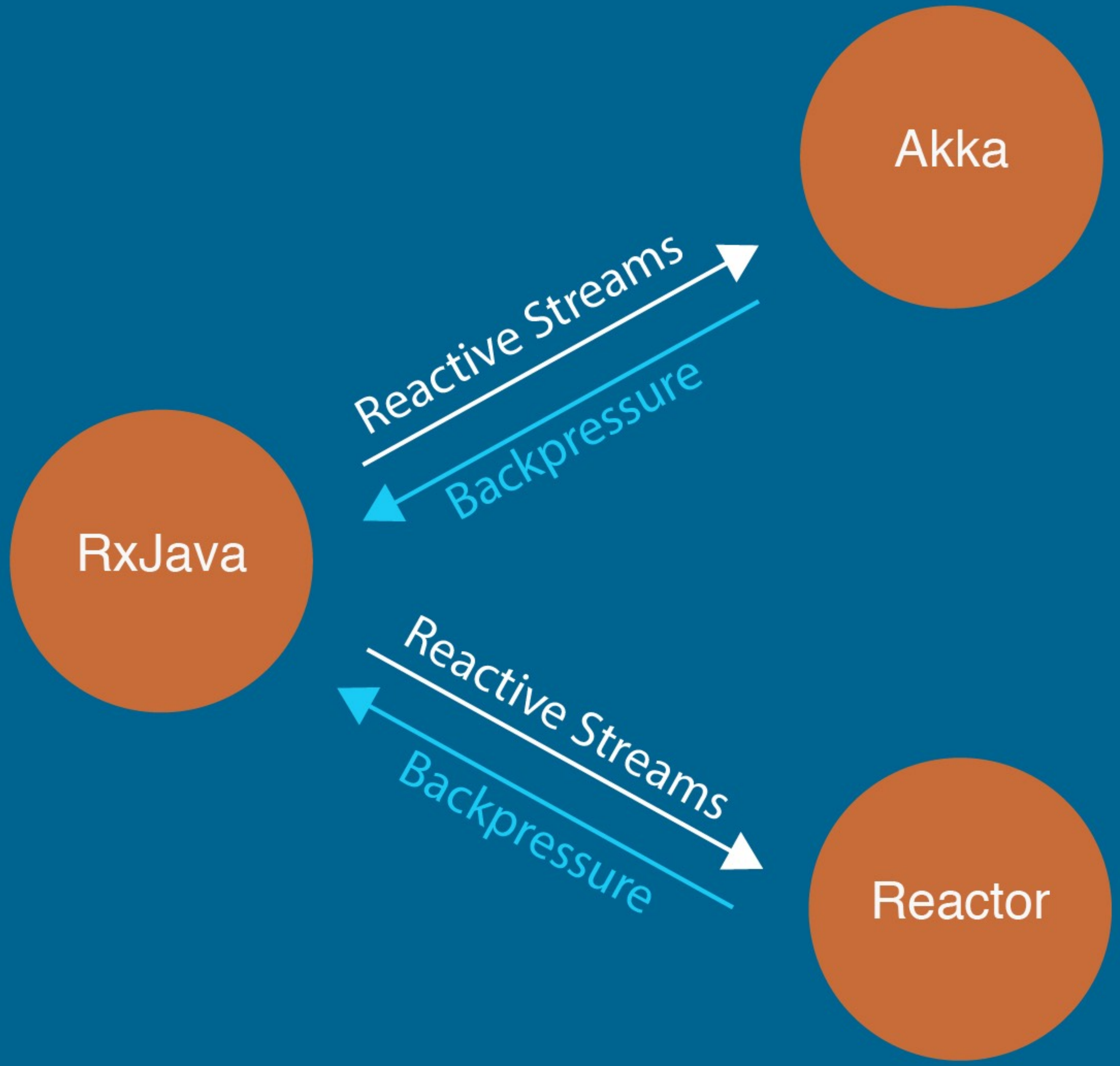


Reactive Streams

- ✓ RxJava 2
- ✓ Project Reactor
- ✓ Akka Streams
- ✓ Java 9 Flow API

Java vs. Reactive Streams

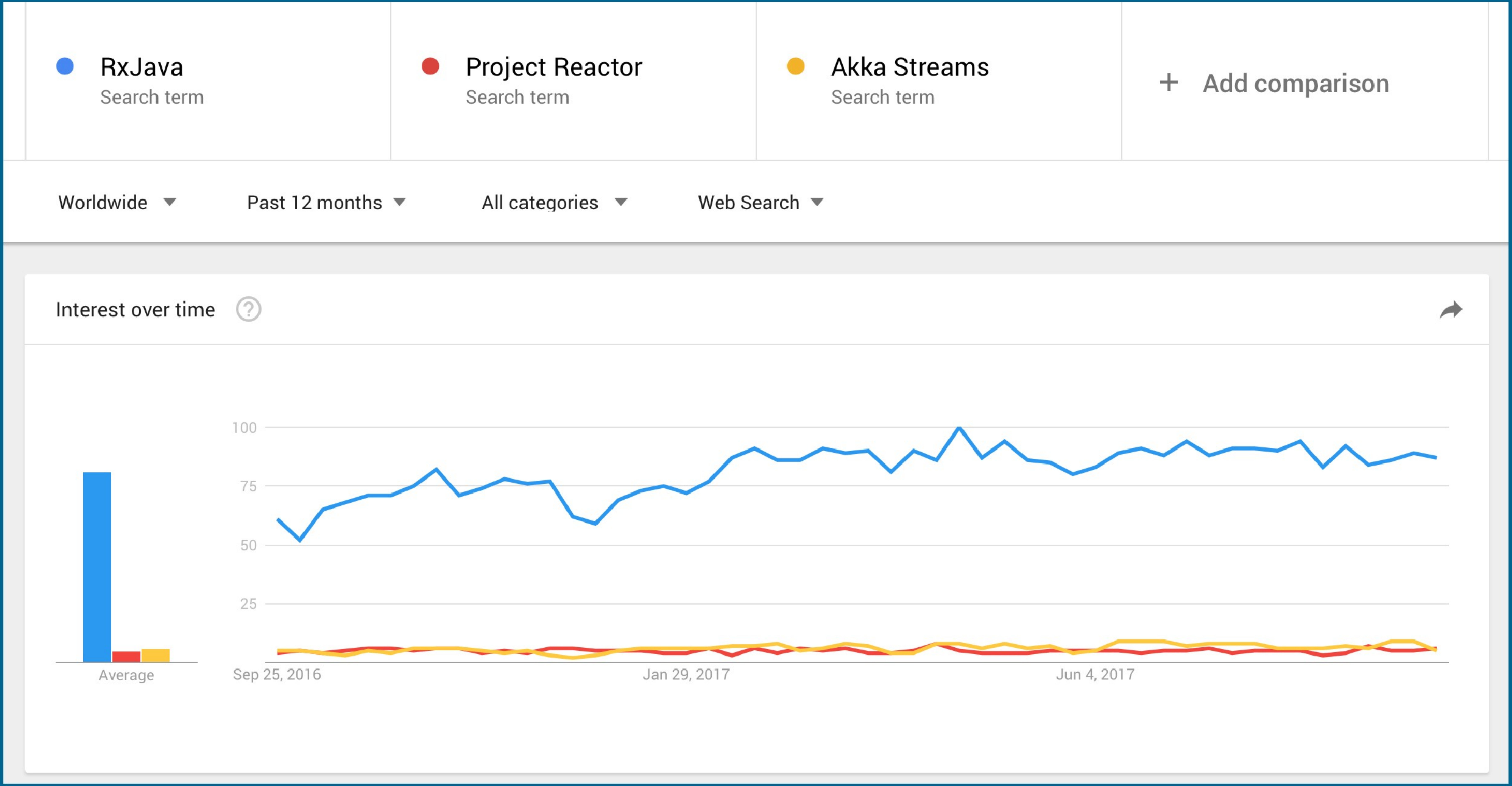
	No Value	Single Value	Multiple Values
Java Blocking	void	T	Iterable<T>
Java Non-blocking	CompletableFuture<Void>	CompletableFuture<T>	CompletableFuture<List<T>>
Reactive Streams	Publisher<Void>	Publisher<T>	Publisher<T>
RxJava	Observable<Void>	Single<T>	Observable<T>
Project Reactor	Mono<Void>	Mono<T>	Flux<T>
Akka Streams	Source<Void>	Source<T>	Source<T>
Java 9 Flow	Flow.Publisher<Void>	Flow.Publisher<T>	Flow.Publisher<T>


Akka
Reactive Streams
Backpressure
RxJava
Reactive Streams
Backpressure
Reactor

Java 9

- ✓ Flow API
- ✓ Interfaces copied from reactive streams
- ✓ Connecting different Rx implementations
- ✓ Easier to use Reactive Frameworks

Popularity

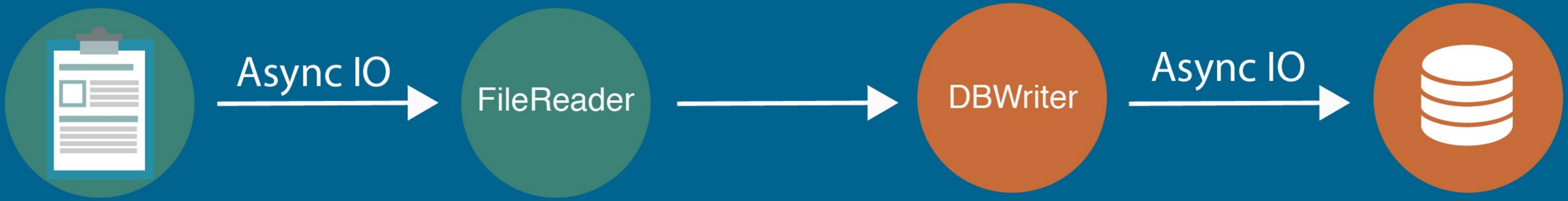


● RxJava
Search term

● Project Reactor
Search term

● Akka Streams
Search term

Async operations

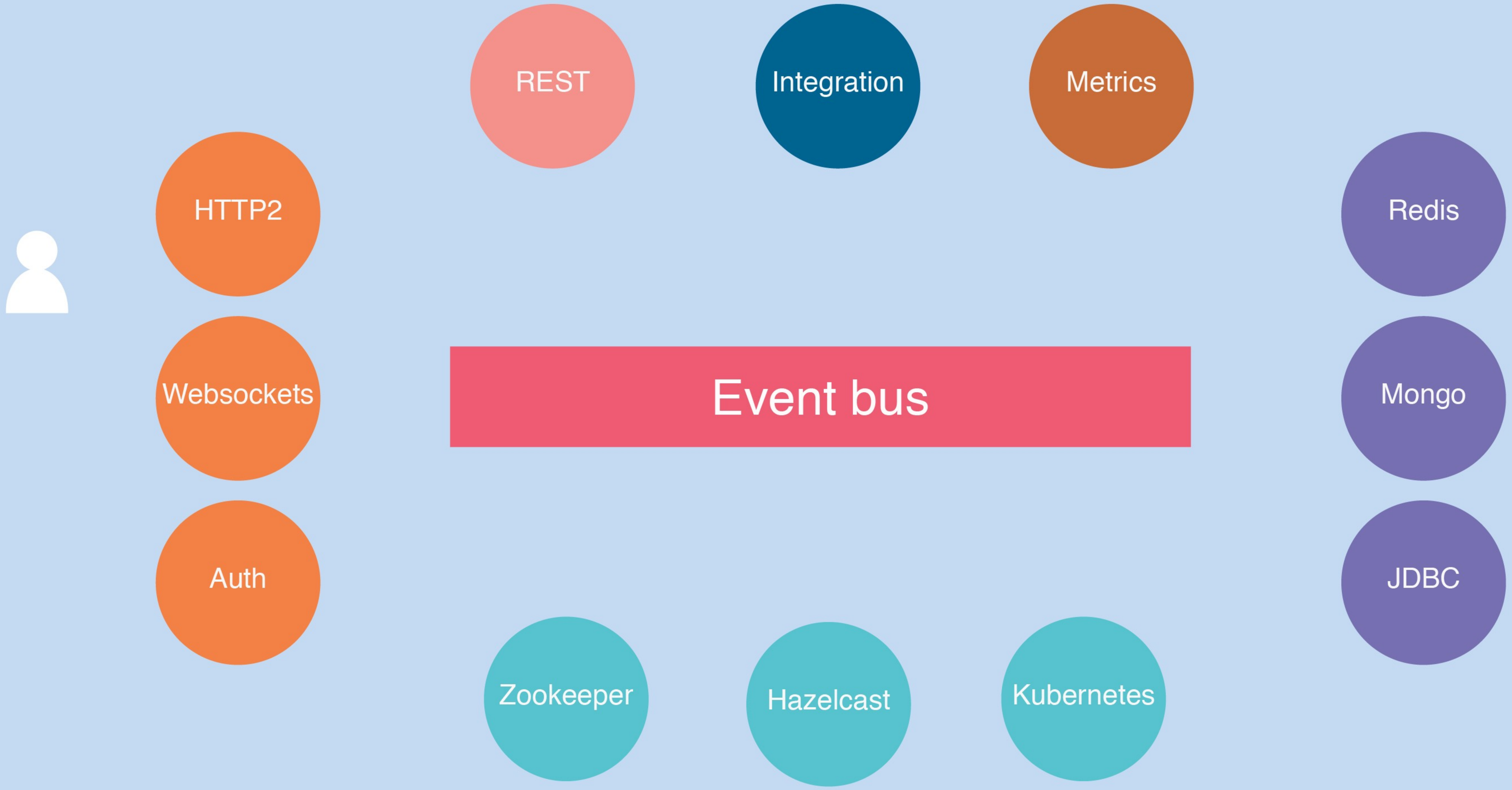


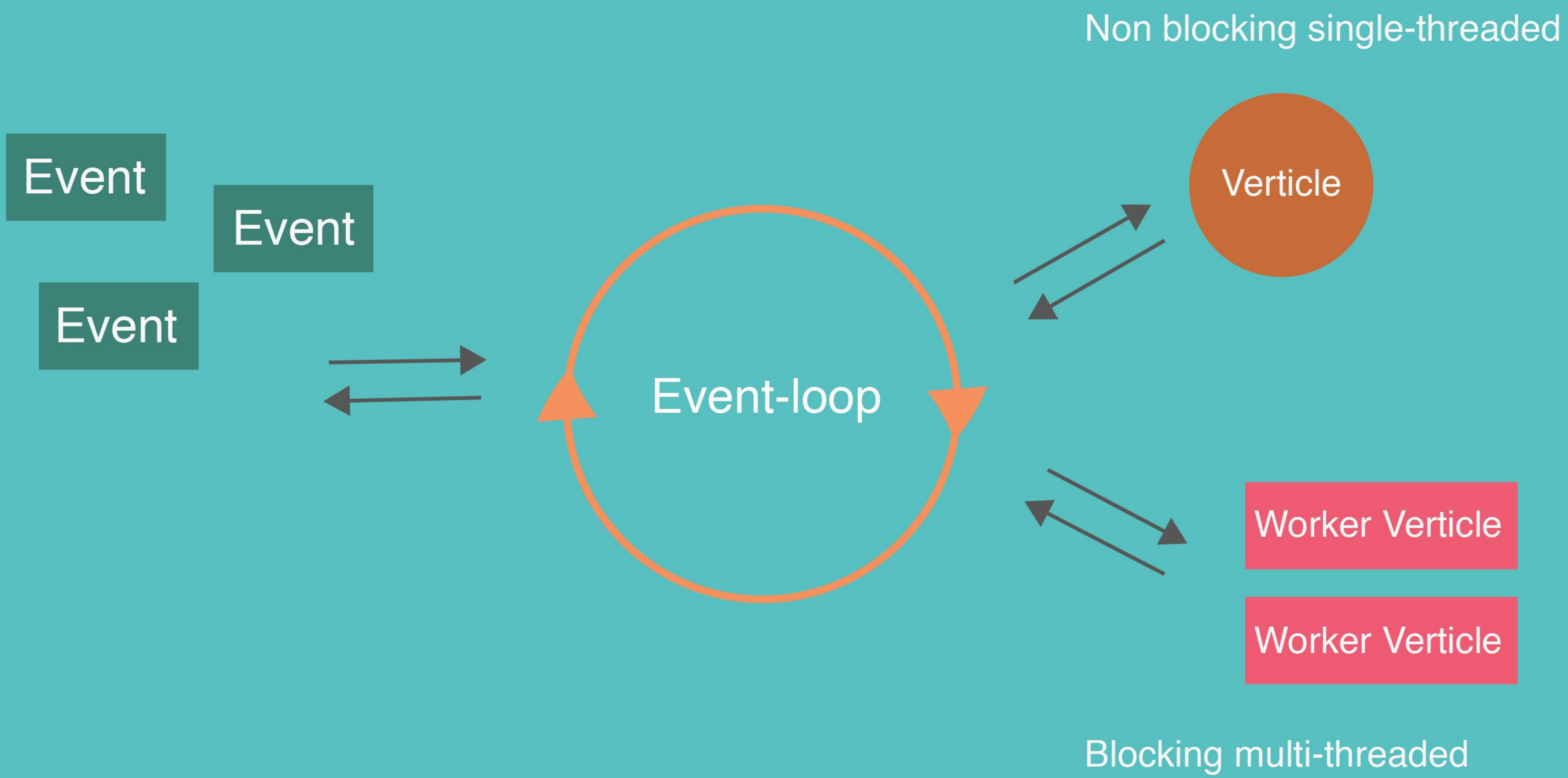
Reactive Frameworks

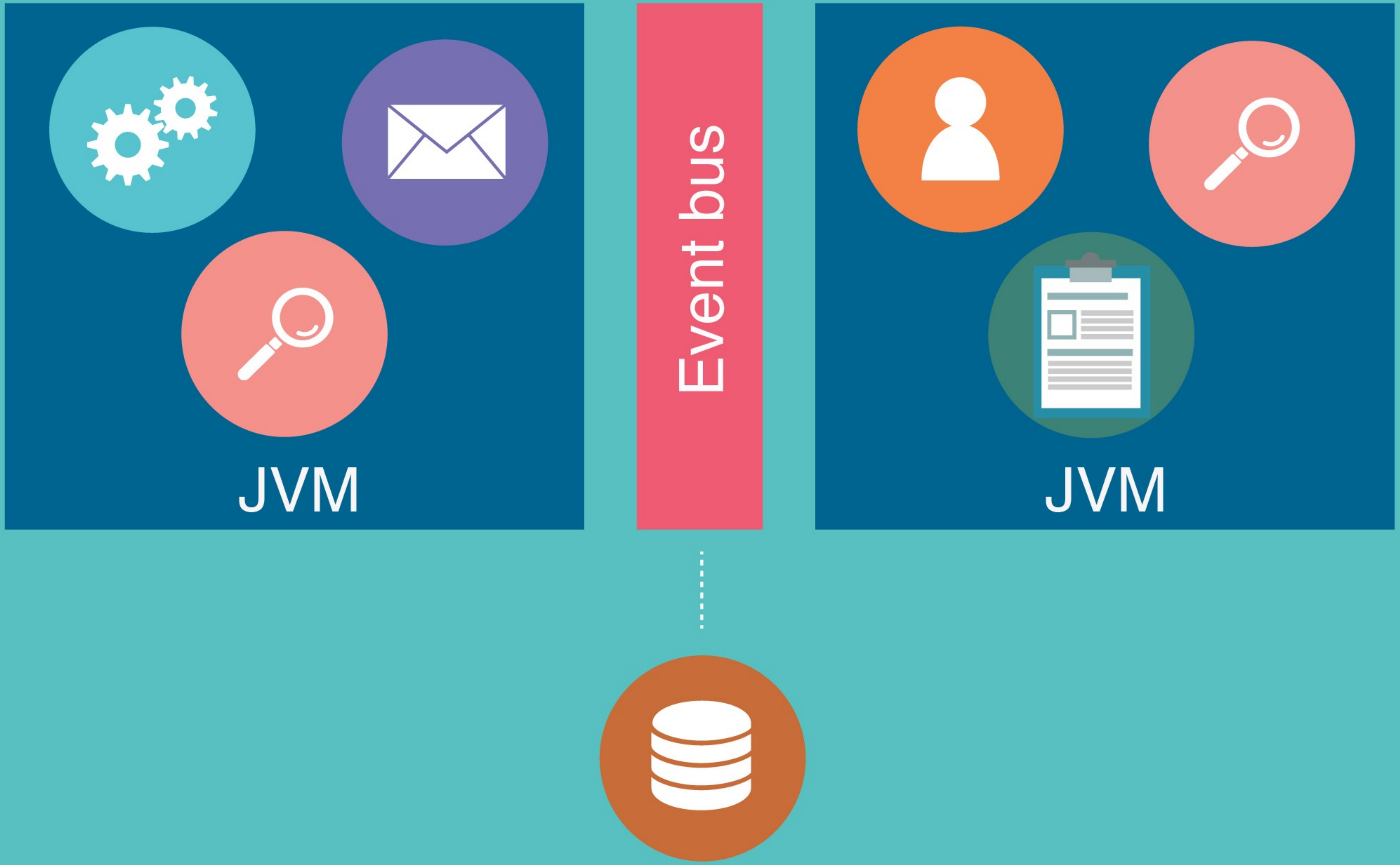
- ✓ Vert.x
- ✓ Spring 5
- ✓ Akka

VERT.X

- ✓ Runnable Jar
- ✓ Reactive
- ✓ Polyglot
- ✓ Distributed


REST
Integration
Metrics
HTTP2
Redis
Websockets
Event bus
Mongo
Auth
JDBC
Zookeeper
Hazelcast
Kubernetes


Non blocking single-threaded
Event
Event
Event
Event-loop
Verticle
Worker Verticle
Worker Verticle
Blocking multi-threaded


JVM
Event bus
JVM

```java
public class HelloWorldVerticle extends AbstractVerticle{

  @Override
  public void start() throws Exception {
    vertx.eventBus().consumer("hello-channel",message -> System.out.println(message.body()));

    vertx.eventBus().send("hello-channel","Hello world!");
  }
}
```

```java
public class HelloWorldRestVerticle extends AbstractVerticle{

  @Override
  public void start() {
    Router router = Router.router(vertx);
    router.get("/hello").handler(routingContext -> {
      routingContext.response()
        .end(new JsonObject().put("message", "Hello World").encode());
    });

    vertx.createHttpServer().requestHandler(router::accept).listen(8080);
  }
}
```

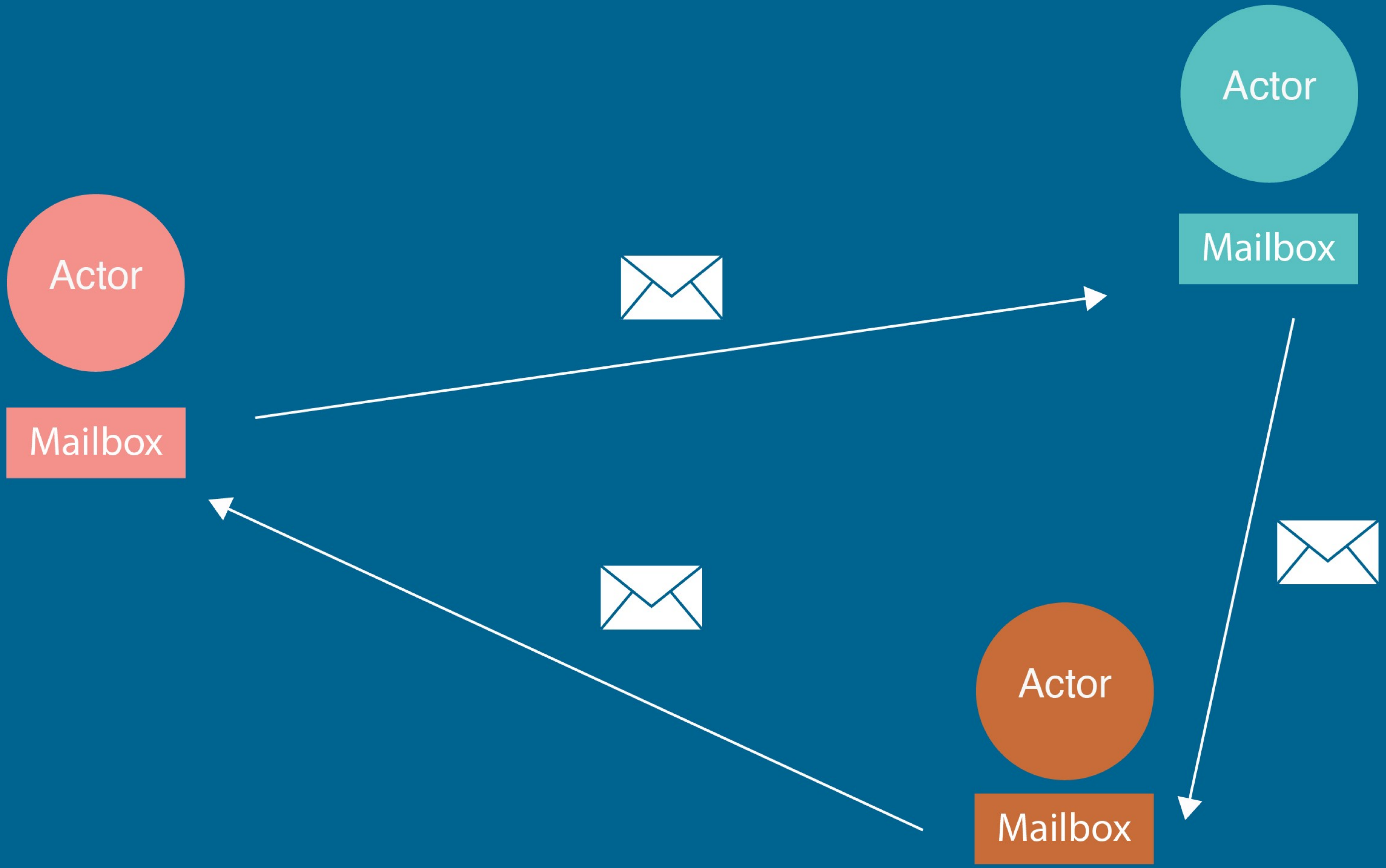
Spring 5

- ✓ Spring Webflux
- ✓ Project Reactor
- ✓ Reactive Data Repositories
- ✓ Project Reactor event bus

```java
@RestController
public class HelloController {

  @GetMapping("/hello")
  Flux<String> hello() {
    return ServerResponse.ok().body(fromObject("Hello World"));
  }
}


@RestController
class PersonController {
  private final PersonRepository people;

  @GetMapping("/people")
  Flux<String> namesByLastname(@RequestParam Mono<String> lastname) {

    Flux<Person> result = repository.findByLastname(lastname);
    return result.map(it -> it.getFullName());
  }
}
```

AKKA

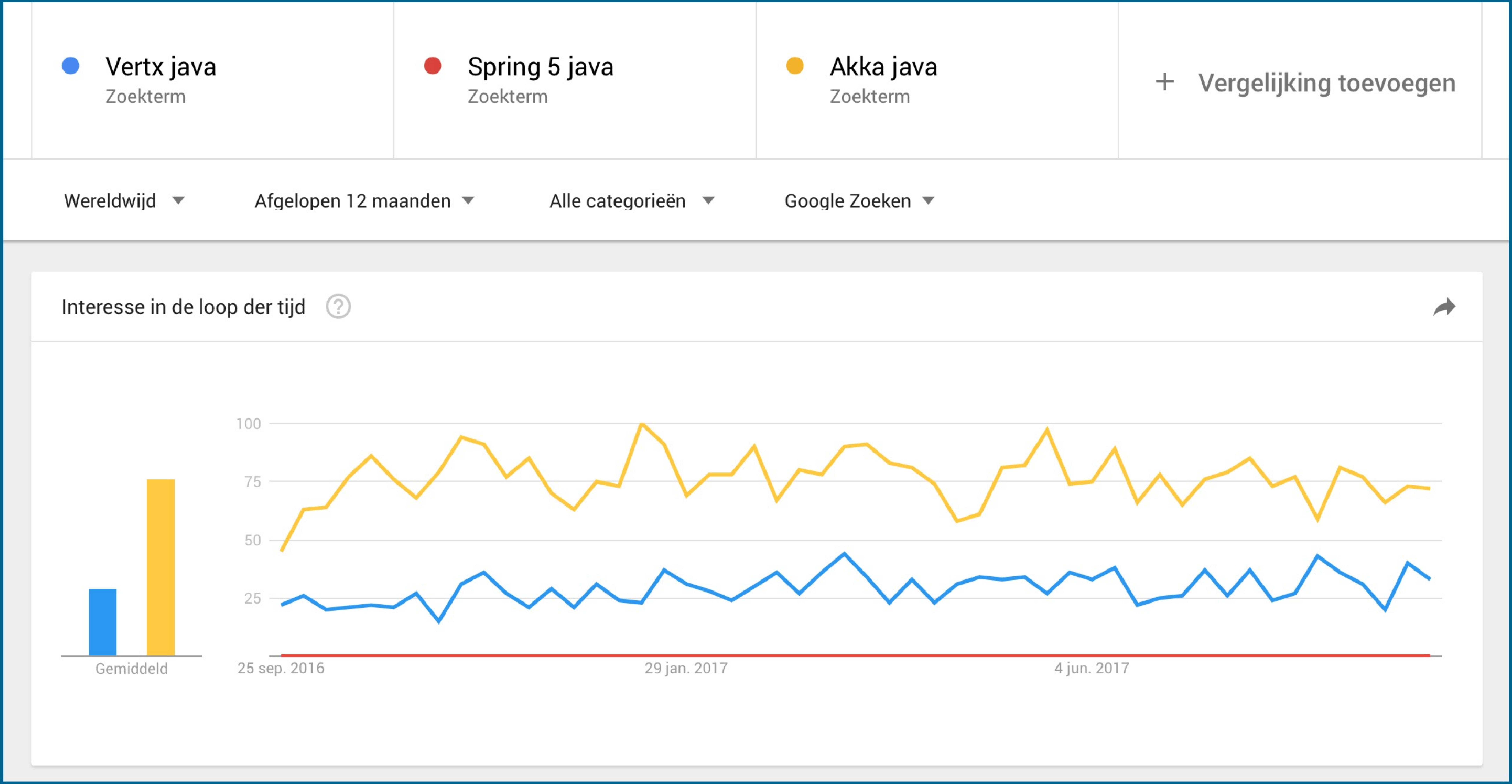
- ✓ Actor model
- ✓ Akka HTTP
- ✓ Scala
- ✓ Message driven


Actor
Mailbox
Actor
Mailbox
Actor
Mailbox

```java
public class HelloWorld extends UntypedActor {

  @Override
  public void preStart() {
    // create the greeter actor
    final ActorRef greeter = getContext().actorOf(Props.create(Greeter.class), "greeter");
    // tell it to perform the greeting
    greeter.tell(Greeter.Msg.GREET, getSelf());
  }

  @Override
  public void onReceive(Object msg) {
      getContext().stop(getSelf());
  }
}

public class Greeter extends UntypedActor {

  @Override
  public void onReceive(Object msg) {
      System.out.println("Hello World!");
      getSender().tell(Msg.DONE, getSelf());
  }
}
```

```java
public class HttpServer extends HttpApp {

    public static void main(String[] args) throws IOException {
        ActorSystem system = ActorSystem.create();

        new HttpServer().bindRoute("localhost", 8080, system);
    }
    @Override
    public Route createRoute() {
        Route helloRoute = handleWith((ctx)
                -> ctx.complete("Hello World!"));

        return route(get(path("hello").route(helloRoute)));
    }
}
```

Popularity



● Vertx java
Zoekterm

● Spring 5 java
Zoekterm

● Akka java
Zoekterm

+ Vergelijking toevoegen

Wereldwijd ▼ Afgelopen 12 maanden ▼ Alle categorieën ▼ Google Zoeken ▼

Interesse in de loop der tijd

Vert.x vs Spring vs Akka

VERT.X

spring by Pivotal™

akka

Landscape overview

	Event model	Annotations	Actor model
Framework	VERT.X	spring by Pivotal	akka
API	RxJava	PROJECT REACTOR	akka streams

“Unless you can model your entire system synchronously, a single asynchronous source breaks imperative programming”

Jake Wharton