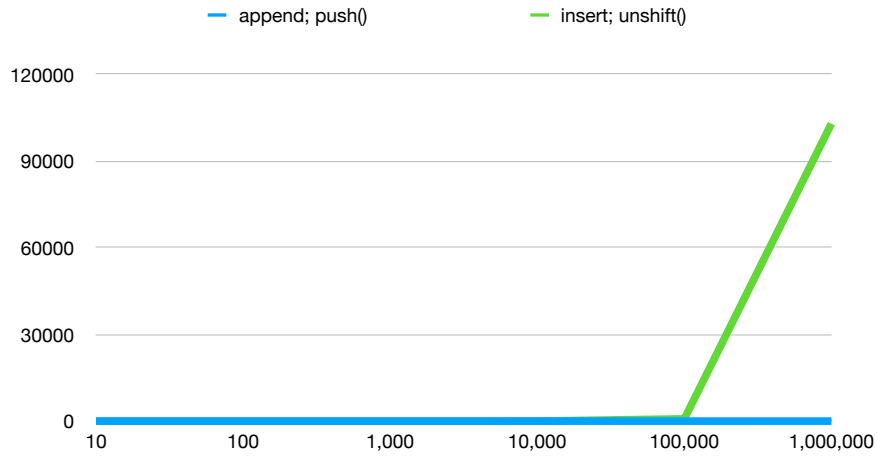
Testing Time to execute array functions: Times in ms.

Array Length	append; push()		insert; unshift()	
10	0.06375		0.018	
100	0.096375	151%	0.027	147%
1,000	0.132917	138%	0.145	537%
10,000	0.439	330%	6.776	4,662%
100,000	2.220791	506%	773.625	11,418%
1,000,000	9.230167	416%	103,000.000	13,314%



The push() function takes significantly less time, especially with large arrays than does unshift(). The relationship between the push() function and the length of the array is less than linear. When increasing the array size from 1,000 to 10,000 the length of time increased slightly over three-fold, from .13ms to .43ms. Similarly, when increasing the array size ten-fold, from 100,000 to 1,000,000, the length of time only increased approximately four-fold, from 2.2 ms to 9.2ms. And finally, the increase from an array of 100 to an array 1,000,000 (a ten-thousand—fold increase), the time required only increased from .096ms to 9.23ms (a ninety-five—fold increase). It is clear that this function is highly efficient in time.

The unshift() function is another story. At each step, the time required increases exponentially. On the upper end, the ten-fold increase from 100,000 to 1,000,000 resulted in a four-hundred—fold increase in the time required, from 773ms to 103 seconds. And finally, the increase from an array of 100 to an array 1,000,000 (a ten-thousand—fold increase), the time required only increased from .027ms to 103s (nearly a 4-million—fold increase). It is clear that this function is not efficient in time - particularly in large arrays.

The difference seems to be rooted in how arrays work overall in javascript. The first function, push(), adds an element to the end of an array. To do so, the length of the array is simply increased by 1, and the element is placed in the final position. The second function, adds an element to the beginning of an array. This is accomplished by increasing the size of the array by 1, and then shifting the position of the every single element in the array. The new element is then inserted in position 0.

Online research shows that the push() function should have a time complexity of O(1). This means that the length of time it takes to perform should be the same regardless of the length of the array. The unshift() function should have a time complexity of O(n). This means that the length of time it takes to perform the operation should increase linearly. A ten-fold increase in the length of the array should result in a ten-fold increase in the length of time it takes to perform the action.

It is interesting that my tests did not have these same results. It is also interesting that with very small arrays of less than 1,000 elements, the push() function consistently performed better in time than the unshift() function - the opposite of larger arrays.

It is clear overall, that the unshift() function has a far greater time-complexity than the push() function. This becomes particularly apparent when working with larger arrays.