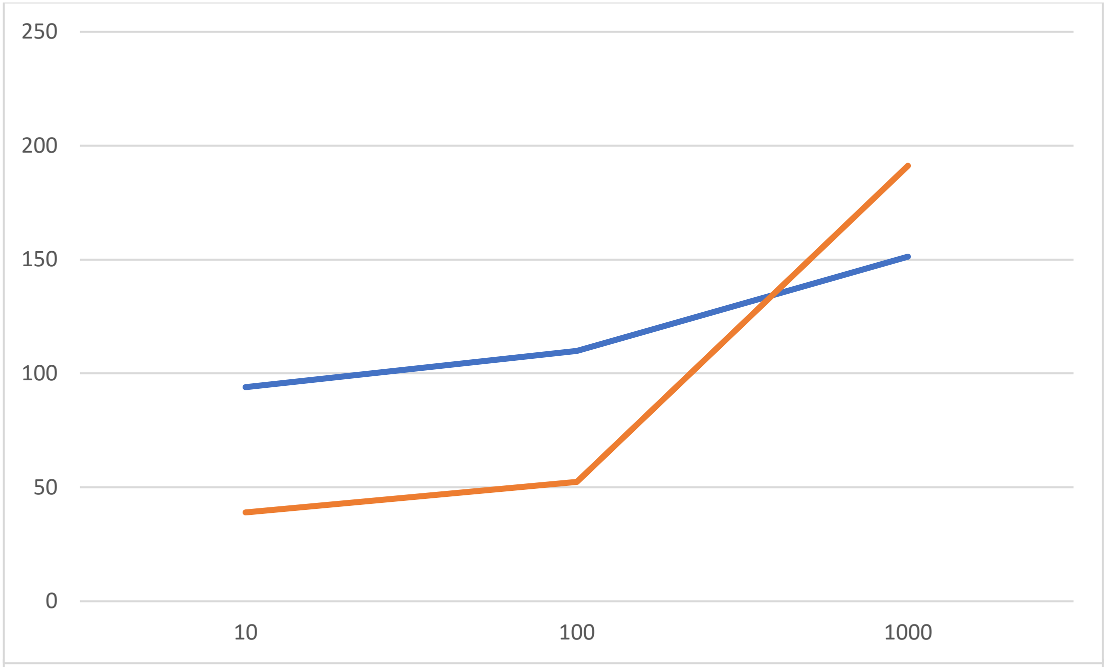
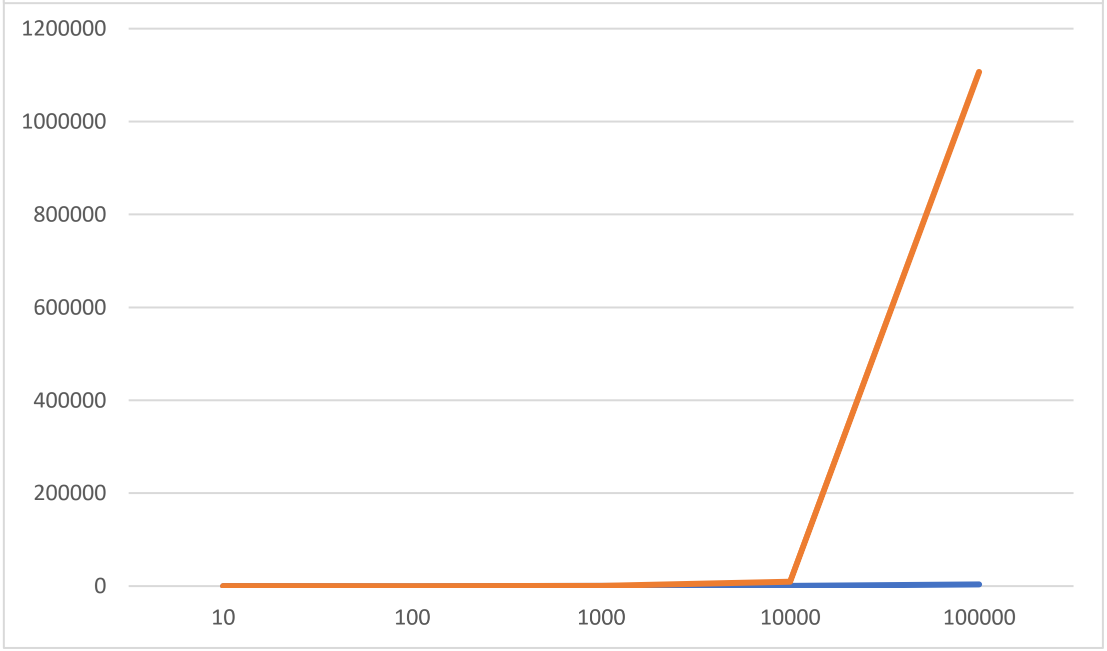
Array	Size of Array	doublerAppend	doublerInsert
tinyArray	10	94	39
smallArray	100	109.9	52.4
mediumArray	1000	151.3	191.2
largeArray	10000	561.2	9659
extraLargeArray	100000	3612.7	1106788.4

After logging all the results, I've converted the values into the same units (microseconds) so that they could be graphed meaningfully. While it is true that the doublerInsert performs better with smaller arrays, it does not scale as well as the data sets get larger. With the medium array of 1000, the doublerAppend function is more performative. This performative difference becomes a factor of 20:1 with the largeArray and a factor of a little over 300:1 with the extraLargeArray. As a guess, doublerAppend is O(n) and doublerInsert is O(n^2).





When doing the research for the primary cause for the difference in performance, the simplest explanation for the difference in performance is that the array.push function requires fewer manipulations of the memory structure "near the metal". Essentially, when you create an array, a certain amount of memory is allocated for it. When JS senses you will need more space for it, it allocates a larger section of memory and copies the original data to the new location. This does not occur for every operation, so array.push is fairly performative. Array.unshift is a different story altogether. Inserting the value at the front of the array requires the underlaying memory structure to be manipulated and copied for EVERY insertion. While this does not make a large impact at smaller scales, it adds up over time!