Runtime Analysis on runtime.js

Timing results for extraLargeArray:

Function	**Runtime**
doublerAppend	12.303852 ms
doublerInsert	2.5145464840000002 s

Timing results for calling the doublerAppend and doublerInsert functions with all of the differently sized arrays:

Array	**doublerAppend Runtime**	**doublerInsert Runtime**
tinyArray (10)	153.815 µs	90.693 µs
smallArray (100)	328.109 µs	85.425 µs
mediumArray (1000)	235.986 µs	529.464 µs
largeArray (10000)	1.61014 ms	21.908076 ms
extraLargeArray (100000)	30.455126 ms	3.846845328 s

Patterns explained:

I am seeing an increase in total runtime as the array size gets larger. Each function scales very incrementally and only slightly increases in time. I can't see any true scale pattern in the runtime. The doublerAppend function has a better scaletime overall.

Extra Credit:

The doublerAppend function uses the .push() method adding the numbers to the end of an array while the doublerInsert method uses the .unshift() method which shifts the newly added numbers to the front of the array. Given the that .unshift has to shift numbers before they are added to an array increase its total runtime. Because of this, .push has a constant runtime while .unshift has a linear runtime. Because .push is constant is is faster