Speeds

	tinyArray	***smallArray***	***mediumArray***	***largeArray***	***extraLargeArray***
Insert	17.167 µs	24.375 µs	114.916 µs	7.01875 ms	693.315542 ms
Append	56.083 µs	58.417 µs	71.541 µs	505.75 µs	1.936333 ms

Runtime Analysis

Read over the results, and write a paragraph that explains the pattern you see. How does each function "scale"? Which of the two functions scales better? How can you tell?

Based on the data it seems the insert function is increasing in an exponential manner. As the array is getting bigger the speed is decreasing, thus becoming less efficient. To break it down further, the insert function has a for loop which would be n, it also working an unshift process which is another n since it is having to move each element. This results in n*n so possible notation of O(n to the 2nd). In regards to the append function, this function seems to maintain speed closely to the size of array which may mean it is growing in a linear form. When breaking it down the for loop is n and since shift is not really making change that is a constant 1. This results in n*1 which is O(n). I believe the append function would be the most efficient of the two do to the processes involved in each as broken down earlier.

(Extra credit part1…..part2 in writecode.js)
Why the slower function from 'runtime.is' is slower?

Adding to why I believe the 'Insert' function is the slower one is because as the array increases there is more effort to process the data. The insert function includes .unshift which would require moving every single element from its current position to a new one while the append does not need to do this. This is why I believe the Insert function is slower.