

# **EMPIRICAL ANALYSIS ON JAVASCRIPT ENGINE PERFORMANCE**

Mohottige Don Thilina Dulanjana

179317N

M.Sc. in Computer Science

Department of Computer Science and Engineering

Faculty of Engineering

University of Moratuwa

Sri Lanka

August 2022

# **EMPIRICAL ANALYSIS ON JAVASCRIPT ENGINE PERFORMANCE**

Mohottige Don Thilina Dulanjana

179317N

Thesis submitted in partial fulfillment of the requirements for the degree

M.Sc. in Computer Science

Department of Computer Science and Engineering

Faculty of Engineering

University of Moratuwa

Sri Lanka

August 2022

## DECLARATION

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other University or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

*UOM Verified Signature*

05/08/2022

M. D. T. Dulanjana

Date

The above candidate has carried out research for the Masters thesis under my supervision. I confirm that the declaration made above by the student is true and correct.

*UOM Verified Signature*

05/08/2022

Dr. Kutila Gunasekara

Date

## ABSTRACT

JavaScript is a prominent scripting language in web browsers. Earlier JavaScript was only used for client-side programming of browsers. JavaScript has been becoming a mainstream and a significant language in all types of large-scale applications from recent past.

JavaScript is an interpreted loosely typed language. JavaScript does not compile the source beforehand instead it translates codes at run time, line by line. JavaScript does not contain data types. Therefore, it is called a loosely typed language as well.

JavaScript does not consist fully pledged Object-Oriented concepts implemented. Instead, it uses prototypes to mimic the behavior of Object Orientation.

JavaScript was initially developed in 10 days by the founder Brendan Eich during the emergence of dot-com bubble and browser wars. At later points he himself had also admitted regarding some design mistakes of JavaScript.

Due to these reasons JavaScript has certain innate performance issues. To address these issues many researches have been done. Most of these researches have been focusing on fixing the performance issues of interpreting and loosely typed nature, by such solutions as JIT compilers and Typescript.

Hence this research will be focusing on an area which was given less significance, namely the coding standards. Research will first validate the common notion of coding standards have an impact on performance of JS and continue to provide an analytic document as a reference document of setting up a coding standards for any particular entity.

## **ACKNOWLEDGEMENTS**

I would like to express profound gratitude to my advisors, Dr. Kutila Gunasekara and Dr. Malaka Walpola, for their invaluable support by providing relevant knowledge, materials, advice, supervision, and useful suggestions throughout this research work. Their expertise and continuous guidance enabled me to complete my work successfully.

Further I would like to thank all my fellow students who helped me on finding the relevant research material, sharing knowledge and experience and for their encouragement.

I am as ever, especially indebted to my parents for their love and support throughout my life. Finally, I wish to express my gratitude to all my colleagues at all my workplaces, for the support given to me to manage my MSc research work.

## TABLE OF CONTENTS

LIST OF ABBREVIATIONS	14
CHAPTER 1	15
INTRODUCTION	15
1.1 Introduction to JavaScript .....	15
1.2 Problem Identification .....	15
1.3 Research Objectives .....	16
1.4 Motivation .....	16
1.5 Summary of Research Findings.....	18
1.5.1 Coding standard changes .....	18
1.5.2 Just in Time Compiling.....	21
1.5.3 Type Deconstructing.....	22
CHAPTER 2	24
LITERATURE REVIEW	24
2.1 Status of JavaScript Research.....	24
2.2 JavaScript Engines.....	25
2.3 Analyzing JavaScript .....	26
2.3.1 Dynamic Analysis.....	26
2.3.2 Static Analysis .....	26
2.4 Benchmarks .....	27
2.5 Profilers .....	29
2.6 Performance.....	29
2.6.1 Execution time .....	30
2.6.2. Throughput .....	30
2.6.3 Resource Consumption.....	30
2.6.4 Response Time.....	30
2.6.5 Bandwidth.....	31

2.7 Properties of Performance evaluation methods .....	31
2.7.1 Accuracy .....	31
2.7.2 Impact .....	31
2.7.3 Usability.....	32
2.7.4 Portability .....	32
2.8 JavaScript Frameworks.....	32
2.9 Root Causes for Performance Degradation .....	32
CHAPTER 3	36
METHODOLOGY	36
3.1 Overview of the Prospective Methodology .....	36
3.2 Identifying coding standards .....	37
3.3 Analyzing Optimizations .....	37
3.4 Building Standards .....	39
CHAPTER 4	40
IMPLEMENTATION	40
4.1 Introduction .....	40
4.2 Benchmarking.....	40
1.5 Understanding and avoiding common mistakes in Performance Analysis .	43
4.4 System Environment.....	44
4.5 Output of the implementation.....	45
CHAPTER 5	47
EVALUATION	47
5.1 Chapter Overview.....	47
5.2 Evaluation process.....	47
5.3 Impact of platform variance on Standardization document.....	48
5.3.1 Performance impact by various hardware capacities .....	48
5.3.2 Performance impact by various Operating Systems.....	50

5.4 Performance Test Results .....	52
5.4.1 Variable Handling .....	52
5.4.1.1 Usage of var and let .....	52
5.4.2 String Operations .....	54
5.4.2.1 String concatenation .....	54
5.4.2.2 Concatenate list of strings.....	55
5.4.2.3 Find the existence of a given term.....	57
5.4.2.4 Find a given term.....	58
5.4.2.5 Splitting a string.....	60
5.4.3 Array Operations.....	62
5.4.3.1 Search an element of an array.....	63
5.4.3.2 Search an object element of an array.....	65
5.4.3.3 Adding an element to array by index.....	67
5.4.3.4 Remove an element from an array.....	68
5.4.3.5 Concatenate Arrays .....	69
5.4.4 Object Operations .....	71
5.4.4.1 Find an Add/search/edit/delete a prop of object in a list by index .....	71
5.4.4.2 Merge Objects.....	73
5.4.4.3 Add a Property to an Object .....	75
5.4.4.4 Remove a property from an object .....	76
5.4.4.5 Check the Existence of a Prop.....	78
5.4.5 Iteration.....	79
5.4.5.1 For loop.....	79
5.4.6 Asynchronous functions .....	81
5.4.7 Other special operations .....	83
5.4.7.1 Equals Operation .....	83



CHAPTER 6	85
CONCLUSION	85
6.1 Research Findings and Concerns.....	85
6.2 Future work.....	86
6.2.1 Roadmap of the Web application and Future Concerns .....	88
REFERENCES .....	90

## LIST OF FIGURES

<u>Figure 1. 1 Number of operations per second for string concatenations</u> .....	18
<u>Figure 1. 2 iGoogle HTTP traffic with an empty cache.</u> .....	20
<u>Figure 2.1 Identified issues in JavaScript projects</u> .....	33
<u>Figure 2. 2 Principal Root causes</u> .....	34
<u>Figure 2. 3 Count of issue types in API usage</u> .....	35
<u>Figure 3. 1 Overview of suggested methodology</u> .....	36
<u>Figure 3. 2 Improved performance against the effort</u> .....	38
<u>Figure 3. 3 Speed variation with SpiderMonkey versions</u> .....	38
<u>Figure 3. 4 Speedup variation with V8 versions</u> .....	39
<u>Figure 4.1:Input screen of jsbench.github.io</u> .....	40
<u>Figure 4.2: Output screen of jsbench.github.io</u> .....	41
<u>Figure 4.3: Input screen of jsben.ch</u> .....	41
<u>Figure 4.4: Output screen of jsben.ch</u> .....	42
<u>Figure 4.5: Input screen of measurethat.net</u> .....	42
<u>Figure 4.6: Output results screen of measurethat.net</u> .....	42
<u>Figure 4.7: Output chart screen of measurethat.net</u> .....	42
<u>Figure 5.1: Chart of results depicting impact of hardware platform difference on javascript performance</u> .....	49
<u>Figure5.2: Chart of results depicting impact of operating system difference on javascript performance</u> .....	51
<u>Figure 5.3: Results chart of var and let</u> .....	53
<u>Figure 5.4: Result chart of String concatenation</u> .....	55
<u>Figure 5.6: Results chart of match and search</u> .....	58
<u>Figure 5.7: Results chart of index of, include, match and search</u> .....	60
<u>Figure 5.8: Result chart of Splitting a string</u> .....	62

<u>Figure 5.9: Results chart of Search an element of an array</u> .....	64
<u>Figure 5.10: Results chart of Search an object element of an array</u> .....	67
<u>Figure 5.11: Results chart of adding an element to array by index</u> .....	68
<u>Figure 5.12: Results chart of remove an element from an array</u> .....	69
<u>Figure 5.13: Results chart of concatenate arrays</u> .....	71
<u>Figure 5.14: Results chart of find an add/search/edit/delete a prop of object in a list by index</u> .....	73
<u>Figure 5.15: Results chart of Merge Objects</u> .....	75
<u>Figure 5.16: Result chart of Add a Property to an Object</u> .....	76
<u>Figure 5.17: Result chart of Remove a property from an object</u> .....	77
<u>Figure 5.18: Result chart of Check the Existence of a Prop</u> .....	79
<u>Figure 5.19: Result chart of For Loop</u> .....	81
<u>Figure 5.20: Result chart of Asynchronous functions</u> .....	83
<u>Figure 5.21: Result chart of Equals Operation</u> .....	84
<u>Figure 6.1 Proposed web interface of performance digital document part 1</u> .....	87
<u>Figure 6.2: Proposed web interface of performance digital document part 2</u> .....	88

## LIST OF TABLES

<u>Table 2.1 Details of JavaScript engines</u> .....	25
<u>Table 2.2 Details of JavaScript Benchmarks</u> .....	28
<u>Table 5.1: Higher and Lower hardware specifications</u> .....	49
<u>Table 5.2: Results depicting impact of hardware platform difference on javascript performance</u> .....	49
<u>Table 5.3: Results depicting impact of operating system difference on javascript performance</u> .....	50
<u>Table 5.4: Results set, usage of var and let</u> .....	53
<u>Table 5.5: Results set, usage of String concatenation</u> .....	54
<u>Table 5.6: Results set, usage of Concatenate list of strings</u> .....	56
<u>Table 5.7: Results set, usage of match and search</u> .....	58
<u>Table 5.8: Results set, usage of index of, include, match and search</u> .....	59
<u>Table 5.9: Results set, Splitting a string</u> .....	62
<u>Table 5.10: Results set, Search an element of an array</u> .....	64
<u>Table 5.11: Results set, Search an object element of an array</u> .....	66
<u>Table 5.12: Results set, adding an element to array by index</u> .....	68
<u>Table 5.13: Results set, remove an element from an array</u> .....	69
<u>Table 5.14: Results set, concatenate arrays</u> .....	70
<u>Table 5.15: Result set, find an add/search/edit/delete a prop of object in a list by index</u> .....	72
<u>Table 5.16: Result set, Merge Objects</u> .....	74
<u>Table 5.17: Result set, Add a Property to an Object</u> .....	76
<u>Table 5.18: Result set, Remove a property from an object</u> .....	77
<u>Table 5.19: Result set, Check the Existence of a Prop</u> .....	78

<u>Table 5.20: Result set, For Loop</u> .....	80
<u>Table 5.21: Result set Asynchronous functions</u> .....	8282
<u>Table 5.22: Result set of Equals Operation</u> .....	84

## **LIST OF ABBREVIATIONS**

API – Application Package Interface

CDN – Content Delivery Network

CSS – Cascading Style Sheets

DNS – Domain Name Server

HTML – Hyper Text Markup Language

JIT – Just in Time

JS – JavaScript

JSON – JavaScript Object Notation

SQL – Structured Query Language