# Evaluating Java Programming Paradigms


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

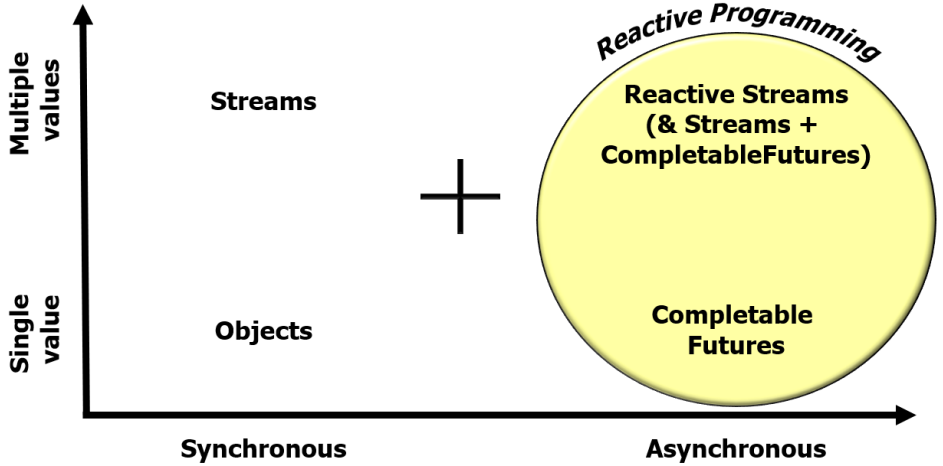
Professor of Computer Science

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, & sync/ async functional programming



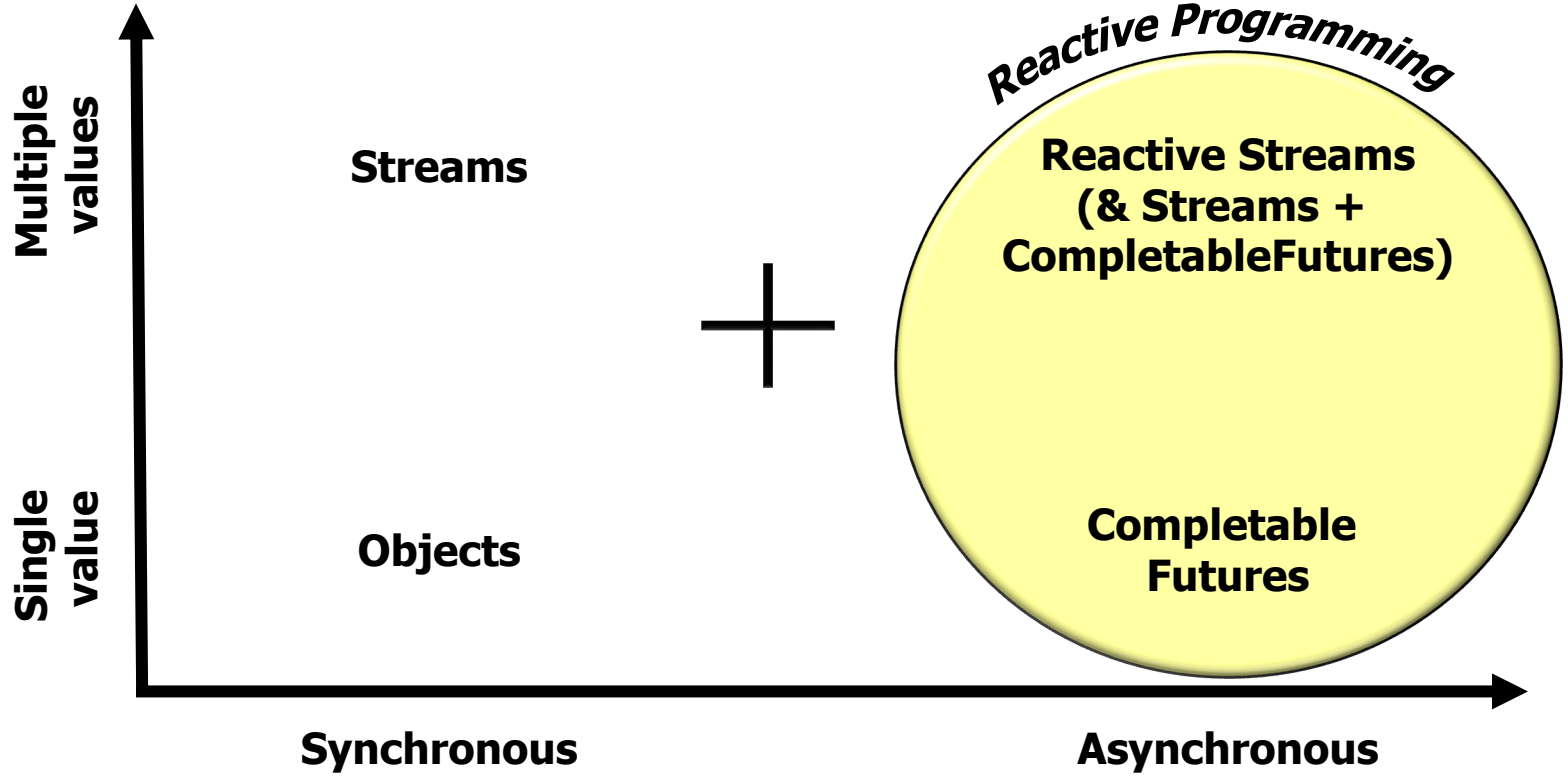
# Learning Objectives in this Part of the Lesson

- 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

# Comparing Reactive Programming with Other Paradigms

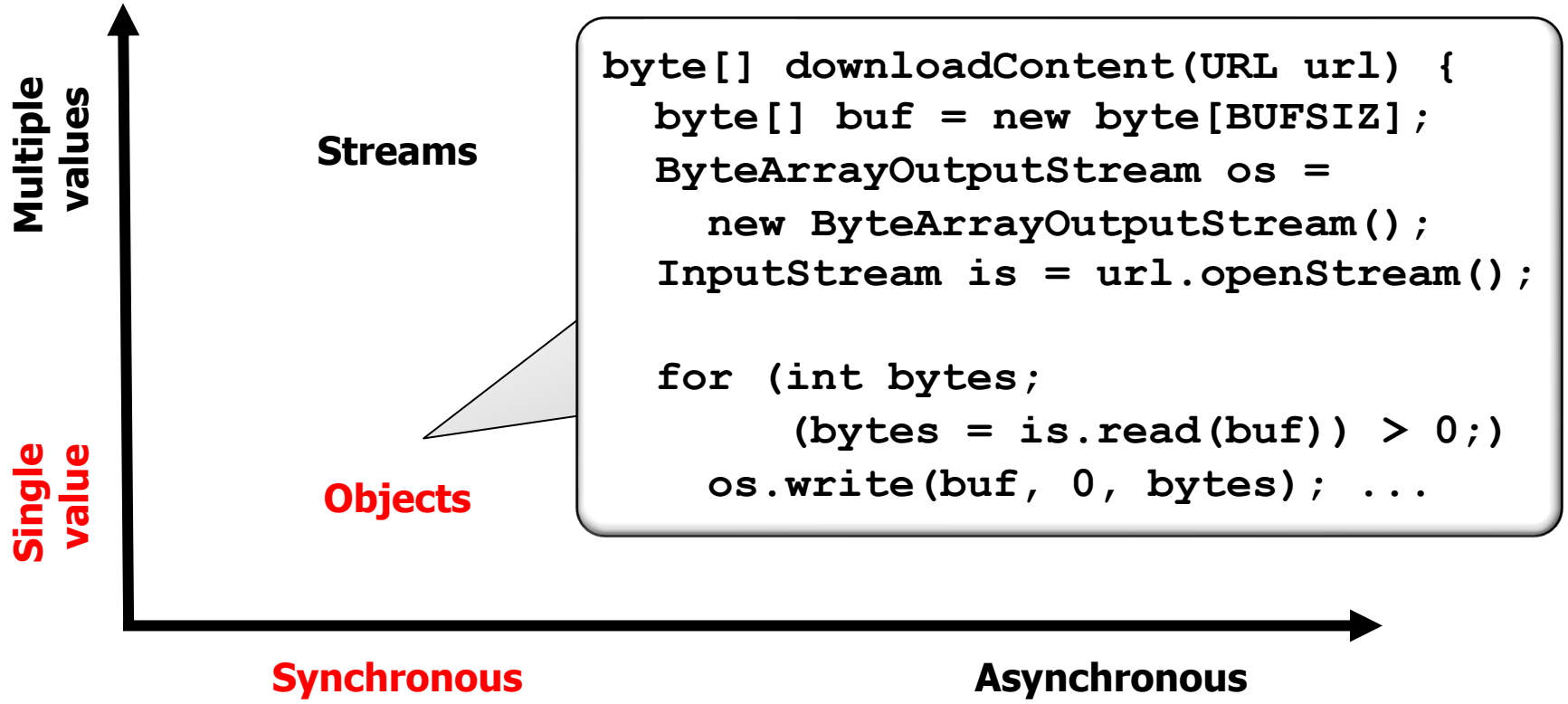
# Comparing Reactive Programming with Other Paradigms

- Reactive programming is one of several Java programming paradigms



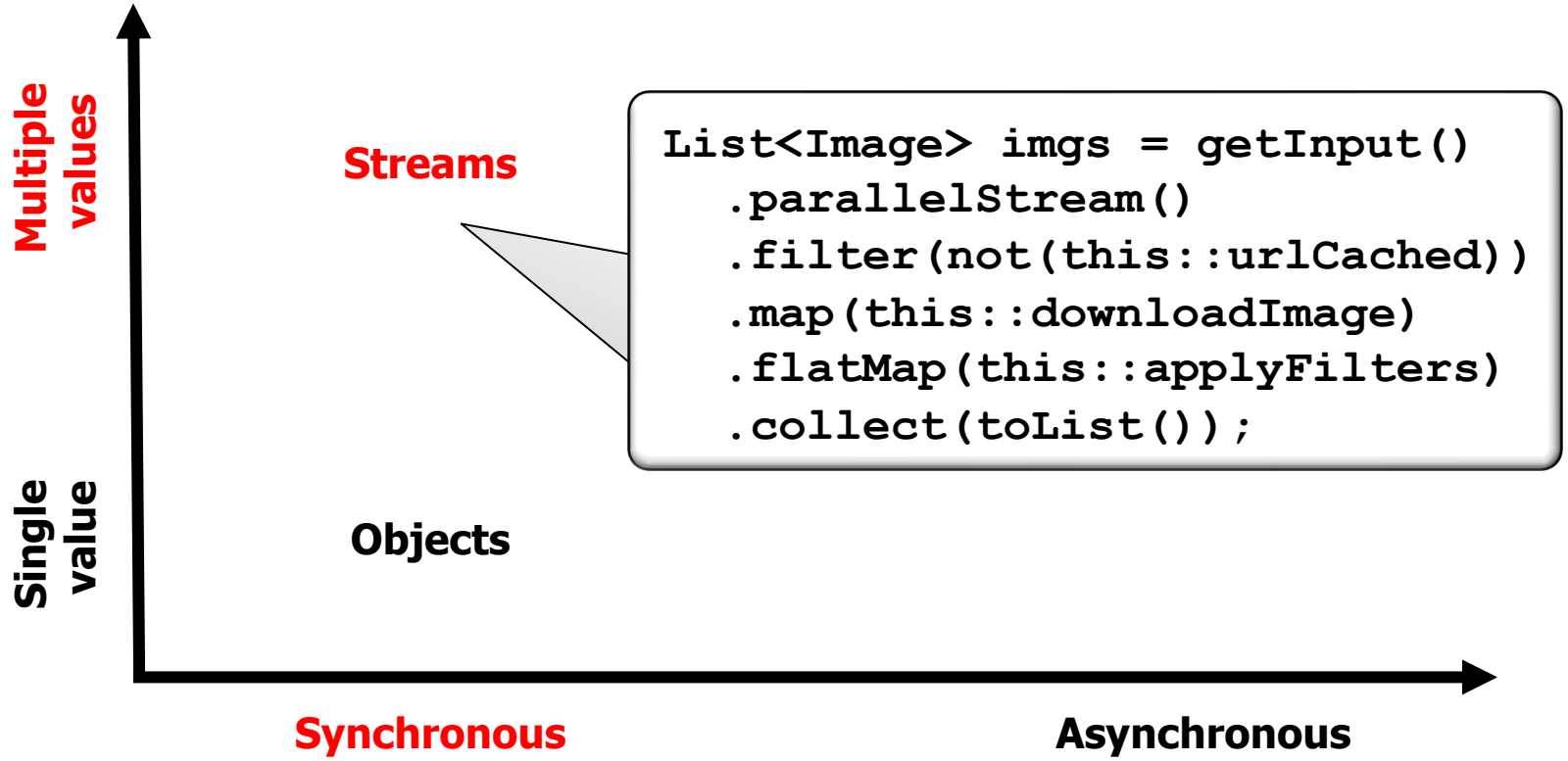
# Comparing Reactive Programming with Other Paradigms

- Reactive programming is one of several Java programming paradigms



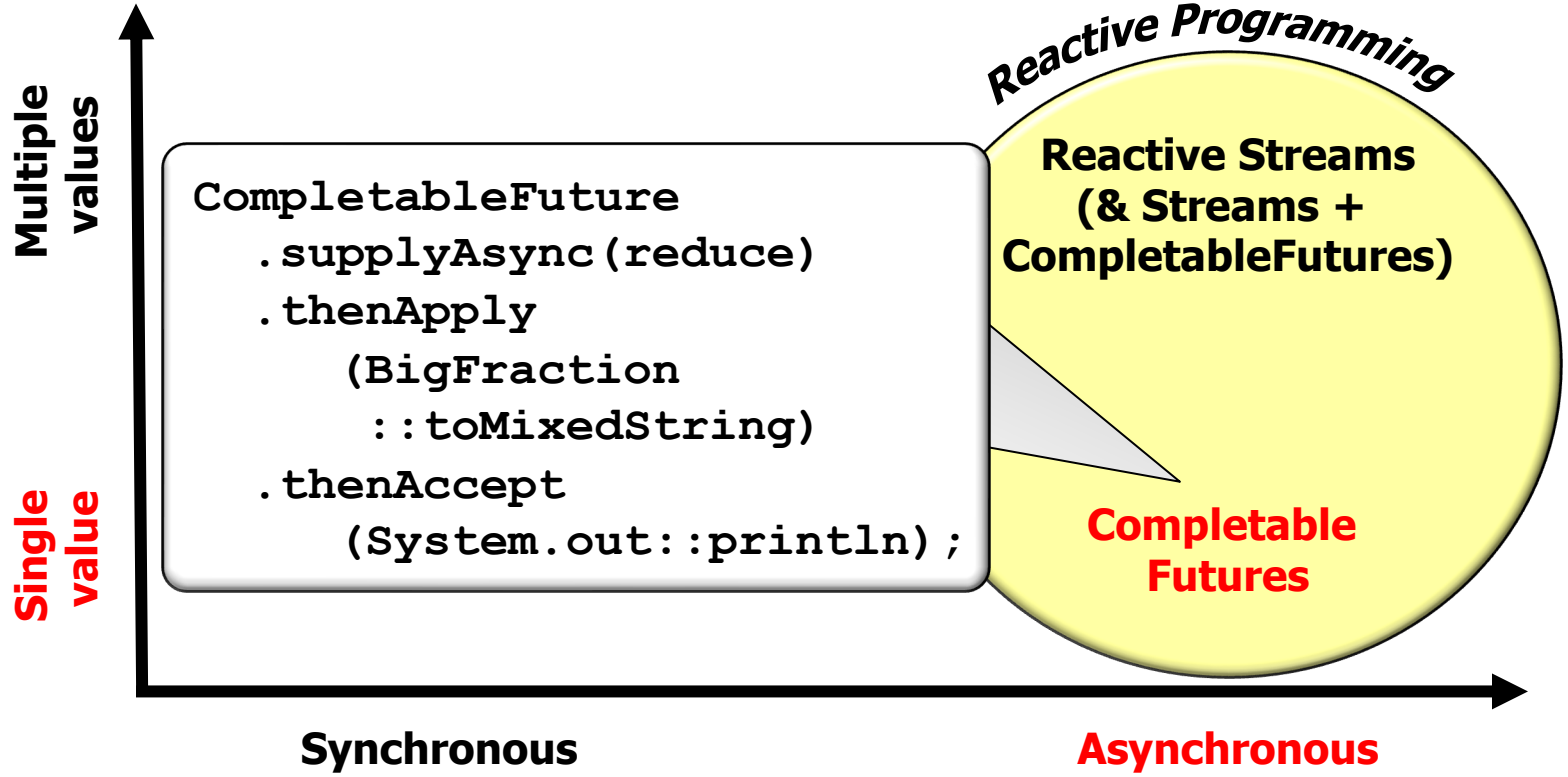
# Comparing Reactive Programming with Other Paradigms

- Reactive programming is one of several Java programming paradigms



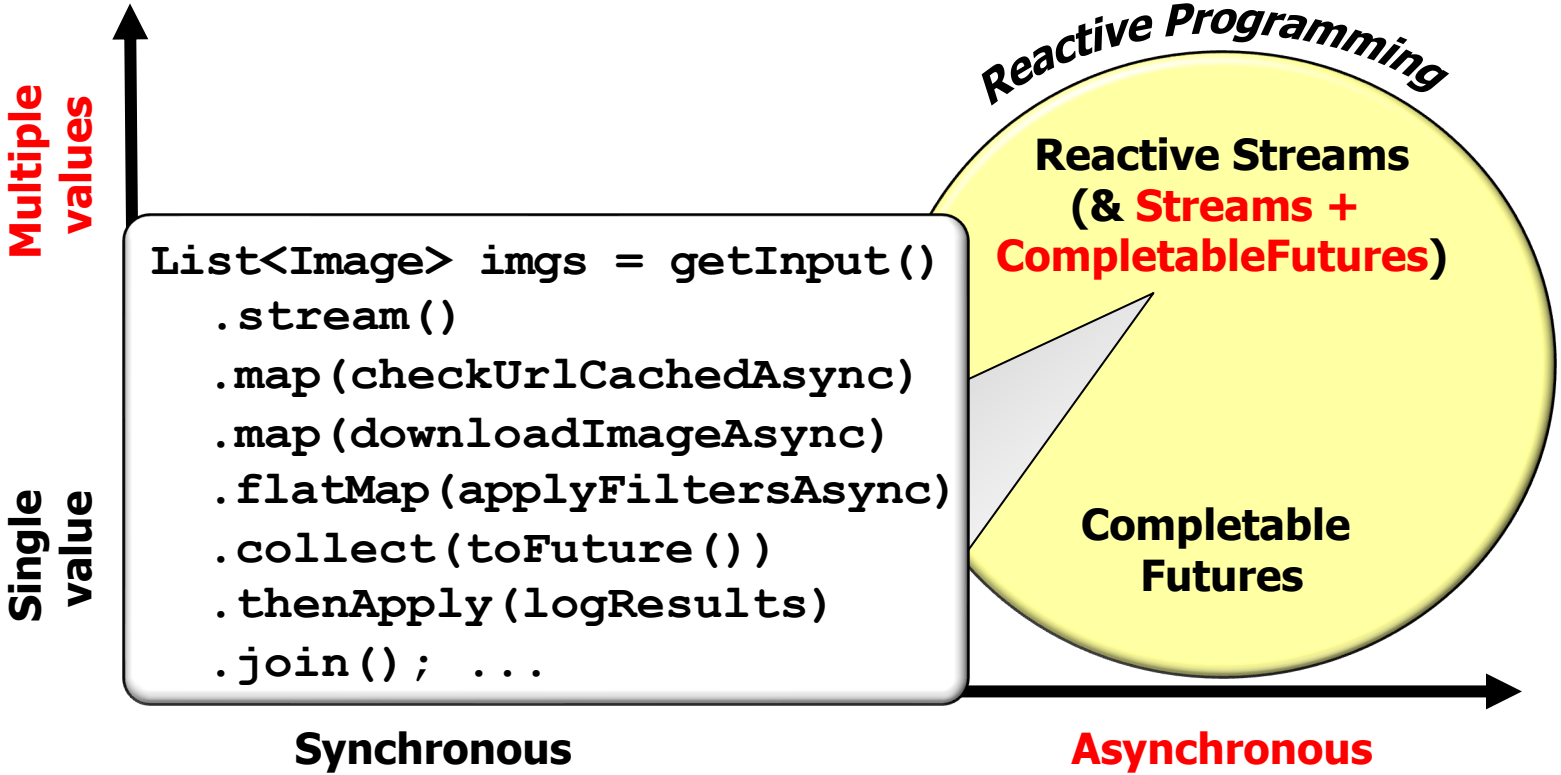
# Comparing Reactive Programming with Other Paradigms

- Reactive programming is one of several Java programming paradigms



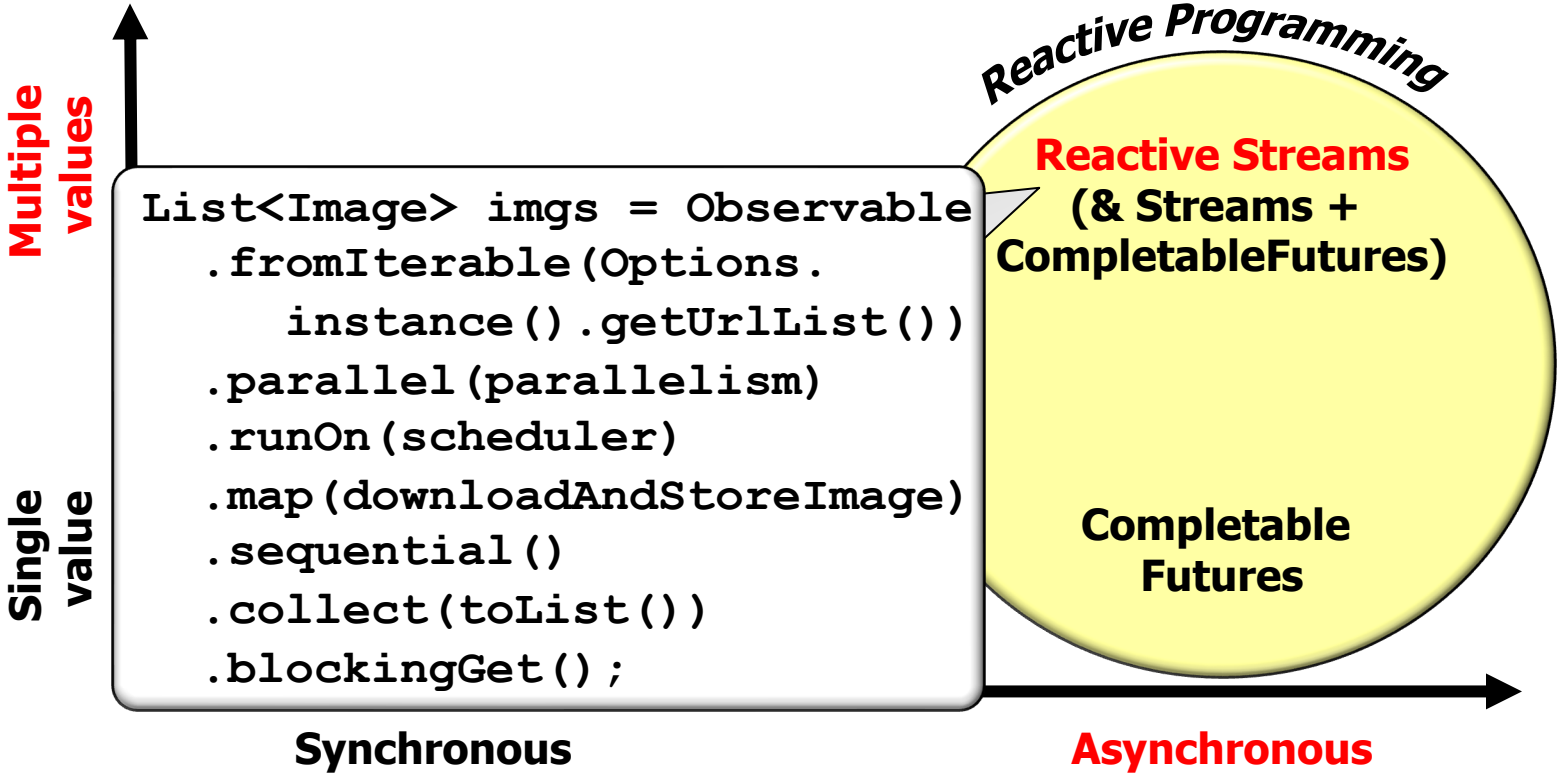
# Comparing Reactive Programming with Other Paradigms

- Reactive programming is one of several Java programming paradigms



# Comparing Reactive Programming with Other Paradigms

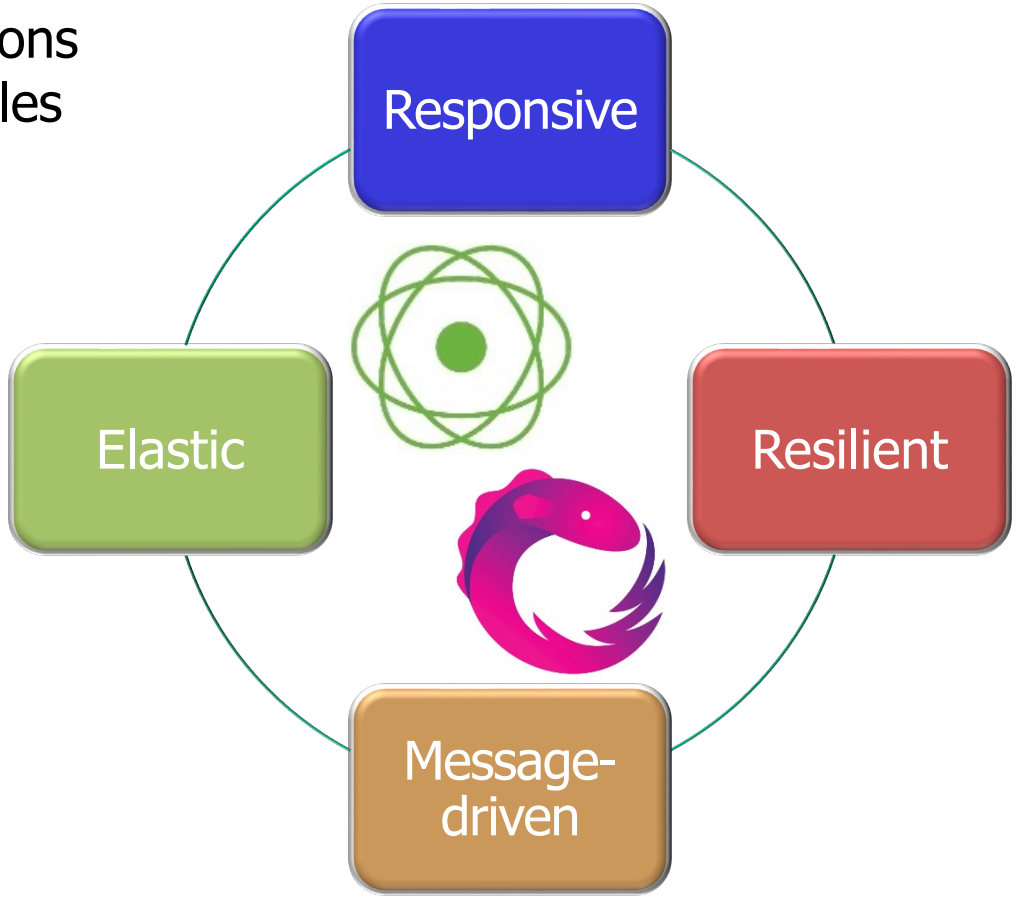
- Reactive programming is one of several Java programming paradigms



# 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



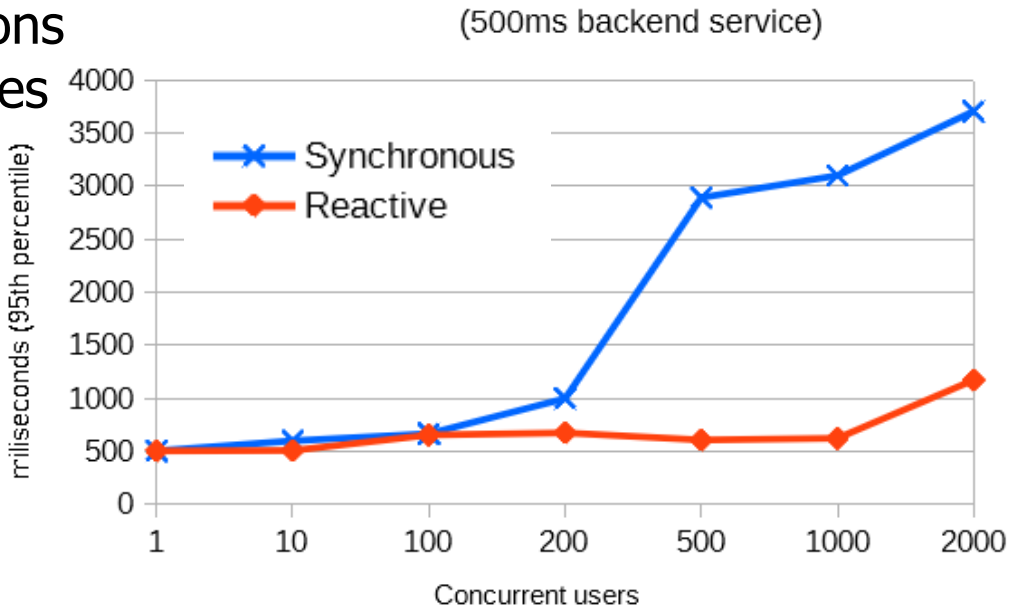
# Pros & Cons of Java Reactive Streams Platforms

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

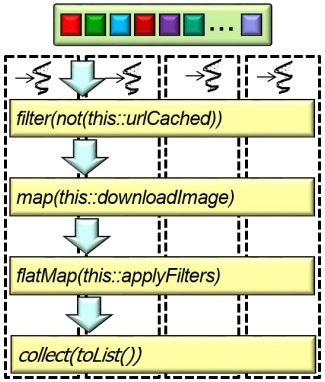
- Java reactive streams implementations apply reactive programming principles to achieve several benefits
  - Support concurrency with a minimal number of threads via a range of thread pools
  - Explicit synchronization and/or threading is rarely needed when applying these frameworks

Alleviates many accidental & inherent complexities of concurrency/parallelism

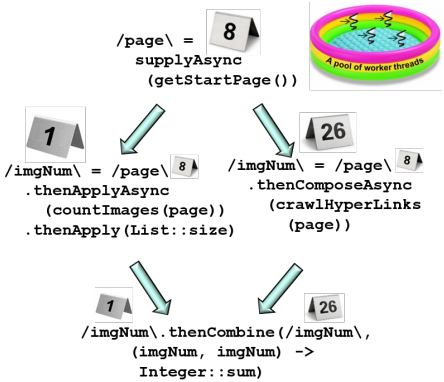
# Pros & Cons of Java Reactive Streams Platforms

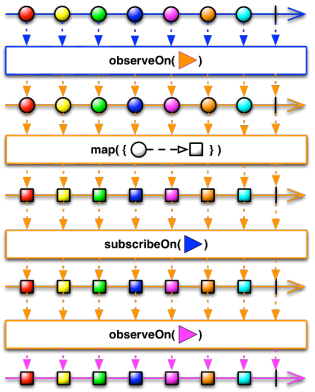
- Java reactive streams implementations apply reactive programming principles to achieve several benefits
  - Support concurrency with a minimal number of threads via a range of thread pools
  - Explicit synchronization and/or threading is rarely needed when applying these frameworks

**Parallel Streams**




**Completable Futures**



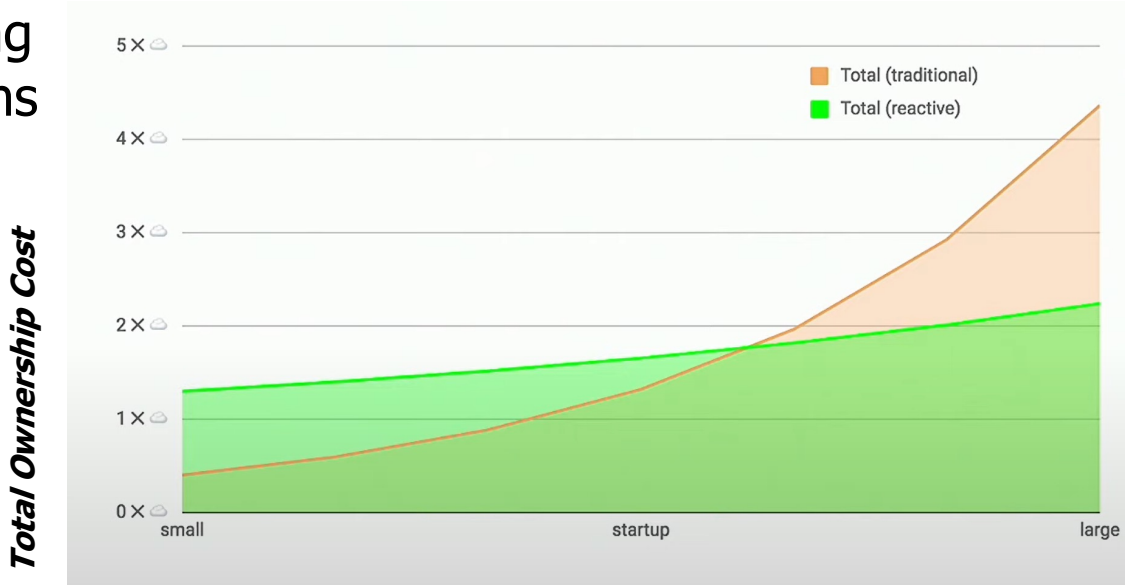




**Reactive Streams**

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

# Pros & Cons of Java Reactive Streams Platforms

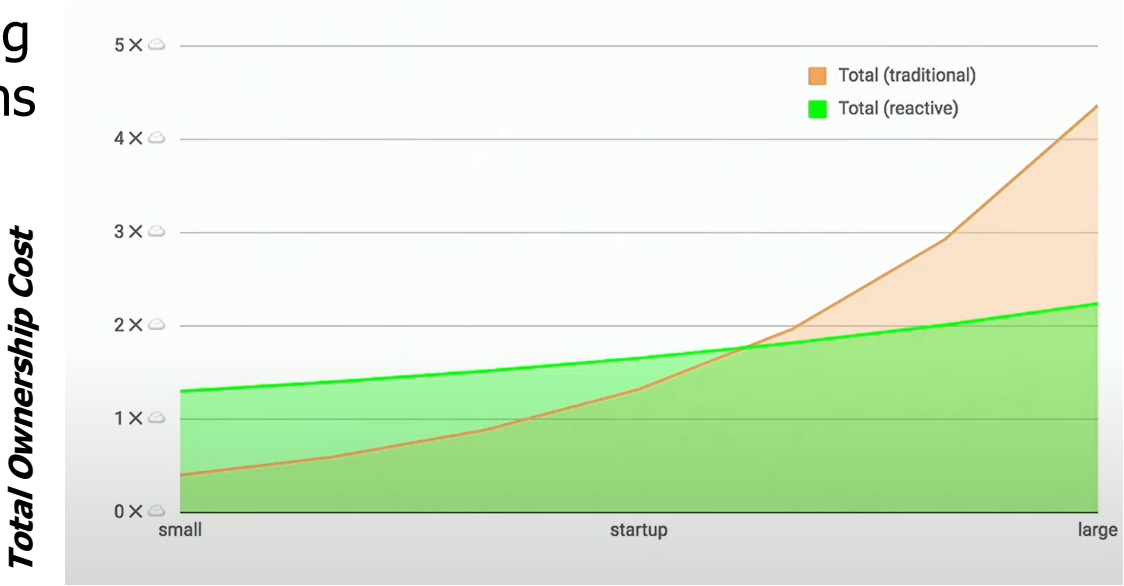
- However, reactive programming isn't appropriate in all situations




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

# Pros & Cons of Java Reactive Streams Platforms

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


It's essential to master the learning curve of reactive programming!

# End of Evaluating Java Programming Paradigms