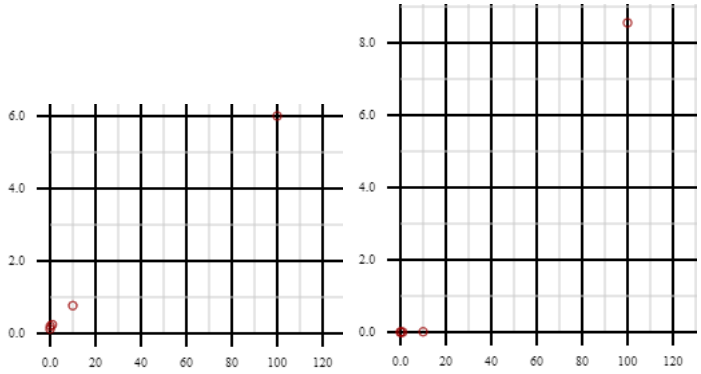
Runtime Analysis

	XL	L	M	S	XS
APPEND(.PUSH())	6.02 ms	767.7 us	242.1us	201.7 us	120 us
INSERT(.UNSHIFT())	855.39 ms	9.02 ms	206.1us	108.9 us	41.6 us



These data points make it a little hard to see the curve, but when plotted out on a graph it is pretty clear that the Append function is following a linear curve while the Insert function is following a quadratic curve. For smaller inputs it almost seems like Insert takes less time, but as the input increases in size, the time taken increases rapidly over the time taken by the Append function. At an input size of 100,000 the Insert function takes more than 100 times as long to run. I tried 1,000,000 and got tired of waiting for it to finish. On the other hand, the Append function is always scale proportionally to the size of the input and can handle much larger input sizes. With an input of 1,000,000 the append child array only takes 29 ms to complete.

This all comes down to the difference between .Push() and .Unshift(). .push() simply appends the value onto the end of an array and creates a new index for it. It will always only be these two operations when called giving it a constant runtime or O(1). On the other hand, unshift() appends the value onto the beginning of the array and gives it an index of 0. This requires that the index of every value in the array be changed every time a value is inserted. This leads to a proportional runtime or O(n). Both functions also have to loop over the length of the array which takes a proportional runtime O(n). When combined together the .push() = O(n) * O(1) = O(n) while the .unshift() = O(n) * O(n) = O(n^2).