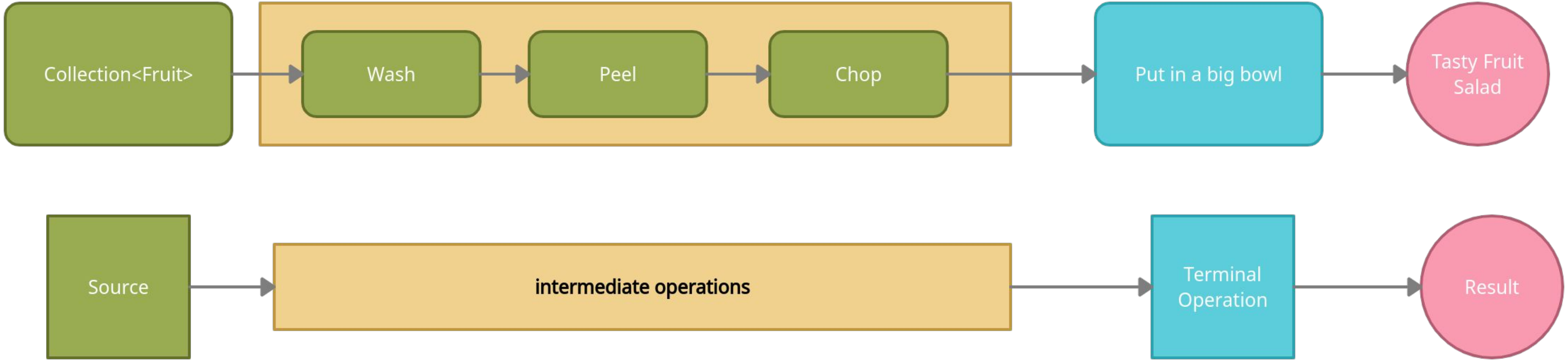
Anatomy of a Stream

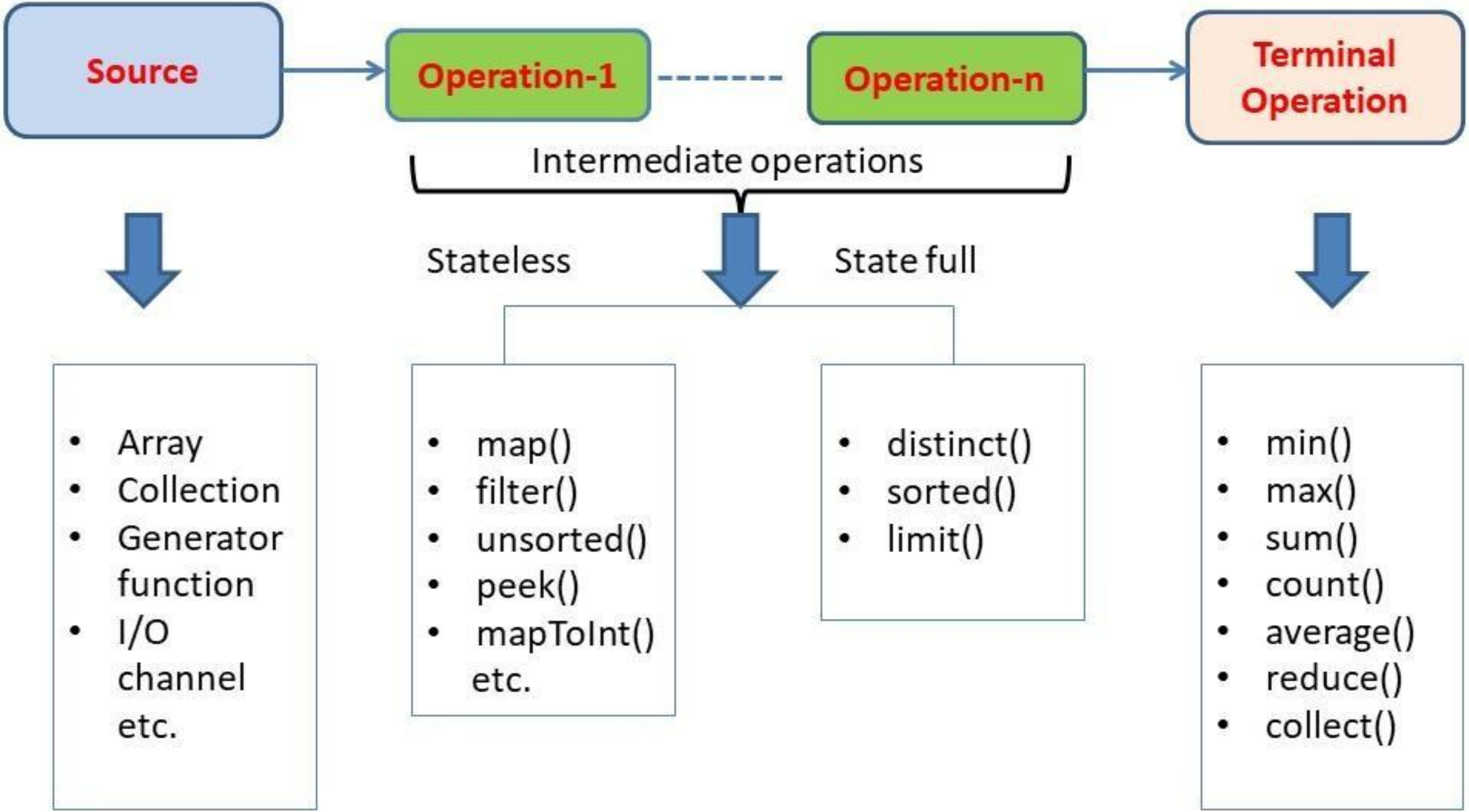
By: Suresh Melvin Sigera

What is a stream?

A collection of elements to be processed are considered as a stream, which is transmitted in a pipeline. These elements can be processed on nodes of the pipeline, such as filter, sort, and aggregate.

Components of a stream pipeline vs tasty fruit salad





- Array
- Collection
- Generator function
- I/O channel

etc.

- map()
- filter()
- unsorted()
- peek()
- mapToInt()

etc.

- distinct()
- sorted()
- limit()

- min()
- max()
- sum()
- count()
- average()
- reduce()
- collect()

Components of a stream pipeline

1. Source

The stream can be created from the array, Set, List, Map or any Collection, any generator function like a random function generator, from a file or any IO channel, some computational pipeline.

2. Intermediate operation

From a source, a stream of elements is generated on which some operations are applied to achieve the desired result. In a stream pipeline, there can be one or more intermediate operations. Intermediate operations can be of two types stateless and stateful. Each intermediate operation consumes a stream and produces another stream as per the implemented logic.

3. Stateless operations

Operation's which doesn't require to maintain the state of the stream and has nothing to do with the other elements of the stream, Operations like map (), filter () mapToInt (), mapToDouble(), peek(), unsorted(), etc.

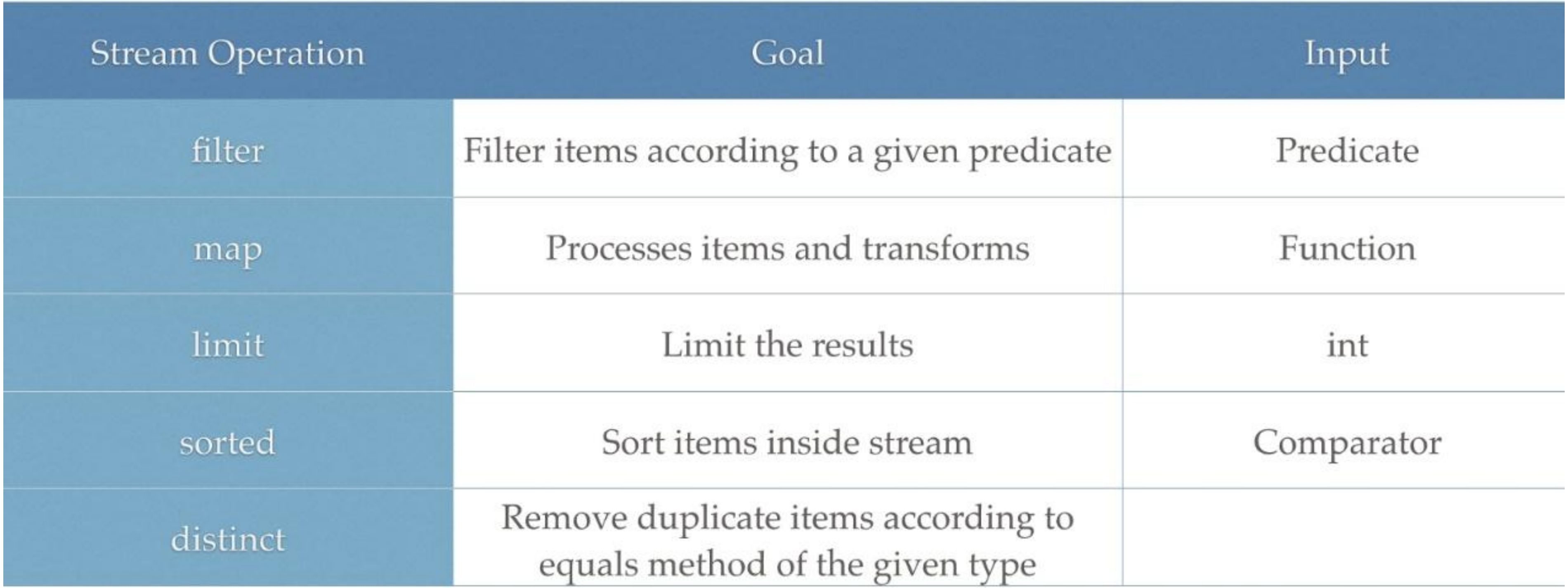
4. Stateful operations

Operations in which elements can't be processed individually and they are required to do some comparison with other elements, like distinct (), sorted (), limit (), etc.

Components of a stream pipeline

5. Terminal Operations

In a stream pipeline there can be only one terminal operation that produces some desired single result or side effect, it consumes a stream but doesn’t produce a stream. Example terminal operations min() , max(), sum(), average(), collect(), for Each(), reduce() etc.

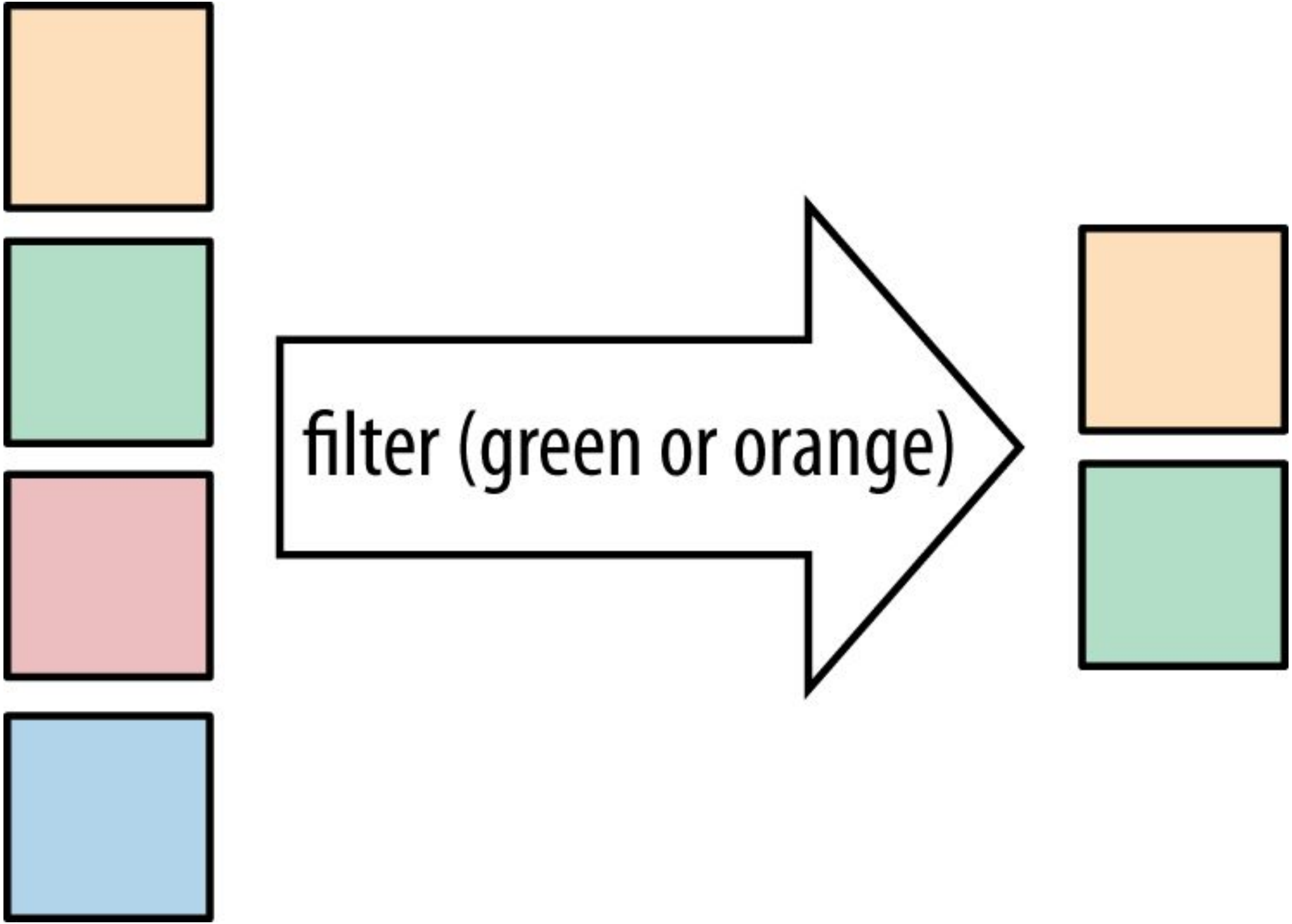


Stream Operation	Goal	Input
filter	Filter items according to a given predicate	Predicate
map	Processes items and transforms	Function
limit	Limit the results	int
sorted	Sort items inside stream	Comparator
distinct	Remove duplicate items according to equals method of the given type	

STREAM OPERATION	GOAL	INPUT
forEach	For every item, outputs something	Consumer
count	Counts current items	
collect	Reduces the stream into a desired collection	

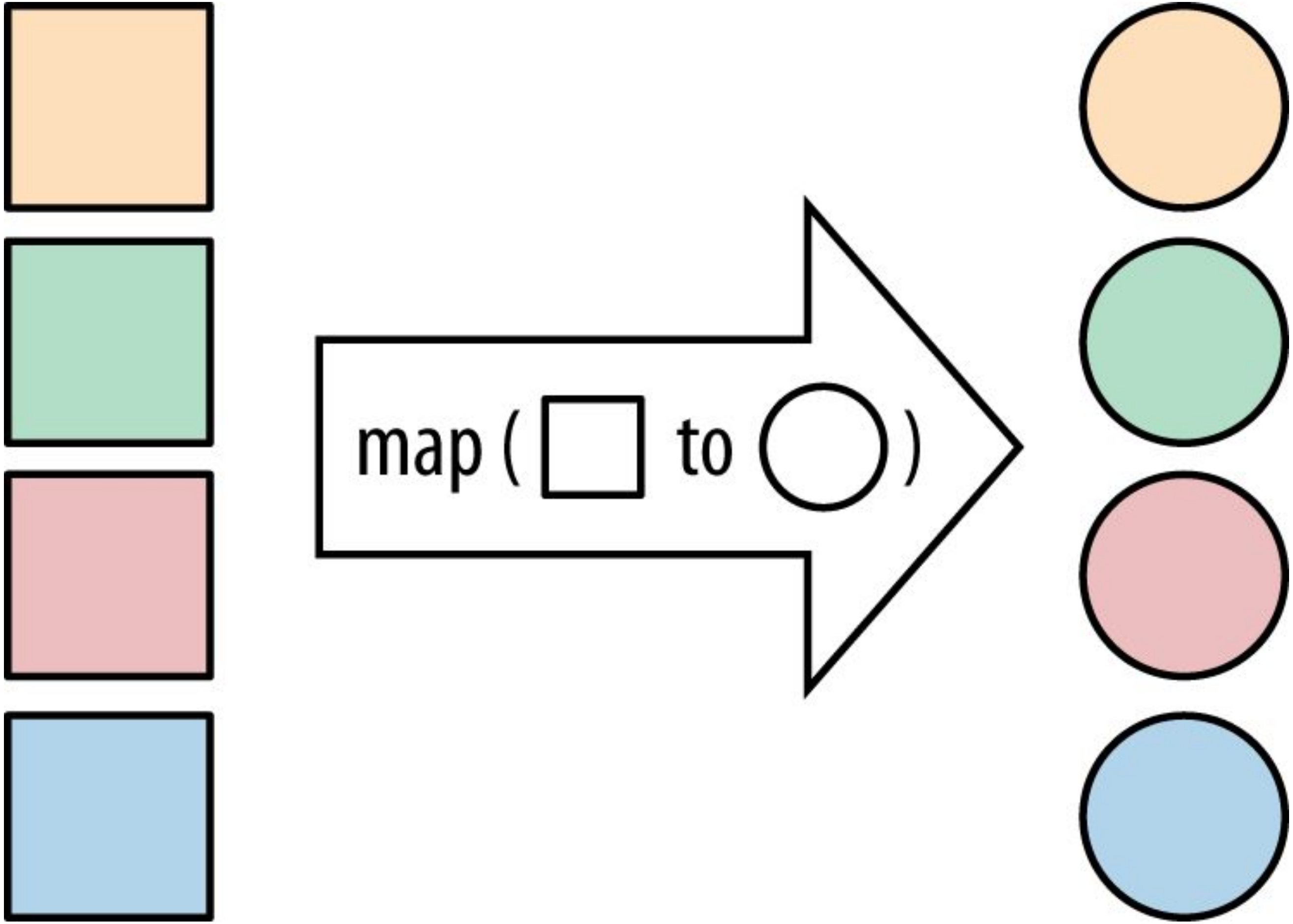
filter

Any time you’re looping over some data and checking each element, you might want to think about using the new filter method on Stream.


filter (green or orange)

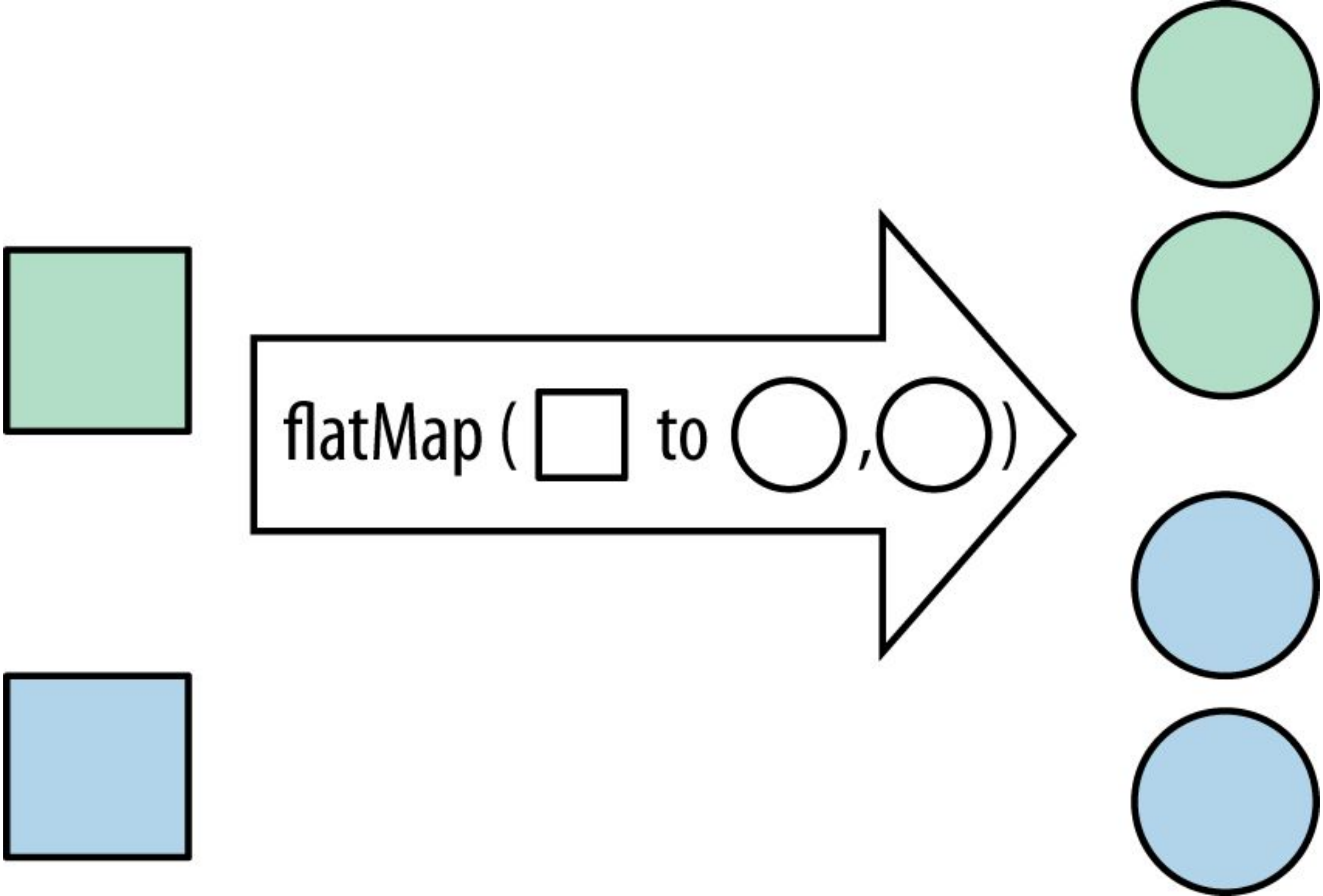
map

If you’ve got a function that converts a value of one type into another, map lets you apply this function to a stream of values, producing another stream of the new values.


map (□ to ○)

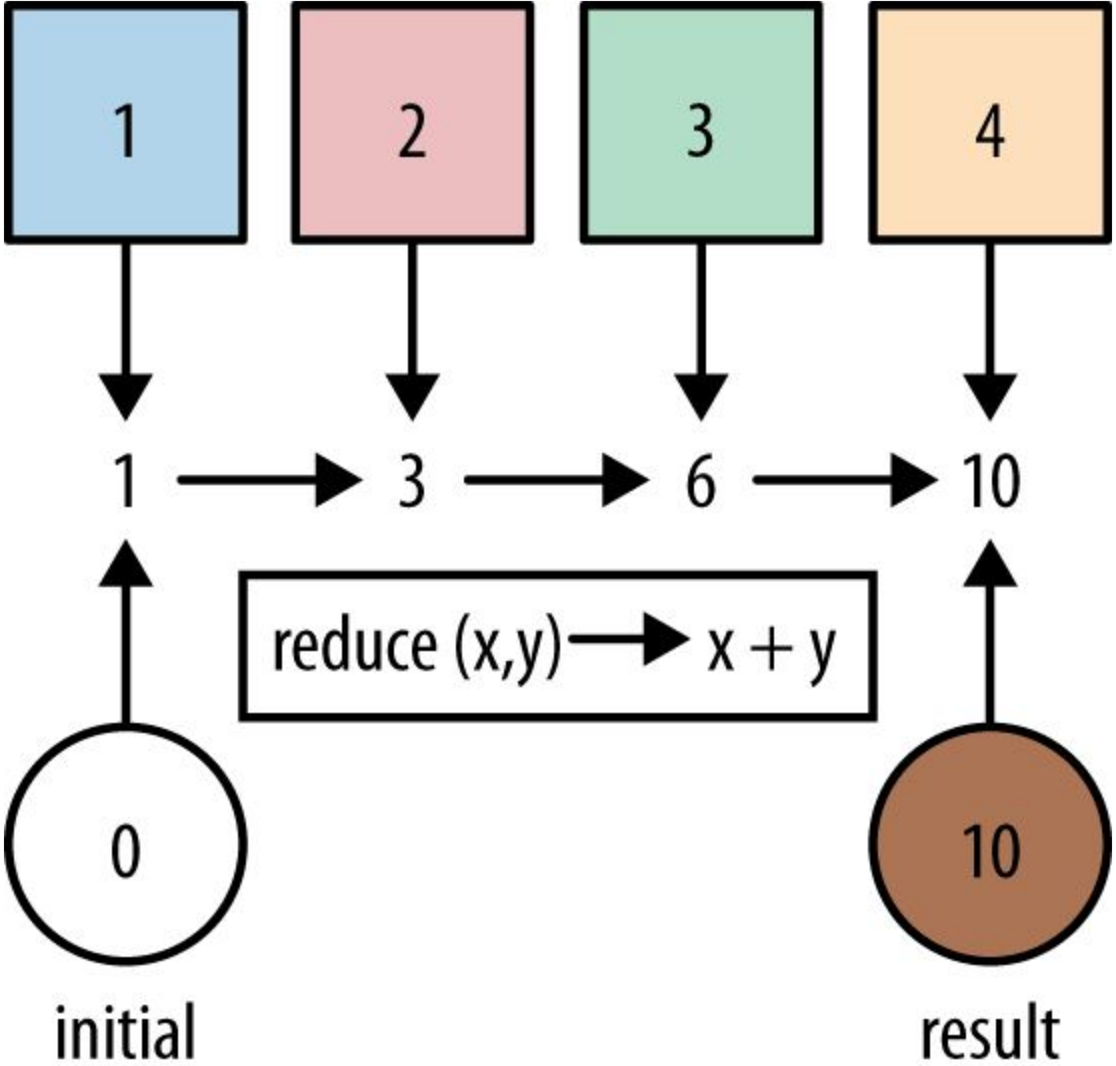
flatmap

lets you replace a value with a Stream and concatenates all the streams together.


flatMap (□ to ○,○)

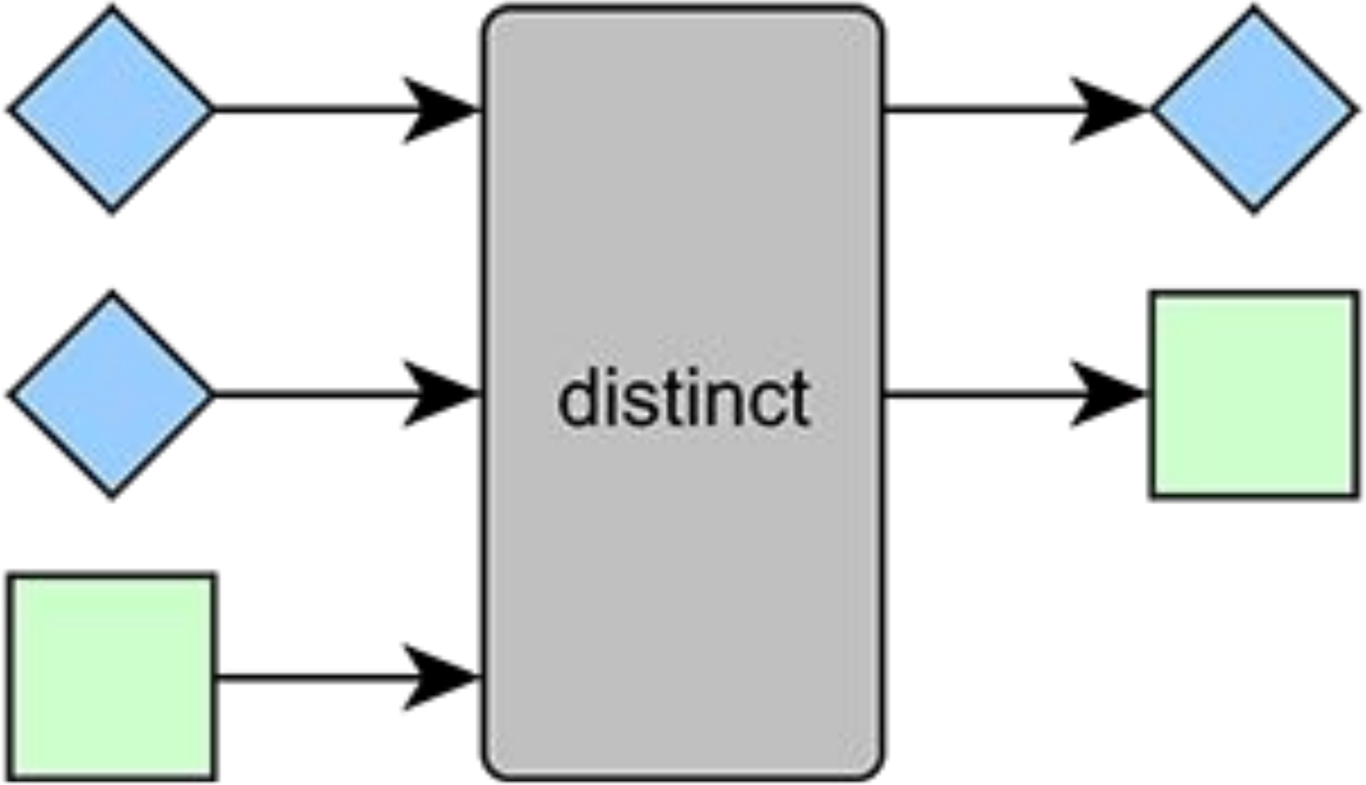
reduce

use the reduce operation when you’ve got a collection of values and you want to generate a single result.


1
2
3
4
1
3
6
10
reduce (x,y) → x + y
0
initial
10
result

distinct

returns a new stream containing only the unique elements in a stream. Used the the default equality comparer defined on the type.


distinct