AKTUELLE TECHNOLOGIEN ZUR ENTWICKLUNG VERTEILTER JAVA-ANWENDUNGEN

# REACTIVE WEB-BACKENDS MIT SPRING WEBFLUX

# AGENDA



- Motivation
- Konzepte des Reaktive Programmierens
  - Grundlagen
  - Reaktive Systeme
  - Project Reactor
- Das Spring WebFlux-Modul –> Live Coding
- Fazit

Repository für Live Coding: https://github.com/theexiile1305/webflux-demo

# MOTIVATION

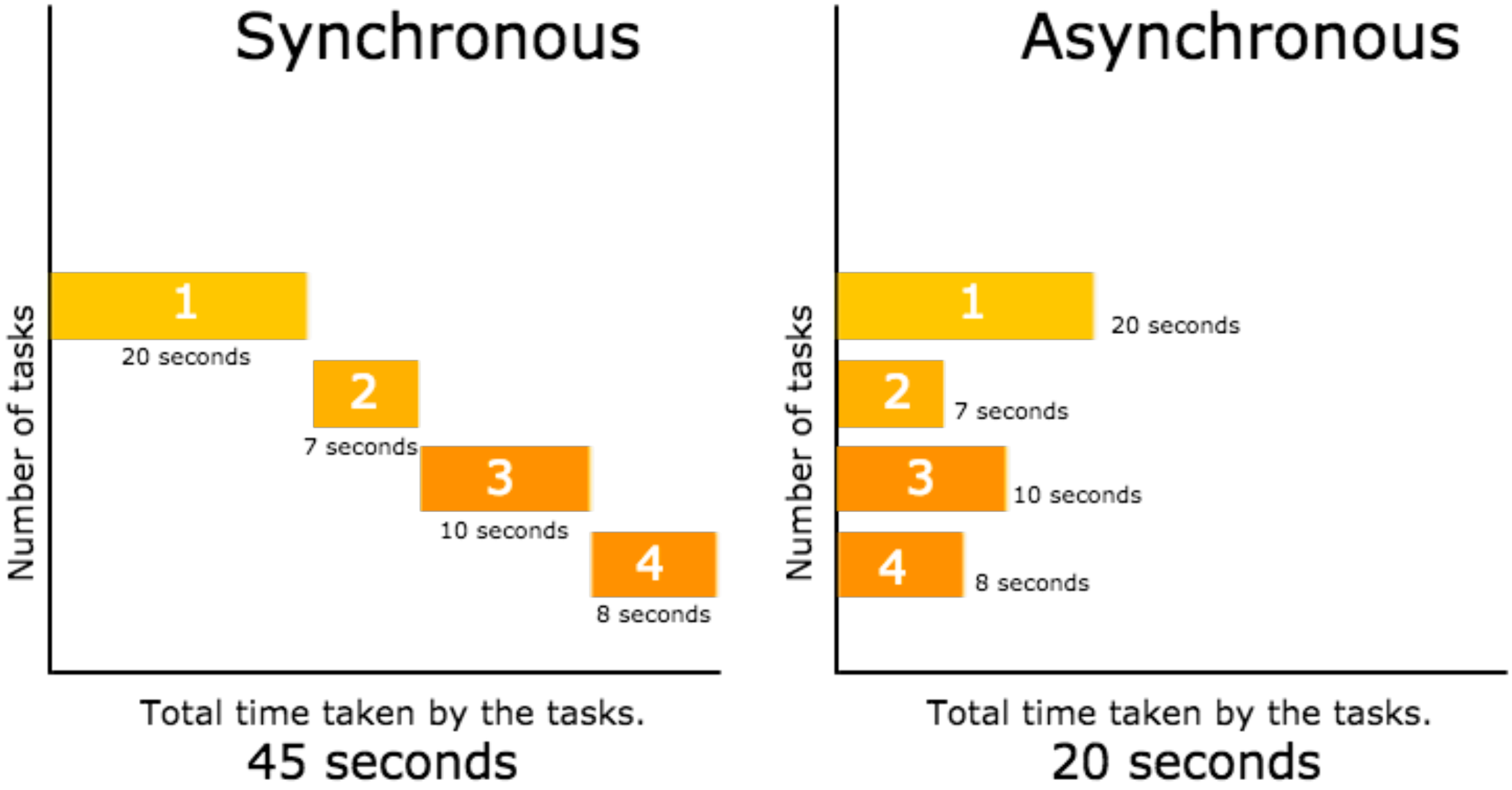
# UBIQUITOUS COMPUTING

„IN THE 21ST CENTURY THE TECHNOLOGY REVOLUTION WILL MOVE INTO THE EVERYDAY, THE SMALL AND THE INVISIBLE.“ [1]

— Mark Weiser (1952 - 1999)

# KONZEPTE DES REAKTIVEN PROGRAMMIERENS

# SYNCHRONE VS. ASYNCHRONE VERARBEITUNG VON REQUESTS




Synchrone und Asynchrone Verarbeitung von Tasks [2]

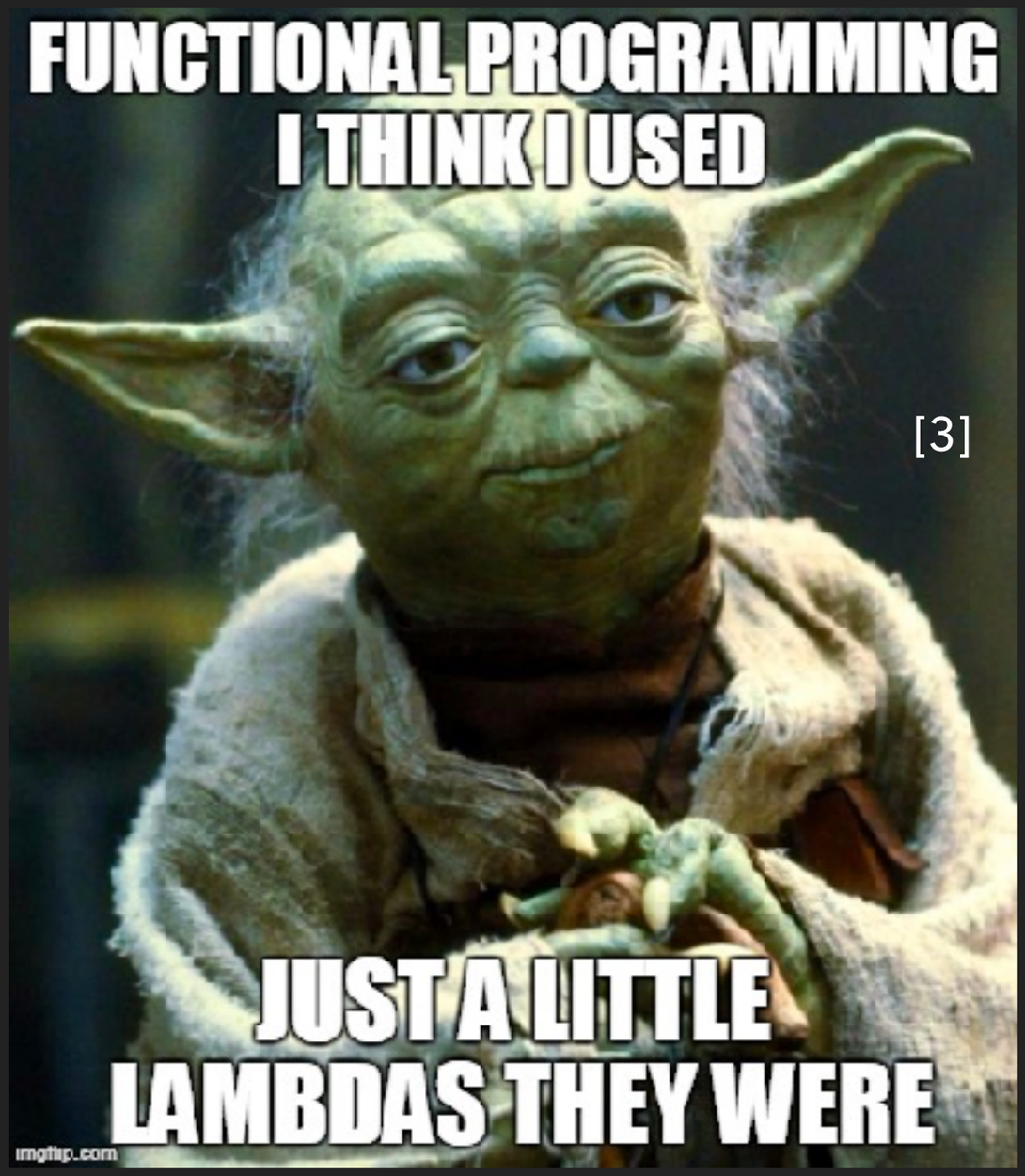
# GRUNDLAGEN

Multiprozessor-Architektur erfordert parallele Verarbeitung

Sichere Implementierung

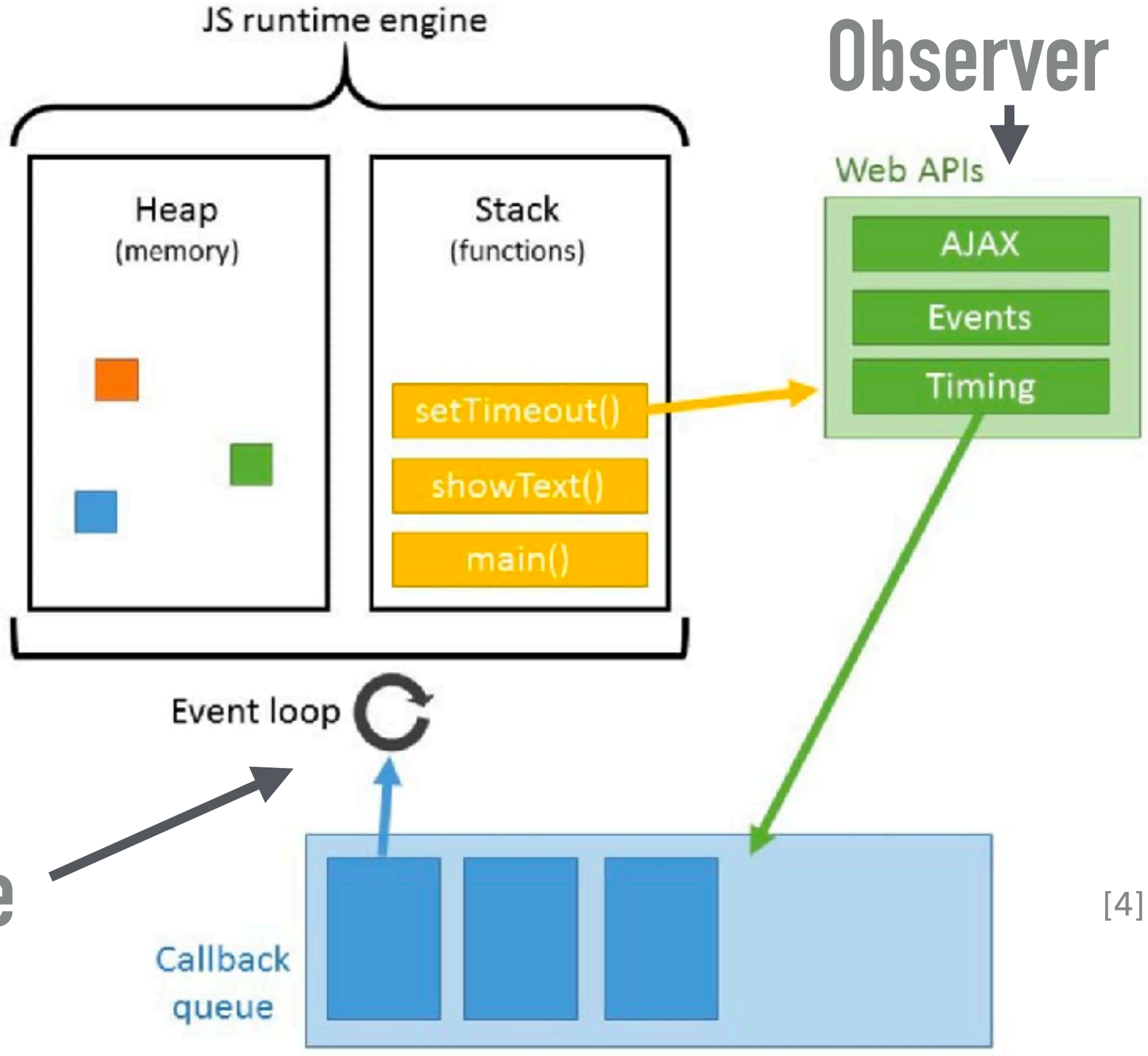
Imperativ **mit** Mehraufwand

Funktional **ohne** Mehraufwand




[3]

# GRUNDLAGEN

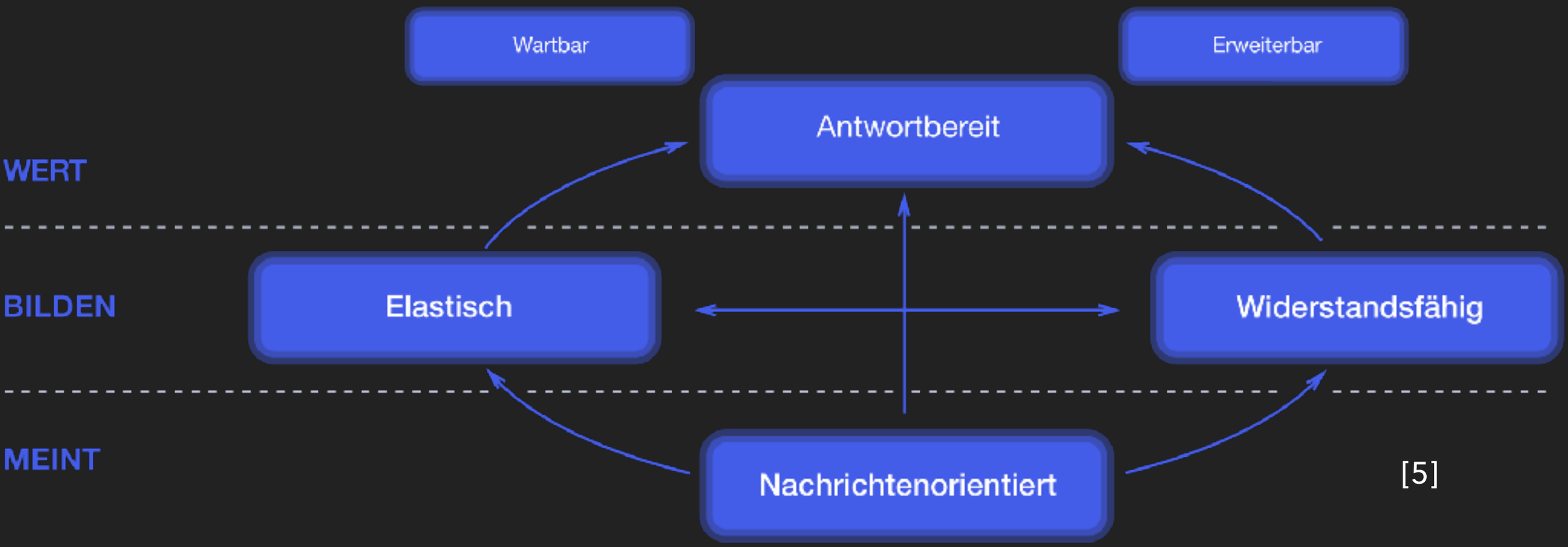



[4]

KONZEPTE DES REAKTIVEN PROGRAMMIERENS

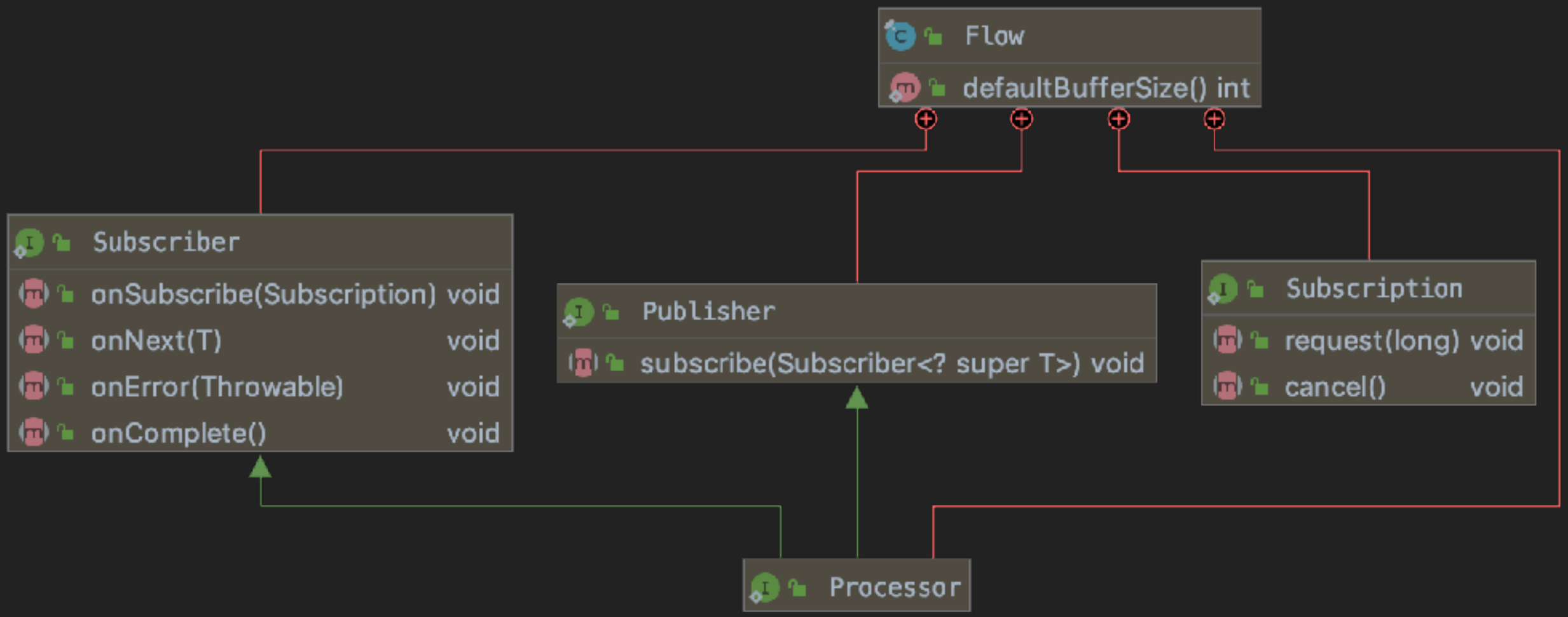
# GRUNDLAGEN – BEISPIEL

# REAKTIVE SYSTEME – ANFORDERUNGEN

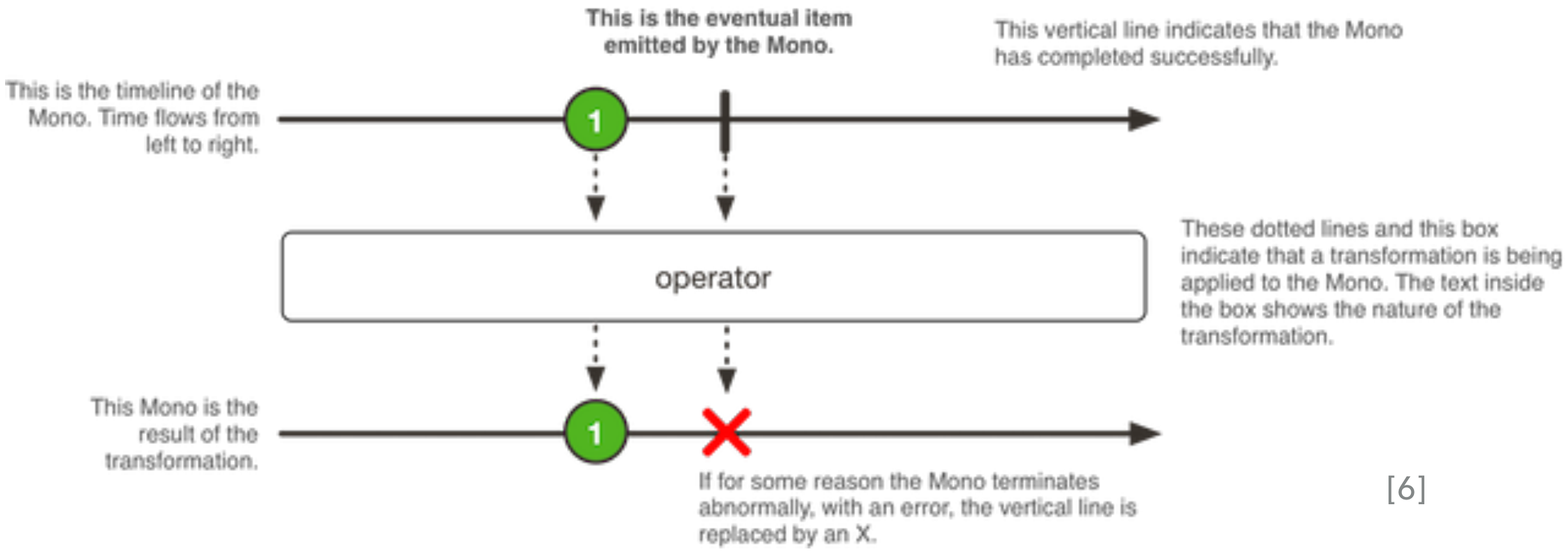



[5]

# REAKTIVE SYSTEME – REACTIVE STREAMS DES JDK9

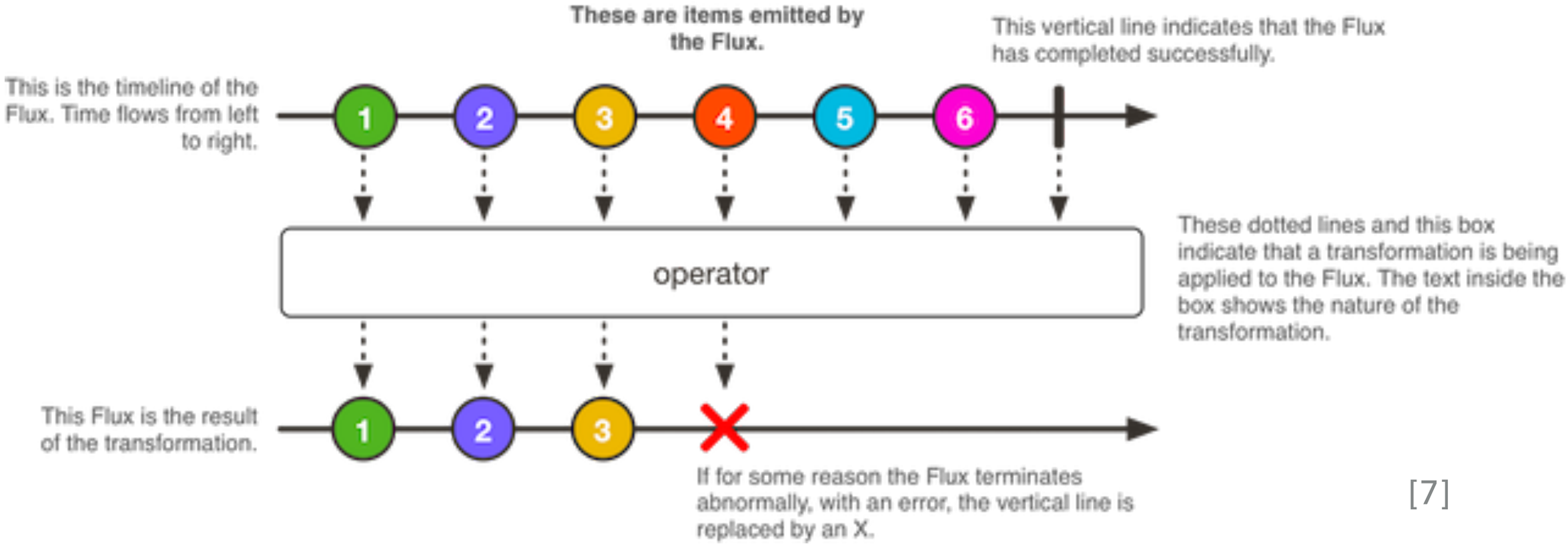


# PROJECT REACTOR – MONO<T>




[6]

# PROJECT REACTOR – FLUX<T>




[7]

KONZEPTE DES REAKTIVEN PROGRAMMIERENS

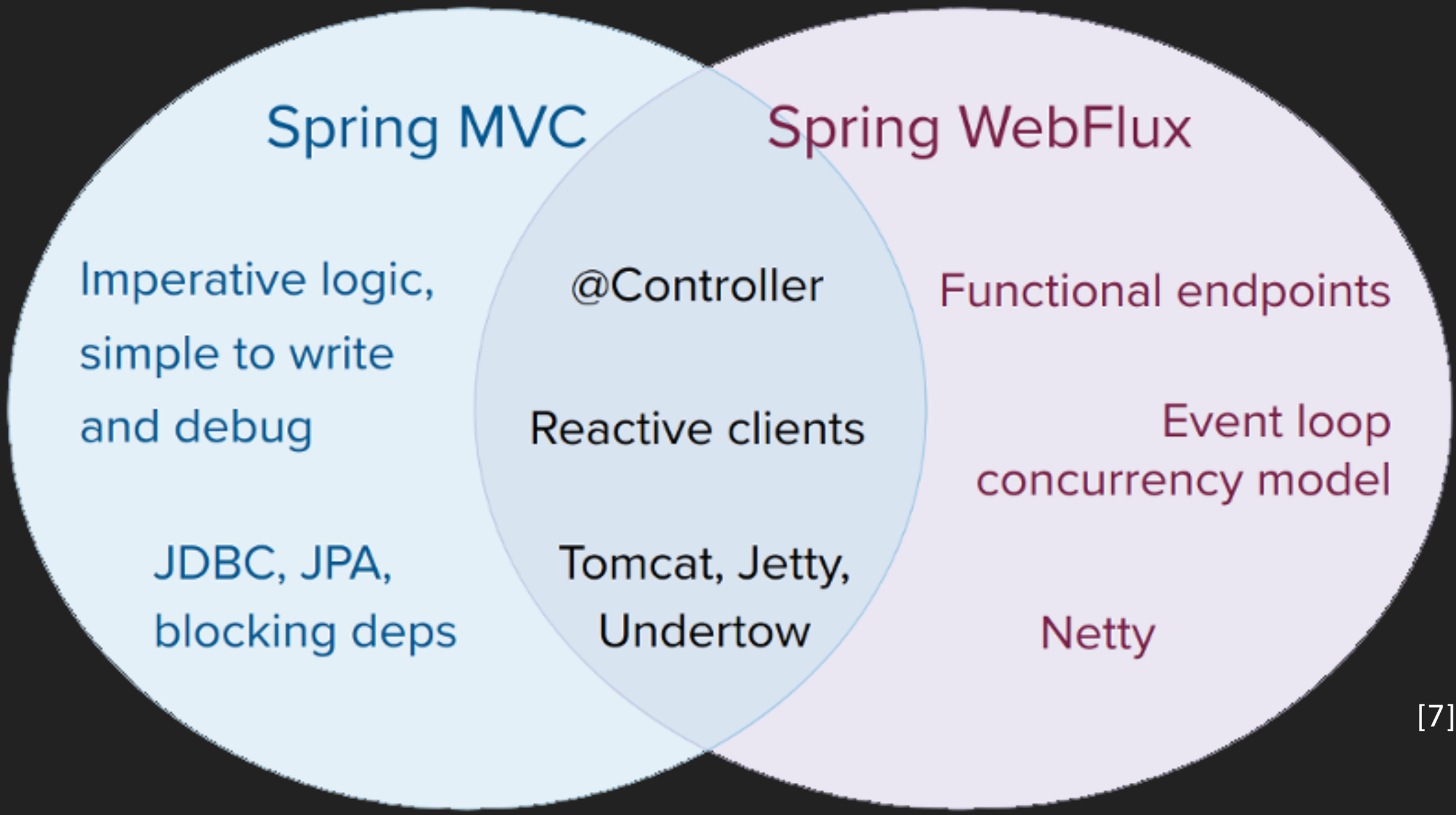
# PROJECT REACTOR – BEISPIEL

# SPRING WEBFLUX

# ÜBERBLICK

- Spring WebFlux erfordert reaktive Ende-zu-Ende Implementierungen
  - Eigentliche Bussinus-Logik
  - Reaktive Datenbanken
    - Webserver, die eine asynchrone Schnittstelle bereitstellen
- Unterschiede zu Spring MVC
  - Parallelitätsmodell
  - Standardannahmen für das Blockieren der Threads

# SPRING MVC VS. SPRING WEB FLUX




[7]

SPRING WEBFLUX

# LIVE CODING

# ZIEL

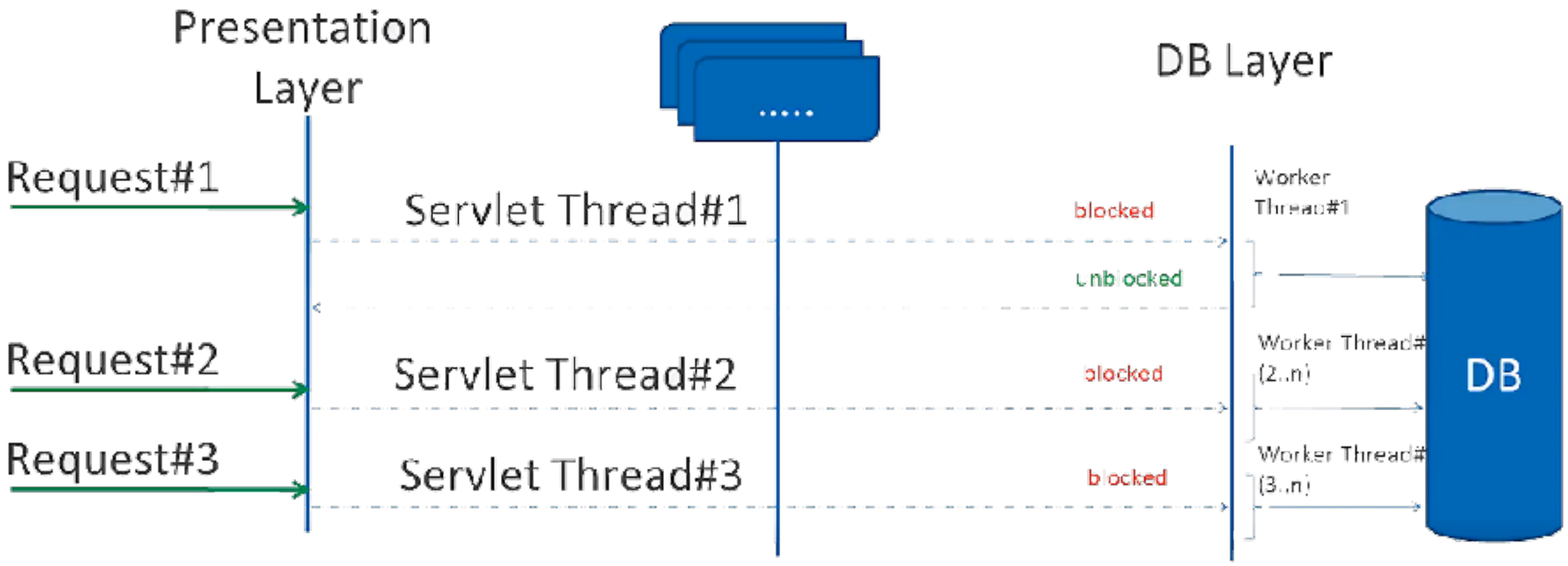
- Implementierung einer einfachen REST-Schnittstelle (*CRUD*-Operationen) durch TDD
  - Reaktives Testen –> WebClient
  - Klassisch annotierte Controller
  - Datenbankanbindung
- „Hello-World“ Anwendung als funktionaler Endpunkt

# FAZIT

# ANWENDBARKEIT NACH USE-CASES

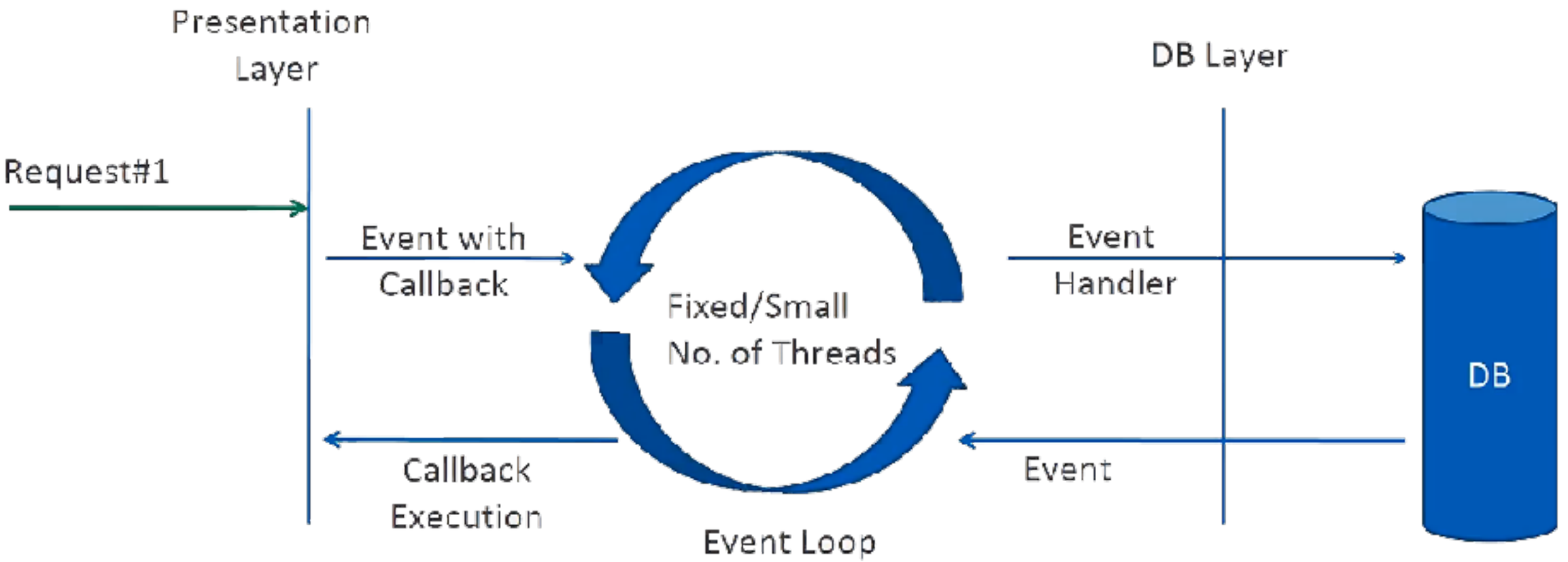
- Bestehende MVC-Anwendungen –> **MVC**
- Kompakte und funktionale Programmierung –> **WebFlux**
- Microservices –> **WebFlux**
- Niedrige Latenz bei WebClient –> **WebFlux**
- Blockierende Datenbank –> **MVC**

# PERFORMANCE – SERVLET-STACK




[9]

# PERFORMANCE – REACTIVE-STACK




[10]

# QUELLEN




- [1] Mark Weiser. „The Computer for the 21st Century“. In: Scientific American 265 (Sep. 1991), S. 66–75.
- [2] http://www.phpmind.com/blog/wp-content/uploads/2017/05/synchronous-asynchronous-javascript.png
- [3] https://i.imgflip.com/1tfz4g.jpg
- [4] https://scotch-res.cloudinary.com/image/upload/w_1000,q_auto:good,f_auto/media/4974/xCkAPcmuQNqQCGpO2avR_Event-loop.png.jpg
- [5] https://www.reactivemanifesto.org/images/reactive-traits-de.svg
- [6] https://raw.githubusercontent.com/reactor/reactor-core/v3.0.7.RELEASE/src/docs/marble/mono.png
- [7] https://raw.githubusercontent.com/reactor/reactor-core/v3.0.7.RELEASE/src/docs/marble/flux.png
- [8] https://docs.spring.io/spring/docs/current/spring-framework-reference/images/spring-mvc-and-webflux-venn.png
- [9] https://cdn1.howtodoinjava.com/wp-content/uploads/2019/02/Blocking-request-processing.png
- [10] https://cdn2.howtodoinjava.com/wp-content/uploads/2019/02/Non-blocking-request-processing.png