Paige Rosynek
SE 2840 031
Dr. Lembke
12.14.2022

Lab 02 - Javascript Coin Flipper

Results:

Firefox (Version 107.0.1)	**Microsoft Edge (Version 108.0.1462.46)**	**Java 11**
30.00 ms	93.30 ms	202 ms
30.00 ms	90.70 ms	212 ms
30.00 ms	86.50 ms	208 ms
31.00 ms	92.90 ms	211 ms
30.00 ms	90.30 ms	207 ms
Average = 30.2 ms	**Average** = 90.74 ms	**Average** = 208 ms

Analysis:

For my experiment, I timed the execution time of the coin flipper on Firefox and Microsoft Edge. Note that, average execution times were calculated based on 5 runs with 10 coins and 1,000,000 flips.

Javascript:

i) Firefox

```
Number of times each head count occurred in 1000000 flips of 10 coins :
 0  986
 1  9616  *
 2  44044  ****
 3  117852  ***********
 4  204877  ********************
 5  245730  ************************
 6  204520  ********************
 7  117503  ***********
 8  44059  ****
 9  9893  *
 10  920
Coin flipper time : 30.000000 ms
```

ii) Microsoft Edge

```
Number of times each head count occurred in 1000000 flips of 10 coins :
 0  981
 1  9728  *
 2  43981  ****
 3  117568  ***********
 4  204499  ********************
 5  247222  *************************
 6  204414  ********************
 7  117111  ***********
 8  43845  ****
 9  9687  *
 10  964
Coin flipper time : 93.2999999821186 ms
```

iii) Java

```
Enter the number of coins to be flipped: 10
Enter the number of flips: 1000000
Number of times each head count occurred in 1000000 flips of 10 coins:
 0  963
 1  9782  *
 2  43862  ****
 3  117524  ***********
 4  204816  ********************
 5  246390  *************************
 6  204907  ********************
 7  117153  ***********
 8  43832  ****
 9  9756  *
 10  1015
Coin Flipper Time: 202ms
```

1) **Was javascript faster or slower than Java? Why do you think this is the case?**

   After completing each experiment, the average time it took for the Javascript implementation to execute the coin flipper was consistently faster than the Java implementation. The average Javascript execution time after 5 runs on Firefox was 30.2 milliseconds and on Edge the average execution time was 90.74 milliseconds. For Java, the average execution time was 208

milliseconds which is longer than both Javascript engines tested. Commonly, compiled languages like Java are faster than interpreted languages like Javascript. However, in my experiment I observed the opposite of this notion; Javascript was consistently faster than the Java implementation. As explained in the DiscoverDataScience article, found here: https://www.discoverdatascience.org/articles/java-vs-javascript/, comparing these two languages depends on the context the languages are being used because they are run in different environments with different dependencies. The performance of the Java code is dependent on the JVM and the performance of the Javascript code is dependent on the browser's Javascript engine. Therefore, I think that for the relatively simple loops tested for both languages, the both web browsers' Javascript engine better optimized the tested code than Java.

2) **What browsers did you use in your evaluation? Which one was fastest, slowest?**
   As mentioned above, I used Firefox (Version 107.0.1) and Microsoft Edge (Version 108.0.1462.46) to evaluate the execution time of the coin flipper. Firefox was faster than Microsoft Edge. The average Javascript execution time after 5 runs on Firefox was 30.2 milliseconds and on Edge the average execution time was 90.74 milliseconds.

3) **Research your browser's JavaScript execution engine to determine why you got those execution speeds? Do not forget to cite your resources.**
   The Javascript execution engine used by Firefox is the C++-built engine, SpiderMonkey. According to W3Schools, the new Microsoft Edge is Chromium based and uses the Javascript execution engine called V8 which is also used by Chrome. According to a comparison done by Geeks4Geeks, it is evident that the process SpiderMonkey and V8 go through to interpret and execute Javascript are very similar. V8 first generates an abstract syntax tree and generates bytecode from it. Next, the bytecode is compiled to machine code using just-in-time compiling (JIT) before executing. At runtime, V8 further optimizes the code dynamically. Similarly, SpiderMonkey first parses the code into an abstract syntax tree and generates bytecode from this, according to the SpiderMonkey documentation. The bytecode is then interpreted and converted to machine code. SpiderMonkey also uses JIT to speed up the

execution of the bytecode. SpiderMonkey also has a garbage collector to manage any memory allocated on the heap. After researching both Javascript engines, I came to the conclusion that Firefox's SpiderMonkey did a better job optimizing the tested code better than the V8 engine.

SpiderMonkey: https://firefox-source-docs.mozilla.org/js/index.html
W3Schools New Edge: https://www.w3schools.com/js/js_ie_edge.asp
Geeks4Geeks comparison:
https://www.geeksforgeeks.org/what-happens-inside-javascript-engine/
V8: https://en.wikipedia.org/wiki/V8_(JavaScript_engine)

Suggestions:

I enjoyed this lab. I liked how the lab was simply focused on Javascript without requiring us to connect the Javascript to HTML elements right away after we learned the language. In other words the lab was not overwhelming. Having the Java code to translate was nice to have, however it actually made the lab really easy because we only had to worry about the structure/syntax of the Javascript implementation without having to worry about also having to decipher the logic of the functions. The one thing I struggled with in this lab was finding recent and relevant information on the Javascript engines for the browsers I used. Throughout my research, I found a lot of conflicting information which made answering question 3 difficult. Overall, I think that this lab was a good exercise in learning and experimenting with the Javascript language for the first time.