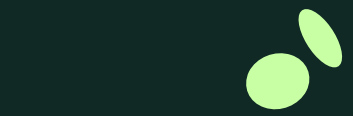


holisticon

holisticon

WeAreDevelopers World Congress 2024

Java's Project Loom & Co

The end for reactive programming?

Jan Weinschenker · 19 July 2024

Agenda

- Introduction
- About Loom and reactive programming
- Reactive Frameworks
- The Looming Doom (live coding)
- My opionionated conclusion
- Questions

Introduction

Transform today

Holisticon AG

... is a tech consultancy that operates holistically by managing strategy, organization and technology right from the beginning to empower and future-proof the digitalization of innovative business models.

- 17+ Years
- 80+ Consultants
- 250+ Clients
- 850+ Projects
- 200+ Conference-Talks

Jan Weinschenker

Consultant, Developer, Nerd at Holisticon

- I‘m working in software development for 18 years now.
- I like to build software that works.
- Big fan of reactive frameworks.

What is Loom?

Project Loom

JDK Enhancement Proposal 425

„Project Loom aims to drastically reduce the effort of writing, maintaining, and observing high-throughput concurrent applications that make the best use of available hardware.“

Ron Pressler (Tech lead, Project Loom)

https://cr.openjdk.org/~rpressler/loom/loom/sol1_part1.html

Project Loom

JDK Enhancement Proposal 425

*„Introduce **Virtual Threads** to the Java Platform. **Virtual Threads** are lightweight threads that dramatically reduce the effort of writing, maintaining, and observing high-throughput concurrent applications.“*

https://openjdk.org/jeps/425

Project Loom

JDK Enhancement Proposal 425

„I think Project Loom is going to kill Reactive Programming.“

Brian Goetz (Java Language Architect at Oracle)

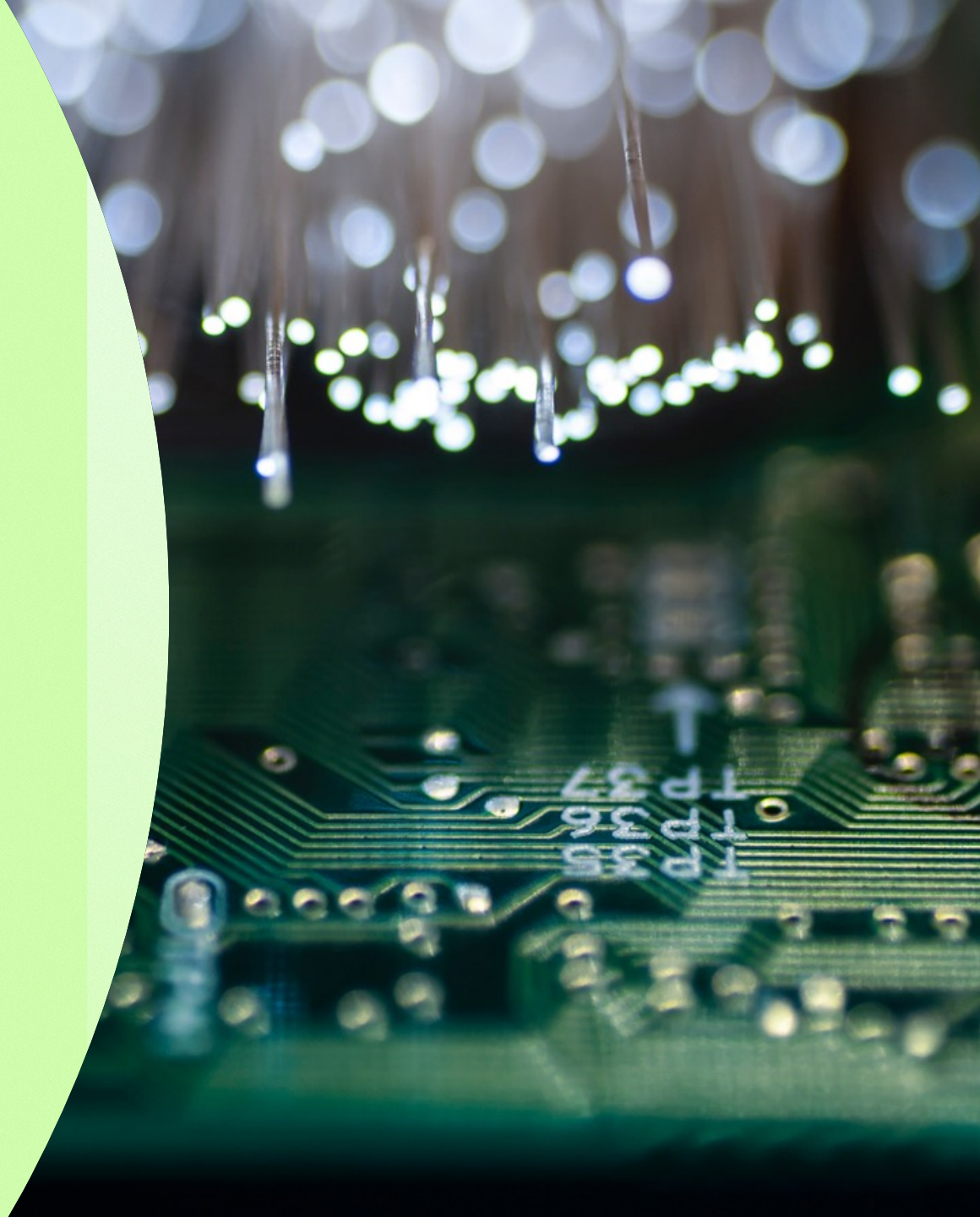
https://www.youtube.com/watch?v=9si7gK94gLo&t=1367s

Project Loom

JDK Enhancement Proposal 425

- *Easy means to implement concurrency with JDK core functionality.*
- *Released with Java 21 on 19 September 2023*

About reactive programming



Why should we write reactive software?

Software that we build interacts with the outside world.

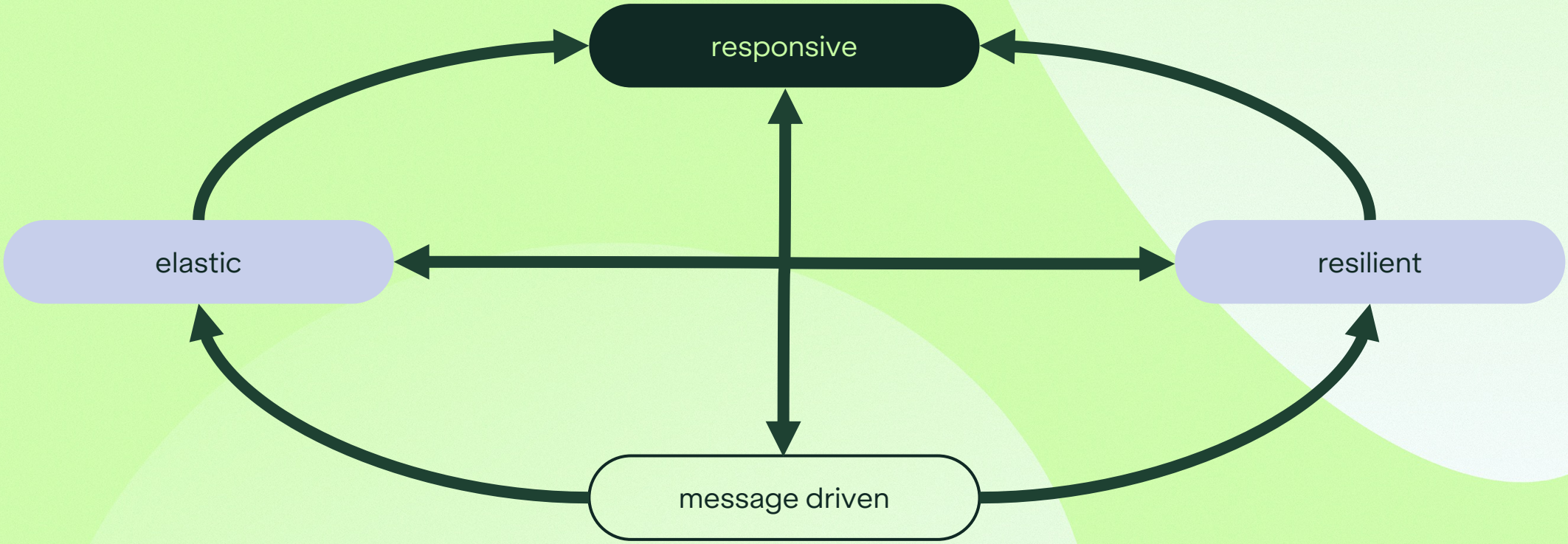
- Buttons pressed
- Incoming requests
- Database query results

Why should we write reactive software?

Users should always have the feeling that the application is working.

- The application should not block, it should be responsive.
- Responsiveness is one of the cornerstones of good usability.

The Reactive Manifesto

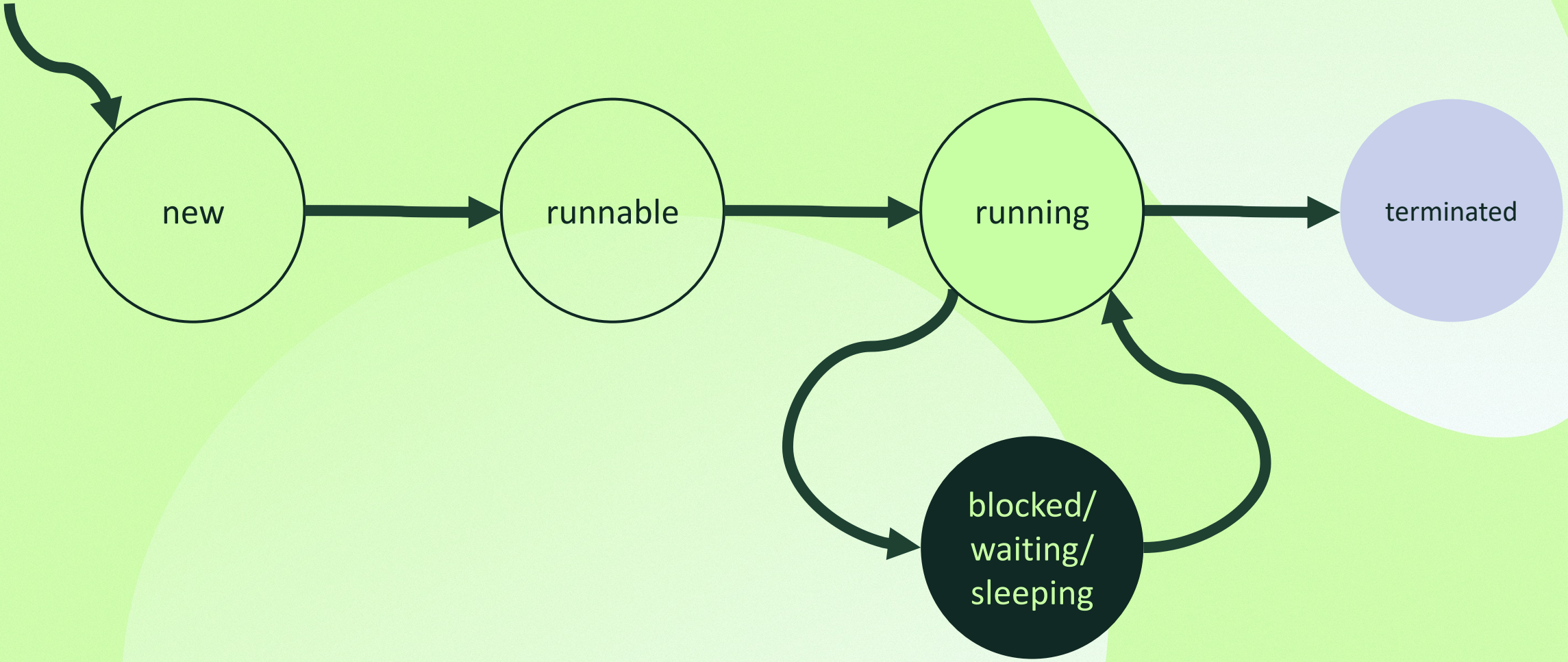


https://www.reactivemanifesto.org

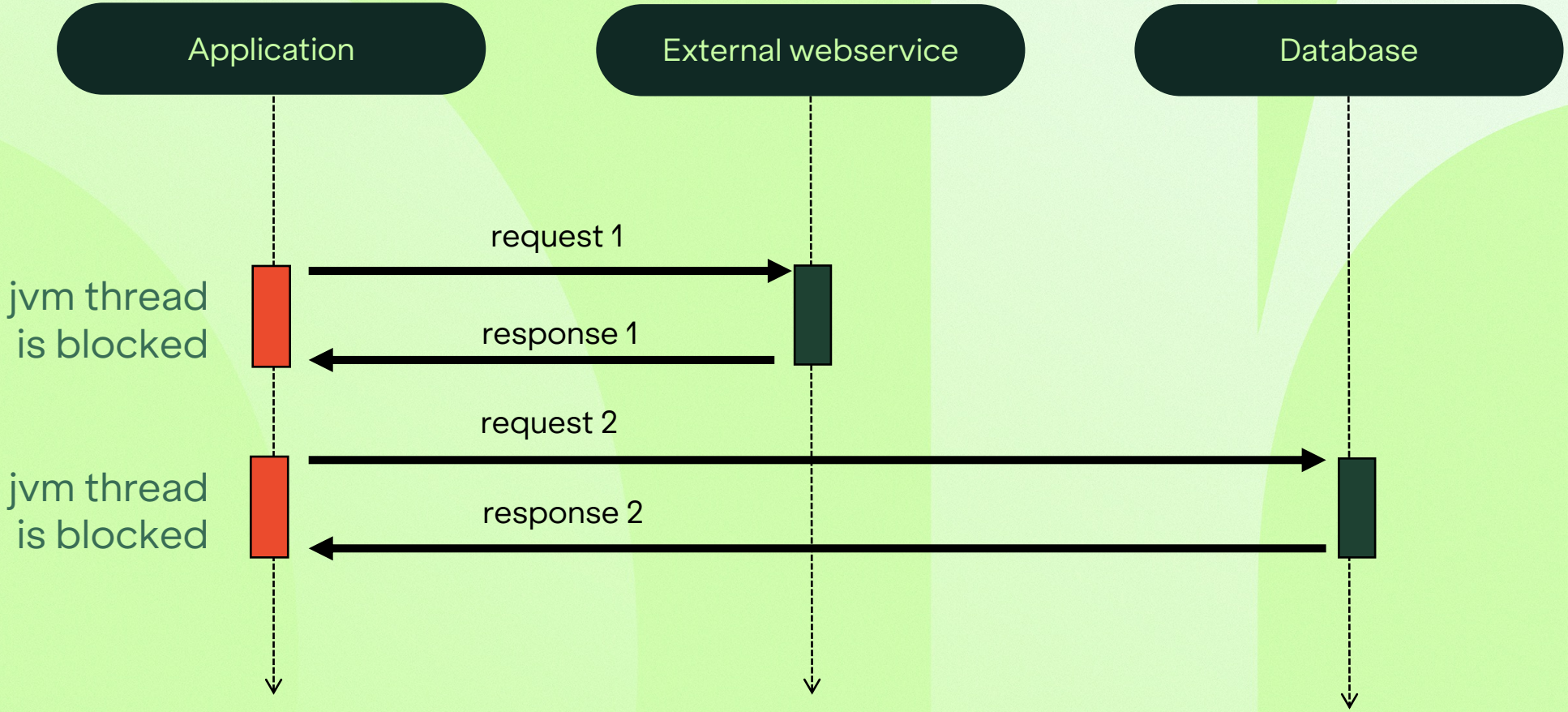
The issue with Blocked Threads

Reactive Frameworks – what do they do?

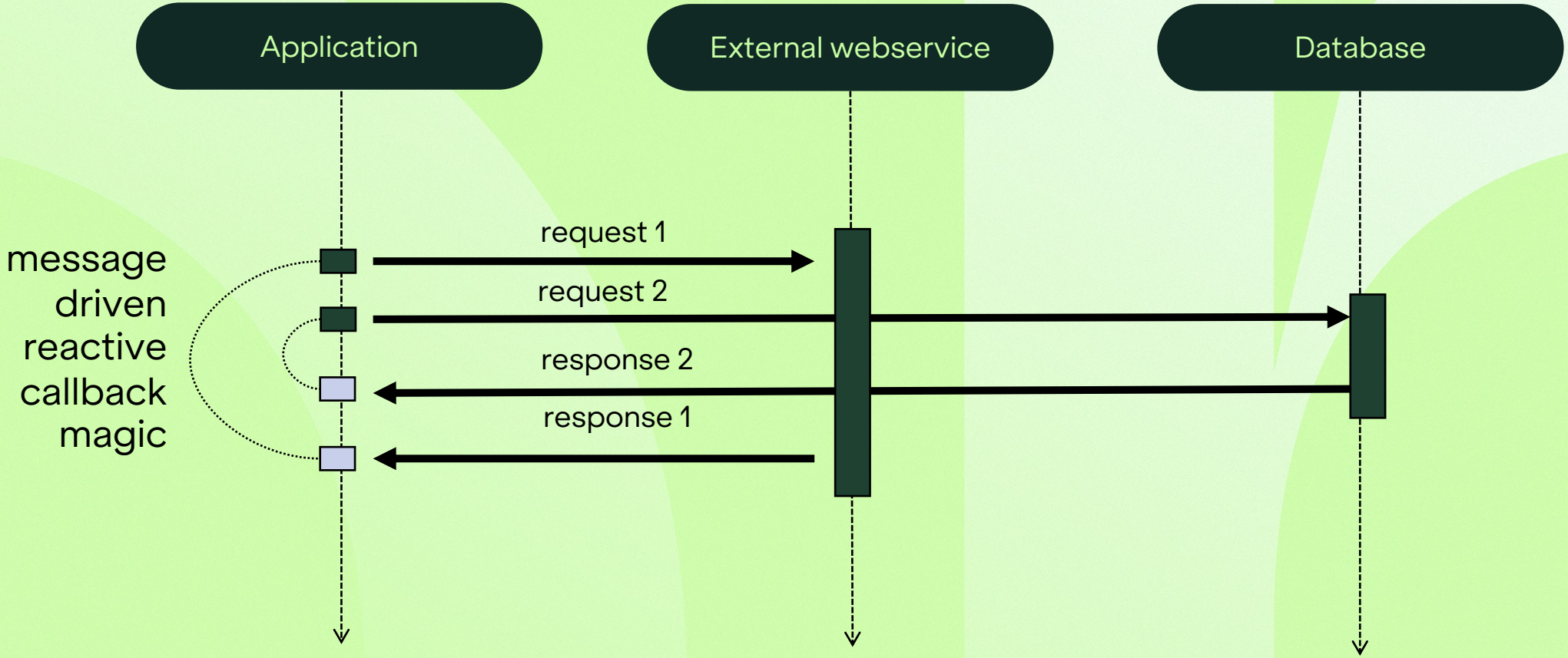
States of a jvm thread (simplified)



Reactive Frameworks – what do they do?



Reactive Frameworks – what do they do?



How to achieve non-blocking software?

- Responsive and non-blocking heaven would be:

  One task per thread.

- But threads are a limited resource!

- How to allocate a potential unlimited amount of work to a limited number of threads?

How to achieve non-blocking software?

Many concepts exist:

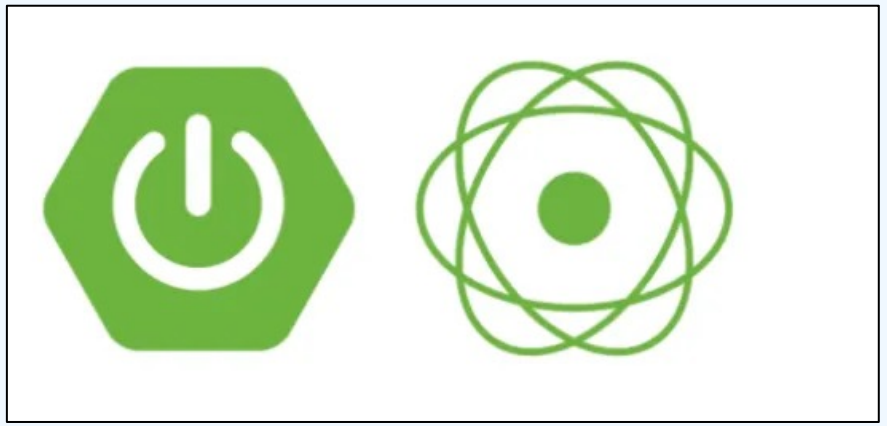
- Use of callbacks
- Reactive programming
- Coroutines

Use of callbacks

1. Perform a task
2. Add an asynchronous callback function, that processes the result afterwards.
3. Welcome to callback hell!
4. Completable Futures (Java 8, 2014)

Goodbye readable code!

Reactive Frameworks



- Abstraction from JVM threads
- Reactive Programming model
- Powerful DSL to describe flow of events

QUARKUS

VERT.X

Not a simple way to code ☹

Reactive Java Frameworks (RF)

- Reactive Streams (2013)
- **WebFlux / Project Reactor (Nov 2013)**
- Vert.x (May 2013)
- RxJava (May 2013)
- Helidon (Sept 2018)
- Mutiny (Dec 2019)

Reactive Frameworks (RF) – what do they do?

- Blocked threads wait for input from another resource.
- RF prevent blocked threads.
- RF aim to use processing time productively that would otherwise be wasted while waiting.

Reactive Frameworks – anything else?

Yes! A powerful DSL to

- model event flows: `merge, split, synchronize ...`
- handle heavy load: `delay, defer, sample, ...`
- handle errors: `retry, repeat, drop, ...`

Coroutines

Approach using `runBlocking` or `async/await` keywords.

- Blocking code into blocking functions
- Non-blocking code into suspending functions.

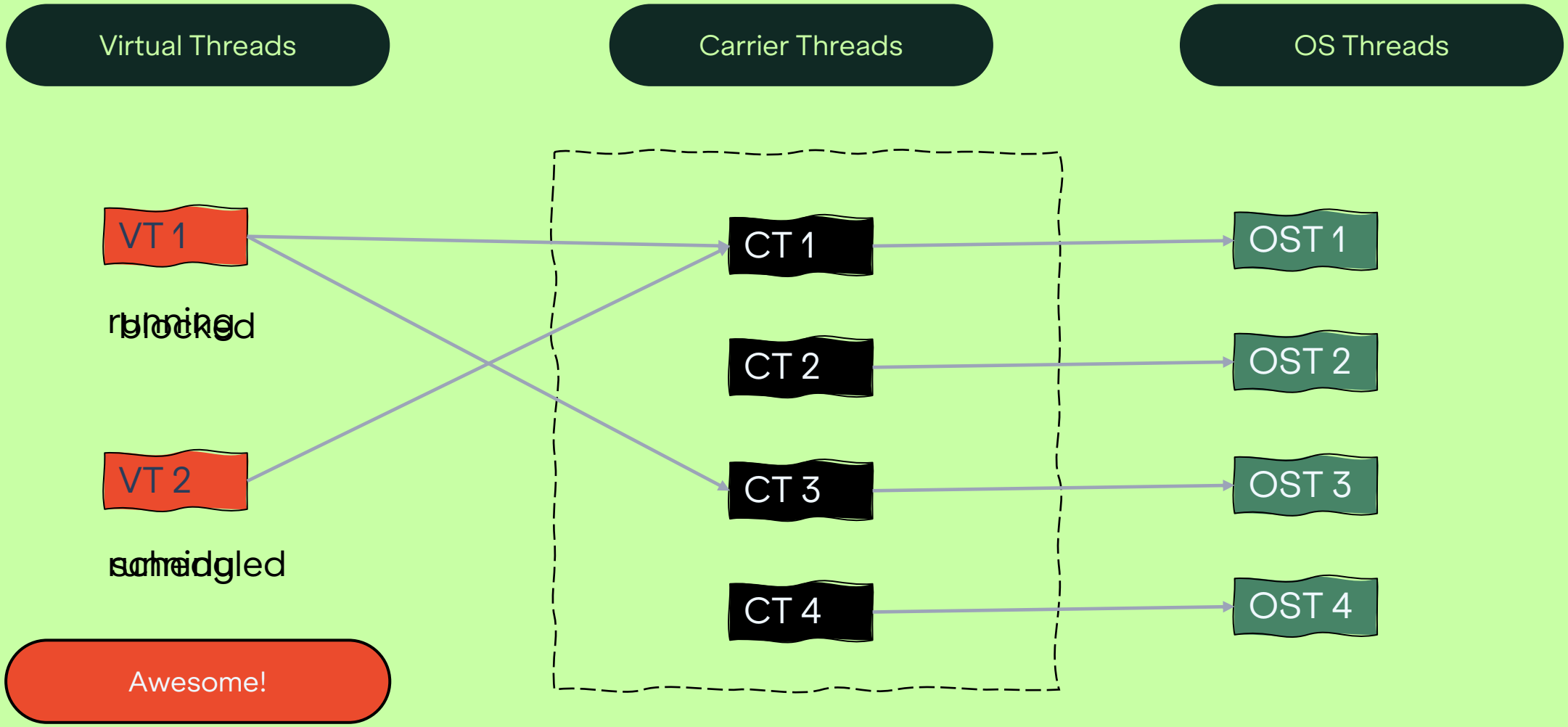
Code Examples

- Count from 0 to 1,000,000
- 10 second delay after each count
- Use concurrency to speed it up

Let‘s see some code ...

(and the looming doom)

Loom Virtual Threads



My short opinionated conclusion (1)

- Efficient thread utilization was an issue for Java applications.
- Reactive frameworks or other JVM languages provide mature solutions (since 2013).
- Project Loom now solves it on JVM level.

My short opinionated conclusion (2)

- Solve concurrency problems with simple sequential code and Virtual Threads
- Try the easy stuff first before doing something weird. (quoting Brian Goetz)
- But: Reactive frameworks still have their uses

Any questions?

Source Code

Available on GitHub:

https://github.com/holisticon/world-congress-demo-2024

Thank you!

holisticon