Runtime Analysis

tinyArray	Results for the tinyArray	
	insert:	22.292 µs
	append:	57.583 µs
smallArray	Results for the smallArray	
	insert:	30.5 µs
	append:	67.416 µs
mediumArray	Results for the mediumArray	
	insert:	188.542 µs
	append:	134.792 µs
largeArray	Results for the largeArray	
	insert:	8.98425 ms
	append:	588.584 µs
extraLargeArray	Results for the extraLargeArray	
	insert:	768.021042 ms
	append:	4.202 ms

After reviewing the results, it appears that as the array size got larger the time it took for the append and insert function took longer to scale as well. Although the functions took longer to scale as the size got bigger for scalability purposes, it appears that the append function performed at a better rate. For smaller arrays (10 - 100), the insert function performed at a faster rate and would be the choice to use in those settings. For large-scale arrays (1000 - 100,000), the append function performed at a faster rate than its counterpart. For scalability purposes, it would make sense to utilize that function moving forward.

Extra Credit:
After doing some research, it appears that one of the reason's why the unshift/insert function is slower with larger arrays is because the items in the array need to shift over in the process. whereas the append/push function only adds an item to the end (less work).