Array size	Time for doublerInsert	Time for doublerAppend
Extra large	1.3097987 seconds	4.3861 milliseconds
Large	11.1008 milliseconds	471.4 microseconds
Medium	166.4 microseconds	50.2 microseconds
Small	14.4 microseconds	10.4 microseconds
Tiny	7.4 microseconds	5.2 microseconds

Based on the result above, it is clear that the insert function takes a lot more time than the append function, especially for larger arrays. The time complexity of the insert function is O(n^2), and the time complexity of the append function is O(n). That is because the insert function is looping over the first array, and using the unshift method, adding value to a second array. With each iteration of the loop, the code is running a nested loop on the second array, and with that nested loop, it is shifting every item of the second array one spot to the right, and adding the new item of the first array to the first spot of the second array. So as the second array gets bigger, the nested loop also gets bigger, and it does so exponentially. The append function does not do a nested loop, it is simply running a single loop on the first array, and adding the items of the first array to the last spot of the second array using the push method. The space complexity of both the insert and the append function is O(n), because for both functions the code takes an existing array, allocates the memory space for a brand new array.

For the above reason, we can see that the append function, going from tiny to extra large array size, the time for execution goes up by roughly a factor of a thousand. But for the insert function, it goes up roughly by a factor of a million. Also for tiny array, we can see the time for the two functions is almost the same, although insert time is still higher than append time. But for extra large array, the insert time is almost thousand times higher than append time.