Seminararbeit im Seminar Trends der Softwaretechnik
Sommersemester 2025

Vor- und Nachteile verschiedener Typsysteme im Kontext von Web-Anwendungen

Nils Derenthal (7217566)
nils.derenthal002@stud.fh-dortmund.de
Informatik Dual

Inhaltsverzeichnis

1 Einleitung

Die Verlässlichkeit von Web-Anwendungen hat in der heutigen Zeit eine so hohe Relevanz wie noch nie zuvor und die Wirtschaftlichkeit von Dienstleistenden IT-Unternehmen hängt stark von der Qualität ihrer Software ab.

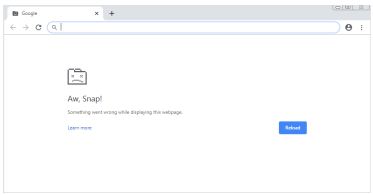
1.1 Grundlagen

Was muss jemand mit eurem Wissensstand zusätzlich wissen, um euer Thema zu verstehen.

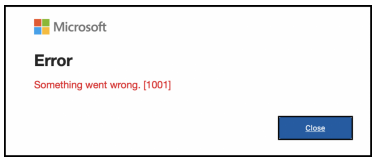
Für die Abeit wird im folgenden angenommen, dass Leser*innen ein Basiswissen in der Webentwicklung besitzen, auch wenn keine Details diesbezüglich notwendig sind. Es ist desweiteren hilfreich, wenn Kenntnisse über eine Sprache mit typisierung vorhanden sind (Beispielsweise Klassen und Objekttypen in Java, types in TypeScript oder enums in Rust)

1.2 Problemstellung

Fehler in der Softwareentwicklung sind auch heutzutage noch sehr präsent. Ob in einem Studienprojekt, der Website einer Behörde, oder dem Betriebssystem eines Millardenkonzerns: Im Alltag trifft man nicht selten auf solche Fehler.



(a) Ein Crash im Chrome-Browser



(b) Ein Fehler während des Microsoft Logins

Abbildung 1: Fehler in alltäglichen Anwendungen

Kaum eine Software is frei von Fehlern, welche je nach schwere ein großes Betriebsrisiko darstellen, weswegen die Vermeidung von diesen eine hohe Bedeutung in der Entwicklung hat. Ein Ansatz, welcher in den letzen Jahren zunehmend an Bedeutung gewonnen hat sind stärkere Typsysteme, welche vermeintlich dafür sorgen, dass Fehler bereits in der Entwicklung abgefangen werden können. Beispiele für jüngere Entwicklungen in diesem Bereich sind Technologien wie TypeScript, welches versucht JavaScript zu typisieren, oder Rust, als eine Sprache die einen Fokus auf ein ausdrucksstarkes Typsystem legt.

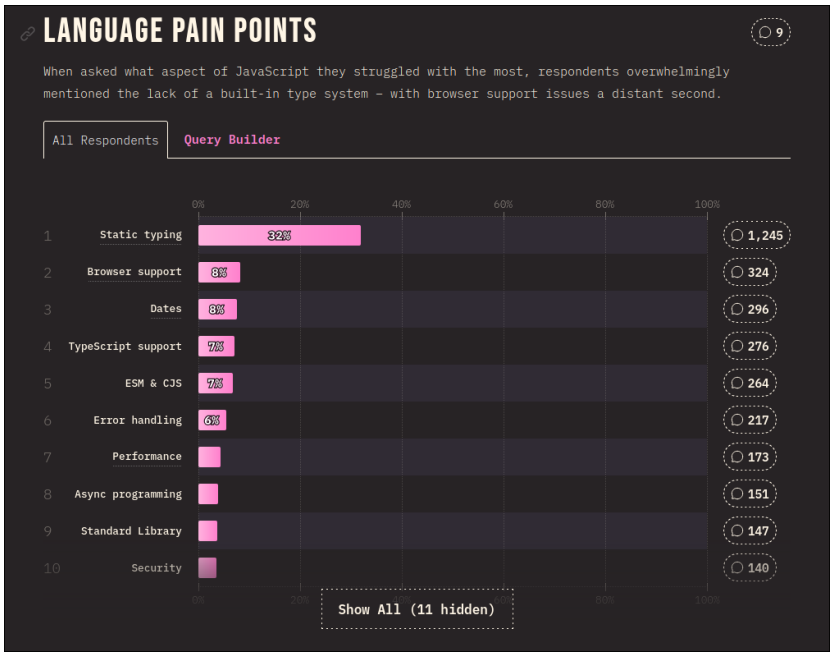


Abbildung 2: 32% der Befragten in der „State of JavaScript 2024“ Umfrage gaben an das (fehlende) statische Typisierung einer ihrer größten Probleme mit der Sprache sei [GB24]

1.3 Ziel der Arbeit

Was hat der/die Leser*in von dieser Arbeit? Was soll er/sie nach dem Lesen gelernt haben?

Im folgenden sollen Eigenschaften und Vor- wie auch Nachteile von Typsystemen dargestellt werden, um der Leserin

2 Systematische Literaturrecherche

Dieses Kapitel beschreibt in knapper Form das Vorgehen zur Literaturrecherche. Genaueres ist in [Kee+07] zu finden. Dieses Kapitel sollte ca. 20% vom Umfang der Arbeit ausmachen. Nachfolgend die Leitfragen für die Recherche aus der Einführungsveranstaltung:

Definition eines Rechercheziels - Was möchte nach der Recherche gelernt haben? Schützen Atemschutzmasken vor einer Corona-Infektion?

Definition relevanter Orte/Quellen für Literatur
Bild-Zeitung, Science, The Lancet, ...

Definition relevanter Suchbegriffe
Corona, COVID-19, Maske, Mask, FFP-2

Durchführung der Recherche mit Protokoll
Datum, Quelle, Suchbegriffe, Ergebnisliste

Bewertung der Ergebnisse
Aktualität, Glaubwürdigkeit, Beitrag zum Rechercheziel

3 Hauptteil

Eigentlicher Inhalt der Arbeit mit geeigneter Einteilung in Unterkapitel und ggf. Querbezügen zu anderen Themen des Seminars und der persönlichen Reflexion des Trends (siehe Folien).

Der Hauptteil sollte ca. 50% vom Umfang der Arbeit ausmachen

4 Zusammenfassung und Ausblick

Mit der Zusammenfassung und dem Ausblick wird die Arbeit geschlossen. Dieses Kapitel sollte ca. 10% vom Umfang der Arbeit ausmachen.

4.1 Zusammenfassung

Was sind die zentralen Ergebnisse/Erkenntnisse eurer Arbeit?

4.2 Kritische Reflektion

Betrachtet die Arbeit und euer eigenes Vorgehen kritisch. Was hätte man im Nachhinein anders/besser machen können.

4.3 Ausblick

Wohin geht die Reise? Was könnte man im Anschluss an eure Arbeit tun?

5 Anhang

Der Anhang dient als ergänzender Teil der Arbeit und zählt nicht zur Zeichenbeschränkung.

Literatur

[GB24] Sacha Greif und Eric Burel. *State of javascript 2024*. 2024. URL: https://2024.stateofjs.com/en-US/features/#language_pain_points.

[Kee+07] Staffs Keele u. a. *Guidelines for performing systematic literature reviews in software engineering*. Techn. Ber. Citeseer, 2007.

Suchergebnisse	**Notizen**
N. H. Madhavji und I. R. Wilson. „Cray Pascal“. In: *Proceedings of the 1982 SIGPLAN Symposium on Compiler Construction*. SIGPLAN '82. Boston, Massachusetts, USA: Association for Computing Machinery, 1982, S. 1–14. ISBN: 0897910745. DOI: 10.1145/800230.806975. URL: https://doi.org/10.1145/800230.806975	Notizen

Tabelle 1: Literaturliste zum Rechercheprotokoll