Functions	extraLargeArray	largeArray	mediumArray	smallArray	tinyArray
doublerInsert	912.7998 ms	22.4266 ms	318.2 µs	92.8 µs	41.3 µs
doublerAppend	3.9416 ms	1.2614 ms	297.7 µs	101.6 µs	85.5 µs

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?

Ans: Insert is using unshift and append is using push method. Unshift method puts the numbers in the beginning of the array, while push adds it to the end. For some of them when I ran it multiples I got different runtimes. For example, for largeArray I also got 18.5923 ms runtime for Insert. The scale for the doublerInsert function is faster when the array is smaller. The runtime for tinyArray and smallArray is a few microseconds faster than doublerAppend function. However, the runtime increases by a lot as the array gets bigger. At the extraLargeArray for doublerInsert function the runtime is 912 milliseconds. The scale for the doublerAppend function is similar no matter the size of the array. It increases when the array becomes larger but it's still closer to the other arrays. The extraLargeArray runtime for doublerAppend function is only 3.9 milliseconds. The doublerAppend function scales better because no matter how large the array becomes it takes less time than using unshift.

For extra credit, do some review / research on why the slower function is so slow, and summarize the reasoning for this.

Ans: The time complexity of push is O(1), while unshift is O(n). Push only appends elements to the end of the array, with O(1) it stays at a constant runtime. However, unshift needs to put the elements in the beginning of the array, which means it needs to increment the elements. It takes n number of elements which increases its runtime based on the n. This is why the doublerInsert function which uses unshift is slower.