Technology
Arts Sciences
TH Köln

Featureentwicklung in modernen Browsern

eine Analyse moderner Browser im Bezug auf die Umsetzungsvorgehen der Web Spezifikationen

Bachelorarbeit

ausgearbeitet von

Raphael Höser

zur Erlangung des akademischen Grades
Bachelor of Science (B.Sc.)

vorgelegt an der

Technischen Hochschule Köln
Campus Gummersbach
Fakultät für Informatik und
Ingenieurwissenschaften

im Studiengang
Medieninformatik

Erster Prüfer/in:	Prof. Christian Noss Technische Hochschule Köln
Zweiter Prüfer/in:	Prof. Dr. Christian Kohls Technische Hochschule Köln

Gummersbach, im Juni 2023

Adressen: Raphael Höser
Zur Dornhecke 12
51674 Wiehl
raphael.hoeser@smail.th-koeln.de

Prof. Christian Noss
Technische Hochschule Köln
Institut für Informatik
Steinmüllerallee 1
51643 Gummersbach
christian.noss@th-koeln.de

Prof. Dr. Christian Kohls
Technische Hochschule Köln
Institut für Informatik
Steinmüllerallee 1
51643 Gummersbach
christian.kohls@th-koeln.de

Inhaltsverzeichnis

Kurzfassung

Webentwicklung ist ein sich laufend weiterentwickelndes Feld mit einer Vielzahl an Stakeholder*innen wie Browserhersteller*innen, Standardisierungsorganisationen, Entwickler*innen und Nutzer*innen. Diese Weiterentwicklung läuft gerade in Browsern oftmals nicht synchron ab und so ist die tatsächlich Unterstützung besonders für neue Features in unterschiedlichen Browsern für Entwickler*innen oft schwierig einzuschätzen. Diese Enschätzung ist jedoch für Entwickler*innen relevant, da sie die Nutzung neuer Features in ihren Projekten abwägen müssen. Diese Arbeit zeigt auf, wie die Standardisierung und Umsetzung neuer Features funktioniert und wie Google Chrome eine führende Rolle einnimmt und Apples Safari sich in den letzten Jahren geändert hat. Dies wird unter anderem durch eine Analyse historischer Kompatibilitätsdaten aus unterschiedlichen Quellen wie CanIUse erreicht. Des weiteren zeigt diese Arbeit, welche Ansätze zur Unterstützung neuer Features in allen Browsern genutzt werden und dass Safari nicht wie teilweise behauptet der neue Internet Explorer ist und gerade seit 2022 deutlich in der Featureunterstützung auf Google Chrome und Mozilla Firefox aufholt.

Abstract

Web development is a constantly moving field with many stakeholders like browser developers, standardisation bodies, developers and users. This movement often does not happen in all browsers at the same pace and henceforth it's often difficult for developers to assess the support of new features in different browsers. This assessment is important to developers, since they have to decide wether they can use a feature in their projects. This work shows how the standardisation and implementation of new features works and how Google Chrome takes a leading role and how Apple's Safari has changed in the last years. This is achieved by analysing historical compatibility data from different sources like CanIUse. Furthermore this work shows which approaches are used to support new features in all browsers and that Safari is not the new Internet Explorer as some people claim and that it's catching up with Google Chrome and Mozilla Firefox in feature support since 2022.

1 Problemstellung

Webentwicklung ist ein sehr schnelllebiges und dynamisches Entwicklungsfeld (einzelne Werkzeuge und Bibliotheken können im Zeitraum von nur einem Jahr im zweistelligen Prozentbereich Nutzer*innen gewinnen und verlieren (jam, 2022)), dass sich gerade in den letzten Jahren durch neue Features wie die sogenannten "Fugu"-APIs (Schnittstellen, die Fähigkeiten nativer Apps ins Web bringen sollen (fug, 2022)) oder die Einführung von Progressive-Web-Apps (PWAs) stark gewandelt hat. Dies zeigt sich auch in der Übersicht der zuletzt abgeschlossenen Proposals im ECMAScript Standard (ecm, 2022) und der Nutzung von WebAssembly (einem Feature, dass erst vor fünf Jahren eingeführt wurde und heute bereits bei ca. zwei Prozent der Seitenaufrufe genutzt wird (was, 2022)) oder Web-Components (die ebenfalls in den letzten fünf Jahren eingeführt wurden und aktuell bei etwa 18 % der Seitenaufrufe im Chrome genutzt werden (wc-, 2022)).

Die Unterstützung der Features in gerade den großen Browser-Engines Blink, WebKit und Gecko ist weder einheitlich noch vollständig (Deveria, 2022) und selbst eingeführte oder getestete Features finden unter Umständen keine weite Verbreitung oder langfristigen Support (cus, 2022) (jpe, 2022).

Für Entwickelnde ist es hier interessant, welcher Browser als erstes neue Features zum Testen freigibt und wie lange es dauert, bis neue Standards in allen Browsern ankommen. Zudem ist die "Stabilität" der neuen Funktionen wichtig, also dass die Features bei der ersten Verbreitung bereits möglichst dem (zukünftigen) Standard entsprechen.

Hierzu existieren zwar zahlreiche verstreute Quellen wie "caniuse" (Deveria, 2022), verschiedene Seiten der Browserhersteller*innen, sowie ein Dashboard des W3C über die Web Platform Tests (wpt, 2022), diese sind jedoch teilweise nicht über mehrere Browser hinweg aktuell (so erwähnt Chrome explizit, dass "nachdem ein Feature in Chrome ausgeliefert ist, wird die Aktualität der [Angaben zur Unterstützung in anseren Browsern] nicht mehr garantiert." (chr, 2022a)) oder geben nur schwer eine Gesamtübersicht.

Das Problem der divergierenden Standardimplementierungen ging zwischenzeitlich so weit, dass sich Sätze wie "Safari ist der neue Internet Explorer" (saf, 2022b) oder dass ein Web in dem Chrome 95 % Marktanteil hätte, eine schreckliche Zukunft wäre (Simmons, 2022), regelmäßig im Internet von Webentwickler*innen und Browserentwickler*innen wieder finden. Als Reaktion darauf ist 2021 von den Unternehmen hinter den drei größten Browsern (Chrome/Edge, Firefox, Safari) das Projekt "Interop 2021" ins Leben gerufen und 2022 verlängert worden, um die Unterschiede zwischen den einzelnen Browsern zu beseitigen (int, 2022).

2 Zielsetzung

Diese Arbeit soll einen Einblick verschaffen, wie die Browserplattform sich weiterentwickelt, welche Browser in welchen Bereichen als Treiber oder Nachläufer eingestuft werden können und ob es Auffälligkeiten in der Entwicklung einzelner Browser oder Standards gibt. Ein gesondertes Augenmerk wird hier auch auf die Vorgehensweise der Browserhersteller*innen gelegt, wie diese sich ändernde Standards umsetzen und zum Beispiel vorab zur Verfügung stellen und nicht standardkonforme Implementationen wieder entfernen.

3 Einleitung

In den folgenden Kapiteln wird eine datenbasierte Analyse und Interpretation der Featureentwicklung in den Browsern Safari, Firefox und Chrome durchgeführt. Hierfür werden die historischen Kompatibilitätsdaten aus mehreren Quellen zusammengeführt und automatisiert ausgewertet. Abseits von diesen Daten werden auch die Standardisierungsprozesse für wichtige Webtechnologien betrachtet, da diese die Grundlage für neue Features in der Webplattform darstellen. Hier wird ebenfalls betrachtet, inwieweit Außenstehende Einfluss auf die Standardisierungsprozesse nehmen können. Zudem werden auch die Prozesse der Umsetzung in den Browsern Chrome, Safari und Firefox verglichen. Hier liegt ein besonderes Augenmerk darauf, wie mit neuen Features vor einem endgültigen Release umgegangen wird.

Nachdem die Prozesse aufgearbeitet wurden, betrachten die Kapitel 7 und 8, welche existierenden Datensätze als Basis für die nachfolgenden Analysen genutzt und kombiniert werden können und wie aus den Rohdaten möglichst automatisiert die Grafiken für die Analysen erstellt werden können.

Diese Graphen werden dann in Kapitel 10 weiterverwendet, um anhand objektiver Betrachtungen erste Vergleiche zwischen den betrachteten Browsern zu ziehen. Hierbei werden vor allem die einzelnen Graphen selbst erläutert, bevor diese in Kapitel 11 in Bezug zueinander gesetzt und interpretiert werden, um neue Schlüsse im Kontext der Zielsetzung zu erarbeiten.

Um den Lesefluss zu erleichtern, wird die Chronologie der Erarbeitung der Analysen und der Automatisierung der Datenaufarbeitung nicht beibehalten, sondern nach Themenbereich in Kapitel sortiert.

In dieser Arbeit wird häufig von "Features" die Rede sein. Gemeint sind hier Funktionen, Funktionsgruppen und Bestandteile der Webplattform, die durch die Standardisierungsprozesse definiert und von den Browsern umgesetzt werden.

Ebenso wird in der Arbeit der Begriff der "Webplattform" verwendet. Als Webplattform wird die Sammlung aller Features bezeichnet, die ein moderner Browser nach den Webspezifikationen umsetzen kann. Hierzu zählen alle Funktionen, die ohne die Nutzung externer Bibliotheken oder Frameworks vorhanden sind oder von Entwickelnden selbst direkt für das vorliegende Problem implementiert wurden. Zudem werden Kompatibilitätsbibliotheken (sogenannte Polyfills) explizit erlaubt, solange sie sich auf offene Standards oder Standardproposals stützen.

4 Entwicklung von Standards

Die Entwicklung im Web greift auf unterschiedliche Technologien zurück. Darunter HTML, CSS, EcmaScript/JavaScript und neuerdings auch WebAssembly. Damit diese Technologien in allen Browsern möglichst gleich funktionieren, werden sie von einem jeweilgen Kommittee zentral spezifiziert. In diesem Abschnitt wird betrachtet wer für die Standardisierung welcher Technologien verantwortlich ist, wie der Prozess hierfür aussieht und wie die Teilhabe von außen bei der Standardisierung funktionieren kann.

4.1 W3C

4.1.1 Zusammensetzung

Das W3C Konsortium besteht aus mehreren Untergruppen. Es pflegt und entwickelt eine Vielzahl an Standards gerade im Bereich der Webstandards. Auf die für diese Arbeit relevanten wird im Folgenden weiter eingegangen.

4.1.2 Technologien

Unter anderem die Standards für die folgenden Bereiche werden durch das W3C Konsortium beziehungsweise eine Untergruppe gepflegt:

- HTML
- CSS
- WebAssembly

4.1.3 Teilhabe von Außen

Die Untergruppen ("Charters") des W3Cs erlauben es Mitglieder*innen Beiträge ("Submissions") einzubringen, welche dann bei einem Treffen der Arbeitsgruppe diskutiert und potentiell im Standardisierungsprozess auf die nächste Stufe gehoben werden (w3c).

Eine Teilhabe von Außen ist im Standardisierungsprozess, abseits von allgemeinen Rückmeldungen (genaueres im folgenden Abschnitt), nicht vorgesehen, wird aber oftmals durch Mitglieder*innen der Arbeitsgruppen ermöglicht, indem diese stellvertretend einen Beitrag an das Kommittee herantragen.

4.1.4 Standardisierungsprozess

Für alle Arbeitsgruppen des W3C gilt im Allgemeinen folgender Standardisierungsprozess (w3c):

W3C Submission

Ein W3C Mitglied trägt einen Beitrag für den Web Standard an das Kommittee heran. Dieses entscheidet ob der Beitrag weiter verfolgt und diskutiert wird.

W3C Notes

Sollte ein Beitrag weiter verfolgt werden, so wird dieser zu einer "Notiz" gemacht und veröffentlicht. Eine Veröffentlichung ist allerdings nur eine Diskussionsgrundlage und hat noch keinerlei bindende Bedeutung. Zu diesem Zeitpunkt kann sich die Notiz noch zu jedem Zeitpunkt ändern oder gar entfernt werden.

W3C Working Groups

Sollte eine W3C Note vom W3C akzeptiert werden, so wird eine Arbeitsgruppe geformt, welche aus dem Beitrag einen Vorschlag für einen möglichen Standard erarbeitet. Diese hält neben dem neuen Standard auch Arbeitsprotokolle fest. Die Arbeitsgruppe fertigt zudem einen Zeitplan an.

W3C Working Drafts

Der Working Draft der Arbeitsgruppe wird vom W3C veröffentlicht, damit jeder auch von außen Feedback geben kann. Zu diesem Zeitpunkt kann der Standard noch immer jeder Zeit geändert oder verworfen werden.

W3C Candidate Recommendations

Diese Stufe ist optional und wird nur für besonders komplexe Standardänderungen verwendet. Diese Candidate Recommendations können festere potentielle Versionen des neuen Standards beinhalten, die es zum Beispiel erlauben testweise Implementierungen umzusetzen. Es können mehrere Candidate Recommendations für einen Standard zeitgleich bestehen, um zum Beispiel alternative Ansätze zu verfolgen. Ebenfalls können diese Ansätze, ähnlich den Working Drafts, noch nicht als stabil angesehen werden.

W3C Proposed Recommendations

Eine Proposed Recommendation ist das abschließende Artefakt einer Arbeitsgruppe. Auch wenn offiziell eine Proposed Recommendation noch keine Stabilitätsgarantien mitbringt, so werden inhaltlich im Normalfall kaum noch Änderungen akzeptiert und auch zeitlich ist die finale Standardisierung "nah".

W3C Recommendation

Das W3C hat den Standard offiziell aufgenommen und die Mitglieder*innen sowie der/die Direktor*in haben dem Standard zugestimmt. An dieser Stelle darf der Standardisierungsprozess als abgeschlossen angesehen werden.

4.1.5 Anmerkung zu WebAssembly

WebAssembly wird von einer Community Group (CG) des W3C spezifiziert und folgt leicht abweichenden Prozessen, die deutlich mehr Beitragsmöglichkeiten von außen bieten (WASM CG). So ist es zum Beispiel möglich, Änderungsvorschläge in den GitHub Repositories beizutragen und zu diskutieren. Zudem hat sich die Bytecode Alliance, als Gruppe von Unternehmen mit dem Ziel Implementationen und Standards im WebAssembly Feld voranzutreiben, gegründet (Bytecode Alliance).

Im Rahmen dieser Arbeit werden die Besonderheiten der WebAssembly Spezifikation nicht weiter betrachtet.

4.2 TC39

Das Technical Committee 39 (TC39) der ECMA-International Association behandelt die Spezifikation von ECMAScript (allgemein als JavaScript bekannt). Wie sich im Folgenden zeigt, ist der Standardisierungsprozess signifikant offener und hat klarere, praxisorientierte Leitlinien.

4.2.1 Zusammensetzung

Das Technical Committee (TC) besteht aus 123 Personen (tc3, c) von 49 Unternehmen (tc3, a). Anzumerken ist jedoch, dass die Mitarbeit von Außen im Prozess erwünscht ist.

4.2.2 Technologien

Das TC39 beschäftigt sich ausschließlich mit der Standardisierung von ECMAScript.

4.2.3 Teilhabe von Außen

Der Prozess des TC39 erlaubt es Außenstehenden sowohl neue Standards und Standardänderungen einzubringen, als auch existierende Standardproposals zu diskutieren. Um dies zu begünstigen, gibt es an vielen Stellen keine formellen Vorgaben für Beiträge zum Standard und der Prozess wird offen auf GitHub gelebt.

Es wird lediglich ein "Champion" (Vertreter*innen aus dem Kommittee) ab Stage 1 für jedes Proposal benötigt, der aus Sicht des Kommittees für einen Beitrag verantwortlich ist. Zudem gibt es Treffen des Kommittees, welche Außenstehenden zwar nicht zugänglich sind, jedoch werden die Notizen der Sitzungen nachträglich veröffentlicht (tc3, b).

4.2.4 Standardisierungsprozess

Die Standardisierung im TC39 ist in sogenannte "Stages" (zu deutsch Stufen) unterteilt. Hierbei muss ein Vorschlag bei einem Treffen des Kommittees auf die nächste Stufe gehoben werden.

Die Stufen nach Standardisierungsprozess sind (tc3, d):

Stage 0

Die nullte Stufe des Prozesses ist dazu da, Beiträge zum Standard zu ermöglichen. Hier existieren keine Anforderungen an die gestellten Beiträge. Sollte ein Beitrag als sinnhaft erachtet werden, so muss sich ein Champion aus dem Kommittee finden, um den Beitrag auf Stufe 1 zu heben.

Stage 1

Auf Stufe 1 geht es darum, klarer herauszuarbeiten, warum ein Beitrag Teil des Standards werden sollte, welche Risiken der Beitrag mit sich führt und welche Form eine mögliche Umsetzung haben könnte.

Damit ein Beitrag auf Stufe 1 gehoben werden kann, müssen folgende Bedingungen erfüllt sein:

- Ein Champion ist gefunden
- Ein Text, der das Problem und die generelle Form der Lösung umschreibt, existiert
- Exemplarische Usecases und Beispiele wurden dargestellt
- Eine "High-Level" API wurde erarbeitet
- Algorithmen, Abstraktionen und Semantiken wurden diskutiert
- Potentielle Querabhängigkeiten, Risiken und Komplexitäten wurden identifiziert
- Ein öffentliches Repository für die Arbeiten an dem Beitrag wurde erstellt

Sollte ein Beitrag auf Stufe 1 gehoben werden, so heißt dies, dass das Kommittee erwartet, in Zukunft Zeit in den Beitrag zu investieren, um diesen genauer zu untersuchen. Es kann erwartet werden, dass es noch große Änderungen zum endgültigen Standard gibt.

Eine Demo und/oder ein existierender "Polyfill" sind wünschenswert, aber nicht zwingend notwendig.

Stage 2

Stufe 2 dient dazu, die Syntax und Semantik des Beitrags mit formaler Spezifikationssprache zu beschreiben.

Damit ein Beitrag auf Stufe 2 gehoben werden kann, müssen folgende Bedingungen erfüllt sein:

- Die Bedingungen der vorangegangenen Stufen
- Ein initialer Spezifikationstext existiert

Die Spezifikation sollte zu diesem Zeitpunkt bereits alle größeren Semantik, Syntax und API Änderungen enthalten, aber Platzhalter sind noch erlaubt.

Sollte ein Beitrag auf Stufe 2 gehoben werden, so heißt dies, dass das Kommittee erwartet den Beitrag irgendwann in den Standard aufzunehmen. Von hier an sollten nur noch inkrementelle Änderungen am Standard passieren.

Erste experimentelle Implementationen sind wünschenswert, aber nicht zwingend notwendig.

Stage 3

Mit Stufe 3 soll die Stabilisierung vorangetrieben werden. Sollte ein Beitrag Stufe 3 verlassen, so sollen Änderungen nur noch nach Absprachen mit Nutzer*innen und Implementator*innen möglich sein.

Damit ein Beitrag auf Stufe 3 gehoben werden kann, müssen folgende Bedingungen erfüllt sein:

- Die Bedingungen der vorangegangenen Stufen
- Ein vollständiger Spezifikationstext existiert
- Designierte Reviewer*innen haben den aktuellen Spezifikationstext akzeptiert
- Alle ECMAScript Editoren haben dem aktuellen Spezifikationstext zugestimmt

Die Spezifikation sollte zu diesem Zeitpunkt die gesamte Semantik, Syntax und API vollständig beschreiben.

Sollte ein Beitrag auf Stufe 3 gehoben werden, so heißt dies, dass das Kommittee den Standard als fertig ansieht und keine weitere Arbeit an dem Beitrag nötig ist. Es existieren valide Implementationen und der Standard wird nur noch in schwerwiegenden Fällen geändert.

Stage 4

Stage 4 bedeutet, dass ein Beitrag fertig ist, um in den formalen ECMAScript Standard aufgenommen zu werden.

Damit ein Beitrag auf Stufe 4 gehoben werden kann, müssen folgende Bedingungen erfüllt sein:

- Die Bedingungen der vorangegangenen Stufen
- Test262 Akzeptanztests wurden für die Hauptszenarien geschrieben und gemerged (tes)
- Mindestens zwei kompatible Implementationen, die die Tests erfüllen, existieren
- Mindestens zwei Implementationen mit signifikanter "in-the-field" Erfahrung existieren
- Eine Pullrequest an den Hauptstandard wurde gestellt
- Die ECMAScript Editoren haben der Pullrequest zugestimmt

Die Spezifikation sollte zu diesem Zeitpunkt fertig sein.

Sollte ein Beitrag in Stufe 4 akzeptiert werden, so bedeutet dies, dass der Beitrag zum nächstmöglichen Zeitpunkt in den Standard aufgenommen wird. Von hieran sollten keine Änderungen mehr am Standard geschehen.

5 Betrachtete Browser

5.1 Eingrenzung der betrachteten Browser und Browserengines

Alle existierenden Browser zu betrachten würde den Umfang dieser Arbeit signifikant erhöhen, da allein die unvollständige Liste von "caniuse" bereits 17 verschiedene Browser kennt (Deveria, 2023).

Um die Liste der betrachteten Browser einzugrenzen, wurden alle Browser zusammengefasst, die auf eine Browserengine aufbauen und dann jeweils der meistgenutzte Browser pro Engine betrachtet (zum Beispiel Chrome für Chromium, Edge, Opera, Samsung Internet und weitere).

Aus diesem Vorgehen ergibt sich laut statcounter.com die folgende Liste an betrachteten Browserengines mit jeweiligen Browsern (Statcounter, 2023):

Engine	Marktanteil	Browser	Marktanteil
Blink	>75 %	Chrome	65,43 %
Webkit	18,69 %	Safari	18,69 %
Gecko	3 %	Firefox	3 %

Tabelle 5.1: Betrachtete Browserengines und Browser

Anmerkung zu Edge: Edge basiert seit Version 79 auf Chromium und verwendet somit die Blink-Engine (Microsoft Edge Chromium, 2023). Bis einschließlich Version 18 wurde EdgeHTML verwendet. EdgeHTML wird in dieser Arbeit nicht betrachtet.

Statcounter.com wurde hier als Basis der Browserstatistiken verwendet, da nach eigenen Angaben über fünf Milliarden Seitenaufrufe pro Monat auf über 1,5 Millionen Webseiten die Datengrundlage bilden (About Statcounter, 2023). Zudem bildet Statcounter ebenfalls die Grundlage für die Statistiken von caniuse.com (Deveria, 2023).

5.2 Updatestrategien der Browser

Verschiedene Browserhersteller*innen nutzen unterschiedliche Strategien, um neue Funktionen und Standards in ihren Browsern umzusetzen ohne dabei langfristige Stabilitätsprobleme zu verursachen. Im Folgenden werden für alle betrachteten Browser einmal die wichtigsten Updatestrategien vorgestellt und verglichen. Hierbei wird jeweils nur auf die Strategien für die aktuellste beziehungsweise nächste Version eingegangen. Strategien für zum Beispiel Versionen mit Langzeitunterstützung wie Firefox ESR werden nicht betrachtet.

5.2.1 Chrome

Das Chromium Projekt hinter dem Google Chrome unterscheidet Featureänderungen im Browser in mehrere Arten, die alle eine eigene Updatestrategie haben (chr, 2021c):

- Neue Featureeinführung
- Implementierung von existierenden Standards
- Abschaffung
- Codeänderungen, die für Webentwickler*innen sichtbar sein könnten

Es ist hierbei anzumerken, dass die Art eines Features während des Prozesses geändert werden kann.

Da eine ausführliche Betrachtung aller Schritte für alle Featurearten den Rahmen dieser Arbeit zu umfangreich wäre, werden nur die hier relevanten Schritte aufgeführt und beschrieben.

Für alle Features gilt, dass der aktuelle Stand der Entwicklung in einem Chromestatus Feature festgehalten wird (chr, 2021a). Da diese Seite öffentlich zugänglich ist, können sowohl Chromium Entwickler*innen als auch Webentwickler*innen die aktuellen Entwicklungen verfolgen.

Update Frequenz

Das Chromium Projekt veröffentlicht seit Ende 2021 (ab Version 94) alle vier Wochen eine neue Version des Browsers (vorher alle sechs Wochen) (chr, 2021d).

Neue Featureeinführung

Inkubation Während dieser Phase wird das neue Feature beschrieben und eine erste Version eines möglichen neuen Standards vorbereitet. Ziel dieser Phase ist es, Feedback von der Community einzuholen und Usecases zu sammeln (chr, 2021c). Deshalb ist es vorgeschrieben, dass unter anderem der sogenannte "Explainer" (Erklärung der vorgeschlagenen Spezifikation) öffentlich zugänglich ist und in die entsprechenden Standardrepositorien gepostet wird.

Prototyp Sobald der Standardvorschlag ein detaillierter ausgearbeitetes API Design hat, kann mit der Implementierung eines Prototyps begonnen werden. Hierfür wird eine "Intent to Prototype" (Absicht zum Prototypen) eingereicht. Die Implementierung wird zu diesem Zeitpunkt nur hinter sogenannten "Feature Flags" veröffentlicht. Diese Flags können in den Einstellungen des Browsers aktiviert werden, um als Webentwickler*in das Feature zu testen (chr, 2021c). Auf diese Weise kann sichergegangen werden, dass Entwickler*innen, die dieses Feature nutzen, sich bewusst sind, dass es sich um eine experimentelle Funktion handelt, die sich noch grundlegend ändern kann.

Komplettes Feature hinter einer Feature Flag und erweitertes Feedback Sobald das Feature größtenteils vollständig implementiert ist, liegt der Fokus darauf das Feature durch den voranstehend beschriebenen Standardisierungsprozess zu bringen, Unterstützung anderer Browserhersteller*innen für dieses Feature zu erlangen und den Prototypen basierend auf dem Feedback der Community und den Standardänderungen zu verbessern (chr, 2021c). Ziel dieser Arbeit ist es, dass einmal veröffentlichte Features in Chrome nicht wieder zurückgezogen werden müssen, weil der Standard sich geändert hat oder sich keine Einigkeit im Standardkommitee finden ließ. Beides ist in der Vergangenheit vorgekommen, wie sich zum Beispiel mit den "Custom Elements V0" zeigt (cus, 2022).

(Optional) Origin Test Origin Tests (engl. Origin Trials) sind der erste Punkt, an dem Features für die Nutzung mit echten Webseitenbesucher*innen verfügbar werden. Sie funktionieren, indem Webseitenersteller*innen sich für einen Origin Trial mit einem spezifischen Origin bewerben. Sie erhalten dann einen Token, der via HTTP Header oder Meta-Tag genutzt werden kann, um im Browser das besagte Feature zu aktivieren (chr, 2022d). Auf diese Weise können die Features bereits mit echten Nutzer*innen getestet werden, jedoch können die Webseitenbetreiber*innen jeder Zeit das Feature wieder deaktivieren.

Des Weiteren haben Origin Trials neben der oben genannten Registrierung einige wichtige Unterschiede im Vergleich zu anderen Methoden wie zum Beispiel den "Vendor Prefixes", wie der damalige Google Chrome Standards Tech Lead Alex Russell in seinem Blog ausführt (Russell, 2015):

Globale Nutzungslimits Das Blink Projekt hält es nur für plausibel, Features zu entfernen, die weniger als etwa 0,03 % aller Webseitenaufrufe nutzen. Origin Trials bieten hier die Möglichkeit, ab einer gewissen Nutzung keine weiteren Tokens mehr auszustellen und damit die Verbreitung zu hindern. Zudem können Origin Trials vor allem für Origins mit mehr als 0,01 % aller Webseitenaufrufe mit einer Drosselung durch den Browser versehen werden. Wenn ein Origin also beispielsweise 0,1 % aller Webseitenaufrufe versorgt, heißt das zum Beispiel, dass nur jede zehnte Nutzer*in das Feature nutzen kann.

Automatische Selbstzerstörung des Features Jeder Origin Trial ist normalerweise für nur sechs Meilensteine verfügbar (Verlängerungen sind begründet möglich). Danach wird das Feature automatisch wieder deaktiviert, selbst wenn ein Token von der Webseite bereitgestellt wird. Früher gab es zudem eine verpflichtende einwöchige "breaking period", in der das Feature vor einem stabilen Release deaktiviert wurde. Dieses Vorgehen findet sich jedoch nicht mehr im aktuellen Prozess (chr, 2021b).

Stufenweiser Release Sollte ein Feature diese Schritte überstehen, kann es mit einem "Intent to Ship" in Canary Versionen des Browsers veröffentlicht werden, von wo aus das Feature über andere Releasekanäle langfristig in den stabilen Release wandert. Dieses Vorgehen wird genutzt, damit Entwickelnde bereits frühzeitig neue Features und

Änderungen mit den eigenen Webseiten testen können. Im Gegenzug ist die Browserversion nicht zwingend stabil und kann noch Fehler enthalten, die bis zum stabilen Release ausgebessert oder gar zurückgezogen werden (chr, 2022c).

Featureänderungen und -entfernungen

Featureänderungen und -entfernungen folgen im Chromium Projekt einem sehr ähnlichen Prozess. Dies liegt daran, dass sowohl Änderungen als auch Entfernungen "breaking changes", also nicht abwärtskompatible Änderungen sind. Dies ist auch der Grund, warum hiervon meist Abstand genommen wird und solche "breaking changes" sehr gut argumentiert werden müssen. Zudem gilt, wie bereits oben erwähnt, dass nur Features mit weniger als 0,03 % Nutzung entfernt werden sollten (chr, 2021c).

Motivation In diesem Schritt muss die Motivation hinter der Änderung dargelegt werden. Hierbei ist es wichtig herauszuarbeiten, warum die Gründe für die Änderung die möglichen Kosten und Probleme überwiegen.

Dev Trial Dev Trials aktivieren die Änderung im Canary und Dev Kanal des Browsers. Zudem wird in den anderen Versionen meist eine Feature Flag für das Feature eingeführt. Dadurch können Entwickelnde bereits frühzeitig die Änderung vorbereiten und testen und noch vor einer stabilen Veröffentlichung Feedback geben.

(Optional für Entfernungen) Deprecation Trial Für Entfernungen kann zusätzlich optional ein "Deprecation Trial" durchgeführt werden. Diese funktionieren wie umgedrehte "Origin Trials" in dem Sinne, dass nicht ein Feature vorzeitig aktiviert wird, sondern länger verfügbar bleibt. Dies wurde zum Beispiel für die Reduktion der "User-Agents" genutzt (Beyad u. Tan, 2022).

Veröffentlichung Sollten keine Probleme in den vorstehenden Schritten festgestellt werden, wird die Änderung in allen Kanälen des Browsers veröffentlicht.

5.2.2 Safari

Im Vergleich zu Firefox und dem Chromium Prozess ist Apple, welches das WebKit Projekt leitet und den Safari Browser veröffentlicht, deutlich weniger offen in den Prozessen.

So heißt es beispielsweise in den sieben Jahre alten FAQs im WebKits Trac als Antwort auf die Fragen wann Apple neue Safari Versionen veröffentlicht und ob ein Feature in der nächsten Version enthalten ist, "Apple does not comment on future releases of Safari, their timing, nor their contents." ("Apple kommentiert keine zukünftigen Versionen von Safari, deren Zeitplan oder deren Inhalte.") (web, 2016). Auch wenn das WebKit Trac inzwischen größtenteils durch das GitHub Wiki des Projekts abgelöst wurde, so findet sich im GitHub Wiki keine vergleichbare Aussage, es wird aber auf die alte Trac Instanz verwiesen (web, 2022b).

WebKit Docs Sehr spät während der Erstellung dieser Arbeit hat Apple die neue Dokumentationsseite für das WebKit Projekt "https://docs.webkit.org/" veröffentlicht (ann, 2023). Es wurde mit Hilfe der dort verfügbaren Suchfunktion und bei einem einfachen Durchsehen der Navigationsmenüs nach neuen Informationen gesucht, jedoch scheint die Seite aktuell nur Informationen aus dem WebKit Trac und dem WebKit GitHub Projekt zu enthalten und neu aufzubereiten.

Update Frequenz

Safari veröffentlicht im Vergleich zu Firefox und Chrome am seltensten neue Updates. Dies ist in der Grafik 5.1 ersichtlich, welche in Kapitel 10 genauer erläutert wird.

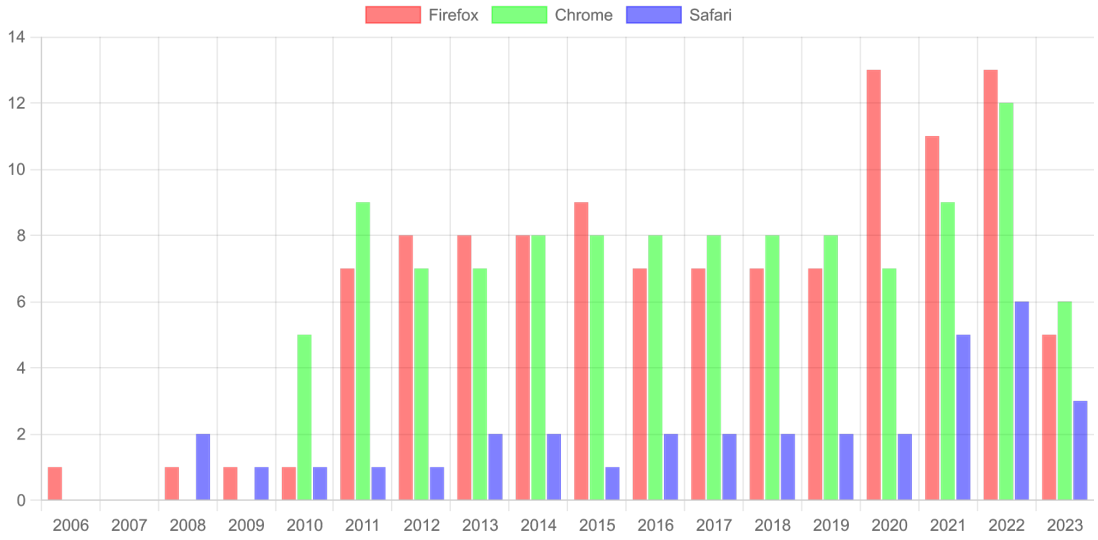

Abbildung 5.1: Browser Releases pro Jahr

Featureflags

Safari nutzt ähnlich wie Chrome Featureflags, um Entwickelnden vorab die Möglichkeit zu geben, zukünftige Funktionen zu testen. Jedoch ist hier anzumerken, dass Apple weder das Hinzufügen, noch das Entfernen von einzelnen Flags vorab angekündigt.

Featureflags im WebKit Projekt unterscheiden sich in zwei Arten. Zum einen die Runtime Flags, welche zur Laufzeit zum Beispiel über die Einstellungen geändert werden können und Compiler Flags, welche bereits beim Bauen des Browsers gesetzt werden müssen (web, 2022a).

Die folgenden Absätze beschreiben beide Arten an Featureflags im WebKit Projekt basierend auf der WebKit Feature Policy (web, 2022a).

Runtime Flags Die Runtime Flags können unter iOS in den Einstellungen wie in Abbildung 5.2 zu sehen verwaltet werden.

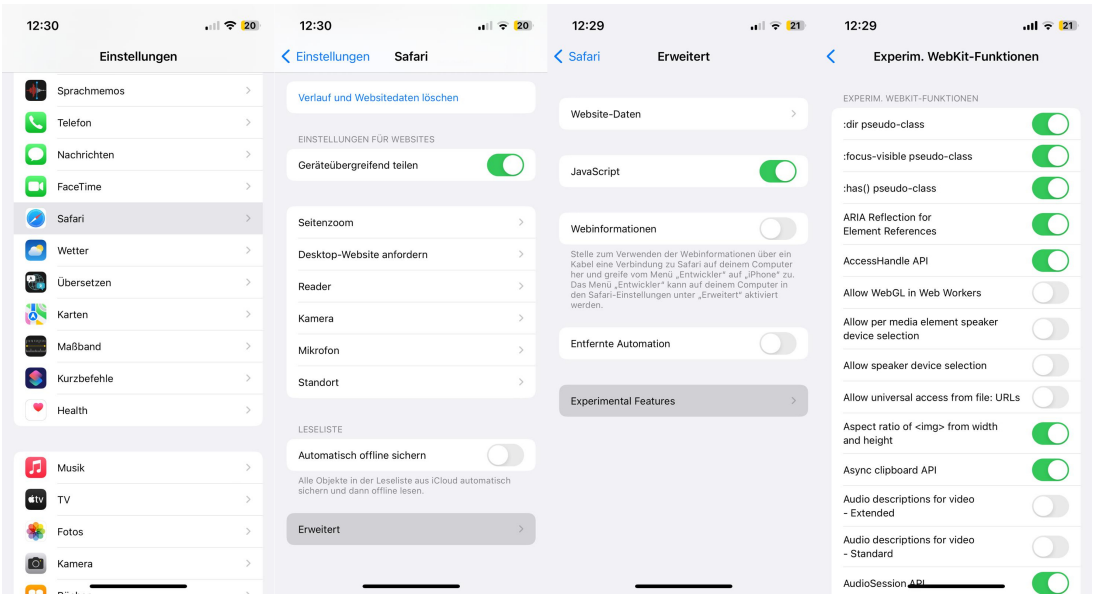

Abbildung 5.2: iOS Safari Einstellungen Feature Flags

Sie sind im WebKit Projekt die bevorzugte Variante neue Features zu testen, da sie es erlauben, Funktionen nur für Entwickler*innen zu aktivieren und diesen die Kontrolle geben, welche Funktionen sie testen wollen.

Runtime Flags starten normal standardmäßig deaktiviert und können bei Bedarf aktiviert werden. Sollte eine neue Funktion stabil genug sein, dass eine inkompatible Änderung in Zukunft nicht mehr zu erwarten ist, so kann die Flag standardmäßig aktiviert werden.

Eine Runtime Flag wird normalerweise entfernt, wenn die Deaktivierung der Funktion nicht mehr geplant ist.

Compiler Flags Die Compiler Flags werden bereits beim Bauen des Browsers gesetzt und können nicht zur Laufzeit geändert werden. Aus diesem Grund sollen sie im WebKit Projekt nur genutzt werden, wenn Runtime Flags nicht ausreichen. Compiler Flags können in Kombination mit Runtime Flags genutzt werden, indem eine Funktion durch eine Compiler Flag ermöglicht wird, aber erst mit einer Runtime Flag aktiviert wird.

5.2.3 Firefox

Mozilla setzt im Firefox Projekt auf ähnliche Strategien wie Google im Chromium Projekt. Die Kommunikation ist ebenfalls im Vergleich mit Apples Safari und dem darunterliegenden WebKit Projekt deutlich offener.

Im Gegenzug zu dem Chromium Projekt ist die Außenkommunikation im Firefox deutlich weniger formalisiert und ebenfalls nicht zentralisiert. So finden sich im Bezug auf experimentelle Funktionen Informationen im Mozilla Developer Network (mdn, 2023a) und im Mozilla Wiki (moz, 2022b).

Getrackt wird der Prozess für neue und alte Funktionen im Mozilla Bug Tracker (moz, 2022a).

Update Frequenz

Mozilla veröffentlicht regulär alle vier Wochen eine neue Version des Firefox (fir, 2022).

Featureflags

Wie Chrome und Safari nutzt auch Firefox Featureflags, um zukünftige Änderungen vorab Entwickelnden zur Verfügung zu stellen. Die Einstellungen hierfür sind im Firefox auf der Seite "about:config" zu finden (mdn, 2023a). Der Prozess wie entschieden wird, welche Funktionen eine Featureflag zur Einführung erhalten und wie der Prozess für solche Funktionen aussieht, ist nicht genauer beschrieben.

Origin Trials

Im Gegensatz zu Safari nutzt Firefox wie Chrome Origin Trials um experimentelle Funktionen vorab für bestimmte Origins zu aktivieren. Der Prozess hierfür ist nicht so detailliert wie im Chromium Projekt beschrieben, doch die Mozilla Wiki Seite referenziert diese als "ähnlich" (moz, 2022b). Um an einem Origin Trial teilzunehmen, muss ein Antrag im Bugtracker "Bugzilla" gestellt werden, welcher von Mozilla geprüft wird. Sollte der Antrag angenommen werden, so wird ein Token erstellt, welches im "origin-trial" HTTP Header oder in einem meta HTML Tag gesetzt werden muss, um das Feature zu aktivieren (moz, 2022b).

6 Sonderfall iOS

Wie bereits in der Problemstellung zu bemerken war, ist Apples mobiles Betriebssystem ein Sonderfall für die Webentwicklung. Dies stammt von der AppStore Review Guidelines Regel 2.5.6, welche alle Hersteller*innen von Applikationen, die "das Web browsen", dazu zwingt, die in iOS dafür eingebauten Webkit Module zu nutzen (app, 2022). Diese Regel verbietet es anderen Browserhersteller*innen, ihre eigenen Engines auf iOS zur Verfügung zu stellen und somit sind alle Browser unter iOS an die von Apple zur Verfügung gestellten Fähigkeiten gebunden und könnten nicht problemlos neue Standards wie zum Beispiel WebNFC ohne Apples Unterstützung umsetzen, wie Alex Russell, ehemaliger Google Chrome Entwickler und jetzt Projektmanager bei Microsoft Edge in seinem Blog erklärt (Russell, 2022).

Zusätzlich zu der Einschränkung der Wahl der Engine kommt erschwerend hinzu, dass Apple nicht alle Features der Webkit Engine für Dritthersteller*innen sofort bereitstellt. So wurde als Beispiel WebRTC bereits in iOS 11 mit Safari 11 im November 2017 ausgeliefert (saf, 2022a), jedoch wurde diese Funktion erst im Dezember 2020 mit iOS 14.3 für Hersteller*innen anderer Browser freigeschaltet (ios, 2022). Aus diesem Grund waren Zoom, Teams, Jitsi und andere auf WebRTC aufbauende WebApps lange Zeit nicht außerhalb von Safari nutzbar.

Unabhängig von eventuellen Funktionseinschränkungen sorgen diese Vorgaben für einen Mehraufwand in der Pflege der Bugtracker für Chromium und Firefox, da immer wieder Bugs in den Bugtrackern auftauchen, welche im darunterliegenden WebKit auftreten und nicht von Chromium oder Firefox gelöst werden können. Sucht man zum Beispiel im Chromium Bugtracker nach Bugs, die unter iOS gemeldet wurden, mit "WontFix" (dt. wird nicht behoben) geschlossen wurden und den Text "WebKit" beinhalten, so erhält man allein für die letzten fünf Jahre fast einhundert Ergebnisse (chr, 2023).

Diese drastischen Einschränkungen sind laut Apple Developer Advocat Jen Simmons mit Grund dafür, dass Google Chrome bisher unter iOS keine marktbeherrschende Stellung einnehmen konnte (Simmons, 2022). In Zukunft könnte sich diese Position aufgrund des 2024 in Kraft tretenden "Gesetz über digitale Märkte" ändern, da dieses vorraussichtlich sogenannten "Torwächtern", zu denen Apple zählen würde, verbietet, die Wahl der Engine einzuschränken (Parlament, 2022).

7 Kompatibilitätsdaten

Für die weiterführende Analyse der Entwicklung des Browsersupports unterschiedlicher Funktionen vergleichend über alle betrachteten Browser sind möglichst zuverlässige und vollständige Datenquellen notwendig.

Es wäre möglich gewesen, die Daten für diese Arbeit durch das Abtesten mit eigenen Tests alle verfügbaren Browserversionen auf gewünschte Funktionen selbst zu erstellen, allerdings wäre dies sehr zeitaufwändig und fehleranfällig gewesen und hätte wenig Zeit für die Auswertung gelassen. Aus diesem Grund fiel die Entscheidung auf die Nutzung existierende Datenquellen.

Zur Auswahl der genutzten Datenquellen wurde erst explorativ nach möglichen Datenquellen gesucht und diese dann anhand von Zuverlässigkeit, Vollständigkeit, Aktualität und Verarbeitbarkeit bewertet. Hierfür wurden die möglichen Datenquellen stichprobenhaft und basierend auf der eigenen Dokumentation gegeneinander abgewogen, abgeglichen und teilweise im Endergebnis kombiniert, um einen möglichst hochwertigen Basisdatensatz zu erhalten.

7.1 Datenquellenkandidaten

Es existieren mehrere sehr umfassende Datenquellen zur Browserkompatibilität, welche jedoch in ihrer Qualität und vor allem Aktualität sowie Fokus stark variieren. Die betrachteten Quellen werden im Folgenden näher erläutert und basierend auf den oben genannten Kriterien bewertet.

Chromestatus

Die Chromestatus Seite des Chromium Projekts bietet eine sehr umfassende Auflistung aller Funktionen in modernen Standards und einen genauen Einblick in die Position des Features im Chromium Projekt (chr, 2021a).

Stärken

- Umfassende Auflistung aller Features
- Verlinkung auf Standards und Bugtracker
- Gesammelte Standardpositionen der drei großen Browser
- Sehr detailliertes Tagging

Schwächen

- Nur aktueller Datenstand / keine historischen Daten
- Detaillierte Daten nur für Chromium Browser
- Daten werden nach Release in Chrome nicht mehr aktualisiert

Wegen der schwerwiegenden Schwächen wurde Chromestatus nur zum stichprobenhaften Abgleichen der anderen Datenquellen genutzt.

MDN Web Docs

"MDN browser compat data" ist eine von Mozilla gepflegte Sammlung an Browserkompatibilitätsdaten (mdn, 2023b).

Stärken

- Umfassende Auflistung aller Features
- Verlinkung auf Standards und MDN Dokumentation
- Einschätzung in experimentell und veraltet für alle Features
- Version des ersten Supports für alle relevanten Browser und Features
- Detaillierte Release Informationen

Schwächen

- Inkonsistentes Datenformat
- Features sind nur in einer Featuregruppe

Aufgrund des inkonsistenten Datenformats (zum Beispiel sind Supportangaben manchmal Booleans statt Strings und jede Featuregruppe hat eine andere Struktur) wurden für die abschließende Analyse nur die detaillierten Informationen über Browser Releases herangezogen. Die Daten zur Featureunterstützung wurden stichprobenhaft mit den anderen Datenquellen abgeglichen.

Can I Use

"caniuse.com" ist eine von Alexis Deveria gepflegte Sammlung an Browserkompatibilitätsdaten, die basierend auf Browsertests mit BrowserStack geprüft wird (Deveria, 2023). Zudem sind Beiträge von der Community möglich und die Daten werden mit den Daten des MDN teilweise abgeglichen (Scholz u. Deveria, 2019).

Stärken

- Maschinenfreundliches Datenformat
- Sehr umfassendes Datenset
- Enthält globale Nutzungsstatistiken
- Umfassende historische Daten
- Sinnvolle Gruppierung der Features

Schwächen

- Browserversionen ohne Featureänderungen werden zusammengefasst
- Mobile Browser nur mit eingeschränkter Historie (Deveria, 2017)

Wegen der gut maschinenlesbaren Daten wurde CanIUse als Basis für einen Großteil der Analysen genutzt. Hierbei wurden die Daten mit den Daten von MDN kombiniert und stichprobenartig mit allen Datenquellen abgeglichen.

Web Platform Tests

"Web Platform Tests" ist die offizielle Testsuite des W3C (wpt, 2022). Diese Testsuite wird von den Browserhersteller*innen genutzt um die Implementierung der Standards zu testen und wird regelmäßig automatisiert auf den aktuellen Desktopversionen von Chrome, Edge, Firefox und Safari ausgeführt.

Die Web Platform Tests sind mit Abstand die umfassendste Datenquelle mit Testergebnissen für über 1,8 millionen Tests.

Zusätzlich zu den Ergebnissen der Web Platform Tests wird auch das Interop Projekt, welches Teil der Web Platform Tests Seite ist, als Quelle herangezogen (int, 2023). Dieses Projekt befasst sich als Kooperation zwischen Chromium, Firefox und Safari seit 2021 jedes Jahr damit, in gemeinsam gewählten Fokusgebieten die Interoperabilität der Browser zu verbessern.

Stärken

- Umfassender Datensatz
- Basiert auf Standards
- Wird von den betrachteten Browsern mit getragen
- Daten für experimentelle und stabile Versionen der Browser
- Existierende Daten sehr zuverlässig

Schwächen

- Nur Daten für die jeweils aktuellste Version der Browser
- Features nicht gruppiert
- Features werden nach Testverfügbarkeit gewichtet

Aufgrund der hohen Zuverlässigkeit der Daten von Web Platform Tests werden sie als Referenz herangezogen und die Ergebnisse in der Analyse mit betrachtet.

8 Datenverarbeitung

Nachdem im voranstehenden Kapitel die Datenquellen beschrieben und verglichen wurden, befasst sich dieses Kapitel damit, wie die Daten aus den Quellen zusammengetragen und verarbeitet wurden. Dabei wird ebenfalls beschrieben, wie die Daten vorbereitend auf die Analyse strukturiert und gespeichert wurden und wie eine automatisierte Verarbeitungspipeline für die Analysen entwickelt wurde.

8.1 Datenerfassung

Die Datenquellen, die für diese Arbeit herangezogen wurden, bieten alle eine maschinenlesbare Schnittstelle im JSON Format. Aus diesem Grund wurde ein Node.js Skript entwickelt, welches alle Datenquellen herunterlädt und die Daten in einer gemeinsamen Datei ablegt, um diese weiterzuverarbeiten. Dieses Vorgehen ermöglicht es, alle weiteren Schritte auch ohne eine aktive Netzwerkverbindung durchzuführen und reduziert den zur Reproduktion nötigen Datensatz auf eben diese Bündelung an Rohdaten.

8.2 Datenstrukturierung

Da die Daten in allen Datenquellen in massiv unterschiedlichen Formaten und Strukturen vorliegen, wurde eine komplett neue Datenstruktur entworfen, welche Komponenten der Struktur aus den Datenquellen übernimmt und diese erweitert. Es wurde versucht, die originalen Daten möglichst vollständig in der resultierenden Struktur abzubilden, um einen Datenverlust zu vermeiden und die nachstehende Analyse nicht einzuschränken.

Das genaue Datenmodell ist in Abbildung 8.1 dargestellt. Besonders anzumerken ist hier, dass die Felder feature.first_supported und feature.is_supported nachträglich basierend auf den Supportinformationen gefüllt werden, um die Analyse zu beschleunigen.

8.3 Speicherung

Die verarbeiteten Daten in der oben genannten Struktur zu speichern wäre in JSON Dateien möglich gewesen, jedoch eignet sich eine Datenbank eher für die Abfragen einer nachfolgenden Analyse.

Nachdem die Entscheidung getroffen wurde, die Daten in einer Datenbank zu speichern, musste eine Datenbank gewählt werden. Im Folgenden werden hierfür kritische Entscheidungen vorgestellt und begründet.

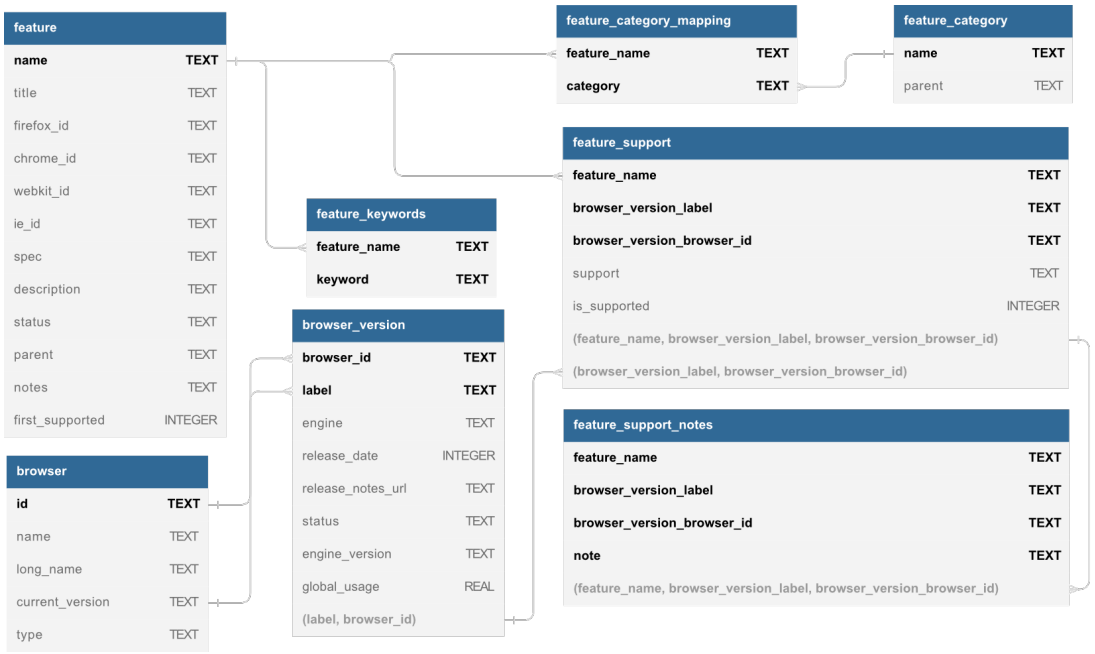

Abbildung 8.1: Struktur der erstellten Datenbank

8.3.1 Ziele der Speicherung

- Daten sollen in der oben genannten Form gespeichert werden
- Daten sollen einfach abfragbar sein
- Komplexe Abfragen sollen möglich sein
- Datenbank soll sich leicht archivieren lassen
- Keine extra Dienste

8.3.2 SQL vs. NoSQL

Da der zu speichernde Datensatz eine strikte Struktur einhält und im Falle einer Strukturänderung innerhalb weniger Minuten automatisiert neu erstellt werden kann, wurde sich für eine relationale Datenbank entschieden. Da SQL für den Autor eine vertraute Sprache für Datenbankabfragen ist, wurde die Auswahl zudem auf SQL-kompatible Datenbanken beschränkt.

8.3.3 Dateibasiert vs. Serverbasiert

Da die Daten für dieses Projekt im Rohformat nur etwa 40 MB groß sind und das Setup möglichst einfach gehalten werden sollte, wurde sich für eine dateibasierte Datenbank entschieden. Die Wahl fiel auf SQLite, da diese Datenbank eine gute Unterstützung in Node.js bietet und die Datenbank in einer einzelnen Datei gespeichert wird (npm, 2023b).

8.4 Automatisierte Datenverarbeitung

Nachdem die Wahl auf ein Datenbanksystem gefallen ist und auch die Zielstruktur der Daten festgelegt wurde, müssen die Daten aus dem Rohformat in die gewünschte Struktur überführt werden und in der Datenbank gespeichert werden. Dieser Prozess wurde ebenfalls automatisiert, um mit wenigen Schritten die Daten aktualisieren zu können. Hierfür wurde auf Node.js zur Verarbeitung gesetzt.

8.4.1 Datenkombination

Um die Daten aus einzelnen Quellen zusammenzuführen und einen gemeinsamen Datensatz zu erhalten, wurden die Daten nach Browser und Versionsnummer zugeordnet und in ein gemeinsames Schema überführt. Dies ermöglichte die Daten aus dem CanIUse Datensatz mit weiteren Informationen anzureichern.

Versionen

CanIUse fasst teilweise Browserversionen zusammen, bei denen sich die Kompatibilitätsdaten nicht geändert haben (zum Beispiel Safari Desktop Version 15.2-15.3). MDN Browser Compat hingegen listet jede Version einzeln und bietet somit detailliertere Releasedaten.

Die betrachteten Browser nutzen alle nicht "Semantic Versioning" (Preston-Werner, 2022). Apple veröffentlicht neue Safari Versionen immer zusammen mit iOS (app, 2023), Firefox inkrementiert die Versionsnummer immer mit jedem regulären Release (fir, 2023) und Chrome erhöht wie Firefox und zusätzlich bei unplanmäßigen "breaking changes" (chr, 2022b). Dies bedeutet jedoch, dass es mehr Major Versionen gibt als zwingend notwendig wären und alle Versionsnamen der betrachteten Browser mit dem SemVer Paket für Node.js parsebar sind (npm, 2023a). Zusammengefasste Versionen im CanIUse Datensatz werden immer nach dem Schema "minversion-maxversion" benannt. Diese kann man nun gegen alle MDN Versionen des Browsers abgleichen und damit eine vollständige Liste der Versionen und Kompatibilitätsdaten erhalten.

Engine

Chromium hat ursprünglich die WebKit Engine verwendet und diese 2013 "geforkt" und daraus die Blink Engine entwickelt (chr, 2013). Dieser Wechsel spiegelt sich auch in den Daten von MDN wieder und wurde in die Datenstruktur übernommen.

8.5 Visualisierung

Zur Visualisierung der Daten wurde auf Chart.js gesetzt (cha, 2023). Somit konnten mit SQL und JavaScript auch komplexe Analysen auf dem Datensatz ermöglicht werden und auf ein breites Spektrum an Diagrammtypen zurückgegriffen werden. Architekturell wurde sich für ein Skript zur Erzeugung aller Diagramme entschieden, welches die Konfiguration für jedes Diagramm enthält (Ausgabegröße, Ausgabedatei, Chart.js

Konfiguration). Die Analysefunktionen werden in diesem Skript aus Unterskripten importiert. Auf diese Weise bleibt die "Hauptdatei" übersichtlicht und die Unterskripte spezifisch für eine Analyse ohne Seiteneffekte auf andere Analysen.

8.6 Automatisierung

Wie bereits erwähnt, ist der gesamte Prozess von der Datensammlung bis zur Visualisierung weitestmöglich automatisiert. In der README.md des Begleitrepositories wurde ein Befehl dokumentiert, der alle Schritte automatisch ausführt.

Der Fluss der Daten wurde in Abbildung 8.2 einmal veranschaulicht. Die Konvertierung von SVG zu PNG wird vorgenommen, um die Diagramme in diese Arbeit einbetten zu können, da LaTeX nativ keine SVGs unterstützt. Es wurden nicht direkt PNG Dateien erstellt, um keine Details für die Analyse an das Rastering zu verlieren.

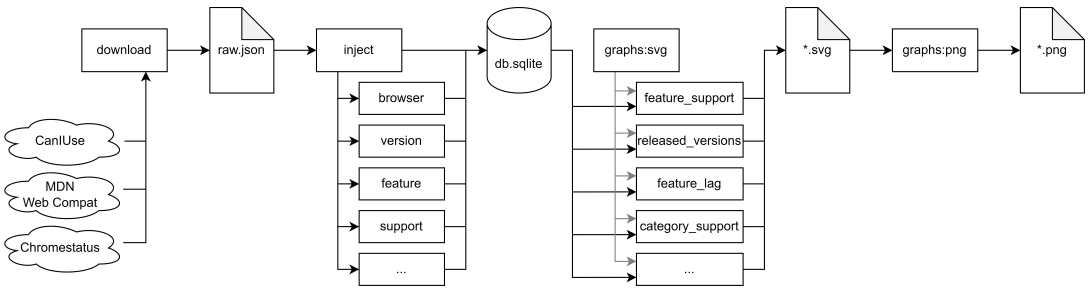

Abbildung 8.2: Datenfluss von Download bis Visualisierung

8.7 Veröffentlichung

Alle Skripte, die für diese Arbeit erstellt wurden, sind in einem öffentlichen Repository auf GitHub verfügbar (Höser, 2023). Dort findet sich ebenfalls die Dokumentation, um sämtliche Daten und Diagramme zu reproduzieren und zu aktualisieren. Der Datensatz ist in der Version der Arbeit im Repository enthalten und wird durch Ausführen eines Builds aktualisiert.

9 Erhaltener Datensatz

Bevor im nachfolgenden Kapitel mit der Analyse des Datensatzes begonnen wird, enthält dieses Kapitel noch einige allgemeine Anmerkungen und Metriken über den erarbeiteten Datensatz, um den Kontext und die Aussagekraft der Ergebnisse besser einschätzbar zu machen.

9.1 Datensatzumfang

Der betrachtete Datensatz umfasst 19 Browser mit insgesamt 497 Versionen, wobei für neun Versionen kein Releasedatum zugeordnet werden konnte, was hauptsächlich daran liegt, dass die entsprechenden Versionen noch nicht veröffentlicht wurden. 13 dieser Browser sind mobile Browser. Ebenfalls beinhaltet der Datensatz Informationen zu 536 Features in zwölf Kategorien, wovon zehn Features noch in keinem Browser implementiert sind. Die größte Tabelle des Datensatzes ist die Zuordnung von Informationen über die Featureunterstützung zu den Browserversionen mit insgesamt knapp 270.000 Einträgen.

9.2 Vollständigkeit

Für die betrachteten Browser sind die genutzten Daten für die stabil veröffentlichten Versionen nahezu vollständig. Einzelne Diskrepanzen zwischen den genutzten Datenquellen sorgen für vereinzelte "NULL" Werte in der Tabelle, diese belaufen sich jedoch meist auf Spalten, die nicht für die Analysen genutzt wurden oder sind Einzelfälle (weniger als 0,1 % der Einträge).

9.3 Schwächen

9.3.1 Kategorisierung

Die Kategorien für Features im Datensatz enthalten sich teilweise gegenseitig. So gibt es zum Beispiel die Kategorien "CSS", "CSS2" und "CSS3", wobei die letzten beiden der ersten untergeordnet sind. Ebenfalls gibt es die Kategorien "Other", "SVG" und "PNG". Dies ist vor allem auffällig, weil andere (Bild-)Formate in der Kategorie "Other" eingeordnet sind.

Die Kategorisierung ist dennoch ausreichend im Kontext dieser Arbeit, da es eher um eine grobe Einteilung geht und eine genauere Einteilung aller Features in eine eigene Kategorisierung nicht in den Umfang dieser Arbeit passt.

9.3.2 Mobile Browser

Abseits vom iOS Safari enthält der Datensatz für die mobilen Varianten der Browser nur Daten für die aktuellste Version. Dies liegt daran, dass CanIUse Autor Alexis Deveria alte Versionen von Android Browsern nicht einfach testen kann. Die jeweilige aktuellen Daten für die mobilen Browser wurden während der Erstellung mehrfach stichprobenartig mit der Desktopversion verglichen und es ergab sich ein ähnliches Bild wie zwischen der Desktop und mobilen Variante des Safaris, weshalb diese Schwäche akzeptiert wurde.

9.3.3 Kein Datum der Standardisierung

Die Datenquelle CanIUse enthält kein Datum, wann ein Feature standardisiert wurde. Diese Daten zu erarbeiten wäre ein immenser manueller Aufwand, da unterschiedlichste Repositorien über die letzten 20 Jahre durchsucht und ausgewertet werden müssten. Zudem ist für Entwickler*innen signifikant wichtiger wann ein Feature in einem oder allen Browsern verfügbar ist, als wann es standardisiert wurde, da erst mit Verfügbarkeit ein Feature nutzbar ist.

9.4 Kompatibilitätsangaben

Der erarbeitete Datensatz erkennt mehrere Stufen an Kompatibilität. Diese sind:

- (Yes) Standardmäßig unterstützt
- (Almost) Teilweise unterstützt
- (No) Nicht unterstützt
- (Polyfill) Unterstützt durch Polyfill
- (Prefixed) Unterstützt mit Prefix
- (Unknown) Unbekannte Unterstützung
- (Disabled) Standardmäßig deaktiviert (Aktivierung über Feature Flags möglich)

Im Kontext dieser Arbeit werden die Stufen "Yes", "Almost" und "Polyfill" und "Prefixed" als unterstützt gewertet, da es in einer Produktionsumgebung möglich ist, dieses Feature in der stabilen Version des Browsers mit echten Nutzer*innen zu verwenden. Polyfill Unterstützung wurde hier bewusst mit einbezogen, da es möglich ist, selbst ohne native Unterstützung, ein solches Feature zu nutzen.

10 Analyse

Nachdem in Kapitel 8 auf die Datenverarbeitung eingegangen wurde, wird in diesem Kapitel die tatsächliche Analyse vorgestellt. Hierfür werden jeweils die vorgenommenen Analysen betrachtet, begründet und die daraus resultierenden Ergebnisse präsentiert. Hierbei wird sich auf die direkt aus den jeweiligen Daten abgeleiteten Ergebnisse beschränkt. Eine Interpretation dieser Ergebnisse wird im folgenden Kapitel vorgenommen. Einige Analysen beinhalten sowohl Mobile, als auch Desktopbrowser, andere beschränken sich hingegen nur auf die Desktopbrowser, da die Datenlage für mobile Browser unvollständig ist. Zudem wird gerade für den historischen Vergleich oftmals Internet Explorer (kurz IE) hinzugenommen, da dieser früher weit verbreitet war und somit einen Zeitlichen Vergleichskontext für die Zeit bis Ende 2013 bieten kann (Statista Research Department, 2023).

10.1 Globale Nutzung

Als erste Analyse wird der globale Marktanteil nach Browser verglichen, um die erhobenen Daten erstmals mit externen Datenquellen abzugleichen. Zudem erlaubt diese Analyse einen Einblick in den Einfluss, den eine Engine beziehungsweise ein Browser auf den Markt hat, da ein fehlendes Feature in einer weit verbreiteten Engine deutlich mehr Nutzer*innen betrifft und somit die Nutzung des Features potenziell für Entwickelnde uninteressanter macht. Als Vergleich werden hier die Werte aus Tabelle 5.1 herangezogen. Kleinere Abweichungen sind hierbei zu erwarten, da die Daten aus unterschiedlichen Quellen stammen und zu unterschiedlichen Zeitpunkten erhoben wurden. Im Vergleich zu den referenzierten Daten haben die Chrome Browser einen leicht geringeren Marktanteil mit ca. 63 % statt ca. 65 % und Safari einen leicht höheren Marktanteil mit ca. 20 % statt ca. 19 % (siehe Grafik 10.1).

Aus den Daten zeigt sich deutlich, dass abgesehen von Firefox die mobile Version der Browser jeweils einen deutlich höheren Marktanteil hat und auch insgesamt mobile Browser einen höheren Marktanteil als Desktopbrowser haben.

10.2 Releasekadenz

Neben dem globalen Marktanteil ist auch die Releasekadenz, die tatsächlich erreicht wurde, eine interessante Metrik, da gerade stark abweichende Werte auf eventuelle Auffälligkeiten mit einem Browser hinweisen können. Hierfür wurden zwei Graphen erstellt, welche den selben Datensatz aus unterschiedlichen Gesichtspunkten betrachten. Auf der einen Seite wurde der Abstand in Wochen zwischen zwei Releases betrachtet und auf der anderen Seite die Anzahl der Releases pro Jahr.

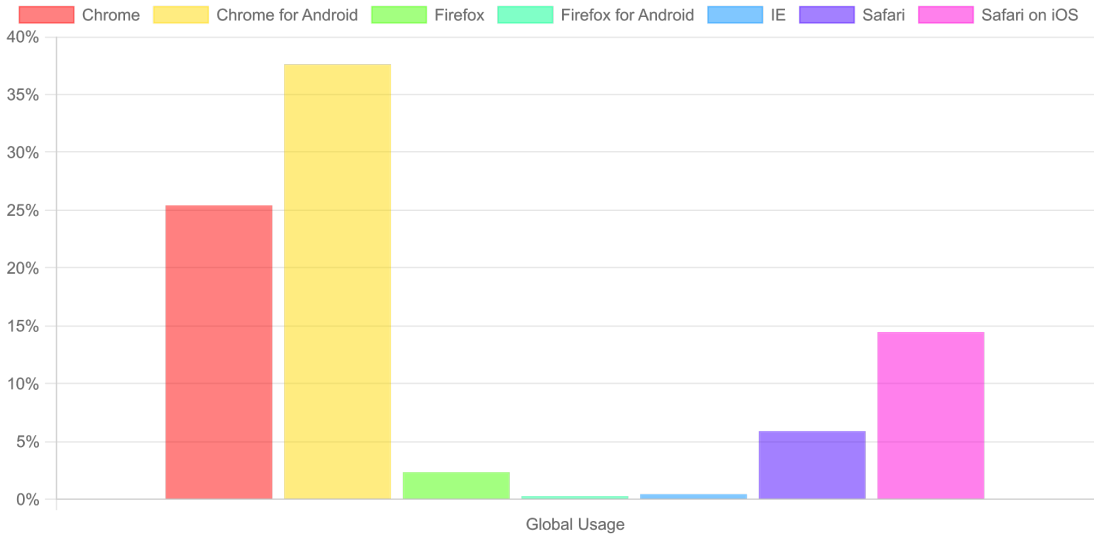

Abbildung 10.1: Globale Nutzung der Browser basierend auf dem erfassten Datensatz.

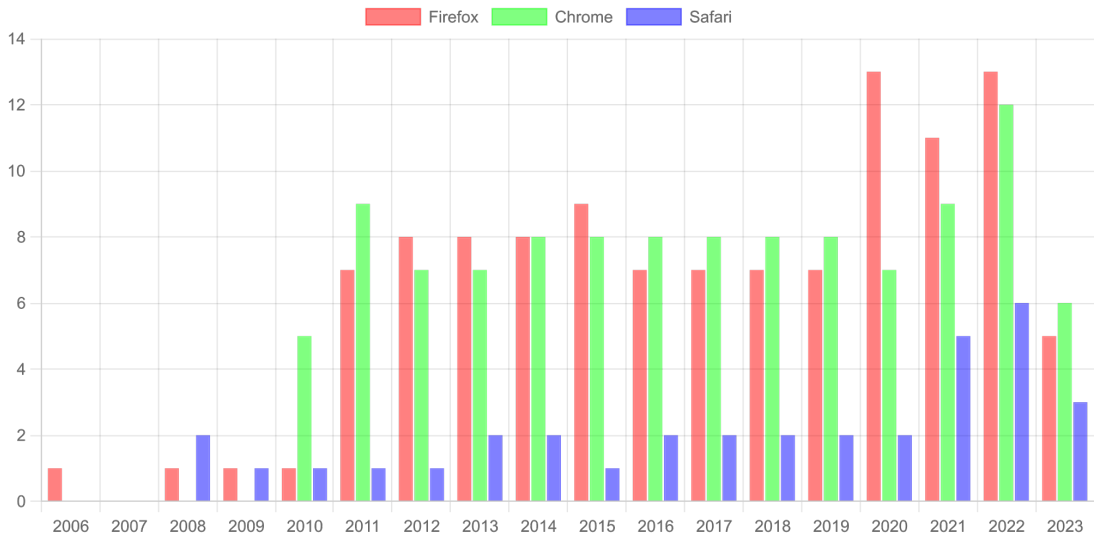

Abbildung 10.2: Veröffentlichte Versionen pro Kalenderjahr.

Auffällig ist bei den Releases pro Jahr in Grafik 10.2, dass alle Browser ihre selbstgesetzte Releasekadenz (soweit bekannt) recht genau einhalten und Chrome und Firefox etwa gleich viele Releases pro Jahr veröffentlichen. Safari hingegen veröffentlichte bis einschließlich 2020 nur etwa ein bis zwei Versionen im Jahr, was sich jedoch 2021 und 2022 deutlich geändert hat und fast an die alte Releasekadenz von Firefox und Chrome heranreicht. Diese hat sich jedoch in den letzten Jahren ebenfalls erhöht. Die Zahlen für 2023 sind hierbei noch nicht aussagekräftig, da das Jahr noch läuft.

In der Grafik 10.3 ist die Prozentangabe jeweils relativ zu der Gesamtzahl der Versionen des Browsers angegeben. Dies wurde als Darstellung gewählt, damit die drei betrachteten Browser auch bei einer stark voneinander abweichenden Zahl an Versio-

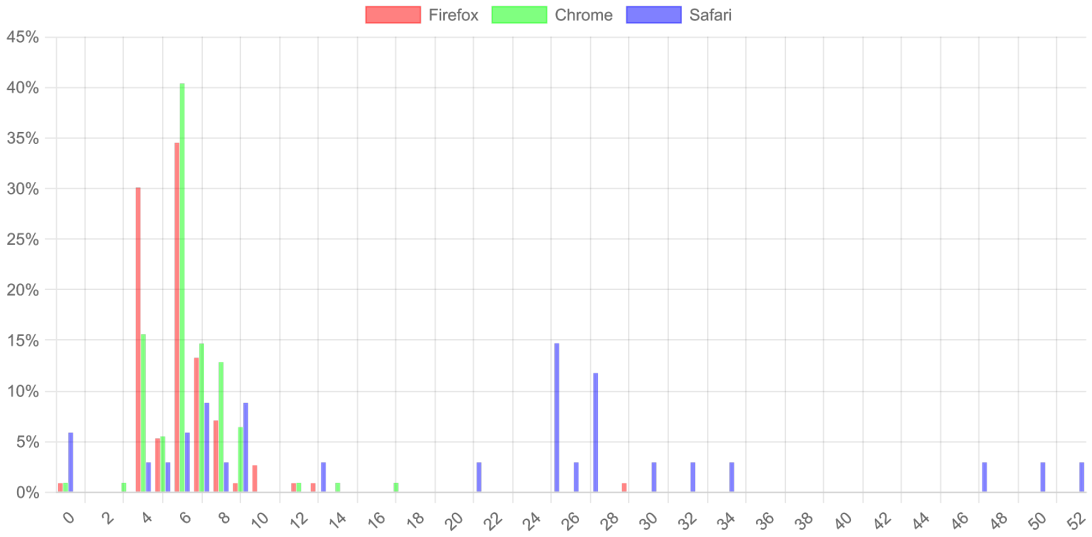

Abbildung 10.3: Zeit in Wochen zwischen veröffentlichten Versionen.

nen vergleichbar bleiben. Der betrachtete Datensatz enthält Daten für 116 Versionen des Firefox, 112 Versionen des Chromes und nur 37 Versionen des Safaris, was eine massive Abweichung darstellt.

Für die Zeit zwischen zwei Versionen zeigt sich sehr deutlich, dass sowohl Chrome, als auch Firefox ihre Releasekadenz von vier und früher sechs Wochen in der Vergangenheit recht genau eingehalten haben. Bei Safari hingegen ist kein so deutlicher Trend zu erkennen. Hier gibt es zwar eine Häufung bei 25 und 27 Wochen, jedoch sind nur etwa 30 % der Versionen mit einer Kadenz von 25 - 27 Wochen veröffentlicht worden. Dagegen steht eine weite Streuung an Kadenzwerten, die von 0 bis 52 Wochen reichen. Bei Chrome und Firefox sind Streuungen auf mehr als 10 Wochen nur vereinzelte Ausreißer.

10.3 Unterstützung aller Features

Nachdem die vorhergehenden Analysen nicht mit dem Featuresupport zu tun hatten, wird in dieser Analyse das erste Mal vergleichend auf eben diesen eingegangen. Hierfür werden für alle vorhandenen Browserversionen der Prozentsatz aller unterstützten Features pro Version über die Zeit aufgezeichnet. Wichtig hierbei ist, dass die jetzige Liste aller Features als 100 % Basis genommen wird. Dies beinhaltet auch Features, die nachträglich wieder aus dem Standard entfernt wurden oder es gar nicht bis zur endgültigen Standardisierung geschafft haben (wie zum Beispiel Custom Elements V0). Diese Features wurden hier dennoch nicht herausgefiltert, um Browser, die frühzeitig Aufwand in die Umsetzung dieser Features gesteckt haben, nicht zu benachteiligen. Als Darstellungsform für die Linien wurde eine schrittweise Erhöhung bei jeder neuen Version gewählt, da bis zum Release der neuen Version die allgemein verfügbare Unterstützung noch auf dem Stand der alten Version steht. Vorabversionen der Browser wären mit einem linear interpolierten Graphen zwar besser dargestellt, allerdings

fehlen hier Daten und zudem sind Vorabversionen nicht für den produktiven Einsatz gedacht.

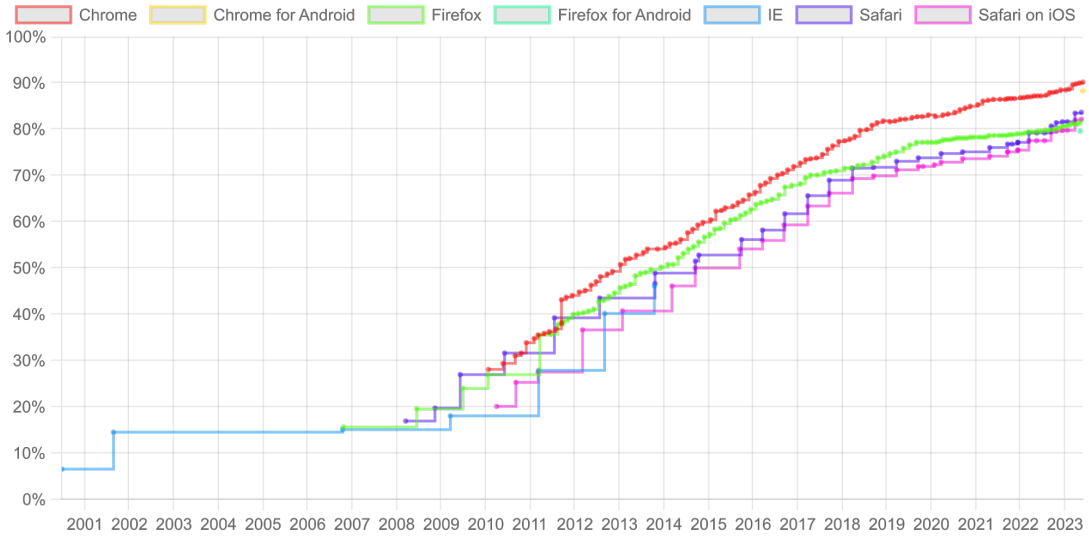

Abbildung 10.4: Prozent der unterstützten Features in allen Browserversionen über die Zeit. (Höher ist besser)

Wichtig anzumerken ist hier für den allgemeinen Datensatz, dass die Android Versionen von Firefox und Chrome nur die jeweils aktuelle Version kennen. Aus diesem Grund werden in den meisten nachfolgenden Analysen nur die Desktopvarianten der Browser betrachtet. Die iOS Versionen von Chrome und Firefox werden wegen den AppStore Richtlinien, welche in Kapitel 6 näher erläutert wurden, nicht betrachtet.

Die Grafik 10.4 gibt nicht nur Aufschluss darüber, wie sich die Unterstützung im Laufe der Zeit geändert hat, sondern auch darüber, wann Versionen der Browser veröffentlicht wurden. So sieht man hier zum Beispiel deutlich, dass Safari im Zeitraum von 2009 bis 2016 nur etwa eine Version pro Jahr veröffentlicht hat und seit Ende 2021 die Releasekadenz deutlich angezogen hat, was die Häufung an Releaseabständen mit sechs bis 10 Wochen in Grafik 10.3 erklärt. Eine ähnliche Änderung lässt sich 2010 auch bei Firefox erkennen. Bemerkenswert ist zudem, dass Internet Explorer zwischen Herbst 2001 und Winter 2006 in einem Zeitraum von über 5 Jahren keine neue Version veröffentlicht hat.

In der Grafik 10.4 ist ebenfalls deutlich zu erkennen, dass sich zwischen 2013 und 2022 eine klare Rangordnung in Bezug auf den allgemeinen Featuresupportanteil ablesen lässt. In diesem Zeitraum lag Chrome teilweise sehr deutlich vor Firefox, welcher wiederum teilweise genauso deutlich mit knapp 10 % im Herbst 2016 vor Safari lag. Dieser Abstand zwischen Firefox und Safari hat sich jedoch spätestens seit 2018 deutlich verringert und 2022 hat Safari Firefox erstmals über mehrere Versionen hinweg überholt.

Bezüglich mobiler Versionen lässt sich erkennen, dass bei allen drei Browsern die mobile Version hinter der Desktopversion hinterher hängt. Besonders bei Safari ist ablesbar (da hier historische Daten vorhanden sind), dass die Desktopversion bis Herbst

2015 immer zur Jahresmitte veröffentlicht wurde, während die mobile Version immer zum Jahresanfang des Folgejahres veröffentlicht wurde. Seit 2015 werden die beiden Varianten jedoch fast zeitgleich veröffentlicht, die mobile Version jedoch teilweise bis zu einer Woche früher.

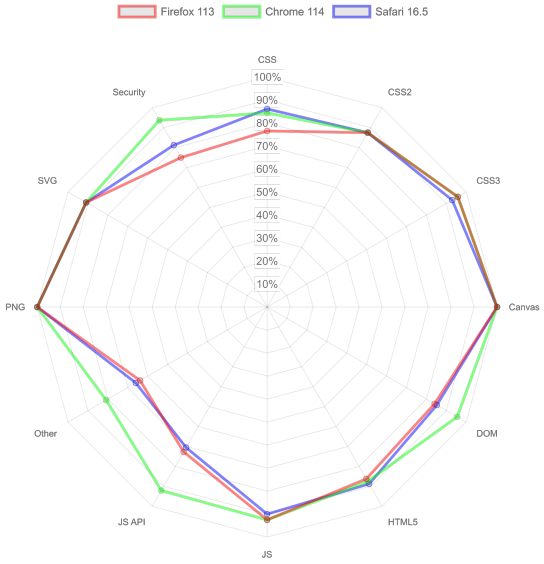

Abbildung 10.5: Prozent an Features pro Kategorie, die in der aktuellen Version unterstützt werden. (Höher ist besser)

Schlüsselt man die Unterstützung der aktuellsten Browserversionen nach Kategorie auf, so lassen sich potenziell erste Fokusbereiche der Browser erkennen. Hierfür wurde in der Grafik 10.5 die Anzahl aller Features in einer Kategorie als 100 % Wert genommen und für jeden der Browser aufgetragen, welchen Anteil der Features in einer Kategorie dieser unterstützt. Eine tiefere Bewertung der Grafik wird in dem Kapitel 11 folgen. An dieser Stelle ist nur anzumerken, dass Firefox und Safari recht gleichauf wirken und Chrome in mehreren Kategorien als "Ausreißer" nach oben auffällt.

10.4 Unterstützung aktueller Features über die Zeit

Die vorherige Analyse gibt bereits erste Aufschlüsse darüber, welche Browser tendenziell mehr Features unterstützen als andere. Da jedoch alle Features des Datensatzes als 100 % Basis genommen werden, fallen hier auch solche mit hinein, die zum Zeitpunkt der Veröffentlichung einer Browserversion noch nicht existiert haben. Um dies auszugleichen, wurde eine weitere Analyse getätigt, welche nur Funktionen betrachtet, die zum Releasezeitpunkt einer Browserversion bereits von mindestens einem Browser unterstützt wurden. Bei dieser Metrik kann jedoch ein Release für einen anderen Browser die eigene Metrik beeinflussen. Aus diesem Grund wird nicht ein Datenpunkt pro Browserrelease, sondern ein Datenpunkt pro Release in einem Browser erhoben. Für den Internet Explorer wurde in der Grafik 10.6 nur die Daten bis zum letzten Release betrachtet, um die Grafik nicht mit Daten zu füllen, die keine Aussagekraft haben,

da der Browser nicht mehr weiterentwickelt wird. Ebenfalls ist anzumerken, dass die Unterstützung in irgendeinem Browser als Datenbasis genommen wird und nicht die Unterstützung in einem der dargestellten Browser. Dies ergibt jedoch so gut wie keine Unterschiede, da fast alle betrachteten Features zuerst in einem der betrachteten Browser unterstützt wurden. Da für die mobilen Versionen von Chrome und Firefox keine historischen Daten vorhanden sind, werden diese hier nicht betrachtet.

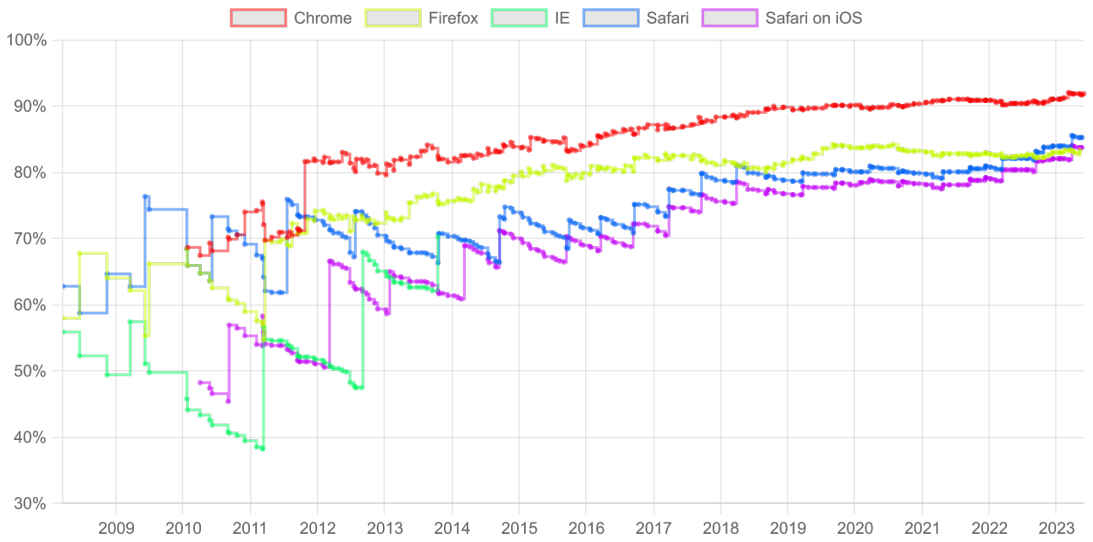

Abbildung 10.6: Anteil unterstützter Features, die bereits von mindestens einem Browser unterstützt werden. (Höher ist besser)

Die Grafik 10.6 sieht gerade zum Ende hin sehr ähnlich aus wie die Grafik 10.4, da hier die Datenbasis quasi vollständig ist. Gerade die erste Hälfte unterscheidet sich hingegen stark. Dies ermöglicht einen klareren Vergleich zwischen den Browsern, da die Browser anhand der jeweils aktuellen Features verglichen werden. Ebenfalls auffällig ist, dass Browser, die längere Releasezyklen haben, zwischen zwei Releases tendenziell in ihrer Bewertung sinken, da Marktbegleiter durch Weiterentwicklungen die 100 %-Grenze anheben. Damit zusammenhängend haben einige Entwicklungen und Werte in diesem Graphen eine besondere Bedeutung. Läge zum Beispiel ein Browser bei 100 %, so hieße dies, dass dieser alle Features unterstützt und diese zudem mit als Erstes unterstützt hat. Eine fallende Kurve bedeutet (wie zuvor angemerkt), dass der Browser nicht mit den aktuellen Entwicklungen der Platform und den Marktbegleitern mithalten kann. Eine waagerechte Kurve weist auf ein Mithalten hin und eine steigende Kurve deutet auf eine Entwicklung hin, die schneller als der Schnitt voranschreitet.

10.5 Treiber neuer Features

Abseits von der Betrachtung des aktuellen und historischen "Ist"-Zustands ist ebenfalls, wie oben beschrieben, eine Betrachtung von Nachläufern und Treibern der Platform interessant. Erste Trends hierzu kann man bereits aus der voranstehenden Analyse ableiten, doch diese und die nächste Analyse gehen hier einen Schritt tiefer. Zuerst wer-

den potenzielle Treiber betrachtet. Hierfür wurde in Grafik 10.7 gemessen, wie viele Features ein Browser pro Jahr als erstes unterstützt hat. Um Rücksicht auf gewollt gleichzeitig ausgerollte Unterstützung zu nehmen, wurde eine Kulanzzeit von einer Woche eingebaut. Sollte also Beispielsweise Feature X als Erstes am 01.01.2023 von Browser A unterstützt werden und ab dem 06.01.2023 von Browser B, so werden beide Browser als "Treiber" für dieses Feature gezählt. Eine Woche wurde hier als Zeitraum gewählt, da dies lang genug ist, um typische Absprachen zu erlauben, jedoch gleichzeitig kurz genug, um nicht einen ganzen Releasezyklus von vier Wochen abzuwarten. Es wurde ebenfalls der Wert auf einen Tag und einen Monat geändert, um die Auswirkungen zu betrachten, wobei die Ergebnisvarianz gering ausfiel.

Abseits von der historischen Analyse wurde zudem in Grafik 10.8 nach Kategorie getrennt aufgeschlüsselt, welcher Browser in welcher Kategorie die meisten Features als Erstes unterstützt hat.

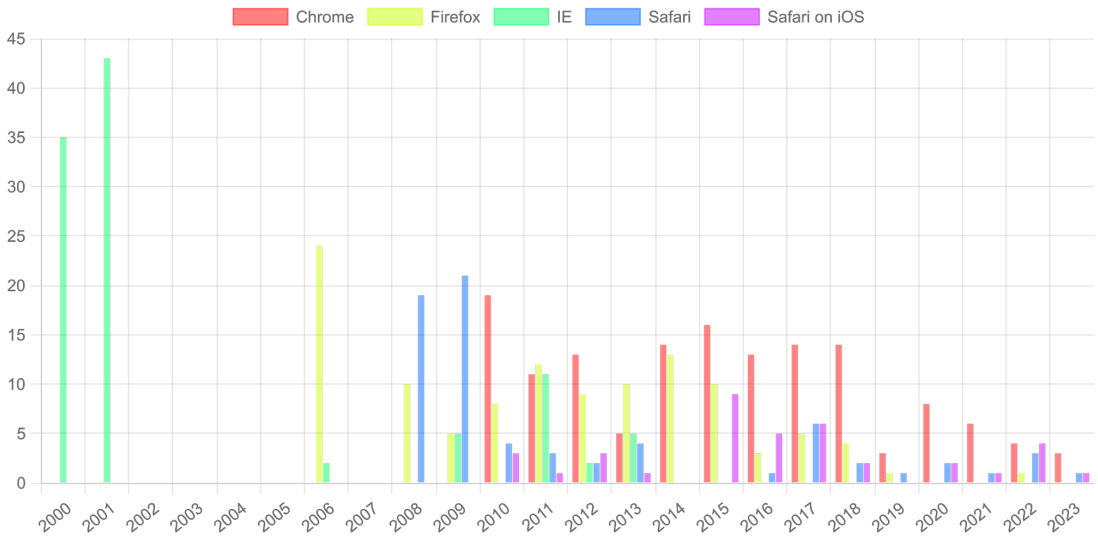

Abbildung 10.7: Anzahl neue Features, die in einem Jahr von einem Browser als Erstes unterstützt wurden. (Höher ist besser)

In Grafik 10.7 ist zu erkennen, dass es in den Jahren vor 2008 kaum neue Features in der Webplattform oder in den betrachteten Browsern gab, da teilweise über Jahre hinweg kein Browser neue Features als erstes veröffentlicht hat. Hierbei ist natürlich anzumerken, dass einige der betrachteten Browser erst später veröffentlicht wurden, weshalb eine Betrachtung erst ab 2008 wirklich Sinn ergibt. Der folgende Abschnitt befasst sich deshalb hauptsächlich mit den Jahren 2008 bis 2023.

Auffällig ist, dass die Webplattform jedes Jahr weniger neue Features hinzubekommt, was sich in den seit 2015 im Schnitt fallenden Werten in der Grafik widerspiegelt. Ebenfalls auffällig ist, dass mit der Veröffentlichung der ersten Version des Google Chrome im Jahr 2010 Safari schlagartig nicht mehr die meisten Features als erstes unterstützt. Diese Position geht ab da an fast immer an Chrome und Safari spielt in dieser Hinsicht nur noch eine untergeordnete Rolle. Bemerkenswerte Ausnahme ist hier das Jahr 2013, in dem Chrome von der WebKit Engine auf den eigenen Fork "Blink"

umgestiegen ist. In diesem Jahr hat Firefox die meisten neuen Features als Erstes unterstützt. Die Rolle des Firefox ist auch aus weiteren Gründen interessant. Während 2006 noch Firefox im Vergleich zum Internet Explorer der klare Treiber war und von 2011 bis 2014 Chrome und Firefox in der betrachteten Metrik etwa gleich auf lagen, ist Firefox seitdem als Treiber für die Webplattform immer weniger in Erscheinung getreten und hat in den letzten 5 Jahren gerade einmal zwei neue Funktionen als Erstes unterstützt.

Abbildung 10.8: Anzahl neue Features pro Kategorie, die von einem Browser als Erstes unterstützt wurden. (Höher ist besser)

Die Betrachtung nach Kategorien aufgeteilt zeichnet ebenfalls ein interessantes Bild. Hier ist Chrome ein deutlicher Treiber in den Kategorien "Security", "JS API" und "DOM", während gerade in den CSS Kategorien Firefox und Safari deutlich mehr Funktionen als Erstes unterstützen. Bemerkenswert ist in der Kategorie "HTML5", dass hier der Internet Explorer die meisten Features als Erstes unterstützt hat. Dies kann jedoch daran liegen, dass die HTML 5 Spezifikation bereits im Jahr 2008 und damit vor der ersten Version einiger der betrachteten Browser veröffentlicht wurde.

10.6 Nachläufer

Genauso interessant wie eine Betrachtung der Treiber ist eine Betrachtung der Nachläufer in diesem Bereich. Als "Nachläufer" wird in dieser Arbeit ein Browser bezeichnet, der trotz einer signifikanten Verbreitung ein Feature als Einziges nicht unterstützt. Dies ist wichtig, da Features, die von genau einem Browser nicht unterstützt werden, für viele Entwickelnde quasi nicht zur Verfügung stehen, da die Nutzung des Features potentiell zu Problemen bei den Nutzenden führt. Dies ist ein signifikantes Problem der Platform, was unter anderem dazu geführt hat, dass Google in Kooperation mit Microsoft, Mozilla und Apple 2023 das Projekt "Baseline" ins Leben gerufen hat, um

Entwickelnde darüber zu informieren welche Features in allen gängigen Browsern einsetzbar sind (pro, 2023). Besonders an dieser Analyse im Kontext dieser Arbeit ist, dass eine vergleichbarer Ansatz bereits von dem W3C in Form des "Web Platofrm Tests Dashboard" geführt wird und damit ein vergleichender Referenzpunkt existiert (wpt, 2022). In Grafik 10.9 ist der aktuelle Stand für stabile Browserversionen zum Zeitpunkt dieser Arbeit festgehalten. Als Vergleich wurde sich für die stabile Version der Grafik entschieden, da auch der Basisdatensatz für diese Arbeit hauptsächlich stabile Versionen der Browser betrachtet. Wichtig anzumerken ist hier allerdings, dass die Zahlen zwischen den eigenen Daten und denen aus der Grafik 10.9 in keiner Weise vergleichbar sind, da hier unterschiedliches gemessen wird. Während die Web Platform Tests nur eine Darstellung über die Zahl der nicht erfolgreich ausgeführten Tests ist, werden im eigenen Datensatz Features als Basis genommen. Dies ist eine wichtige Unterscheidung, da ein Feature mehrere Tests haben kann und die Zahl der Tests für ein Feature massiv voneinander abweichen kann. Ebenfalls sorgt dies dafür, dass ohne aktive Arbeit der Browserhersteller*innen steigende Werte zu erwarten sind, da die Tests laufend erweitert werden und neue Tests für existierende und zukünftige Features hinzukommen, aber selten Tests entfernt werden.

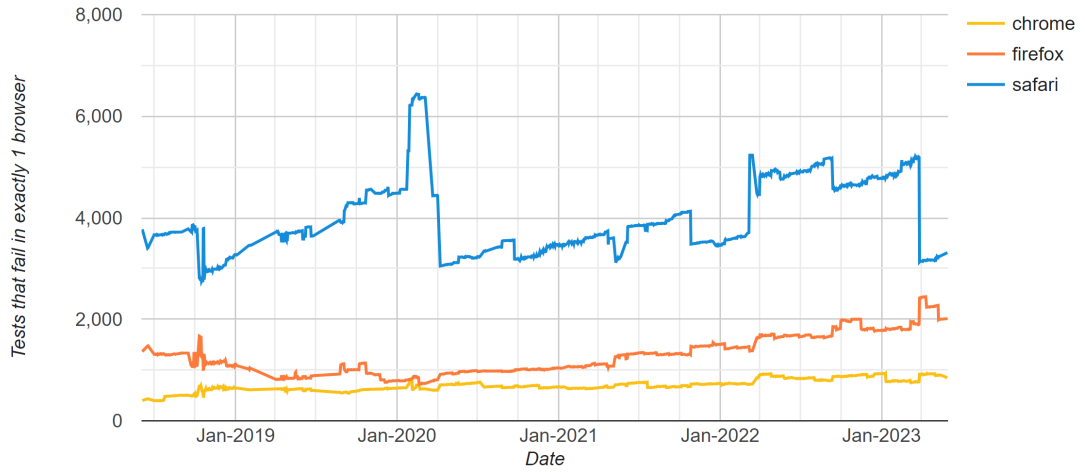

Abbildung 10.9: Anzahl Web Platform Tests Tests, die in genau einem Browser fehlschlagen. (Niedriger ist besser) Quelle: (wpt, 2022)

In Grafik 10.10 wurden eben diese Features, die nur in genau einem Browser fehlschlagen aufgezeichnet. Genau wie in Grafik 10.6 gilt auch hier, dass der Internet Explorer ab Ende 2013 mit der letzten Version aus der Grafik herausgenommen wurde, da ab diesem Zeitpunkt dieser keine Updates mehr erhalten hat und ansonsten den Graphen in der Y-Skalierung unleserlich machen würde. In diesem Kontext ist auch der Sprung von Firefox und Safari Ende 2013 zu betrachten, da Features, die zuvor in Internet Explorer und einem anderen Browser nicht unterstützt wurden, plötzlich als "nur ein Browser unterstützt das Feature nicht" gewertet werden. Für den historischen Vergleich ist der Internet Explorer jedoch relevant, weshalb er nicht komplett aus der Grafik herausgenommen wurde.

Ebenfalls auffällig sind die starken Ausreißer am Ende der Grafik, welche daher stammen, dass Browser nur miteinander verglichen werden, wenn auch eine neuere Version des Browsers existiert. Aus diesem Grund wird sich die Auswertung auf den Bereich 2009 bis Anfang 2023 beschränken. Betrachtet man die Grafik bis zum Jahr 2013, so ist klar zu erkennen, dass der Internet Explorer mit Abstand die meisten Funktionen als Einziger nicht unterstützt hat. Geht man dann weiter auf die Jahre 2015 bis 2022, so lässt sich ebenfalls eine klare Reihenfolge erkennen mit Chrome vor Firefox, der wiederum vor Safari kommt. Safari hat also in diesem Zeitraum am meisten als Nachläufer agiert. Betrachtet man jedoch die Jahre 2022 bis heute, so hat WebKit massiv an dieser Nachläuferposition gearbeitet und statt doppelt so viele Features wie Firefox als Einziges nicht zu unterstützen, ist nun Firefox der Browser mit den meisten Features, die als Einziges nicht unterstützt werden.

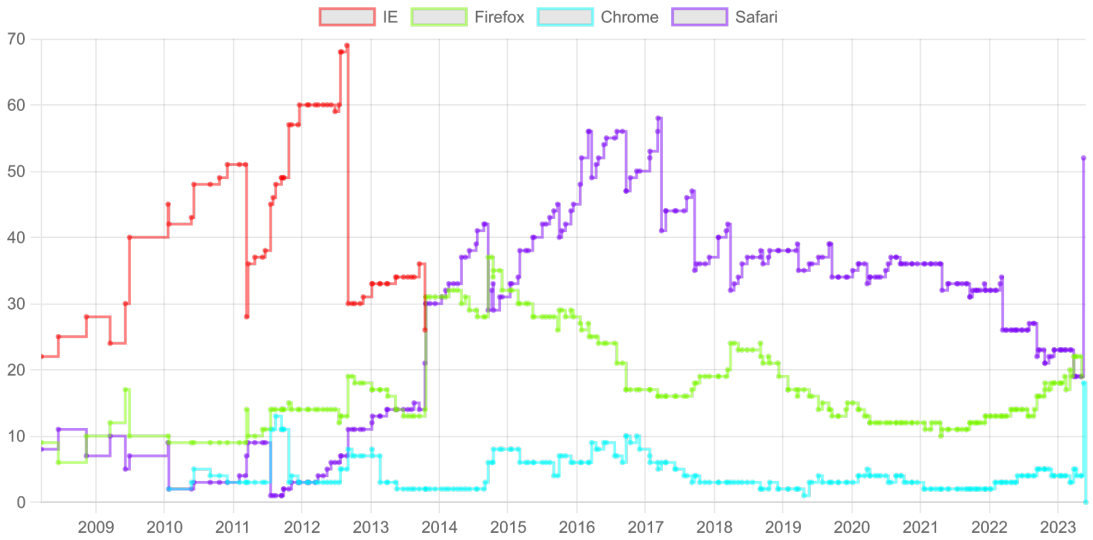

Abbildung 10.10: Anzahl Features, die in genau einem Browser nicht implementiert sind. (Niedriger ist besser)

Ähnlich wie bei den Treibern wurde auch für die Nachläufer der Datensatz für die aktuellen Versionen der Browser in Grafik 10.11 nach Kategorien aufgeschlüsselt.

Hierbei fällt auf, dass Chrome in dieser Grafik quasi nicht auftaucht. Die CSS Kategorie wird von Firefox mit dem höchsten Wert belegt und in fast allen anderen Kategorien hat Safari den höchsten Wert, teils mit deutlichem Abstand.

10.7 Instabilitäten

Abseits von der Information, ob ein Feature unterstützt oder nicht unterstützt wird, ist für Entwickelnde ebenfalls relevant, ob ein Feature korrekt und vollständig zur Verfügung steht. Zu diesem Zweck wurden zwei Indikatoren für Instabilitäten in den Daten ausgewertet und miteinander verglichen. Zum einen enthält der Datensatz eine "support" Angabe, die den Zustand "partial" (zu Deutsch "teilweise") kennt und zum anderen können Datensätze zur Unterstützung mit Notizen angereichert werden.

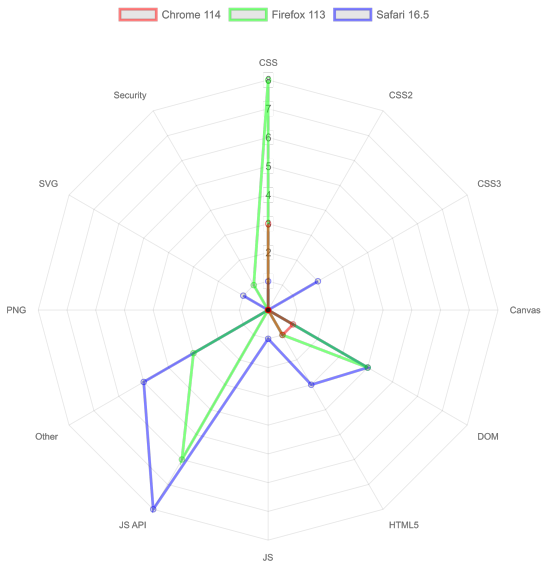

Abbildung 10.11: Anzahl Features, die in genau einem Browser nicht implementiert sind. Aufgeschlüsselt nach Kategorie (Niedriger ist besser)

Hier wurde betrachtet, wie viele grundsätzlich unterstützte Feature nach Datensatz für jede Browserversion eine Notiz enthalten. Dies deutet darauf hin, dass das Feature nicht korrekt unterstützt wird. Da die Art der Notiz nicht genauer analysiert wird (es befinden sich knapp 34.000 Notizen im Datensatz) werden die beiden oben genannten Indikatoren miteinander verglichen.

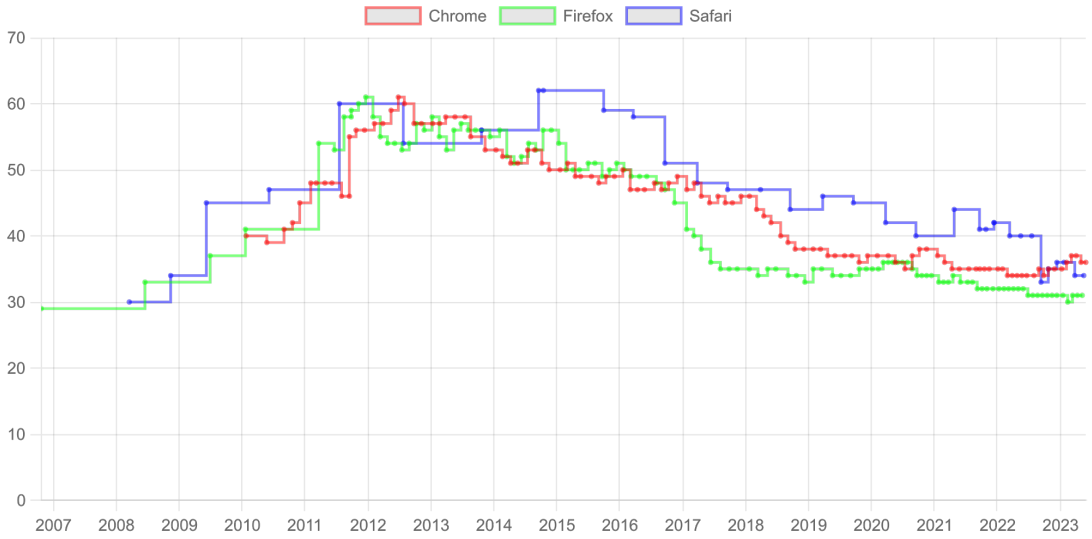

Abbildung 10.12: Anzahl Features, die nur teilweise unterstützt werden. (Niedriger ist besser)

In Grafik 10.12 sind die Anzahl der Features für jede Browserversion über die Zeit aufgetragen, die in dem CanIUse Datensatz als "teilweise" unterstützt markiert sind. Auffällig ist hier, wie ähnlich sich die drei Browser bis Ende 2014 verhalten. Ab Ende 2014 hat Safari jedoch mit Ausnahme des Jahres 2017 einen deutlich höheren Wert als Chrome und Firefox, bis sich die Werte Mitte 2022 wieder angeglichen haben und zum jetzigen Zeitpunkt Chrome die meisten teilweise unterstützten Features hat. Zu Firefox ist ebenfalls anzumerken, dass dieser um den Jahreswechsel zu 2017 eine starke Verbesserung aufweist und seitdem mit einer kurzen Ausnahme 2020 die Metrik anführt.

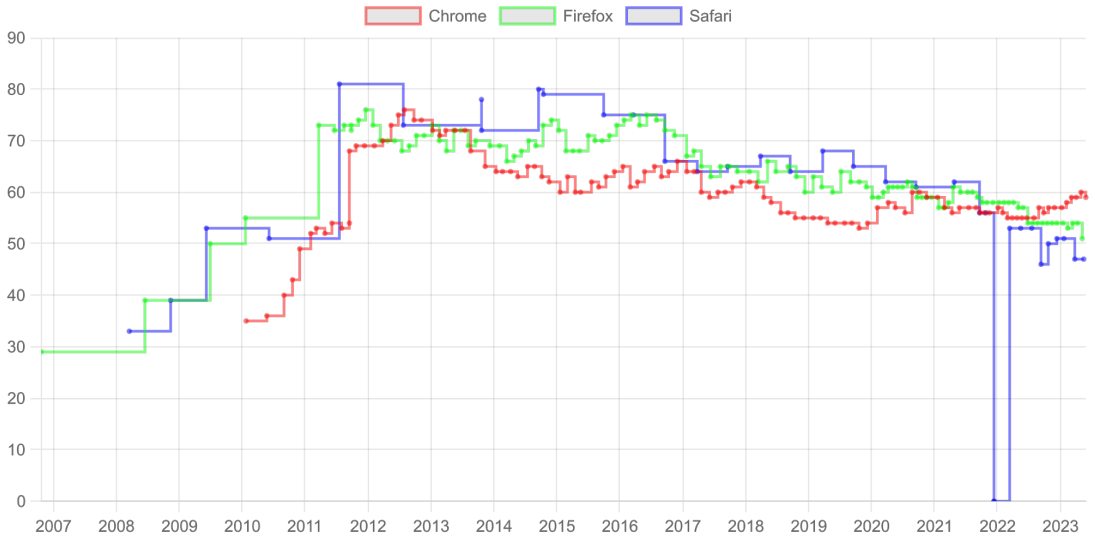

Abbildung 10.13: Anzahl unterstützter Features, die eine Notiz für die Browserversion enthalten. (Niedriger ist besser)

Vergleichend zu Grafik 10.12 wurden in Grafik 10.13 die Anzahl unterstützter Features aufgetragen, die eine Kompatibilitätsnotiz verknüpft haben. 2022 ist auffällig, dass die Linie für Safari kurzzeitig auf 0 fällt. Dies liegt daran, dass hier die Versionen "15.2" und "15.3" veröffentlicht wurden, für die der CanIUse Datensatz keine eigenen Daten aufweist. Zur besseren Reproduzierbarkeit wurden die beiden Datenpunkte nicht manuell entfernt.

Auf den ersten Blick sehen die beiden Grafiken sehr ähnlich aus und es lassen sich Segmente wiedererkennen. Einige entscheidende Punkte weichen jedoch voneinander ab. So ist die starke Verbesserung von Firefox in 2017 in Grafik 10.12 nur ein Wiederaufholen auf Chrome in Grafik 10.13. Ebenfalls verschiebt sich die Reihenfolge der Browser in Bezug auf diese Metrik. Zudem verlaufen alle Kurven in der zweiten Grafik flacher. Die Zahl der teilweise unterstützten Features halbiert sich zwischen 2012 und 2023 fast, die Anzahl der Notizen sinkt aber nur auf etwa zwei Drittel.

11 Interpretation

Dieses Kapitel befasst sich mit der Interpretation der Ergebnisse aus Kapitel 10. Hierbei werden unter anderem Verbindungen zwischen den einzelnen Analysen erarbeitet und die Ergebnisse in Bezug zu der Problemstellung aus Kapitel 1 gesetzt.

11.1 Ist Safari der neue Internet Explorer

Wie bereits in der Problemstellung referenziert, ist "Safari ist der neue Internet Explorer" ein vielfach gefundener Ausruf (saf, 2022b) (Slapka, 2022) (gerardnll, 2019).

Mit einem Blick auf die Releasekadenz von Safari bis 2015, die Unregelmäßigkeit der Releases und besonders dem schlechten Abschneiden in der Nachläuferanalyse bis 2022 (siehe Grafik 10.10) kann man diese Aussage durchaus nachvollziehen. Gerade in der Grafik 10.10 schließt sich die Linie des Safaris geradezu nahtlos an das Ende des Internet Explorers an und enthält im weiteren Verlauf auch ein ähnlich sprunghaftes Verhalten wie die des Internet Explorers.

Dennoch unterscheiden sich die Situationen hier signifikant. Internet Explorer hat zum Beispiel weitreichende, nicht mit den Standards kompatible Änderungen vorgenommen und ActiveX unterstützt (Microsoft, 2022). Dies hat den HTML Standard wiederum gezwungen ungewöhnliche Änderungen am Standard vorzunehmen, weil die Nutzung einiger Features im Internet Explorer zu weit verbreitet war. So gibt auch heute noch die Konvertierung von "document.all" in ein Boolean "false" zurück, auch wenn es nach Regeln der Sprache eigentlich "true" sein müsste (doc, 2023). Safari hingegen stellt sich zwar immer wieder offen gegen die Umsetzung standardisierter Features oder schiebt diese Teils über mehrere Jahre auf, jedoch ist dies nicht mit dem Verhalten Microsofts zu vergleichen (WebKit, 2022).

11.2 Safari Kursänderung

Wie bereits im vorherigen Kapitel angedeutet, hat sich die Situation rund um Safari in den letzten Jahren deutlich geändert. Während Firefox etwas schwächelt (auf Details wird im nächsten Abschnitt eingegangen), hat Safari die Zeit der Corona-Pandemie genutzt, um einige grundlegende Dinge zu ändern. So wurde zum Beispiel wie in Grafik 10.4 ersichtlich 2021 der Abstand zwischen zwei Releases deutlich reduziert, 2022 war Safari nach über zehn Jahren das erste Mal wieder als Vorreiter bei der Unterstützung neuer Features mit Chrome auf dem ersten Platz (siehe Grafik 10.7) und auch in den Daten für die Nachläuferposition ist eine deutliche Verbesserung zu erkennen (siehe Grafiken 10.10 und 10.9). Ebenfalls stimmen die Daten der Instabilitätsanalyse (siehe Grafik 10.12 und 10.13) einer Verbesserung zu.

Warum diese positive Wandlung in den letzten Jahren stattgefunden hat, ist aus den Ergebnissen dieser Arbeit nicht ableitbar. Dies überschneidet sich jedoch zeitlich stark mit dem Interop Projekt, welches 2021 ins Leben gerufen wurde (int, 2022).

Ebenfalls bemerkenswert ist, dass Apple mit Safari 17 eine Kursanpassung bezüglich WebApps vornimmt. Während in früheren Versionen das Hinzufügen von WebApps zum Homescreen oder Dock nur bedingt möglich war und viele wichtige APIs für PWA nicht verfügbar waren, stärkt Safari 17 die Position von WebApps auf Apple Systemen erheblich (Apple).

11.3 Firefox Schwierigkeiten

Mozilla hat 2020 250 Mitarbeiter*innen entlassen, von denen viele auch Teil des Firefox Teams waren (Brodkin, 2020). Gerade in den Grafiken 10.6 und 10.10 lässt sich dies wiedererkennen, da Firefox bis dahin auf einem stabilen Verbesserungskurs war. Ab 2020 hat sich diese Entwicklung abgeflacht und im Fall der Nachläufer sogar ab 2022 verschlechtert. Daraus lässt sich folgern, dass Firefox eventuell aktuell nicht mehr ausreichend Kapazitäten hat, um mit der Entwicklung der Platform und Safari und Chrome mitzuhalten. Dies lässt sich jedoch nicht endgültig abschätzen, denn diese Entwicklungen sind langwierig und zeichnen sich erst seit ein bis zwei Jahren wirklich ab. Es könnte also auch nur ein Berg wie 2018 sein (siehe Grafik 10.10).

Ebenfalls muss beachtet werden, dass Mozilla als Geldgeber hinter Firefox im Vergleich zu den Marktbegleitern Google und Apple auf signifikant weniger finanzielle Ressourcen zurückgreifen kann, was das Mithalten mit den anderen Browsern zusätzlich erschwert. So hat Mozilla 2021 gerade einmal etwa 200 Millionen US Dollar für jegliche Softwareentwicklung ausgegeben (Jose u. Mateo, 2022). Als Vergleich, der Google Mutterkonzern Alphabet hat im selben Jahr 31,5 Milliarden US Dollar für Forschung und Entwicklung investiert (Bianchi, 2023b).

Zusätzlich hat Firefox auch noch mit einer schwindenden Nutzerbasis zu kämpfen, was die Rechtfertigung für neue Investitionen in den Browser schwieriger gestalten könnte (Statista Research Department, 2023). Dies ist jedoch nur eine Vermutung des Autors, da zukünftige strategische Investitionsentscheidungen in Mozilla und den Browser nicht öffentlich sind.

11.4 Chrome als Vorreiter

Unter Betrachtung der Ergebnisse aus Kapitel 10 kann man Chrome allgemein als Marktführer bezeichnen. Dies spiegelt sich, wie bereits erwähnt, ebenfalls in der Nutzerbasis wider. Gerade in der Treiber- und Nachläuferanalyse hebt sich Chrome signifikant von den Marktbegleitern ab. Interessant ist hier jedoch die absolute Betrachtung nach Featurekategorien aufgeschlüsselt in Grafik 10.5. Hier fällt auf, dass Chrome in den Bereichen "JS API", "Other" und "Security" deutlich vor den anderen Browsern liegt. Dies wird umso deutlicher, wenn anstatt einer prozentualen Aufschlüsselung die absoluten Zahlen verwendet werden. Dies wurde in Grafik 11.1 einmal dargestellt.

Hier wird deutlich, dass Chrome eigentlich hauptsächlich im Bereich "JS API" den Vorsprung aufbaut und die Browser sich in den anderen Bereichen absolut betrachtet

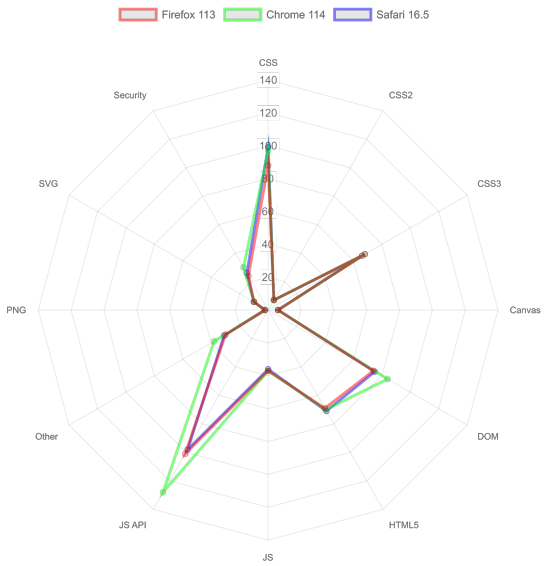

Abbildung 11.1: Anzahl Features pro Kategorie, die in der aktuellen Version unterstützt werden. (Höher ist besser)

sehr ähnlich sind. Wirft man einen genaueren Blick in den Datensatz auf die Features, die nur im Chrome aus der Kategorie JS unterstützt werden, so fällt auf, dass dies zu einem Teil neue Webstandards sind wie zum Beispiel die "CSS Painting API", der alle Browserhersteller*innen positiv gegenüber stehen, für die man Chrome als Vorreiter betrachten kann, aber zum anderen auch Features wie "WebBluetooth" oder "WebUSB", die von den anderen Browserhersteller*innen abgelehnt werden (WebKit, 2022) (Mozilla, 2023). Hier hat Google oftmals den Standardisierungsprozess mit dem Projekt Fugu vorangetrieben und liefert bereits eine Unterstützung in stabilen Browsern, obwohl der Standard aktuell noch aktiv von anderen Hersteller*innen abgelehnt wird. Dieses Verhalten kann man sehr kritisch sehen, da es dafür sorgen kann, dass Google im Zweifel seine eigenen "Standards" erstellt und hier ein Verhalten ähnlich dem alten Internet Explorer zeigt. Hier treffen potenziell zwei Philosophien aufeinander. Zum einen die des ursprünglichen Internets als Dokumentenplattform und zum anderen die des Internets als Applikationsplattform. Auf diese Thematik tiefer einzugehen und die vielschichtigen Nuancierungen der Debatte zu beleuchten passt jedoch nicht in den Kontext dieser Arbeit. Hier bleibt festzuhalten, dass basierend auf den Daten erkennbar ist, dass Google klar darauf hinarbeitet WebApps als valide Alternative zu nativen Apps zu positionieren, was auch Ziel des oben genannten Fugu Projekts ist (fug, 2022).

11.5 Wirtschaftliche Interessen

Sowohl Google als auch Apple haben neben einem Browser noch eine native Applikationsplattform mit verbundenen Stores, die Gewinne in Milliardenhöhe für die Unternehmen einbringen (Ceci, 2022). Das Web als Applikationsplattform könnte dafür sorgen, dass Gewinne statt in den Stores direkt in den WebApps getätigt werden. Dies würde dafür sorgen, dass Google und Apple nicht an diesen Umsätzen beteiligt sind. Hieraus ergibt sich ein wirtschaftlicher Interessenkonflikt zwischen der Ermöglichung des Webs als Appplattform und den Gewinnen des eigenen Stores. Google scheint hier eine aggressiv offene Strategie zu fahren, die sich, wie im voranstehenden Abschnitt gezeigt, mit einer Stärkung des Webs als Appplattform äußert. Apple hingegen schien in der Vergangenheit eher restriktiver zu sein, was sich auch in der Grafik 10.11 widerspiegelt. Gerade in den Bereichen "JS API" und "Other" ist Apple hier deutlich hinter der Konkurrenz und in genau diese Bereiche fallen viele der Fugu Features, die das Web als Appplattform attraktiver machen. Ebenfalls hat Apple teils über Jahre die Umsetzung von Funktionen im Safari herausgezögert, die für viele Appkategorien essentiell sind, wie zum Beispiel die Unterstützung von WebRTC für Live Kommunikation oder Gamepads für Spiele (Russell, 2022).

Doch auch hier ist anzumerken, dass Apple in den letzten Jahren immer mehr Features unterstützt hat und den "Feature Lag" aus Grafik 10.10 deutlich reduziert. Gerade mit den oben genannten Änderungen bezüglich einer Kursänderung von Safari kann es gut sein, dass Apple in Zukunft einen Kurs einschlagen wird, der dem von Google deutlich ähnlicher ist. Dies kann zudem durch das neue EU Gesetz für digitale Märkte bestärkt und begründet werden, welches in dieser Arbeit bereits in Kapitel 6 erläutert wurde. Sollte Apple hier nicht einen Browser bereitstellen, der mit der Konkurrenz mithalten kann, so könnten Nutzer nach Inkrafttreten des Gesetzes einfach auf alternative Browser umsteigen.

Ebenfalls festzuhalten ist, dass Google nicht nur Gewinne aus dem Store unter Android erzielt, sondern mit dem Werbenetzwerk Google AdSense ebenfalls an einem starken Web verdient. AdSense ist laut 6sense mit einem Marktanteil von über 85 % der Marktführer im Bereich Onlinewerbung und hat 2022 einen Umsatz von über 224 Milliarden US Dollar erzielt (6sense, 2023) (Bianchi, 2023a). Aus diesem Grund ist eine Reduktion des Umsatzes aus dem Store, der vermutlich zumindest teilweise durch Werbeanzeigen im Web ausgeglichen wird, für Google weniger problematisch als für Apple.

11.6 Allgemeine Entwicklung der Plattform

Betrachtet man die Entwicklung aller Browser zusammen und schließt vor allem die Ergebnisse des Projekts Interop mit ein, so zeigt sich deutlich, dass das Web als Plattform dank neuer Features immer fähiger wird und gleichzeitig die Menge an Features mit breiter Unterstützung stetig wächst (int, 2022) (int, 2023).

Mozilla demonstriert seit Jahrzehnten, dass man auch als (im Vergleich zur Konkurrenz) kleine Unternehmung die Plattform mitgestalten kann und Microsoft hat im Gegenzug mit Internet Explorer und dem Wechsel von Edge zur Chromium Engine ge-

zeigt, dass selbst große Unternehmen sich nicht durchsetzen können, wenn man nicht mit der Plattform geht. Hier zeigt sich also, dass Konkurrenz in Kombination mit offenen Standards die Gesamtsituation für Verbraucher*innen und Entwickler*innen verbessert, da diese alle Hersteller*innen dazu anleitet fortlaufend zu investieren und Fortschritt zu ermöglichen. Grafik 10.4 zeigt hier deutlich anhand des Internet Explorers bis 2008, was passiert, wenn es diese Konkurrenz nicht gibt.

Ebenfalls zeigt sich besonders aus der Grafik 10.12, dass Features mit teilweiser, beziehungsweise problematischer Unterstützung seltener werden.

12 Fazit

Abschließend für diese Arbeit lässt sich basierend auf den voranstehenden Kapiteln festhalten, dass die Webplattform, dank der offenen Standardisierung, die auch Beteiligungen von außen erlauben und der Konkurrenz auf dem Browsermarkt, in einer sich stetig verbessernden Position ist und so viele Features bereitstellt wie noch nie. Ebenfalls erhöht sich die Stabilität der umgesetzten Funktionen laufend und gerade Safari hat in den letzten Jahren seine Position bezüglich unterstützter Features deutlich verbessert.

In Bezug auf die Hauptfrage aus der Problemstellung, welcher Browser am ehesten als Treiber oder Nachläufer agiert, lässt sich Chrome zum aktuellen Zeitpunkt als Treiber gerade für neue JS Features erkennen. Bis vor wenigen Jahren hätte man zudem Safari klar als Nachläufer beschreiben können, doch dank einer erkennbaren Kursänderung seit 2020 liegen hier Safari und Firefox mehr oder weniger gleich auf.

Die Frage wie lange es dauert, bis ein neues Feature von der initialen Standardisierung bis in den Browser braucht, hat sich in dieser Arbeit leider nicht beantworten lassen, da das Datum der Standardisierung zum einen nicht mit vertretbarem Aufwand für alle Features erarbeitbar war und zum anderen erste Umsetzungen für neue Features oft schon vor der Standardisierung in den Browsern zu finden sind. Von der ersten Unterstützung bis zur Unterstützung in allen Browsern zu messen wäre ebenfalls nicht sinnvoll gewesen, da nicht alle Features überhaupt langfristig in allen Browsern unterstützt werden. Eine genauere Beantwortung dieser Frage erscheint dennoch als sinnvoll und empfiehlt sich als Leitfrage für eine weitere Arbeit.

Im Sinne der Zielsetzung wurde in dieser Arbeit erfolgreich ein Einblick in die Weiterentwickelung der Webplattform gegeben und gerade die Aspekte der Treiber- und Nachläuferpositionen beleuchtet und die Vorgehen zur Umsetzung neuer Features von Standards und Browsern erläutert. Auch wenn nicht alle Fragen beantwortet werden konnten, ist die Arbeit dennoch als Erfolg anzusehen, da ein Großteil der Problemstellung und Zielsetzung behandelt wurde und begründet werden konnte, warum die ausstehende Frage nicht im Rahmen dieser Arbeit beantwortbar war.

Als Ausblick in die Zukunft deuten die Kursänderung von Safari, Gesetzgebungen wie das "Gesetz über digitale Märkte" und das Projekt Fugu darauf hin, dass das Web als Plattform sich weiterhin entwickeln wird und neue Features und APIs bereitstellen wird. Dadurch ist abzusehen, dass die Webplattform mehr die Fähigkeiten als Applikationsplattform erhalten wird und auch in Zukunft eine wichtige Rolle in der Softwareentwicklung spielen wird.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[tc3 a] *Ecma TC39 Companies*. https://www.ecma-international.org/tc39-royalty-free-technical-committee-members/. – Letzter Zugriff: 13.06.2023

[tc3 b] *Ecma TC39 Meeting Notes*. https://github.com/tc39/notes/tree/481844fdaefbb90521b5e904ff521bbfb4b5927f. – Letzter Zugriff: 13.06.2023

[tc3 c] *Ecma TC39 People*. https://github.com/orgs/tc39/people. – Letzter Zugriff: 13.06.2023

[w3c] *The TC39 Process*. https://tc39.es/process-document/. – Letzter Zugriff: 13.06.2023

[tes] *Test262*. https://github.com/tc39/test262. – Letzter Zugriff: 13.06.2023

[tc3 d] *The W3C Process*. https://w3schools.sinsixx.com/w3c/w3c_process.asp.htm. – Letzter Zugriff: 13.06.2023

[chr 2013] *Blink: A rendering engine for the Chromium project*. https://blog.chromium.org/2013/04/blink-rendering-engine-for-chromium.html. Version: 2013. – Letzter Zugriff: 13.06.2023

[web 2016] *Trac WebKit FAQ*. https://trac.webkit.org/wiki/FAQ. Version: 2016

[chr 2021a] *Chrome Platform Status*. https://chromestatus.com/features. Version: 2021. – Letzter Zugriff: 13.06.2023

[chr 2021b] *Chromium Project old-process*. https://www.chromium.org/blink/launching-features/old-process/. Version: 2021. – Letzter Zugriff: 13.06.2023

[chr 2021c] *The Chromium Projects - Launching Features*. https://www.chromium.org/blink/launching-features/. Version: 2021. – Letzter Zugriff: 13.06.2023

[chr 2021d] *Speeding up Chrome's release cycle*. https://blog.chromium.org/2021/03/speeding-up-release-cycle.html. Version: 2021

[saf 2022a] *Announcing WebRTC and Media Capture*. https://webkit.org/blog/7726/announcing-webrtc-and-media-capture/. Version: 2022. – Letzter Zugriff: 13.06.2023

[app 2022] *App Store Review Guidelines*. https://developer.apple.com/app-store/review/guidelines/#software-requirements. Version: 2022. – Letzter Zugriff: 13.06.2023

[moz 2022a] *Bugzilla*. https://bugzilla.mozilla.org/. Version: 2022

[chr 2022a] *Chrome Platform Status: Feature: CSS Anchor Positioning*. https://chromestatus.com/feature/5124922471874560. Version: 2022. – Letzter Zugriff: 13.06.2023

[chr 2022b] *Chromium Version Numbers*. https://www.chromium.org/developers/version-numbers/. Version: 2022

[cus 2022] *Custom Elements v0 support*. https://caniuse.com/?search=Custom%20Elements%20V0. Version: 2022

[ios 2022] *Explore WKWebView additions*. https://developer.apple.com/videos/play/wwdc2021/10032?time=758. Version: 2022. – Letzter Zugriff: 13.06.2023

[jam 2022] *Findings from the Jamstack Community Survey 2022*. https://jamstack.org/survey/2022/. Version: 2022

[ecm 2022] *Finished Proposals*. https://github.com/tc39/proposals/blob/main/finished-proposals.md. Version: 2022

[fir 2022] *The Firefox release process*. https://wiki.mozilla.org/Release_Management/Release_Process. Version: 2022

[fug 2022] *The Fugu Project*. https://www.chromium.org/teams/web-capabilities-fugu/. Version: 2022

[chr 2022c] *Google Chrome-Release-Versionen*. https://support.google.com/chrome/a/answer/9027636?hl=de. Version: 2022. – Letzter Zugriff: 13.06.2023

[wc- 2022] *HTML and JavaScript usage metrics - CustomElementRegistryDefine*. https://chromestatus.com/metrics/feature/timeline/popularity/1689. Version: 2022

[was 2022] *HTML and JavaScript usage metrics - WebAssemblyInstantiation*. https://chromestatus.com/metrics/feature/timeline/popularity/2237. Version: 2022

[int 2022] *Interop 2021 Dashboard*. https://wpt.fyi/interop-2021. Version: 2022. – Letzter Zugriff: 13.06.2023

[jpe 2022] *JPEGXL support*. https://caniuse.com/?search=jpegxl. Version: 2022

[moz 2022b] *Mozilla Wiki - Origin Trials*. https://wiki.mozilla.org/Origin_Trials. Version: 2022

[chr 2022d] *Running an Origin Trial*. https://www.chromium.org/blink/origin-trials/running-an-origin-trial/. Version: 2022. – Letzter Zugriff: 13.06.2023

[saf 2022b] *Safari is the new Internet Explorer*. https://www.safari-is-the-new-ie.com/. Version: 2022. – Letzter Zugriff: 13.06.2023

[wpt 2022] *Web Platform Tests Dashboard*. https://wpt.fyi/. Version: 2022. – Letzter Zugriff: 13.06.2023

[web 2022a] *WebKit Feature Policy*. https://webkit.org/feature-policy/. Version: 2022

[web 2022b] *WebKit GitHub Repository*. https://github.com/WebKit/WebKit. Version: 2022

[app 2023] *Apple-Sicherheitsupdates*. https://support.apple.com/de-de/HT201222. Version: 2023

[pro 2023] *Baseline*. https://web.dev/baseline/. Version: 2023

[cha 2023] *Chart.js*. https://www.chartjs.org/. Version: 2023

[chr 2023] *Chromium iOS Bugs WontFix WebKit*. https://bugs.chromium.org/p/chromium/issues/list?q=OS%3DiOS%20webkit%20status%3DWontFix&can=1. Version: 2023

[mdn 2023a] *Experimental features in Firefox*. https://developer.mozilla.org/en-US/docs/Mozilla/Firefox/Experimental_features. Version: 2023

[fir 2023] *Firefox Release Calendar*. https://wiki.mozilla.org/Release_Management/Calendar. Version: 2023

[doc 2023] *HTML Living Standard*. https://html.spec.whatwg.org/multipage/common-dom-interfaces.html#htmlallcollection. Version: 2023

[int 2023] *Interop 2023 Dashboard*. https://wpt.fyi/interop-2023. Version: 2023. – Letzter Zugriff: 13.06.2023

[ann 2023] *Introducing WebKit Documentation*. https://webkit.org/blog/14139/introducing-webkit-documentation/. Version: 2023

[mdn 2023b] *MDN Browser Compat Data*. https://github.com/mdn/browser-compat-data/. Version: 2023

[npm 2023a] *npm SemVer*. https://www.npmjs.com/package/semver. Version: 2023

[npm 2023b] *SQLite Client for Node.js Apps*. https://www.npmjs.com/package/sqlite. Version: 2023

[6sense 2023] *Market Share of Google AdSense*. https://6sense.com/tech/ad-serving/google-adsense-market-share#:~:text=Google%20AdSense%20has%20market%20share,net%20with%201.98%25%20market%20share. Version: 2023. – Letzter Zugriff: 13.06.2023

[About Statcounter 2023] *About StatCounter*. https://gs.statcounter.com/about. Version: 2023. – Letzter Zugriff: 13.06.2023

[Apple] *Safari 17 Beta Release Notes*. https://developer.apple.com/documentation/safari-release-notes/safari-17-release-notes#Web-apps. – Letzter Zugriff: 13.06.2023

[Beyad u. Tan 2022] Beyad, Ali ; Tan, Victor: *User-Agent Reduction deprecation trial*. https://developer.chrome.com/blog/user-agent-reduction-deprecation-trial/. Version: 2022. – Letzter Zugriff: 13.06.2023

[Bianchi 2023a] Bianchi, Tiago: *Advertising revenue of Google from 2001 to 2022*. https://www.statista.com/statistics/266249/advertising-revenue-of-google/. Version: 2023. – Letzter Zugriff: 13.06.2023

[Bianchi 2023b] Bianchi, Tiago: *Annual research and development expenditure of Alphabet from 2013 to 2022*. https://www.statista.com/statistics/507858/alphabet-google-rd-costs/. Version: 2023. – Letzter Zugriff: 13.06.2023

[Brodkin 2020] Brodkin, Jon: *Mozilla cuts 250 jobs, says Firefox development will be affected*. https://arstechnica.com/information-technology/2020/08/firefox-maker-mozilla-lays-off-250-workers-says-covid-19-lowered-revenue/. Version: 2020

[Bytecode Alliance] *Bytecode Alliance*. https://bytecodealliance.org/. – Letzter Zugriff: 13.06.2023

[Ceci 2022] Ceci, L.: *Worldwide consumer spending on mobile apps and games in Google Play and the Apple App Store as of 3rd quarter 2022*. https://www.statista.com/statistics/183469/app-stores-global-revenues/. Version: 2022. – Letzter Zugriff: 13.06.2023

[Deveria 2017] Deveria, Alexis: *Older versions of Chrome on Android?* https://github.com/Fyrd/caniuse/issues/3518#issuecomment-308931205. Version: 2017

[Deveria 2022] Deveria, Alexis: *CanIUse Browser Comparison*. https://caniuse.com/?compare=chrome+105,edge+105,safari+15.6,firefox+104,opera+90,ie+11,and_chr+104,ios_saf+15.6,samsung+18.0,op_mini+all,op_mob+64,and_uc+12.12,android+104,and_ff+101,and_qq+10.4,baidu+7.12,kaios+2.5&compareCats=all. Version: 2022. – Letzter Zugriff: 13.06.2023

[Deveria 2023] Deveria, Alexis: *CanIUse*. https://caniuse.com/. Version: 2023. – Letzter Zugriff: 13.06.2023

[gerardnll 2019] gerardnll: *Why is Safari the new IE*. https://news.ycombinator.com/item?id=19609028. Version: 2019. – Letzter Zugriff: 13.06.2023

[Höser 2023] Höser, Raphael: *Bachelor Tools*. https://github.com/Snapstromegon/bachelor-tools. Version: 2023. – Letzter Zugriff: 13.06.2023

[Jose u. Mateo 2022] Jose, San ; Mateo, Son: *Mozilla Foundation and Subsidiaries - Independent Auditor's Report and Consolidated Financial Statements*. https://assets.mozilla.net/annualreport/2021/mozilla-fdn-2021-fs-final-1010.pdf. Version: 2022. – Letzter Zugriff: 13.06.2023

[Microsoft 2022] *Verwenden von ActiveX-Steuerelementen für Internet Explorer 11*. https://support.microsoft.com/de-de/windows/verwenden-von-activex-steuerelementen-f%C3%BCr-internet-explorer-11-25738d05-d357-39b4-eb2f-fdd074bbf347. Version: 2022

[Microsoft Edge Chromium 2023] *Microsoft Edge (Chromium)*. https://support.microsoft.com/de-de/microsoft-edge/herunterladen-der-neuen-auf-chrome-basierenden-version-von-microsoft-edge-0f4a3dd7-55df-60f5-739f-00010dba52cf. Version: 2023. – Letzter Zugriff: 13.06.2023

[Mozilla 2023] Mozilla: *Mozilla Specification Positions*. https://mozilla.github.io/standards-positions/. Version: 2023. – Letzter Zugriff: 13.06.2023

[Parlament 2022] Parlament, Europaeisches: *Standpunkt des europaeischen Parlaments Gesetz ueber digitale Maerkte*. https://www.europarl.europa.eu/doceo/document/TC1-COD-2020-0374_DE.pdf. Version: 2022. – Letzter Zugriff: 13.06.2023

[Preston-Werner 2022] Preston-Werner, Tom: *Semantic Versioning 2.0.0*. https://semver.org/lang/de/. Version: 2022

[Russell 2015] Russell, Alex: *Doing Science On The Web*. https://infrequently.org/2015/08/doing-science-on-the-web/. Version: 2015. – Letzter Zugriff: 13.06.2023

[Russell 2022] Russell, Alex: *Progress Delayed Is Progress Denied*. https://infrequently.org/2021/04/progress-delayed/. Version: 2022. – Letzter Zugriff: 13.06.2023

[Scholz u. Deveria 2019] Scholz, Florian ; Deveria, Alexis: *Caniuse and MDN compatibility data collaboration*. https://hacks.mozilla.org/2019/09/caniuse-and-mdn-compat-data-collaboration/. Version: 2019

[Simmons 2022] Simmons, Jen: *Do we really want to live in a 95 percent Chromium browser world*. https://twitter.com/jensimmons/status/1490747578526404608. Version: 2022. – Letzter Zugriff: 13.06.2023

[Slapka 2022] Slapka, Miroslav: *Safari is the new Internet Explorer*. https://medium.com/geekculture/safari-is-the-new-internet-explorer-475ae84d4c1. Version: 2022. – Letzter Zugriff: 13.06.2023

[Statcounter 2023] *Browser Market Share Worldwide*. https://gs.statcounter.com/browser-market-share. Version: 2023. – Letzter Zugriff: 13.06.2023

[Statista Research Department 2023] *Market share comparison of the leading browsers for internet usage in Germany from January 2009 to January 2023*. https://www.statista.com/statistics/461879/browsers-market-share-germany/. Version: 2023. – Letzter Zugriff: 13.06.2023

[WASM CG] *WEBASSEMBLY COMMUNITY GROUP*. https://www.w3.org/community/webassembly/. – Letzter Zugriff: 13.06.2023

[WebKit 2022] WebKit: *WebKit's positions on emerging web specifications*. `https://github.com/WebKit/standards-positions`. Version: 2022. – Letzter Zugriff: 13.06.2023

Eidesstattliche Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben.

Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Wiehl, 16. Juni 2023

Raphael Höser