Evaluating Java Reactive Streams Programming

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

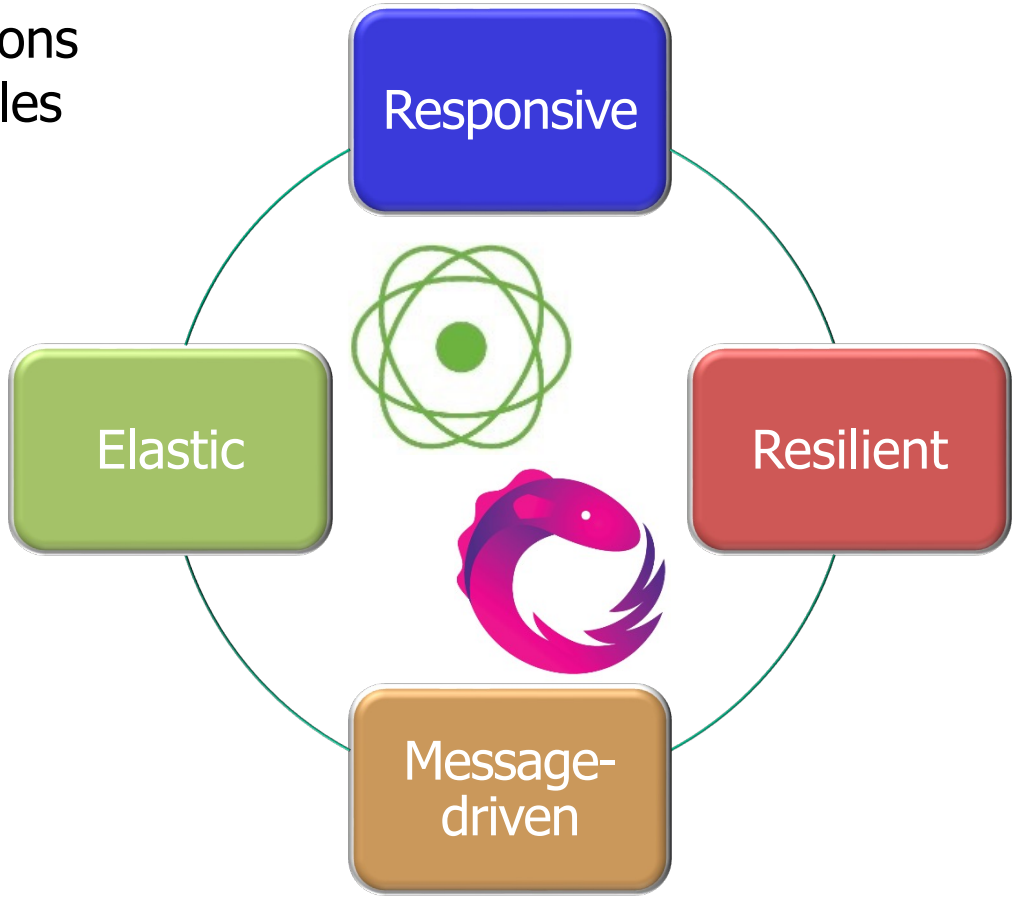
Learning Objectives in this Part of the Lesson

- 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
- Be aware of the pros & cons of Java reactive streams platforms vs. alternatives

Pros of Java Reactive Streams Platforms

Pros of Java Reactive Streams Platforms

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



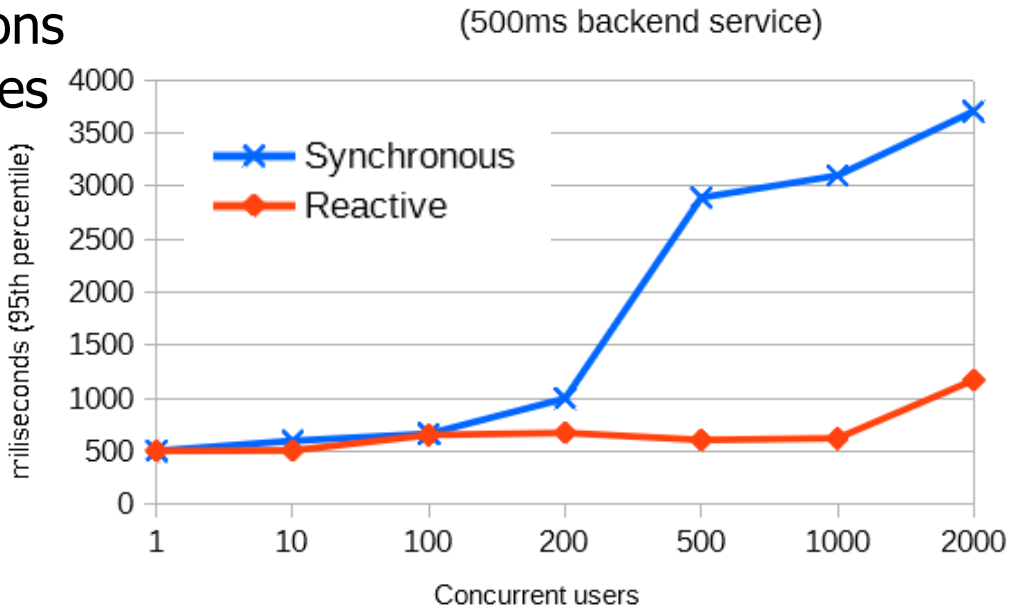
Pros of Java Reactive Streams Platforms

- Java reactive streams implementations apply reactive programming principles to achieve several benefits
  - *Improved 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 of Java Reactive Streams Platforms

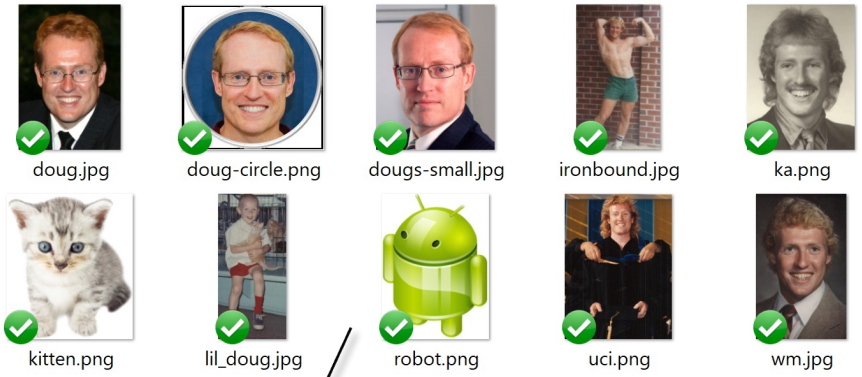
- Java reactive streams implementations apply reactive programming principles to achieve several benefits
  - *Improved resource utilization*
    - Support concurrency with a minimal number of threads via a range of thread pools
      - Scale up performance with relatively few resources



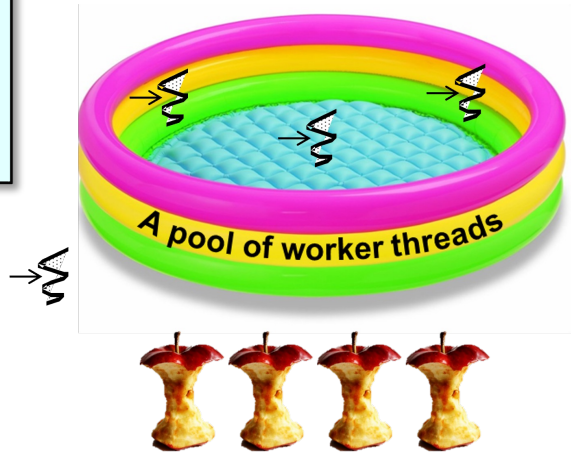
See dzone.com/articles/spring-boot-20-webflux-reactive-performance-test

Pros of Java Reactive Streams Platforms

- Java reactive streams implementations apply reactive programming principles to achieve several benefits
  - *Improved resource utilization*
    - Support concurrency with a minimal number of threads via a range of thread pools
      - Scale up performance with relatively few resources
    - Particularly useful for I/O-bound operations in Java programs using traditional threading models



e.g., downloading more images than the number of cores



Pros of Java Reactive Streams Platforms

- Java reactive streams implementations apply reactive programming principles to achieve several benefits
  - *Improved resource utilization*
    - Support concurrency with a minimal number of threads via a range of thread pools
  - Provide a rich set of operators to manage asynchronous data flows & backpressure
    - Prevents subscribers from being over-whelmed in high-load environments

See www.wideopeneats.com/i-love-lucy-chocolate-factory

Pros of Java Reactive Streams Platforms

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

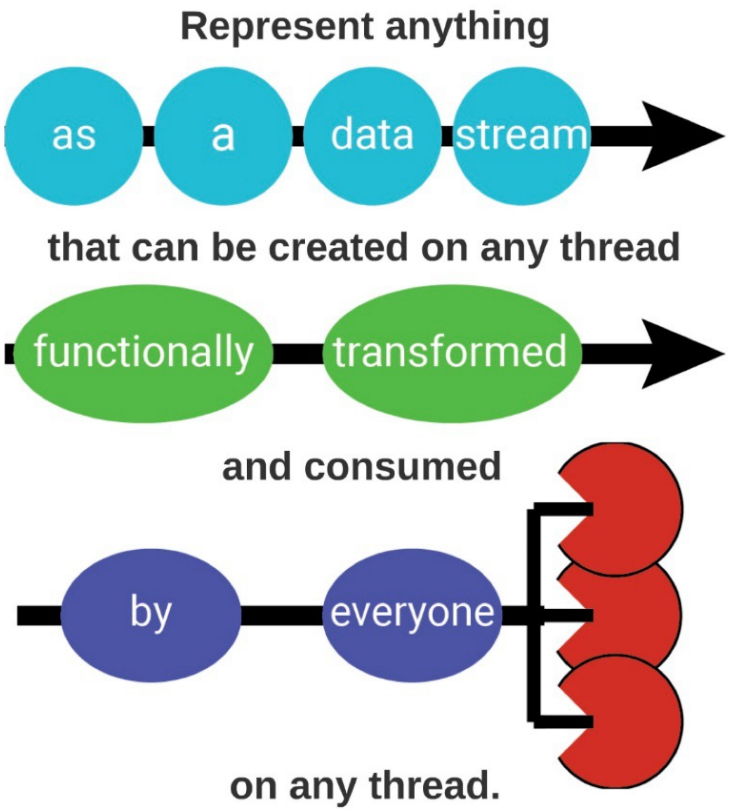
Pros of Java Reactive Streams Platforms

- Java reactive streams implementations apply reactive programming principles to achieve several benefits
  - *Improved resource utilization*
  - *Hides concurrent programming*
  - *Enhanced error handling*
    - Contributes to building resilient systems that can gracefully handle failures & maintain functionality under adverse conditions

See kalpads.medium.com/error-handling-with-reactive-streams-77b6ec7231ff

Pros of Java Reactive Streams Platforms

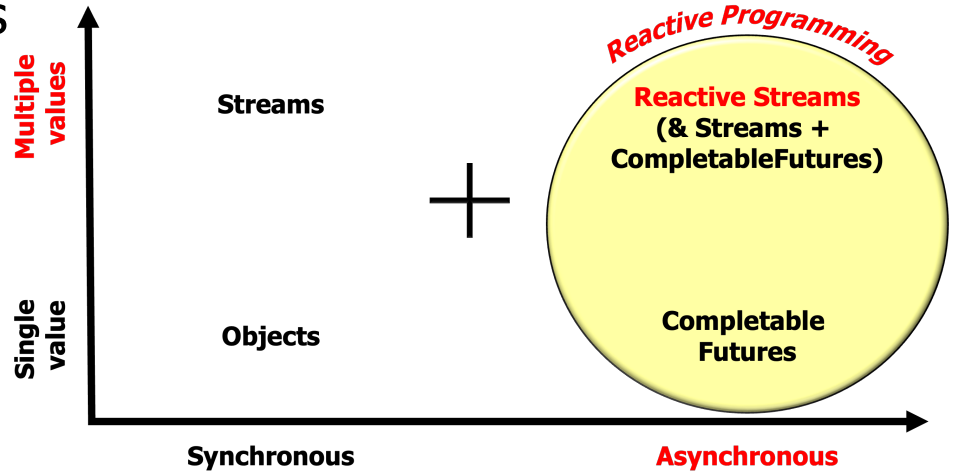
- Java reactive streams implementations apply reactive programming principles to achieve several benefits
  - *Improved resource utilization*
  - *Hides concurrent programming*
  - *Enhanced error handling*
  - *Simplified composition & reuse*
    - Enable complex data processing pipelines that are easier to understand, maintain, & debug

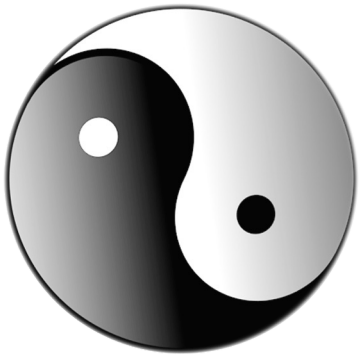


See www.tatvasoft.com/blog/java-reactive-programming

Pros of Java Reactive Streams Platforms

- Java reactive streams implementations apply reactive programming principles to achieve several benefits
  - *Improved resource utilization*
  - *Hides concurrent programming*
  - *Enhanced error handling*
  - *Simplified composition & reuse*
  - *Seamlessly integrates paradigms*
    - Integrates concurrency & asynchrony more seamlessly than other Java paradigms





Pros of Java Reactive Streams Platforms

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

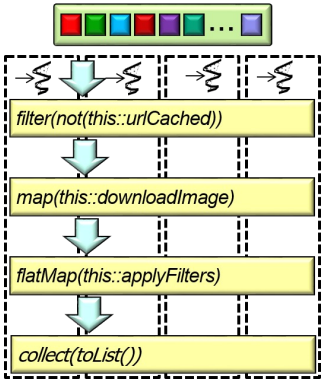
```
List<Image> imgs = Observable
   .fromIterable(Options.
      instance().getUrlList())
   .parallel(parallelism)
   .runOn(scheduler)
   .map(downloadAndStoreImage)
   .sequential()
   .collect(toList())
   .blockingGet();
```

Pros of Java Reactive Streams Platforms

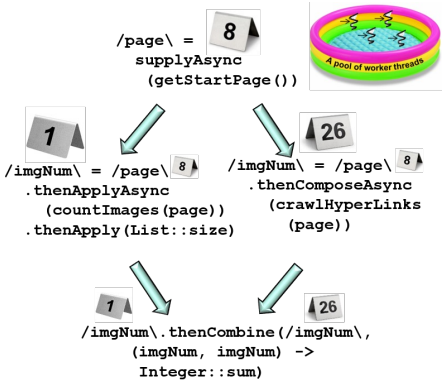
- Java reactive streams implementations apply reactive programming principles to achieve several benefits
  - *Improved resource utilization*
  - *Hides concurrent programming*
  - *Enhanced error handling*
  - *Simplified composition & reuse*
  - *Seamlessly integrates paradigms*

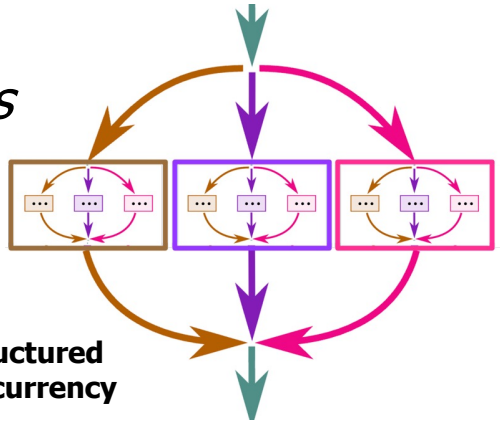
These benefits are not unique to reactive streams, however!!

Parallel Streams

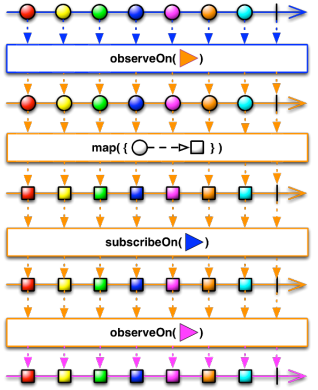


Completable Futures





Structured Concurrency

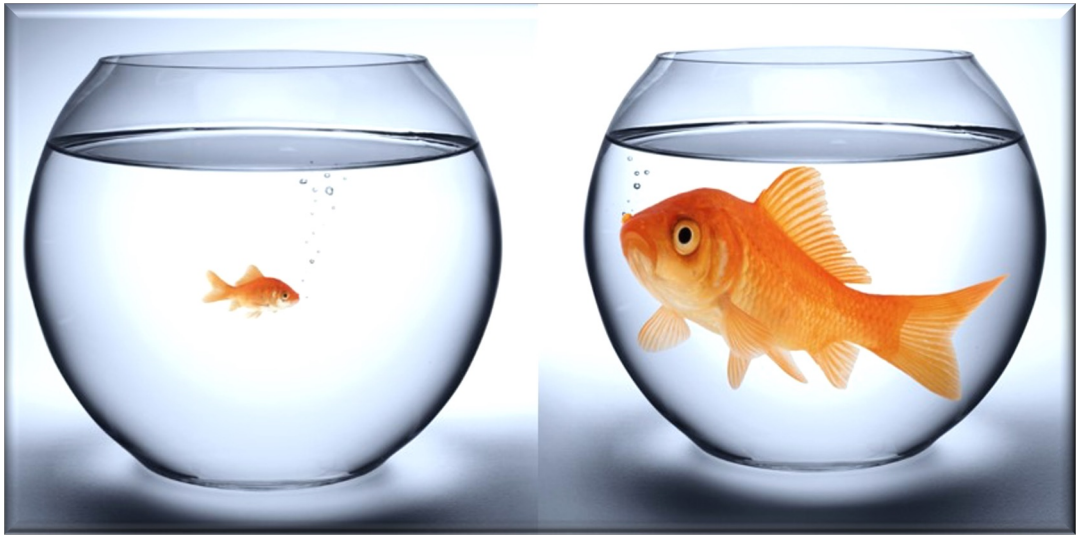


Reactive Streams

Cons of Java Reactive Streams Platforms

Cons of Java Reactive Streams Platforms

- Reactive programming is not appropriate in all situations



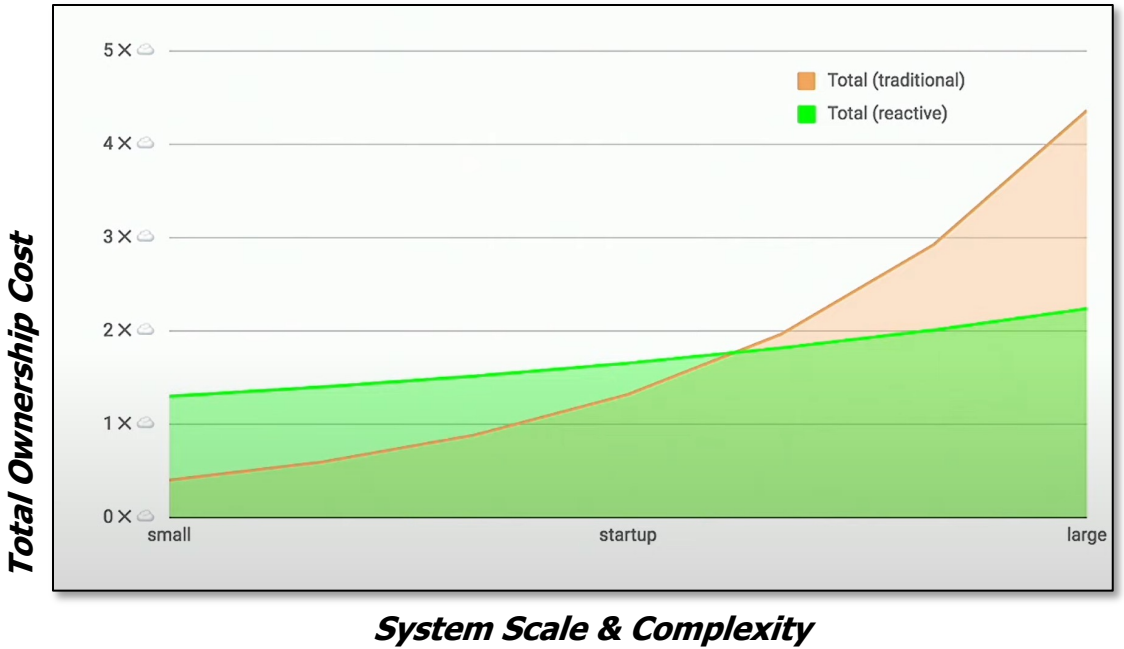
See dev.to/stealthmusic/dont-drink-too-much-reactive-cool-aid-20lk

Cons of Java Reactive Streams Platforms

- Reactive programming is not appropriate in all situations
  - *Complexity*
    - Java developers are more familiar with the OO style than the reactive style

Cons of Java Reactive Streams Platforms

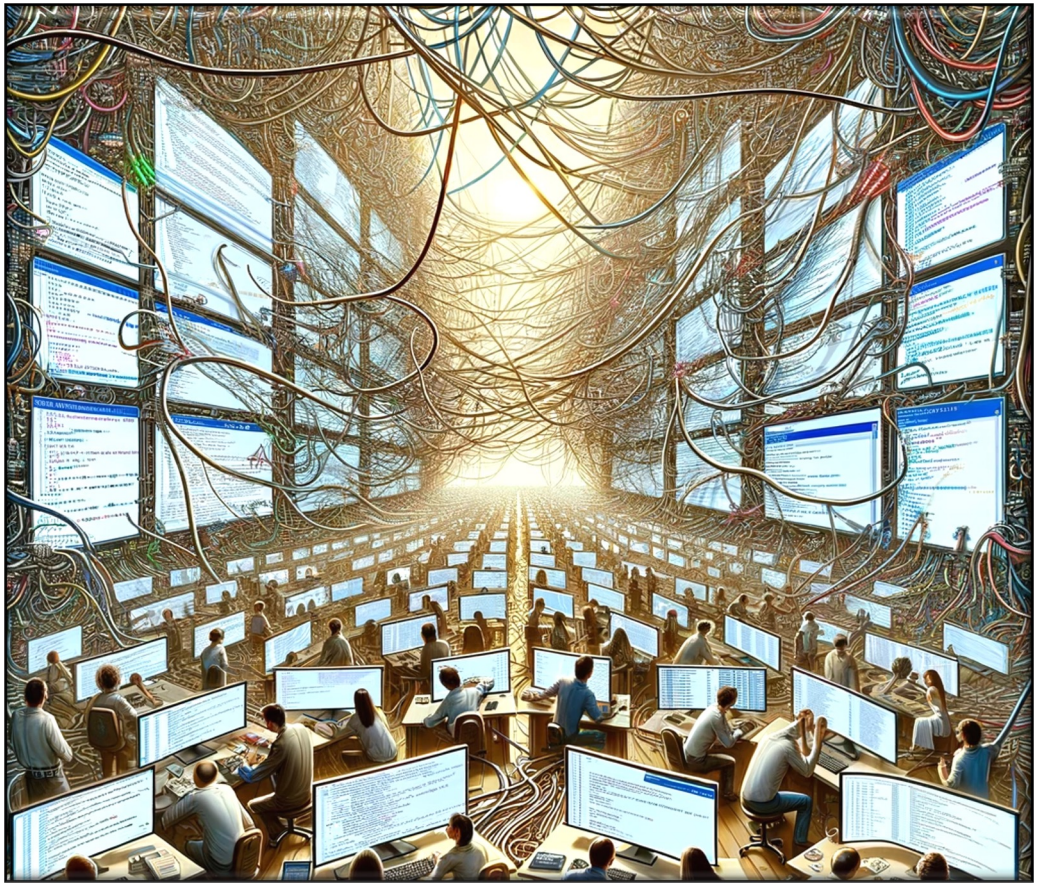
- Reactive programming is not appropriate in all situations
  - *Complexity*
    - Java developers are more familiar with the OO style than the reactive style
      - The reactive style has a steeper learning curve



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

Cons of Java Reactive Streams Platforms

- Reactive programming is not appropriate in all situations
  - *Complexity*
  - *Debugging*
    - Can be harder due to async & concurrent operations



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

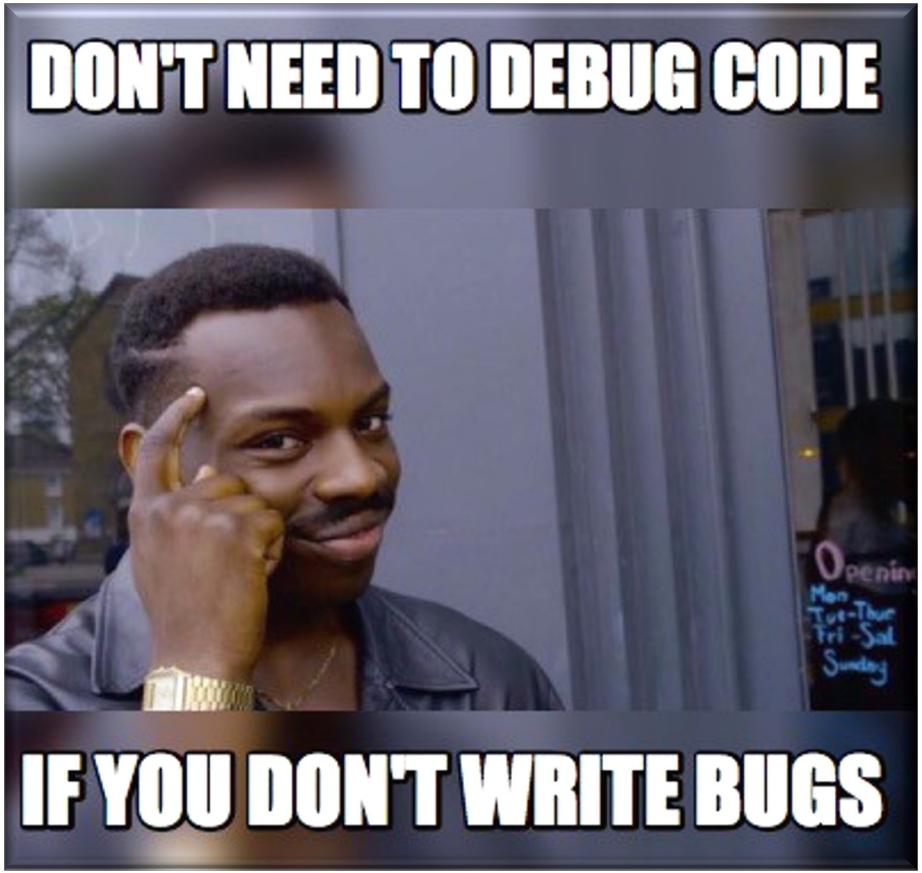
Cons of Java Reactive Streams Platforms

- Reactive programming is not appropriate in all situations
  - *Complexity*
  - *Debugging*
    - Can be harder due to async & concurrent operations
    - There's also often a lack of meaningful stack traces

See medium.com/digitalfrontiers/debugging-basics-in-project-reactor-5ef762c23df4

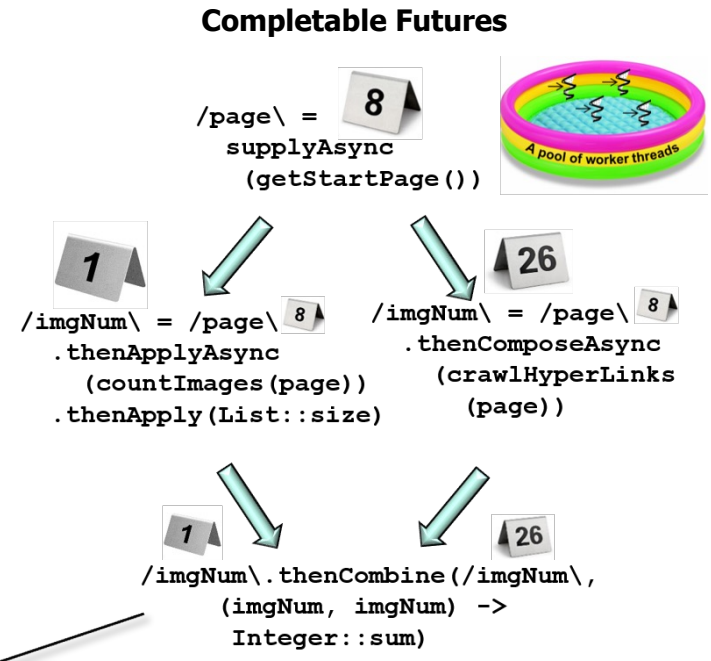
Cons of Java Reactive Streams Platforms

- Reactive programming is not appropriate in all situations
  - *Complexity*
  - *Debugging*
    - Can be harder due to async & concurrent operations
    - There's also often a lack of meaningful stack traces



Cons of Java Reactive Streams Platforms

- Reactive programming is not appropriate in all situations
  - *Complexity*
  - *Debugging*
    - Can be harder due to async & concurrent operations
    - There's also often a lack of meaningful stack traces



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

Cons of Java Reactive Streams Platforms

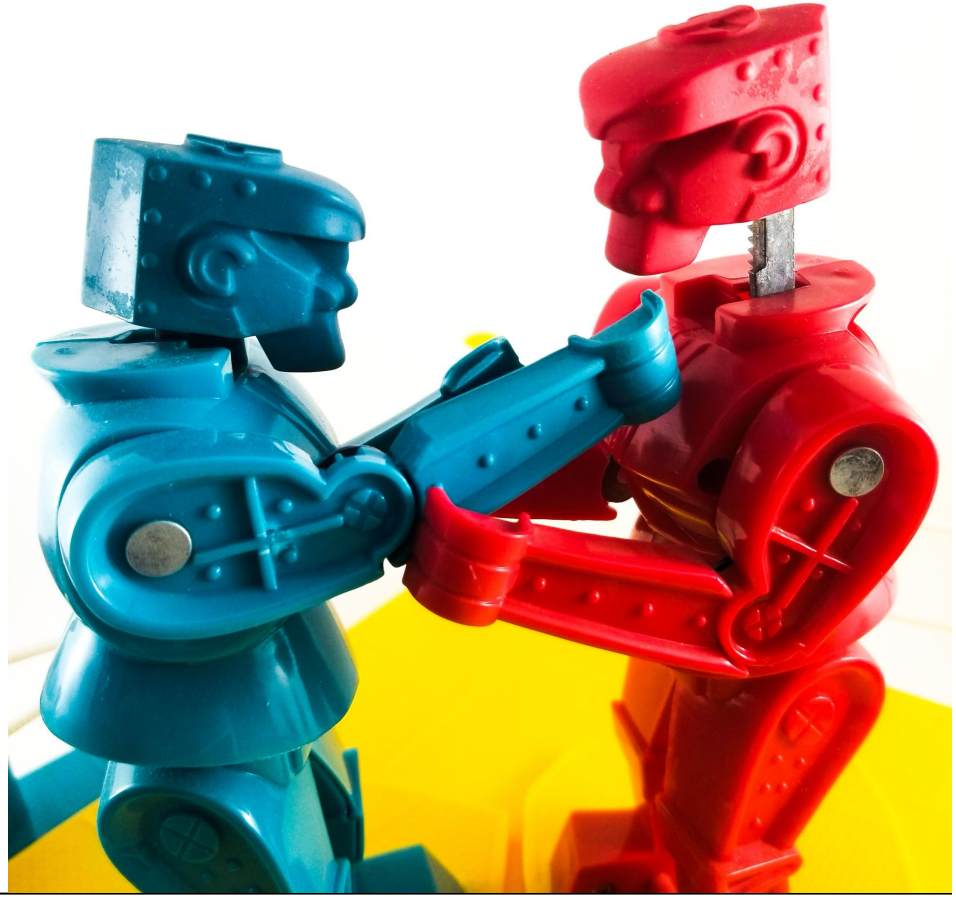
- Reactive programming is not appropriate in all situations
  - *Complexity*
  - *Debugging*

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

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

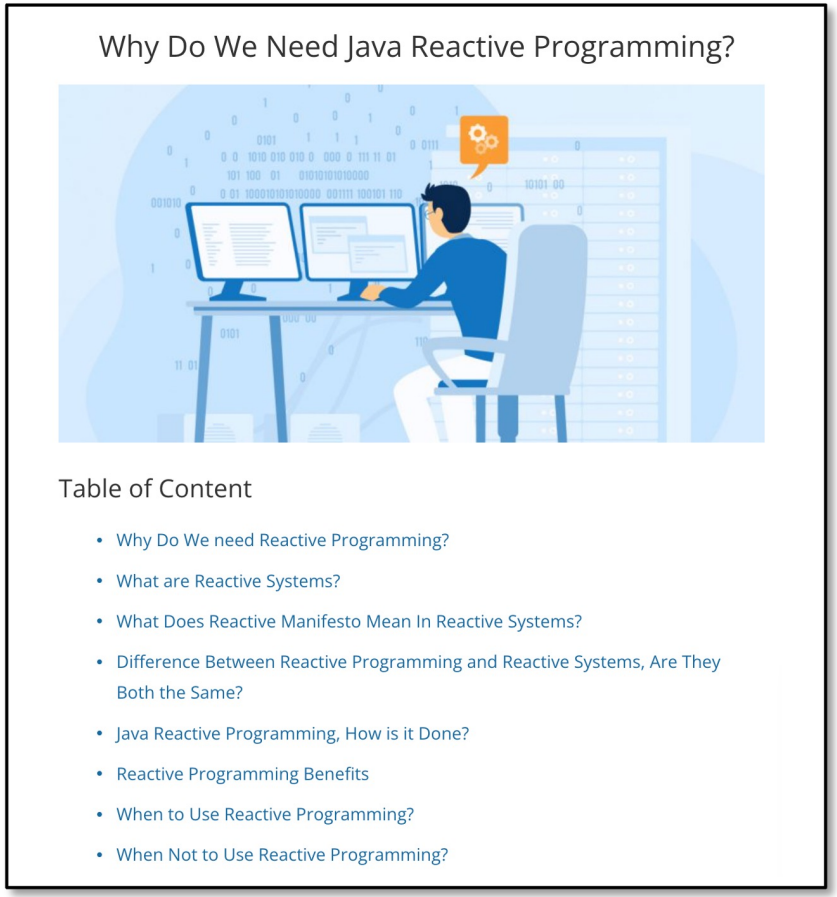
Cons of Java Reactive Streams Platforms

- There are various perspectives on Java reactive programming vs. Java structured concurrency!



Cons of Java Reactive Streams Platforms

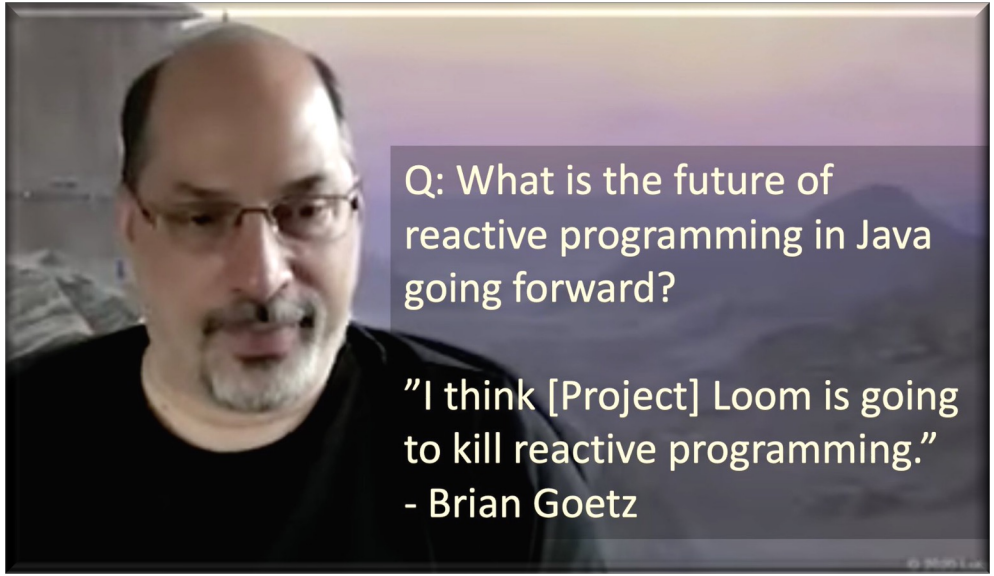
- There are various perspectives on Java reactive programming vs. Java structured concurrency!
  - Pro



See www.tatvasoft.com/blog/java-reactive-programming

Cons of Java Reactive Streams Platforms

- There are various perspectives on Java reactive programming vs. Java structured concurrency!
  - Con



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

End of Evaluating Java Programming Paradigms