Institute of Software Technology

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelor Thesis

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

Gamze Şevik

Course of Study:	Software Engineering
Examiner:	Prof. Dr. Stefan Wagner
Supervisor:	Dr. Justus Bogner
Commenced:	May 30, 2022
Completed:	February 28, 2022

Abstract

Since its introduction for small client-side tasks in the browser in 1995, JavaScript has become the lingua franca of web development. It was voted the most popular language and was the most used language on GitHub until the last quarter of 2021. 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, 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 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". 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. However, there is still a lack of empirical evidence how different asynchronous programming constructs in JavaScript impact software quality.

Contents

1 Introduction **17**
1.1 Motivation 17
1.2 Research Questions 17
1.3 Thesis structure 18

2 Background and Fundamentals **19**
2.1 JavaScript 19
2.2 Asynchronous Programming 19
2.3 Software Quality 19

3 Related Work **21**

4 Methodology **23**
4.1 Study Objects 23
4.2 Data Collection 23
4.3 Hypotheses 23
4.4 Statistical Analysis 23

5 Analysis and Results **25**
5.1 RQ1: Does using “callbacks” lead to less functional correctness or maintainability? 25
5.2 RQ2: Does using “async/await” lead to less functional correctness or maintainability? 25
5.3 RQ3: Does using “promises” shows better functional correctness or maintainability? 25

6 Discussion **27**

7 Threats to Validity **29**

8 Conclusion **31**

List of Figures

List of Tables

List of Listings

List of Algorithms

Acronyms

API Application Programming Interface. 17

1 Introduction

1.1 Motivation

In 1995, *JavaScript*[1] was developed and introduced for small client-side tasks in the browser. With the GitHub Application Programming Interface (API).

1.2 Research Questions

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

1.3 Thesis structure

The structure of the thesis will be presented as follows:

Chapter 2 (Background and Fundamentals) ....

2 Background and Fundamentals

2.1 JavaScript

2.2 Asynchronous Programming

2.2.1 Asynchronous Callbacks

2.2.2 ES6: Generator Functions

2.2.3 ES7: Async/Await

2.2.4 Promises

2.3 Software Quality

3 Related Work

4 Methodology

4.1 Study Objects

4.1.1 Selection of the Mining Platform

4.1.2 Requirements

4.1.3 Sampling

4.2 Data Collection

4.2.1 Metrics

4.3 Hypotheses

4.4 Statistical Analysis

4.4.1 Hypothesis Testing

4.4.2 Correlation Tests

4.4.3 Descriptive Evaluation

5 Analysis and Results

5.1 RQ1: Does using “callbacks” lead to less functional correctness or maintainability?

5.2 RQ2: Does using “async/await” lead to less functional correctness or maintainability?

5.3 RQ3: Does using “promises” shows better functional correctness or maintainability?

6 Discussion

This chapter discusses the collected results by addressing each of the research questions with their respective hypotheses.

7 Threats to Validity

There are many threats to validity in a large-scale study. Reasons for this include the unavoidable automation and the vast amount of data. It is important to describe these threats and explain how they were mitigated so that the results are credible and trustworthy. Furthermore, researchers can take them into account in replication studies or in further research. Therefore, the following points are listed where threats can occur and how attempts have been made to keep them to a minimum. The metrics bug-fix commit ratio, average time a bug issue is open, code smells per LoC and cognitive complexity per LoC were adopted from other studies and adjusted to incorporate the researchers' experiences and solutions to the same challenges.

Sample

Data Collection

8 Conclusion

This chapter summarizes the thesis and suggests possible directions for future work.

All links were last followed on February 15, 2023.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature