Module 7 Project

Step 1:

1. extraLargeArray
    - doublerAppend – 3.28 milliseconds
    - doubleInsert – 870.49 milliseconds
2. largeArray
    - doublerAppend – 650.01 microseconds
    - doubleInsert – 6.51 milliseconds
3. mediumArray
    - doublerAppend –159.41 microseconds
    - doubleInsert – 171.39 microseconds
4. smallArray
    - doublerAppend –86.38 microseconds
    - doubleInsert – 42.37 microseconds
5. tinyArray
    - doublerAppend –96.24 microseconds
    - doubleInsert –38.97 microseconds

Paragraph explanation:

When looking at the run times of both functions, as the size of the argument increases, the doubler Insert function becomes less efficient/slower and the doubler Append function maintains efficiency/speed. This is because doubler append uses the built in array method of push, which simply adds a new number to the end of an array without adjusting any other numbers in the array, while doubler Insert utilizes the built in array method of unshift, which moves every number in the array by one position placing the new number at the front of the array. With relatively few numbers in the tiny, small, and medium arrays, both functions run at similar speeds because doublerInsert isn't shifting hundreds of thousands of numbers. However, when the array size increases to a five or six figure number, the runtime length of doublerInsert sees exponential or quadratic growth because it is now having to run hundreds of thousands and even millions of operations more than the simpler doublerAppend function (which shows linear growth). All in all, doublerAppend scales better and would be the optimal function to use for one's code, especially if argument's lengths will be millions of numbers long.