Step 2: Runtime Analysis

extraLArgeArray - 100k
insert: 957.96635 ms
append: 3.0477 ms

largeArray - 10k
insert: 9.6436 ms
append: 0.7028 ms

mediumArray - 1k
insert: 193.7 µs
append: 156.3 µs

smallArray - 100
insert: 55.1 µs
append: 116.1 µs

tinyArray - 10
insert: 38.1 µs
append: 92.6 µs

Based on the results of testing different size arrays into the 2 functions (doublerAppend and doublerInsert), doublerInsert is the more time efficient function up until the array length exceeds an array of a little over 1k in length. Then, doublerAppend becomes the more time efficient function for the larger array sizes. Therefore, the doublerAppend scales better for larger array sizes.

These results make sense because the unshift array method in the doublerInsert function gives the function an overall time complexity slightly slower than the push array method in the doubler Append function. This is because each time unshift is called, every location in the array needs to be moved, taking time. Whereas with the push array method, the function can dist add another index at the end of the array, not affecting any of their other indexes.