

Hochschule für Technik Stuttgart

Bachelor-Arbeit

Als Prüfungsleistung nach der SPO-2018
Ausgeführt für die Bachelor-Prüfung im
Sommersemester 2024

Analyse und Vergleich führender JavaScript-basierter Web Mapping Lösungen

Erstprüfer und Betreuer:
Zweitprüfer:

Prof. Dr.-Ing. Franz-Josef Behr
Prof. Dr. Jörg Homberger

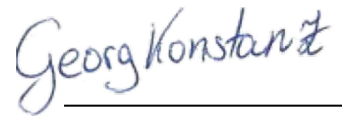
Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Gefertigt:

24.06.2024 Oberstenfeld

Ort, Datum



(Georg Konstanzt)

Gesehen:

Erstprüfer und Betreuer:

Datum

(Prof. Dr.-Ing. Behr)

Zweitprüfer:

Datum

(Prof. Dr. Homberger)

Kurzfassung

Im Juni 2024 verwenden weltweit etwa 23 Millionen Webseiten eine Web Mapping Lösung. Die Nutzung von Kartenlösungen erlebte einen deutlichen Anstieg aufgrund der Herausforderungen durch die Coronapandemie. Diese Thesis identifiziert, analysiert und vergleicht drei der führenden Web Mapping Lösungen anhand relevanter Metriken. Dabei werden auch die grundlegenden Konzepte und Technologien erläutert, die für das Verständnis dieser Lösungen bedeutend sind.

Schlüsselbegriffe:

Web Mapping, JavaScript, Google Maps, Leaflet, Mapbox, HTML, CSS, HTTP, API, GeoJSON, KML, WMS, WFS, OGC, Server, Karte, Kachel

Inhaltsverzeichnis

| | |
|--|------------|
| Eidesstattliche Erklärung | i |
| Kurzfassung | ii |
| Inhaltsverzeichnis | iii |
| Abbildungsverzeichnis | vi |
| Tabellenverzeichnis | ix |
| Abkürzungsverzeichnis | x |
| 1 Einleitung..... | 1 |
| 1.1 Motivation | 1 |
| 1.2 Aufbau der Arbeit | 4 |
| 1.3 Forschungsfragen | 4 |
| 1.4 Zielsetzung | 5 |
| 1.5 Forschungsstand | 6 |
| 2 Grundlagen..... | 7 |
| 2.1 Web Mapping..... | 7 |
| 2.1.1 Begriffsdefinition | 7 |
| 2.1.2 Entstehung und Historie von Web Mapping | 8 |
| 2.2 Webentwicklungstechnologien | 11 |
| 2.2.1 Hypertext Markup Language | 11 |
| 2.2.2 Cascading Style Sheets | 12 |
| 2.2.3 JavaScript..... | 15 |
| 2.2.4 Webserver | 18 |
| 2.2.5 Hypertext Transfer Protokoll..... | 18 |
| 2.2.6 Application Programming Interface | 19 |
| 2.3 Geodaten-Austauschformate | 21 |
| 2.3.1 Geo JavaScript Object Notation | 21 |
| 2.3.2 Keyhole Markup Language | 22 |
| 2.4 Kartendarstellung im Web | 23 |
| 2.4.1 Raster Tiles | 23 |
| 2.4.2 Vector Tiles | 24 |
| 2.5 Geodienste-Standards | 24 |
| 2.5.1 Web Map Service..... | 24 |
| 2.5.2 Web Feature Service | 25 |
| 3 Auswahl der Technologien und Methodik | 27 |
| 3.1 Identifikation führender Web Mapping Lösungen..... | 27 |
| 3.2 Testumgebung | 29 |
| 3.3 Performance | 30 |
| 3.3.1 Ladegeschwindigkeit..... | 30 |
| 3.3.2 Reaktionsgeschwindigkeit..... | 30 |
| 3.3.3 Ressourcenverbrauch | 30 |

| | | |
|----------|--|-----------|
| 3.4 | Funktionalitäten..... | 31 |
| 3.4.1 | Event-Handling | 31 |
| 3.4.2 | Unterstützung von Geodatenformaten | 31 |
| 3.4.3 | Unterstützung von Geodiensten..... | 31 |
| 3.4.4 | Erweiterbarkeit..... | 31 |
| 3.4.5 | Interoperabilität | 32 |
| 3.5 | Entwicklungsgeschwindigkeit..... | 32 |
| 3.5.1 | Dokumentation..... | 32 |
| 3.5.2 | Community-Aktivität | 32 |
| 3.6 | Datenschutz und Sicherheit | 32 |
| 3.7 | Kosten und Lizenzierung..... | 33 |
| 4 | Web Mapping Lösungen im Vergleich | 34 |
| 4.1 | Google Maps JavaScript API..... | 34 |
| 4.1.1 | Nutzung und Integration..... | 34 |
| 4.1.2 | Ladegeschwindigkeit..... | 34 |
| 4.1.3 | Reaktionsgeschwindigkeit..... | 43 |
| 4.1.4 | Ressourcenverbrauch | 45 |
| 4.1.5 | Funktionalitäten..... | 48 |
| 4.1.6 | Event-Handling | 50 |
| 4.1.7 | Unterstützung von Geodatenformaten | 51 |
| 4.1.8 | Unterstützung von Geodiensten | 52 |
| 4.1.9 | Erweiterbarkeit..... | 52 |
| 4.1.10 | Interoperabilität | 54 |
| 4.1.11 | Entwicklungsgeschwindigkeit..... | 55 |
| 4.1.12 | Dokumentation..... | 56 |
| 4.1.13 | Community-Aktivität | 56 |
| 4.1.14 | Kosten und Lizenzierung..... | 57 |
| 4.1.15 | Datenschutz und Sicherheit | 57 |
| 4.2 | Leaflet..... | 59 |
| 4.2.1 | Nutzung und Integration..... | 59 |
| 4.2.2 | Ladegeschwindigkeit..... | 59 |
| 4.2.3 | Reaktionsgeschwindigkeit..... | 64 |
| 4.2.4 | Ressourcenverbrauch | 66 |
| 4.2.5 | Funktionalitäten..... | 67 |
| 4.2.6 | Event-Handling | 68 |
| 4.2.7 | Unterstützung von Geodatenformaten | 68 |
| 4.2.8 | Unterstützung von Geodiensten | 69 |
| 4.2.9 | Erweiterbarkeit..... | 71 |
| 4.2.10 | Interoperabilität | 72 |
| 4.2.11 | Entwicklungsgeschwindigkeit..... | 72 |
| 4.2.12 | Dokumentation..... | 73 |
| 4.2.13 | Community- Aktivität | 73 |
| 4.2.14 | Kosten und Lizenzierung..... | 74 |
| 4.3 | Mapbox Graphic Library JavaScript API | 75 |
| 4.3.1 | Nutzung und Integration..... | 75 |
| 4.3.2 | Ladegeschwindigkeit..... | 75 |
| 4.3.3 | Reaktionsgeschwindigkeit..... | 81 |
| 4.3.4 | Ressourcenverbrauch | 81 |

| | | |
|---|--|-----------|
| 4.3.5 | Funktionalitäten..... | 82 |
| 4.3.6 | Event-Handling | 83 |
| 4.3.7 | Unterstützung von Geodatenformaten | 83 |
| 4.3.8 | Unterstützung von Geodiensten..... | 84 |
| 4.3.9 | Erweiterbarkeit..... | 84 |
| 4.3.10 | Interoperabilität | 85 |
| 4.3.11 | Entwicklungsgeschwindigkeit | 85 |
| 4.3.12 | Dokumentation..... | 85 |
| 4.3.13 | Community-Aktivität | 86 |
| 4.3.14 | Kosten und Lizenzierung..... | 86 |
| 5 | Ergebnisse..... | 87 |
| 5.1 | Diskussion der Ergebnisse..... | 87 |
| 6 | Zusammenfassung und Ausblick..... | 89 |
| 6.1 | Zusammenfassung..... | 89 |
| 6.2 | Kritische Bewertung | 89 |
| 6.3 | Ausblick | 90 |
| Anhang A. JMeter Testplan (Google Maps JS API) | | 91 |
| Anhang B. JMeter Testplan (OpenStreetMap) | | 92 |
| Anhang C. JMeter Testplan (Mapbox GL JS)..... | | 93 |
| Literaturverzeichnis | | 94 |

Abbildungsverzeichnis

| | |
|---|----|
| Abbildung 1: Popularität von Programmiersprachen (Stand: 2023) | 3 |
| Abbildung 2: Popularität von APIs nach Forks (Stand: 2023)..... | 3 |
| Abbildung 3: Zeitstrahl der Web Mapping Entstehung | 10 |
| Abbildung 4: Einfaches HTML-Dokument..... | 11 |
| Abbildung 5: Baumdiagramm zur hierarchischen Anordnung in einem HTML- Dokument. | 12 |
| Abbildung 6: Einfache CSS-Anweisungen..... | 13 |
| Abbildung 7: CSS-Medienabfragen (Breakpoints) | 14 |
| Abbildung 8: Anteil mobiler Endgeräte an allen Seitenaufrufen nach Regionen weltweit im Jahr 2022 | 14 |
| Abbildung 9: Beispiel für eine JavaScript Funktion zur DOM-Manipulation | 15 |
| Abbildung 10: Funktionsweise einer JavaScript Engine..... | 16 |
| Abbildung 11: Meistgenutzten Web-Frameworks unter Entwicklern weltweit (Stand: 2023) | 17 |
| Abbildung 12: Client-Server Architektur. Beispiel für HTTP-Anfrage / HTTP-Antwort... | 19 |
| Abbildung 13: Beispiel für eine API-Anfrage und Antwort (Wetterdaten im JSON- Format) | 20 |
| Abbildung 14: Beispiel für die Struktur einer GeoJSON-Datei. | 21 |
| Abbildung 15: Beispiel für die Struktur einer KML-Datei. | 22 |
| Abbildung 16: Beispiel für eine Kachel. | 23 |
| Abbildung 17: Beispiel für eine WMS-Anfrage und XML als Antwort..... | 25 |
| Abbildung 18: Beispiel für eine WFS-Anfrage und XML als Antwort..... | 26 |
| Abbildung 19: Netzwerkgeschwindigkeit..... | 29 |
| Abbildung 20: Marktanteile der führenden Browserfamilien an der Internetnutzung weltweit von Januar 2009 bis Mai 2024..... | 29 |
| Abbildung 21: Google Chrome Dev Tools Performance Insights (Google Maps JS API)..... | 35 |
| Abbildung 22: Google Chrome Performance-Analyse (Google Maps JS API)..... | 35 |
| Abbildung 23: Durchschnittliche Ladegeschwindigkeit..... | 37 |
| Abbildung 24: Karten-ID Zuordnung und Kachelumstellung | 38 |
| Abbildung 25: Fehler beim Laden der Vektorkarte (Google Maps JS API) | 38 |
| Abbildung 26: Messungen des Selenium Tests in Chart.js visualisiert (Google Maps JS API)..... | 41 |

| | |
|--|----|
| Abbildung 27: Antwortzeiten des Servers „maps.googleapis.com“ (JMeter) | 42 |
| Abbildung 28: Google Lighthouse Test (Google Maps JS API) | 43 |
| Abbildung 29: Messung der Reaktionsgeschwindigkeit per JS-Code (Google Maps JS API)44 | |
| Abbildung 30: Nachladedauer von Ressourcen ohne Cache (Google Maps JS API) ... | 44 |
| Abbildung 31: Nachladen von Ressourcen mit Cache (Google Maps JS API) | 45 |
| Abbildung 32: Arbeitsspeicher-Nutzung (Google Maps JS API)..... | 45 |
| Abbildung 33: GPU-Nutzung (Google Maps JS API) | 46 |
| Abbildung 34: CPU-Nutzung durch Skripte (Google Maps JS API)..... | 46 |
| Abbildung 35: RAM-Nutzung während des Selenium Test (Google Maps JS API)..... | 47 |
| Abbildung 36: CPU-Auslastung während Selenium Tests (Google Maps JS API)..... | 48 |
| Abbildung 37: Google Maps inklusive Marker (svg inline) und einer Polyline..... | 49 |
| Abbildung 38: Google Maps mit einem Polygon | 49 |
| Abbildung 39: Google Maps Infofenster..... | 50 |
| Abbildung 40: GeoJSON als Overlay Schicht (Google-Maps JS API)..... | 51 |
| Abbildung 41: KML-Overlay als Schicht (Google Maps JS API)..... | 51 |
| Abbildung 42: Stilldeklaration in der Cloud-Plattform | 53 |
| Abbildung 43: Google Chrome Performance-Analyse (Leaflet + OSM) | 62 |
| Abbildung 44: Messungen des Selenium Tests in Chart.js visualisiert (Leaflet + OSM)..... | 62 |
| Abbildung 45: 50 parallele HTTP-Anfragen an den Server von OSM | 63 |
| Abbildung 46: Google Lighthouse Test (Leaflet + OSM) | 64 |
| Abbildung 47: Konsolenausgabe im Browser beim Zoomen (Leaflet + OSM)..... | 64 |
| Abbildung 48: Konsolenausgabe beim Verschieben der Karte (Leaflet + OSM) | 65 |
| Abbildung 49: Konsolenausgabe des Seleniumtests zur Reaktionsgeschwindigkeit (Leaflet + OSM) | 65 |
| Abbildung 50: Arbeitsspeicher-Nutzung von Leaflet..... | 66 |
| Abbildung 51: Ram-Nutzung während des Selenium Tests (Leaflet + OSM) | 66 |
| Abbildung 52: Beispiel für eine Schichtverwaltung (Leaflet)..... | 67 |
| Abbildung 53: Marker und Infofenster in Leaflet | 67 |
| Abbildung 54: GeoJSON in Leaflet..... | 68 |
| Abbildung 55: KML in Leaflet..... | 69 |
| Abbildung 56: WMS-Antwort in Leaflet | 70 |
| Abbildung 57: WFS-Antwort in Leaflet..... | 71 |
| Abbildung 58: Konsolenausgabe JS-Messung der Ladegeschwindigkeit (Mapbox GL JS)..... | 76 |
| Abbildung 59: Google Chrome Dev Tools Performance Insights (Mapbox)..... | 76 |

| | |
|--|----|
| Abbildung 60: Google Chrome Performance-Analyse (Mapbox)..... | 77 |
| Abbildung 61: Messungen der Selenium Tests in Chart.js visualisiert (Mapbox GL JS)..... | 79 |
| Abbildung 62: Performance Analyse Google Chrome (Mapbox GL JS) | 79 |
| Abbildung 63: Antwortzeiten zu 50 parallele Anfragen an den Mapbox Server | 80 |
| Abbildung 64: Google Lighthouse Test (Mapbox GL JS)..... | 80 |
| Abbildung 65: Messung der Reaktionsgeschwindigkeit per JS-Code (Mapbox GL JS) | 81 |
| Abbildung 66: Arbeitsspeicher-Nutzung (Mapbox GL JS) | 81 |
| Abbildung 67: Grafikkarten-Nutzung (MapBox GL JS)..... | 82 |
| Abbildung 68: RAM-Nutzung während des Selenium-Tests (Mapbox GL JS) | 82 |

Tabellenverzeichnis

| | |
|---|----|
| Tabelle 1: JavaScript Engines der gängigsten Browser | 16 |
| Tabelle 2: Parameter für eine WMS-Anfrage | 24 |
| Tabelle 3: Parameter für eine WFS-Anfrage | 25 |
| Tabelle 4: Verwendete Hardware..... | 29 |
| Tabelle 5: Messungen per JavaScript-Code ohne Cache (Google Maps API)..... | 36 |
| Tabelle 6: Messungen per JavaScript-Code mit Cache (Google Maps API)..... | 36 |
| Tabelle 7: Messungen aus den Entwicklerkonsolen ohne Cache (Google Maps API) .. | 36 |
| Tabelle 8: Messungen aus den Entwicklerkonsolen mit Cache (Google Maps API) | 37 |
| Tabelle 9: Raster Kacheln vs. Vektor Kacheln (Google Maps JS API) | 39 |
| Tabelle 10: Interoperabilität von der Google Maps API mit JS-Frameworks..... | 54 |
| Tabelle 11: Community-Aktivität (Google Maps JS API)..... | 56 |
| Tabelle 12: Messungen per JavaScript-Code ohne Cache (Leaflet + OSM) | 60 |
| Tabelle 13: Messungen per JavaScript-Code mit Cache (Leaflet + OSM)..... | 60 |
| Tabelle 14: Messungen aus den Entwicklerkonsolen ohne Cache (Leaflet + OSM)..... | 60 |
| Tabelle 15: Messungen aus den Entwicklerkonsolen mit Cache (Leaflet + OSM)..... | 61 |
| Tabelle 16: Interoperabilität von der Google Maps API mit JS-Frameworks..... | 72 |
| Tabelle 17: Community-Aktivität (Leaflet) | 73 |
| Tabelle 18: Messungen per JavaScript-Code ohne Cache (Mapbox GL JS)..... | 77 |
| Tabelle 19: Messungen per JavaScript-Code mit Cache (Mapbox GL JS)..... | 77 |
| Tabelle 20: Messungen aus den Entwicklerkonsolen ohne Cache (Mapbox GL JS) | 78 |
| Tabelle 21: Messungen aus den Entwicklerkonsolen mit Cache (Mapbox GL JS) | 78 |
| Tabelle 22: Interoperabilität von Mapbox mit JS-Frameworks..... | 85 |
| Tabelle 23: Community-Aktivität (Mapbox) | 86 |
| Tabelle 24: Ergebnistabelle hinsichtlich der Geodatenformate- Unterstützung..... | 88 |
| Tabelle 25: Ergebnistabelle zur Unterstützung von Geodiensten | 88 |
| Tabelle 26: Ergebnistabelle zu Kosten und Lizenzierung | 88 |

Abkürzungsverzeichnis

| | |
|---------|--|
| INSPIRE | Infrastructure for Spatial Information in Europe |
| GIS | Geografisches Informationssystem |
| API | Application Programming Interface |
| HTML | Hypertext Markup Language |
| CSS | Cascading Style Sheets |
| WebGL | Web Graphics Library |
| KML | Keyhole Markup Language |
| GeoJSON | Geospatial JavaScript Object Notation |
| AJAX | Asynchronous JavaScript and XML |
| XML | Extensible Markup Language |
| WMS | Web Map Service |
| WFS | Web Feature Service |
| W3C | World Wide Web Consortium |
| DOM | Document Object Model |
| OGC | Open Geospatial Consortium |
| SVG | Scalable Vector Graphics |
| XHR | XMLHttpRequest |
| URL | Uniform Resource Locator |
| CDN | Content Delivery Network |
| CSV | Comma Separated Value |
| POI | Point of Interest |
| OSM | Open Street Map |
| JS | JavaScript |
| DSGVO | Datenschutz-Grundverordnung |

1 Einleitung

1.1 Motivation

Durch die Coronapandemie war weltweit ein Anstieg bei der Nutzung von Karten und Geodaten zu verzeichnen. Karten waren ein wichtiges Werkzeug, um die Pandemie einzudämmen, Hotspots zu identifizieren und Ressourcen möglichst effizient zu verteilen. Gemäß der Studie von Mordor Intelligence aus dem Erhebungsjahr 2021 wird der Markt für digitale Karten im Jahr 2024 auf 25,55 Milliarden US-Dollar prognostiziert und soll bis 2029 auf 47,88 Milliarden US-Dollar ansteigen.¹ Das Aufkommen von Schnittstellen sowie Standards für den Datenaustausch haben zu einer signifikanten Zunahme der Integration von Karten Webanwendungen geführt.² Die weltweite Digitalisierung ist ohne geografische Daten undenkbar. Die Beantwortung simpler Fragen, wie beispielsweise „Wo ist das nächste Restaurant?“, stellt dabei nur einen Aspekt dar. Vielmehr unterstützen Geodaten bei der Bewältigung komplexer globaler Probleme und politischer Entscheidungen.³ Sie ermöglichen fundierte Entscheidungen, verbessern Dienstleistungen und tragen maßgeblich zur Steigerung der Lebensqualität der Bürger bei. Die Bedeutung von Geodaten wird durch die Initiative der Europäischen Kommission INSPIRE untermauert.⁴ In diesem Kontext haben sich Web Mapping Lösungen als entscheidendes Instrument erwiesen, um geographische Daten in einer interaktiven und zugänglichen Form zu präsentieren. Infolgedessen hat sich eine Vielzahl von JavaScript Web Mapping APIs etabliert, die sich hinsichtlich ihrer jeweiligen Stärken, Schwächen und Einsatzmöglichkeiten unterscheiden.

¹ Vgl. Mordor Intelligence Marktgrößen- und Marktanteilsanalyse für digitale Karten – Wachstumstrends und Prognosen (2024 – 2029). <https://www.mordorintelligence.com/de/industry-reports/digital-map-market>

² Vgl. Henning S. Online Karten im Fokus: Praxisorientierte Entwicklung und Umsetzung. Wichmann / VDE Verlag. (2016), S. 5.

³ Vgl. Interministerieller Ausschuss für Geoinformationswesen. 4. Geo-Fortschrittsbericht der Bundesregierung. (2017), S. 3f.

⁴ Vgl. Rogall-Grothe. gis.Business Ausgabe 5-6/2014. S. 28f.

Die verfügbaren Lösungen weisen unterschiedliche Funktionen und Lizenzierungsmodelle auf und eignen sich daher für verschiedene Anwendungsfälle. Die rasante Entwicklung dieser Technologie in den vergangenen Jahren hat dazu geführt, dass sie heute in einer Vielzahl von Bereichen zum Einsatz kommt. Als Anwendungsbeispiele können die Navigation, das Umweltmanagement, die Logistik sowie der Tourismus genannt werden. JavaScript ist eine plattformunabhängige Programmiersprache, die sich hervorragend für die Entwicklung dynamischer und benutzerfreundlicher Webanwendungen eignet. Die Ergebnisse der Stack Overflow Entwickler-Umfrage des letzten Jahres belegen deutlich die anhaltende Beliebtheit von JavaScript (vgl. Abbildung 1).⁵ JavaScript basierte Web Mapping Lösungen bieten gegenüber traditionellen GIS-Anwendungen zahlreiche Vorteile und erfreuen sich daher wachsender Beliebtheit. Kartendienste sind eine der wichtigsten APIs für Mashups und werden daher häufig von Entwicklern genutzt (vgl. Abbildung 2).⁶ Die Analyse von BuiltWith^{7 8} verdeutlicht die Relevanz von JavaScript-Web-Mapping-APIs. Derzeit wird die Google Maps JavaScript API von mehr als fünf Millionen Websites verwendet, während die Leaflet JavaScript API von rund 500.000 Webseiten genutzt wird. Die Thematik steht auch deutlich im Zusammenhang mit dem Studiengang Informationslogistik, da JavaScript Web Mapping Lösungen die Datenverarbeitung und Informationsbereitstellung unterstützen, wodurch das logistische Ziel der richtigen Informationsbereitstellung zur richtigen Zeit am richtigen Ort erreicht wird.⁹

⁵ Stack Overflow. (2023). Stack Overflow Developer Survey 2023. <https://survey.stackoverflow.co/2023/>

⁶ Vgl. Alhosaini, Hadeel & Alharbi, Sultan & Wang, Xianzhi & Xu, Guandong. (2023). API Recommendation For Mashup Creation: A Comprehensive Survey. The Computer Journal. 10.1093/comjnl/bxad112. S. 3f.

⁷ BuiltWith Google Maps API Usage Statistics. <https://trends.builtwith.com/mapping/Google-Maps-API>

⁸ BuiltWith Leaflet JS usage Statistics. <https://trends.builtwith.com/mapping/Leaflet-JS>

⁹ Vgl. Heinrich Martin. Transport und Lagerlogistik Planung, Struktur, Steuerung und Kosten von Systemen der Intralogistik. Vieweg + Teubner Verlag / Springer Verlag (2011), S. 493.

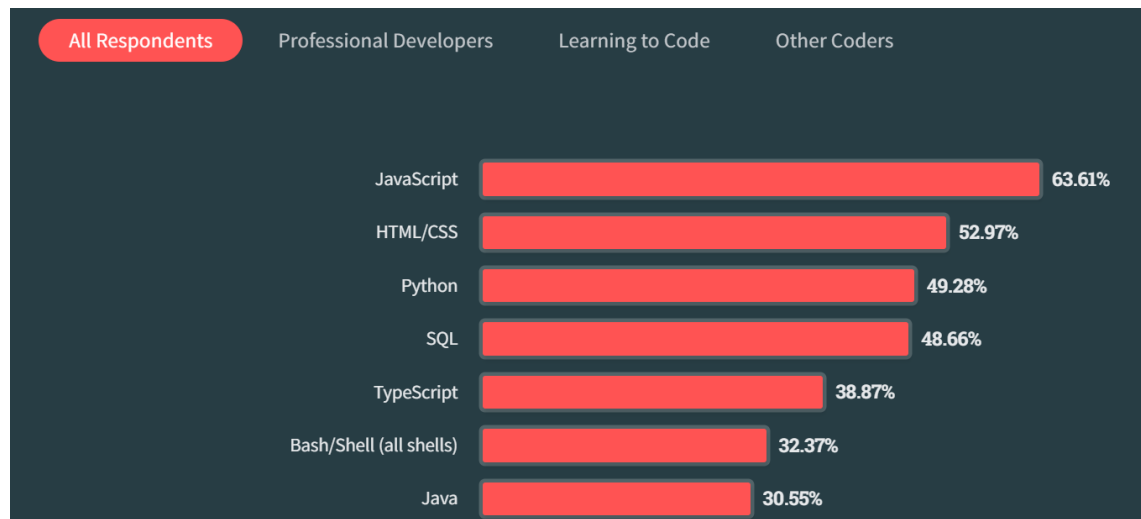


Abbildung 1: Popularität von Programmiersprachen (Stand: 2023)

Quelle: <https://survey.stackoverflow.co/2023/#technology-most-popular-technologies>

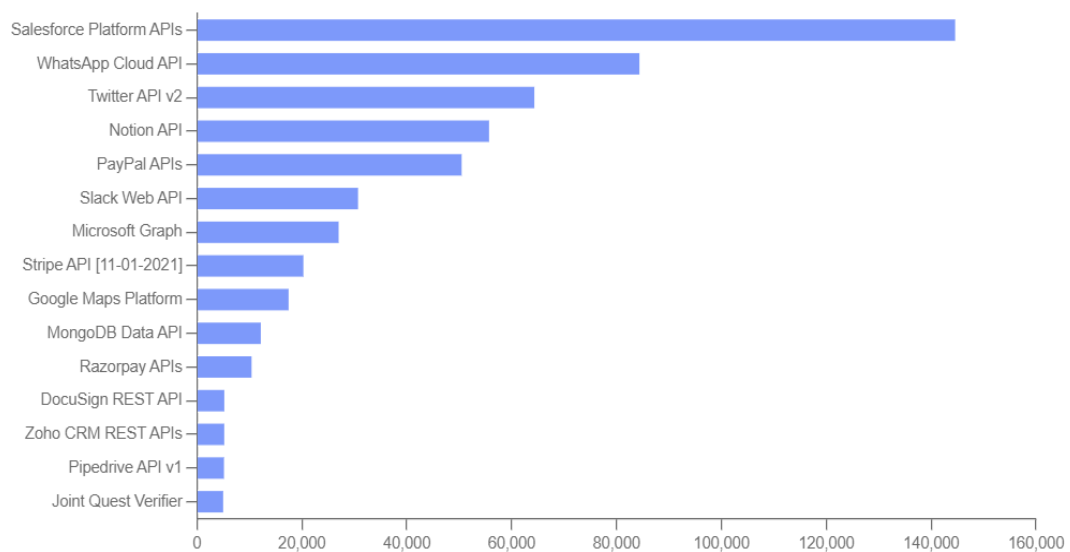


Abbildung 2: Popularität von APIs nach Forks (Stand: 2023)

Quelle: <https://www.postman.com/state-of-api/api-global-growth/#most-popular-apis>

1.2 Aufbau der Arbeit

Die vorliegende Arbeit ist in sechs Kapitel unterteilt. Das erste Kapitel umfasst die grundlegende Motivation, definiert die Zielsetzung und gibt Aufschluss über den aktuellen Forschungsstand. In Kapitel zwei werden die Grundlagen, auf denen die Arbeit aufbaut, näher erläutert, um ein fachliches Verständnis zu schaffen. In Kapitel drei werden die führenden JavaScript-basierten Web-Mapping Lösungen identifiziert sowie die Vergleichskriterien näher erörtert. Im vierten Kapitel erfolgt eine Untersuchung der einzelnen Mapping-Technologien sowie die Implementierung von Tests. Ein abschließender Vergleich basierend auf den Erkenntnissen der Arbeit sowie wird im fünften Kapitel präsentiert. Das sechste Kapitel beinhaltet eine Zusammenfassung, eine kritische Reflexion sowie einen Ausblick.

1.3 Forschungsfragen

- Welche Vorteile bieten Open-Source-Lösungen gegenüber proprietären Lösungen?
- Welche Einschränkungen haben proprietäre Lösungen?
- Wie gut werden Geodatenformate und Geodatendienste von den einzelnen Lösungen unterstützt?
- Welche Faktoren beeinflussen die Performance einer Web Mapping Lösung?
- Welche Lizenzierungsmodelle bieten die verschiedenen Lösungen?

1.4 Zielsetzung

Im Rahmen der vorliegenden Untersuchung wird das Ziel verfolgt, führende JavaScript-basierte Web Mapping-Lösungen zu identifizieren, welche in der Entwicklergemeinschaft eine hohe Verbreitung aufweisen. Die identifizierten Lösungen dienen anschließend als Basis für den Vergleich. Das primäre Bestreben dieser wissenschaftlichen Arbeit ist die Durchführung einer gründlichen Analyse. Im Zuge dieser werden folgende Schwerpunkte festgelegt, die als Vergleichskriterien herangezogen werden. Dabei werden die verfügbaren Funktionalitäten und Features jeder Lösung untersucht, um jeweilige Stärken und Schwächen hervorzuheben. Die Performance der Lösungen wird anhand verschiedener Metriken wie Ladezeit, Reaktionsfähigkeit und Ressourcenverbrauch gemessen, um eine fundierte Bewertung hinsichtlich der Leistungsfähigkeit zu treffen. Die Evaluierung der Entwicklungsgeschwindigkeit erfolgt durch die Analyse verfügbarer Dokumentationen sowie anhand der Implementierung von Beispielen. Hinzukommend wird in der Ausarbeitung die Unterstützung der Lösungen für verschiedene Geodatenformate und -dienste analysieren, um deren Flexibilität bei der Arbeit mit unterschiedlichen Datenquellen zu bewerten. Des Weiteren wird die Kostenstruktur sowie die Lizenzierung der einzelnen Lösungen untersucht, um die finanzielle Machbarkeit und etwaige Einschränkungen zu bewerten.

1.5 Forschungsstand

Einige wissenschaftliche Untersuchungen widmeten sich dem Vergleich der JavaScript basierter Web Mapping Lösungen. Die Mehrheit der Untersuchungen beschränkte sich auf eine Auswahl von drei oder lediglich Open-Source-Lösungen.

In der Abschlussarbeit aus dem Jahr 2015 von Farkas wurden JavaScript-Bibliotheken verglichen, wobei sich die Untersuchung auf OpenLayers 2, OpenLayers 3 und Mapbox JS beschränkte.¹⁰ Im Buch „Online-Karten im Fokus“ aus dem Jahre 2016 von Henning et al. verglichen im Kapitel 7 Morper-Busch und Weinke JavaScript Web-Mapping- APIs.¹¹ Eine jüngere Untersuchung, welche als wissenschaftlicher Artikel im International Journal of Geo-Information veröffentlichte wurde, beschäftigte sich im Jahre 2020 mit der Evaluierung der Performance von Web Mapping Bibliotheken anhand von einer Fallstudie mit Daten des argentinischen Lebensindexes. Zum Einsatz kamen Leaflet, Mapbox JS und OpenLayers.¹² Eine weitere wichtige Studie, veröffentlicht im Juli 2023, untersuchte die Analyse von kostenlosen Web Mapping Bibliotheken als Werkzeug zur Entscheidungsunterstützung für das World Heat Flow Database Project¹³. In dieser Arbeit wurden verschiedene Web Mapping Bibliotheken hinsichtlich ihrer Eignung für das Projekt evaluiert.¹⁴

¹⁰ Farkas, G. (2015). Comparison of Web Mapping Libraries for Building WebGIS Clients. ResearchGate. https://www.researchgate.net/publication/296049993_Comparison_of_Web_Mapping_Libraries_for_Building_WebGIS_Clients

¹¹ Morper-Busch, Weinke, Hennig, S. (2016). Online-Karten im Fokus: Praxisorientierte Entwicklung und Umsetzung. Wichmann / VDE- Verlag. <https://www.vde-verlag.de/buecher/537589/online-karten-im-fokus.html>

¹² Zunino, A.; Velázquez, G.; Celemin, J.P.; Mateos, C.; Hirsch, M.; Rodriguez, J.M. Evaluating the Performance of Three Popular Web Mapping Libraries: A Case Study Using Argentina's Life Quality Index. ISPRS Int. J. Geo-Inf. 2020, 9, 563. <https://doi.org/10.3390/ijgi9100563>

¹³ <http://heatflow.world/project>

¹⁴ Ott, N., & Mäs, S. (2023). Analysis of free web mapping libraries. Zenodo. <https://doi.org/10.5281/zenodo.8139086>

2 Grundlagen

In diesem Kapitel erfolgt eine Erläuterung der verwendeten Technologien für die Untersuchung, um ein grundlegendes Verständnis zu schaffen und Zusammenhänge hervorzuheben.

2.1 Web Mapping

2.1.1 Begriffsdefinition

Im Folgenden soll das Verständnis für den Terminus geschaffen werden. Anhand der Definitionen von Morper-Busch und Weinke (2016, S. 125) im Buch von S. Henning Online Karten im Fokus praxisorientierte Entwicklung und Umsetzung:

„Was ist eine interaktive Online-Karte? Internetbasierte Karten (Karten im kartografischen Sinn) stehen jedem Betrachter mit Internetzugang zur Verfügung. Interaktiv sind Karten dann, wenn sie im Gegensatz zu gedruckten Karten dem Nutzer Interaktionen ermöglichen, also beispielsweise den Maßstab oder den Ausschnitt der Karte (Zoom und Pan) zu verändern. Im Desktop-Bereich sind solche „Bildschirmkarten“ schon lange bekannt, allerdings wird spezielle GIS-Software (GIS = Geoinformationssysteme) benötigt, die in der Regel auch nur von Experten bedient werden kann. Ganz anders dagegen Online-Karten, die auf eine breite Zielgruppe auch ohne GIS-Fachwissen abzielen.“

Die Technologie hinter Web Mapping umfasst einige Komponenten auf die in den nachfolgenden Kapiteln näher eingegangen wird. Der Begriff Web-Mapping bezeichnet die Darstellung, Nutzung und Bearbeitung von Karten über das Internet. Zusammenfassend lässt sich sagen, dass interaktive Online-Karten den Zugang zu geografischen Informationen erleichtern und es Nutzern ermöglichen, Karten interaktiv zu erkunden, ohne spezielle Softwarekenntnisse zu benötigen.¹⁵

¹⁵ Vgl. Michael Dorman. Introduction to Web Mapping. CRC Press. (2020) S. XI.

2.1.2 Entstehung und Historie von Web Mapping

Der Begriff „Web Mapping“ lässt sich historisch betrachtet bis zu den Anfängen des Internets zurückverfolgen. Die erste interaktive Karte im Web wurde im Jahr 1993 im Rahmen des Xerox PARC Map Viewers Projekts in Perl geschrieben.¹⁶ Die nachfolgende Entwicklung von JavaScript im Jahr 1995 eröffnete neue Optionen. So konnten erstmals dynamische und interaktive Karten implementiert werden. Im darauffolgenden Jahr, 1996, erschien auch die erste JavaScript-basierte Web Mapping Bibliothek mit dem Namen „MapQuest“. Diese Bibliothek gilt bis heute als Pionier auf dem Gebiet und ermöglichte bereits zu dieser Zeit den Benutzern interaktive Funktionalitäten wie Zoomen und Verschieben.¹⁷ Anfang der 2000er Jahre wurde ein Paradigmenwechsel mit der Einführung von AJAX eingeläutet. Dadurch konnten Webseiten im Hintergrund mit den Servern kommunizieren bzw. Daten austauschen, ohne dabei neu geladen werden zu müssen. Aus dieser Technologie resultierte die Grundlage für heutige interaktive Web Mapping Lösungen.¹⁸ Ein weiterer bedeutender Meilenstein war die Festlegung der Standards Web Map Service und Web Feature Service zur Geodaten-Abfrage über das Web durch das OGC. Web Map Service erschien im Jahre 2000, dicht gefolgt von Web Feature Service im Jahre 2002.^{19 20 21} Im Jahr 2004 wurde OpenStreetMap gegründet, dabei handelt es sich um ein kollaboratives Projekt zur Erstellung einer freien Weltkarte.²² Im Jahr 2005 wurde Google Maps veröffentlicht. Diese Web-Mapping-Lösung bot nicht nur Satellitenbilder, sondern auch Straßenkarten und im weiteren Verlauf der Entwicklung ebenfalls Luftaufnahmen. Im Jahr 2007 ergänzte Google Maps sein Repertoire um Street View,

¹⁶ Vgl. Bert, Veenendaal. (2016). ERAS OF WEB MAPPING DEVELOPMENTS: PAST, PRESENT AND FUTURE. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. S. 248.

¹⁷ Vgl. Matt, Forrest (2021). A Brief History of Web Maps. <https://forrest.nyc/a-brief-history-of-web-maps/>

¹⁸ Vgl. Król, Karol. (2020). EVOLUTION OF ONLINE MAPPING: FROM WEB 1.0 TO WEB 6.0. Geomatics, Landmanagement and Landscape. S.36 – S. 46.

¹⁹ Vgl. Basics of the WMS specification | GEOG 585. <https://www.e-education.psu.edu/geog585/node/699>

²⁰ Vgl. WFS and editing vector data on the web | GEOG 585. <https://www.e-education.psu.edu/geog585/node/779>

²¹ Vgl. WFS Introduction OGC e-Learning 2.0.0 documentation. <https://opengeospatial.github.io/e-learning/wfs/text/basic-main.html>

²² Vgl. OpenStreetMap Deutschland - Die freie Wiki-Weltkarte. <https://www.openstreetmap.de/>

welches anfangs lediglich Aufnahmen von ausgewählten Städten der USA enthielt.²³ Projekte wie OpenLayers, das im Jahr 2006 initiiert wurde, und später Leaflet, das im Jahr 2011 veröffentlicht wurde, ermöglichten es Entwicklern, Mapping-Funktionalitäten in ihre Webanwendungen zu integrieren, ohne dabei auf proprietäre Lösungen angewiesen zu sein.²⁴ Die kontinuierliche Weiterentwicklung der Webtechnologien führte zu einer signifikanten Verbesserung der Leistungsfähigkeit und Darstellungsmöglichkeiten. Im Jahr 2010 wurde das Open-Source-Start-up Mapbox als Kartengrundlage für Non-Profit-Organisationen gegründet. Im Jahr 2020 erfolgte eine Änderung des Geschäftsmodells, sodass Mapbox sich zu einem proprietären Anbieter wandelte. In Konsequenz dessen kam es noch am nächsten Tag zur Gründung eines Community-Abspaltungsprojekt unter dem Namen MapLibre.²⁵

²³ Vgl. Reid, E. (2020). A look back at 15 years of mapping the world. Google.
<https://blog.google/products/maps/look-back-15-years-mapping-world/>

²⁴ Vgl. Matt, Forrest (2021). A Brief History of Web Maps.
<https://forrest.nyc/a-brief-history-of-web-maps/>

²⁵ Vgl. MapLibre: Mapbox GL open-source fork. (2021). MapTiler.
<https://www.maptiler.com/news/2021/01/maplibre-mapbox-gl-open-source-fork/>

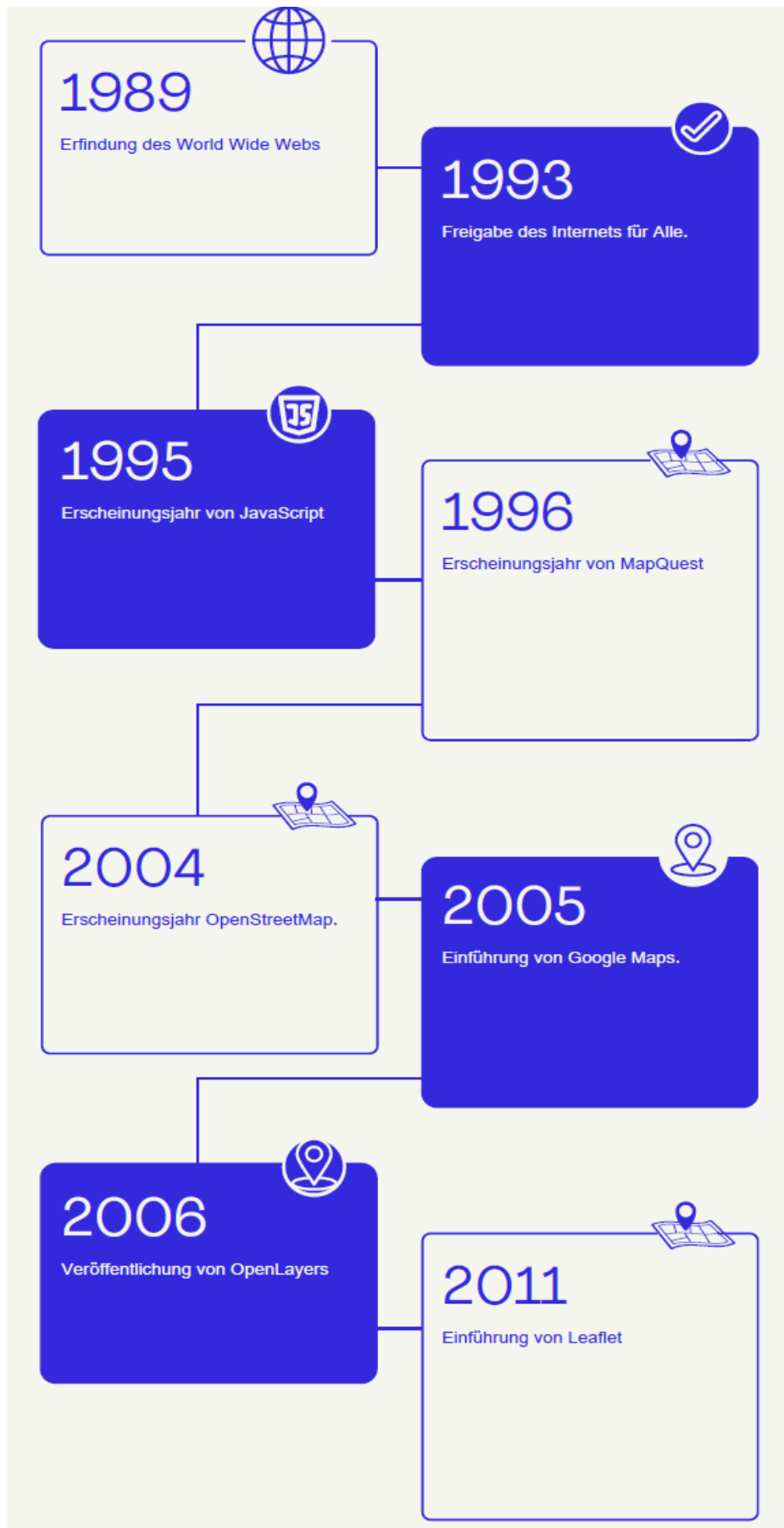


Abbildung 3: Zeitstrahl der Web Mapping Entstehung

Quelle: Eigene Darstellung (erstellt mit Canva)

2.2 Webentwicklungstechnologien

2.2.1 Hypertext Markup Language

HTML ist eine Standardauszeichnungssprache des W3C, welche auf XML basiert. Die aktuelle Version umfasst HTML 5.2, welche im Jahr 2017 veröffentlicht wurde. HTML kann als das Grundgerüst einer Webseite bezeichnet werden, da es den strukturellen Aufbau definiert sowie den Inhalt in verschiedene Elemente gliedert. Es besteht aus einer Sammlung von Tags (vgl. Abbildung 4), welche vom Browser interpretiert und angezeigt werden. Die Syntax der Tags ist durch umschließende spitze Klammern gekennzeichnet. Dabei besteht ein Paarbildungskonzept, das besagt, dass ein offener Tag in der Regel einen schließenden Tag aufweisen muss. Es gibt jedoch einige Ausnahmen, die als selbstschließend gelten. Ein Tag-Paar resultiert in einem HTML-Element, wobei ein öffnender Tag auch Attribute enthalten kann, die ein Element spezifizieren. (vgl. Abbildung 5). Die Elemente können durch ihre Attribute mittels Cascading Style Sheets-Regeln und JavaScript-Codes adressiert werden.^{26 27}

Im Kontext des Web Mappings wird HTML als Grundlage für die Integration von kartenspezifischen Elementen in Webseiten verwendet. Durch die Verwendung von HTML-Elementen können Kartencontainer erstellt werden, in denen JavaScript-basierte Mapping-Lösungen eingebettet werden. Des Weiteren werden die HTML-Elemente dazu verwendet um Karten-Layer, Steuerelemente, Marker und Popups zu definieren.

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Title der Seite</title>
  </head>
  <body>
    <p>Lorem Ipsum</p>
  </body>
</html>
```

Abbildung 4: Einfaches HTML-Dokument.

Quelle: Eigene Darstellung

²⁶ Vgl. Phillip Ackermann. Fullstack-Entwicklung. Das Handbuch für Webentwicklung. Rheinwerk Verlag. (2023), S. 55 ff.

²⁷ Vgl. Michael Dorman. Introduction to Web Mapping. CRC Press. (2020) S. 3 ff.

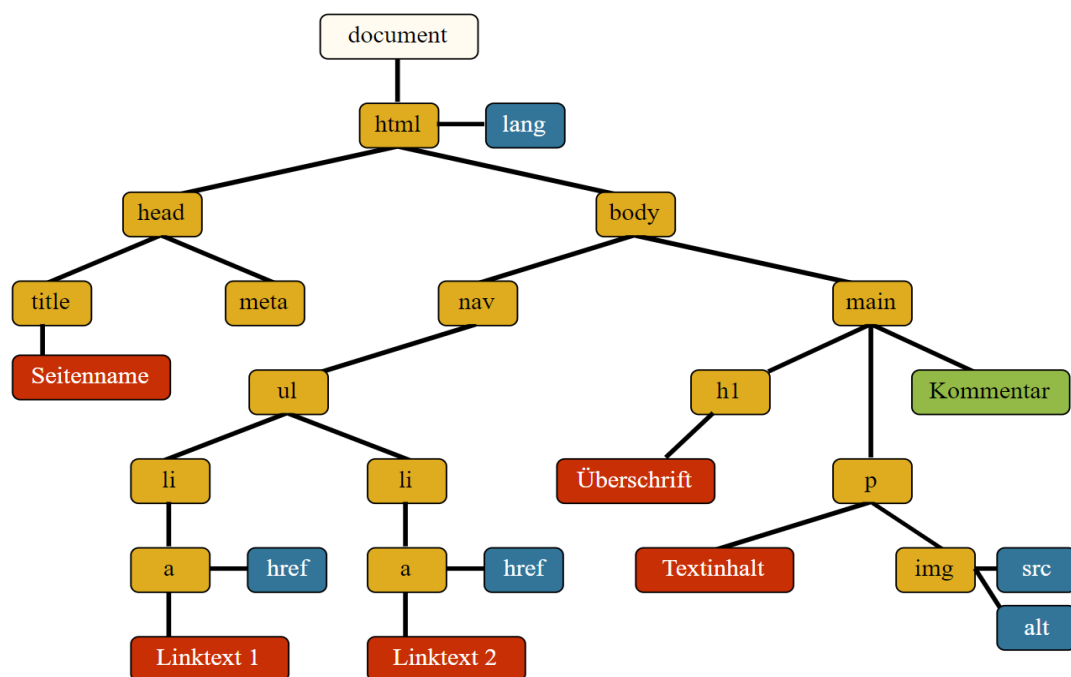


Abbildung 5: Baumdiagramm zur hierarchischen Anordnung in einem HTML-Dokument.

Quelle: <https://wiki.selfhtml.org/images/0/01/DOM-1.svg>

2.2.2 Cascading Style Sheets

CSS definiert die Darstellung von HTML-Elementen, durch festgelegte Regeln. Eine CSS-Regel ist dabei von geschweiften Klammern eingeschlossen, welche die HTML-Elemente durch eine Klasse oder eine ID selektieren. Dabei ist es möglich, Standardtags direkt zu selektieren, ohne die Angabe eines Punkts für die Klasse oder einer Raute für eine ID (vgl. Abbildung 6).^{28 29} Die Selektion von HTML-Elementen kann auch über viele weitere Attribute in CSS erfolgen.

In Bezug auf Web Mapping Lösungen kann CSS verwendet werden, um die visuelle Darstellung von Kartenelementen anzupassen. Dazu zählen beispielsweise die Anpassung der Symbole und der Beschriftungen einer Karte.

²⁸ Vgl. Phillip Ackermann. Fullstack-Entwicklung. Das Handbuch für Webentwicklung. Rheinwerk Verlag. (2023), S. 85 ff.

²⁹ Vgl. Michael Dorman. Introduction to Web Mapping. CRC Press. (2020) S. 33 ff.

Ein wichtiger Aspekt von CSS ist die Responsivität. Der Begriff impliziert die Anpassung einer Webseite an die Größe des verwendeten Endgeräts sowie der Ausrichtung. Dies wird durch die Verwendung von CSS Media Queries (zu Deutsch: Medienabfragen) ermöglicht. Hier zu kommen sogenannte Breakpoints zum Einsatz welche die Auflösung verschiedener Geräte definieren (vgl. Abbildung 7). Dadurch wird sichergestellt, dass beispielsweise die Kartenansicht auf Desktop-Computern und mobilen Endgeräten gleichermaßen adäquat funktioniert und gut lesbar ist.³⁰ Somit ermöglicht man eine benutzerfreundliche Erfahrung, unabhängig vom verwendeten Gerät. In der Webentwicklung wird das Prinzip „mobile first“ als grundlegend erachtet.

³¹ Die erhobene Statistik aus dem Jahre 2022 von Statista, soll die Relevanz der Anpassung für mobile Endgeräte unterstreichen (vgl. Abbildung 8). Die heutige Webentwicklungstechnologie weist einige CSS-Erweiterungen auf, welche die Entwicklung beschleunigen können, darunter beispielsweise Sass³² welches CSS um Variablen und Schleifen erweitert. Zudem gibt es unzählige CSS-Frameworks welche eine breite Palette an vordefinierten Regeln mit sich bringen und die Entwicklung signifikant beschleunigen, darunter Bootstrap³³, Materialize³⁴ und Tailwind³⁵

```
body {  
  font-size: 16px;  
  color: white;  
  background-color: blue;  
}
```

Abbildung 6: Einfache CSS-Anweisungen

Quelle: Eigene Darstellung

³⁰ Vgl. Horbiński T, Cybulski P, Medyńska-Gulij B. Web Map Effectiveness in the Responsive Context of the Graphical User Interface. *ISPRS International Journal of Geo-Information*. 2021; 10(3):134. <https://doi.org/10.3390/ijgi10030134>

³¹ Prakash, Bhanu. (2020). Retrofitting Mobile First Design, Responsive Design: Driving Factors, Approach, Best Practices and Design Considerations. *Current Trends in Computer Sciences & Applications*. S. 167ff.

³² <https://sass-lang.com/>

³³ <https://getbootstrap.com/>

³⁴ <https://materializecss.com/>

³⁵ <https://tailwindcss.com/>

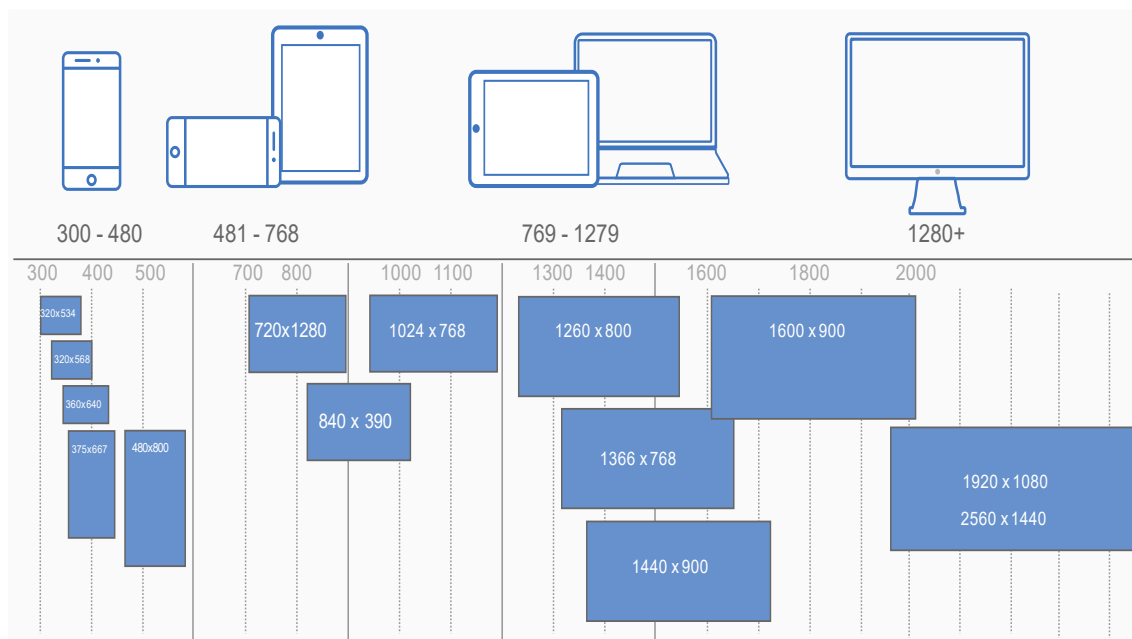


Abbildung 7: CSS-Medienabfragen (Breakpoints)

Quelle: <https://www.mediaevent.de/css/svg/device-class.svg>

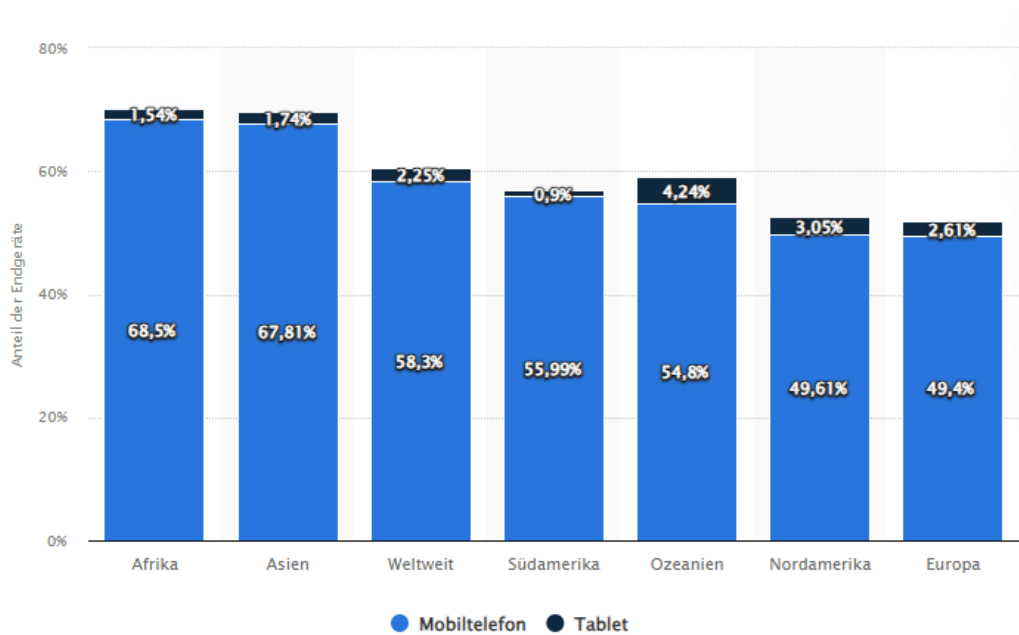


Abbildung 8: Anteil mobiler Endgeräte an allen Seitenaufrufen nach Regionen weltweit im Jahr 2022

Quelle: <https://de.statista.com/statistik/daten/studie/217457/umfrage/anteil-mobiler-endgeraete-an-allen-seitenaufrufen-weltweit/>

2.2.3 JavaScript

JavaScript ist eine clientseitige Skriptsprache, welche auf dem ECMAScript Standard basiert. Der Code wird zur Browserlaufzeit interpretiert und ausgeführt. Die Verwendung von JavaScript ermöglicht den Zugriff und die Manipulation von Elementen innerhalb des DOMs (vgl. Abbildung 9). Dies kann zur Erstellung dynamischer Inhalte oder zur Reaktion auf Benutzerinteraktionen genutzt werden. JavaScript unterstützt folgende Programmierkonzepte: Variablen, Datentypen, Schleifen, Operatoren, Funktionen und Bedingungen. Die Deklaration von Variablen erfolgt durch die Verwendung von Schlüsselwörtern wie „var“, „let“ oder „const“.^{36 37}

```
function changeColor() {  
    var my_element = document.getElementById("examplediv");  
  
    my_element.style.backgroundColor = "blue";  
}  
  
changeColor();
```

Abbildung 9: Beispiel für eine JavaScript Funktion zur DOM-Manipulation

Quelle: Eigene Darstellung

Der JavaScript-Code weist in unterschiedlichen Browsern möglicherweise unterschiedliche Ausführungszeiten auf. Dies ist auf die verschiedenen JavaScript Engines der Browser zurückzuführen (vgl. Tabelle 1 und Abbildung 10). Ein essenzieller Bestandteil von JavaScript ist AJAX, dessen Funktion der asynchrone Datenaustausch zwischen Client und Server ist³⁸. Diese Technologie wird im Web Mapping für das Nachladen von Ressourcen bei Benutzerinteraktionen genutzt, wie beispielsweise die Ladeprozedur von Kacheln beim Zoomen oder Verschieben der Karte.

³⁶ Vgl. Phillip Ackermann. Fullstack-Entwicklung. Das Handbuch für Webentwicklung. Rheinwerk Verlag. (2023), S. 145 ff.

³⁷ Vgl. Michael Dorman. Introduction to Web Mapping. CRC Press. (2020) S. 85 ff.

³⁸ Vgl. Phillip Ackermann. Fullstack-Entwicklung. Das Handbuch für Webentwicklung. Rheinwerk Verlag. (2023), S. 233 ff.

Des Weiteren existieren JavaScript-Bibliotheken wie beispielsweise JQuery ³⁹, die über eine größere Methodenvielfalt sowie eine kürzere Syntax verfügen. Die Bibliothek Three.js ⁴⁰ deckt ein breites Spektrum an Methoden zur Erstellung und Manipulation von 3D-Grafiken ab.

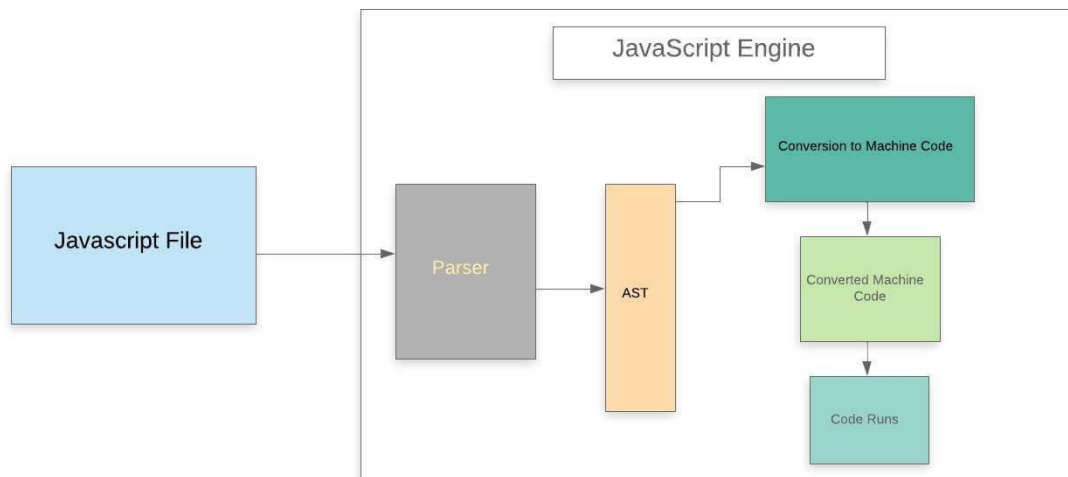


Abbildung 10: Funktionsweise einer JavaScript Engine

Quelle: <https://static.javatpoint.com/javascriptpages/images/how-does-javascript-work2.jpg>

| | |
|-------------------|--|
| JavaScript-Engine | Browser |
| V8 | Google Chrome, Opera, Edge (ab Version 79) |
| SpiderMonkey | Mozilla Firefox |
| JavaScriptCore | Safari |

Tabelle 1: JavaScript Engines der gängigsten Browser

Quelle: <https://www.javatpoint.com/how-does-javascript-work>

Heutige Webseiten umfassen längst nicht mehr statische Inhalte sondern sind dynamische Webanwendungen, dies ist der Verwendung von JavaScript Frameworks zu verdanken. Darunter React, Angular, Vue.js (vgl. Abbildung 11) und etliche mehr. Durch Node.js konnte sich ein weiteres Einsatzgebiet für JavaScript etablieren, nämlich die serverseitige Programmierung. Bei Node.js handelt es sich um eine Laufzeitumgebung, die JavaScript-Code außerhalb des Browser ausführen kann.

³⁹ <https://jquery.com/>

⁴⁰ <https://threejs.org/>

Somit können Webanwendungen implementiert werden deren Backend vollständig in JavaScript geschrieben ist.⁴¹

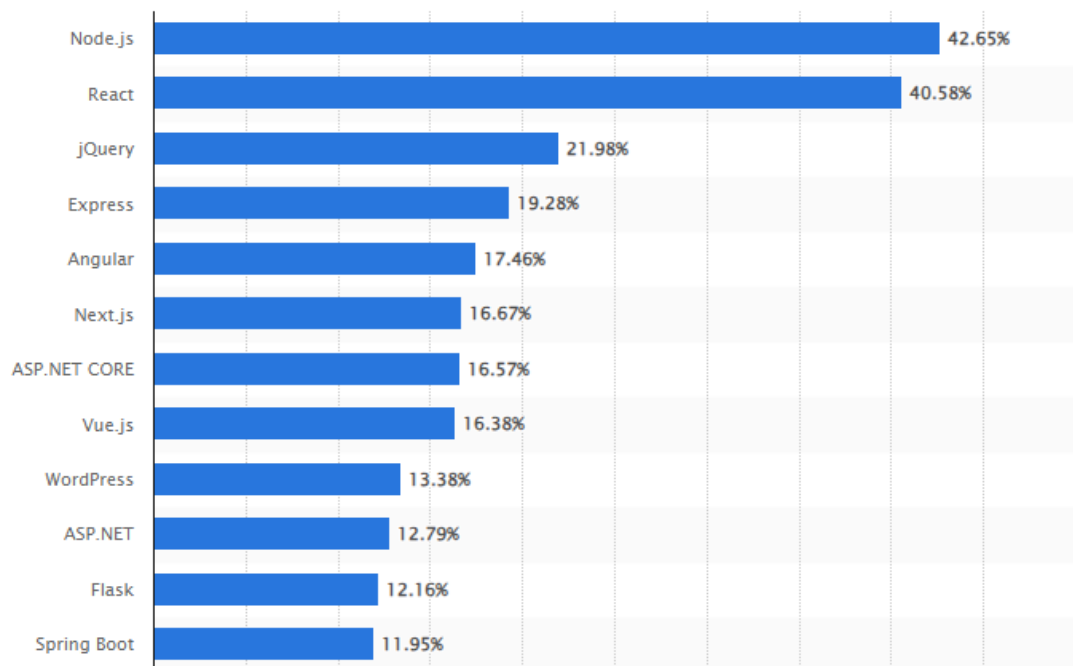


Abbildung 11: Meistgenutzten Web-Frameworks unter Entwicklern weltweit (Stand: 2023)

Quelle: <https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/>

In Web Mapping Lösungen wird JavaScript verwendet, um folgende Funktionalität von Kartenanwendungen zu implementieren:

- Kartensteuerung
- Datenvisualisierung
- Interaktive Funktionen
- Nachladen von Inhalten (AJAX)

⁴¹ Vgl. Phillip Ackermann. Fullstack-Entwicklung. Das Handbuch für Webentwicklung. Rheinwerk Verlag. (2023), S. 435 ff.

2.2.4 Webserver

Der Terminus „Webserver“ bezeichnet das Zusammenspiel von Hard- und Software, die eine gemeinsame Funktion erfüllen. Diese besteht in der Entgegennahme von HTTP-Anfragen von Clients (Browser) sowie der darauffolgenden Antwort, welche die Bereitstellung von Webinhalten wie HTML-Seiten, CSS-Dateien und JavaScript-Code umfasst.⁴²

2.2.5 Hypertext Transfer Protokoll

HTTP-Anfrage: Die Kommunikation mit einem Webserver wird durch HTTP-Anfragen bestimmt. Die Methoden, die zum Einsatz kommen, umfassen GET, POST, PUT, DELETE, CONNECT, TRACE, HEAD sowie OPTIONS. Die GET-Methode dient der Bereitstellung von Inhalten, die vom Client angefragt werden. Die POST-Methode bewirkt eine entgegengesetzte Aktion, nämlich das Versenden zum Server hin (vgl. Abbildung 12)⁴³

HTTP-Antwort: Eine HTTP-Antwort liefert eine numerische Information zurück, welche über den Status der Anfrage Auskunft gibt.

Das Spektrum der Statuscodes ist in fünf Ebenen unterteilt:

100-199 = Informationelle Antwort

200-299 = Erfolgreiche Antwort

300-399 = Weiterleitungen

400-499 = Clientseitiger Fehler

500-599 = Serverseitiger Fehler⁴⁴

⁴² Vgl. Michael Dorman. Introduction to Web Mapping. CRC Press. (2020) S. 118 ff.

⁴³ Vgl. Phillip Ackermann. Fullstack-Entwicklung. Das Handbuch für Webentwicklung. Rheinwerk Verlag. (2023), S. 174 ff.

⁴⁴ Vgl. HTTP response status codes - HTTP | MDN. (2023). MDN Web Docs.
<https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>

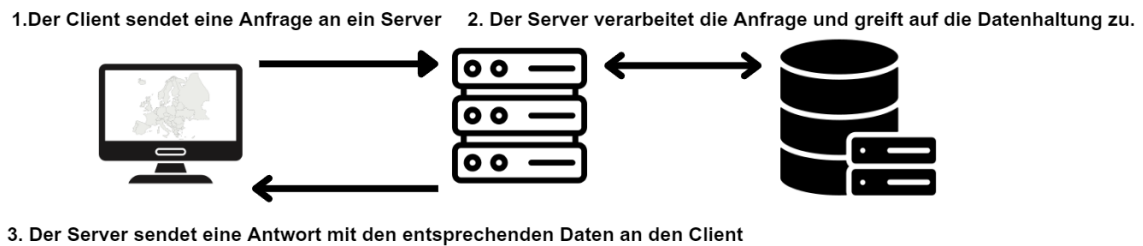


Abbildung 12: Client-Server Architektur. Beispiel für HTTP-Anfrage / HTTP-Antwort.

Quelle: Eigene Darstellung (erstellt mit Canva)

2.2.6 Application Programming Interface

Application Programming Interface bezeichnet eine Programmierschnittstelle, welche die Kommunikation zwischen zwei verschiedenen Softwareanwendungen über Protokolle, wie beispielweise http, ermöglicht. Des Weiteren wird durch die Verwendung von APIs die Datenübertragung unterstützt, sodass eine Anwendung auf Funktionen oder die Datenbank der anderen zugreifen kann (vgl. Abbildung 13). Durch APIs wird die Wiederverwendbarkeit von Code gefördert und die Integration verschiedener Systeme erleichtert.⁴⁵

Eine passende Definition um den Terminus sowie den Kontext zu erläutern findet sich im Buch Fullstack-Entwicklung von Phillip Ackermann (2023, S.223):

„Unter einer API (Application Programming Interface) versteht man eine Programmierschnittstelle, die verschiedene Objekte und Methoden bereitstellt, die wiederum in Implementierungen (sprich den konkreten Umsetzungen des jeweiligen Interfaces) vorhanden sein müssen. Implementierungen können sich untereinander dabei unterscheiden, wichtig ist, dass die Schnittstelle eingehalten wird.“

⁴⁵ Vgl. What is an API? Postman API Platform. <https://www.postman.com/what-is-an-api/>

Call

```
http://api.weatherapi.com/v1/current.json?key=3f5ce69970f643169a9150148241606&q=Stuttgart&aql=no
```

Response Code

```
200
```

Response Body

```
{
  "location": {
    "name": "Stuttgart",
    "region": "Baden-Wuerttemberg",
    "country": "Germany",
    "lat": 48.77,
    "lon": 9.18,
    "tz_id": "Europe/Berlin",
    "localtime_epoch": 1718706250,
    "localtime": "2024-06-18 12:24"
  },
  "current": {
    "last_updated_epoch": 1718705700,
    "last_updated": "2024-06-18 12:15",
    "temp_c": 26.3,
    "temp_f": 79.3,
    "is_day": 1,
    "condition": {
      "text": "Sunny",
      "icon": "///cdn.weatherapi.com/weather/64x64/day/113.png",
      "code": 1000
    },
    "wind_mph": 3.8,
    "wind_kph": 6.1,
    "wind_degree": 60,
    "wind_dir": "ENE",
    "pressure_mb": 1014.0,
    "pressure_in": 29.94,
    "precip_mm": 0.01,
    "precip_in": 0.0,
    "humidity": 58,
    "cloud": 0,
    "feelslike_c": 27.1,
    "feelslike_f": 80.8,
    "windchill_c": 26.3,
    "windchill_f": 79.4,
    "heatindex_c": 27.2,
    "heatindex_f": 80.9,
    "dewpoint_c": 16.2,
    "dewpoint_f": 61.1,
    "vis_km": 10.0,
    "vis_miles": 6.0,
    "uv": 6.0,
    "gust_mph": 8.3,
    "gust_kph": 13.3
  }
}
```

Abbildung 13: Beispiel für eine API-Anfrage und Antwort (Wetterdaten im JSON-Format)

Quelle: Eigene Darstellung

Datenquelle: <https://www.weatherapi.com/>

2.3 Geodaten-Austauschformate

2.3.1 Geo JavaScript Object Notation

GeoJSON ist ein Datenformat für den Austausch geografischer Daten im Web. Es zeichnet sich durch eine einfache Syntax aus und ist leichtgewichtig. Dieses Datenformat wird im Klartext dargestellt und ist somit leicht zu lesen und zu verstehen. Da GeoJSON auf dem Grundkonzept von JSON (JavaScript Object Notation) basiert, lässt es sich optimal mit der Programmiersprache JavaScript verarbeiten.⁴⁶ Die Syntax besteht aus geschweiften Klammern, welche Objekte repräsentieren, sowie eckigen Klammern, die Arrays darstellen. Innerhalb dieser sind Attribute enthalten, welche die Informationen gemäß dem Schlüssel-Wert-Paar-Prinzip spezifizieren. In der Regel sind in einer Feature Collection Features enthalten, die Koordinaten aufweisen, welche anschließend als geometrische Form definiert werden (vgl. Abbildung 14).

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {},
      "geometry": {
        "coordinates": [
          [
            [
              9.170825868747556,
              48.78125115042761
            ],
            [
              9.170825868747556,
              48.779299269669366
            ],
            [
              9.174053552137508,
              48.779299269669366
            ],
            [
              9.174053552137508,
              48.78125115042761
            ],
            [
              9.170825868747556,
              48.78125115042761
            ]
          ]
        ]
      }
    },
    {
      "type": "Polygon"
    }
  ]
}
```

Abbildung 14: Beispiel für die Struktur einer GeoJSON-Datei.

Quelle: Eigene Darstellung

⁴⁶ Michael Dorman. Introduction to Web Mapping. CRC Press. (2020) S. 169 ff.

2.3.2 Keyhole Markup Language

Im Jahr 2004 wurde KML von Google aufgekauft und in der Folgezeit weiterentwickelt, bis es schließlich im Jahr 2008 vom OGC als Standard anerkannt wurde.⁴⁷ Bei KML handelt es sich um ein XML-basiertes Datenformat, welches für den Transport und die Speicherung von Geodaten eingesetzt wird. Die XML-Grundstruktur wird dabei verwendet, um geografische Merkmale zu beschreiben (vgl. Abbildung 15). Das Wurzelement KML schließt ein Placemark-Element ein, welches einen Namen, eine Beschreibung und Koordinaten enthalten kann. Das Element „Placemark“ kann dabei auch geografische Merkmale wie „Point“, „LineString“ und „Polygon“ enthalten. Diese Elemente können dazu genutzt werden, um bestimmte Gebiete auf einer Karte hervorzuheben.⁴⁸

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.2">
  <Placemark>
    <name>Stuttgart</name>
    <description></description>
    <Point>
      <coordinates>9.179248809814455,48.77818318543398,0</coordinates>
    </Point>
  </Placemark>
</kml>
```

Abbildung 15: Beispiel für die Struktur einer KML-Datei.

Quelle: Eigene Darstellung

⁴⁷ Vgl. Open Geospatial Consortium. (2023). KML - Open Geospatial Consortium.
<https://www.ogc.org/standard/kml/>

⁴⁸ Vgl. KML-Anleitung. (o. D.). Google For Developers.
https://developers.google.com/kml/documentation/kml_tut?hl=de

2.4 Kartendarstellung im Web

2.4.1 Raster Tiles

Raster Tiles stellen eine grundlegende Methode zur Darstellung von Karten im Web dar. Dadurch ist es möglich, große Karten effizient zu laden. Das Prinzip basiert auf der Unterteilung einer großen Karte in viele quadratische Bilder, sogenannte „Tiles“ (zu Deutsch Kacheln) diese werden in einem Raster angeordnet. Es werden nur die Kacheln geladen, die für den aktuell dargestellten Kartenausschnitt benötigt werden, wobei das Ganze vom Zoom-Level anhängig ist. Das bedeutet, dass sich die Anzahl der Kacheln bei jedem Zoom-Level vervielfacht. Die Identifikation der Kacheln erfolgt über die Parameter z, x und y. Dabei stehen z, x und y für den jeweiligen Zoom-Level sowie die X- und Y-Koordinate im Raster. Das Format der Kacheln umfasst in der Regel PNG oder JPEG (Vgl. Abbildung 16). Diese können zudem clientseitig gecached werden, was die Ladezeit optimiert. Die Zuordnung des Zoom-Levels so wie X und Y-Koordinate kann je nach Anbieter bzw. System variieren.^{49 50 51 52}



Abbildung 16: Beispiel für eine Kachel.

Quelle: <https://tile.openstreetmap.org/7/63/42.png>

⁴⁹ Vgl. Raster tiles - OpenStreetMap Wiki. (o. D.). Abgerufen von: https://wiki.openstreetmap.org/wiki/Raster_tiles

⁵⁰ Vgl. Martin Elias. Raster vs Vector Map Tiles: What Is the Difference Between the Two Data Types? 2023. Abgerufen von: <https://documentation.maptiler.com/hc/en-us/articles/4411234458385-Raster-vs-Vector-Map-Tiles-What-Is-the-Difference-Between-the-Two-Data-Types>

⁵¹ Vgl. GEOFABRIK // Raster Tiles. (o. D.). <https://www.geofabrik.de/maps/rastertiles.html>

⁵² Vgl. ODonohue, D. (2022). RASTER TILES – What you need to know - mapscaping.com. <https://mapscaping.com/raster-tiles-what-you-need-to-know/>

2.4.2 Vector Tiles

Die Vector Tiles enthalten keine Bilder, sondern geographische Daten in einem vektorbasierten Format. Diese Rohdaten werden clientseitig im Browser gerendert. Dies impliziert, dass die Ladegeschwindigkeit bei dieser Methode signifikant von der Hardware des Endnutzers abhängig ist. Ähnlich wie Raster Tiles sind Vector Tiles in Zoom-Levels und Tile-Koordinaten organisiert. Bei jedem Zoom-Level wird die Welt in ein Raster von Kacheln unterteilt. Die Anzahl der Kacheln steigt mit dem Zoom-Level exponentiell an. Die Hauptunterschiede zu Raster Tiles liegen in der Dateigröße, dem Rendering sowie der Auflösung.^{53 54 55}

2.5 Geodienste-Standards

2.5.1 Web Map Service

Der Web Map Service (WMS) ist ein vom Open Geospatial Consortium (OGC) festgelegter Standard für die Anfrage von Kartenbildern über das Internet. Die Anfrage erfolgt dabei über eine URL an den WMS-Server. Durch das Hinzufügen eines Fragezeichens zur WMS-Serveradresse, können die entsprechenden Anfrageparameter angehängt werden (vgl. Tabelle 2 und Abbildung 17).⁵⁶

| Parameter | Funktion |
|-------------------------|---|
| Service=WMS | Signalisiert dem Server um welchen Dienstyp es sich handelt |
| Request=GetMap | Forderung eines Kartenbildes |
| Request=GetCapabilities | Liefert Informationen über Inhalte des WMS-Servers |

Tabelle 2: Parameter für eine WMS-Anfrage

Quelle: Eigene Darstellung

⁵³ Vgl. Vector tiles - OpenStreetMap Wiki. (o. D.). https://wiki.openstreetmap.org/wiki/Vector_tiles

⁵⁴ Vgl. WEB VEKTOR - basemap.de. (o. D.). <https://basemap.de/web-vektor/>

⁵⁵ Vgl. What are vector tiles and why you should care. (2019). MapTiler. <https://www.maptiler.com/news/2019/02/what-are-vector-tiles-and-why-you-should-care/>

⁵⁶ Vgl. WMS - Introduction — OGC e-Learning 2.0.0 documentation. (o. D.). <https://opengeospatial.github.io/e-learning/wms/text/basic-main.html>

```

index_leaflet_wms.html:125
https://owsproxy.lgl-bw.de/owsproxy/ows/WMS_LGL-BW_ATKIS_BasisDLM_VerwGr_E?service=WMS&request=GetCapabilities
index_leaflet_wms.html:126
#document (file:///C:/Users/Anwender/thesis_webapp/index_leaflet_wms.html)
<WMS_Capabilities xmlns="http://www.opengis.net/wms" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wms="http://www.opengis.net/wms" version="1.3.0" xsi:schemaLocation="http://www.opengis.net/wms http://schem
as.opengis.net/wms/1.3.0/capabilities_1_3_0.xsd http://inspire.ec.europa.eu/schemas/inspire_vs/1.0 http://inspire.
ec.europa.eu/schemas/inspire_vs/1.0/inspire_vs.xsd">
  <Service> ☹ </Service>
  <Capability> ☹ </Capability>
</WMS_Capabilities>

```

Abbildung 17: Beispiel für eine WMS-Anfrage und XML als Antwort.

Quelle: Eigene Darstellung.

Datenquelle: LGL, www.lgl-bw.de, dl-de/by-2-0.

2.5.2 Web Feature Service

Bei Web Feature Service (WFS) handelt es sich ebenfalls um einen Standard des OGC. Dadurch wird der Zugriff auf geografische Features über das Internet ermöglicht. Im Gegensatz zu WMS, liefert WFS keine Kartenbilder, sondern die tatsächlichen geografischen Daten im Vektorformat. Der Begriff „Features“ bezeichnet in diesem Kontext eine Abstraktion der realen Welt (vgl. Tabelle 3 und Abbildung 18)..⁵⁷

| Parameter | Funktion |
|-------------------------|--|
| Service=WFS | Signalisiert dem Server um welchen Diensttyp es sich handelt |
| Request=GetFeature | Forderung der Feature |
| Request=GetCapabilities | Liefert Informationen über Inhalte des WFS-Servers |

Tabelle 3: Parameter für eine WFS-Anfrage

Quelle: Eigene Darstellung

⁵⁷ Vgl. WFS - Introduction — OGC e-Learning 2.0.0 documentation. (o. D.).
<https://opengeospatial.github.io/e-learning/wfs/text/basic-main.html>

```
https://maps.dwd.de/geoserver/ows?service=WFS&acceptversions=2.0.0&request=GetCapabilities index_leaflet_wfs.html:43
index_leaflet_wfs.html:44
▼ #document (file:///C:/Users/Anwender/thesis_webapp/index_leaflet_wfs.html)
  <wfs:WFS_Capabilities xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.opengis.net/wfs/2.0" xmlns:wfs="http://www.opengis.net/wfs/2.0" xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:gml="http://www.opengis.net/gml/3.2" xmlns:fes="http://www.opengis.net/fes/2.0" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:inspire_dls="http://inspire.ec.europa.eu/schemas/inspire_dls/1.0" xmlns:inspire_common="http://inspire.ec.europa.eu/schemas/common/1.0" xmlns:dwd="https://www.dwd.de" version="2.0.0" xsi:schemaLocation="http://www.opengis.net/wfs/2.0 https://maps.dwd.de/geoserver/schemas/wfs/2.0/wfs.xsd http://inspire.ec.europa.eu/schemas/inspire_dls/1.0 https://inspire.ec.europa.eu/schemas/inspire_dls/1.0/inspire_dls.xsd" updateSequence="20391">
    ▶ <ows:ServiceIdentification> ☰ </ows:ServiceIdentification>
    ▶ <ows:ServiceProvider> ☰ </ows:ServiceProvider>
    ▶ <ows:OperationsMetadata> ☰ </ows:OperationsMetadata>
    ▶ <FeatureTypeList> ☰ </FeatureTypeList>
    ▶ <fes:Filter_Capabilities> ☰ </fes:Filter_Capabilities>
  </wfs:WFS_Capabilities>
```

Abbildung 18: Beispiel für eine WFS-Anfrage und XML als Antwort.

Quelle: Eigene Darstellung.

Datenquelle: DWD, www.dwd.de

In Kapitel fünf erfolgt eine detaillierte Auseinandersetzung mit den implementierten Beispielen.

3 Auswahl der Technologien und Methodik

Dieses Kapitel fokussiert sich auf die Identifikation führenden Akteure im Markt der digitalen Kartographie. Des Weiteren erfolgt eine detaillierte Erläuterung der Vergleichsmetriken sowie eine Einführung in die Testumgebung.

3.1 Identifikation führender Web Mapping Lösungen

Die vorliegende Arbeit konzentriert sich auf eine Auswahl repräsentativer JavaScript-basierter Mapping-Frameworks, die aktuell auf dem Markt führend sind. Die Auswahl basiert auf den Bekanntheitsgrad sowie der Verbreitung in der Praxis.

- Google Maps JavaScript API
- Leaflet
- Mapbox

Google Maps: Die Monopolstellung von Google Maps im Mapping-Bereich ist unbestreitbar. Analysen von BuiltWith ⁵⁸ zeigen, dass es derzeit über fünf Millionen Live-Domains im Internet gibt, welche die Google Maps JavaScript API zur Einbindung der Karte nutzen. Im Jahr 2021 wurde durch eine Marktforschung von Mordor Intelligence festgestellt, dass Google Maps einer der führenden Akteure auf dem digitalen Kartenmarkt ist. Gemäß der Marktforschung von Enlyft ⁵⁹ besitzt Google Maps einen Anteil von 78% besitzt im B2B Markt. Eine weitere Studie von Market Research Future ⁶⁰ identifiziert Google ebenfalls als Treiber im Markt für digitale Karten. Diese Statistiken verdeutlichen, dass Google Maps die dominierende Position innehat. Mit seiner weit verbreiteten Verwendung und seiner umfassenden Funktionalität dominiert Google Maps den Markt für Web-Mapping-Lösungen. Die hohe Gesamtanzahl von 147.057 ⁶¹ Fragen zu Google Maps auf Stack Overflow, davon 66.065 ⁶² neue Fragen,

⁵⁸ Google Maps API Usage Statistics. (o. D.). <https://trends.builtwith.com/mapping/Google-Maps-API>

⁵⁹ Top 5 products in the Web Mapping market. (o. D.). <https://enlyft.com/tech/web-mapping>

⁶⁰ Market Research Future. (2021). Digitale Karte: Marktgröße, Branchenanteil und Trends — 2030
<https://www.marketresearchfuture.com/de/reports/digital-map-market-6600>

⁶¹ <https://stackoverflow.com/search?q=google+maps>

zum Stand vom 07.05.2024, lässt auf die kontinuierliche Relevanz und das anhaltende Interesse der Entwicklergemeinschaft an dieser API schließen. Die hohe Aktivität und Beliebtheit der Google Maps API auf Plattformen wie GitHub ⁶³ lässt sich anhand von verschiedenen Statistiken belegen. So existieren derzeit über 16.200 JavaScript-Repositories, 37.000 Pull-Requests, 19 Millionen Commits und 76 Repositories, die jeweils mehr als 100 Sterne erhalten haben. Dies zeigt das starke Interesse und die breite Nutzung der Google Maps API in der Entwicklergemeinschaft.

Leaflet: Leaflet stellt ein flexibles Open-Source-JavaScript-Framework zur Erstellung interaktiver Kartenanwendungen bereit. Es umfasst ausschließlich die für die Kartendarstellung erforderlichen Werkzeuge und Funktionen, beinhaltet jedoch keine eigenen Geodaten oder Kartenlayer. Im B2B Markt liegt Leaflet auf dem 2 Platz ⁶⁴ mit einem Marktanteil von 8,12%. Im gesamten World Wide Web gibt es 588,680 Webseiten ⁶⁵, welche die Leaflet JS API verwenden. Leaflet wird nicht nur von der Europäischen Kommission, sondern auch von zahlreichen weiteren namhaften Institutionen und Unternehmen weltweit genutzt.

Mapbox: Die Marktstudie von Market Research Future ⁶⁶ aus dem Jahr 2021 zählt Mapbox zu den weltweit führenden Akteuren auf dem Markt für digitale Kartographie. Laut BuiltWith gibt es derzeit 337.036 Domains, die Mapbox ⁶⁷ verwenden. Auf GitHub verzeichnet Mapbox 57 Repositories mit mehr als 100 Sternen ⁶⁸ von insgesamt 6.100. Auf StackOverflow gibt es 19.518 Fragen ⁶⁹ zum Thema Mapbox. Darüber hinaus konnte Mapbox zahlreiche namhafte Kunden gewinnen.

⁶² <https://stackoverflow.com/questions/tagged/google-maps>

⁶³ <https://github.com/search?q=google+maps+language%3AJavaScript&type=repositories>

⁶⁴ Leaflet commands 8.74% market share in Web Mapping. (o. D.). <https://onlyft.com/tech/products/leaflet>

⁶⁵ Leaflet JS usage Statistics. (o. D.). <https://trends.builtwith.com/mapping/Leaflet-JS>

⁶⁶ Market Research Future. (2021). Digitale Karte: Marktgröße, Branchenanteil und Trends — 2030
<https://www.marketresearchfuture.com/de/reports/digital-map-market-6600>

⁶⁷ <https://trends.builtwith.com/mapping/Mapbox/Market-Share>

⁶⁸ <https://github.com/search?q=MapBox+language%3AJavaScript+stars%3A>100&type=repositories>

⁶⁹ <https://stackoverflow.com/search?q=MapBox>

3.2 Testumgebung

Als Testumgebung werden die meistgenutzten Browser herangezogen (vgl. Abbildung 20). Die Tests finden unter einer Netzwerkgeschwindigkeit von ca. 64 Mbit/s durchgeführt (vgl. Abbildung 19). Der implementierte Quellcode wird in einem öffentlich zugänglichen Repository auf GitHub bereitgestellt.⁷⁰ Bei der Evaluierung kommt folgende Hardware zum Einsatz (vgl. Tabelle 4).

| | |
|-----------------------|-------------------------|
| Prozessor (CPU) | AMD Ryzen 9 5900x |
| Grafikkarte (GPU) | Nvidia GeForce RTX 3070 |
| Arbeitsspeicher (RAM) | 32 GB |

Tabelle 4: Verwendete Hardware

Quelle: Eigene Darstellung

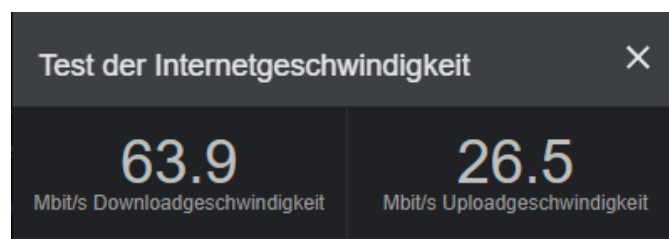


Abbildung 19: Netzwerkgeschwindigkeit

Quelle: Eigene Darstellung

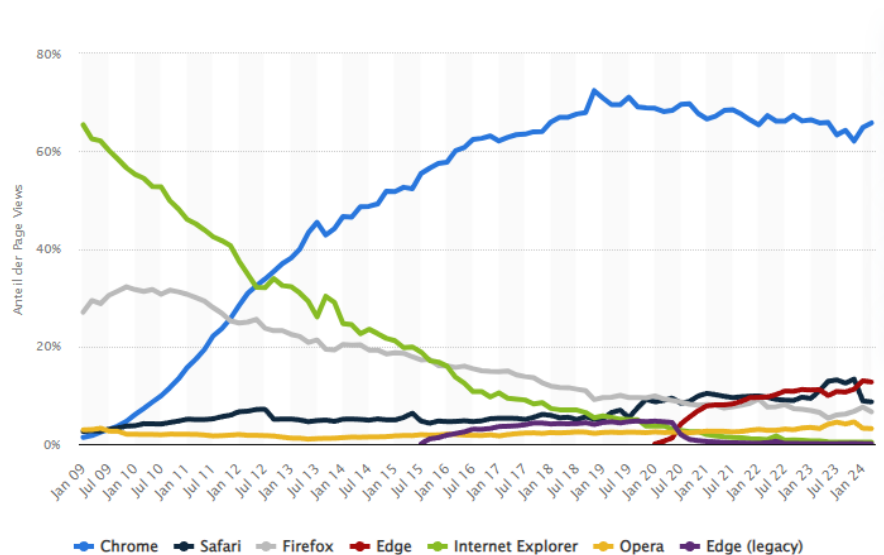


Abbildung 20: Marktanteile der führenden Browserfamilien an der Internetnutzung weltweit von Januar 2009 bis Mai 2024

Quelle: <https://de.statista.com/statistik/daten/studie/157944/umfrage/marktanteile-der-browser-bei-der-internetnutzung-weltweit-seit-2009/>

⁷⁰ https://github.com/geo1337/js_web_mapping_comparison

3.3 Performance

3.3.1 Ladegeschwindigkeit

Die Metrik Ladegeschwindigkeit bezieht sich auf die Zeit, die benötigt wird, um alle notwendigen Daten (Kacheln, Skripte) vom Server herunterzuladen und im Browser verfügbar zu machen. Die Ladegeschwindigkeit besagt, wie schnell der Endnutzer die Karte sehen und mit ihr interagieren kann. Die Reduktion der Ladezeiten führt zur schnelleren Bereitstellung der Inhalte und somit zu einer Steigerung der Benutzererfahrung.⁷¹

3.3.2 Reaktionsgeschwindigkeit

Die Reaktionsgeschwindigkeit einer Web-Mapping-Lösung spiegelt die Zeit wider, die benötigt wird, um auf Benutzeraktionen zu reagieren. Dies können Interaktionen wie Zoomen, Verschieben der Karte oder Klicken auf ein Kartenobjekt sein. Eine Reaktionszeit bis zu 100 ms wird von den Endbenutzern als unmittelbar wahrgenommen.⁷²

3.3.3 Ressourcenverbrauch

Der Ressourcenverbrauch gibt Aufschluss darüber wie viel an Systemressourcen in den einzelnen Browsern bzw. von den einzelnen Web Mapping Lösungen genutzt werden. Darunter werden folgende Ressourcen berücksichtigt:

- Arbeitsspeicher (RAM)
- Prozessor (CPU)
- Grafikkarte (GPU)

⁷¹ Vgl. Jeremy L Wagner. Web Performance in Action: Building Faster Web Pages. Manning Verlag. (2017), S. 2 ff.

⁷² Vgl. Ilya Grigorik. High Performance Browser Networking. O'Reilly Verlag (2013), S. 188.

3.4 Funktionalitäten

3.4.1 Event-Handling

Das Event-Handling (zu Deutsch: Ereignisbehandlung) bildet die Kernfunktionalität für eine interaktive Web Mapping Lösung. Zu den typischen Ereignissen zählen:

- Mausereignisse: Klicks und Hover.
- Kartenereignisse: Zoomen, Verschieben der Karte, Änderung der Kartensicht, Änderung der Kartenschicht.
- Implementierung von benutzerdefinierten Funktionen basierend auf Ereignissen.

3.4.2 Unterstützung von Geodatenformaten

Bei der Betrachtung der Unterstützung von Geodatenformaten geht es darum, welche Formate zur Darstellung geografischer Daten verwendet werden können.

3.4.3 Unterstützung von Geodiensten

Die Untersuchung der Unterstützung von Geodiensten in JavaScript-basierten Web Mapping Lösungen fokussiert auf die Evaluierung der Integration externer geografischer Daten- und Kartendienste in die Anwendung. Dabei wird geprüft welche Web Mapping Lösungen die Integration von WMS und WFS unterstützen.

3.4.4 Erweiterbarkeit

Die Erweiterbarkeit bezieht sich darauf, wie flexibel und einfach die Anwendung um zusätzliche Funktionen, Integration mit anderen Diensten oder Plugins erweitert werden kann.

3.4.5 Interoperabilität

Holistisch betrachtet ist Web Mapping bzw. die Implementierung von Web Mapping Lösungen eine Komponente der Webentwicklung, deshalb wird hierbei die Interoperabilität mit gängigen JavaScript-Frameworks der Webentwicklung untersucht.

3.5 Entwicklungsgeschwindigkeit

Die Entwicklungsgeschwindigkeit wird durch mehrere Faktoren beeinflusst, darunter vordefinierte Elemente und Funktionen, Dokumentation und Community-Aktivität. Im Rahmen dieser Untersuchung wird eine Bewertung der Entwicklungsgeschwindigkeit einzelner Web Mapping Lösungen vorgenommen.

3.5.1 Dokumentation

Eine ausführliche Dokumentation ist ein bedeutender Faktor für die Entwicklung. Im Rahmen der wissenschaftlichen Arbeit erfolgt eine Einschätzung des Umfangs und der Vollständigkeit der Dokumentation.

3.5.2 Community-Aktivität

Der Begriff „Community-Aktivität“ bezeichnet in diesem Kontext die Beteiligung, Vernetzung und den Austausch innerhalb einer Entwicklergemeinschaft, die eine bestimmte Web Mapping Lösung nutzt. Die Analyse der Aktivität erfolgt über gängige Entwickler-Plattformen wie Stack Overflow und Github. Des Weiteren erfolgt eine Analyse der auf YouTube verfügbaren Tutorials. Die Erhebung von Kennzahlen erfolgt für alle betrachteten Plattformen.

3.6 Datenschutz und Sicherheit

Die Risiken bei der Verwendung eines API-Keys werden hervorgehoben und mögliche Sicherheitsmaßnahmen vorgestellt. Anschließend erfolgt eine Untersuchung der Erfassung, Speicherung und Nutzung von Nutzerdaten.

3.7 Kosten und Lizenzierung

Preismodelle und Kostenstrukturen können ausschlaggebend bei der Entscheidungsfindung sein. Daher werden die Preismodelle der einzelnen Web Mapping Lösungen untersucht und verglichen. Des Weiteren erfolgt eine Untersuchung der Lizenzierungsbedingungen.

4 Web Mapping Lösungen im Vergleich

4.1 Google Maps JavaScript API

4.1.1 Nutzung und Integration

Die Nutzung setzt die Einrichtung eines Google-Kontos sowie die Registrierung in der Google Cloud Plattform ⁷³ voraus. Des Weiteren ist Einrichtung eines Projekts sowie die Generierung eines API-Schlüssels ⁷⁴ zur Authentifizierung erforderlich. Im Zuge der Arbeit wurde dieser Schlüssel generiert, wobei das kostenlose Kontingent von 200 \$ ⁷⁵ pro Monat in Anspruch genommen wurde (Stand: Juni 2024). Die Integration erfolgt über Skript-Tags in einem HTML-Dokument oder über Komponenten in einem Framework.

4.1.2 Ladegeschwindigkeit

Beim erstmaligen Laden der Webseite werden Anfragen versendet, um die benötigten Ressourcen zu laden. Dazu zählen unter anderem Schriftarten, Bilder, Vektorgrafiken und JavaScript-Dateien. Die API versendet eine XHR-Anfrage an den Host, diese Anfrage beinhaltet Parameter, die durch die eigene Implementierung mit JavaScript generiert wurden. Bei einem erneuten Aufruf der Webseite werden die entsprechenden Dateien aus dem lokalen Cache geladen, nämlich von der Festplatte und dem Arbeitsspeicher. Die Entwickler Tools von Google Chrome gewähren einen tieferen Einblick, es lässt sich erkennen das die JavaScript-Dateien sowie Bilder bzw. Kacheln hauptsächlich für die Ladezeit verantwortlich sind und CSS-Dateien sowie Schriftarten eher einen geringeren Einfluss haben (vgl. Abbildung 21).

⁷³ <https://console.cloud.google.com/>

⁷⁴ <https://developers.google.com/maps/documentation/javascript/get-api-key?hl=de>

⁷⁵ <https://mapsplatform.google.com/pricing/>

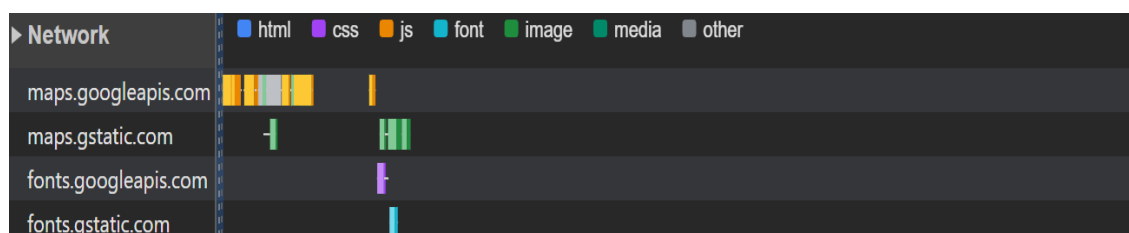


Abbildung 21: Google Chrome Dev Tools Performance Insights (Google Maps JS API)

Quelle: Eigene Darstellung

Die Ladegeschwindigkeit wird durch die Inanspruchnahme zusätzlicher Features, wie beispielsweise Marken oder Infofenstern, erhöht, da für diese Objekte zusätzliche Skripte und Stylesheets geladen und ausgeführt werden müssen. Des Weiteren erfasst die Performance-Analyse von Google Chrome die Zeiten, die für die Ausführung des Skripts und das Rendering benötigt wurden (vgl. Abbildung 22).⁷⁶

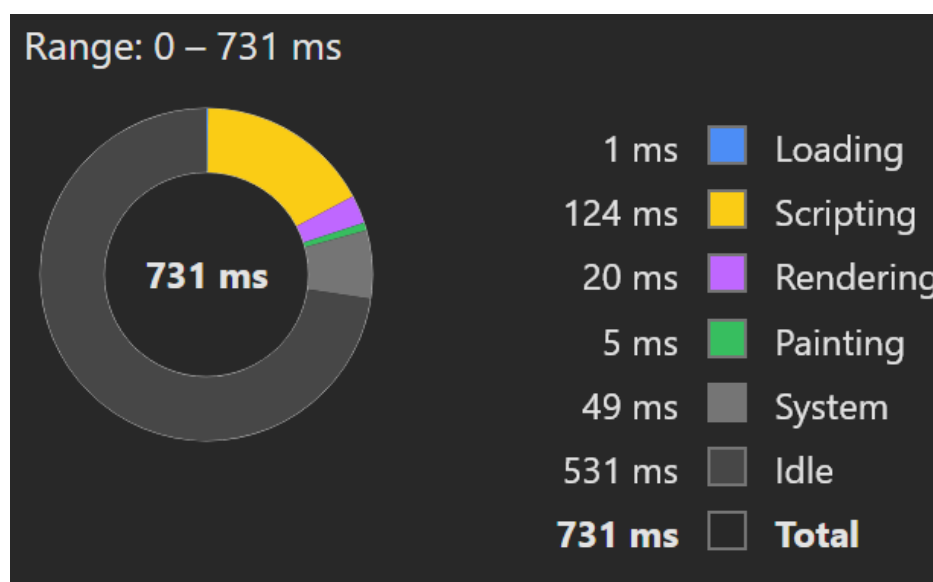


Abbildung 22: Google Chrome Performance-Analyse (Google Maps JS API)

Quelle: Eigene Darstellung

Im Rahmen der Implementierung wurde ein Skript implementiert, das beim Laden der Webseite die Methode „performance.now()“⁷⁷ aufruft. Anschließend wird auf das Ereignis „tilesloaded“⁷⁸ von der API gewartet. Die Zeitmessung erfolgt zwischen dem Aufruf von „performance.now()“ und dem Eintreten des „tilesloaded“-Ereignisses (vgl. Tabelle 5 und Tabelle 6).

⁷⁶ <https://developer.chrome.com/docs/devtools/performance/reference?hl=de>

⁷⁷ <https://developer.mozilla.org/en-US/docs/Web/API/Performance/now>

⁷⁸ <https://developers.google.com/maps/documentation/javascript/reference/map?hl=de#Map.tilesloaded>

https://github.com/geo1337/js_web_mapping_comparison/blob/main/google_maps_api/index_google_loadspeed.html

Im ersten Testfall wird eine einfache Karte, die über Stuttgart zentriert ist, untersucht. Die Messungen in Tabelle 7 und 8 wurden aus den Entwicklerkonsolen der jeweiligen Browser entnommen.

| Browser | Benötigte Zeit in (ms) | Anzahl der Ressourcen |
|-----------------|------------------------|-----------------------|
| Google Chrome | 634.80 | 29 |
| Microsoft Edge | 709.40 | 29 |
| Mozilla Firefox | 785 | 27 |
| Opera | 719.10 | 29 |

Tabelle 5: Messungen per JavaScript-Code ohne Cache (Google Maps API)

Quelle: Eigene Darstellung

| Browser | Benötigte Zeit in (ms) | Anzahl der Ressourcen |
|-----------------|------------------------|-----------------------|
| Google Chrome | 371.80 | 29 |
| Microsoft Edge | 416.20 | 29 |
| Mozilla Firefox | 392 | 11 |
| Opera | 381.90 | 29 |

Tabelle 6: Messungen per JavaScript-Code mit Cache (Google Maps API)

Quelle: Eigene Darstellung

| Browser | Ladevorgang (ms) | Fertigstellen (ms) |
|-----------------|------------------|--------------------|
| Google Chrome | 294 | 771 |
| Microsoft Edge | 502 | 901 |
| Mozilla Firefox | 236 | 999 |
| Opera | 517 | 853 |

Tabelle 7: Messungen aus den Entwicklerkonsolen ohne Cache (Google Maps API)

Quelle: Eigene Darstellung

| Browser | Ladevorgang (ms) | Fertigstellen (ms) |
|-----------------|------------------|--------------------|
| Google Chrome | 51 | 423 |
| Microsoft Edge | 72 | 462 |
| Mozilla Firefox | 82 | 461 |
| Opera | 83 | 458 |

Tabelle 8: Messungen aus den Entwicklerkonsolen mit Cache (Google Maps API)

Quelle: Eigene Darstellung

Eine weitere Möglichkeit zur Messung der Ladegeschwindigkeit besteht in der Implementierung einer JavaScript-Funktion, welche die einzelnen Messungen in das lokale Speicher-Objekt (localStorage) ⁷⁹ schreibt. Sobald eine Anzahl von 20 Einträgen erreicht ist, wird der Durchschnitt gebildet (vgl. Abbildung 23).

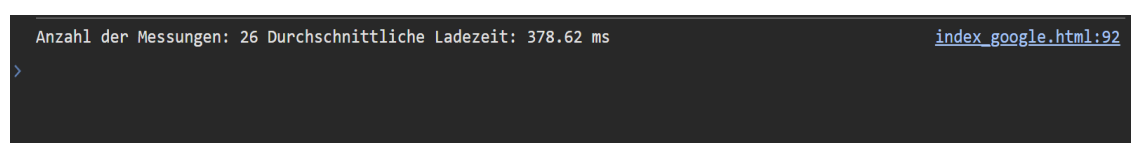


Abbildung 23: Durchschnittliche Ladegeschwindigkeit

Quelle: Eigene Darstellung

Die Dauer der Ladeprozedur von Google Maps ist abhängig von der Art der verwendeten Kacheln. Dabei kann zwischen Raster und Vektorkacheln unterschieden werden. Rasterkacheln weisen in der Regel eine längere Ladezeit auf, da es sich um Bilddateien handelt. Diese sind von der serverseitigen Performance abhängig.

Vektorkacheln laden schneller und sind in mehr von der Hardware des Endbenutzers abhängig. Dies impliziert, dass die Leistungsfähigkeit des Endgeräts den Ladevorgang von Vektorkacheln beschleunigen kann. ^{80 81} Die Rasterkacheln von Google werden in den Formaten JPG und WebP bereitgestellt.

⁷⁹ <https://www.mediaevent.de/javascript/local-storage.html>

⁸⁰ Vgl. Martin Elias. Raster vs Vector Map Tiles: What Is the Difference Between the Two Data Types? 2023. <https://documentation.maptiler.com/hc/en-us/articles/4411234458385-Raster-vs-Vector-Map-Tiles-What-Is-the-Difference-Between-the-Two-Data-Types>

⁸¹ Vgl. Néték, Rostislav & Masopust, Jan & Pavlicek, & Pechanec, Vilém. (2020). Performance Testing on Vector vs. Raster Map Tiles—Comparative Study on Load Metrics. ISPRS International Journal of Geo-Information. 9. 101. 10.3390/ijgi9020101. S. 21.

Die Kachelart kann in der Google Plattform unter Kartenverwaltung umgestellt werden (vgl. Abbildung 24).

Karten-ID bearbeiten

Name *
Test_1

Beschreibung

Kartentyp
JavaScript

☒ Raster
☐ Vektor

Diese Einstellungen können auch im Code überschrieben werden.

ABBRECHEN

AKTUALISIEREN

Abbildung 24: Karten-ID Zuordnung und Kachelumstellung

Quelle: Eigene Darstellung

Im Rahmen der Tests konnte die Vektorkarte allerdings nicht fehlerfrei bzw. vollständig geladen werden. Diese Problematik wurde bereits im Google Issue Tracker von anderen Entwicklern thematisiert.^{82 83} Dieser Fehler trat nur unter der Verwendung bzw. Kombination mit WebGL auf (vgl. Abbildung 25).

```

❗ Attempted to load a Vector Map, but failed. Falling back to Raster. Please see https://developers.google.com/maps/documentation/javascript/webgl/support for more info js?libraries=maps&gl=de&map_id=144
...Pj @ js?libraries=maps&gl=de&map_id=144
(anonymous) @ map.js:117
Promise.then (async) @ map.js:116
Promise.then (async) @ map.js:115
r0.fg @ map.js:115
(anonymous) @ js?libraries=maps&gl=de&map_id=1221
Promise.then (async) @ js?libraries=maps&gl=de&map_id=1220
Io @ js?libraries=maps&gl=de&map_id=1220
initMap @ index_google https://maps.googleapis.com/maps/api/js?libraries=maps&key=AIzaSyDk_ekwFvC1bWcQs8aR-A33-
await in initMap (async) K2cpKQ8v=weekly&callback=google.maps._ib_1220
(anonymous) @ index_google.html:71

❗ The map is not a vector map, which will prevent use of WebGLOverlayView. js?libraries=maps&gl=de&map_id=1377
log @ js?libraries=maps&gl=de&map_id=1377
a @ js?libraries=maps&gl=de&map_id=1375
await in a (async) @ js?libraries=maps&gl=de&map_id=1375
(anonymous) @ js?libraries=maps&gl=de&map_id=1375
Promise.then (async) @ js?libraries=maps&gl=de&map_id=1375
...on.map_changed @ js?libraries=maps&gl=de&map_id=1375
uk @ js?libraries=maps&gl=de&map_id=1375
...Sk.set @ js?libraries=maps&gl=de&map_id=1375
(anonymous) @ js?libraries=maps&gl=de&map_id=1375
initMap @ index_google.html:67
await in initMap (async) @ index_google.html:67
(anonymous) @ index_google.html:67

```

Abbildung 25: Fehler beim Laden der Vektorkarte (Google Maps JS API)

Quelle: Eigene Darstellung

⁸² <https://issuetracker.google.com/issues/330098647?pli=1>

⁸³ <https://issuetracker.google.com/issues/263838427>

Es wurden die Metriken beim erstmaligen Laden einer Karte zwischen Raster und Vektor- Kacheln in Google Chrome verglichen. In Anbetracht der übertragenen Datenmenge und Ressourcengröße ist erkennbar das Vektorkacheln schneller laden (vgl. Tabelle 9).

| Kachel | Anfragen | Übertragene Datenmenge | Ressourcen- größe | Fertigstellung | Ladevorgang |
|--------|----------|------------------------|----------------------|----------------|-------------|
| Raster | 62 | 808 kB | 1,6 MB | 843 ms | 466 ms |
| Vektor | 94 | 2,1 MB | 5,8 MB | 1.09 s | 444 ms |

Tabelle 9: Raster Kacheln vs. Vektor Kacheln (Google Maps JS API)

Quelle: Eigene Darstellung

Zur Erweiterung der Testmöglichkeiten wurde eine Webapplikation in einer Node.js^{84 85}-Umgebung implementiert, die das Selenium-Framework⁸⁶ beinhaltet. Dieses Framework ermöglicht die Automatisierung von Browsern sowie das Schreiben von Tests für Webanwendungen. Im Rahmen der Testdurchführung wurden drei Aufrufe der Webseite simuliert, deren Messergebnisse anschließend lokal in einer CSV-Datei gespeichert und mittels ChartJS⁸⁷ visualisiert wurden. Der lokale Webserver wurde mittels XAMPP⁸⁸ bzw. Apache realisiert. Die einzelnen Messungen werden im Skript aufsummiert und durch die Anzahl dividiert, um den Durchschnitt zu erhalten. Im Skript wird die Zeit gemessen, die nach dem Aufrufen der Webseite bis zum Eintreten des Ereignisses „tilesloaded“⁸⁹ der API vergeht (vgl. Abbildung 26). Dieses Ereignis signalisiert, dass alle sichtbaren Komponenten vollständig geladen wurden und die Karte gerendert wurde. Die automatisierten Tests wurden in den Webbrowsern Google Chrome und Microsoft Edge durchgeführt. Dabei werden auch die Prozessornutzung sowie die Verwendung des Arbeitsspeichers durch das Skript gemessen. Die CPU-

⁸⁴ <https://nodejs.org/en/>

⁸⁵ Vgl. Phillip Ackermann. Fullstack-Entwicklung. Das Handbuch für Webentwicklung. Rheinwerk Verlag. (2023), S. 435 ff.

⁸⁶ <https://www.selenium.dev/>

⁸⁷ <https://www.chartjs.org/>

⁸⁸ <https://www.apachefriends.org/de/index.html>

⁸⁹ <https://developers.google.com/maps/documentation/javascript/reference/map?hl=de#Map.tilesloaded>

Auslastung wird mittels der WMIC-API⁹⁰ von Windows erfasst. Die Messung erfolgt zwischen zwei Zeitpunkten, nämlich vor dem Laden der Webseite und nach dem Laden der Karte. Die Messung der RAM-Nutzung erfolgt zu denselben Zeitpunkten mittels des „os“-Moduls⁹¹ in Node.js. Die Dauer der einzelnen Testintervalle von jeweils zehn Sekunden wurde bewusst gewählt, um eine Stabilisierung der Systemressourcen zu erreichen. Hinsichtlich der CPU-Messungen erwies sich die Bildung der Differenz zwischen zwei Ereignissen bzw. Zeitpunkten als unpräzise, da Schwankungen im Betriebssystem auftauchen durch andere Prozesse. Der bessere Ansatz wäre es die CPU-Nutzung im Verlauf alle 100ms zu messen. Zusätzlich werden die Temperatur der Grafikkarte sowie deren prozentuale Auslastung erfasst, was mittels Befehlszeilen des NVIDIA System Management Interface⁹² realisiert wurde. Darüber hinaus werden die Anzahl der Netzerkennungen sowie deren Latenz festgehalten. Die Messungen werden in zwei Szenarien durchgeführt: zunächst mit einer Standard-Karte und anschließend mit einem Stresstest, der durch das Hinzufügen von 2000 Markern simuliert wird. Die Ausführung des automatisierten Tests in Firefox war nicht möglich, da das asynchrone Skript nicht ausgeführt wurde und ein Timeout-Fehler auftrat. Dies ist höchstwahrscheinlich auf die Sicherheitseinstellungen von Firefox zurückzuführen. Zu dieser Problematik wurde ebenfalls eine Stackoverflow-Frage⁹³ verfasst, die jedoch ebenfalls keine Lösung lieferte. Als Hindernis stellte sich auch die Isolierung der Testumgebung heraus. Der Quellcode zum Test findet sich unter der folgenden URL:

https://github.com/geo1337/js_web_mapping_comparison/blob/main/google_maps_api/test_google_maps_api.js

⁹⁰ <https://www.windows-faq.de/2018/12/22/per-wmic-befehl-informationen-zur-cpu-auslesen/>

⁹¹ https://www.w3schools.com/nodejs/ref_os.asp

⁹² <https://developer.nvidia.com/system-management-interface>

⁹³ <https://stackoverflow.com/questions/78503115/firefox-browser-blocking-execution-of-asynchronous-javascript>



Abbildung 26: Messungen des Selenium Tests in Chart.js visualisiert (Google Maps JS API).

Quelle: Eigene Darstellung

Im Rahmen der vorliegenden Arbeit wurde eine weitere Testmöglichkeit erarbeitet, bei der 30 gleichzeitige HTTPs-Anfragen per JMeter ⁹⁴ an den Server „maps.googleapis.com“ versendet wurden. Die Parameter beinhalteten den API-Schlüssel sowie Breiten und Längengrade für die Geolocation-API, wobei es sich jedoch um denselben Server handelt, der für die Bereitstellung der Ressourcen für die Google Maps JS API zuständig ist. Daher sind die Antwortzeiten dieses Servers direkt relevant für die Beurteilung der Performance der Google Maps JS API, da beide APIs höchstwahrscheinlich auf dieselbe Infrastruktur zugreifen. Anschließend wurden die Antwortzeiten in einem aggregierten Graph visualisiert und als Bild abgespeichert (vgl. Abbildung 27). Als Antwort erhält man die zugehörigen Informationen zu der Position im JSON-Format (siehe Anhang A).

⁹⁴ <https://jmeter.apache.org/>

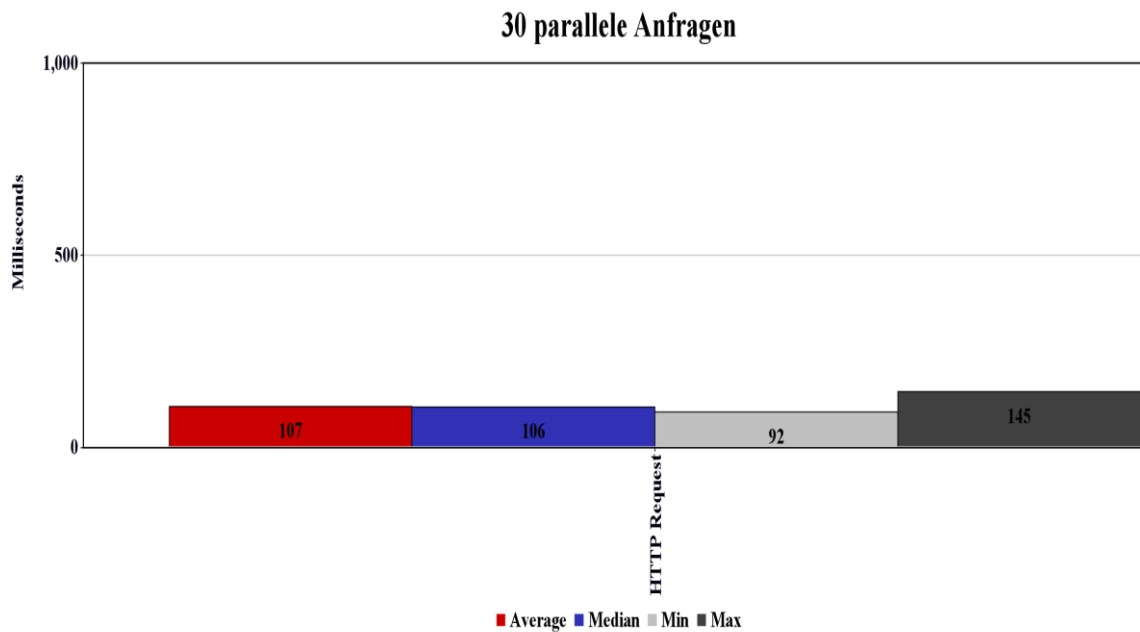


Abbildung 27: Antwortzeiten des Servers „maps.googleapis.com“ (JMeter)

Quelle: Eigene Darstellung

Zur Evaluierung der Ladezeit auf mobilen Endgeräten wurde das Tool „Lighthouse“ von Google ⁹⁵ eingesetzt. Das Tool „Lighthouse“ von Google simuliert einen mobilen Zugriff auf die Webseite und misst Performance-Metriken. Der Test wird dabei unter Verwendung eines langsamen 4G-Netzes ausgeführt (vgl. Abbildung 28). Die Metrik First Contentful Paint liefert ein ähnliches Ergebnis wie die Messungen per JavaScript-Code und im automatisierten Test. Die 0.6s könnte man auf das Rendern der sichtbaren Kacheln beziehen. ⁹⁶

⁹⁵ https://developer.chrome.com/docs/lighthouse/performance/performance-scoring?utm_source=lighthouse&utm_medium=devtools&hl=de

⁹⁶ <https://web.dev/articles/fcp?hl=de>

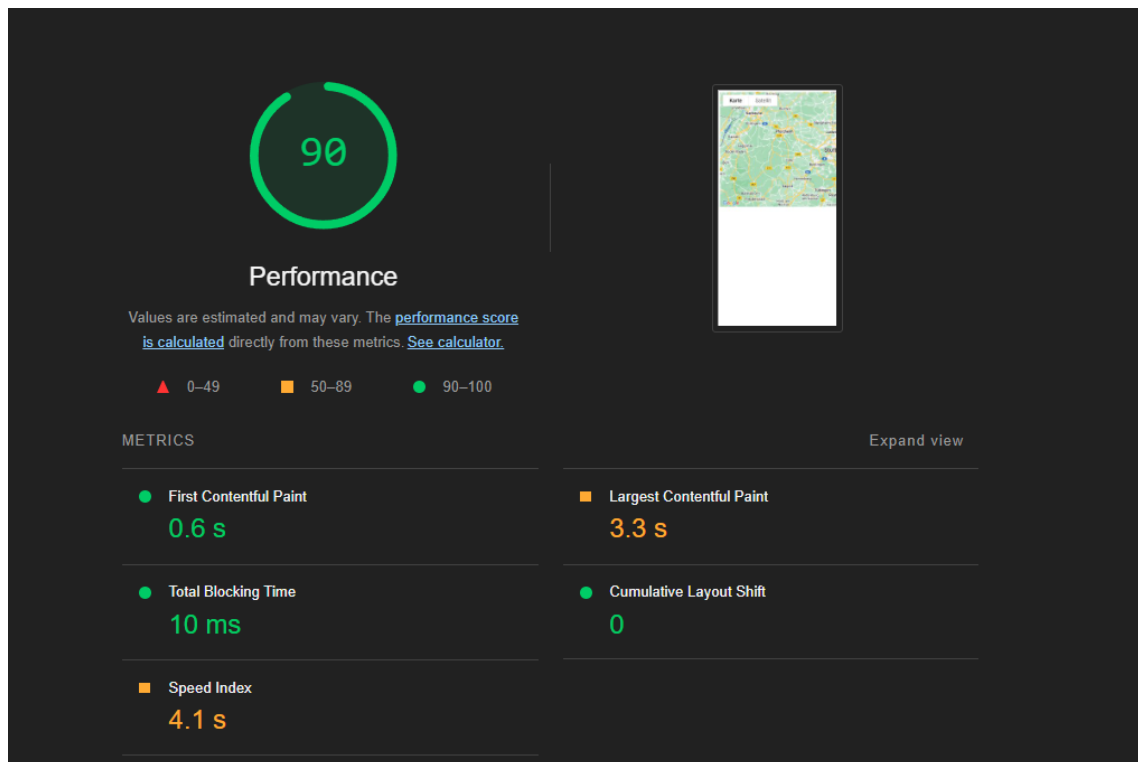


Abbildung 28: Google Lighthouse Test (Google Maps JS API)

Quelle: Eigene Darstellung

4.1.3 Reaktionsgeschwindigkeit

Die dokumentierten Ereignisse⁹⁷ dienten als Grundlage für die Implementierung von Ansätzen zur Messung der Reaktionsgeschwindigkeit. Diese beinhalteten die Messung des Nachladens der Karte beim Zoomen oder Verschieben, sowie das Klicken auf einen Marker und das darauffolgende Anzeigen eines Infofensters. Die Google Maps API bietet derartige Ereignisse standardmäßig an, sodass Listener auf diese Events gelegt werden konnten. Diese wurden als Auslöser für die Startzeit verwendet und anschließend wurde auf das Event „tilesloaded“ gewartet, welches als Trigger für die Endzeit diente (vgl. Abbildung 29). Der hier beschriebene Ansatz zielt darauf ab, eine möglichst realistische User-Interaktion zu simulieren. Die Tatsache, dass lediglich sichtbare Ausschnitte (Kacheln)⁹⁸ und nicht die gesamte Karte nachgeladen werden, führt zu einer hohen Geschwindigkeit und Effizienz.

⁹⁷ <https://developers.google.com/maps/documentation/javascript/events?hl=de>

⁹⁸ <https://developers.google.com/maps/documentation/javascript/coordinates?hl=de>

Der Quellcode ist unter folgender URL auffindbar:

https://github.com/geo1337/js_web_mapping_comparison/blob/main/google_maps_api/index_google_reactionspeed.html

| |
|--|
| Ladezeit der restlichen Karte beim Zoom: 248.50 ms oder 0.25 s |
| Ladezeit der restlichen Karte beim Zoom: 420.10 ms oder 0.42 s |
| Ladezeit der restlichen Karte beim Zoom: 228.30 ms oder 0.23 s |
| Ladezeit der restlichen Karte beim Zoom: 247.10 ms oder 0.25 s |
| Zeit zwischen dem Klicken auf einen Marker und dem Anzeigen des Infofensters 8.60 ms |
| Ladezeit der restlichen Karte beim Verschieben: 348.70 ms oder 0.35 s |
| Ladezeit der restlichen Karte beim Verschieben: 86.00 ms oder 0.09 s |
| Ladezeit der restlichen Karte beim Verschieben: 317.10 ms oder 0.32 s |
| Ladezeit der restlichen Karte beim Verschieben: 288.50 ms oder 0.29 s |
| Ladezeit der restlichen Karte beim Verschieben: 52.80 ms oder 0.05 s |

Abbildung 29: Messung der Reaktionsgeschwindigkeit per JS-Code (Google Maps JS API)

Quelle: Eigene Darstellung

Die Zeit der Reaktionsgeschwindigkeit variiert ebenfalls in Abhängigkeit vom Caching. Um Diese Hypothese zu belegen wurde ein JavaScript-Code implementiert, welcher, sobald ein Zoom-Event eintritt über die Performance API alle Dateien mit den Endungen JPG oder Webp trackt. Durch die Performance API kann dabei die Ladezeit berechnet werden da die Startzeit sowie die Endzeit der jeweiligen Ressource bereitsteht (vgl. Abbildung 30 und Abbildung 31). Beim Caching werden die benötigten Kacheln von der Festplatte geladen.

| | |
|--|---|
| Bildformat: jpg, Ladezeit: 31.80ms | index_google_reactionspeed.html:109 |
| Bildformat: jpg, Ladezeit: 36.30ms | index_google_reactionspeed.html:109 |
| Bildformat: jpg, Ladezeit: 41.20ms | index_google_reactionspeed.html:109 |
| Bildformat: jpg, Ladezeit: 46.90ms | index_google_reactionspeed.html:109 |
| Bildformat: jpg, Ladezeit: 56.00ms | index_google_reactionspeed.html:109 |
| Bildformat: jpg, Ladezeit: 57.60ms | index_google_reactionspeed.html:109 |
| Bildformat: jpg, Ladezeit: 62.50ms | index_google_reactionspeed.html:109 |
| Bildformat: jpg, Ladezeit: 70.60ms | index_google_reactionspeed.html:109 |
| Bildformat: jpg, Ladezeit: 76.30ms | index_google_reactionspeed.html:109 |
| Bildformat: jpg, Ladezeit: 81.00ms | index_google_reactionspeed.html:109 |
| Bildformat: jpg, Ladezeit: 88.70ms | index_google_reactionspeed.html:109 |
| Bildformat: jpg, Ladezeit: 94.40ms | index_google_reactionspeed.html:109 |
| Ladezeit der restlichen Karte beim Zoomen: 301.00 ms oder 0.30 s | index_google_reactionspeed.html:127 |

Abbildung 30: Nachladedauer von Ressourcen ohne Cache (Google Maps JS API)

Quelle: Eigene Darstellung

| | | |
|---|-----------------------------------|---|
| 2 | Bildformat: jpg, Ladezeit: 0.90ms | index_google_reactionspeed.html:109 |
| | Bildformat: jpg, Ladezeit: 1.10ms | index_google_reactionspeed.html:109 |
| | Bildformat: jpg, Ladezeit: 1.30ms | index_google_reactionspeed.html:109 |
| 2 | Bildformat: jpg, Ladezeit: 1.40ms | index_google_reactionspeed.html:109 |
| | Bildformat: jpg, Ladezeit: 1.20ms | index_google_reactionspeed.html:109 |
| | Bildformat: jpg, Ladezeit: 1.50ms | index_google_reactionspeed.html:109 |
| | Bildformat: jpg, Ladezeit: 1.40ms | index_google_reactionspeed.html:109 |
| 3 | Bildformat: jpg, Ladezeit: 1.30ms | index_google_reactionspeed.html:109 |

Abbildung 31: Nachladen von Ressourcen mit Cache (Google Maps JS API)

Quelle: Eigene Darstellung

Des Weiteren wurde ein automatisierter Test zur Ermittlung der Reaktionsgeschwindigkeit implementiert. Hierbei betätigt das Selenium Skript alle fünf Sekunden insgesamt fünfmal den Vergrößern-Knopf (zoom in) und fünfmal den Verkleinern-Knopf (zoom out). Die Konsolen-Ausgaben des JavaScript-Codes der Index.html-Datei werden gespeichert, die Werte der einzelnen Ladezeiten nach einem Zoomvorgang werden extrahiert, addiert und anschließend durch die Anzahl der simulierten Zoomvorgänge dividiert.

https://github.com/geo1337/js_web_mapping_comparison/blob/main/google_maps_api/test_zoom.js

4.1.4 Ressourcenverbrauch

Der Performance-Tab von Google Chrome gewährt Einblick in die Arbeitsspeichernutzung der Webseite (vgl. Abbildung 32). Zu Beginn ist die Arbeitsspeicher-Auslastung konstant bei 5,6 MB sobald die Skripte von Google geladen und ausgeführt werden gibt es einen anstieg auf 11,8 MB.

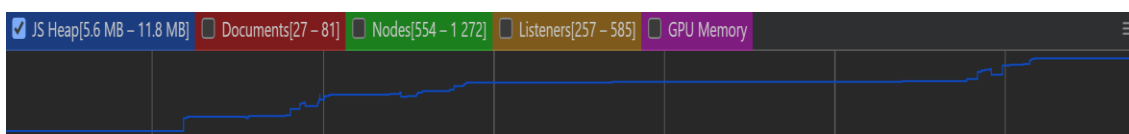


Abbildung 32: Arbeitsspeicher-Nutzung (Google Maps JS API)

Quelle: Eigene Darstellung

Es liefert auch Informationen über die Grafikkarten-Aktivität, diese wird knapp 74 ms für das Rendering beansprucht (vgl. Abbildung 33).

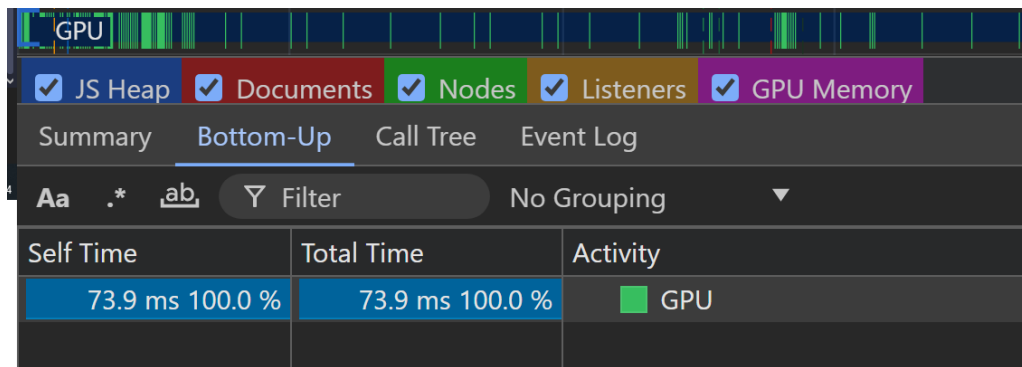


Abbildung 33: GPU-Nutzung (Google Maps JS API)

Quelle: Eigene Darstellung

Des Weiteren kann eingesehen werden, welche Skripte oder Ereignisse die CPU am längsten verwendeten (vgl. Abbildung 34).

| Summary Bottom-Up Call Tree Event Log | | | |
|--|-----------|--------------|--------------------------|
| Aa .* ab Filter No Grouping ▼ | | | |
| All <input type="checkbox"/> Loading <input type="checkbox"/> Messaging <input checked="" type="checkbox"/> Scripting <input type="checkbox"/> Rendering <input type="checkbox"/> Painting | | | |
| Start Time | Self Time | Total Time ▼ | Activity |
| 728.9 ms | 0.0 ms | 22.2 ms | ▶ Timer Fired |
| 707.8 ms | 0.1 ms | 17.8 ms | ▶ Evaluate Script |
| 227.6 ms | 0.0 ms | 13.6 ms | ▶ Run Microtasks |
| 0.3 ms | 0.3 ms | 13.0 ms | ▶ Evaluate Script |
| 332.3 ms | 0.0 ms | 9.8 ms | ▶ XHR Ready State Change |
| 216.4 ms | 0.1 ms | 8.2 ms | ▶ Evaluate Script |
| 177.7 ms | 0.4 ms | 7.6 ms | ▶ Evaluate Script |

Abbildung 34: CPU-Nutzung durch Skripte (Google Maps JS API)

Quelle: Eigene Darstellung

Während des Selenium-Tests wurde ebenfalls eine Überwachung und Visualisierung des Ressourcenverbrauchs durchgeführt. Allerdings war eine isolierte Analyse der API nicht möglich, da der Overhead des Browsers sowie der Kommunikation zwischen Selenium und dem Browser in die Berechnung mit einfließen. Die Balkendiagramme veranschaulichen den durchschnittlichen Arbeitsspeicherverbrauch, der aus drei aufeinanderfolgenden Aufrufen resultiert. Zudem wird der Unterschied zwischen einer normalen Karte und einem Belastungstest mit 2000 Markern verdeutlicht (vgl. Abbildung 35).

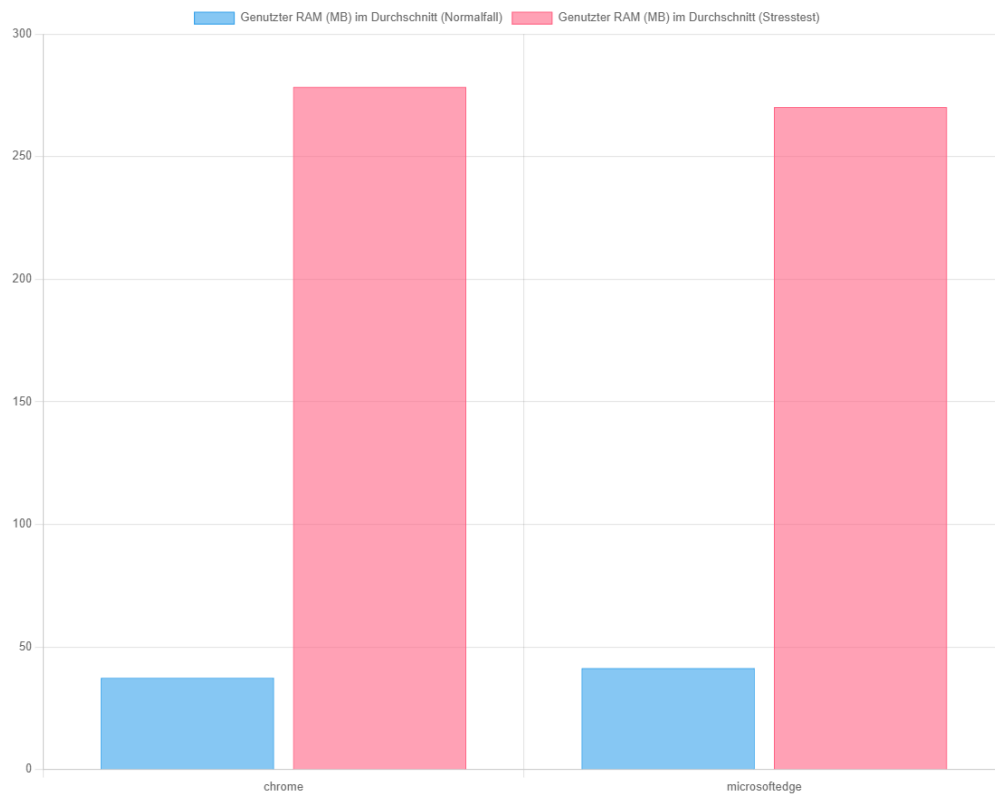


Abbildung 35: RAM-Nutzung während des Selenium Test (Google Maps JS API)

Quelle: Eigene Darstellung

Das Liniendiagramm visualisiert die Differenz in der CPU-Auslastung zwischen den Ereignissen „pageload“ und „tilesloaded“. Die ersten drei Tests in der X-Achse repräsentieren den Normalfall bzw. die normale Karte, während die nachfolgenden drei Tests den Stresstest abbilden (vgl. Abbildung 36). Die zuvor erwähnten Ereignisse wurden im Rahmen des Tests hinsichtlich der GPU-Temperatur und -Auslastung überwacht, um etwaige Anstiege zu illustrieren. Die gesamte CSV-Datei kann hier eingesehen werden:

https://github.com/geo1337/js_web_mapping_comparison/blob/main/google_maps_api/results_no_cache.csv

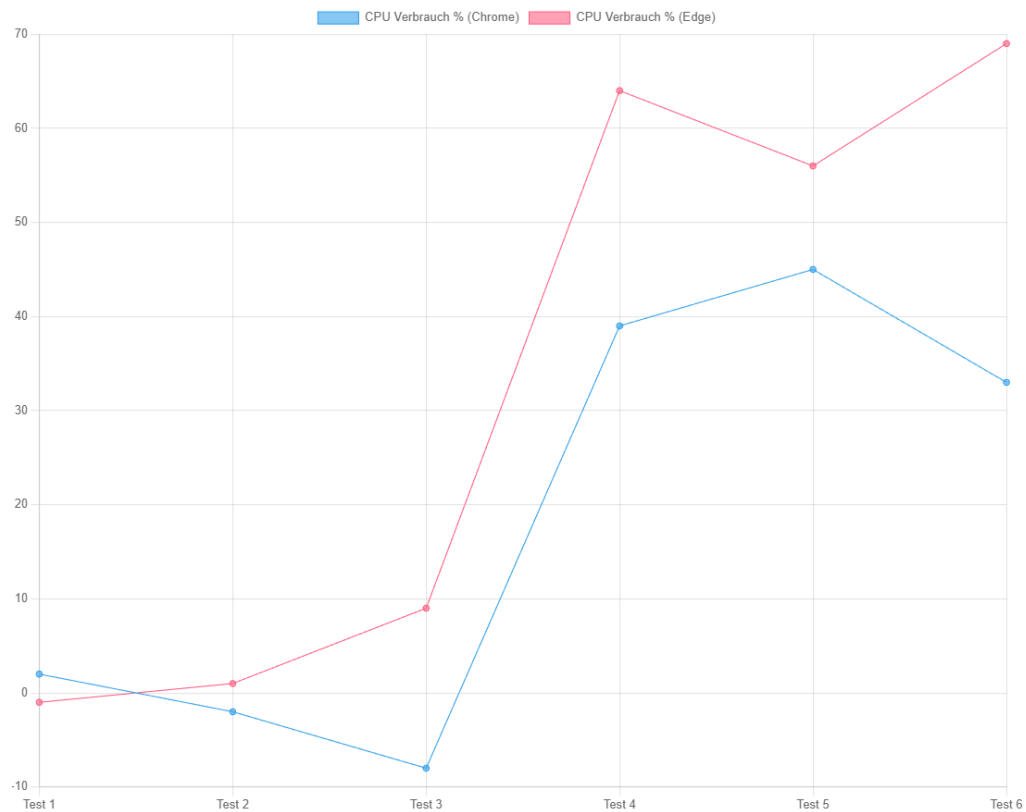


Abbildung 36: CPU-Auslastung während Selenium Tests (Google Maps JS API)

Quelle: Eigene Darstellung

4.1.5 Funktionalitäten

Die Google Maps JavaScript API verfügt standardmäßig über eine Vielzahl an Funktionalitäten. Es ist dem Endbenutzer ermöglicht, eigene Marker zu platzieren sowie den Inline-Inhalt des Markers zu bestimmen, beispielsweise ein SVG oder den Marker ganz mit einer eigenen Grafik zu ersetzen. Es besteht die Möglichkeit der Implementierung einer Polylinien⁹⁹ oder Polygone¹⁰⁰ zwischen zwei Markern oder beliebigen Koordinaten (vgl. Abbildung 37 und Abbildung 38).

⁹⁹ <https://support.esri.com/de-de/gis-dictionary/polyline>

¹⁰⁰ <https://support.esri.com/de-de/gis-dictionary/polygon>

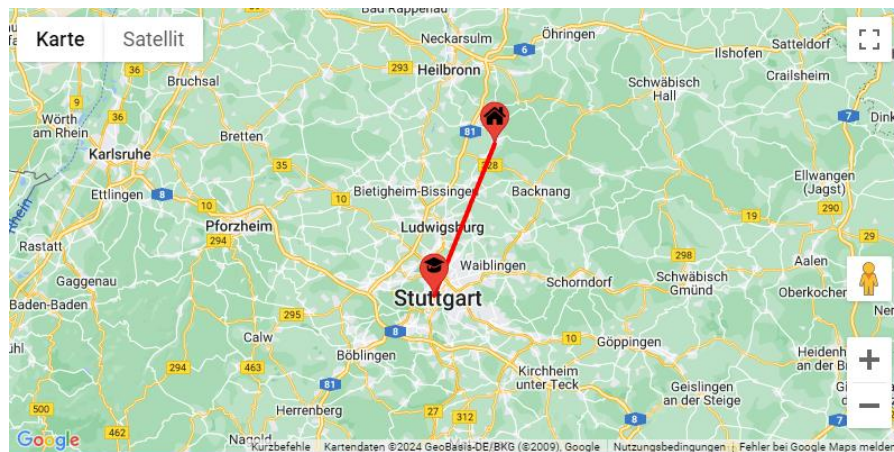


Abbildung 37: Google Maps inklusive Marker (svg inline) und einer Polyline.

Quelle: Eigene Darstellung



Abbildung 38: Google Maps mit einem Polygon

Quelle: Eigene Darstellung

In diesem Kontext können diverse Kartentypen ¹⁰¹ zum Einsatz kommen, darunter Straßenkarten, Satellitenbilder, Geländekarten sowie eine hybride Mischung aus herkömmlichen Karten und Satellitenbildern. Es gibt eine Reihe vordefinierter Elemente, darunter das „Infofenster“, das einem Marker zugeordnet ist und dessen Sichtbarkeit durch ein Klickereignis ausgelöst wird (vgl. Abbildung 39). Ein Alleinstellungsmerkmal der API ist die Integration von Street View. Damit wird den Nutzern ermöglicht, interaktiv 360-Grad-Ansichten von Straßen weltweit zu erkunden.

¹⁰¹ <https://developers.google.com/maps/documentation/javascript/maptypes?hl=de>



Abbildung 39: Google Maps Infenster

Quelle: Eigene Darstellung

4.1.6 Event-Handling

Ein weiterer wesentlicher Bestandteil der Funktionalitäten der Google Maps JavaScript API ist das Event-Handling. Diese Funktionen ermöglichen es Entwicklern, auf Interaktionen von Benutzern mit der Karte oder den darauf platzierten Markern zu reagieren. Als Beispiele können hier Klicks, Mouseover oder Änderungen der Kartenansicht genannt werden. Event Listener können sowohl einmalig als auch permanent an Google Maps-Objekte gebunden werden. Nach dem Eintreten eines Ereignisses besteht die Möglichkeit, diese auch per Code wieder zu entfernen. Die Events ermöglichen zudem die Erfassung der zugehörigen Längen- und Breitengrade. Die entsprechende Referenz dazu findet sich in der Dokumentation.¹⁰²

¹⁰² <https://developers.google.com/maps/documentation/javascript/reference?hl=de>

4.1.7 Unterstützung von Geodatenformaten

Darüber hinaus bietet die Google Maps JavaScript API Unterstützung für verschiedene Geodatenformate, dazu zählen GeoJSON und KML. Als Implementierungsbeispiel dient ein öffentlich gehostetes GeoJSON-Objekt, dessen Anfrage über die „fetch“-Methode in JavaScript erfolgt. Anschließend werden die Daten mit der Methode „addGeoJSON“ von der Google Maps JavaScript API zur Karte hinzugefügt (vgl. Abbildung 40).^{103 104} Des Weiteren wurde im Rahmen der vorliegenden Ausarbeitung, eine KML-Datei über die „fetch“-Methode abgefragt und anhand der Methode „KmlLayer“ der Karte hinzugefügt (vgl. Abbildung 41). Andere Geodateiformate werden standardmäßig nicht unterstützt und können somit nicht als Schicht über die Karte gelegt werden. Außer man konvertiert diese zu GeoJSON oder KML.¹⁰⁵



Abbildung 40: GeoJSON als Overlay Schicht (Google-Maps JS API).

Quelle: Eigene Darstellung

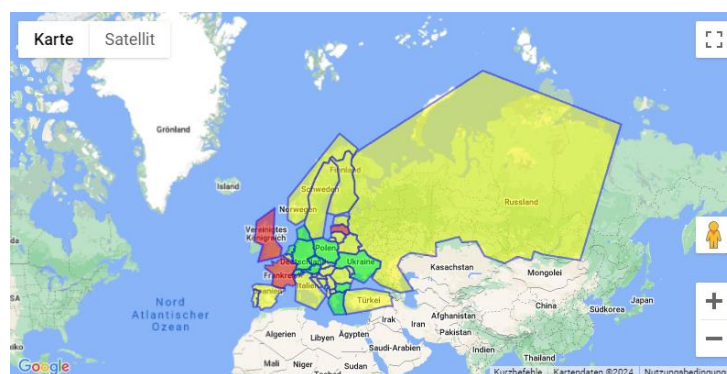


Abbildung 41: KML-Overlay als Schicht (Google Maps JS API)

Quelle: Eigene Darstellung

¹⁰³ https://raw.githubusercontent.com/isellsoap/deutschlandGeoJSON/main/1_deutschland/2_hoch.geo.json

¹⁰⁴ https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

¹⁰⁵ <https://developers.google.com/maps/documentation/javascript/layers>

https://github.com/geo1337/js_web_mapping_comparison/blob/main/google_maps_api/google_maps_kml_and_geojson.html

4.1.8 Unterstützung von Geodiensten

Die API unterstützt weder den WMS-Dienst noch den WFS-Dienst nativ.

4.1.9 Erweiterbarkeit

Die Google Maps JavaScript API bietet eine nahtlose Integration mit verschiedenen Google-Diensten, welche die Funktionalität und Benutzererfahrung erheblich erweitern können. Die entsprechenden Dienste müssen vorher in der Google Cloud Plattform aktiviert werden. Im Folgenden werden einige Dienste aufgeführt und erläutert. Die Google Directions API stellt einen Dienst zur Verfügung, der durch das Versenden von HTTP-Anfragen als Antwort Routen zwischen Standorten in Form von JSON- oder XML-Daten zurückgibt. Durch die Kombination der Google Directions API mit der Google Maps JavaScript API ist es Entwicklern möglich, eine Routenberechnung zu implementieren, die verschiedene Transportarten berücksichtigt.¹⁰⁶ Die Google Distance Matrix API stellt einen Dienst zur Berechnung von Entfernungen und Reisezeiten zwischen mehreren Ursprungs- und Zielpunkten dar. Die Anfrage erfolgt über HTTPS, wobei das Rückgabeformat aus JSON sowie XML besteht. Dies eröffnet Entwicklern die Möglichkeit, spezifische Zielgruppen anzusprechen, beispielsweise Reisende oder Logistikmitarbeiter, die eine Lieferung planen.¹⁰⁷ Die Google Maps Elevation API stellt einen Dienst bereit, der Höheninformationen für eine oder mehrere geographische Positionen liefert. Die Funktionalität erweist sich insbesondere für Anwendungen als nützlich, die topografische Daten benötigen, wie beispielsweise für die Bereiche Wandern, Stadtplanung oder Umweltstudien.¹⁰⁸ Bei der Google Maps Geocoding-API handelt es sich um einen Dienst, der die Konvertierung von Adressen in geografische Koordinaten (Geokodierung) und umgekehrt (Reverse-Geokodierung) ermöglicht.¹⁰⁹ Der Street View Dienst ermöglicht Entwicklern die Integration von 360°-Panoramabildern. Die Kombination des Street-View-Dienstes mit der Google Maps JavaScript API führt zu einer optimierten Benutzererfahrung. Dies kann sowohl im

¹⁰⁶ <https://developers.google.com/maps/documentation/directions/overview?hl=de>

¹⁰⁷ <https://developers.google.com/maps/documentation/distance-matrix/overview?hl=de>

¹⁰⁸ <https://developers.google.com/maps/documentation/javascript/elevation?hl=de>

¹⁰⁹ <https://developers.google.com/maps/documentation/geocoding/overview?hl=de>

Rahmen virtueller Touren als auch beim Betrachten von Sehenswürdigkeiten erfolgen.

¹¹⁰ Die Integration eingebetteter JSON-Stildekларationen in der Google Maps JavaScript API bedingt eine Modifikation der visuellen Darstellung von Karten. Diese Modifikation erlaubt es, bestimmte Elemente beispielsweise POIs ¹¹¹ auf der Karte hervorzuheben oder zu eliminieren, um eine spezifische Karte zu kreieren, welche auf eine Zielgruppe zugeschnitten ist. Die JSON-Syntax für die Definition von Stilen in der Google Maps JavaScript API ist simpel und erlaubt verschiedene Eigenschaften der Karte wie Farben, Linienstärken und Symbolformen anzupassen. Die aktuelle Version der Google Maps JS API unterstützt die Einbindung von JSON-Stildekларationen lediglich in der Google Cloud (vgl. Abbildung 42). Dazu muss der Karte einmalig eine Stil-ID im Backend zugeordnet werden, welche der Referenzierung des Stils dient. IDs lassen ebenfalls eine Auswahl der Kachelart zu. Dadurch kann die Google Maps JavaScript API mit WebGL ¹¹² verwendet werden oder gängigen Grafikbibliotheken wie Three.js ¹¹³. Dadurch resultieren unzählige Vorteile wie Tiefenverdeckung durch 3D-Gebäudegeometrie und das Synchronisieren von 3D- Inhalten mit der Basiskarte. ¹¹⁴ Als Beispiel für die praktische Anwendung kann der Digital Twin ¹¹⁵ aufgeführt werden.

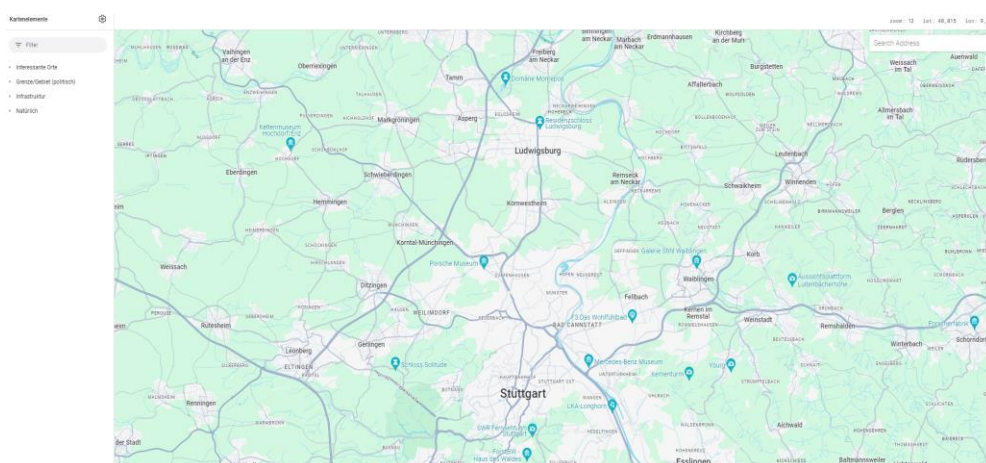


Abbildung 42: Stildekларation in der Cloud-Plattform

Quelle: Eigene Darstellung

¹¹⁰ <https://developers.google.com/maps/documentation/javascript/streetview>

¹¹¹ https://wiki.openstreetmap.org/wiki/Points_of_interest

¹¹² https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API?retiredLocale=de

¹¹³ <https://threejs.org/>

¹¹⁴ <https://developers.google.com/maps/documentation/javascript/webgl/webgl-overlay-view?hl=de>

¹¹⁵ <https://mapsplatform.google.com/resources/blog/webgl-powered-maps-features-now-generally-available/>

4.1.10 Interoperabilität

In Anbetracht der gegenwärtigen Relevanz und Verbreitung von JavaScript-Frameworks in der Webentwicklung wurde die Interoperabilität der Google Maps JavaScript API insbesondere in diesem Kontext untersucht (vgl. Tabelle 10). Durch Diesen Ansatz wurde sichergestellt, dass die API in verschiedenen Entwicklungsumgebungen funktioniert. Die API lässt sich in moderne JavaScript Frameworks integrieren, somit können alle Funktionalitäten für Webanwendungen übernommen werden. Als Implementierungsbeispiel wurde eine simple Karte in React¹¹⁶ erstellt, da bei diesem Framework die meisten Vorkenntnisse bestanden.

| Framework | Interoperabel |
|-------------------------|---------------|
| React | Ja |
| Vue.js ¹¹⁷ | Ja |
| Angular ¹¹⁸ | Ja |
| Ember.js ¹¹⁹ | Ja |
| Svelte ¹²⁰ | Ja |

Tabelle 10: Interoperabilität von der Google Maps API mit JS-Frameworks

Quelle: Eigene Darstellung

Der Code kann unter dieser URL auf Github eingesehen werden:

https://github.com/geo1337/js_web_mapping_comparison/blob/main/google_maps_api/react/App_2.js

¹¹⁶ <https://visgl.github.io/react-google-maps/>

¹¹⁷ <https://vue-map.netlify.app/>

¹¹⁸ <https://github.com/angular/components>

¹¹⁹ <https://github.com/sandydoo/ember-google-maps>

¹²⁰ <https://github.com/timhall/svelte-google-maps>

4.1.11 Entwicklungsgeschwindigkeit

Die Google Maps JavaScript API weist eine Reihe von Aspekten auf, welche die Entwicklung von Kartenanwendungen erheblich beschleunigen und vereinfachen können. Die API präsentiert eine umfangreiche Sammlung vordefinierter UI-Elemente. Entwickler können diese Elemente entweder direkt verwenden oder nach Belieben anpassen. Dies führt zu einer Reduktion des erforderlichen Codes. Des Weiteren übernimmt die API die Verarbeitung komplexer Geodaten. Dies resultiert in einer Reduktion des Zeitaufwands im Vergleich zur manuellen Implementierung dieser Funktionen. Die vordefinierten Event-Listener ermöglichen es Entwicklern, Zeit bei der Implementierung interaktiver Funktionen einzusparen. Die Anbindung an die Google Cloud Plattform ermöglicht die einfache Skalierung und Bereitstellung von Kartenanwendungen, um hohen Traffic und schwankende Anforderungen zu bewältigen. Des Weiteren verfügt die API über eine umfassende Dokumentation, in der detaillierte Anleitungen sowie Code-Beispiele enthalten sind. Zudem ist eine große aktive Community vorhanden, die bei der Behebung von Fehlern oder bei der Beantwortung offener Fragen hilfreich sein kann.¹²¹ Darüber hinaus sind mobile Touchscreen-Interaktionen wie das Verschieben per Ziehen der Karte oder das Zoomen mit zwei Fingern, standardmäßig in der API enthalten. Die Elemente der Karte sind per CSS zugänglich, so können Entwickler die Karte an verschiedene Bildschirmgrößen anpassen. Hinzukommend erfolgt die Lokalisierung der Spracheinstellungen des Browsers automatisch durch die API, sodass Entwickler diese Funktionalität ohne eigene Implementierung nutzen können. Die Verwaltung von API-Schlüsseln sowie die Implementierung von Zugriffsbeschränkungen in der Cloud-Plattform ermöglichen die Abdeckung von Sicherheitsaspekten. Dies entlastet Entwickler, die keine Sicherheitslösungen selbst konzipieren und implementieren müssen. Die tatsächliche Entwicklungsgeschwindigkeit ist von verschiedenen Faktoren abhängig, wie beispielsweise von der Komplexität der Karte sowie den Vorkenntnissen des Entwicklers.

¹²¹ <https://developers.google.com/maps/developer-community?hl=de>

4.1.12 Dokumentation

Die Dokumentation der Google Maps JS API ist äußerst umfangreich und beinhaltet sowohl Schritt-für-Schritt-Anleitungen als auch Best-Practice-Beispiele. Zur Erlangung aller Dokumentationsseiten wurde die folgende Suchanfrage verwendet: „site:https://developers.google.com/maps/documentation/javascript“. Die Suchfunktion von Google ermöglicht die Eingrenzung der Suche auf eine bestimmte Domain. Die Recherche ergab 1.530 Ergebnisse, wobei bei der Suche mehrere Varianten von Seiten in verschiedenen Sprachen enthalten sind. Dies unterstreicht die Breite der Dokumentation.¹²²

4.1.13 Community-Aktivität

Die Statistik zu Stack Overflow basiert auf der Suchanfrage nach „Google Maps“. Dabei werden auch Synonyme wie „gmaps“ oder „Google Maps API“ miteinbezogen. Die Anzahl der Repositories auf GitHub reflektiert sämtliche Repositories, in denen JavaScript mitenthalten ist. Da YouTube keine Anzahl der Suchergebnisse bereitstellt wurde die folgende Suchanfrage in Google verwendet „site:youtube.com intitle:Google Maps JavaScript API“ (vgl. Tabelle 11). Die Plattform Google Maps ist auf GitHub vertreten, um die Entwicklung von Community-Forks zu fördern. Es existiert ein offizieller YouTube-Kanal der Google Maps Plattform, auf dem regelmäßiger Tutorials oder Best Practices Beispiele veröffentlicht werden. Des Weiteren existiert ein öffentlicher offizieller Discord-Server, auf dem sich die Community vernetzen und austauschen kann.¹²³

| Plattform | Aktivität | Details |
|----------------|----------------------------|---------|
| Stack Overflow | Fragen | 66.000+ |
| Github | Repositories | 16.500+ |
| Issue Tracker | Gemeldete und gelöste Bugs | 2500+ |
| YouTube | Tutorials | 5000+ |
| Discord | Mitglieder | 8000+ |

Tabelle 11: Community-Aktivität (Google Maps JS API)

Quelle: Eigene Darstellung

¹²² <https://developers.google.com/search/docs/monitor-debug/search-operators/all-search-site?hl=de>

¹²³ <https://developers.google.com/maps/developer-community?hl=de>

4.1.14 Kosten und Lizenzierung

Die Google Maps JavaScript API verwendet ein nutzungsbasiertes Preismodell, bei dem die Kosten auf der Anzahl der Anfragen und der Art der API-Aufrufe basieren. Google bietet ein monatliches kostenfreies Kontingent an API-Anfragen, was besonders für kleine Anwendungen oder Entwicklungs- und Testzwecke nützlich ist. Dieses Kontingent ermöglicht es Entwicklern, die API ohne anfängliche Kosten auszuprobieren. Hierbei steht ein Kontingent von 200 \$ zur Verfügung, was laut Google selbst 28.500 API-Aufrufen entspricht, andere API-Dienste sind inbegriffen (Stand: Juni 2024).¹²⁴ Allerdings muss man ein Zahlungsmittel hinterlegen, welches lediglich eine Kreditkarte, Bankkonto oder PayPal umfasst. Die verfügbaren Zahlungsmittel sind Standort abhängig.¹²⁵ ¹²⁶ Beim Erreichen des Kontingentlimits kosten tausend weitere Aufrufe 7\$. Bei einer Abrechnung werden die entstandenen Kosten in die Landeswährung umgerechnet. Die Kosten variieren je nach Art der API-Aufrufe, spezielle Funktionen wie Geocoding, Places API oder Directions API können höhere Kosten verursachen als einfache Kartenaufrufe. Es besteht die Möglichkeit, Nutzungskontrollen und Budgetalarme in der Google Cloud Plattform einzurichten. Auf diese Weise lässt sich die monatliche Nutzung überwachen und eine Warnung wird generiert, sobald das kostenlose Kontingent überschritten wurde. Für Unternehmen mit einem hohen Nutzungsvolumen bietet Google Rabatte an, sodass individuelle Preisvereinbarungen getroffen werden können. Mit der Nutzung geht man eine rechtliche Vereinbarung mit Google ein. Die Referenz hierzu sind die Allgemeinen Nutzungsbedingungen¹²⁷

4.1.15 Datenschutz und Sicherheit

Es wird davon abgeraten, den API-Schlüssel direkt im Quellcode zu integrieren, da dieser von Dritten eingesehen werden kann. Daher wird empfohlen, Konfigurationsdateien außerhalb des Quellcodes zu erstellen, sogenannte Module. Diese sollten serverseitig abgelegt und anschließend importiert werden, um auf die Variable des API-Schlüssels zugreifen zu können. Zusätzlich sollten tägliche Zugriffslimits festgelegt und Budgetwarnungen eingerichtet werden. Gemäß Abschnitt

¹²⁴ https://mapsplatform.google.com/pricing/?hl=de&_gl=1

¹²⁵ <https://support.google.com/cloudidentity/answer/1230192?hl=DE>

¹²⁶ <https://support.google.com/cloudidentity/answer/2380700>

¹²⁷ https://cloud.google.com/maps-platform/terms?_gl

4.4.1 der Allgemeinen Geschäftsbedingungen sammelt Google und erhält Daten von Kunden und Endbenutzern, wie beispielsweise Suchbegriffe, IP-Adressen und geografische Koordinaten. Laut Abschnitt 4.4.3 a) legt Google Cookies auf dem Gerät der Endnutzer ab und kann auf andere Cookies zugreifen. Dies impliziert, dass die Anwendung, welche die Google Dienste enthält, DSGVO-konform sein muss.¹²⁸

¹²⁸ https://cloud.google.com/maps-platform/terms?_gl

4.2 Leaflet

4.2.1 Nutzung und Integration

Im Gegensatz zur Google Maps JS API handelt es sich bei Leaflet um eine Open-Source-Lösung. Es ist weder die Einrichtung eines Benutzerkontos noch die Hinterlegung eines Abrechnungskontos erforderlich. Die Einbindung der API erfolgt hierbei im Head des HTML-Dokuments. Dazu wird zunächst ein Link zum Stylesheet integriert. Anschließend wird ein Script-Tag mit einem Link zum JavaScript-File eingefügt.¹²⁹ Da Leaflet standardmäßig keine eigene Karte bzw. keinen eigenen Layer anbietet, wird für die wissenschaftliche Arbeit die Karte von Basemap und OpenStreetMap verwendet.^{130 131}

4.2.2 Ladegeschwindigkeit

Die Leaflet-Bibliothek bietet das „load“-Event, welches erst dann ausgeführt wird, wenn alle sichtbaren Kacheln vollständig geladen worden sind. Infolgedessen war die Implementierung eines ähnlichen Codes wie bei der Google Maps API möglich. Das „load“-Event wird dabei auf die Schicht gelegt in welche die Kacheln geladen werden (vgl. Tabelle 12). Sowohl das Stylesheet als auch die JavaScript-Datei werden clientseitig auf der Festplatte gecached, was zu einer Reduzierung der Ladezeiten der Karte bzw. Kacheln führt (vgl. Tabelle 13 und Tabelle 15). Durch das Caching von Ressourcen werden wiederholte Seitenaufrufe beschleunigt, da diese nicht erneut vom Server abgerufen werden müssen.¹³² Der Ansatz per JavaScript bezieht sich auf die Kacheln innerhalb der Schicht im Leaflet-Container, wobei die Entwicklerkonsolen die Gesamtheit erfassen bzw. messen, sprich Skripte, Stylesheets etc. inbegriffen. Die Spalte Anzahl der Ressourcen bezieht sich dabei auf die Anzahl der geladenen Kacheln. Es sei drauf verwiesen, dass die Ergebnisse durch Schwankungen der Internetgeschwindigkeit variieren können, sowie durch die Systemauslastung. Der zugehörige Quellcode kann unter der folgenden URL eingesehen werden:

¹²⁹ <https://leafletjs.com/examples/quick-start/>

¹³⁰ <https://basemap.de/>

¹³¹ <https://openstreetmap.de/>

¹³² Vgl. Redaktion, I. (2022). Was ist Caching? IONOS Digital Guide.
<https://www.ionos.de/digitalguide/hosting/hosting-technik/was-ist-caching/>

https://github.com/geo1337/js_web_mapping_comparison/blob/main/leaflet/index_leaflet_normal.html

| Browser | Benötigte Zeit in (ms) | Anzahl der Ressourcen |
|-----------------|------------------------|-----------------------|
| Google Chrome | 164.80 | 9 |
| Microsoft Edge | 148.30 | 9 |
| Mozilla Firefox | 178.00 | 9 |
| Opera | 162.70 | 9 |

Tabelle 12: Messungen per JavaScript-Code ohne Cache (Leaflet + OSM)

Quelle: Eigene Darstellung

| Browser | Benötigte Zeit in (ms) | Anzahl der Ressourcen |
|-----------------|------------------------|-----------------------|
| Google Chrome | 14.00 | 9 |
| Microsoft Edge | 11.70 | 9 |
| Mozilla Firefox | 7.00 | 9 |
| Opera | 10.40 | 9 |

Tabelle 13: Messungen per JavaScript-Code mit Cache (Leaflet + OSM)

Quelle: Eigene Darstellung

| Browser | Ladevorgang (ms) | Fertigstellen (ms) |
|-----------------|------------------|--------------------|
| Google Chrome | 369 | 368 |
| Microsoft Edge | 262 | 261 |
| Mozilla Firefox | 277 | 278 |
| Opera | 243 | 244 |

Tabelle 14: Messungen aus den Entwicklerkonsolen ohne Cache (Leaflet + OSM)

Quelle: Eigene Darstellung

| Browser | Ladevorgang (ms) | Fertigstellen (ms) |
|-----------------|------------------|--------------------|
| Google Chrome | 57 | 68 |
| Microsoft Edge | 67 | 79 |
| Mozilla Firefox | 47 | 69 |
| Opera | 63 | 84 |

Tabelle 15: Messungen aus den Entwicklerkonsolen mit Cache (Leaflet + OSM)

Quelle: Eigene Darstellung

Die Ergebnisse der Metrik Ladegeschwindigkeit, weisen bei Leaflet eine deutlich bessere Performance auf. Dies ist unter anderem auf die geringe Größe von Leaflet zurückzuführen, welche lediglich 42 KB ¹³³ beträgt. Es sei jedoch darauf hingewiesen, dass die Ladegeschwindigkeit nicht alleinig durch Leaflet bestimmt wird. Vielmehr kann diese durch die Auswahl des Servers, welcher die Karte bereitstellt, variieren. Zur Ausführung des Selenium-Testskripts war ein kleiner Workaround erforderlich. Dazu wird in der Index-Datei dem Body ein Attribut hinzugefügt, sobald das „load“-Ereignis eintrifft. Dieses Ereignis wird asynchron abgefangen und repräsentiert das Ende des Ladevorgangs. Die übrigen Messungen werden gemäß der im vorherigen Kapitel dargestellten Vorgehensweise durchgeführt (vgl. Abbildung 44). Das Scripting und Rendering nehmen deutlich weniger Zeit in Anspruch (vgl. Abbildung 43).

https://github.com/geo1337/js_web_mapping_comparison/blob/main/leaflet/test_leaflet.js

Gesamten Ergebnisse des Tests können hier eingesehen werden:

https://github.com/geo1337/js_web_mapping_comparison/blob/main/leaflet/results_no_cache_leaflet.csv

¹³³ <https://leafletjs.com/>

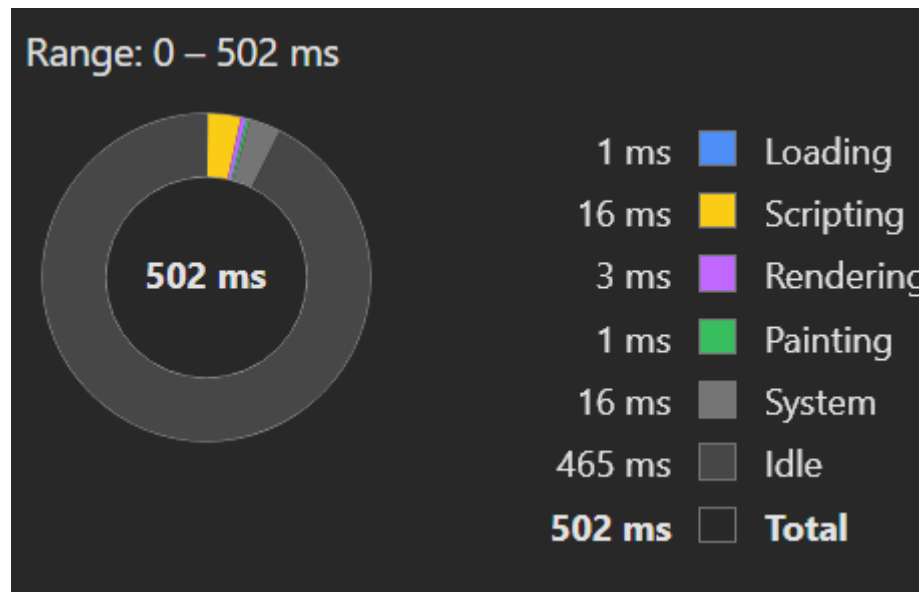


Abbildung 43: Google Chrome Performance-Analyse (Leaflet + OSM)

Quelle: Eigene Darstellung

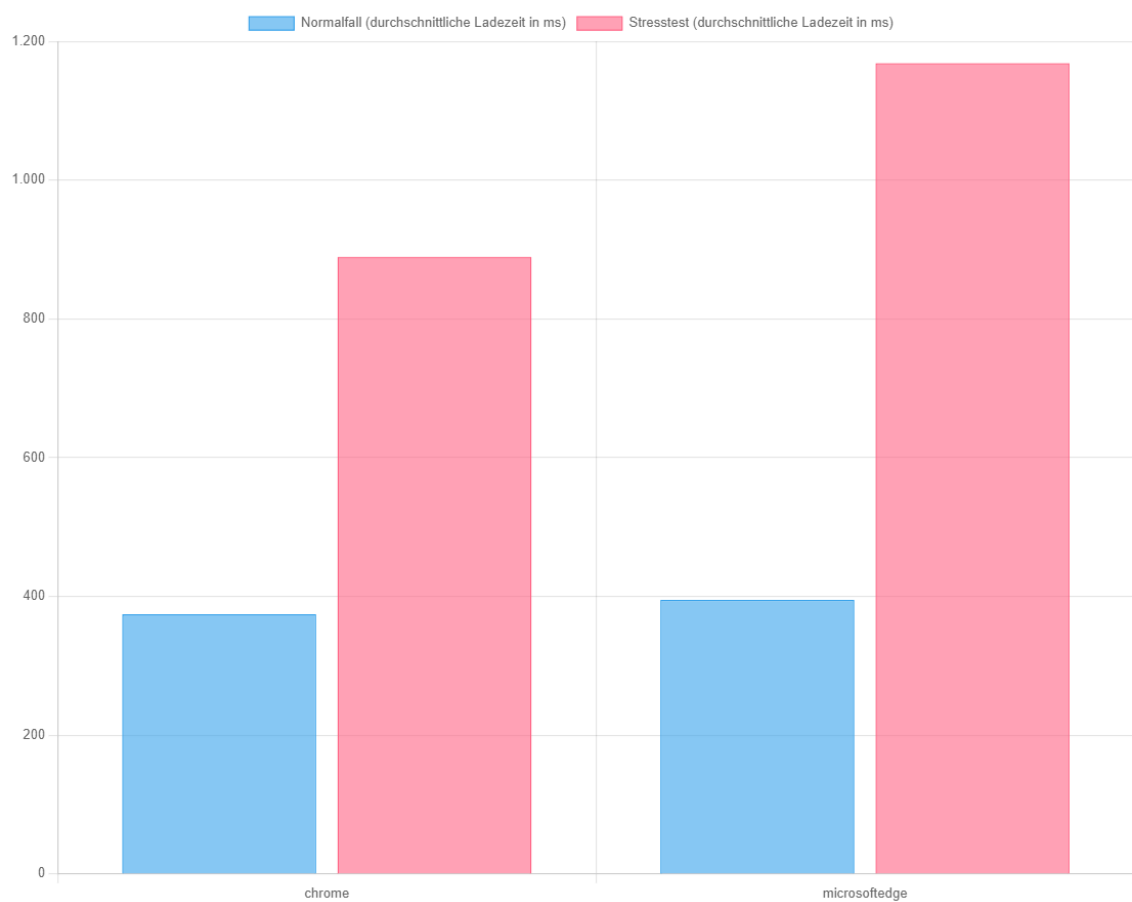


Abbildung 44: Messungen des Selenium Tests in Chart.js visualisiert (Leaflet + OSM)

Quelle: Eigene Darstellung

Die Ladegeschwindigkeit der Karte ist nicht alleinig von der Bibliothek abhängig, sondern wird maßgeblich vom Webserver von OpenStreetMap (<https://tile.openstreetmap.org/>) beeinflusst. Im Rahmen eines Tests wurde ein Plan in Apache JMeter simuliert, der 50 parallele Anfragen an den Server sendet, um eine zufällige Kachel zu erhalten (vgl. Abbildung 45). Die Anfrage-URL beinhaltete am Ende die Parameter für z, x und y. Die Abkürzung „Z“ steht dabei für die Zoomstufe, „x“ für die X-Koordinate und „y“ für die Y-Koordinate. Die zufällige Wahl der Werte diente der Generierung einer Kachel als Antwort (siehe Anhang B).

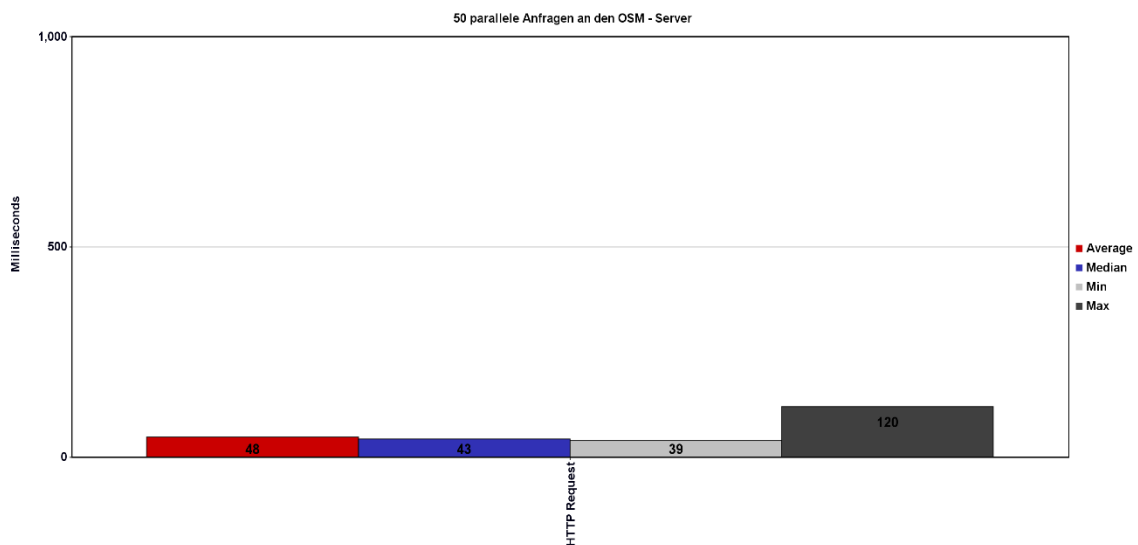


Abbildung 45: 50 parallele HTTP-Anfragen an den Server von OSM

Quelle: Eigene Darstellung

Auch auf mobilen Endgeräten weist Leaflet eine schnellere Ladegeschwindigkeit auf (vgl. Abbildung 46).

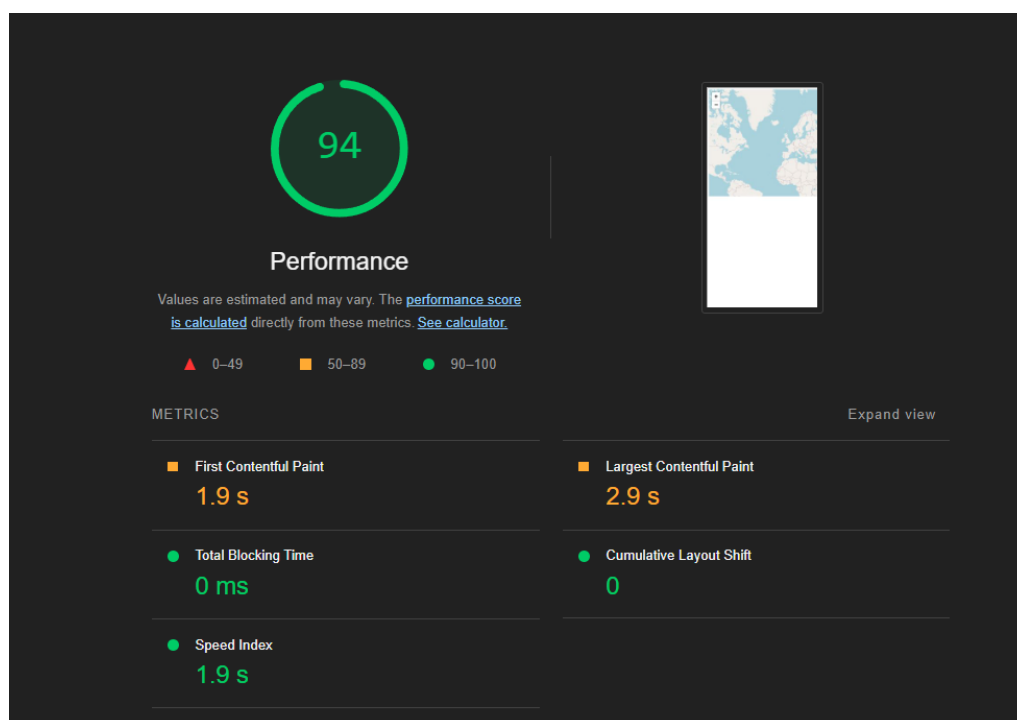


Abbildung 46: Google Lighthouse Test (Leaflet + OSM)

Quelle: Eigene Darstellung

4.2.3 Reaktionsgeschwindigkeit

Unter Verwendung der Ereignisse „zoomstart“ und „zoomend“¹³⁴ in Kombination mit dem „load“-Event kann ein JavaScript-Code implementiert werden, der die Dauer des Zoomvorgangs sowie das Nachladen der Kacheln misst. Dabei wird jede Zoominteraktion gemessen und in der Konsole ausgegeben (vgl. Abbildung 47).

```
Zoom-Nummer 2 - Alle Kacheln der Karte beim Zoomen geladen in: 128.10 millisekunden index\_leaflet.html:71
Zoom-Nummer 2 Zoom-Dauer: 254.10 millisekunden index\_leaflet.html:65
Zoom-Nummer 3 - Alle Kacheln der Karte beim Zoomen geladen in: 142.30 millisekunden index\_leaflet.html:71
Zoom-Nummer 3 Zoom-Dauer: 252.40 millisekunden index\_leaflet.html:65
Zoom-Nummer 4 - Alle Kacheln der Karte beim Zoomen geladen in: 200.10 millisekunden index\_leaflet.html:71
Zoom-Nummer 4 Zoom-Dauer: 251.80 millisekunden index\_leaflet.html:65
Zoom-Nummer 5 - Alle Kacheln der Karte beim Zoomen geladen in: 174.10 millisekunden index\_leaflet.html:71
Zoom-Nummer 5 Zoom-Dauer: 251.90 millisekunden index\_leaflet.html:65
Zoom-Nummer 6 - Alle Kacheln der Karte beim Zoomen geladen in: 75.70 millisekunden index\_leaflet.html:71
Zoom-Nummer 6 Zoom-Dauer: 252.10 millisekunden index\_leaflet.html:65
```

Abbildung 47: Konsolenausgabe im Browser beim Zoomen (Leaflet + OSM)

Quelle: Eigene Darstellung

¹³⁴ <https://leafletjs.com/reference.html#map-event>

https://github.com/geo1337/js_web_mapping_comparison/blob/main/leaflet/leaflet_reactionspeed.html

Die Nutzung der Events „dargstart“ und „dragend“ in Kombination mit dem „load“-Event ermöglicht die Messung der Zeit, die beim Verschieben der Karte benötigt wird. Diese Messung erfolgt bei jedem Verschiebungsvorgang und die Zeit zum Nachladen der Kacheln wird in der Konsole angezeigt (vgl. Abbildung 48).

```
Drag-Nummer 4 - Alle Kacheln der Karte beim Verschieben geladen in: 498.70 millisekunden index\_leaflet.html:94  
Drag-Nummer 5 - Alle Kacheln der Karte beim Verschieben geladen in: 528.40 millisekunden index\_leaflet.html:94  
Drag-Nummer 6 - Alle Kacheln der Karte beim Verschieben geladen in: 248.80 millisekunden index\_leaflet.html:94  
Drag-Nummer 7 - Alle Kacheln der Karte beim Verschieben geladen in: 243.60 millisekunden index\_leaflet.html:94  
Drag-Nummer 8 - Alle Kacheln der Karte beim Verschieben geladen in: 224.60 millisekunden index\_leaflet.html:94  
Drag-Nummer 9 - Alle Kacheln der Karte beim Verschieben geladen in: 724.90 millisekunden index\_leaflet.html:94  
Drag-Nummer 10 - Alle Kacheln der Karte beim Verschieben geladen in: 361.00 millisekunden index\_leaflet.html:94
```

Abbildung 48: Konsolenausgabe beim Verschieben der Karte (Leaflet + OSM)

Quelle: Eigene Darstellung

Eine simple Leaflet-Karte wurde ebenfalls dem automatisierten Selenium-Test unterzogen, der zehn Klicks simuliert und anschließend den Durchschnitt der Ladezeit der Kacheln beim Zoomen berechnet (vgl. Abbildung 49).

```
Der "Vergrößern"-Button wurde fünfmal angeklickt.  
Der "Verkleinern"-Button wurde fünfmal angeklickt.  
Durchschnittliche Ladezeit beim Zoomen: 232.32 ms  
Der Selenium-Test wurde erfolgreich beendet.
```

Abbildung 49: Konsolenausgabe des Seleniumtests zur Reaktionsgeschwindigkeit (Leaflet + OSM)

Quelle: Eigene Darstellung

4.2.4 Ressourcenverbrauch

Die durchgeführte Leistungsanalyse in Google Chrome sowie eigene Messungen belegen, dass die Ressourceneffizienz von Leaflet im Vergleich zu anderen JavaScript-Bibliotheken höher ist (vgl. Abbildung 50 und Abbildung 51).

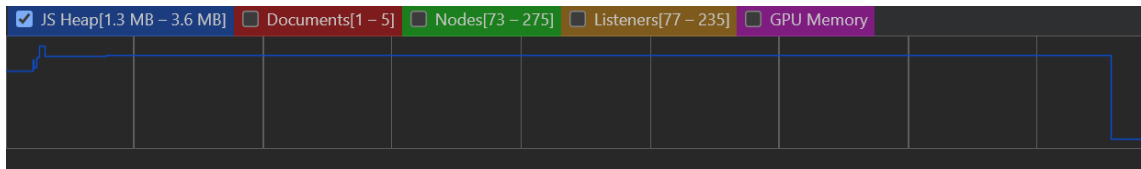


Abbildung 50: Arbeitsspeicher-Nutzung von Leaflet

Quelle: Eigene Darstellung

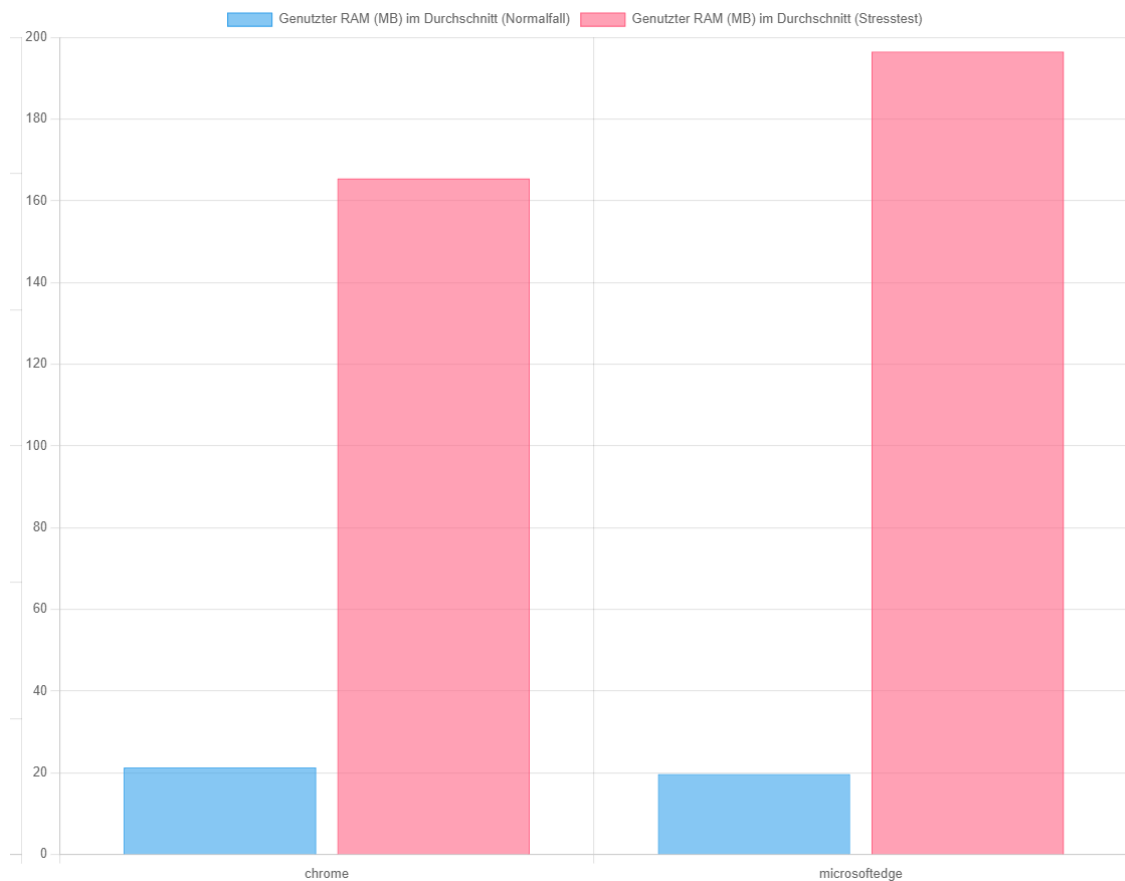


Abbildung 51: Ram-Nutzung während des Selenium Tests (Leaflet + OSM)

Quelle: Eigene Darstellung

4.2.5 Funktionalitäten

Leaflet verfügt standradmäßig über die Möglichkeit der Schichtverwaltung. Dies bedeutet, dass mehrere Schichten per Kontrollschaltfläche dynamisch ein- und ausgeblendet werden können (vgl. Abbildung 52).

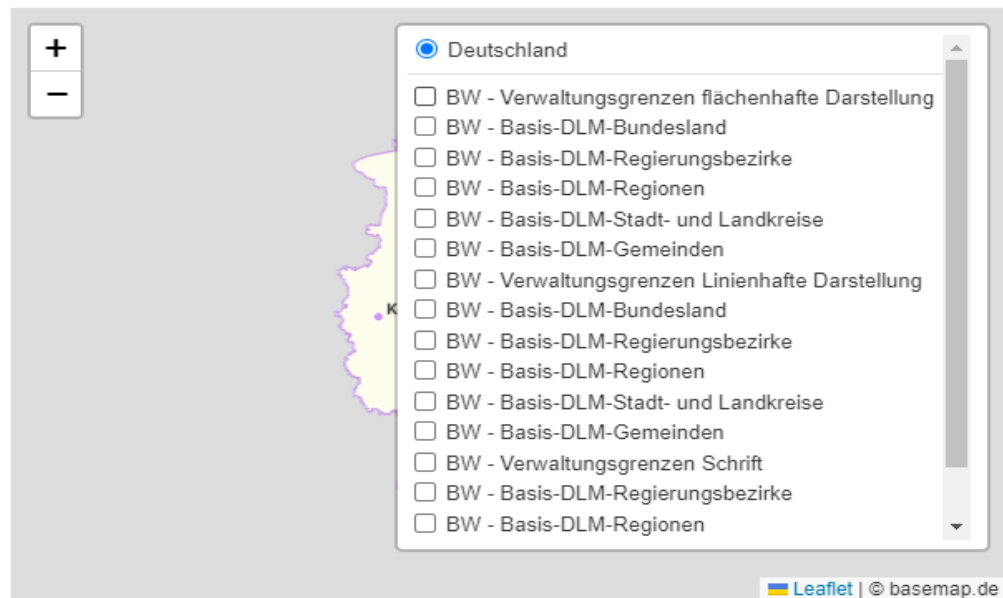


Abbildung 52: Beispiel für eine Schichtverwaltung (Leaflet)

Quelle: Eigene Darstellung

Leaflet bietet benutzerdefinierte Markierungen auf Karten, welche mit Pop-ups verbunden werden können. Diese bieten zusätzliche Informationen oder Interaktionsmöglichkeiten(vgl. Abbildung 53).^{135 136}

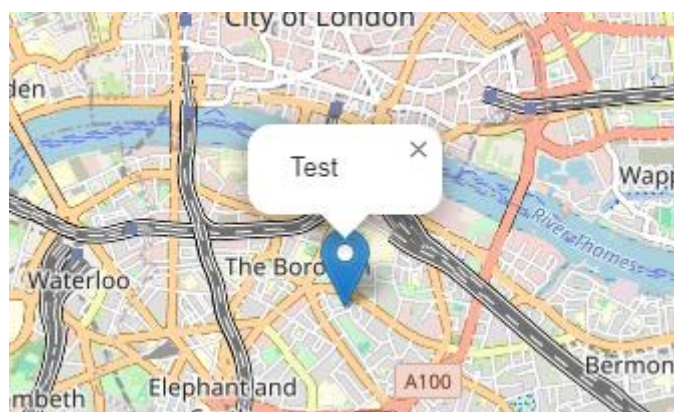


Abbildung 53: Marker und Infofenster in Leaflet

Quelle: Eigene Darstellung

¹³⁵ <https://leafletjs.com/reference.html#marker>

¹³⁶ <https://leafletjs.com/reference.html#popup>

Geometrische Formen werden ebenfalls unterstützt, somit können Bereiche auf der Karte mit Kreisen oder Polygonen markiert werden.^{137 138}

4.2.6 Event-Handling

Es wird seitens der API ein umfangreiches Ereignis-Handling bereitgestellt, welches eine Reaktion auf Benutzerinteraktionen wie Klicks, Mauszeigerpositionierungen und andere Ereignisse ermöglicht.¹³⁹

4.2.7 Unterstützung von Geodateformaten

Die Unterstützung von Geodateiformaten wie beispielsweise GeoJSON ist ebenfalls geboten.¹⁴⁰ KML wird standardmäßig nicht unterstützt, um dies zu realisieren muss ein Plugin verwendet werden. Zum Einsatz bei dieser wissenschaftlichen Arbeit kam das Plugin „Leaflet-omnivore“.¹⁴¹ Das Plugin extrahiert die Geodaten aus der KML und konvertiert sie in ein GeoJSON-Format (vgl. Abbildung 54 und 55).



Abbildung 54: GeoJSON in Leaflet

Quelle: Eigene Darstellung

¹³⁷ <https://leafletjs.com/reference.html#circlemarker>

¹³⁸ <https://leafletjs.com/reference.html#polygon>

¹³⁹ <https://leafletjs.com/reference.html#map-event>

¹⁴⁰ <https://leafletjs.com/reference.html#geojson>

¹⁴¹ <https://github.com/mapbox/leaflet-omnivore>

https://github.com/geo1337/js_web_mapping_comparison/blob/main/leaflet/leaflet_geojson.html

https://github.com/geo1337/js_web_mapping_comparison/blob/main/leaflet/leaflet_kml.html

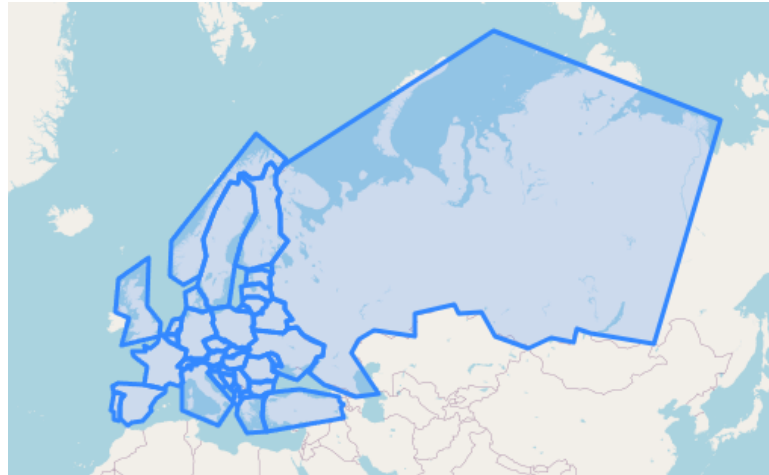


Abbildung 55: KML in Leaflet

Quelle: Eigene Darstellung

4.2.8 Unterstützung von Geodiensten

Im Unterschied zur Google Maps JavaScript API bietet Leaflet standardmäßig die Integration der WMS- und WFS-Dienste an. Hierbei können diverse Quellen für geografische Daten herangezogen werden. Dies erfolgt über eine HTTP-Anfrage an einen entsprechenden Server. Dabei sind bestimmte Parameter zu berücksichtigen, um den Standard einzuhalten, darunter (Service, Version, Request, Format). Es gibt eine Vielzahl von Institutionen und Organisationen, die einen WMS-Dienst offerieren. Für diese wissenschaftliche Arbeit wurde exemplarisch der WMS-Dienst des Landesamts für Geoinformation und Landesentwicklung Baden-Württemberg herangezogen.¹⁴² ¹⁴³ Im nachfolgenden Beispiel wurde eine HTTPS-Anfrage an den Server des Landesamts für Geoinformation und Landesentwicklung Baden-Württemberg mittels der „fetch“-Methode versendet. Anschließend wurden die Schichten aus der Antwort des Servers in das Steuerelement von Leaflet übergeben. Dadurch ist es möglich, die einzelnen Schichten dynamisch und beliebig zu setzen.

¹⁴² <https://www.lgl-bw.de/Produkte/Open-Data/#Geodatendienste>

¹⁴³ <https://metadaten.geoportal-bw.de/geonetwork/srv/ger/catalog.search#/metadata/a9fe7ea2-8cdd-3505-d219-020aca274536>

Das folgende Beispiel demonstriert die Nutzung des WMS-Protokolls in Leaflet und veranschaulicht die Verwaltungsgrenzen von Baden-Württemberg (vgl. Abbildung 56).

https://github.com/geo1337/js_web_mapping_comparison/blob/main/leaflet/index_leaflet_wms.html

https://github.com/geo1337/js_web_mapping_comparison/blob/main/leaflet/index_leaflet_wfs.html



Abbildung 56: WMS-Antwort in Leaflet

Quelle: Eigene Darstellung

Die WFS-Abfrage erfolgte nach demselben Prinzip. Zu diesem Zweck wurde der WFS-Dienst des Deutschen Wetterdienstes ¹⁴⁴ in Anspruch genommen. Dadurch konnten diverse Vektordaten in die Karte eingefügt werden, wie beispielsweise Warnungen vor dem Hochwasser für bestimmte Landkreise (Stand: 03.06.2024) (vgl. Abbildung 57).

¹⁴⁴ <https://www.dwd.de/DE/leistungen/geodienste/geodienste.html>

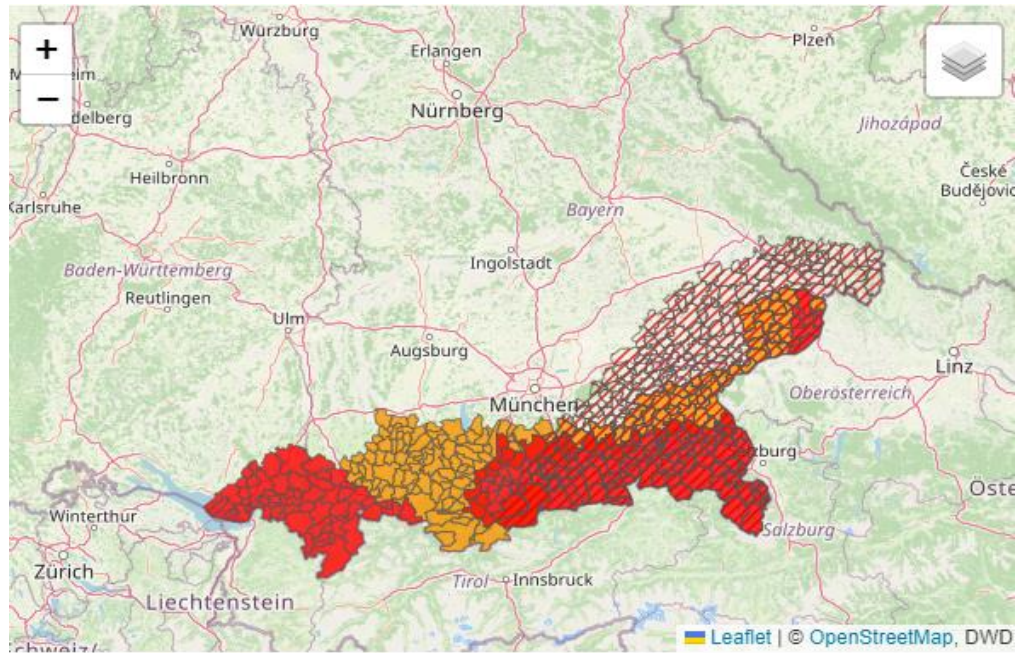


Abbildung 57: WFS-Antwort in Leaflet

Quelle: Eigene Darstellung

4.2.9 Erweiterbarkeit

Leaflets Kernfunktionalitäten lassen sich durch das breitgefächerte Plugin-Ökosystem¹⁴⁵ erweitern, darunter Routing, Geokodierung, Datenvisualisierung und Marker-Cluster. Da es sich um Open-Source-Software handelt, kann dieser von der Community weiterentwickelt werden. Jedes Plugin wird normalerweise von einem eigenen Repository auf Plattformen wie GitHub gehostet, was es Entwicklern ermöglicht, Beiträge zu leisten und Probleme zu melden. Dies fördert Transparenz, Zusammenarbeit und kontinuierliche Verbesserung der Leaflet-Plugins.

¹⁴⁵ <https://leafletjs.com/plugins.html>

4.2.10 Interoperabilität

Ein weiterer ausschlaggebender Aspekt in der Webentwicklung ist die Interoperabilität der Bibliothek Leaflet und moderner JavaScript-Frameworks, so lassen sich Webanwendungen mit kartografischen Funktionen entwickeln (vgl. Tabelle 16). Im Rahmen der wissenschaftlichen Arbeit wurde Leaflet in React ¹⁴⁶ implementiert. Leaflet unterstützt die effiziente, komponentenbasierte Entwicklung der Frameworks. ^{147 148} Die umfassende Interoperabilität von Leaflet ist der Beteiligung und Zusammenarbeit innerhalb der Entwicklergemeinschaft zu verdanken.

| Framework | Interoperabel |
|---------------------------|---------------|
| React | Ja |
| Vue.js ¹⁴⁹ | Ja |
| Angular ¹⁵⁰ | Ja |
| Ember.js ¹⁵¹ | Ja |
| Svelte ^{152 153} | Ja |

Tabelle 16: Interoperabilität von der Google Maps API mit JS-Frameworks

Quelle: Eigene Darstellung

4.2.11 Entwicklungsgeschwindigkeit

Die aktive Community von Leaflet kann die Entwicklungsgeschwindigkeit beschleunigen, indem Bugs schneller behoben oder neue Funktionen schneller implementiert werden. In diesem Kontext sei auf die im Abschnitt „Identifikation führender Web-Mapping-Lösungen“ erwähnten Kennzahlen für die aktive Community verwiesen. Ein weiterer wichtiger Aspekt, der zur beschleunigten

¹⁴⁶ <https://react.dev/>

¹⁴⁷ <https://react-leaflet.js.org/>

¹⁴⁸ <https://github.com/PaulLeCam/react-leaflet>

¹⁴⁹ <https://vue2-leaflet.netlify.app/>

¹⁵⁰ <https://www.npmjs.com/package/@asymmetrik/ngx-leaflet>

¹⁵¹ <https://miguelsobain.github.io/ember-leaflet/>

¹⁵² <https://github.com/ngyewch/svelte-leafletjs>

¹⁵³ <https://www.npmjs.com/package/@types/svelte-leafletjs>

Entwicklungsgeschwindigkeit beiträgt, ist die umfassende Dokumentation der API. Es sind zahlreiche Beispiele und ausführliche Erläuterungen enthalten.¹⁵⁴

Die meisten der erwünschten Funktionalitäten sind durch die auf GitHub verfügbaren Plugins abgedeckt, was den Entwicklern eine Zeitersparnis ermöglicht, da lediglich eine richtige Integration erforderlich ist. Das Event-Handling trägt ebenfalls zur Beschleunigung der Entwicklung bei. Die Entwickler von Leaflet betonen auf ihrer Webseite die Einfachheit der Bibliothek, wodurch eine exponentielle Lernkurve resultieren kann. Da es sich um eine Open-Source-Bibliothek handelt, können viel mehr Entwickler zur Verbesserung beitragen, dies kann positive Auswirkungen auf die Entwicklungsgeschwindigkeit haben. Die mobile Unterstützung und die Kompatibilität mit allen gängigen Browsern, bringen Entwicklern ebenfalls eine Zeitersparnis, da bei der Implementierung keine plattformspezifischen Anpassungen vorgenommen werden müssen.

4.2.12 Dokumentation

Die Implementierung eines automatisierten Selenium-Tests zur Zählung aller CSS-IDs, welche Verlinkungen zu Erklärungen von Methoden oder Funktionalitäten in der Leaflet-Dokumentation darstellen, erlaubt eine Einschätzung des Umfangs. Der hier vorgestellte Ansatz wurde entwickelt, da die gesamte Dokumentation auf einer einzigen Seite präsentiert wird, ohne dass es Unterdomains gibt. Die Untersuchung hat ergeben, dass 2795 CSS-IDs existieren.

4.2.13 Community- Aktivität

| Plattform | Aktivität | Details |
|----------------|--------------|---------|
| Stack Overflow | Fragen | 34.000+ |
| Github | Repositories | 22.600+ |
| YouTube | Tutorials | 1700+ |

Tabelle 17: Community-Aktivität (Leaflet)

Quelle: Eigene Darstellung

¹⁵⁴ <https://leafletjs.com/reference.html>

4.2.14 Kosten und Lizenzierung

Leaflet ist vollständig kostenlos, es fallen keine Lizenzgebühren oder Abonnementkosten an. Die Bibliothek wurde unter der BSD 2-Lizenz¹⁵⁵ ¹⁵⁶ veröffentlicht, dies impliziert das Leaflet in kommerziellen und nicht kommerziellen Projekten verwendet werden darf. Die BSD-2 Lizenz liegt in zwei Varianten vor, wobei die eine zwei und die andere drei Klauseln umfasst¹⁵⁷. Die einzige Bedingung ist die Angabe eines Urheberrechtshinweises, welcher zwingend enthalten sein muss. Des Weiteren ist die Verwendung der Namen der Urheber sowie der Mitwirkenden zu Werbezwecken untersagt, sofern keine schriftliche Genehmigung vorliegt.

¹⁵⁵ Vgl. <https://github.com/Leaflet/Leaflet/blob/main/LICENSE>

¹⁵⁶ Vgl. <https://opensource.org/license/bsd-2-clause>

¹⁵⁷ Vgl. <https://opensource.org/license/bsd-3-clause>

4.3 Mapbox Graphic Library JavaScript API

4.3.1 Nutzung und Integration

Die Erstellung eines Accounts bei Mapbox ist Voraussetzung für die Nutzung. Im nächsten Schritt ist die Angabe einer Kreditkarte als Zahlungsmittel erforderlich. Anschließend kann der Nutzer seine Präferenzen, Ziele und Vorkenntnisse auswählen, je nach Auswahl wird er durch ein abgestimmtes Tutorial geleitet. Im Anschluss besteht die Möglichkeit, einen API-Key bzw. im Falle von Mapbox, ein Token für einen Dienst zu generieren. Die Einbindung in eine Webseite erfolgt über Skript-Tags, indem die JavaScript-Dateien sowie Stylesheets von Mapbox verlinkt werden. In Frameworks besteht zudem die Möglichkeit einer komponentenbasierten Einbindung. Die Deklaration von Längen- und Breitengraden erfolgt gemäß dem Standard [longitude, latitude], um eine Übereinstimmung mit dem GeoJSON-Format zu gewährleisten. In der Google Maps JS API sowie Leaflet ist die Reihenfolge der Koordinaten jedoch umgekehrt.¹⁵⁸ ¹⁵⁹ Ein weiterer Unterschied ist das Mapbox standardmäßig Vektorkacheln verwendet. Mapbox verwendet dabei OpenStreetMap als Datenquelle für ihr Kachelset namens Mapbox Streets.¹⁶⁰ Dabei kommen Formate wie MVT oder PBF zum Einsatz.¹⁶¹

4.3.2 Ladegeschwindigkeit

Die Mapbox-Bibliothek verfügt über ein „load“-Event, welches als Ereignis für das vollständige Laden der Karte genutzt werden kann. Zusätzlich wurde das „idle“-Event implementiert, welches erst ausgelöst wird, wenn die Karte in einen „Ruhezustand“ übergeht, sprich fertig geladen und gerendert worden ist. Diese Ereignisse liegen zeitlich nah beieinander (vgl. Abbildung 58). In der vorliegenden Untersuchung wurden die Ergebnisse nach Caching differenziert (vgl. Tabelle 18 und Tabelle 19). Zusätzlich wurde die Anzahl der geladenen Ressourcen per JavaScript ausgegeben. Inhalte, die nachgeladen werden, beispielsweise PBF-Dateien, sind davon ausgenommen.

Der zugehörige Quellcode kann unter folgender URL abgerufen werden:

¹⁵⁸ <https://docs.mapbox.com/mapbox-gl-js/guides/install/>

¹⁵⁹ <https://docs.mapbox.com/mapbox-gl-js/api/map/>

¹⁶⁰ <https://docs.mapbox.com/help/tutorials/overpass-turbo/>

¹⁶¹ https://wiki.openstreetmap.org/wiki/Vector_tiles

https://github.com/geo1337/js_web_mapping_comparison/blob/main/MapBox%20GL%20JS/mapbox_index_normal.html

```
Karte nach geladen in: 590.80 Millisekunden      MapBox_index.html:47
Karte bereit: 595.80 Millisekunden             MapBox_index.html:58
>
```

Abbildung 58: Konsolenausgabe JS-Messung der Ladegeschwindigkeit (Mapbox GL JS)

Quelle: Eigene Darstellung

Die Ladegeschwindigkeit hinsichtlich des Netzwerks ist hauptsächlich von geladenen JavaScript-Dateien abhängig sowie vektorbasierten Geodaten im PBF-Format die im nachfolgenden Bild grau hinterlegt sind (vgl. Abbildung 59).¹⁶²

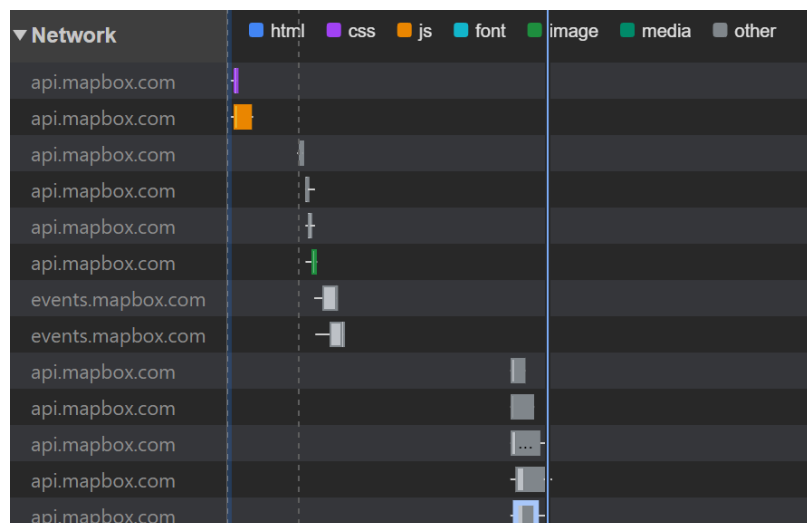


Abbildung 59: Google Chrome Dev Tools Performance Insights (Mapbox)

Quelle: Eigene Darstellung

Die Auswertung der Performance-Analyse von Google Chrome zeigt, dass die Ausführung der Skripte den größten Anteil der Zeit einnimmt. (vgl. Abbildung 60).

¹⁶² https://wiki.openstreetmap.org/wiki/PBF_Format

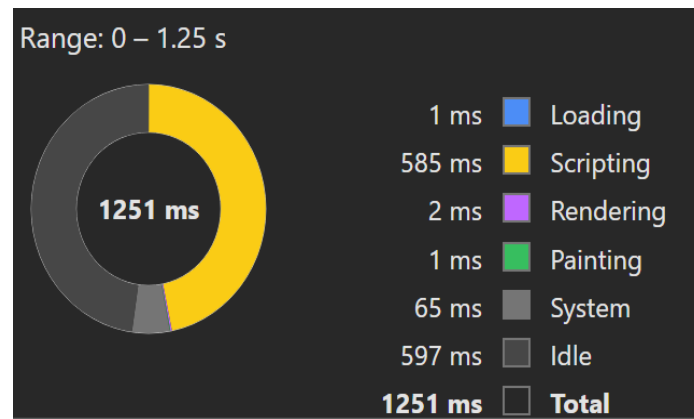


Abbildung 60: Google Chrome Performance-Analyse (Mapbox)

Quelle: Eigene Darstellung

| Browser | Benötigte Zeit in (ms) | Anzahl der Ressourcen |
|-----------------|------------------------|-----------------------|
| Google Chrome | 827.60 | 2 |
| Microsoft Edge | 826.20 | 2 |
| Mozilla Firefox | 1369 | 1 |
| Opera | 947.30 | 2 |

Tabelle 18: Messungen per JavaScript-Code ohne Cache (Mapbox GL JS)

Quelle: Eigene Darstellung

| Browser | Benötigte Zeit in (ms) | Anzahl der Ressourcen |
|-----------------|------------------------|-----------------------|
| Google Chrome | 587.50 | 2 |
| Microsoft Edge | 789 | 2 |
| Mozilla Firefox | 1355 | 1 |
| Opera | 547.80 | 2 |

Tabelle 19: Messungen per JavaScript-Code mit Cache (Mapbox GL JS)

Quelle: Eigene Darstellung

| Browser | Ladevorgang (ms) | Fertigstellen (s) |
|-----------------|------------------|-------------------|
| Google Chrome | 193 | 1.23 |
| Microsoft Edge | 225 | 1.08 |
| Mozilla Firefox | 132 | 1,56 |
| Opera | 218 | 1,67 |

Tabelle 20: Messungen aus den Entwicklerkonsolen ohne Cache (Mapbox GL JS)

Quelle: Eigene Darstellung

| Browser | Ladevorgang (ms) | Fertigstellen (ms) |
|-----------------|------------------|--------------------|
| Google Chrome | 114 | 780 |
| Microsoft Edge | 137 | 956 |
| Mozilla Firefox | 122 | 1580 |
| Opera | 109 | 884 |

Tabelle 21: Messungen aus den Entwicklerkonsolen mit Cache (Mapbox GL JS)

Quelle: Eigene Darstellung

Der bereits in vorherigen Kapiteln detailliert beschriebene Selenium-Test konnte auch mit minimalen Anpassungen für diese Bibliothek angewandt werden (vgl. Abbildung 61).

Die Ergebnisse können unter der folgenden URL in tabellarischer Form eingesehen werden:

https://github.com/geo1337/js_web_mapping_comparison/blob/main/MapBox%20GL%20JS/results_no_cache_mapbox.csv



Abbildung 61: Messungen der Selenium Tests in Chart.js visualisiert (Mapbox GL JS)

Quelle: Eigene Darstellung

In sämtlichen durchgeführten Tests und Untersuchungen konnte festgestellt werden, dass die Ladegeschwindigkeiten bei Mapbox deutlich höher ist. Dies ist auf die Integration von Animationen im Skript sowie das Rendering der Vektorkacheln zurückzuführen (vgl. Abbildung 63).

| Start Time | Self Time | Total Time ▼ | Activity |
|------------|-----------|--------------|-------------------------|
| 313.4 ms | 0.1 ms | 209.3 ms | ▶ Animation Frame Fired |
| 87.6 ms | 0.1 ms | 181.9 ms | ▶ Parse HTML |

Abbildung 62: Performance Analyse Google Chrome (Mapbox GL JS)

Quelle: Eigene Darstellung

Wie bereits für vorherige Web Mapping Lösungen wurden auch hier die Serverantwortzeiten per JMeter gemessen. Hierfür wurde die URL Aus der Dokumentation verwendet.¹⁶³ Der Fakt, dass der Tileservers von Mapbox längere Antwortzeiten aufweist ist auf die Serverseitige Vorverarbeitung und Bereitstellung von Vektorkacheln zurückzuführen (vgl. Abbildung 63). Es gibt keinen spezifischen Contenttype für den HTTP-Header explizit auf Vektorkacheln bezogen da diese in einem binären Format übertragen werden (siehe Anhang C).

¹⁶³ <https://docs.mapbox.com/data/tilesets/guides/vector-tiles-introduction/>

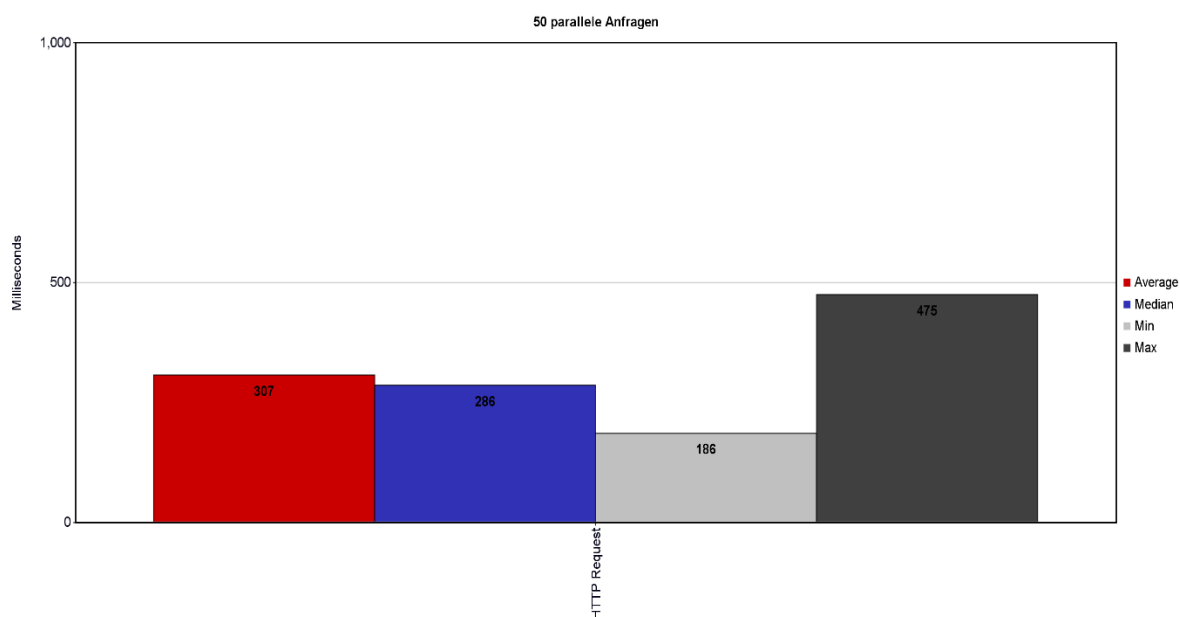


Abbildung 63: Antwortzeiten zu 50 parallele Anfragen an den Mapbox Server

Quelle: Eigene Darstellung

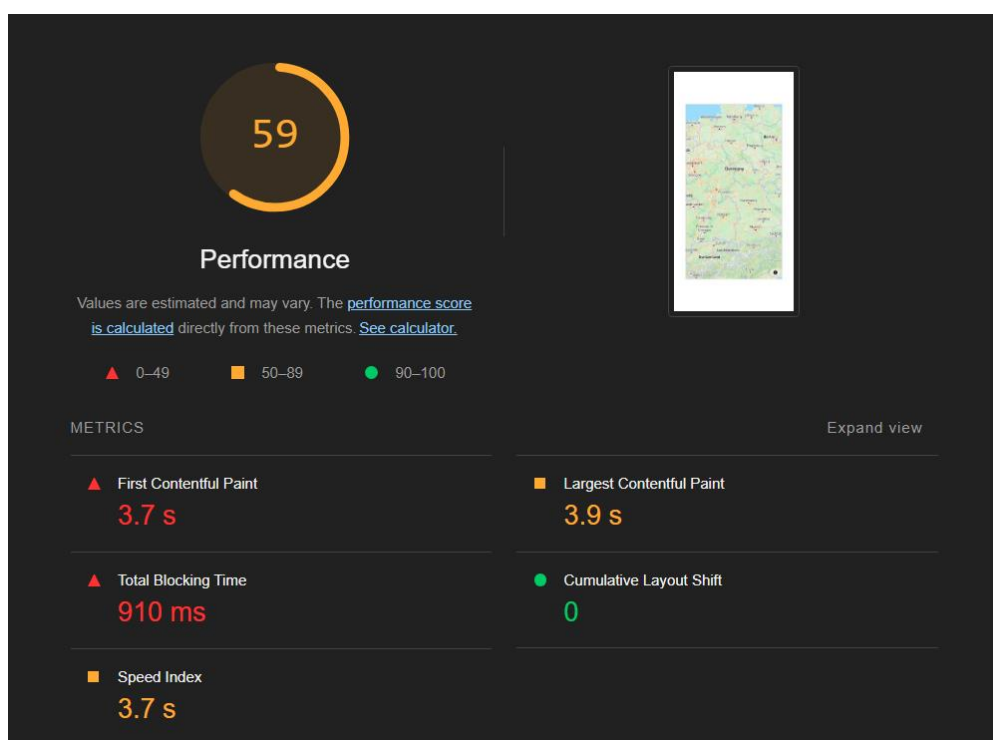


Abbildung 64: Google Lighthouse Test (Mapbox GL JS)

Quelle: Eigene Darstellung

4.3.3 Reaktionsgeschwindigkeit

Es lässt sich feststellen, dass auch MapBox einen längeren Zeitraum für einen Zoomvorgang benötigt, wobei die Dauer mit abnehmender Größe des zu rendernden Abschnitts sinkt (vgl. Abbildung 65). Zur Realisierung dieser Messung wurde ein JavaScript-Code implementiert, der sich die Events „zoomstart“ und „zoomend“ zunutze macht. Der zugehörige Quellcode kann unter folgender URL abgerufen werden:

https://github.com/geo1337/js_web_mapping_comparison/blob/main/MapBox%20GL%20JS/index_mapbox_reactionspeed.html

| | | |
|----------------|-----------------------|--|
| Zoomvorgang 1 | Zoomdauer: 1253.40 ms | index_mapbox_reactionspeed.html:61 |
| Zoomvorgang 2 | Zoomdauer: 1226.40 ms | index_mapbox_reactionspeed.html:61 |
| Zoomvorgang 3 | Zoomdauer: 912.80 ms | index_mapbox_reactionspeed.html:61 |
| Zoomvorgang 4 | Zoomdauer: 914.80 ms | index_mapbox_reactionspeed.html:61 |
| Zoomvorgang 5 | Zoomdauer: 912.20 ms | index_mapbox_reactionspeed.html:61 |
| Zoomvorgang 6 | Zoomdauer: 913.90 ms | index_mapbox_reactionspeed.html:61 |
| Zoomvorgang 7 | Zoomdauer: 913.50 ms | index_mapbox_reactionspeed.html:61 |
| Zoomvorgang 8 | Zoomdauer: 914.90 ms | index_mapbox_reactionspeed.html:61 |
| Zoomvorgang 9 | Zoomdauer: 910.70 ms | index_mapbox_reactionspeed.html:61 |
| Zoomvorgang 10 | Zoomdauer: 912.60 ms | index_mapbox_reactionspeed.html:61 |
| Zoomvorgang 11 | Zoomdauer: 914.20 ms | index_mapbox_reactionspeed.html:61 |
| Zoomvorgang 12 | Zoomdauer: 608.50 ms | index_mapbox_reactionspeed.html:61 |
| Zoomvorgang 13 | Zoomdauer: 606.90 ms | index_mapbox_reactionspeed.html:61 |
| Zoomvorgang 14 | Zoomdauer: 608.40 ms | index_mapbox_reactionspeed.html:61 |
| Zoomvorgang 15 | Zoomdauer: 610.40 ms | index_mapbox_reactionspeed.html:61 |

Abbildung 65: Messung der Reaktionsgeschwindigkeit per JS-Code (Mapbox GL JS)

Quelle: Eigene Darstellung

4.3.4 Ressourcenverbrauch

Die Google Chrome Performance Analyse verdeutlicht, dass Mapbox einen signifikant höheren Arbeitsspeicherbedarf aufweist. Dies ist auf das clientseitige Rendering der Vektor-Kacheln sowie Animationen seitens von Mapbox zurückzuführen. Attribute und Geometrien der Vektor-Kacheln müssen im RAM zwischengespeichert werden, was einen deutlich höheren Arbeitsspeicherverbrauch zur Folge hat als bei Raster-Kacheln (vgl. Abbildung 65 und Abbildung 68).

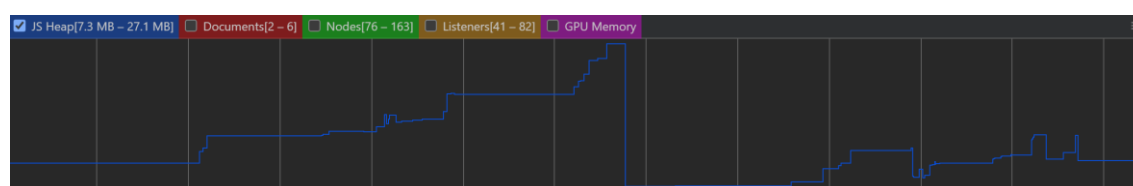


Abbildung 66: Arbeitsspeicher-Nutzung (Mapbox GL JS)

Quelle: Eigene Darstellung

Die Untersuchung belegt, dass Mapbox die Grafikkarte signifikant länger auslastet, was auf die Nutzung von WebGL zum Rendern zurückzuführen ist (vgl. Abbildung 67).

| Self Time | Total Time | Activity |
|------------------|------------------|--|
| 524.3 ms 100.0 % | 524.3 ms 100.0 % | ■ GPU |
| | | |

Abbildung 67: Grafikkarten-Nutzung (MapBox GL JS)

Quelle: Eigene Darstellung

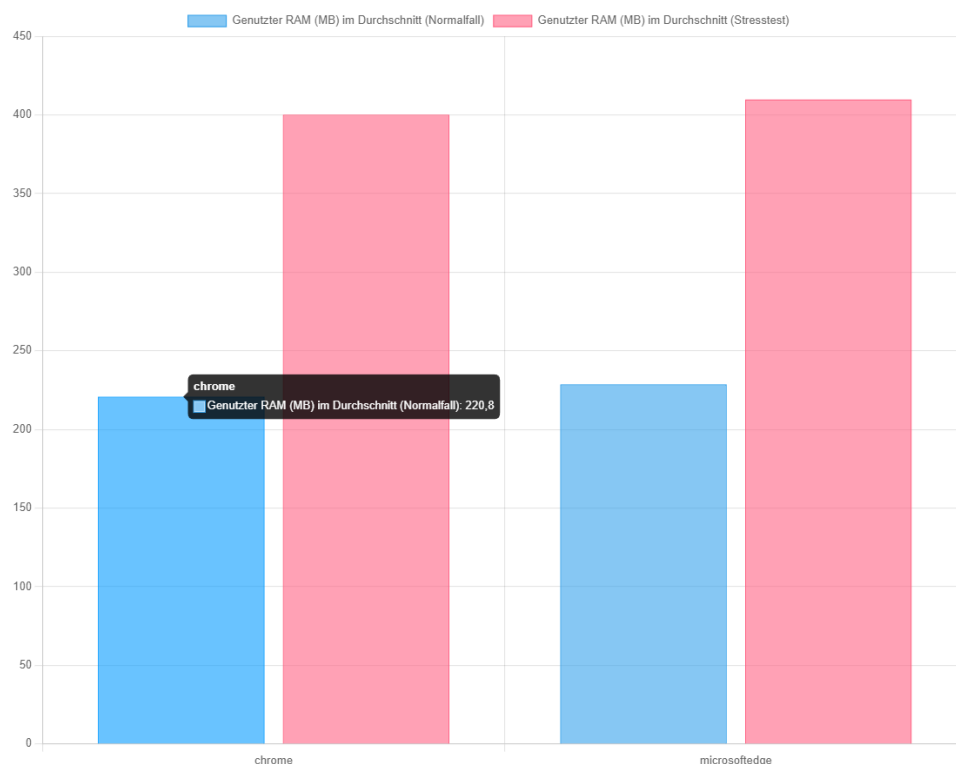


Abbildung 68: RAM-Nutzung während des Selenium-Tests (Mapbox GL JS)

Quelle: Eigene Darstellung

4.3.5 Funktionalitäten

Mapbox unterstützt sowohl Vektorkarten als auch Rasterkarten. Das Kartendesign kann im Mapbox Studio beliebig angepasst werden, wobei sowohl das Kachelset als auch das Datenset modifiziert werden können. Der cloudbasierte Dienst Mapbox Studio ermöglicht die Nutzung einer Vielzahl von Vorlagen für Kachelsets, die nach Belieben bearbeitet werden können. Darüber hinaus können eigene Dateien hochgeladen werden, darunter Formate wie MBTiles, KML, GPX, GeoJSON, Shapefile

und CSV. Anschließend werden diese in Vektorkacheln konvertiert. Für die Erstellung von Rasterkacheln muss eine GeoTIFF Datei hochgeladen werden. Unter anderem besteht die Möglichkeit, benutzerdefinierte Kartenstile zu erstellen. Im Anschluss können die benutzerdefinierten Kachelsets, Datenquellen und Kartenstile in der API Mapbox GL JS visualisiert werden. Geographische Daten können nicht nur visualisiert, sondern auch animiert werden, wie beispielsweise zeitliche Veränderungen oder Bewegungsmuster. Features innerhalb der Vektorkachel können abgefragt und gefiltert werden, wodurch bestimmte Elemente wie beispielsweise POIs hervorgehoben werden können. Zusätzlich können benutzerdefinierte Daten in Echtzeit angezeigt werden, ohne dass die Seite neu geladen werden muss, wie beispielsweise Sensordaten. Da Mapbox eine grafikbasierte Bibliothek ist und standardmäßig WebGL und Vektorkacheln verwendet, können 3D-Visualisierungen und Animationen erstellt werden. Das Sichtfeld der Karte wird unter dem Objekt „Kamera“ definiert und kann Parameter wie Neigung und Ausrichtung enthalten, so dass die Karte als Globus dargestellt werden kann. Darüber hinaus kann eine Vielzahl von interaktiven Elementen hinzugefügt werden, wie z.B. Marker, Pop-ups, Steuerelemente für die Interaktion mit der Karte.^{164 165}

4.3.6 Event-Handling

Die Ereignisse können sowohl der Karte als auch den interaktiven Elementen und den Schichten hinzugefügt werden. Dazu werden Schlüsselwörter wie „on“, „off“, „once“ verwendet, welche die Bindung der Eventhandler an bestimmte Ereignisse definieren. Zudem werden auch mobile Ereignisse wie beispielsweise „Touchstart“ oder „Touchend“ unterstützt.¹⁶⁶

4.3.7 Unterstützung von Geodatenformaten

Nativ wird nur das GeoJSON-Format unterstützt, andere Formate müssen in Mapbox Studio zur Konvertierung hochgeladen werden.¹⁶⁷

¹⁶⁴ <https://studio.mapbox.com/>

¹⁶⁵ <https://docs.mapbox.com/mapbox-gl-js/example/>

¹⁶⁶ <https://docs.mapbox.com/mapbox-gl-js/api/events/>

¹⁶⁷ <https://docs.mapbox.com/help/troubleshooting/uploads/>

4.3.8 Unterstützung von Geodiensten

Der WMS- Standard wird unterstützt, WFS hingegen nicht. ¹⁶⁸

4.3.9 Erweiterbarkeit

Mapbox GL JS kann um weitere Dienste von Mapbox erweitert werden, darunter Mapbox Studio, Mapbox Geocoding API, Mapbox Directions API, Mapbox Static Images API, Mapbox Tilequery API. Zusätzlich existieren 32 Plugins für Mapbox GL JS, die alle auf der Webseite von Mapbox aufgeführt sind. Plugins, die für Map Libre dem Community-Fork von Mapbox erstellt werden könnten, ebenfalls genutzt werden da es um denselben Kern Quellcode der Bibliotheken handelt. Es sollte allerdings sichergestellt werden, ob die Maplibre Plugins mit der aktuellen Version von Mapbox GL JS kompatibel sind. ¹⁶⁹

¹⁶⁸ <https://docs.mapbox.com/mapbox-gl-js/example/wms/>

¹⁶⁹ <https://maplibre.org/news/2023-01-03-aws-maplibre-plugins/>

4.3.10 Interoperabilität

Die API erweist sich als interoperabel mit den ausgewählten JavaScript-Frameworks (vgl. Tabelle 22). Mapbox bietet darüber hinaus weiterführende Tutorials zu diesen Frameworks an. Das verlinkte Repository auf der Mapbox Webseite für React unterstützt die aktuelle React Version 18.3 nicht sondern 16.11, was zu einem Fehler führte.

| Framework | Interoperabel |
|-------------------------|---------------|
| React ¹⁷⁰ | Ja |
| Vue.js ¹⁷¹ | Ja |
| Angular ¹⁷² | Ja |
| Ember.js ¹⁷³ | Ja |
| Svelte ¹⁷⁴ | Ja |

Tabelle 22: Interoperabilität von Mapbox mit JS-Frameworks

Quelle: Eigene Darstellung

4.3.11 Entwicklungsgeschwindigkeit

Mapbox GL JS beinhaltet eine Sammlung vordefinierter Komponenten, darunter Marker, Popups und Steuerelemente. Die Möglichkeit der Wiederverwendung dieser Komponenten sowie deren Anpassbarkeit an spezifische Bedürfnisse erlaubt eine Zeitersparnis. Eine Vielzahl von Code-Beispielen aus der Dokumentation kann mit übernommen werden.

4.3.12 Dokumentation

Die offizielle Dokumentation zu Mapbox GL JS ist lediglich in der englischsprachigen Version verfügbar. Die Dokumentation umfasst insgesamt 206 Unterdomains. Diese decken eine Vielzahl von Themen ab, von grundlegenden Anleitungen, Code-Beispielen bis hin zu fortgeschrittenen Funktionen und API-Referenzen.

¹⁷⁰ <https://github.com/alex3165/react-mapbox-gl>

¹⁷¹ <https://github.com/soal/vue-mapbox>

¹⁷² <https://github.com/Wykks/ngx-mapbox-gl>

¹⁷³ <https://github.com/kturney/ember-mapbox-gl>

¹⁷⁴ <https://docs.mapbox.com/help/tutorials/use-mapbox-gl-js-with-svelte/>

4.3.13 Community-Aktivität

Die Vorgehensweise zur Erhebung dieser Kennzahlen wurde bereits in Kapitel 4.1.13 dargelegt.

| Plattform | Aktivität | Details |
|----------------|--------------|---------|
| Stack Overflow | Fragen | 19.500+ |
| Github | Repositories | 6200+ |
| YouTube | Tutorials | 1000+ |
| Discord | Mitglieder | 12.300+ |

Tabelle 23: Community-Aktivität (Mapbox)

Quelle: Eigene Darstellung

4.3.14 Kosten und Lizenzierung

Mapbox verwendet ein nutzungsbasiertes Preismodell, bei dem die Kosten durch die Anzahl der Anfragen und der Art der Aufrufe zustande kommen. Es wird seitens Mapbox ein kostenloses Kontingent zur Verfügung gestellt. Die Nutzung wird dabei dem Account bezogenen Token zugeordnet. Das kostenlose Kontingent umfasst 250 \$ oder 50.000 monatliche Aufrufe. 1000 weitere Aufrufe kosten 5\$. Um das kostenlose Kontingent in Anspruch nehmen zu können, muss ein Zahlungsmittel hinterlegt werden, es werden nur Kreditkarten akzeptiert. Im Dashboard wird die Anzahl der Aufrufe angezeigt und kann somit überwacht werden, ein Aufruf resultiert aus einer Initialisierung der Karte. Mit der Registrierung sowie Nutzung geht man eine rechtliche Vereinbarung mit Mapbox ein. Die Referenz hier zu findet sich in den Allgemeinen Nutzungsbedingungen.¹⁷⁵

¹⁷⁵ <https://www.mapbox.com/legal/tos>

5 Ergebnisse

5.1 Diskussion der Ergebnisse

Die Integration der Web-Mapping-Lösungen erfolgt in identischer Weise. Für die proprietären Lösungen wird jedoch ein Account sowie ein API-Schlüssel benötigt. Im Rahmen der durchgeführten Leistungstests konnte festgestellt werden, dass Leaflet aufgrund seiner Leichtgewichtigkeit deutlich schnellere Ladezeiten aufweist. Des Weiteren zeichnet sich Google Maps JS API durch kurze Ladezeiten bei Rasterkarten aus. Im Gegensatz dazu weist Mapbox GL JS deutlich langsamere Ladezeiten auf, was auf das Rendern von Vektorkacheln sowie die Nutzung von WebGL zurückzuführen ist. Die Performance von Web Mapping Lösungen wird durch eine Vielzahl von Faktoren beeinflusst, darunter die lokal verwendete Hardware, die Komplexität der Karte, die verwendete Kachelart, die Serverantwortzeiten, das Caching, die Netzwerkverbindung sowie minimal der verwendete Browser. Die Vorteile von Open-Source-Projekten liegen in erster Linie in der kostenlosen Verfügbarkeit sowie in den einfachen Lizenzmodellen, welche keine Nutzungsbeschränkungen beinhalten. Ein weiterer Aspekt ist die Transparenz, da der Quellcode öffentlich einsehbar ist. Bei der Verwendung von Open-Source-Lösungen besteht jedoch keine Gewährleistung der Zuverlässigkeit, insbesondere was die Aktualisierung und Wartung von Plugins betrifft. In Bezug auf den Ressourcenverbrauch zeigt sich, dass Google Maps im moderaten Bereich liegt, Leaflet hingegen in einem sehr geringen Bereich und Mapbox in einem signifikant höheren Bereich. Google Maps verfügt über ein breites Funktionsspektrum, wobei insbesondere die Street-View-Funktion hervorzuheben ist. Leaflet kann sich aufgrund der Verwendung von Plugins gegen proprietäre Lösungen behaupten. Die Mapbox GL JS-Funktionalitäten eignen sich insbesondere für detaillierte Darstellungen, Animationen und 3D-Karten. Mapbox kann sich durch die hohe Anpassbarkeit der Karten hervorheben. Ein Alleinstellungsmerkmal von Leaflet ist die Unterstützung der Standards WMS und WFS. Google Maps JS API ist zwar erweiterbar aber auf das eigene Google- Ökosystem beschränkt. Leaflet lässt sich sehr gut erweitern, da über 100 Plugins existieren. Mapbox ist gut erweiterbar, da es 32 eigene Plugins besitzt. Die Entwicklungsgeschwindigkeit lässt sich schwer bewerten, da diese von diversen Faktoren abhängig ist wie die Vorkenntnisse des Entwicklers, Komplexität der Karte. Leaflets Entwicklungsgeschwindigkeit könnte bei komplexen Karten stagnieren. Die Entwicklungsgeschwindigkeit ist ebenfalls von der

Dokumentation und Community-Aktivität abhängig. Hierbei kann Google die umfangreichste Dokumentation in mehreren Sprachen vorweisen. Google hat auch die größere aktive Community in Relation zu Leaflet und Mapbox, was auf die Monopolstellung von Google selbst zurückzuführen ist. Alle drei Web Mapping sind mit modernen JavaScript-Frameworks interoperabel.

| | GeoJSON | KML |
|--------------------|------------------|--|
| Google Maps JS API | Wird unterstützt | Wird unterstützt |
| Leaflet | Wird unterstützt | Nur über Plugins unterstützt |
| Mapbox GL JS | Wird unterstützt | Nur durch Konvertierung in der Cloud (Mapbox Studio) |

Tabelle 24: Ergebnistabelle hinsichtlich der Geodatenformate- Unterstützung

Quelle: Eigene Darstellung

| | WMS | WFS |
|--------------------|------------------------|------------------------|
| Google Maps JS API | Wird nicht unterstützt | Wird nicht unterstützt |
| Leaflet | Wird unterstützt | Wird unterstützt |
| Mapbox GL JS | Wird unterstützt | Wird nicht unterstützt |

Tabelle 25: Ergebnistabelle zur Unterstützung von Geodiensten

Quelle: Eigene Darstellung

| | Kosten | Lizenzierung |
|--------------------|-------------------------------|---|
| Google Maps JS API | Erst nach 28.500 API-Aufrufen | rechtliche Vereinbarung zwischen Google und den Nutzern (Nutzungsbedingungen) |
| Leaflet | kostenlos | BSD-2 Lizenz |
| Mapbox GL JS | Erst nach 50.000 API-Aufrufen | rechtliche Vereinbarung zwischen Mapbox und den Nutzern (Nutzungsbedingungen) |

Tabelle 26: Ergebnistabelle zu Kosten und Lizenzierung

Quelle: Eigene Darstellung

6 Zusammenfassung und Ausblick

6.1 Zusammenfassung

Aufgrund der bereits vorhandenen Erfahrung in der Webentwicklung sowie der Auseinandersetzung mit den Grundlagen in Kapitel zwei konnten umfangreiche Performance-Tests durchgeführt und Funktionalitätsbeispiele implementiert werden. Das Ziel dieser Bachelorarbeit war die Analyse und der Vergleich führender Web Mapping Lösungen. Diese wurden in Kapitel drei anhand von Statistiken und Marktforschungen identifiziert. In Kapitel vier folgte eine detaillierte Analyse anhand diverser Metriken, welche davor in Kapitel drei erörtert wurden. Die Ergebnisse der eigenen Tests konnten anhand von Analysen der Entwicklerkonsolen untermauert werden.

6.2 Kritische Bewertung

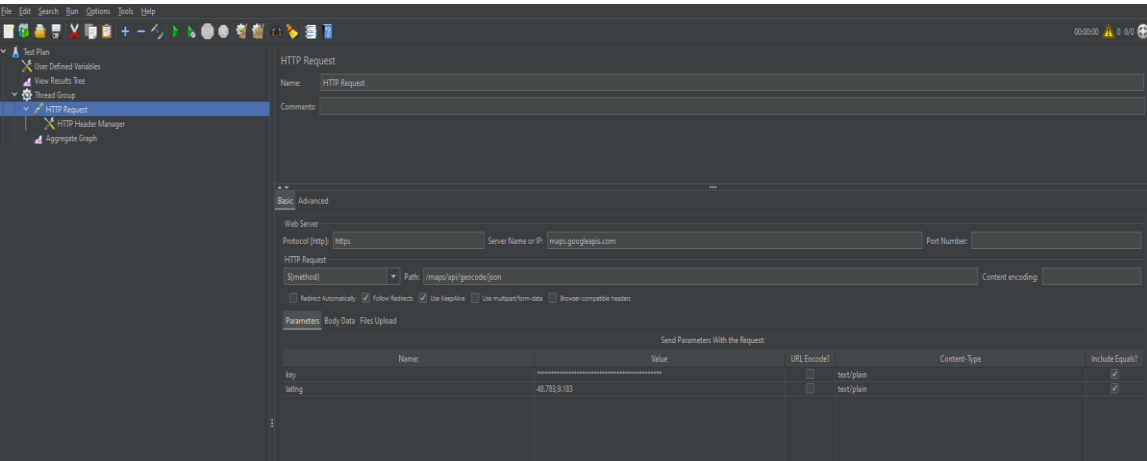
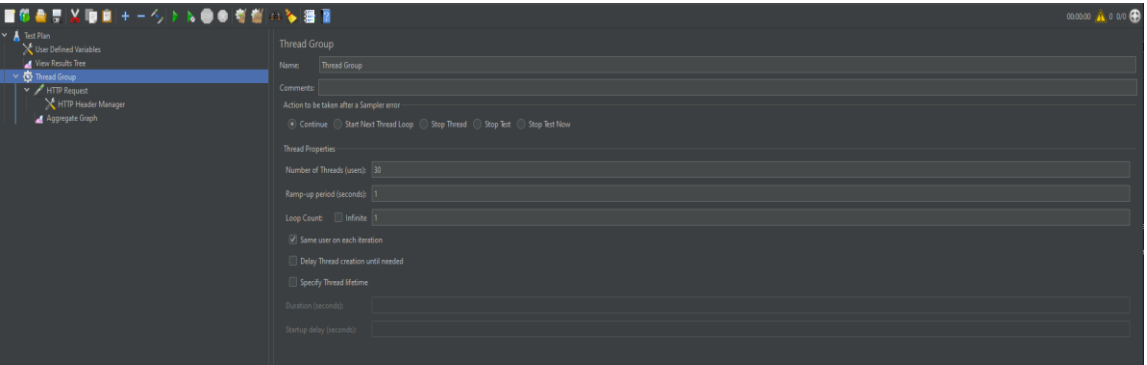
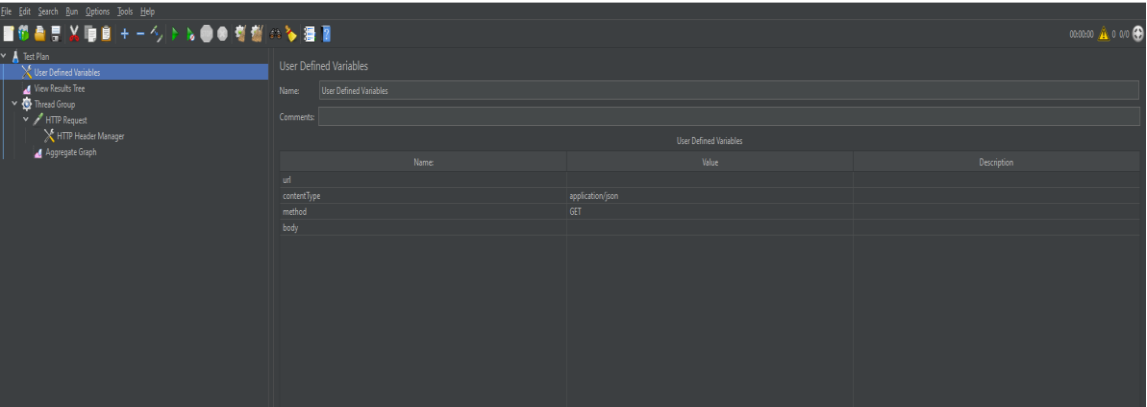
Die Literaturrecherche zum Thema Web Mapping gestaltete sich schwierig, da wenig Bücher zu dieser Thematik existieren. Es wurden mehr Paper, Webseiten und Foren als Referenz herangezogen. Die Messungen hinsichtlich der CPU- sowie GPU-Nutzung sollten kontinuierlich durchgeführt werden und nicht nur aus der Differenz zwischen zwei Zeitpunkten resultieren. Des Weiteren könnten die Messungen unter anderen Netzwerkbedingungen durchgeführt werden. Aufgrund der kurzen Bearbeitungsdauer konnten lediglich drei Web Mapping Lösungen analysiert und verglichen werden. Die Komplexität der Tests könnte noch (nach McCabe) gesenkt werden.¹⁷⁶ In der Berechnung des durchschnittlichen Ressourcenverbrauchs ist ein Fehler enthalten, der sich auf die Metriken der Grafikkarte bezieht.

¹⁷⁶ <https://www.dev-insider.de/was-ist-zyklomatische-komplexitaet-a-7fa40c670052685ff1c56d8169a79481/>

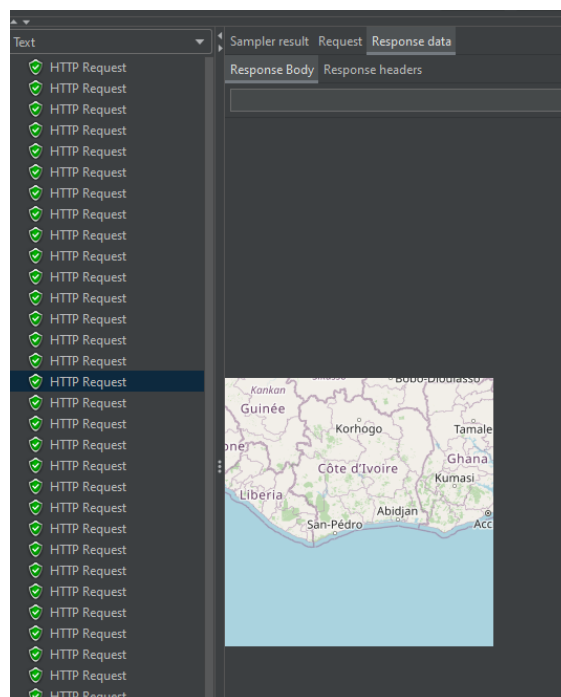
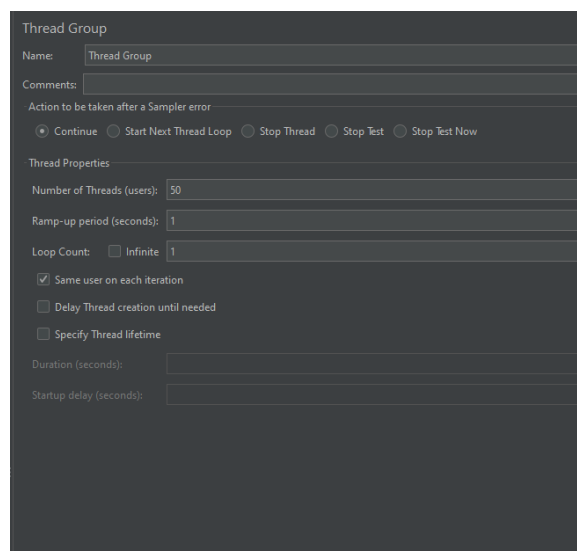
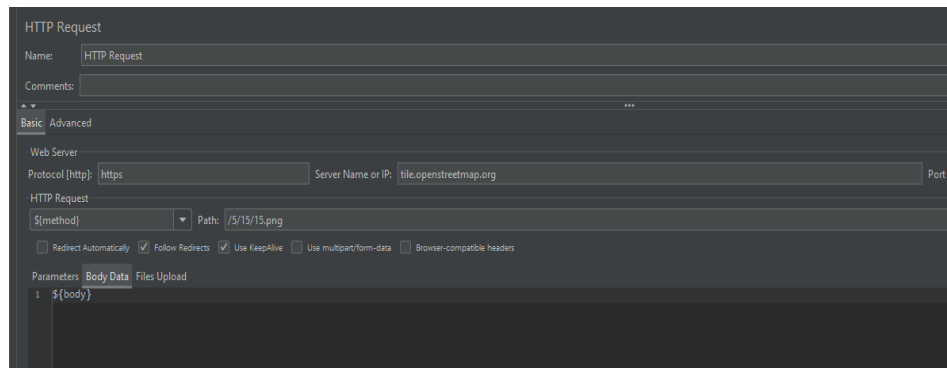
6.3 Ausblick

Die implementierten Beispiele der Funktionalitäten können auf GitHub heruntergeladen und nach Bedarf erweitert werden. Des Weiteren besteht die Möglichkeit, die Selenium-Tests zu erweitern und weitere Web Mapping Lösungen zu untersuchen. Diese könnten in Relation zu den erzielten Ergebnissen der Arbeit gesetzt werden. Die Web Mapping Lösungen könnten für logistische Planungen eingesetzt werden, beispielsweise durch die Kombination der Google Maps API mit der Distance Matrix API oder Leaflet und dem Routing Machine Plugin.

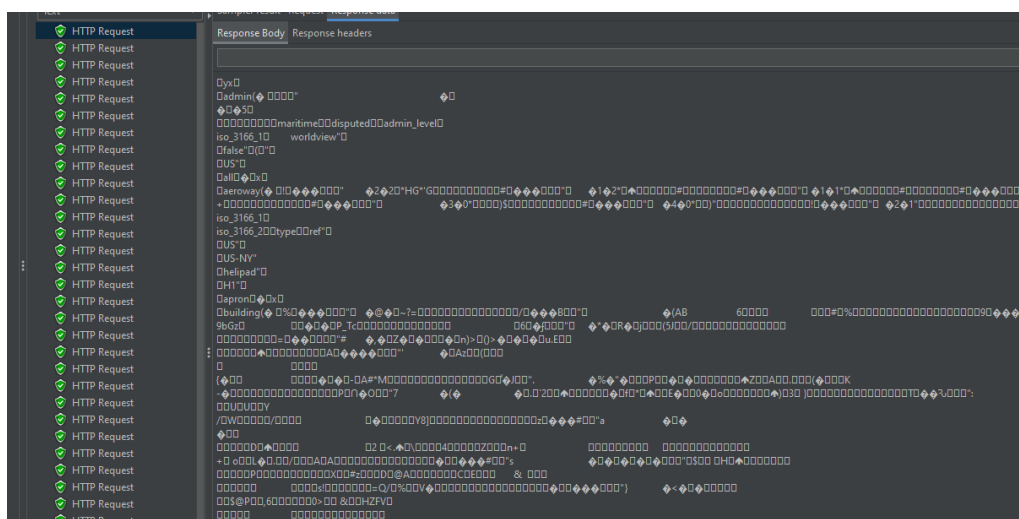
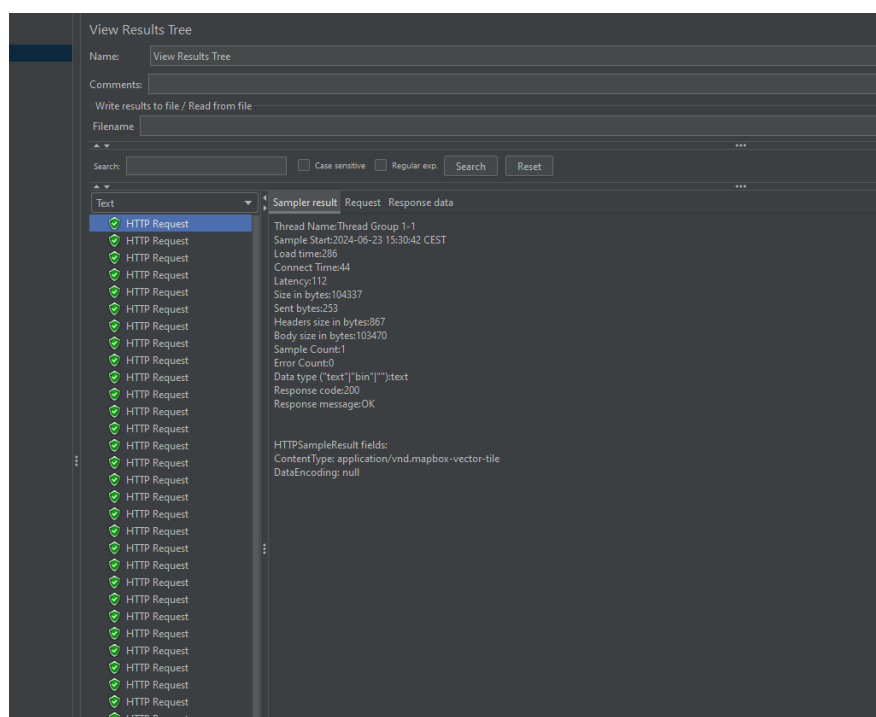
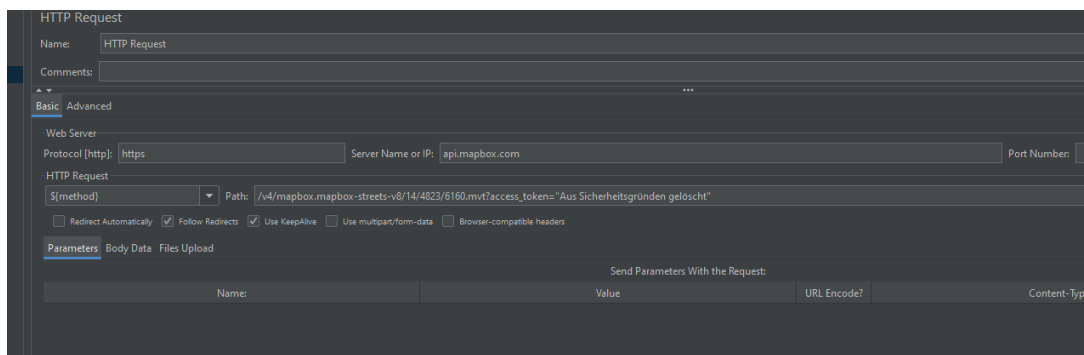
Anhang A. JMeter Testplan (Google Maps JS API)



Anhang B. JMeter Testplan (OpenStreetMap)



Anhang C. JMeter Testplan (Mapbox GL JS)



Literaturverzeichnis

Add a WMS source | Mapbox GL JS. (o. D.). Mapbox.

<https://docs.mapbox.com/mapbox-gl-js/example/wms/> [Letzter Zugriff: Juni 2024]

Add OpenStreetMap data to your Mapbox project | Help. (o. D.). Mapbox.

<https://docs.mapbox.com/help/tutorials/overpass-turbo/> [Letzter Zugriff: Juni 2024]

Alex. (o. D.). GitHub - alex3165/react-mapbox-gl: A React binding of mapbox-gl-js.

GitHub. <https://github.com/alex3165/react-mapbox-gl> [Letzter Zugriff: Juni 2024]

Alhosaini, Hadeel & Alharbi, Sultan & Wang, Xianzhi & Xu, Guandong. (2023). API Recommendation For Mashup Creation: A Comprehensive Survey. The Computer Journal. 10.1093/comjnl/bxad112.

Angular. (o. D.). GitHub - angular/components: Component infrastructure and Material

Design components for Angular. GitHub. <https://github.com/angular/components>

[Letzter Zugriff: Juni 2024]

Apache JMeter - Apache JMeter™. (o. D.). <https://jmeter.apache.org/> [Letzter Zugriff:

Juni 2024]

Basemap.de. (o. D.). <https://basemap.de/> [Letzter Zugriff: Juni 2024]

Basics of the WMS specification | GEOG 585. [https://www.e-](https://www.e-education.psu.edu/geog585/node/699)

[education.psu.edu/geog585/node/699](https://www.e-education.psu.edu/geog585/node/699) [Letzter Zugriff: Juni 2024]

Bert, Veenendaal. (2016). ERAS OF WEB MAPPING DEVELOPMENTS: PAST, PRESENT AND FUTURE. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences.

Blog: WebGL-powered maps features now generally available – Google Maps

Platform. (o. D.). Google Maps Platform.

<https://mapsplatform.google.com/resources/blog/webgl-powered-maps-features-now-generally-available/> [Letzter Zugriff: Juni 2024]

Chart.js. (o. D.). Open Source HTML5 Charts For Your Website.

<https://www.chartjs.org/> [Letzter Zugriff: Juni 2024]

Contributors, M. O. J. T. A. B. (o. D.). Bootstrap. <https://getbootstrap.com/> [Letzter

Zugriff: Juni 2024]

Database Project | Heatflow.X. (o. D.). <http://heatflow.world/project> [Letzter Zugriff: Juni 2024]

Directions API – Übersicht. (o. D.). Google For Developers.

<https://developers.google.com/maps/documentation/directions/overview?hl=de> [Letzter Zugriff: Juni 2024]

Distance Matrix API – Übersicht. (o. D.). Google For Developers.

<https://developers.google.com/maps/documentation/distance-matrix/overview?hl=de> [Letzter Zugriff: Juni 2024]

Documentation - Leaflet - a JavaScript library for interactive maps. (o. D.-a).

<https://leafletjs.com/reference.html#map-event> [Letzter Zugriff: Juni 2024]

- Documentation - Leaflet - a JavaScript library for interactive maps. (o. D.-b).
<https://leafletjs.com/reference.html#marker> [Letzter Zugriff: Juni 2024]
- Documentation - Leaflet - a JavaScript library for interactive maps. (o. D.-c).
<https://leafletjs.com/reference.html#popup> [Letzter Zugriff: Juni 2024]
- Documentation - Leaflet - a JavaScript library for interactive maps. (o. D.-d).
<https://leafletjs.com/reference.html#circlemarker> [Letzter Zugriff: Juni 2024]
- Documentation - Leaflet - a JavaScript library for interactive maps. (o. D.-e).
<https://leafletjs.com/reference.html#polygon> [Letzter Zugriff: Juni 2024]
- Documentation - Leaflet - a JavaScript library for interactive maps. (o. D.-f).
<https://leafletjs.com/reference.html#map-event> [Letzter Zugriff: Juni 2024]
- Documentation - Leaflet - a JavaScript library for interactive maps. (o. D.-g).
<https://leafletjs.com/reference.html#geojson> [Letzter Zugriff: Juni 2024]
- Documentation - Leaflet - a JavaScript library for interactive maps. (o. D.-h).
<https://leafletjs.com/reference.html> [Letzter Zugriff: Juni 2024]
- Documentation - materialize. (o. D.). <https://materializecss.com/> [Letzter Zugriff: Juni 2024]
- Elevation -Dienst. (o. D.). Google For Developers.
<https://developers.google.com/maps/documentation/javascript/elevation?hl=de> [Letzter Zugriff: Juni 2024]
- Ember leaflet. (o. D.). Ember-leaflet. <https://miguelcobain.github.io/ember-leaflet/> [Letzter Zugriff: Juni 2024]
- Ereignisse. (o. D.). Google For Developers.
<https://developers.google.com/maps/documentation/javascript/events?hl=de> [Letzter Zugriff: Juni 2024]
- Events and event types | Mapbox GL JS. (o. D.). Mapbox.
<https://docs.mapbox.com/mapbox-gl-js/api/events/> [Letzter Zugriff: Juni 2024]
- Examples | MapBox GL JS. (o. D.). Mapbox. <https://docs.mapbox.com/mapbox-gl-js/example/> [Letzter Zugriff: Juni 2024]
- Farkas, G. (2015). Comparison of Web Mapping Libraries for Building WebGIS Clients. ResearchGate.
https://www.researchgate.net/publication/296049993_Comparison_of_Web_Mapping_Libraries_for_Building_WebGIS_Clients
- Fetch API - Web APIs | MDN. (2024). MDN Web Docs.
https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API [Letzter Zugriff: Juni 2024]
- Firefox Browser blocking execution of asynchronous JavaScript? (o. D.). Stack Overflow. <https://stackoverflow.com/questions/78503115/firefox-browser-blocking-execution-of-asynchronous-javascript> [Letzter Zugriff: Juni 2024]
- First Contentful paint (FCP). (2024). web.dev. <https://web.dev/articles/fcp?hl=de> [Letzter Zugriff: Juni 2024]
- Free weather API - WeatherAPI.com. (o. D.). <https://www.weatherapi.com/> [Letzter Zugriff: Juni 2024]

- Geocoding API – Übersicht. (o. D.). Google For Developers.
<https://developers.google.com/maps/documentation/geocoding/overview?hl=de>
[Letzter Zugriff: Juni 2024]
- Geofabrik // Raster Tiles. (o. D.). <https://www.geofabrik.de/maps/rastertiles.html>
[Letzter Zugriff: Juni 2024]
- Getting started | MapBox GL JS. (o. D.). Mapbox. <https://docs.mapbox.com/mapbox-gl-js/guides/install/> [Letzter Zugriff: Juni 2024]
- Github 2024
<https://github.com/search?q=google+maps+language%3AJavaScript&type=repositories>
[Letzter Zugriff: Juni 2024]
- Google Cloud Platform. (o. D.). <https://console.cloud.google.com/> [Letzter Zugriff: Juni 2024]
- Google Maps API Usage Statistics. (o. D.).
<https://trends.builtwith.com/mapping/Google-Maps-API> [Letzter Zugriff: Juni 2024]
- Google Maps JavaScript API v3 Reference | Google for Developers. (o. D.). Google For Developers.
<https://developers.google.com/maps/documentation/javascript/reference?hl=de> [Letzter Zugriff: Juni 2024]
- Heinrich Martin. Transport und Lagerlogistik Planung, Struktur, Steuerung und Kosten von Systemen der Intralogistik. Vieweg + Teubner Verlag / Springer Verlag (2011).
- Henning S. Online Karten im Fokus: Praxisorientierte Entwicklung und Umsetzung. Wichmann / VDE Verlag. (2016).
- Home | React Google Maps. (o. D.). <https://visgl.github.io/react-google-maps/> [Letzter Zugriff: Juni 2024]
- Horbiński T, Cybulski P, Medyńska-Gulij B. Web Map Effectiveness in the Responsive Context of the Graphical User Interface. ISPRS International Journal of Geo-Information. 2021; 10(3):134. <https://doi.org/10.3390/ijgi10030134>
- HTTP response status codes - HTTP | MDN. (2023). MDN Web Docs.
<https://developer.mozilla.org/en-US/docs/Web/HTTP/Status> [Letzter Zugriff: Juni 2024]
- Ilya Grigorik (2013). High Performance Browser Networking. O'Reilly Verlag.
- Jeremy L Wagner (2017). Web Performance in Action: Building Faster Web Pages. Manning Verlag.
- Kartentypen. (o. D.). Google For Developers.
<https://developers.google.com/maps/documentation/javascript/maptypes?hl=de>
[Letzter Zugriff: Juni 2024]
- Król, Karol. (2020). EVOLUTION OF ONLINE MAPPING: FROM WEB 1.0 TO WEB 6.0. Geomatics, Landmanagement and Landscape.
- KML-Anleitung. (o. D.). Google For Developers.
https://developers.google.com/kml/documentation/kml_tut?hl=de [Letzter Zugriff: Juni 2024]

Kturney. (o. D.). GitHub - kturney/ember-mapbox-gl: Ember integration for Mapbox GL JS. GitHub. <https://github.com/kturney/ember-mapbox-gl> [Letzter Zugriff: Juni 2024]

Layers. (o. D.). Google For Developers. <https://developers.google.com/maps/documentation/javascript/layers> [Letzter Zugriff: Juni 2024]

Leaflet — an open-source JavaScript library for interactive maps. (o. D.). <https://leafletjs.com/> [Letzter Zugriff: Juni 2024]

Leaflet commands 8.74% market share in Web Mapping. (o. D.). <https://enlyft.com/tech/products/leaflet> [Letzter Zugriff: Juni 2024]

Leaflet JS usage Statistics. (o. D.). <https://trends.builtwith.com/mapping/Leaflet-JS> [Letzter Zugriff: Juni 2024]

Legal information about Terms of Service. (o. D.). <https://www.mapbox.com/legal/tos> [Letzter Zugriff: Juni 2024]

Lighthouse-Leistungsbewertung. (2019). Chrome For Developers. https://developer.chrome.com/docs/lighthouse/performance/performance-scoring?utm_source=lighthouse&utm_medium=devtools&hl=de [Letzter Zugriff: Juni 2024]

Local Storage – Speicher im Browser. (2023). <https://www.mediaevent.de/javascript/local-storage.html> [Letzter Zugriff: Juni 2024]

MapLibre: Mapbox GL open-source fork. (o. D.). MapTiler. <https://www.maptiler.com/news/2021/01/maplibre-mapbox-gl-open-source-fork/> [Letzter Zugriff: Juni 2024]

Markt für digitale Karten Insights. (o. D.). <https://www.mordorintelligence.com/de/industry-reports/digital-map-market#> [Letzter Zugriff: Juni 2024]

MAP | MapBox GL JS. (o. D.). Mapbox. <https://docs.mapbox.com/mapbox-gl-js/api/map/> [Letzter Zugriff: Juni 2024]

Map and Tile Coordinates. (o. D.). Google For Developers. <https://developers.google.com/maps/documentation/javascript/coordinates?hl=de> [Letzter Zugriff: Juni 2024]

Map. (2023). Developing Plugins for MapLibre Interoperability. MapLibre. <https://maplibre.org/news/2023-01-03-aws-maplibre-plugins/> [Letzter Zugriff: Juni 2024]

Mapbox Customers and Market Share. (o. D.). <https://trends.builtwith.com/mapping/Mapbox/Market-Share> [Letzter Zugriff: Juni 2024]

Mapbox. (o. D.). GitHub - mapbox/leaflet-omnivore: universal format parser for Leaflet & Mapbox.js. GitHub. <https://github.com/mapbox/leaflet-omnivore> [Letzter Zugriff: Juni 2024]

Mapping technologies Web Usage Distribution on the Entire Internet. (o. D.). <https://trends.builtwith.com/mapping/traffic/Entire-Internet> [Letzter Zugriff: Juni 2024]

Maps JavaScript API | Google for Developers. (o. D.). Google For Developers. <https://developers.google.com/maps/documentation/javascript/reference/map?hl=de#Map.tilesloaded> [Letzter Zugriff: Juni 2024]

Market Research Future. (2021). Digitale Karte: Marktgröße, Branchenanteil und Trends — 2030 <https://www.marketresearchfuture.com/de/reports/digital-map-market-6600> [Letzter Zugriff: Juni 2024]

Martin Elias. Raster vs Vector Map Tiles: What Is the Difference Between the Two Data Types? 2023. <https://documentation.maptiler.com/hc/en-us/articles/4411234458385-Raster-vs-Vector-Map-Tiles-What-Is-the-Difference-Between-the-Two-Data-Types> [Letzter Zugriff: Juni 2024]

Matt, Forrest (2021). A Brief History of Web Maps. <https://forrest.nyc/a-brief-history-of-web-maps/> [Letzter Zugriff: Juni 2024]

Metainformationssystem GDI-BW. (o. D.). <https://metadaten.geoportal-bw.de/geonetwork/srv/ger/catalog.search#/metadata/a9fe7ea2-8cdd-3505-d219-020aca274536> [Letzter Zugriff: Juni 2024]

Michael Dorman. Introduction to Web Mapping. CRC Press. (2020)

Michael. (2023). Per WMIC Befehl Informationen zur CPU auslesen. Windows FAQ. <https://www.windows-faq.de/2018/12/22/per-wmic-befehl-informationen-zur-cpu-auslesen/> [Letzter Zugriff: Juni 2024]

Mit der Entwickler-Community austauschen. (o. D.). Google For Developers. <https://developers.google.com/maps/developer-community?hl=de> [Letzter Zugriff: Juni 2024]

Nétek, Rostislav & Masopust, Jan & Pavlicek, & Pechanec, Vilém. (2020). Performance Testing on Vector vs. Raster Map Tiles—Comparative Study on Load Metrics. ISPRS International Journal of Geo-Information. 9. 101. 10.3390/ijgi9020101.

Ngyewch. (o. D.). GitHub - ngyewch/svelte-leafletjs: Svelte component for leaflet. GitHub. [Letzter Zugriff: Juni 2024]

Node.js — Run JavaScript everywhere. (o. D.). <https://nodejs.org/en/> [Letzter Zugriff: Juni 2024]

npm: @asymmetrik/ngx-leaflet. (o. D.). Npm. <https://www.npmjs.com/package/@asymmetrik/ngx-leaflet> [Letzter Zugriff: Juni 2024]

npm: @types/svelte-leafletjs. (o. D.). Npm. <https://www.npmjs.com/package/@types/svelte-leafletjs> [Letzter Zugriff: Juni 2024]

NVIDIA System Management Interface. (o. D.). NVIDIA Developer. <https://developer.nvidia.com/system-management-interface> [Letzter Zugriff: Juni 2024]

ODonohue, D. (2023). RASTER TILES – What you need to know - mapscaping.com. <https://mapscaping.com/raster-tiles-what-you-need-to-know/> [Letzter Zugriff: Juni 2024]

Open GeoData. (o. D.). <https://www.lgl-bw.de/Produkte/Open-Data/#Geodatendienste> [Letzter Zugriff: Juni 2024]

OpenJS Foundation - openjsf.org. (o. D.). JQuery. <https://jquery.com/> [Letzter Zugriff: Juni 2024]

- OpenStreetMap Deutschland - Die freie Wiki-Weltkarte. (o. D.).
<https://www.openstreetmap.de/> [Letzter Zugriff: Juni 2024]
- Ott, N., & Mäs, S. (2023). Analysis of free web mapping libraries. Zenodo.
<https://doi.org/10.5281/zenodo.8139086>
- PaulLeCam. (o. D.). GitHub - PaulLeCam/react-leaflet: React components for Leaflet maps. GitHub. <https://github.com/PaulLeCam/react-leaflet> [Letzter Zugriff: Juni 2024]
- Payment options in my country - Cloud Identity Help. (o. D.).
<https://support.google.com/cloudidentity/answer/2380700> [Letzter Zugriff: Juni 2024]
- PBF format - OpenStreetMap Wiki. (o. D.).
https://wiki.openstreetmap.org/wiki/PBF_Format [Letzter Zugriff: Juni 2024]
- Performance: now() method - Web APIs | MDN. (2024). MDN Web Docs.
<https://developer.mozilla.org/en-US/docs/Web/API/Performance/now> [Letzter Zugriff: Juni 2024]
- Phillip Ackermann. Fullstack-Entwicklung. Das Handbuch für Webentwicklung. Rheinwerk Verlag. (2023).
- Plugins - Leaflet - a JavaScript library for interactive maps. (o. D.).
<https://leafletjs.com/plugins.html> [Letzter Zugriff: Juni 2024]
- Point, G. (o. D.). gis.Point - Das Portal für Geoinformation und Geodäsie - Artikelarchiv.
<https://gispoint.de/artikelarchiv/gis/2014/gisbusiness-ausgabe-5-62014.html> [Letzter Zugriff: Juni 2024]
- Points of interest - OpenStreetMap Wiki. (o. D.).
https://wiki.openstreetmap.org/wiki/Points_of_interest [Letzter Zugriff: Juni 2024]
- Polygon – Definition | GIS-Wörterbuch. (o. D.). <https://support.esri.com/de-de/gis-dictionary/polygon> [Letzter Zugriff: Juni 2024]
- Polylinie – Definition | GIS-Wörterbuch. (o. D.). <https://support.esri.com/de-de/gis-dictionary/polyline> [Letzter Zugriff: Juni 2024]
- Platform Pricing & API costs - Google Maps platform. (o. D.). Google Maps Platform.
<https://mapsplatform.google.com/pricing/> [Letzter Zugriff: Juni 2024]
- Prakash, Bhanu. (2020). Retrofitting Mobile First Design, Responsive Design: Driving Factors, Approach, Best Practices and Design Considerations. Current Trends in Computer Sciences & Applications.
- Quick Start Guide - Leaflet - a JavaScript library for interactive maps. (o. D.).
<https://leafletjs.com/examples/quick-start/> [Letzter Zugriff: Juni 2024]
- Raster tiles - OpenStreetMap Wiki. (o. D.).
https://wiki.openstreetmap.org/wiki/Raster_tiles [Letzter Zugriff: Juni 2024]
- Rauch, G. (2023). Was ist zyklomatische Komplexität? Dev-Insider. <https://www.dev-insider.de/was-ist-zyklomatische-komplexitaet-a-7fa40c670052685ff1c56d8169a79481>
[Letzter Zugriff: Juni 2024]
- REACT Leaflet | REACT Leaflet. (o. D.). <https://react-leaflet.js.org/> [Letzter Zugriff: Juni 2024]
- React. (o. D.). <https://react.dev/> [Letzter Zugriff: Juni 2024]

- Redaktion, I. (2022). Was ist Caching? IONOS Digital Guide.
<https://www.ionos.de/digitalguide/hosting/hosting-technik/was-ist-caching/> [Letzter Zugriff: Juni 2024]
- Referenz zu Leistungsfunktionen. (2024). Chrome For Developers.
<https://developer.chrome.com/docs/devtools/performance/reference?hl=de> [Letzter Zugriff: Juni 2024]
- Reid, E. (2020). A look back at 15 years of mapping the world. Google.
<https://blog.google/products/maps/look-back-15-years-mapping-world/> [Letzter Zugriff: Juni 2024]
- Sandydoo. (o. D.). GitHub - sandydoo/ember-google-maps: A friendly Ember addon for working with Google Maps. GitHub. <https://github.com/sandydoo/ember-google-maps> [Letzter Zugriff: Juni 2024]
- SaSS: Syntactically awesome style sheets. (o. D.). <https://sass-lang.com/> [Letzter Zugriff: Juni 2024]
- Selenium. (o. D.). Selenium. <https://www.selenium.dev/> [Letzter Zugriff: Juni 2024]
- Soal. (o. D.). GitHub - soal/vue-mapbox: Vuejs 2 components for interacting with mapbox-gl-js. GitHub. <https://github.com/soal/vue-mapbox> [Letzter Zugriff: Juni 2024]
- Stack Overflow 2024 <https://stackoverflow.com/questions/tagged/google-maps> [Letzter Zugriff: Juni 2024]
- Stack Overflow 2024 <https://stackoverflow.com/search?q=google+maps> [Letzter Zugriff: Juni 2024]
- Stack Overflow. (2023). Stack Overflow Developer Survey 2023.
<https://survey.stackoverflow.co/2023/> [Letzter Zugriff: Juni 2024]
- Street View Service. (o. D.). Google For Developers.
<https://developers.google.com/maps/documentation/javascript/streetview> [Letzter Zugriff: Juni 2024]
- Studio. (o. D.). Mapbox. <https://studio.mapbox.com/> [Letzter Zugriff: Juni 2024]
- Suchoperator „site:“ verwenden | Google Search Central | Dokumentation | Google for Developers. (o. D.). Google For Developers.
<https://developers.google.com/search/docs/monitor-debug/search-operators/all-search-site?hl=de> [Letzter Zugriff: Juni 2024]
- Tailwind CSS - Rapidly build modern websites without ever leaving your HTML. (o. D.). Tailwind CSS. <https://tailwindcss.com/> [Letzter Zugriff: Juni 2024]
- The 2-Clause BSD license. (2023). Open Source Initiative.
<https://opensource.org/license/bsd-2-clause> [Letzter Zugriff: Juni 2024]
- The 3-Clause BSD license. (2023). Open Source Initiative.
<https://opensource.org/license/bsd-3-clause> [Letzter Zugriff: Juni 2024]
- Three.js – JavaScript 3D library. (o. D.). <https://threejs.org/> [Letzter Zugriff: Juni 2024]
- Timhall. (o. D.). GitHub - timhall/svelte-google-maps: Google Maps components for Svelte. GitHub. <https://github.com/timhall/svelte-google-maps> [Letzter Zugriff: Juni 2024]

Top 5 products in the Web Mapping market. (o. D.). <https://enlyft.com/tech/web-mapping> [Letzter Zugriff: Juni 2024]

Upload data to Mapbox | Help. (o. D.). Mapbox.
<https://docs.mapbox.com/help/troubleshooting/uploads/> [Letzter Zugriff: Juni 2024]

Use API keys. (o. D.). Google For Developers.
<https://developers.google.com/maps/documentation/javascript/get-api-key?hl=de>
[Letzter Zugriff: Juni 2024]

Use Mapbox GL JS in a Svelte app | Help. (o. D.). Mapbox.
<https://docs.mapbox.com/help/tutorials/use-mapbox-gl-js-with-svelte/> [Letzter Zugriff: Juni 2024]

Vector tiles - OpenStreetMap Wiki. (o. D.).
https://wiki.openstreetmap.org/wiki/Vector_tiles [Letzter Zugriff: Juni 2024]

Vector tiles introduction | Tilesets | Mapbox Docs. (o. D.). Mapbox.
<https://docs.mapbox.com/data/tilesets/guides/vector-tiles-introduction/> [Letzter Zugriff: Juni 2024]

Vue 3 Google maps. (o. D.). <https://vue-map.netlify.app/> [Letzter Zugriff: Juni 2024]

W3Schools.com. (o. D.). https://www.w3schools.com/nodejs/ref_os.asp [Letzter Zugriff: Juni 2024]

What are vector tiles and why you should care. (o. D.). MapTiler.
<https://www.maptiler.com/news/2019/02/what-are-vector-tiles-and-why-you-should-care>
[Letzter Zugriff: Juni 2024]

Web Vektor - basemap.de. (o. D.). <https://basemap.de/web-vektor/> [Letzter Zugriff: Juni 2024]

WebGL: 2D and 3D graphics for the web - Web APIs | MDN. (2024). MDN Web Docs.
https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API?retiredLocale=de
[Letzter Zugriff: Juni 2024]

WebGL-Overlay-Ansicht. (o. D.). Google For Developers.
<https://developers.google.com/maps/documentation/javascript/webgl/webgl-overlay-view?hl=de> [Letzter Zugriff: Juni 2024]

Wetter und Klima - Deutscher Wetterdienst - Leistungen - Geodienste (WMS/WFS). (o. D.). <https://www.dwd.de/DE/leistungen/geodienste/geodienste.html> [Letzter Zugriff: Juni 2024]

WFS - Introduction — OGC e-Learning 2.0.0 documentation. (o. D.).
<https://opengeospatial.github.io/e-learning/wfs/text/basic-main.html> [Letzter Zugriff: Juni 2024]

WFS - Introduction — OGC e-Learning 2.0.0 documentation. (o. D.).
<https://opengeospatial.github.io/e-learning/wfs/text/basic-main.html> [Letzter Zugriff: Juni 2024]

WFS and editing vector data on the web | GEOG 585: Web Mapping. (o. D.).
<https://www.e-education.psu.edu/geog585/node/779>

What is an API? (o. D.). Postman API Platform. <https://www.postman.com/what-is-an-api/> [Letzter Zugriff: Juni 2024]

WMS - Introduction — OGC e-Learning 2.0.0 documentation. (o. D.).
<https://opengeospatial.github.io/e-learning/wms/text/basic-main.html> [Letzter Zugriff: Juni 2024]

Wykks. (o. D.). GitHub - Wykks/ngx-mapbox-gl: Angular binding of mapbox-gl-js.
GitHub. <https://github.com/Wykks/ngx-mapbox-gl> [Letzter Zugriff: Juni 2024]

XAMPP Installers and Downloads for Apache Friends. (o. D.).
<https://www.apachefriends.org/de/index.html> [Letzter Zugriff: Juni 2024]

Zahlungsoptionen für Google-Dienste - Cloud Identity-Hilfe. (o. D.).
<https://support.google.com/cloudidentity/answer/1230192?hl=DE> [Letzter Zugriff: Juni 2024]

Zunino, A.; Velázquez, G.; Celemín, J.P.; Mateos, C.; Hirsch, M.; Rodriguez, J.M.
Evaluating the Performance of Three Popular Web Mapping Libraries: A Case Study
Using Argentina's LifeQuality Index. ISPRS Int. J. Geo-Inf. 2020, 9, 563.
<https://doi.org/10.3390/ijgi9100563>