On Comparing Software Quality Metrics of JavaScript vs TypeScript Software Project: An Empirical Study

Tulshi Chandra Das[1], Maloy Kanti Sarker[2]
Institute of Information Technology
University of Dhaka

Abstract- With the growth of popularity in dynamic programming world JavaScript have become a well-known technology in every field of software development. It gives good flexibility in programming though some issues arise in project maintenance. TypeScript the superset of JavaScript has come with OOP based programming philosophy aims to provide more maintainable software development environment. In this work we have done a statistical analysis based on some software quality matrix. We compared 10 open source JavaScript projects with 10 TypeScript projects to find significant difference. From the analysis we find significant differences in quality in average cyclomatic complexity, number of statements and maintainability index.

Index Terms- quality matrix, comparison, JavaScript, TypeScript

I. Introduction

JavaScript started its journey as a fronted programming. But now a days JavaScript has gained popularity also in mobile and desktop app development. Node Js framework has come to provide facility to use JavaScript at serve side programming. JavaScript is dynamic typed programming language. Since ES6 class-based programming has been introduced.

Empirical studies that describe what takes place through direct observations, discussion of focus groups and thorough interviews are described as qualitative studies. [1] These include case reports and research studies with a population limited that are not intended to provide statistical linkages between variables. Qualitative empirical studies can provide rich and thorough contextual data to understand a phenomenon. However, the prevalence or incidence of a phenomenon cannot be generalized and the association between factors cannot be shown in general. Empirical studies to highlight or determine the prevalence or incidence of a phenomenon should use quantitative methods such as section-by-section surveys of a properly large sample size. This survey could

describe who, what and where (descriptive) of a phenomenon but couldn't answer the question of why. An analysis or an experimental study is required to answer the question of why (cause).

JavaScript is a popular, dynamically interpreted programming language with a straightforward syntax [2]. JavaScript was used mainly for client-side features and used in a browser. It now also works on the servers. Recent surveys by Stack Overflow show JavaScript topping the rankings of popular programming languages for seven years in a row. [3] Many developers and businesses use JavaScript technology for the manufacturing industry and it is the language with the most active repositories and GitHub pushes. [4]

JavaScript is dynamic, weakly designed, and has premium features. It is a class free programming language oriented towards objects which utilizes a prototypical legacy rather than classical inheritance. In addition, JavaScript is an interpreted language, so developers don't have a compiler to spot faulty and unoptimized code.

TypeScript, a language of Microsoft for JavaScript construction on a scale, is now one of the most common languages for Apple-backed Swift developers. TypeScript is a JavaScript superset that is the leading language for developing companies. Typescript designed to develop a large trans compiling program for JavaScript. TypeScript provides annotations that provide optional verification of the static form at compilation moment.

This paper presents an experimental study on JavaScript and typescript paradigms and measure the quality using object-oriented metrics by calculating number of statements, Cyclomatic

[1] Email: bsse0811@iit.du.ac.bd
[2] Email: bsse0834@iit.du.ac.bd

complexity, number of comments, ration of comments(percentage).

II. BACKGROUND

Software quality metrics concentrate on process, product and project quality elements. The goal of software quality metrics is to identify the improvement of project, planning, process and product. Various studies on software metrics distribution share the same objective of offering a way of improving software development life cycle. Some of them indicate that the renowned Chidamber and Kemerer (CK) suite and the problem of computer defects and maintenance are correlated. [5] [6] [7] Comparing with conventional software development, the object-oriented paradigm demonstrates some peculiarities. In this study, we use object-oriented metrics to analyses the JavaScript and typescript programming language to identify the important differences.

Object-oriented system development promotes prototype creation and utilizes its own object-oriented methods and languages of programming. Object-oriented development is commonly recognized to require a distinct way of thinking than traditional structured development and software projects move to object-oriented design. Its modularity and reusability are the major benefits of object-oriented design.

Metrics are a means to more accurately estimate project milestones and develop a software system with minimum failures. [8] Project-based metrics monitor maintenance, budgeting, and so on. Design-based metrics describe the complexity, size and strength of the methods and track design performance.

JavaScript is an interactive web pages scripting language. It works on the web browser of the user as the programming language has been followed on the client side. But this now also works on the servers. The idea behind the creation of the script is to introduce an additional scripting language. JavaScript is a language of interpretation, lightweight programming. It is a dynamic programming language. No multi-threaded or multi-processing capability is available in JavaScript. It is used for improving HTML pages in web development.

Typescript is JavaScript and additional functions. It may be called a JavaScript superset, which implies that Typescript is JavaScript and more. Typescript is an open source programming language. Statically, it is a compiled language to write straightforward and clear JavaScript code. It can be executed on Node Js or any browser supporting ECMAScript 3 or later. Optional static types, classes and interfaces are provided in typescript. For a big JavaScript project using Typescript, one can use a periodic JavaScript implementation to provide more robust software that can be deployed readily. Typescript can be applied both for the client and server side to create a JavaScript application.

III. LTERATURE REVIEW

A fresh strategy for evaluating big JavaScript apps using static analysis is presented to authorities in (Y. Ko, H. Lee, J. Dolby 2015). The investigation defines their tool and focuses on the evaluation of goods in the programming language of JavaScript.

The writers of (S. Mirshokraie, A. Mesbah 2013) focussed on the programming language of JavaScript and suggested a series of web-specific mutations. They propose a method that complements the program static and dynamic analysis to guide mutation creation processes in program code segments where there is higher risk of mistakes or where the program output can be affected. The document shows the tool for MUTANDIS and provides an efficacy evaluation (S. Mirshokraie, A. Mesbah 2013).

David Kostanjevec et al. reported on quality measurement for JavaScript solutions. They have identified several quality metrics for JavaScript solution. [2]

Marija Selakovic et al. shown the performance issues and their methodology of evaluating the performance impact of the optimizations applied to address those issues. [9]

Likewise, the paper (S. Rostami, L. Eshkevari, D. Mazinanian 2016) introduces the tool JSDeodorant, a plug-in for Eclipse. This tool enables us to observe courses in JavaScript. When examining items with a program code it can recognize the difference between modules and utilities. The primary objective of the paper is to present the methods supplied by the JSDeodorant tool, to compare the tool with the more familiar JSClassFinder tool, and to evaluate the outcomes in quantitatively and qualitatively to acknowledge its restrictions and potential for further changes (S. Rostami, L. Eshkevari, D. Mazinanian 2016).The writers in (Mesbah 2013) introduce the JSNOSE tool which utilizes the method to detect poor schemes. The tool compares the software code to 13 JavaScript samples in order for "smelly" code sections to be found. Some bad code may have a negative impact on the entire project, maintenance, or comprehension. With JavaScript frameworks increasingly used for web applications, the quality of writing code including quick maintenance, reliability and velocity is growing.

IV. GQM TREE

GQM, the initialism for "goal, question, metric", is an approach to software_metrics. [10] GQM defines measurement model of three level.

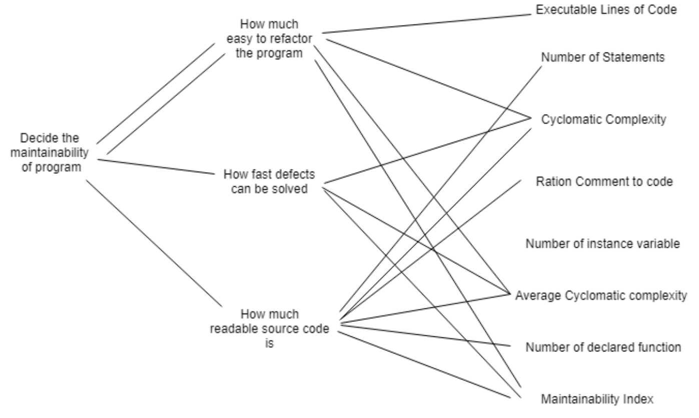
Decide the maintainability of program
How much easy to refactor the program
How fast defects can be solved
How much readable source code is
Executable Lines of Code
Number of Statements
Cyclomatic Complexity
Ration Comment to code
Number of instance variable
Average Cyclomatic complexity
Number of declared function
Maintainability Index

Figure 1: GQM tree to determine maintainability of a program

V. Data Collection

The JavaScript and Typescript projects are collected from GitHub[3]. We selected 10 JavaScript and 10 TypeScript projects with similar features. We included small and medium sized projects in both JavaScript and typescript.

JavaScript Projects

- Web Apps: Messenger, Video Downloader, Dodger game, ShareIt, Gomoku Game, Tetris Game, Connect-Four game
- Desktop Apps: Video to MP3 converter, Audio player, Video player

TypeScript Projects

- Web Apps: Video to MP3 converter, Video Downloader, Dodger game, Gomoku Game, Tetris Game, Connect-Four game
- Desktop Apps: Messenger, Audio player, Video player, ShareIt

Desktop apps are on Electron JS framework.

VI. DATA PROCESSING

To measure cyclomatic complexity and number declared function of the projects we write a TypeScript program. We use NPM package ts-complex in our program. We used VSCode-Counter extension in VSCode editor to get the number of executable line and number comments to projects.

VII. Result

We identified the metrics below:

- Executable Lines of Code
- Number of Statements
- Cyclomatic Complexity
- Ration Comment to code
- Number of instance variable
- Average Cyclomatic complexity
- Number of declared function
- Maintainability Index

From the above metrics we focused the following metrics:

- Average Cyclomatic Complexity
- Ration Comment to code
- Executable lines of code
- Number of declared function
- Maintainability Index

Average Cyclomatic Complexity: We measured the average cyclomatic corresponding to the number of functions/methods. We found a potential difference of average cyclomatic in JavaScript and TypeScript project pairs.

Table[4] 1:Average Cyclomatic Complexity comparison

	c-4	g	t	m-d	v-p	a-p	v-c	s	m	v-d
js	4.26	4.4	1.67	3.19	1.65	2.97	1.25	2.25	2.76	1.8
ts	1.35	2.45	1.68	1.39	1.23	1.63	1.15	1.34	2.03	2

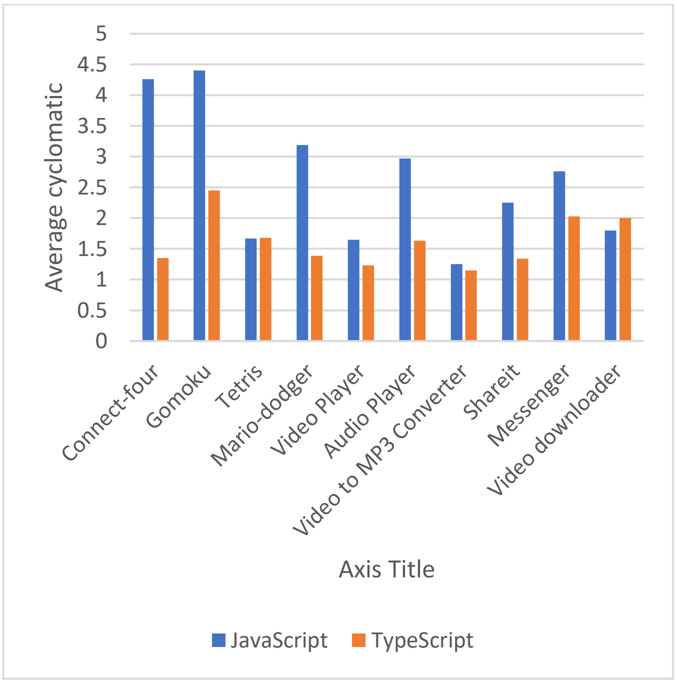

Figure 2: Average Complexity of TypeScript and JavaScript Project

[3] Project source link are given in Appendix section

[4] The full forms of words in tables are given in Appendix section

We find that for each of the 10 JavaScript and TypeScript project pair JavaScript projects have more average cyclomatic than TypeScript.

Ration comment to code:

Table 2:Ration comment comparison (in percentage)

	c-4	g	t	m-d	v-p	a-p	v-c	s	m	v-d
js	24	2.4	15	40	8.5	8	25	19	23	14.58
ts	.76	8.4	6.1	0.17	16.9	25	28	11	17	12.6

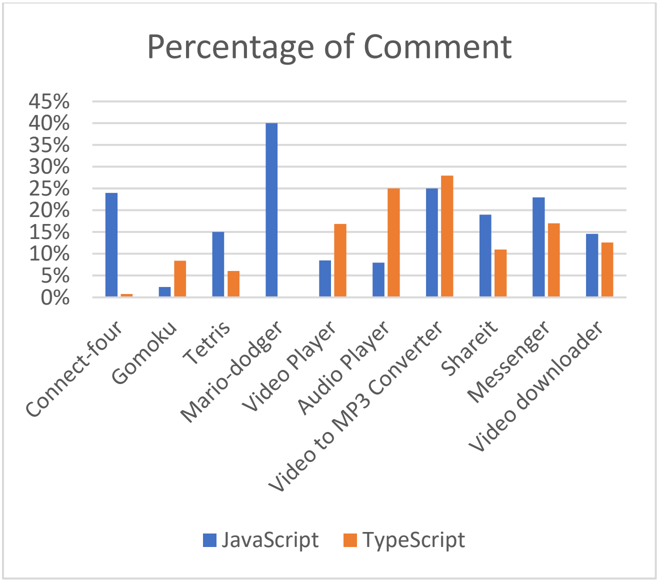

Figure 3: Ration comment to code comparison

The differences of ration comment between JavaScript and TypeScript project are different. 3 out 10 TypeScript projects have more comment ratio than JavaScript project.

Executable Lines of Code:

Table 3:Executable Lines of Code comparison

	c-4	g	t	m-d	v-p	a-p	v-c	s	m	v-d
js	309	920	1480	10409	828	433	69	3471	2751	82
ts	518	1050	645	7245	456	1739	196	1322	2362	1224

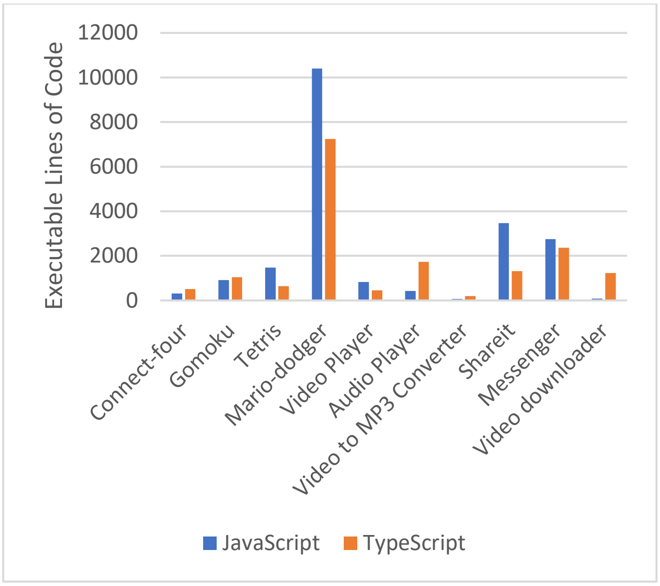

Figure 4:Executable Lines of Code Comparison

We find that number of executable lines of code of JavaScript are more than the TypeScript projects in the large projects but varied result found for the small sized projects.

Number of declared method/function:

Table 4:Number of Function comparison

	c-4	g	t	m-d	v-p	a-p	v-c	s	m	v-d
js	19	64	204	1124	98	214	8	429	429	10
ts	37	113	7	28	232	232	20	162	185	145

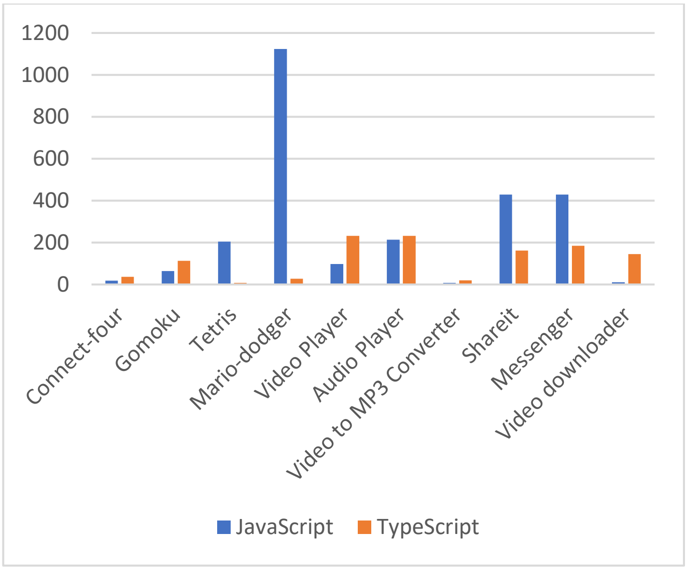

Figure 5:Comparison of number of functions

The ES6 methods of classes and call back functions are also considered in this measurement. The Figure 4 shows that 4 out of

10 JavaScript projects have a greater number of functions than TypeScript projects. In case of large projects JavaScript have more functions.

Maintainability Index:

Table 5: Maintainability Index with LOC of JavaScript Project

LOC	69	82	309	433	828	920	1480	2751	3471	10409
MI	100	98	51.2	92	73	79	82	66	51	35

The table 5 show the maintainability of JavaScript projects according to their lines of code.

Table 6:Maintainablity Index with LOC of TypeScript Projects

LOC	196	456	518	645	1050	1224	1322	1739	2362	7245
MI	36	46	33	68	64	74	73	63.9	55	41.8

The table 6 show the maintainability of TypeScript project with their lines of code.

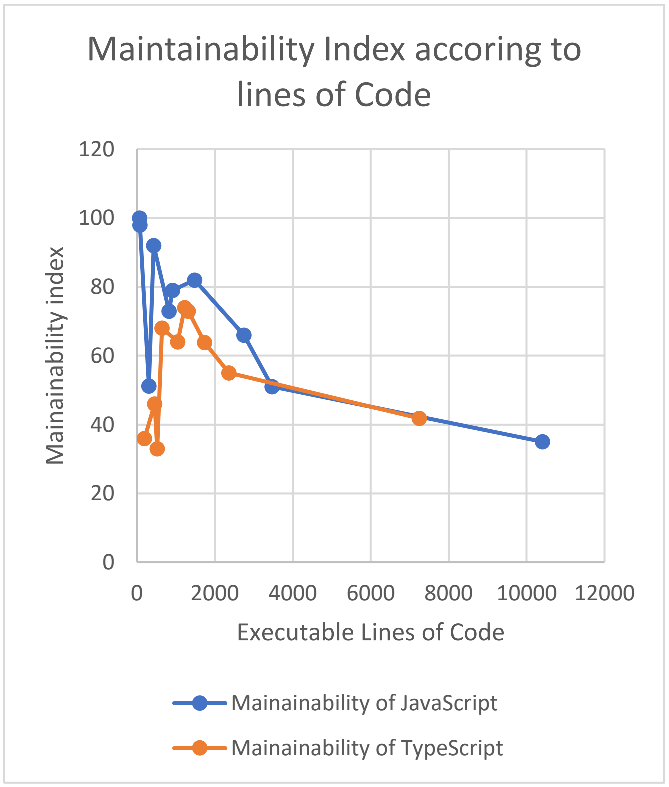

Figure 6: Maintainability Comparison

High maintainability refers good maintenance for projects. Figure-5 show that JavaScript projects with less lines of code have better maintainability. But for larger projects TypeScript project show tendency to have better maintainability level than JavaScript.

VIII. Conclusion

Through our empirical study we tried to identify the potential differences of quality code metrics between JavaScript and TypeScript project. We find a relation of average cyclomatic in JavaScript and TypeScript project. Average cyclomatic complexity of JavaScript projects are more than TypeScript projects. The large JavaScript project have more executable lines of code than TypeScript projects. In case of ration comment we cannot so strong relation between JavaScript and TypeScript projects. We also find tendency to have more maintainability of TypeScript projects than JavaScript for larger size projects.

References

[1] http://betterthesis.dk/research-methods/empirical-studies, "Better Thesis, Empirical Studies," Retrieved September 11, 2019.

[2] D. KOSTANJEVEC, M. PUŠNIK, M. HERIČKO and B. ŠUMAK, "A Preliminary Empirical Exploration of Quality Measurement for JavaScript Solutions.," in *6th Workshop of Software Quality, Analysis, Monitoring, Improvement, and Applications*, Belgrade, Serbia, 2017.

[3] Stackoverflow, "Developer survey Results 2019", from," *https://insights.stackoverflow.com/survey/2019#technology* , 11 sepetember 2019.

[4] Githut, "A SMALL PLACE TO DISCOVER LANGUAGES IN GITHUB.," *[Online]. Available: https://githut.info/*, 11 September 2019.

[5] S. R. Chidamber and C. F. Kemerer, "A Metrics Suite for Object Oriented Design," *IEEE Transactions on Software Engineering*, vol. 20, no. 6, p. 476–493, 1994.

[6] S. R. Chidamber, P. D. Darcy and C. F. Kemerer, "Managerial use of metrics for object-oriented software: An exploratory analysis. Software Engineering," *IEEE Transactions on*, vol. 24, no. 8, p. 629–639, 1998.

[7] T. Zimmermann and N. Nagappan, "Predicting defects using network analysis on dependency graphs," in *In Proceedings of the 30th international conference on Software engineering*, Leipzig, Germany, 2008.

[8] M. Tyagi, D. Bellin and M. Tyler, "Object-Oriented Metrics:An Overview," *Computer Science Department,North Carolina A ,T state University*, Vols. Greensboro, Nc 27411-0002..

[9] M. Selakovic and M. Pradel, "Performance Issues and Optimizations in JavaScript: An Empirical Study.," in *ICSE*

'16 Proceedings of the 38th International Conference on Software Engineering, New York, 2016.

[10] V. . R. Basili, G. Caldiera and H. D. Rombach, "THE GOAL QUESTION METRIC APPROACH," PDF retrieved, 2019.

Appendix

Abbreviations

a-p: Audio player
c-4: Connect four
g: Gomoku
js: JavaScript
m-d: Mario dodger
m: Messenger
s: Shareit
ts: TypeScript
v-c: Video to MP3 converter
v-d: Video downloader
v-p: Video player

Source of JavaScript Projects

1. https://github.com/bryanbraun/connect-four

2. https://github.com/Muzishell/gomoku
3. https://github.com/totaljs/messenger
4. https://github.com/ShareIt-project/ShareIt
5. https://github.com/eashish93/youtube-downloader-app
6. https://github.com/jamesgeorge007/Youtube-Mate
7. https://github.com/emilhaugberg/mario-dodger
8. https://github.com/mdietger/JS-Tetris
9. https://github.com/Aveek-Saha/MusicPlayer
10. https://github.com/nokisnojok/electron-player

Source of TypeScript Projects

1. https://github.com/fabien0102/connect4react
2. https://github.com/jolestar/gomoku-wasm
3. https://github.com/sindresorhus/caprine
4. https://github.com/dapplabs/shareit
5. https://github.com/jimbuck/pully
6. https://github.com/webDva/Baka-Youtube-to-MP3-Converter
7. https://github.com/alexstrive/dodger
8. https://github.com/henshmi/Tetris.ts/tree/master/src
9.https://github.com/akabekobeko/examples-electron/tree/master/audio-player/src
10.https://github.com/eranshmil/video-player/tree/master/projects/main