# Unleashing the Power of Pipelines

**Mark Heath**
SOFTWARE DEVELOPER

@mark_heath www.markheath.net

```
"10,5,0,8,10,1,4,0,10,1"
    .Split(',')
    .Select(int.Parse)
    .OrderBy(n => n)
    .Skip(3)
    .Sum()
```

# Chaining LINQ Extension Methods

**Create a “pipeline”**

# Overview

What are pipelines?

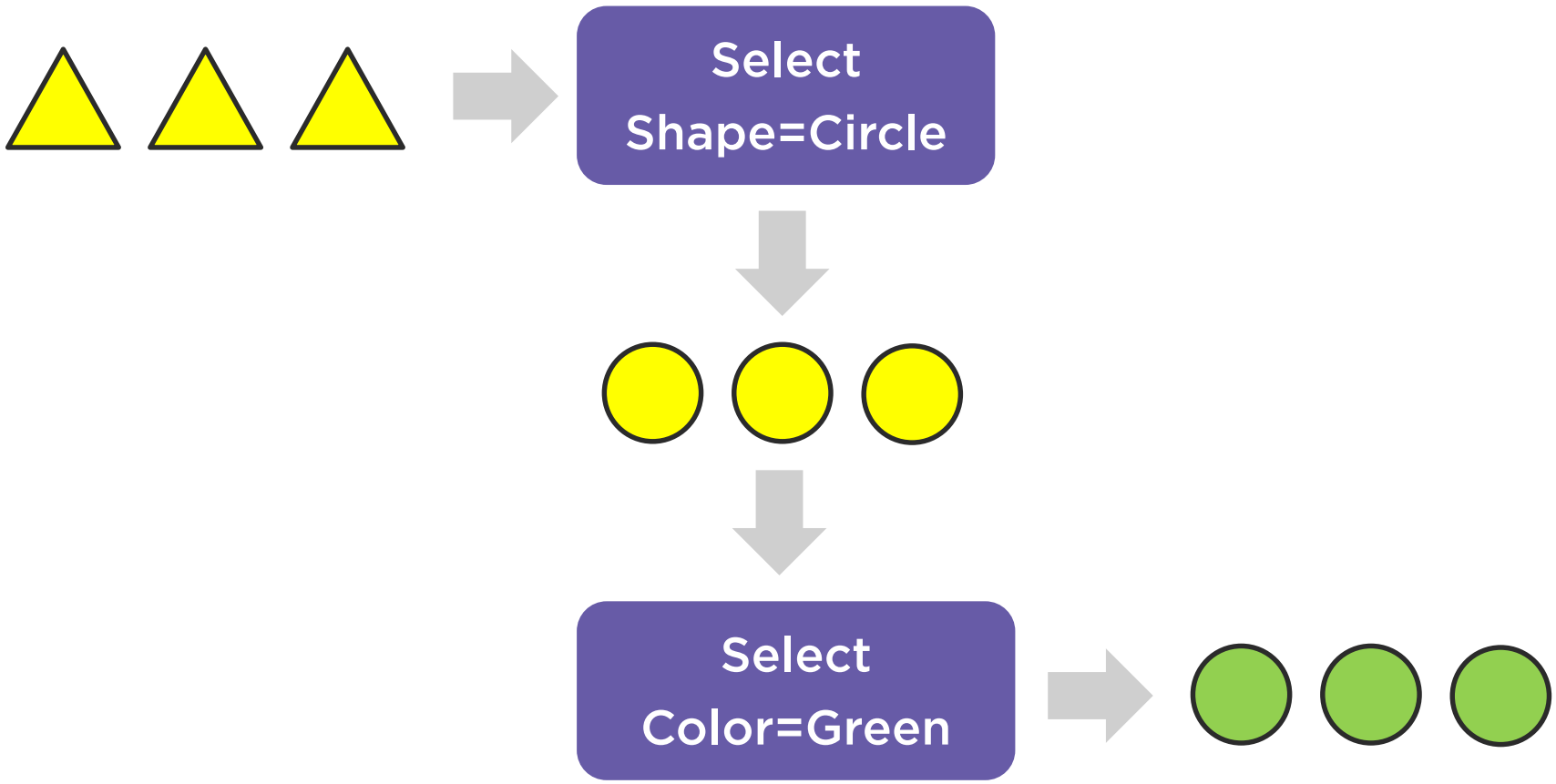
Solve some LINQ challenges

Real-world uses of pipelines

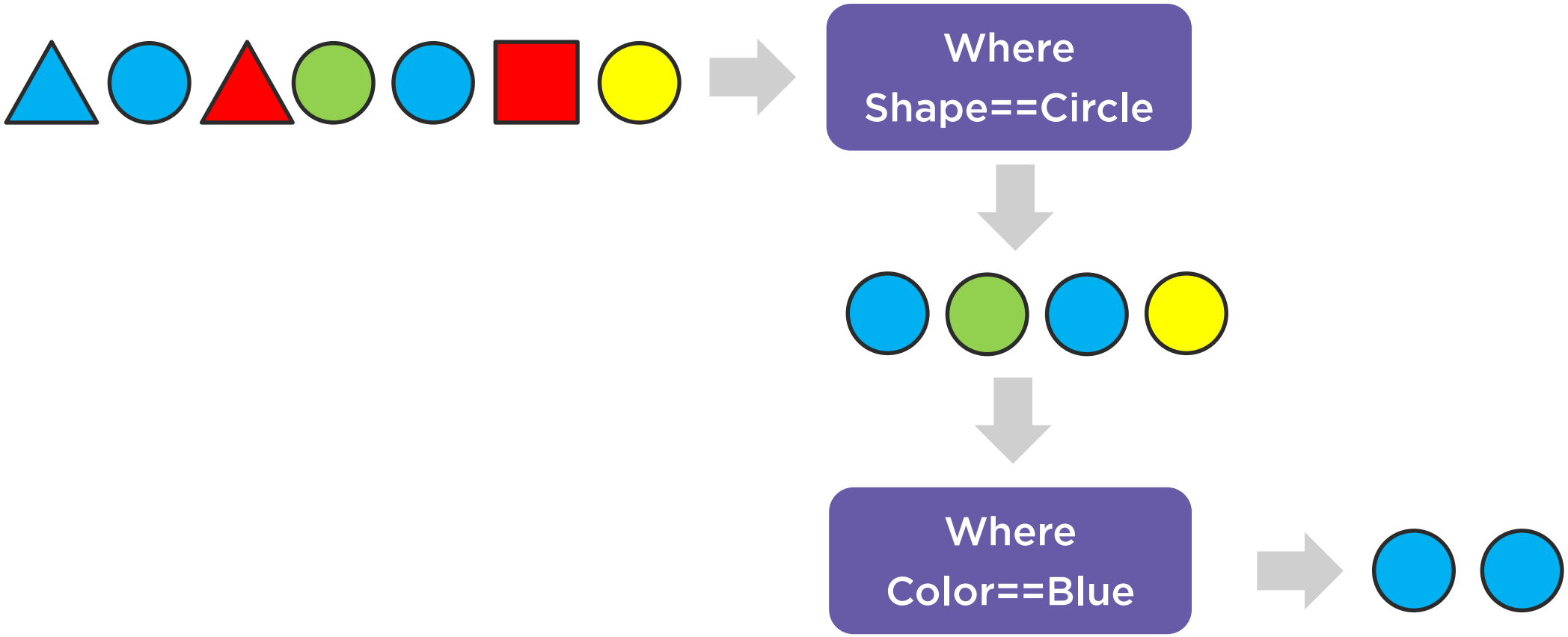
Pipeline building blocks

# Pipelines

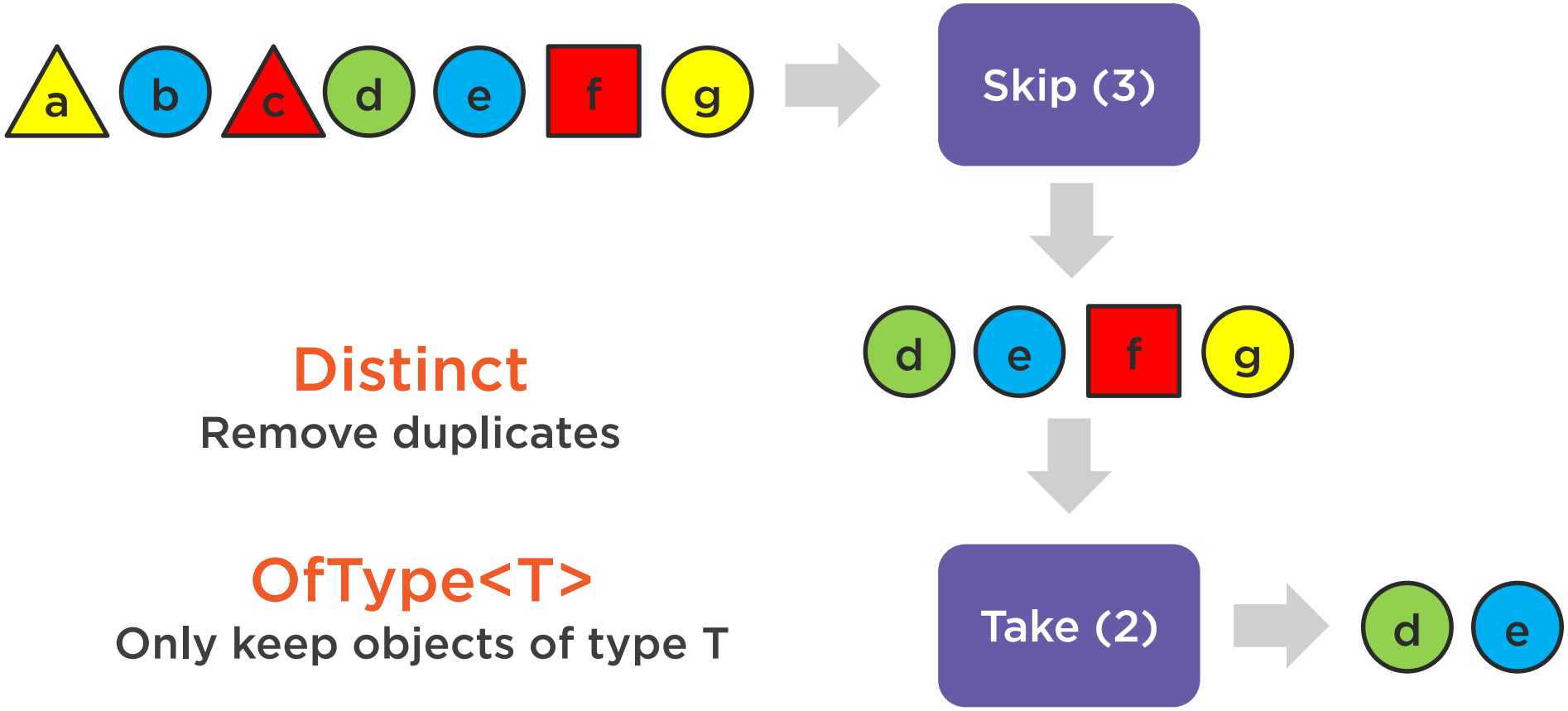
# Transforming Elements



# Filtering Elements



# Filtering Elements




**Distinct**

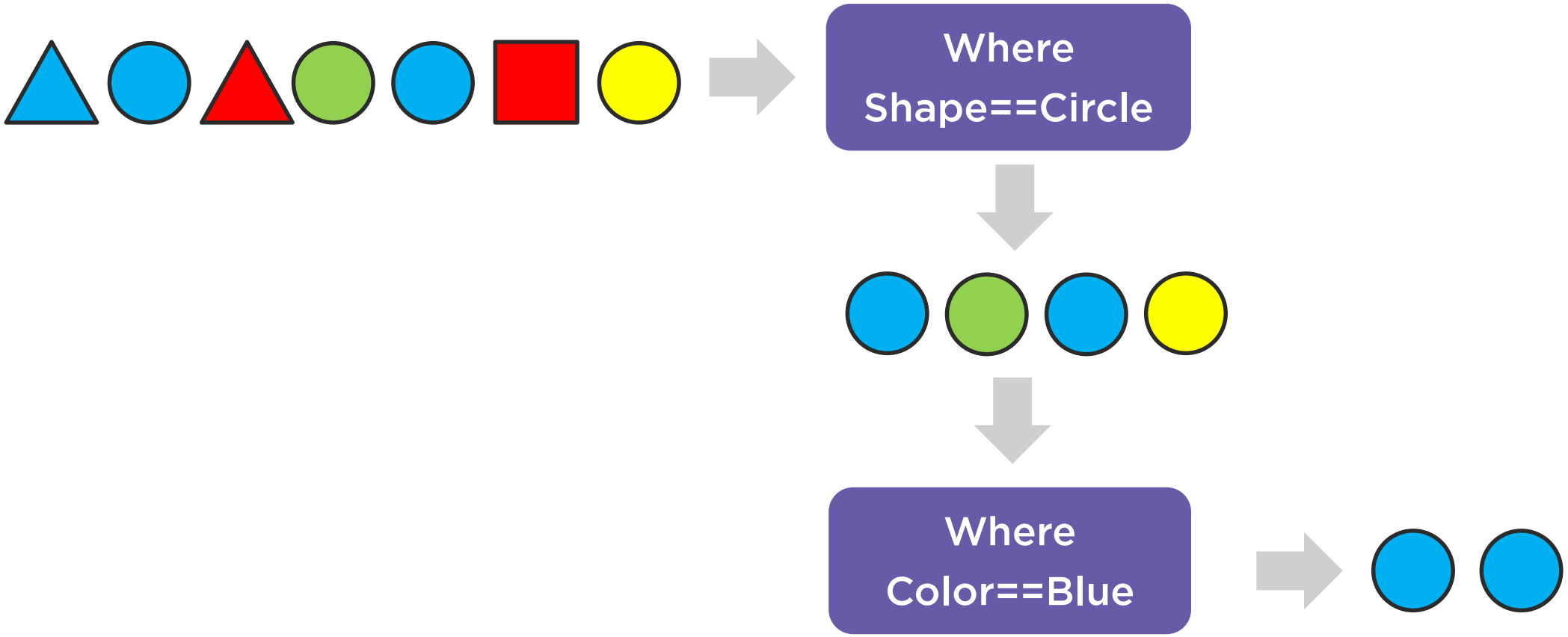
Remove duplicates

**OfType<T>**

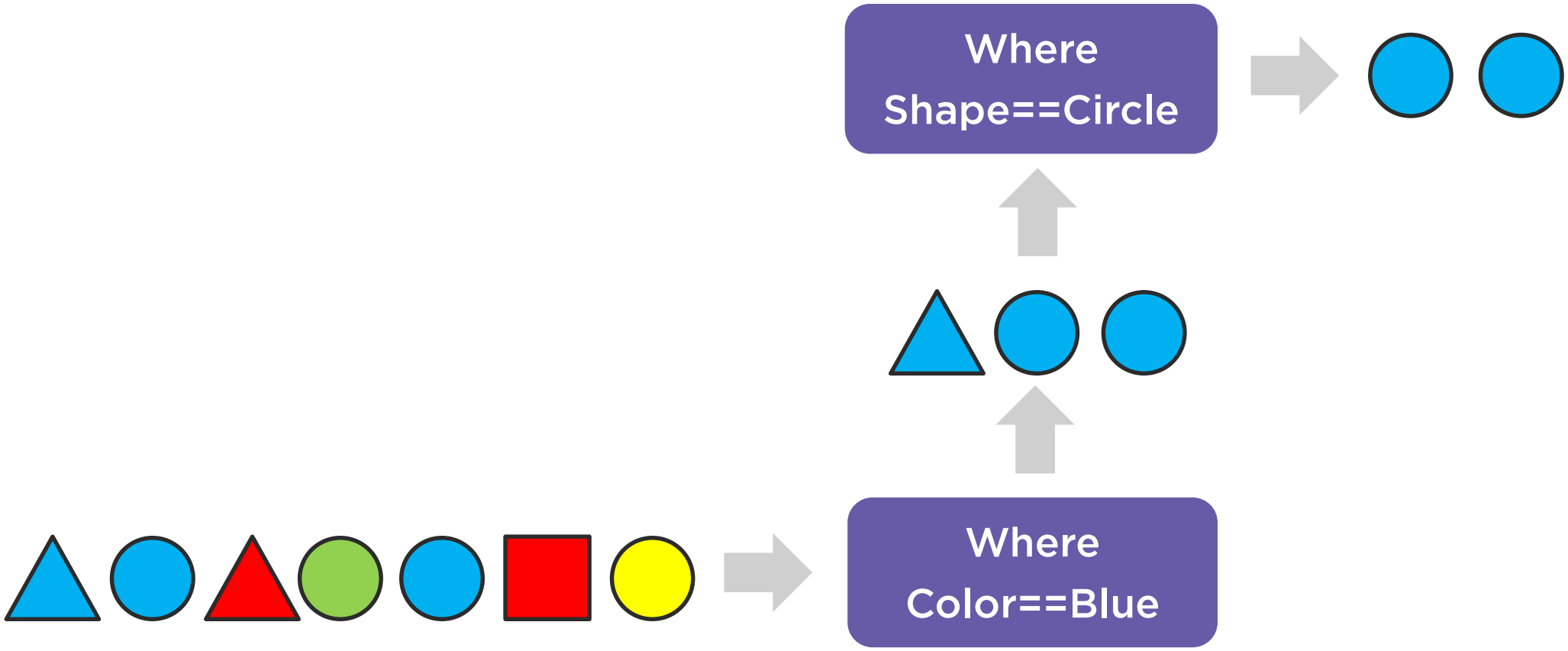
Only keep objects of type T

# Order is Important!

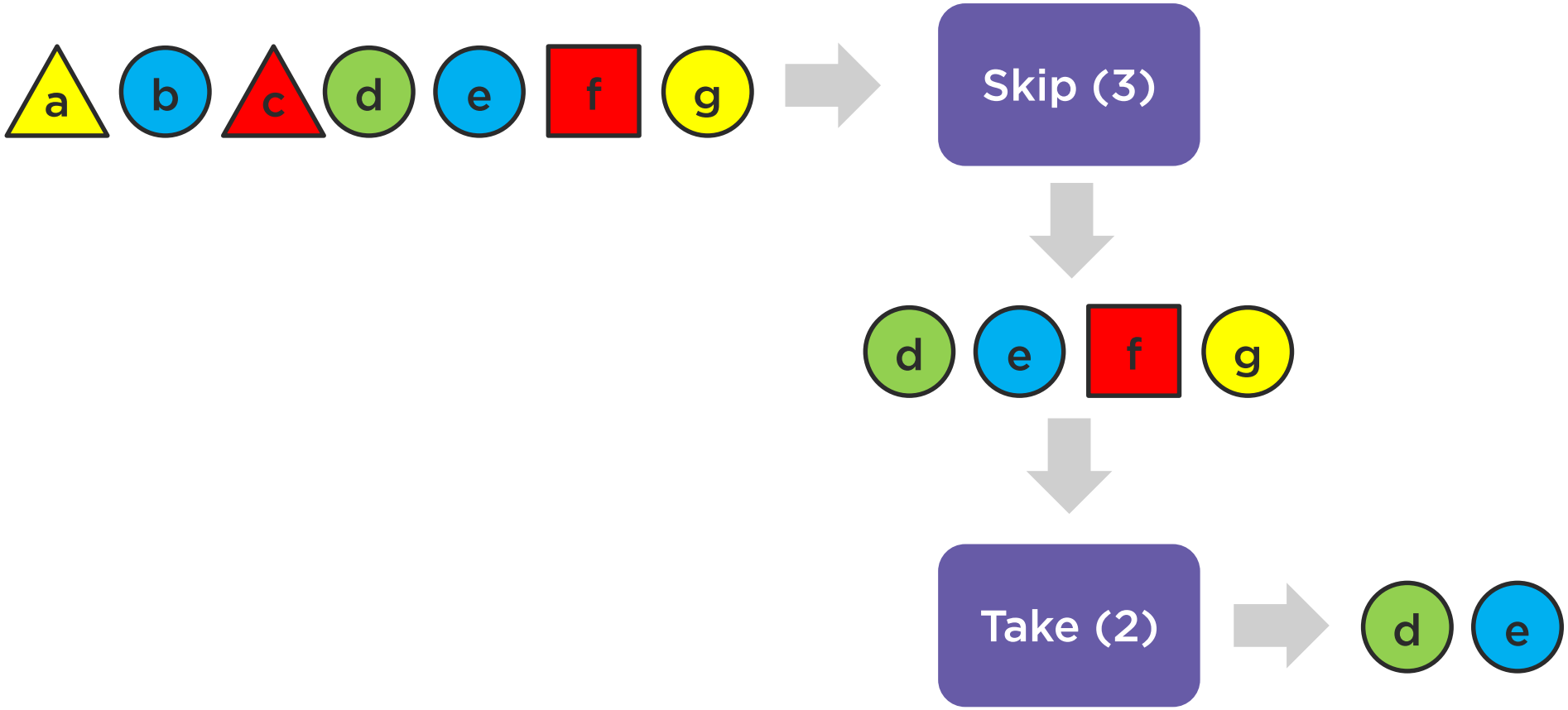
# Reordering Where Nodes



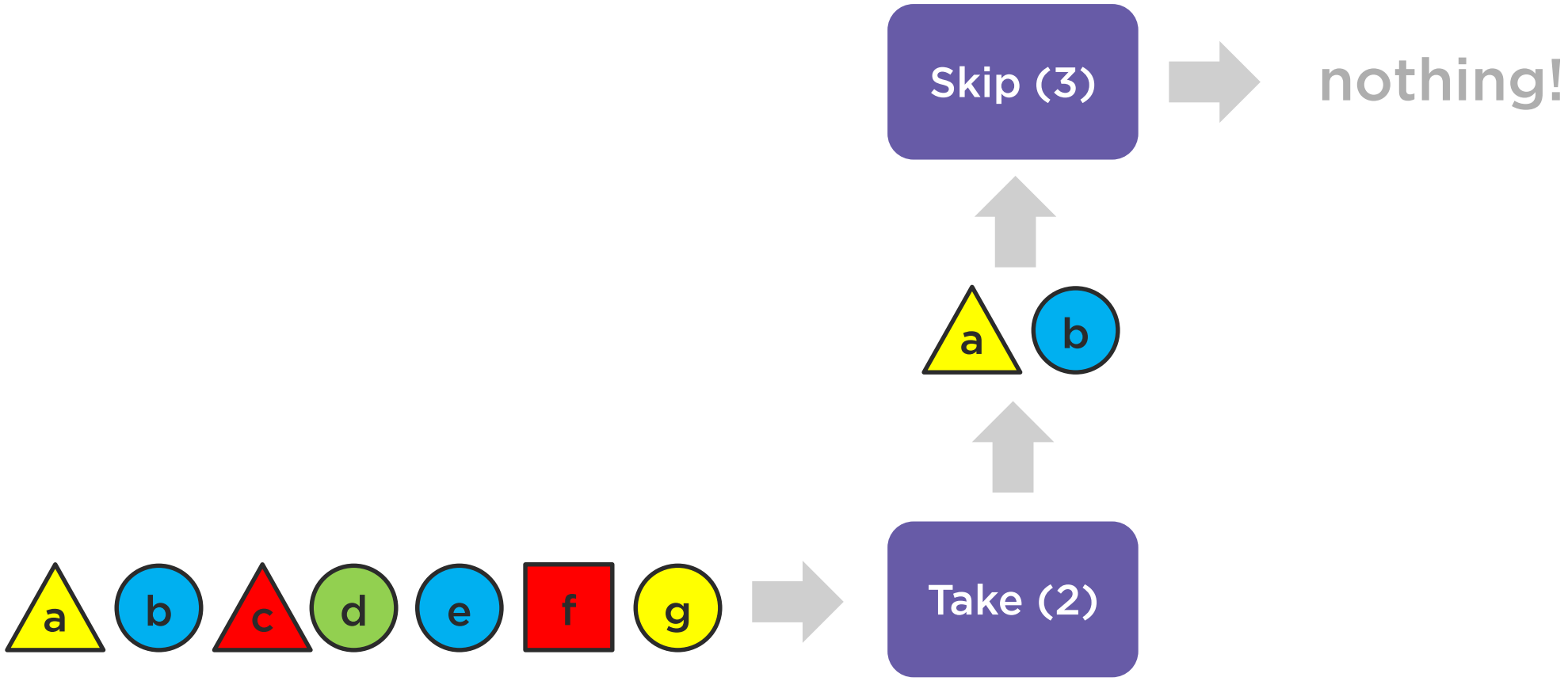
# Reordering Where Nodes



# Reordering Skip and Take



# Reordering Skip and Take

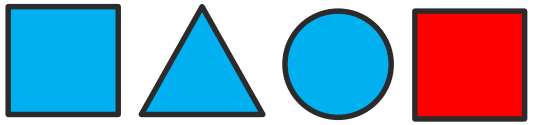


# Reducing Sequences

Count → 7

2 5 8 1

Sum → 16



All = Blue → false

"dog" "cat" "cow"

String.Concat → "dog,cat,cow"

# Many to One Methods

| | | |
|---|---|---|
| First | Count | Min |
| FirstOrDefault | Sum | Max |
| Last | All | Any |
| LastOrDefault | String.Concat | Aggregate |

IEnumerable<T> in, single value out

# Aggregate

General purpose LINQ extension method to apply a function to each element in an IEnumerable<T> to calculate a single value

# Generating Sequences

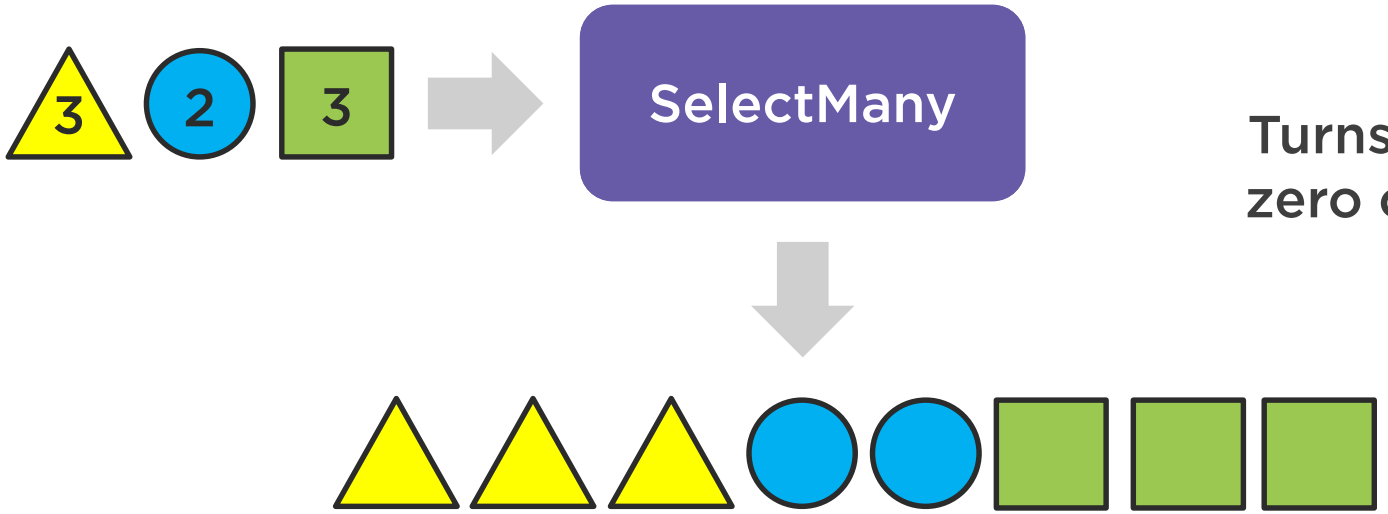

## Enumerable.Repeat

Repeats an item a specified number of times

## Regex.Matches

Returns all the matches on a Regular Expression

# Expanding Sequences




## SelectMany

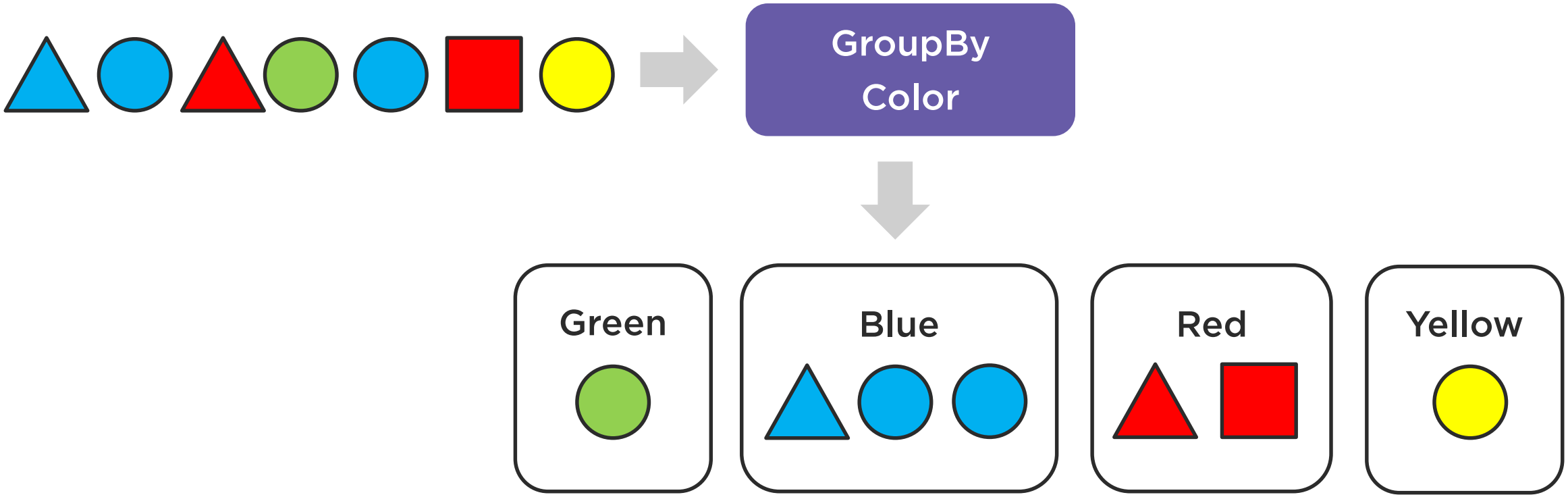
**Turns each input element into zero or more output elements**

# Reordering Sequences

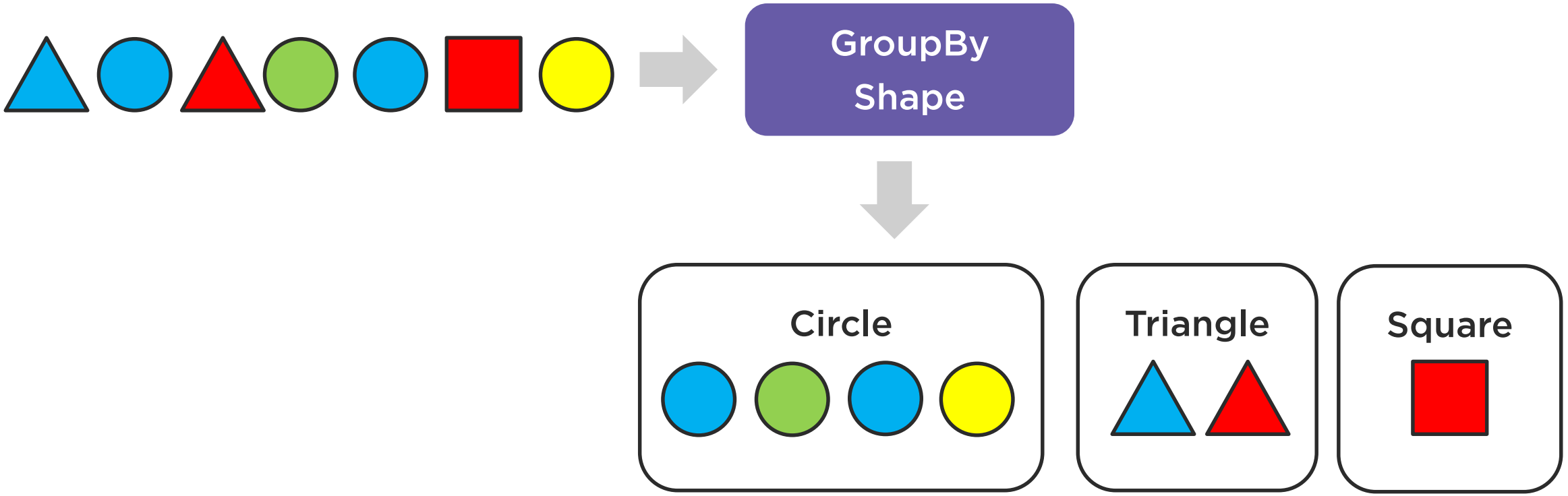

**ThenBy**
Follow OrderBy with a secondary sort order

**Reverse**
Reverses the order of elements

# Grouping Sequences



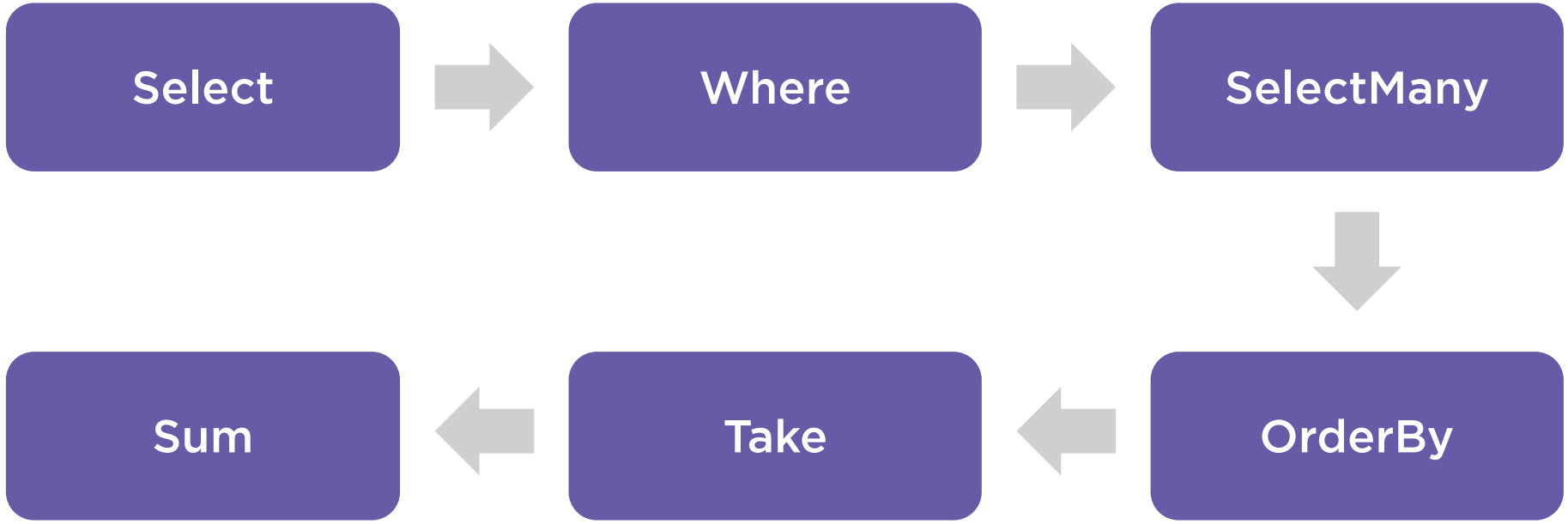
# Grouping Sequences



Reordering methods may require caching the whole sequence in memory

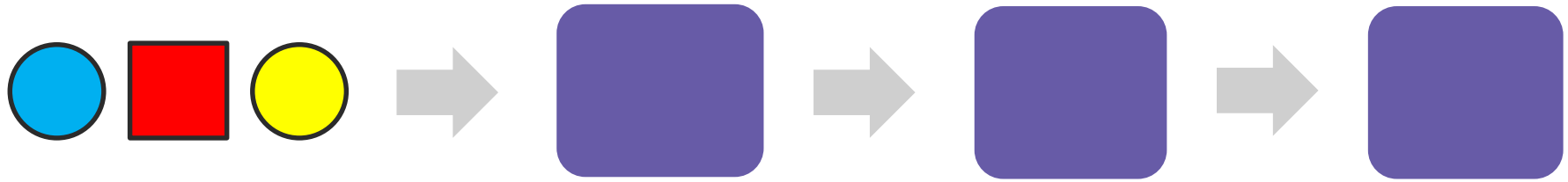
With LINQ to Entities, reordering and grouping is performed by the database

# The Power of Pipelines




Solve complex problems with many simple steps

# Pulling Data Through the Pipeline



Nothing comes through the pipeline until we ask for it


“deferred execution”

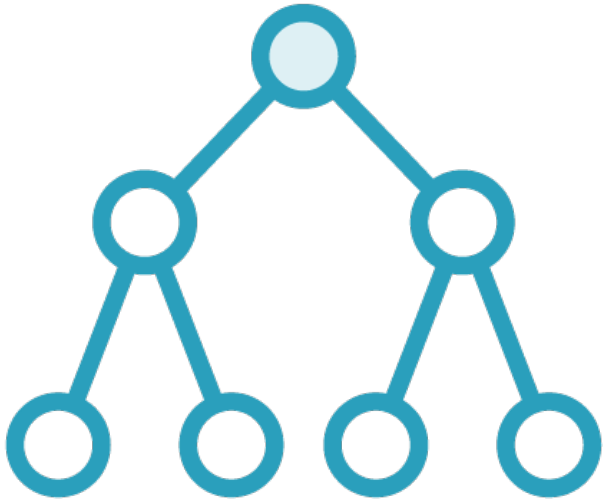
Challenge

# LINQ Challenge

## Album Duration

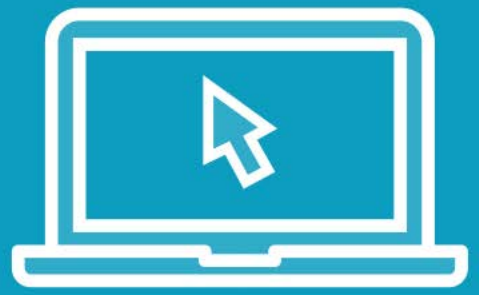
Challenge

# LINQ Challenge

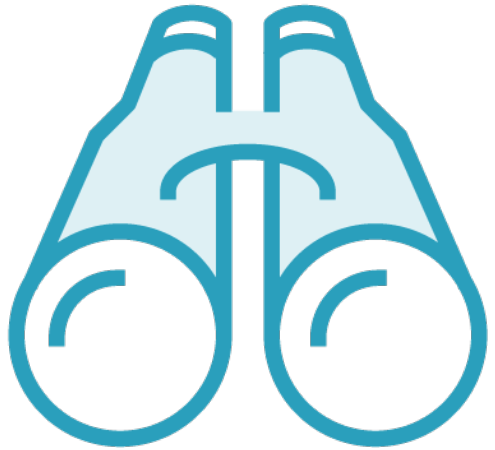


Range Expansion

Demo

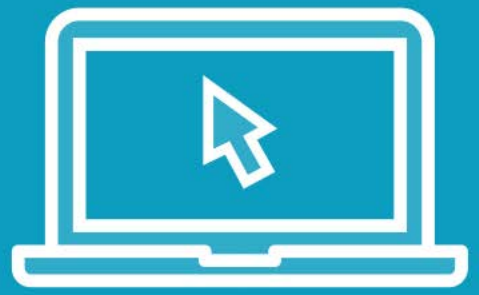


# Real World LINQ



## Find in Files

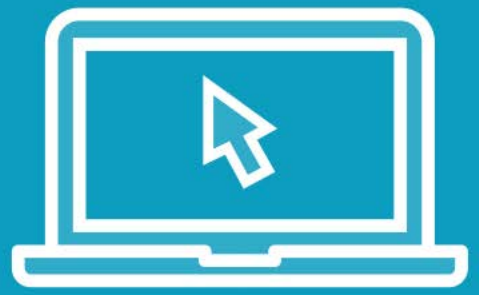
Demo



# Real World LINQ


## Parsing Log Files

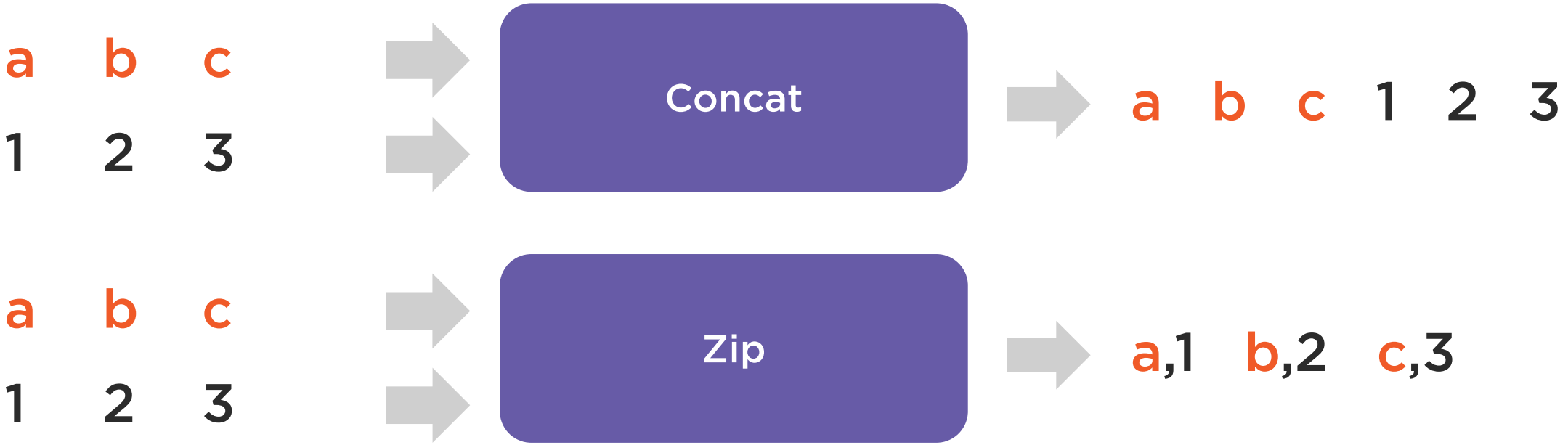
Demo



# Real World LINQ

## Orphaned Project Files

# Combining Sequences




## Except

Returns all elements from the first sequence that aren't in the second

## Intersect

Returns all elements that are present in both sequences

## Union

Returns all distinct elements that are present in either sequence

# Summary

**Pipeline building blocks:**

- From one to many
- Transform elements
- Filter elements
- From many to one
- Combine sequences

**LINQ challenges**

**Real world LINQ**

**Up next: Clean and readable code**