2. Running node runtime.js:
Results for the extraLargeArray
insert 665.6146 ms
append 3.0173 ms

3.

Array size	insert time	append time
extraLargeArray	639.1718 ms	2.0743 ms
largeArray	6.4702 ms	115 µs
mediumArray	117.6 µs	15.5 µs
smallArray	6.5 µs	3.3 µs
tinyArray	18.4 µs	3.2 µs

4. Double append seems to scale roughly in line with the size of the array, though once the size of the array goes under 100 indexes, that runtime seems to not decrease as much anymore. For the double insert function, it scales significantly faster as the size of the array increases, and seems to struggle if the array size is too small, as indicated by tinyArray taking longer than smallArray, even though tinyArray is one-tenth the size of smallArray.

By taking a look at the run times of each array size, the append function scales far better, as its runtime remains largely faster than the insert function. This is due to that for the extraLargeArraym append had about 2 ms, while insert takes 600 ms minimum, and then for the other array sizes, append's runtime was always lower than inserts, with it only getting to be close at smallArray.

5. The insert function is so slow in comparison to the append function due to the array method it uses, .unshift() at the base level seems like a reverse push(), where it adds a new element to the front of the array. However, unlike push, unshift first has to go through the array, and create a new array to be reassigned to the variable, with the element being entered through unshift to be at index 0, and the rest of the original array is moved 1 index over, which makes it take increasingly to perform as the array gets bigger in size meaning more elements have to be moved 1 index to the right each time unshift is called.