**Chain:**

Chain is a data structure that allows constant time prepending and appending. This makes it especially efficient when used as a Monoid, e.g. with Validated or Writer. As such it aims to be used where List and Vector incur a performance penalty.

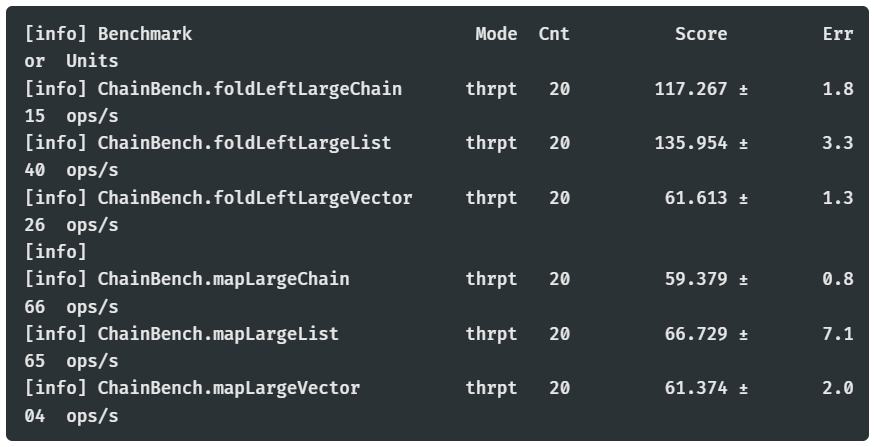
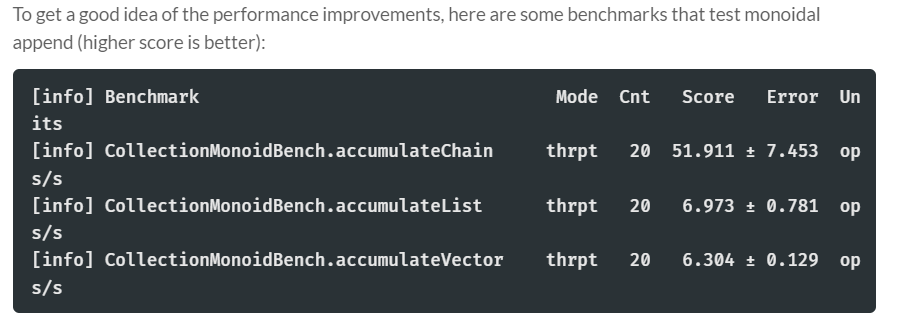
List is simple and good for the most important functions such as fold and map and supports prepending a single element in constant time.

However, List is inefficient traversing through Cat’s data-types (built in types), such as Writer[List[Log], A] or ValidatedNel[Error, A], which will have a runtime of O(n^2).  
This is because, with List, appending a single element requires iterating over the entire data structure and therefore takes linear time.

Chain is also a very simple data structure, but unlike List it supports both constant O(1) time append and prepend. This makes its Monoid instance super performant and a much better fit for usage with Validated,Writer, Ior or Const.

To utilize this Cats includes type aliases like ValidatedNec or IorNec as well as helper functions like groupByNec or Validated.invalidNec.

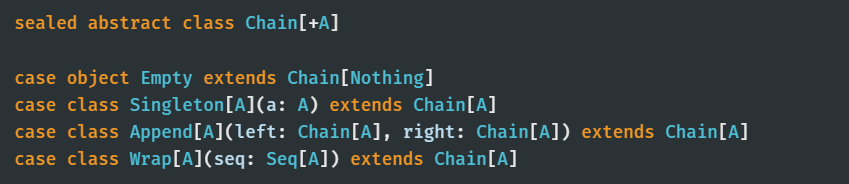
**PERFORMANCE:**



Overview:  
 - Chain is better at accumulating / appending things.  
 - Chain is just as good as List for map & fold  
 - Vector is best for random access performance

**HOW IT WORKS:**

Chain is a simple data structure compared to something like Vector.  
It's a simple ADT that has only 4 cases, it is either:  
 - An empty Chain with no elements  
 - A singleton Chain with exactly one element  
 - A concatenation of two chains  
 - A wrapper for another collection.



The *Append* constructor is what gives fast concatenation ability at constant time O(1).

To append or prepend a single element, it wraps the element with the Singleton constructor and then use the Append constructor to append or prepend the Singleton Chain.

The Wrap constructor lifts any Seq into a Chain. This can be useful for concatenating already created collections that don't have great concatenation performance.

Append(Wrap(list1), Wrap(list2)).foldMap(f) will be much faster than just concatenating list1 and list2 and folding the result.