Week 7 Assessment

Runtime Analysis

- In your notes document, take note of the timing result for the **extraLargeArray** results– comparing when the **extraLargeArray** is passed to **doublerAppend** and **doublerInsert**.

Results for the extraLargeArray
insert 889.2089 ms (passed onto the doublerInsert)
append 4.8885 ms (passed onto the doublerAppend)

- Next, edit the code in runtime.js to obtain timing results for calling the two functions with all of the differently sized arrays– tinyArray, smallArray, mediumArray, largeArray, and extraLargeArray. Notate these in your document in some kind table table so that you can easily compare the different values for the timers in relation to the size of the array that was passed into each function.

Results for the tinyArray
insert 68.4 µs
append 125.3 µs

Results for the smallArray
insert 42.6 µs
append 98.7 µs

Results for the mediumArray
insert 182.4 µs
append 157.3 µs

Results for the largeArray
insert 7.7049 ms
append 461.7 µs

Array:	insert	append
tinyArray	68.4 µs	125.3 µs
smallArray	42.6 µs	98.7 µs
mediumArray	182.4 µs	157.3 µs
largeArray	7.7049 ms	461.7 µs
extraLargeArray	889.2089 ms	4.8885 ms

- 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?

Each array is getting larger as you progress from tiny to small, to medium, etc. As the arrays increase, so does their runtime. The doublerAppend function seems to scale better because it does not exponentially increase as much as the doublerInsert function does. This is because it has a faster runtime, due to the different method being invoked. The doublerAppend function uses the push method, whereas the doublerInsert function uses a method called unshift, which has a slower runtime.

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

The doublerInsert function is slower because it uses the unshift method. The push method is used in the doublerAppend function, and the push method has a runtime of O(1). The doublerInsert function uses the unshift method, which has a runtime of O(n). The higher complexity of unshift is because while a push just adds onto the end of an array (meaning it does not affect the other indexes of said array), the unshift has to increment all elements of the array, so all indexes in the array are modified.