ABSTRACT
The first experiment is composed of creating two arrays and adding their contents together while timing how long it takes to perform such an operation. Two test files were created, one (test.html) for 1D arrays, or standard arrays, and another (test2.html) for 2D arrays, or matrices as they are implemented in Javascript. Arrays in both files are the same size. Both files are HTML files however they both contain inline Javascript and it is the operations in that Javascript which will be under inspection. The html present in the document is just for displaying results.

EXPERIMENT

test.html	test2.html
Array Dimensions: 1D	Array Dimensions: 2D
Array Size: 29,997,529	Array Size: 5,477x5,477 (Total Size: 29,997,529)
Operation: Addition	Operation: Addition

At the start of each file, two arrays are initialized with starting values of 5 and 2. For test.html both of those arrays are 1D and for test2.html the arrays are 2D. Then, before the script enters a while loop, it takes a snapshot of the current time in milliseconds using performance.now. After that first time snapshot, both arrays will perform an addition operation between each other within a while loop and the result of said operation is stored in another array of the same size. After the operation is performed, the script breaks out of the while loop when an arbitrary variable becomes >= resultArray.length. Immediately after it leaves the while loop, another snapshot is taken using performance.now. This entire operation is performed 100 times so that 100 data points are generated. The deltas between all those snapshots are computed, summed up, and divided by 100 to get the average elapsed time to perform an addition operation between two arrays. The elapsed time for performing the addition operation between 2 elements is calculated by calculating the deltas between all the snapshots, sum them up, divide by 100, and dividing the result by the array's total size.

RESULTS

The results for this experiment were surprising. The initial purpose of creating two test files, one for arrays and one for 2D arrays, was to see if an array's dimension affects the speed at which operations can be performed on it. The initial hypothesis was that performing operations on 2D arrays would take longer than for 1D arrays. However, the results of the experiment show the opposite, that operations on 1D arrays are slower than for 2D arrays, even if both arrays are identical in size. This experiment showed that the average amount of time it takes to add two 1D arrays of size 29,997,529 is ~3.421 ms and the average time per operation between two elements in both arrays ~1.141e-7 ms. For test2.html, the average amount of time to add two 2D arrays of size 5,477x5,477 was ~0.051 ms and average time per operation was ~1.702e-9 ms. In conclusion, it takes ~67x longer to add two 1D arrays than it takes to add two 2D arrays. It also took 67x longer to perform addition between two elements in a 1D array as opposed to a 2D array.

CONCLUSION

Mostly likely the reason for such a discrepancy is that the 2D array, despite its dimensions equaling the total size of a 1D array, is actually much smaller in memory than a 1D array. I suspect that the 2D array is actually stored in memory as a 1D array of size 5,477 while the actual 1D array in test.html is of size 29,997,529. Indeed, in Javascript, 2D arrays are implemented as an array of arrays. So when operations are performed between two arrays, for test.html, that operation is done for two large arrays of size 29,997,529 but for test2.html the operation is performed between two smaller arrays of size 5,477, which could significantly affect the time it takes to perform addition between the two arrays.