Step 1: Runtime Analysis

Array Size	doublerAppend Time (ms)	doublerInsert Time (ms)
tinyArray	49.42 µs	18.25 µs
smallArray	73.63 µs	30.83 µs
mediumArray	118.96 µs	168.92 µs
largeArray	553.38 µs	7.84 ms
extraLargeArray	1.81 ms	701.89 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?

For smaller arrays such as tinyArray and smallArray, doublerInsert consistently outperforms doublerAppend. This suggests that for relatively small arrays, the overhead of shifting elements in doublerInsert is less significant than the overhead of array resizing in doublerAppend. However, as the array size increases, doublerAppend becomes more efficient compared to doublerInsert. This is evident from the increasing performance gap between the two functions for larger arrays, with doublerAppend consistently outperforming doublerInsert. The drastic increase in execution time for doublerInsert with large and extra-large arrays indicates that the function does not scale well with array size. On the other hand, doublerAppend exhibits more consistent performance across different array sizes, indicating better scalability. Therefore, doublerAppend scales better than doublerInsert, as it maintains relatively stable performance even as the array size increases.index