Evaluating Java Programming Paradigms

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

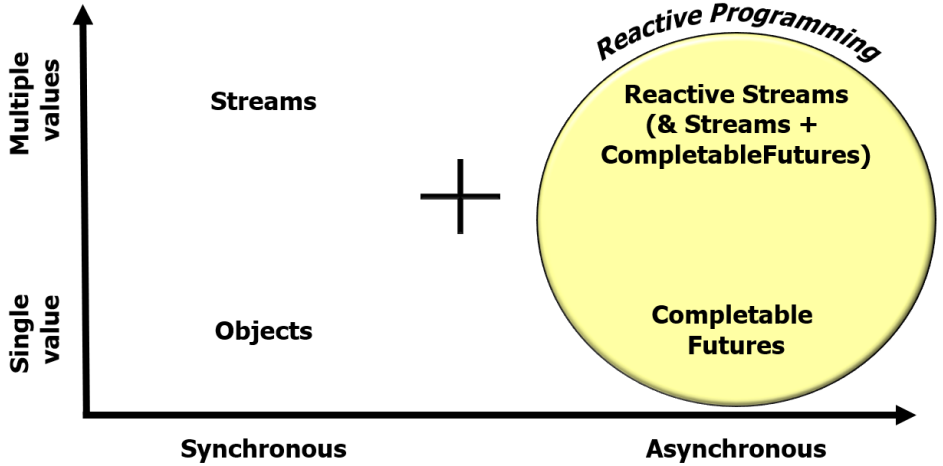
Professor of Computer Science

Institute for Software
Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand the key benefits & principles underlying the reactive programming paradigm
- Know the Java reactive streams API & popular implementations of this API
- Learn how Java reactive streams maps to key reactive programming principles
- Recognize how reactive programming compares with other Java paradigms
  - e.g., OO programming (including structured concurrency), & sync/ async functional programming



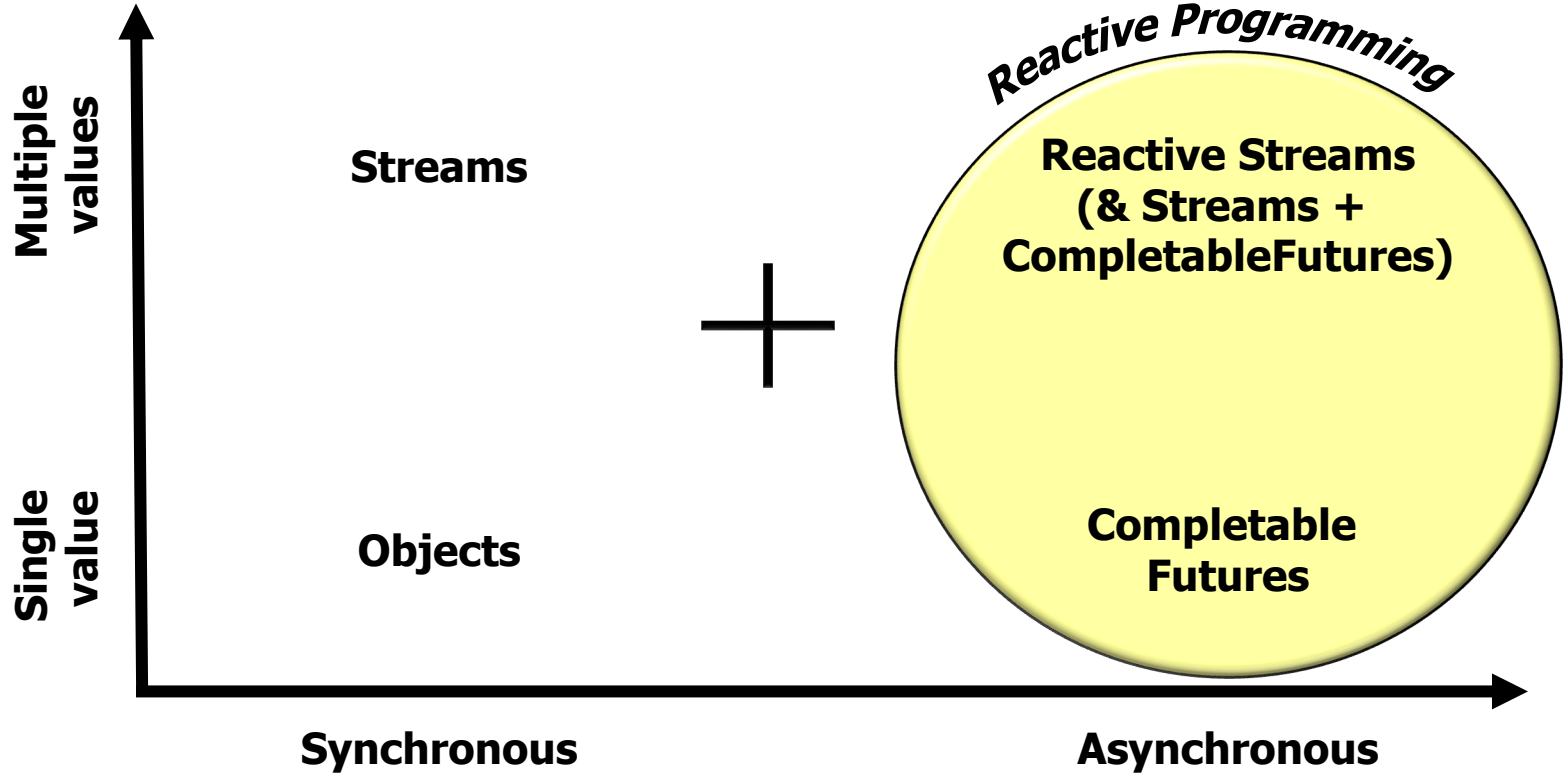
Learning Objectives in this Part of the Lesson

- Understand the key benefits & principles underlying the reactive programming paradigm
- Know the Java reactive streams API & popular implementations of this API
- Learn how Java reactive streams maps to key reactive programming principles
- Recognize how reactive programming compares with other Java paradigms
- Be aware of the pros & cons of reactive streams platforms vs. alternatives

Comparing Reactive Programming with Other Paradigms

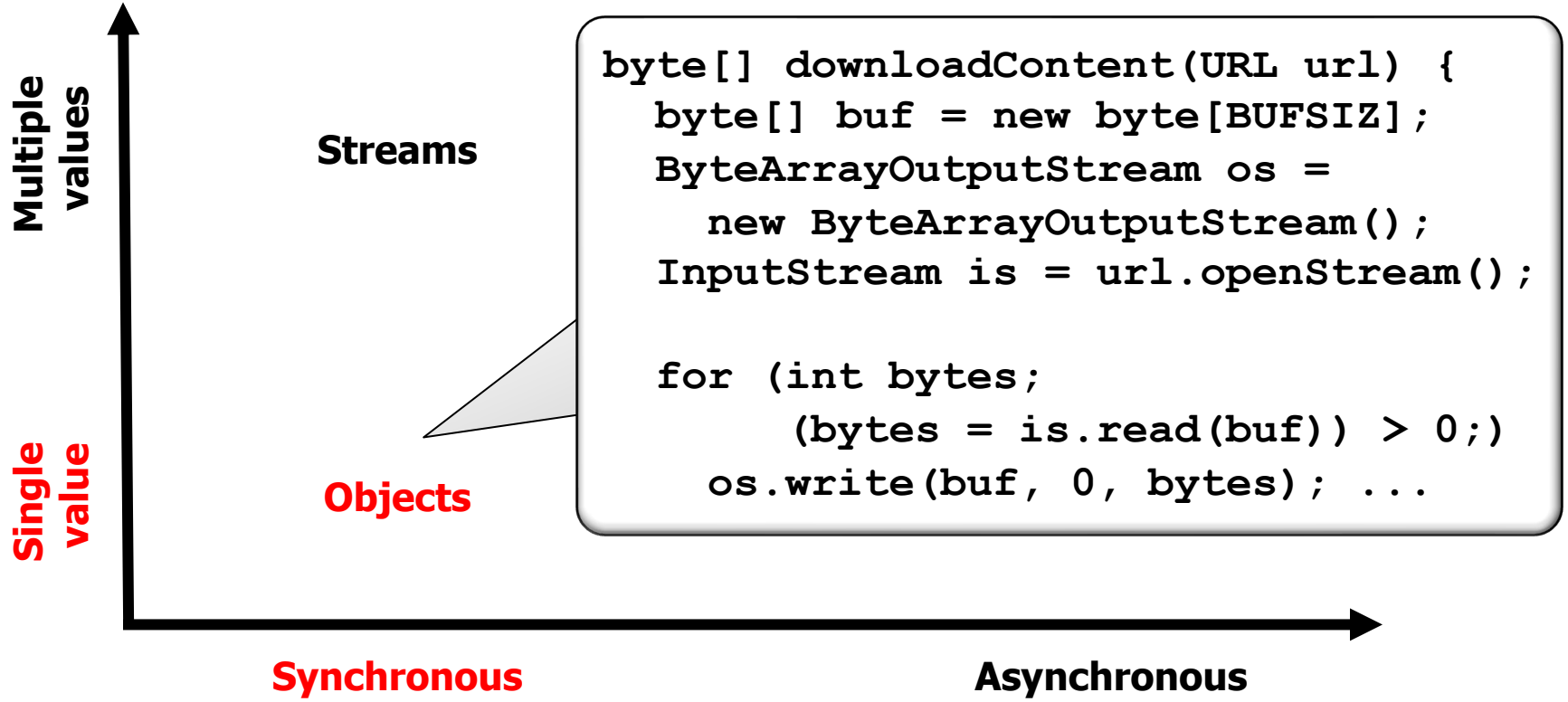
Comparing Reactive Programming with Other Paradigms

- Reactive programming is one of several Java programming paradigms



Comparing Reactive Programming with Other Paradigms

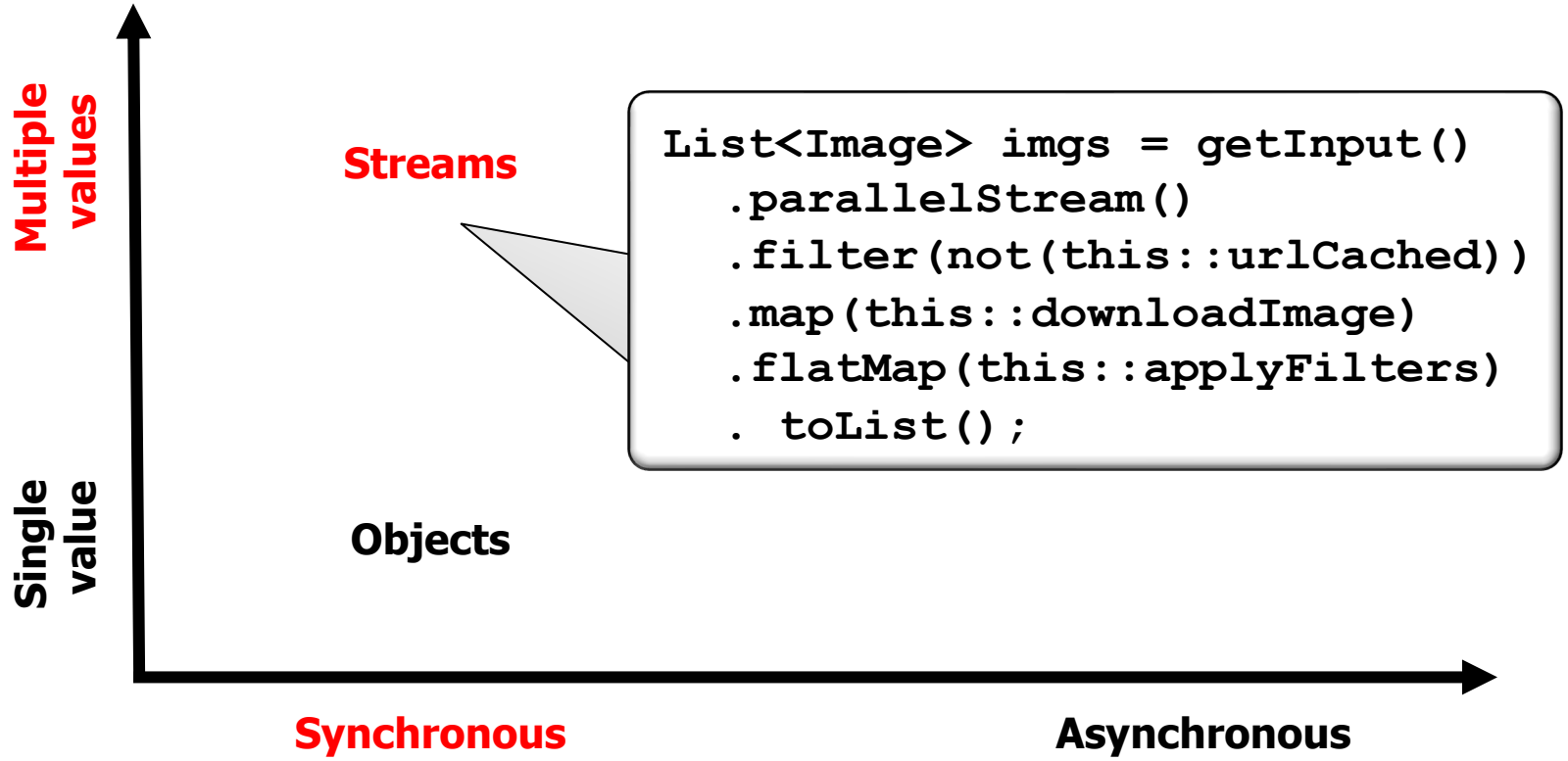
- Reactive programming is one of several Java programming paradigms



Java virtual threads & structured concurrency are making synchronous programming cool again!

Comparing Reactive Programming with Other Paradigms

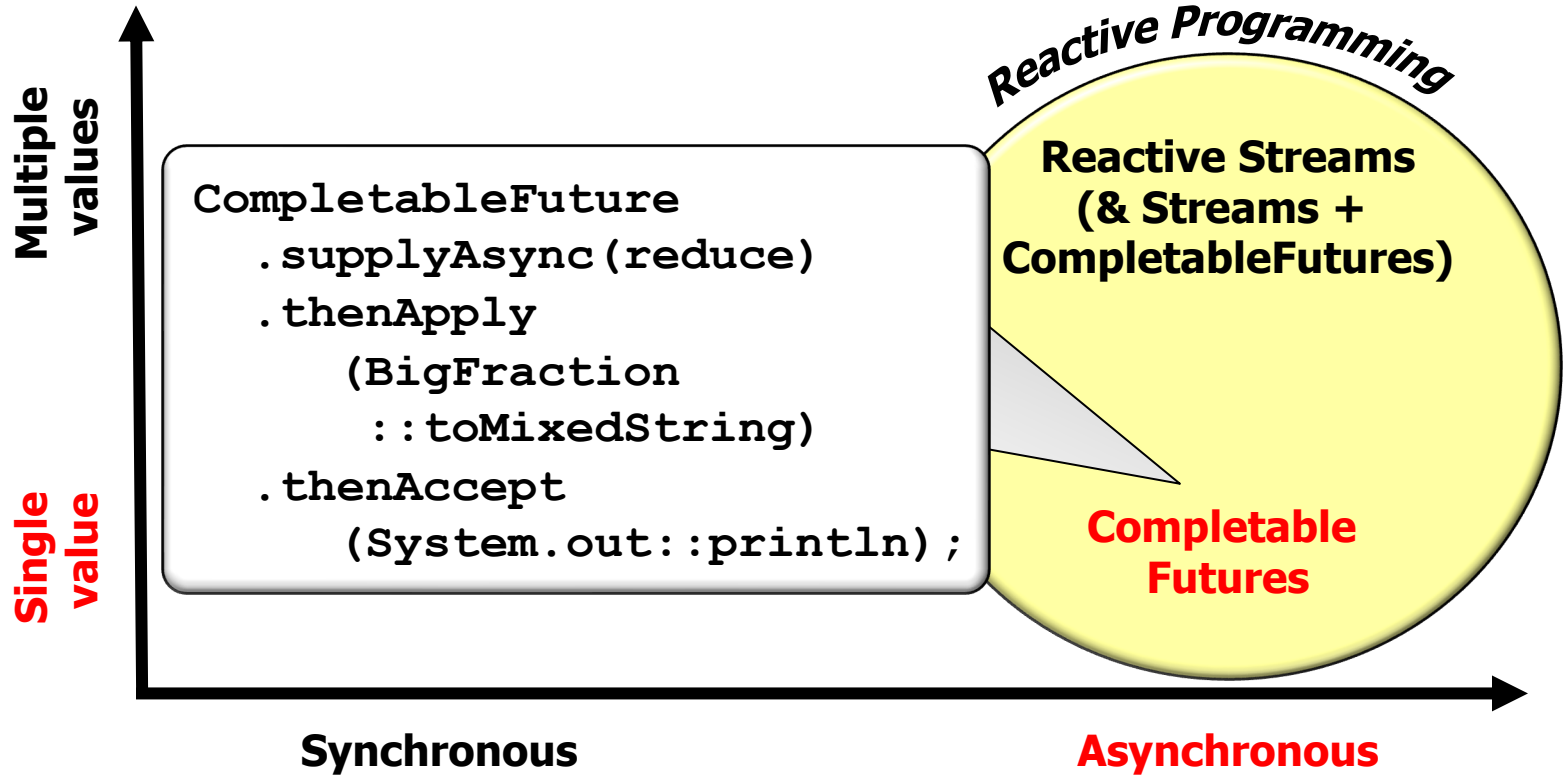
- Reactive programming is one of several Java programming paradigms



See docs.oracle.com/javase/tutorial/collections/streams/parallelism.html

Comparing Reactive Programming with Other Paradigms

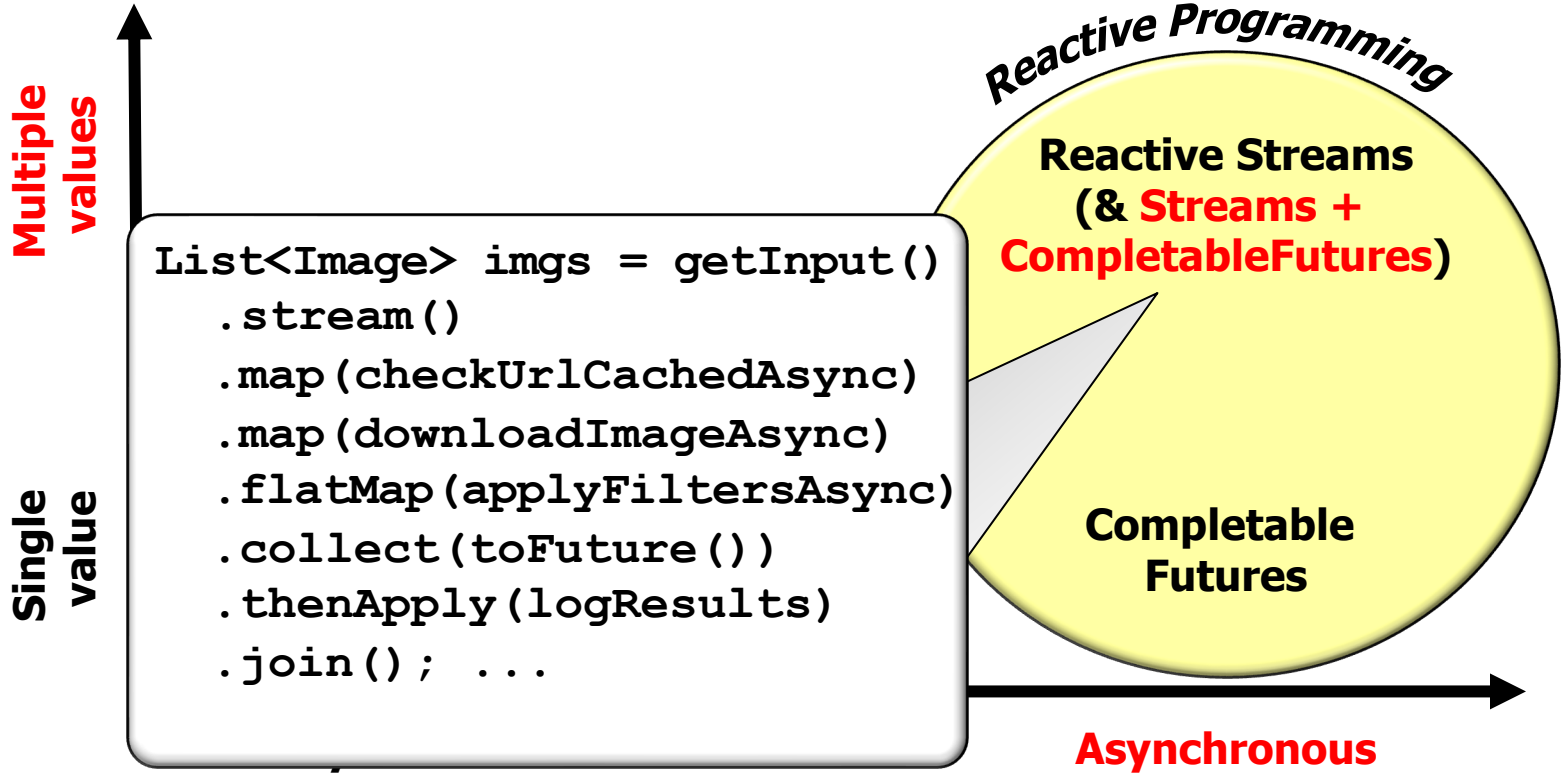
- Reactive programming is one of several Java programming paradigms



See www.baeldung.com/java-completablefuture

Comparing Reactive Programming with Other Paradigms

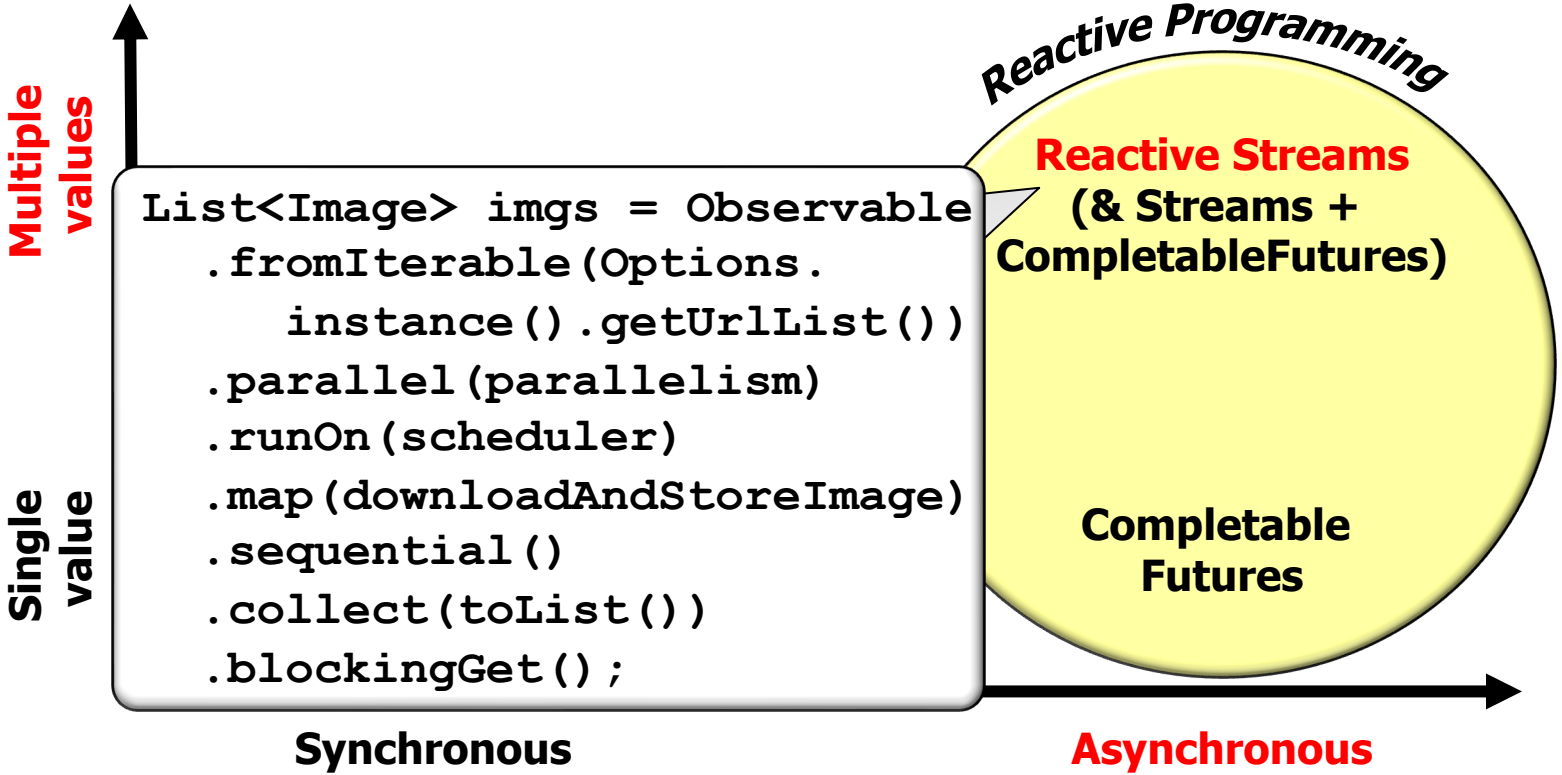
- Reactive programming is one of several Java programming paradigms



See www.linkedin.com/pulse/streams-completablefutures-java-8-gustavo-sousa

Comparing Reactive Programming with Other Paradigms

- Reactive programming is one of several Java programming paradigms

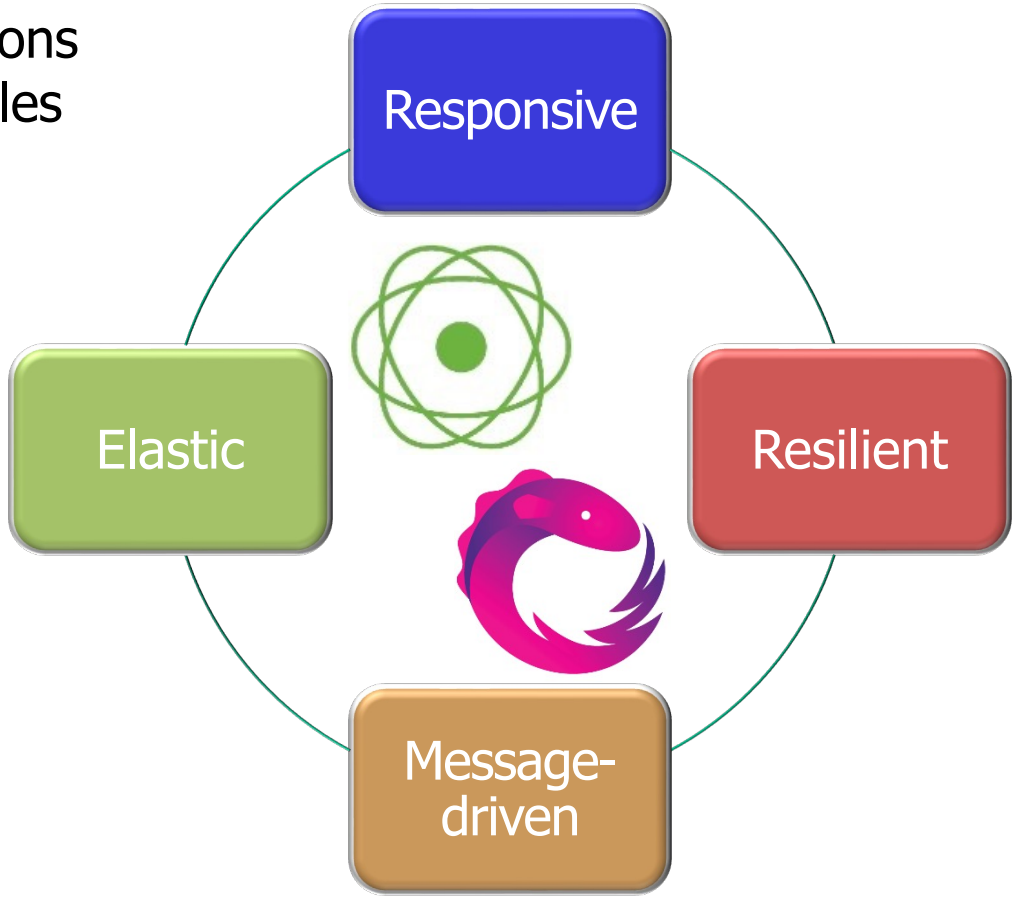


See www.baeldung.com/rx-java

Pros & Cons of Java Reactive Streams Platforms

Pros & Cons of Java Reactive Streams Platforms

- Java reactive streams implementations apply reactive programming principles to achieve several benefits



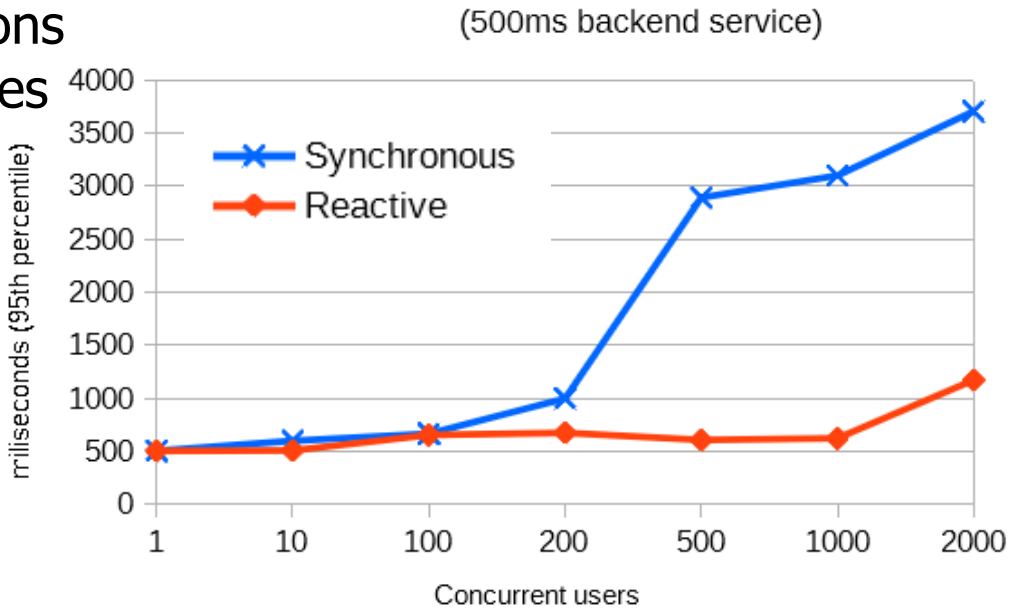
Pros & Cons of Java Reactive Streams Platforms

- Java reactive streams implementations apply reactive programming principles to achieve several benefits
  - *Minimal resource utilization*
    - Support concurrency with a minimal number of threads via a range of thread pools

Name	Description
Schedulers.computation()	Schedules computation bound work (ScheduledExecutorService with pool size = NCPU, LRU worker select strategy)
Schedulers.immediate()	Schedules work on current thread
Schedulers.io()	I/O bound work (ScheduledExecutorService with growing thread pool)
Schedulers.trampoline()	Queues work on the current thread
Schedulers.newThread()	Creates new thread for every unit of work
Schedulers.test()	Schedules work on scheduler supporting virtual time
Schedulers.from(Executor e)	Schedules work to be executed on provided executor

Pros & Cons of Java Reactive Streams Platforms

- Java reactive streams implementations apply reactive programming principles to achieve several benefits
  - *Minimal resource utilization*
    - Support concurrency with a minimal number of threads via a range of thread pools
      - Scale up performance with relatively few resources



See dzone.com/articles/spring-boot-20-webflux-reactive-performance-test

Pros & Cons of Java Reactive Streams Platforms

- Java reactive streams implementations apply reactive programming principles to achieve several benefits
  - *Minimal resource utilization*
  - *Hides concurrent programming*
    - Explicit synchronization and/or threading is rarely needed when applying these frameworks

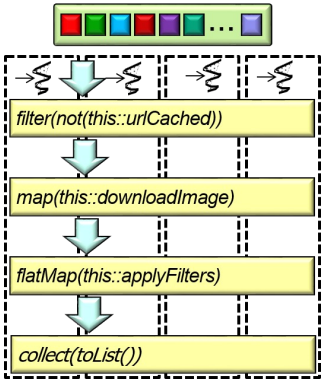
Alleviates many accidental & inherent complexities of concurrency/parallelism

Pros & Cons of Java Reactive Streams Platforms

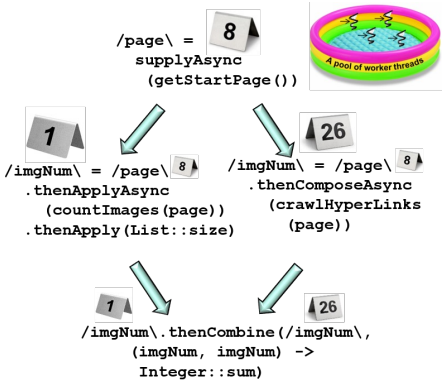
- Java reactive streams implementations apply reactive programming principles to achieve several benefits
  - *Minimal resource utilization*
  - *Hides concurrent programming*

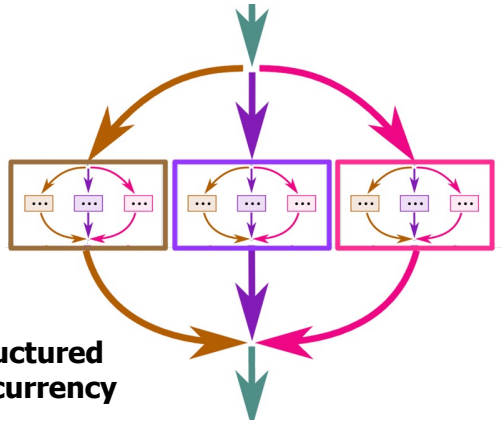
These benefits are not unique to reactive streams, however!!

Parallel Streams

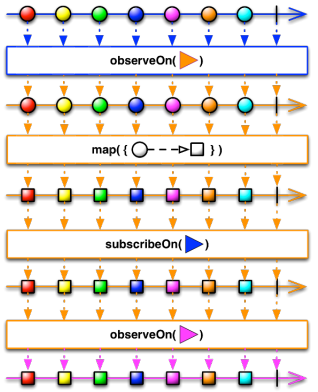


Completable Futures





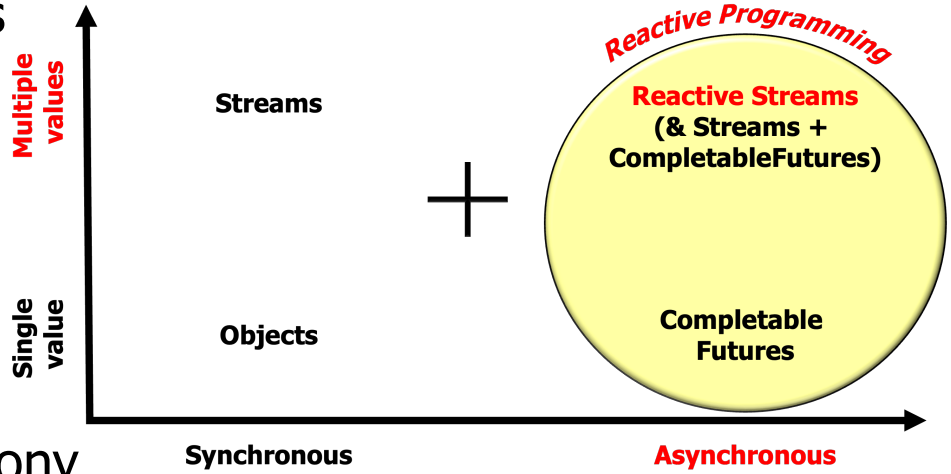
Structured Concurrency

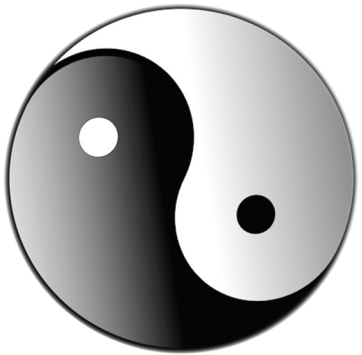


Reactive Streams

Pros & Cons of Java Reactive Streams Platforms

- Java reactive streams implementations apply reactive programming principles to achieve several benefits
  - *Minimal resource utilization*
  - *Hides concurrent programming*
  - *Seamlessly integrates paradigms*
    - Integrates concurrency & asynchrony more seamlessly than other Java programming paradigms





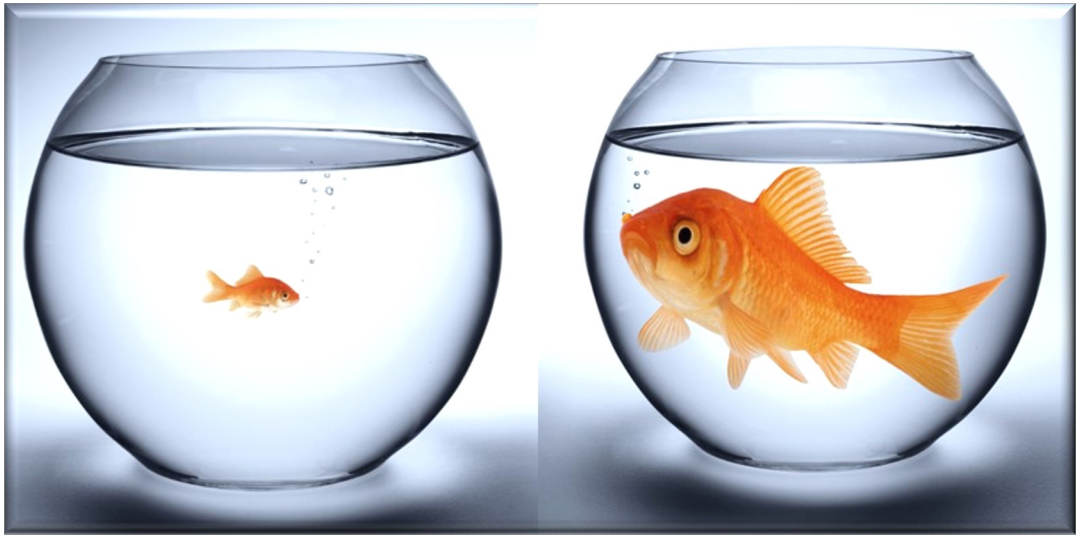
Pros & Cons of Java Reactive Streams Platforms

- Java reactive streams implementations apply reactive programming principles to achieve several benefits
  - *Minimal resource utilization*
  - *Hides concurrent programming*
  - *Seamlessly integrates paradigms*
    - Integrates concurrency & asynchrony more seamlessly than other Java programming paradigms
      - e.g., concurrent/asynchronous programming looks much like synchronous programming

```
List<Image> imgs = Observable
   .fromIterable(Options.
      instance().getUrlList())
   .parallel(parallelism)
   .runOn(scheduler)
   .map(downloadAndStoreImage)
   .sequential()
   .collect(toList())
   .blockingGet();
```

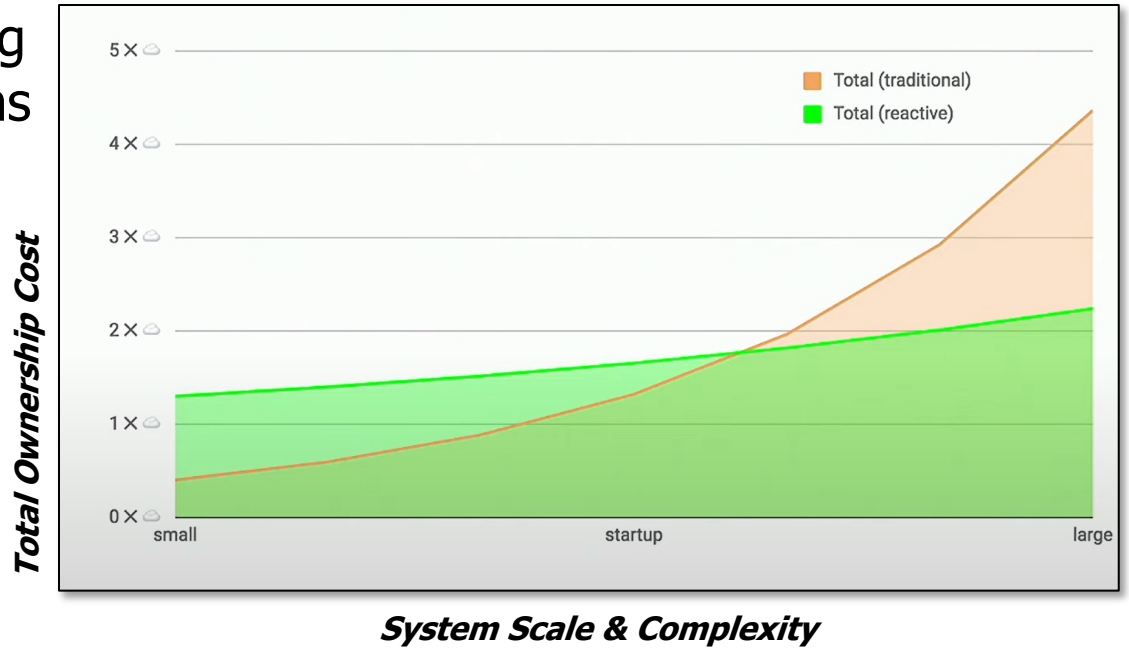
Pros & Cons of Java Reactive Streams Platforms

- However, reactive programming isn't appropriate in all situations



Pros & Cons of Java Reactive Streams Platforms

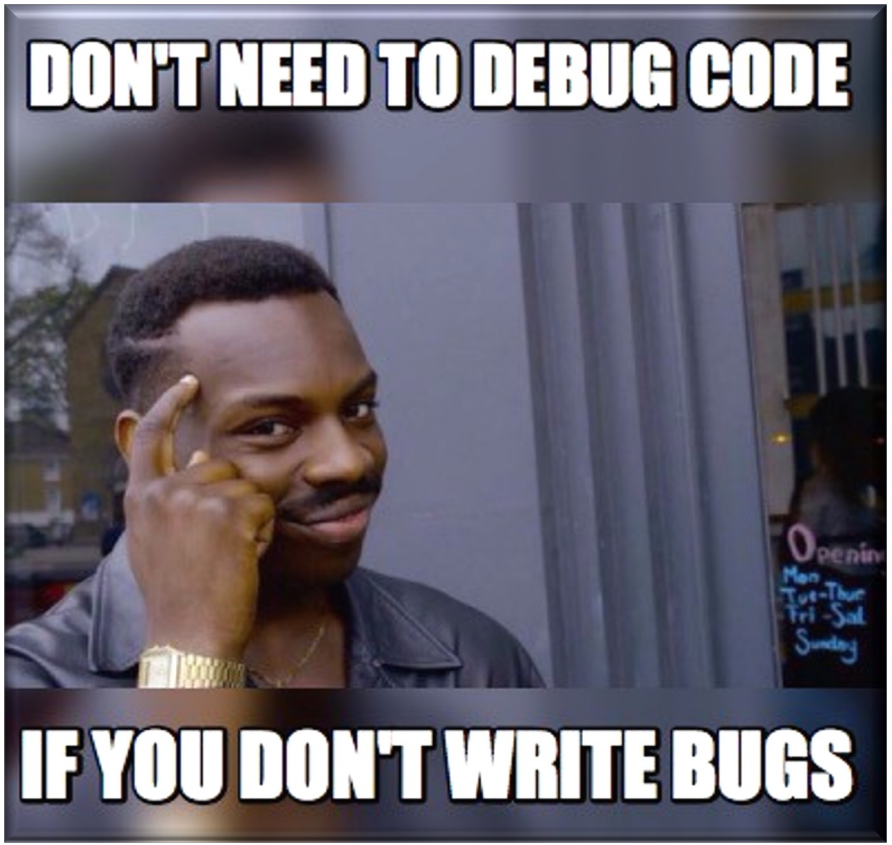
- However, reactive programming isn't appropriate in all situations
  - *Complexity*
    - Most Java developers are familiar with imperative OO programming
      - There is a learning curve associated with introducing a reactive style



See www.youtube.com/watch?v=z0a0N9OgaAA

Pros & Cons of Java Reactive Streams Platforms

- However, reactive programming isn't appropriate in all situations
  - *Complexity*
  - *Debugging*
    - Can be harder due to asynchronous operations & lack of meaningful stack traces



See www.baeldung.com/spring-debugging-reactive-streams

Pros & Cons of Java Reactive Streams Platforms

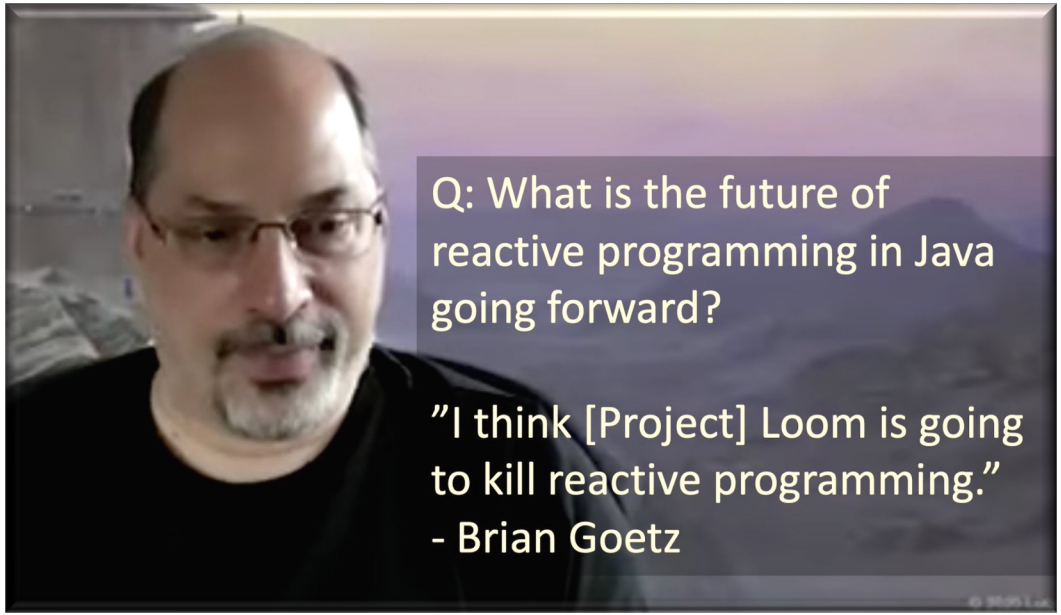
- However, reactive programming isn't appropriate in all situations
  - *Complexity*
  - *Debugging*

It's essential to master the reactive programming learning curve to use it effectively!

See reactive-programming-project-reactor-webflux-oh-my-4bfa470feee7

Pros & Cons of Java Reactive Streams Platforms

- There are various perspectives on reactive microservices vs. micro-services based on Java structured concurrency!



See www.youtube.com/watch?v=9si7gK94gLo&t=1153s

End of Evaluating Java Programming Paradigms