University of Stuttgart
Germany

How Are Different Asynchronous Programming Constructs in JavaScript Related to Software Quality? A Repository Mining Study on GitHub

BSc Thesis

Organization

Examiner:	Prof. Dr. Stefan Wagner
Supervisors:	Dr. Justus Bogner
Students:	Gamze Şevik <st142819@stud.uni-stuttgart.de>
Timeframe:	2022-05-30 – 2022-11-30

Context & Motivation

Since its introduction for small client-side tasks in the browser in 1995, JavaScript [1] has become the lingua franca of web development. It was voted the most popular language [2] and was the most used language on GitHub until the last quarter of 2021 [3]. However, understanding JavaScript applications involves challenges for developers. There are potential factors of JavaScript, such as its dynamic, asynchronous and event-driven nature, the dynamic interplay between JavaScript and the Document Object Model, and the asynchronous communication between client and server [4], which may hinder comprehension.

Because JavaScript is single-threaded, callback, async/await, and promise functions are frequently used to simulate concurrency. Nested, anonymous and asynchronous callback scheduling [5] is used regularly to provide capabilities such as non-blocking I/O and concurrent request handling. Non-trivial callback-oriented programming tends to result in nested hierarchies of callback functions, which makes following the program flow hard - a problem described as "callback hell" [6]. Long term maintenance of large applications may be severely impacted due to tight coupling of callbacks and structural fragility. Handling errors and coordinating asynchronous tasks can quickly get messy if programming discipline is not enforced and proper patterns are not followed. Furthermore, a “callback hell” program comes with increased risk of introducing security vulnerabilities [7][8][9]. However, there is still a lack of empirical evidence how different asynchronous programming constructs in JavaScript impact software quality.

Contact:
Dr. Justus Bogner
justus.bogner@iste.uni-stuttgart.de
Institute of Software Engineering, Empirical Software Engineering Group

Objectives and Tasks

The goal of this study is therefore to empirically analyze a large set of JavaScript projects that use asynchronous programming constructs. The data collection should provide insights into a potential influence of different asynchronous programming constructs like callbacks, promises, etc. on software qualities, e.g., functional correctness, performance efficiency, or maintainability. The concrete quality aspects to be analyzed as well as more detailed research questions should be defined by the student.

Methods

The research should be conducted as a mining software repository (MSR) study [10][11][12] using a large number of open-source projects on GitHub [13][14]. Data collection should be conducted with appropriate tools (e.g., using static analysis tools or the GitHub API) and automated as much as possible to achieve the best possible reproducibility. For the analysis, suitable techniques could be hypothesis testing, correlation, or regression. The detailed study design will be created by the student.

References

[1] https://www.javascript.com

[2] https://insights.stackoverflow.com/survey/2021#most-popular-technologies-language-prof

[3] https://madnight.github.io/githut/#/pull_requests/2021/4

[4] S. Tilkov and S. Vinoski, "Node.js: Using JavaScript to Build High-Performance Network Programs," in IEEE Internet Computing, vol. 14, no. 6, pp. 80-83, Nov.-Dec. 2010, doi: https://doi.org/10.1109/MIC.2010.145

[5] K. Gallaba, A. Mesbah and I. Beschastnikh, "Don't Call Us, We'll Call You: Characterizing Callbacks in Javascript", in Proceedings of ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 2015, pp. 1-10, doi: https://doi.org/10.1109/ESEM.2015.7321196

[6] A. Saboury, P. Musavi, F. Khomh and G. Antoniol, "An empirical study of code smells in JavaScript projects", in Proceedings of IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), 2017, pp. 294-305, doi: https://doi.org/10.1109/SANER.2017.7884630

Contact:
Dr. Justus Bogner
justus.bogner@iste.uni-stuttgart.de
Institute of Software Engineering, Empirical Software Engineering Group

[7] Zheng, Yunhui, Tao Bao, and Xiangyu Zhang. "Statically locating web application bugs caused by asynchronous calls", in Proceedings of the 20th international conference on World wide web, 2011, pp. 805–814, doi: https://doi.org/10.1145/1963405.1963517

[8] A. M. Fard and A. Mesbah, "JSNOSE: Detecting JavaScript Code Smells," 2013 IEEE 13th International Working Conference on Source Code Analysis and Manipulation (SCAM), 2013, pp. 116-125, doi: https://doi.org/10.1109/SCAM.2013.6648192

[9] A. Yamashita and L. Moonen, "Do code smells reflect important maintainability aspects?", 28th IEEE International Conference on Software Maintenance (ICSM), 2012, pp. 306-315, doi: https://doi.org/10.1109/ICSM.2012.6405287

[10] H. Hemmati et al., "The MSR Cookbook: Mining a decade of research", in Proceedings of the 10th Working Conference on Mining Software Repositories (MSR), 2013, pp. 343-352, doi: https://doi.org/10.1109/MSR.2013.6624048

[11] A. E. Hassan, "The road ahead for Mining Software Repositories", in Proceedings of Frontiers of Software Maintenance, 2008, pp. 48-57, doi: https://doi.org/10.1109/FOSM.2008.4659248

[12] G. Robles, "Replicating MSR: A study of the potential replicability of papers published in the Mining Software Repositories proceedings", in Proceedings of 7th IEEE Working Conference on Mining Software Repositories (MSR 2010), 2010, pp. 171-180, doi: https://doi.org/10.1109/MSR.2010.5463348

[13] https://github.com/

[14] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "The promises and perils of mining GitHub", in Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014, 2014, Association for Computing Machinery, New York, NY, USA, 92–101, doi: https://doi.org/10.1145/2597073.2597074

Contact:
Dr. Justus Bogner
justus.bogner@iste.uni-stuttgart.de
Institute of Software Engineering, Empirical Software Engineering Group