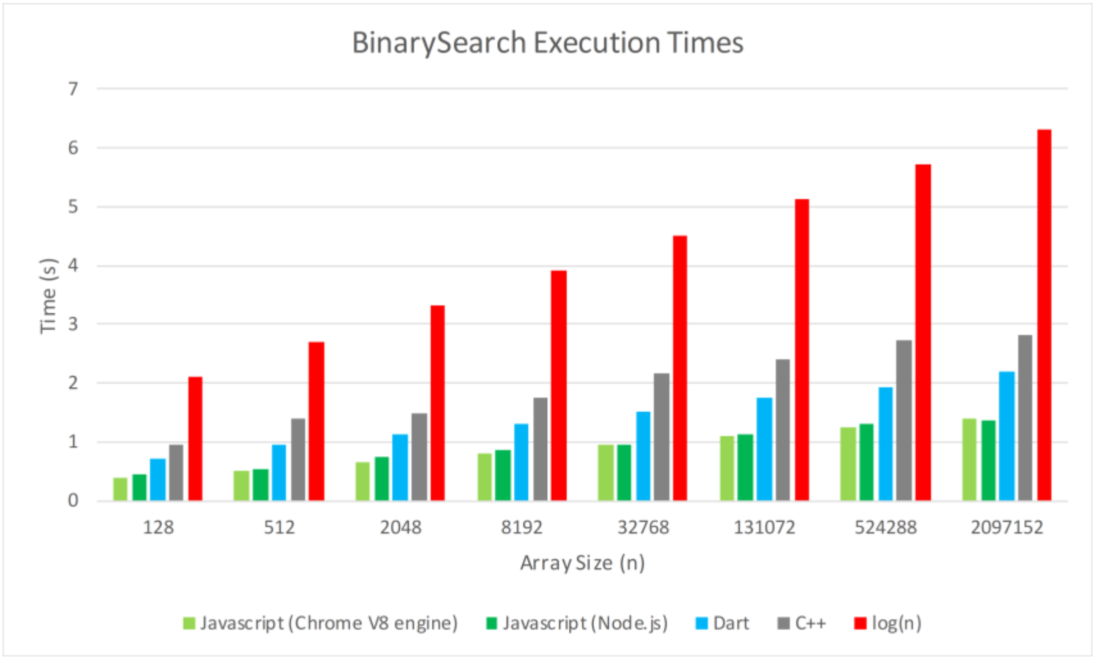
Question 5

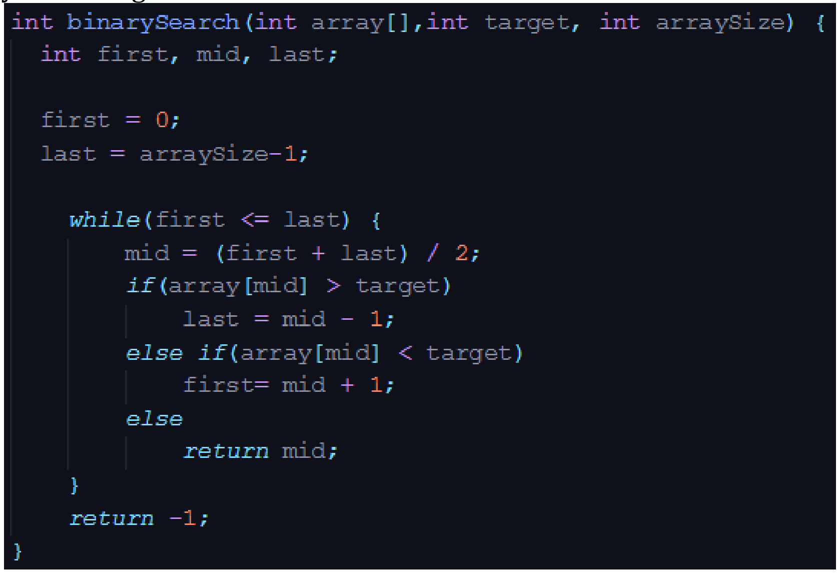
Sunday, January 20, 2019 2:48 PM



Test: This is a comparison of Java, C++, and Dart's worst-case performance with BinarySearch

- Array sizes: 128, 512, 2048, 8192, 32768, 131072, 524288, and 2097152
- Each array element was initialized based off of their element number
  - Example: an array of size 128 is [1,2, … , 127,128]
- The value chosen to search for in each array was the $sizeArr + 1$
- The algorithm's worst-case time complexity, independent of language, is $O(\log(n))$ where n is the array size

BinarySearch Algorithm:



```
int binarySearch(int array[],int target, int arraySize) {
  int first, mid, last;

  first = 0;
  last = arraySize-1;

    while(first <= last) {
        mid = (first + last) / 2;
        if(array[mid] > target)
            last = mid - 1;
        else if(array[mid] < target)
            first= mid + 1;
        else
            return mid;
    }
    return -1;
}
```

Findings: It is clear that JavaScript performed the best while surprisingly C++ was the slowest. Each language has a time growth in a logarithmic fashion, mirroring the growth of $O(\log(n))$. The varied array access times across these languages, causing the time difference, is shown to have minimal long-term growth impact. The main take away is that there is no clear benefit to use any one of these specific languages to gain substantially better performance using BinarySearch.