Async Programming

Simon Robinson

Software Developer

@TechieSimon www.SimonRobinson.com

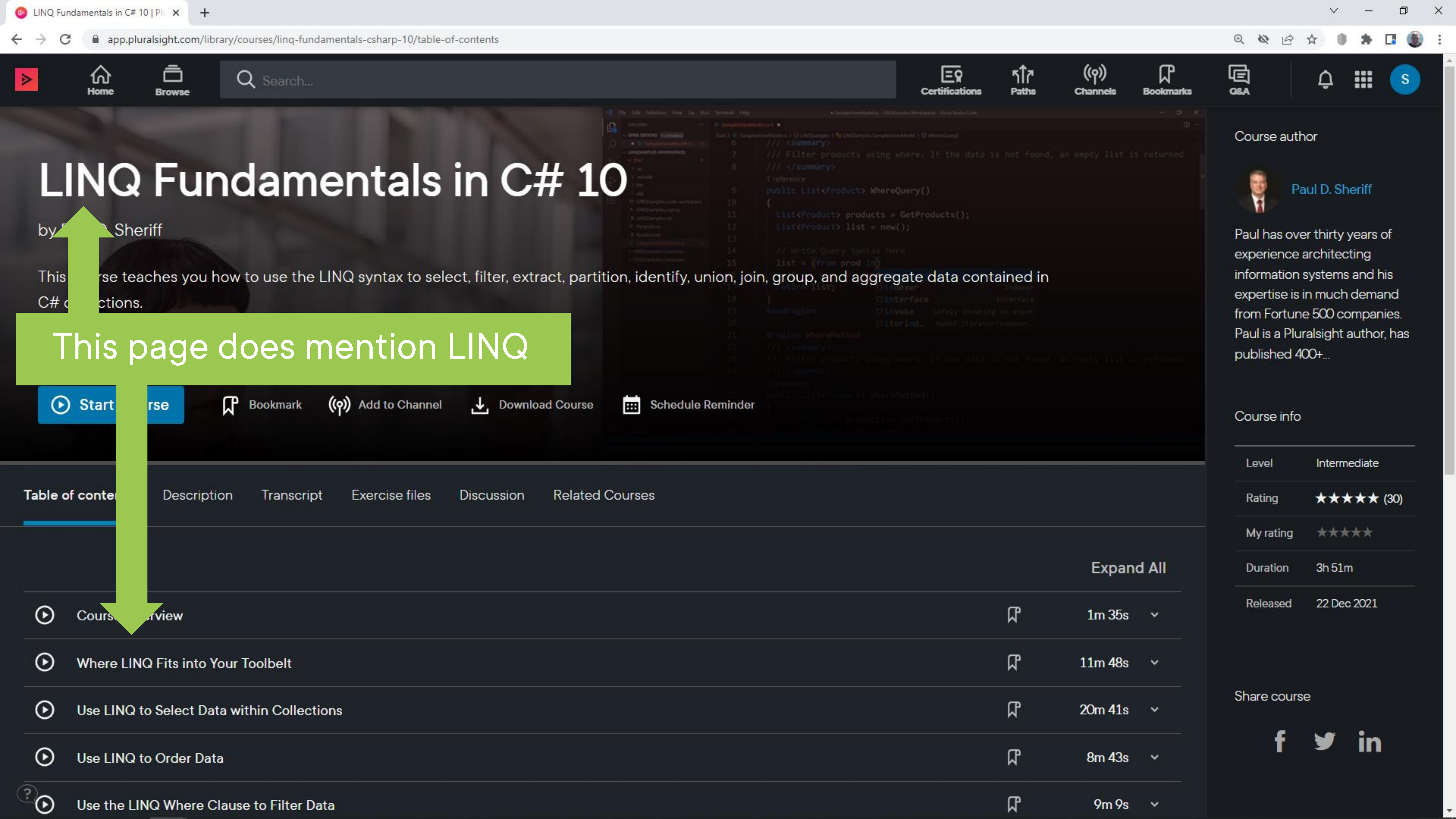
Overview

Launch simultaneous async operations

- Wait until they have all completed
- Alternatively, show each result as soon as it appears

Protect thread-shared data from corruption

Generate and consume async streams



LINQ Fundamentals in C# 10 | Pl
app.pluralsight.com/library/courses/linq-fundamentals-csharp-10/table-of-contents
Home
Browse
Search...
Certifications
Paths
Channels
Bookmarks
Q&A
S

LINQ Fundamentals in C# 10

by Sheriff

This rse teaches you how to use the LINQ syntax to select, filter, extract, partition, identify, union, join, group, and aggregate data contained in C# ctions.

This page does mention LINQ

Cours rview	1m 35s
Where LINQ Fits into Your Toolbelt	11m 48s
Use LINQ to Select Data within Collections	20m 41s
Use LINQ to Order Data	8m 43s
Use the LINQ Where Clause to Filter Data	9m 9s

Course author

Paul D. Sheriff

Paul has over thirty years of experience architecting information systems and his expertise is in much demand from Fortune 500 companies. Paul is a Pluralsight author, has published 400+...

Course info

Level	Intermediate
Rating	★★★★★ (30)
My rating	☆☆☆☆☆
Duration	3h 51m
Released	22 Dec 2021

Share course

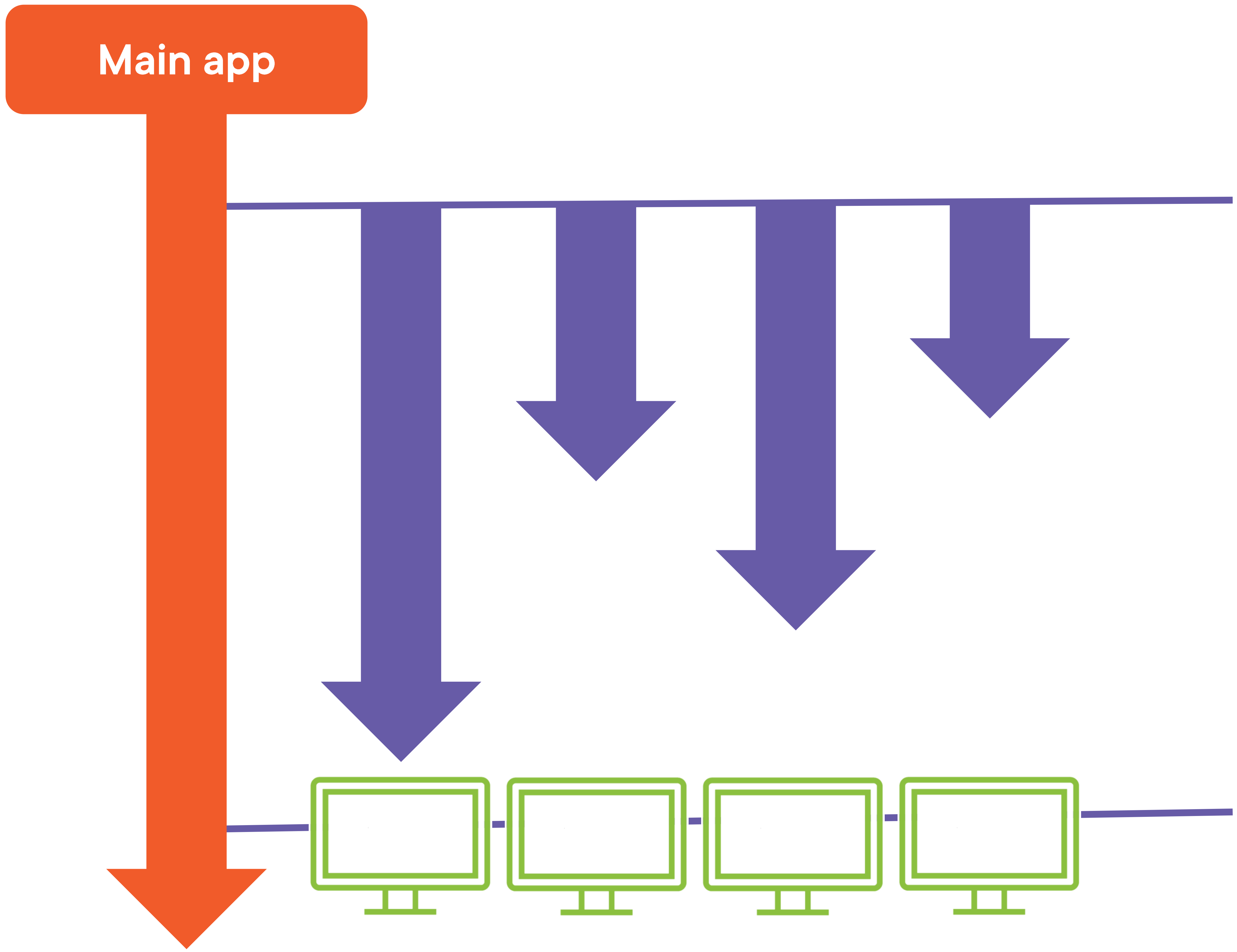
Waiting for Simultaneous Async Tasks

Demo

Use `async` and `await` to search for one course

- Then search for lots of courses at the same time
- Display results when all searches are done

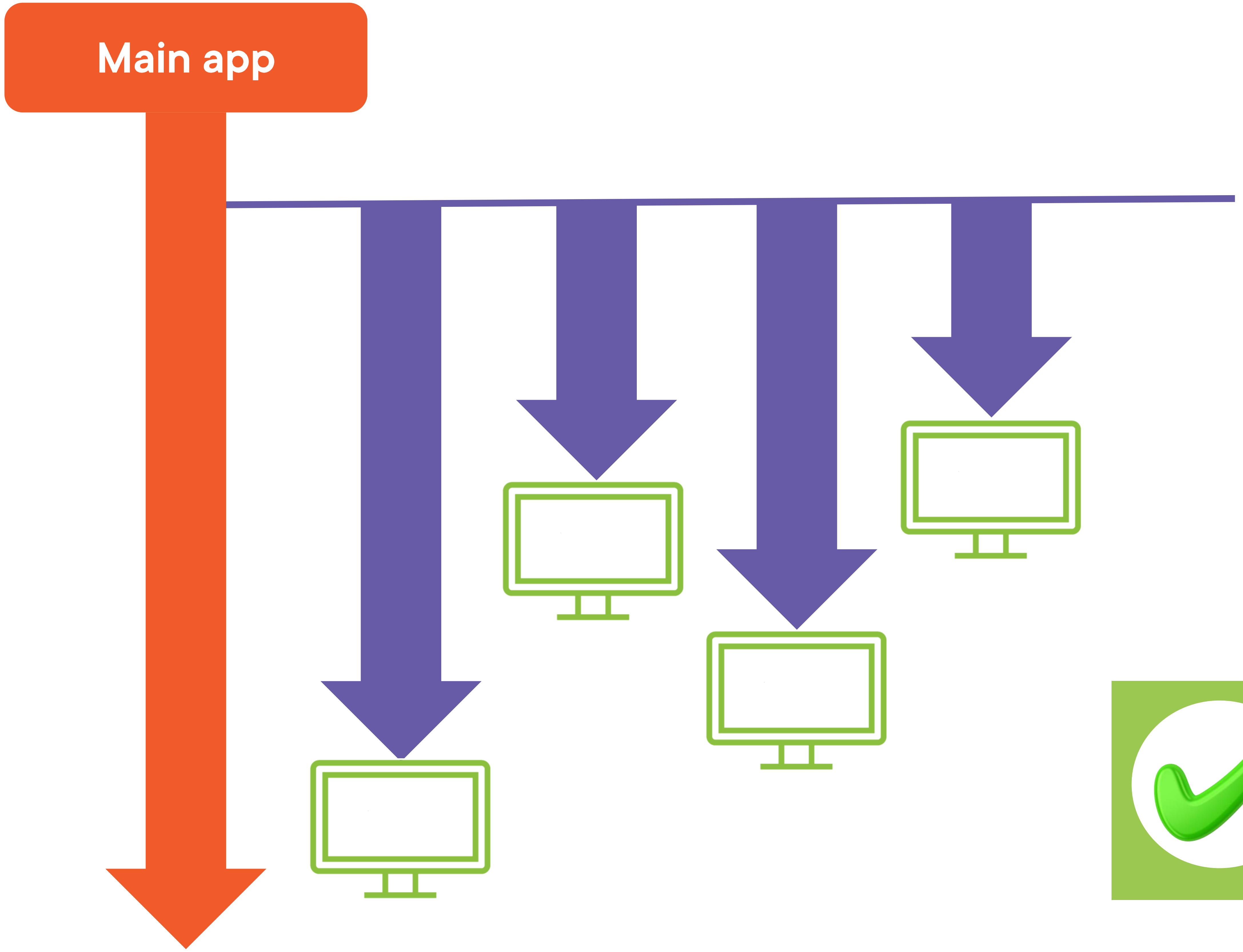
Displaying Results as Each Operation Completes



Button clicked

Async searches launched

User sees results when everything complete



Button clicked

Async searches launched

✓ Better: Show the user each result as it becomes ready

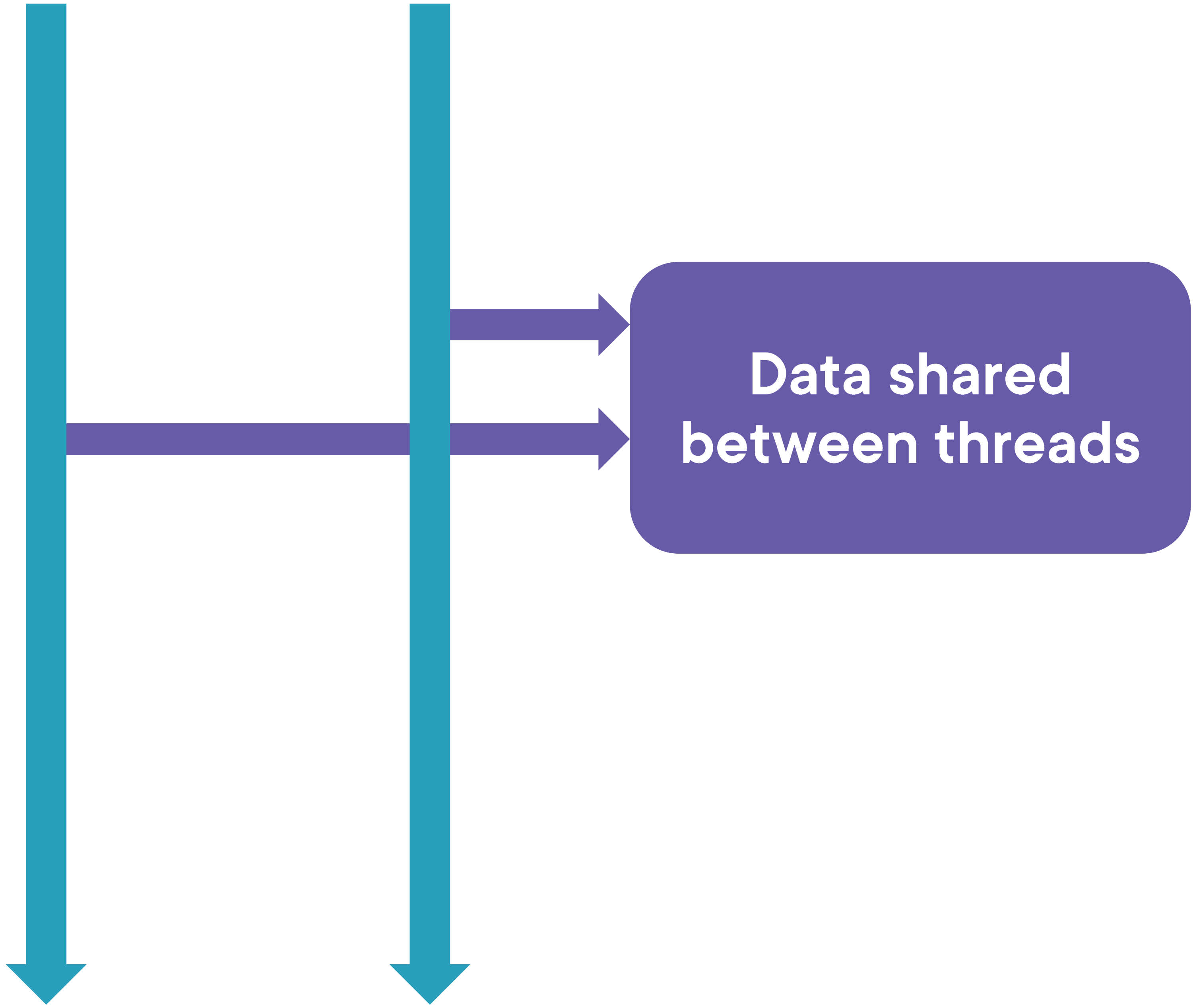
Demo

Modify the app to show results as they arrive

- Best solution uses progress reporting
- You'll see two tempting 'traps' first

Protecting Shared Data from Corruption

Sharing Data between Threads



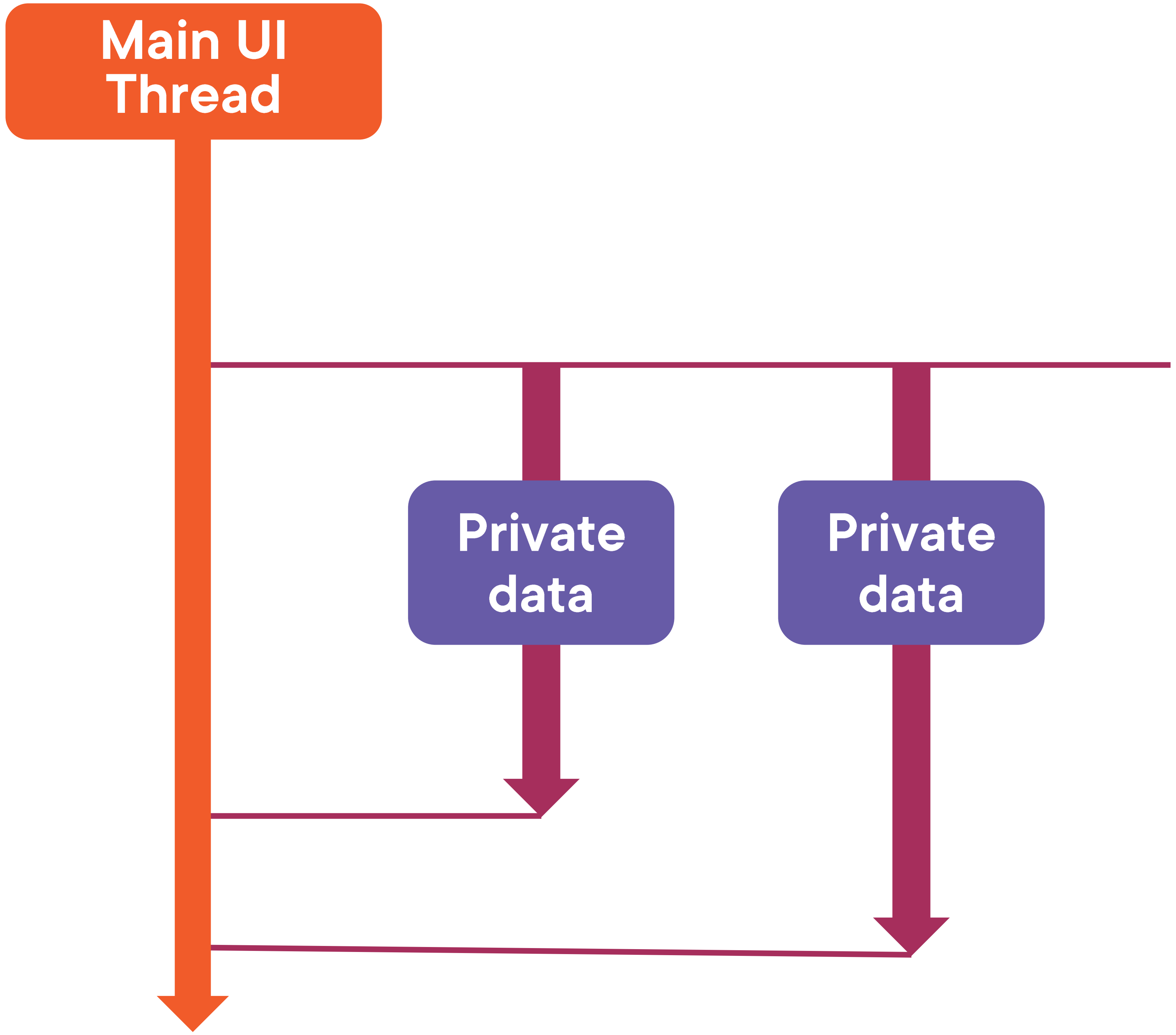
Data visible to multiple threads could become corrupted

- For example, if two threads simultaneously write to the same data

- Not an issue for read-only data

You must protect shared writeable data using thread synchronization

The Demo Hasn't Shared Data



Shared data has not been an issue in the demo

- Because each async operation has its own data

- The async search code does not write to shared data

Demo

Each search task must save pages to a shared cache

- Add code to protect this data
  - Using locks
  - Using thread-safe types

The Best Way to Protect Your Data

The Best Way to Protect Your Data

✓ Design your code so the data doesn’t need thread synchronization protection

Why Avoid Sharing Mutable Data between Threads?

Thread synchronization is hard to get right

- If you get it wrong, debugging thread synchronization issues is particularly difficult

Thread synchronization usually hurts performance

Good Practices

As far as possible, give async operations their own private copy of mutable data

- It’s OK to share immutable data

If sharing mutable data is necessary, use thread synchronization to protect it

Async Streams

Async Tasks vs. Async Streams

Async Tasks	Async Streams
Use if operations need to run asynchronously	Use if enumerating something and enumeration data is slow arriving
Can run simultaneously	
Keep the app responsive	Keep the app responsive
Speed up the overall process	Don’t speed up the overall process

Demo

Enumerate data that is being supplied slowly

- This blocks the UI
- Modify to use an async stream
- This unblocks the UI

Summary

Async tasks

- Waiting for one task just requires async / `await`
- Waiting for multiple tasks requires explicitly using Task instances
  - await `Task.WhenAll()` to wait for completion
- Use `IProgress` infrastructure to display results as soon as each result arrives

Summary

Protecting data shared between threads

- Best solutions:
  - Avoid shared data
  - Use Thread-safe types
  - Use locks

Async streams

- To generate, use `yield return` to return `IAsyncEnumerable<T>`
- To consume, use `await foreach`