Appendix F: understanding the role of browser engines

Introduction

1. As discussed in Chapter 5, a browser engine is a key part of a browser, and its role is to transform web page source code into web pages (or web apps) that people can see and engage with on mobile devices.

2. In this appendix, we first discuss the history of browser engines. We then provide additional technical details on two key topics that are discussed in Chapter 5, namely:

   - web compatibility (which is primarily determined by the browser engine); and

   - Apple requiring browsers on iOS to use its WebKit browser engine (the ‘WebKit restriction’).

3. As discussed in Chapter 5, web compatibility is a key barrier to competition in browser engines, while the requirement to use WebKit on iOS limits the extent to which competing browsers can differentiate themselves from and exert a competitive constraint on Apple’s Safari browser and further limits the support for web apps on iOS.

History of browser engines

4. Every web browser requires a browser engine for layout and rendering. Below, we set out the dates at which new browser engines were created and discontinued, and when browsers switched to and away from those browser engines.

Early proprietary browser engines

5. Browser engines surfaced when the first two commercial-like browsers started competing with each other for popularity. Netscape Navigator (Netscape) and Internet Explorer (IE) both featured proprietary browser engines designed for desktop computers.

6. The intense competition between Netscape and Microsoft in the development of their browser engines led to rapid but incompatible implementations of new

technologies, forcing developers to duplicate their efforts and develop content targeting each browser separately.[1]

7. Netscape was the most popular web browser prior to IE's rise, peaking in 1995 at 83% market share.[2] However, from 1995 onwards, Microsoft invested heavily in IE, spending over $100 million each year on development, distributing it for free and bundling it with its Windows operating system.[3] By 2002, Microsoft had achieved over 90% share of supply in browsers.[4]

8. Netscape did not label its original browser engine separately from the browsers in which it was embedded. In 2000, it replaced its original browser engine with a new browser engine called Gecko.

9. Microsoft initially licenced the Spyglass engine for use in IE,[5] but debuted its own browser engine Trident with the release of IE 4 in 1997. Since then, Microsoft has used Trident in all versions of IE for Windows. Trident is a proprietary browser engine. However, Microsoft permits the development of so-called Trident shells, which are essentially expansions of IE with added functionality.[6,7]

Modern browser engines

10. Most modern browser engines are open-source. This means that their code is available at no cost to be reviewed, copied, modified and used. A key advantage of open-source development is that many contributors can submit proposed changes, including features and bug fixes. However, a 'steward' is ultimately in control of which changes are accepted.

11. We discuss below the three main modern browser engines that are under active ongoing development – Gecko, WebKit, and Blink – in the order they were established.

Gecko

12. Netscape announced Gecko as an open-source browser engine project in 1998.[8] However, Netscape Navigator version 6.0, which was released in 2000 using Gecko, failed to outcompete Microsoft's IE, and in 2003 AOL (which had

[1] Web Design in a Nutshell, Niederst Robbins, 2001, page 4.
[2] History of Web Browser Engines 1990-2020 (eylenburg.github.io).
[3] U.S. V. Microsoft: Court's Findings Of Fact (justice.gov), findings of fact 135 and 136.
[4] TheCounter.com: The Full-Featured Web Counter with Graphic Reports and Detailed Information (archive.org).
[5] Memoirs From the Browser Wars (ericsink.com).
[6] Hosting and Reuse (Internet Explorer) | Microsoft Docs.
[7] Examples of Trident shells include AOL Explorer, which is now discontinued, and MSN Explorer.
[8] Levitt, Jason (April 20, 1998). 'Netscape releases the source'. *InformationWeek* (678). pp. 85–90. ISSN 8750-6874.

acquired Netscape) created the non-profit Mozilla Foundation and made it the steward of Gecko.[9]

13. Gecko benefited from open-source contributions from many organisations, including Google, and by 2009, the Gecko-based Firefox browser had over 25% share of browser usage across all devices.[10] However, Gecko's share of browser usage has declined since, and Gecko-based browsers have never been popular on mobile devices; they had a share of browser usage of less than 1% on mobile devices in 2020.[11]

14. In 2016, Mozilla announced Quantum as an effort to build the next-generation browser engine for Firefox users based on Gecko. This involved merging stable portions of Mozilla's experimental browser engine Servo to Gecko to improve Firefox stability and performance.[12] However, the improvement of Gecko using Servo code did not reverse the decline in Gecko's market share.

WebKit

15. Apple created the WebKit browser engine in 2001 by forking the existing KHTML and KJS libraries from the Unix based K Desktop Environment (KDE), an early open-source project.[13] In 2005, Apple released WebKit as open-source software.[14] Apple told us that while it 'has provided guidance, input, and leadership, WebKit is the result of a massive, industry-wide effort. Since its inception, WebKit has had approximately 2,000 individuals who have contributed code, including employees of organisations such as Google, Adobe, Igalia, Samsung, Intel, and Sony'.

16. WebKit was initially adopted by other browsers, including Google Chrome at Chrome's release in 2008. Apple requires that web browsers use WebKit on iOS.[15]

Blink

17. Google created Blink in 2013 by forking WebKit. Its stated reason for doing so was that its Chromium browser project (which is the basis for Google Chrome) 'uses a different multi-process architecture than other WebKit-based

[9] IFIP AICT 404 - Identifying Success Factors for the Mozilla Project (springer.com).
[10] Statcounter.
[11] See shares of supply in Chapter 5.
[12] Quantum - MozillaWiki.
[13] Don Melton on Twitter: "ATTENTION INTERNETS! WebKit is not 10 years old today. That happened on June 25. I know the date because that's when I started the project." / Twitter.
[14] Molkentin, Daniel (June 7, 2005). "Apple Opens WebKit CVS and Bug Database". *KDE News*. Archived from the original on July 15, 2009.
[15] We assess the impacts of this restriction in Chapter 5 of our interim report.

browsers, and supporting multiple architectures over the years has led to increasing complexity for both the WebKit and Chromium projects. This has slowed down the collective pace of innovation'.[16] One commentator at the time noted that Google had been participating more actively than Apple in developing WebKit, by some measures.[17]

18. Like Gecko and WebKit, Blink today benefits from open-source contributions from many organisations. Major contributors include Microsoft, Opera, Facebook, Adobe, Intel, IBM and Samsung.[18] Moreover, Blink is popular with browser vendors. Since Chrome's switch to Blink, many other browsers have followed suit or adopted the engine upon entry to the browser market, including Opera, Yandex, Samsung Internet, Brave, Vivaldi and Edge.[19]

EdgeHTML

19. When it launched its new Edge browser in 2015 – a replacement for IE – Microsoft forked its proprietary Trident browser engine to create EdgeHTML.[20] One of Microsoft's main motivations for forking Trident was to make it easier to ensure web compatibility.[21] However, Microsoft ultimately decided that replacing EdgeHTML with Blink would best ensure web compatibility, and in 2020 it switched to using Blink as its browser engine for Edge.[22]

Presto

20. Opera originally used its own proprietary browser engine, Presto, which debuted in 2003.[23] However, in 2013 Opera switched to using Blink.[24]

Others

21. As discussed above, most browsers are now powered by Google's browser engine Blink and the supply of browser engines is highly concentrated (as discussed in Chapter 5 in the section on shares of supply).

[16] Blink: A rendering engine for the Chromium project | Google Open Source Blog (googleblog.com).
[17] Hypercritical: Code Hard or Go Home.
[18] AUTHORS - chromium/src.git - Git at Google (googlesource.com).
[19] See Opera Confirms It Will Follow Google, Ditch WebKit for Blink (thenextweb.com); Yandex Signs Google Agreement for Submitting Contributions to the Chromium Project Alongside Nvidia, Opera (thenextweb.com); What's The Deal With The Samsung Internet Browser? An Interview With Jungkee Song — Smashing Magazine; How Vivaldi browser is different from Google Chrome; Brave Unveils Development Plans for Upcoming 1.0 Browser Release, Including Transition to Chromium Front-End | Brave Browser.
[20] A break from the past: the birth of Microsoft's new web rendering engine - Microsoft Edge Blog (windows.com).
[21] Building a more interoperable Web with Microsoft Edge - Microsoft Edge Blog (windows.com).
[22] Microsoft Edge: Making the web better through more open source collaboration | Windows Experience Blog.
[23] Dev.Opera — Opera Mini server upgrade.
[24] Dev.Opera — A First Peek at Opera 15 for Computers.

22. There have been a limited number of entrants over the past decade, including:

- The open-source browser engine Goanna, a fork of Gecko which is stewarded by Moonchild Productions. Officially launched in 2015, Goanna powers both of Moonchild Productions' open-source web browsers, PaleMoon and Basilisk.[25] The browser engine has also been adopted by K-Meleon and Mypal.[26,27]

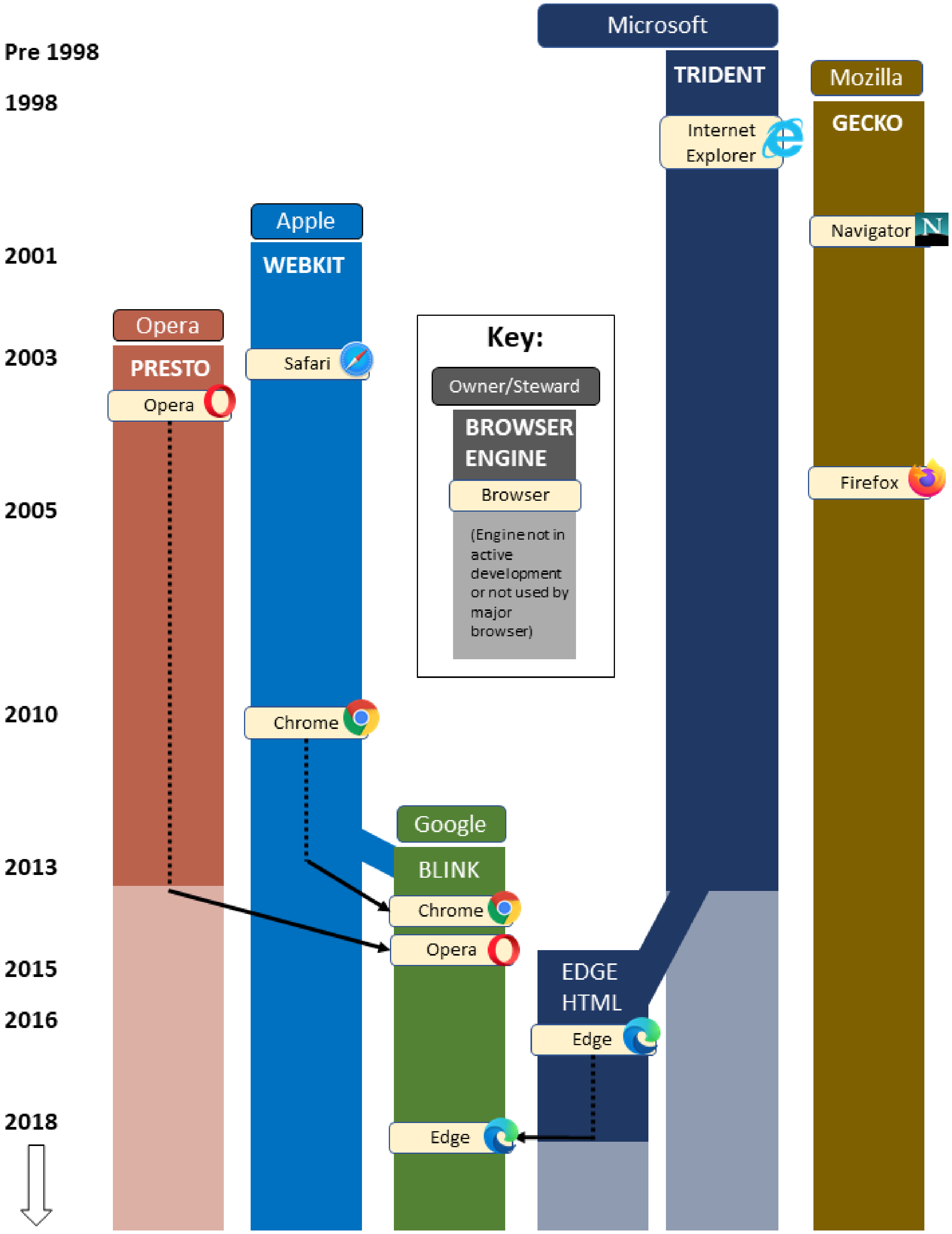
- The proprietary browser engine Flow, launched by UK-based developer Ekioh in its browser (also called Flow) in late 2020.[28] Unlike most browser engines, Flow is not a fork of a previous browser engine codebase. Ekioh is aiming to differentiate Flow from Blink by building a browser for uses where a new browser engine would have clear benefits, such as better performance.[29,30]

23. To date, these browser engines have attracted very limited usage.

24. Figure F.1 sets out the timeline of modern browser engine development, illustrating that WebKit, Blink and Gecko are the only three major browser engines that continue to be under active development.

[25] Pale Moon adopts new Goanna browser engine, fine-tunes interface (betanews.com)
[26] K-Meleon (kmeleonbrowser.org)
[27] Mypal - Official Website (mypal-browser.org)
[28] Unlike many other modern browser engines, Flow is not a fork of an existing codebase.
[29] Flow Browser | The parallel, multithreaded HTML browser (ekioh.com)
[30] Flow: A lightweight browser with a new rendering engine (fastcompany.com)

Figure F.1: Timeline of modern browser engine development



Web compatibility

25. The type of content which can be created by online content providers and accessed by users depends on the capabilities which have been implemented by browser engines. If the browser engine in a user's browser does not have a particular capability, then a user will be unable to properly engage with the relevant web content. For example, if a user's browser engine lacks the ability to process a particular video format, the user will not be able to watch a video using that format which has been uploaded to a web page.

26. In general, online content providers try to ensure that their content is compatible with multiple browser engines so that it reaches as many consumers as possible. However, where browser engines' capabilities differ, online content providers may choose to produce content which is not supported by all browser engines. There is an inherent tension between compatibility and competition between browser engines to provide new capabilities, as any novel capability provided by a browser engine will not be present in other browser engines. A small browser engine which adds a novel capability cannot expect this to be used extensively by online content providers, given the small audience it commands. A capability implemented by a widely used browser engine, on the other hand, is more likely to be adopted. Once a critical mass of online content providers has implemented content using a novel capability, other browser engines may feel pressure to replicate it as their users increasingly encounter web pages which use the new capability (and as a result of the standards-setting work described below).

27. Compatibility is not 'all or nothing', and web developers have adopted a range of strategies to manage compatibility issues. A popular approach is 'progressive enhancement', where web content is structured to be available to as wide a variety of browser engines as possible, but enhanced by newer functionality where a user's browser permits.[31] If a capability is sufficiently important, online content providers can resort to telling users to use a specific browser to access their content.

28. At times, online content providers have found it very difficult to ensure that their content is compatible with multiple browser engines (such as the browser engines in the early versions of Netscape and Internet Explorer). To avoid this, and to maximise compatibility, institutions have been created through

[31] Progressive Enhancement and the Future of Web Design - hesketh.com.

which they can coordinate around novel browser engine capabilities through web standards.

29. Key standards development organisations include:

- **The Internet Engineering Task Force (IETF)**:[32] a loosely self-organised group that contributes to the engineering and evolution of Internet technologies, via producing high quality, relevant technical and engineering documents (such as protocol standards and best current practices documents) that influence the way people design, use, and manage the Internet. It aims to support the evolution of the internet and maintaining the smooth running of the internet as a whole, via developing and maintaining the Request For Comment (RFC) documents that define the open standards by which the internet is managed. These open standards are developed via rough consensus.

- **The World Wide Web Consortium (W3C)**:[33] which develops web standards via its international community of member organisations, a full-time staff, and the public. W3C's primary activity is to develop protocols and guidelines that aim to ensure long-term growth for the web. The W3C adopts a process to get to a 'W3C Recommendation' or 'standard',[34] via workshop, activity proposal and working group, by which specifications and guidelines are reviewed and revised.

30. As a result of work by standards development organisations, there are a series of open standards that should address concerns of web compatibility. However, in practice, compatibility and standards-setting issues remain. In particular:

- Mozilla submitted that dominant players, and in particular Google, can distort markets due to not committing to final standard and/or not respecting the agreed timelines to deploy or deprecate relevant technologies. By way of example, Mozilla told us that in video conferencing services 'this is the implementation of WebRTC in browsers, where premature deployment of a non-standard interface in Chromium resulted in over half a decade of compatibility problems between websites and other browsers'.[35]

[32] IETF | Internet Engineering Task Force.
[33] http://www.w3.org/.
[34] https://www.w3.org/Consortium/Process/.
[35] Mozilla's Response to the Public Consultation on Google's Privacy Sandbox Commitments at Page 7, dated 8 July 2021.

- Apple submitted that with respect to G Suite, which is a suite of web applications created by Google for businesses (now called Google Workspace), users in browsers based on WebKit are unable to access documents offline or voice typing. Apple submitted that it understands that rather than using web standard functionality, G Suite depends on a browser extension that ships with Chrome and is not available on other browsers.

- One browser vendor submitted that website developers do not always develop sites against the standards and instead develop sites using development tools (often provided by browser makers) and test those sites against the web browsers they want to make sure they support. Similarly, an internal document from Microsoft states that 'web developers are explicitly coding first and foremost to Chrome' and that this means that 'in most cases developers are not even consulting web standards or testing on other browsers'.

- One market participant submitted that the strength of Chrome has given Google 'control' over standards bodies, allowing it to push its preferred specifications which must then be implemented by its competitors.

- Although online content providers largely told us that they ensure web compatibility with all major browsers, there are certain exceptions. Out of the 33 online content providers that commented on web compatibility, seven did not list the Firefox mobile browser as a browser for which they ensure compatibility.[36] Additionally, it is unclear which level of support the online content providers considered when answering this question.

31. A group of technical experts told us that they estimated that supporting all browsers costs around 130-160% of the initial implementation cost of a webpage or web app if standards have been followed correctly, compared to a 300-500% cost (in addition to substantial ongoing maintenance) to support multiple platforms for web, Android, iOS and desktop.

Browser engine restriction on iOS

32. As discussed in Chapter 5, Apple requires all browsers on iOS to use WebKit as their browser engine. With respect to this restriction, we below set out: (i) details on the specific security rationale provided by Apple with respect to this

[36] Of these seven, three did not list Firefox at all, while the remaining four only listed the Firefox desktop browser. One respondent submitted that the latest version of Firefox was partially supported.

restriction; and (ii) evidence on the comparison of browser engine performance and feature support.

Specific security claims made by Apple with respect to the WebKit restriction

33. Apple submitted that one of the main reasons for the WebKit restriction is security. In particular, Apple told us the following:

- Modern websites are dynamic applications (in the sense that they can change their behaviour/functionality over time), and websites run a lot of software from third party developers. Modern browsers run Just-In-Time compiling (JIT), such that, with each time the website is compiled, it is slightly different in behaviour. This dynamic behaviour on websites is known as an ‘attack surface’. Through loading webpages users are running software that has the ability to ‘attack’ the iPhone’s security.

- iOS has been programmed to have multiple layers of defence, meaning that in order to successfully attack the iPhone, an attacker would need to chain attacks through multiple layers. The attacker’s end goal is to access the full phone, and ultimately get the phone’s compiler to run the attacker’s own code giving them Remote Code Execution (RCE). One method through which an attacker does this is through the JIT compiling.

- WebKit is the only engine on iOS that can access JIT, thereby allowing Apple to concentrate its security resources and auditing on one target. By mandating the WebKit restriction, Apple can rapidly fix any vulnerability for all apps across the iOS ecosystem.

- There are thousands of non-browser apps using WebKit to render webpages (in-app browsing or some in-app advertisements, for example). If instead apps used a non-WebKit browser engine and it were to require an update, Apple would also have to require thousands of developers to update their own app. This could cause some vulnerabilities to persist for months, if not years.

- WebKit is tightly integrated with device hardware and the iOS operating system to deliver substantial security protections, and third-party browser engines lack important features that Apple harnesses via tight integration between WebKit and iOS device hardware and software.[37]

[37] For example, Apple noted that WebKit benefits from Pointer Authentication Codes engineered into Apple Silicon chips that defeat a major hacking technique and a hardened sandbox profile designed specifically to protect against web-based attacks.

- Apple's own opinion is that WebKit offers a better security level than Blink and Gecko.

34. Apple further submitted that there were privacy reasons for the WebKit restriction as removing the WebKit requirement would allow individual apps to become 'dynamic' and change their behaviour or functionality after app review.

Evidence comparing browser engine feature support and performance

35. We engaged with various stakeholders on test suites that compare WebKit to other browser engines. Several stakeholders proposed measures that assess compatibility and feature support. For example, in response to our request for information, such measures were proposed by Google and several technical experts. Mozilla submitted that while measures that assess feature support may be useful to assess interoperability issues, it listed other factors such as performance and standards compliance as also relevant. Apple submitted that a focus on compatibility or web standards compliance does not adequately reflect attributes of browser engine quality, including quality, performance, stability and privacy, and Apple explained that, in the normal course of business, it tests web app responsiveness, JavaScript performance and graphics performance.

36. On the basis of these submissions, we consider that while a variety of measures are likely to be relevant, compatibility and feature support appear to be particularly important. We have therefore focused on these measures, although we have also considered other measures.

Compatibility and feature support

37. In the below, we present the test suites that were recommended to us and further discuss stakeholders' views on them.

Recommended test suites

38. In response to our request for information, Google as well as several technical experts endorsed a test suite measuring compatibility and feature support called the Web Platform Test (WPT) Dashboard, also referred to as wpt.fyi.[38,39] This project runs tests for various browser technologies and, on

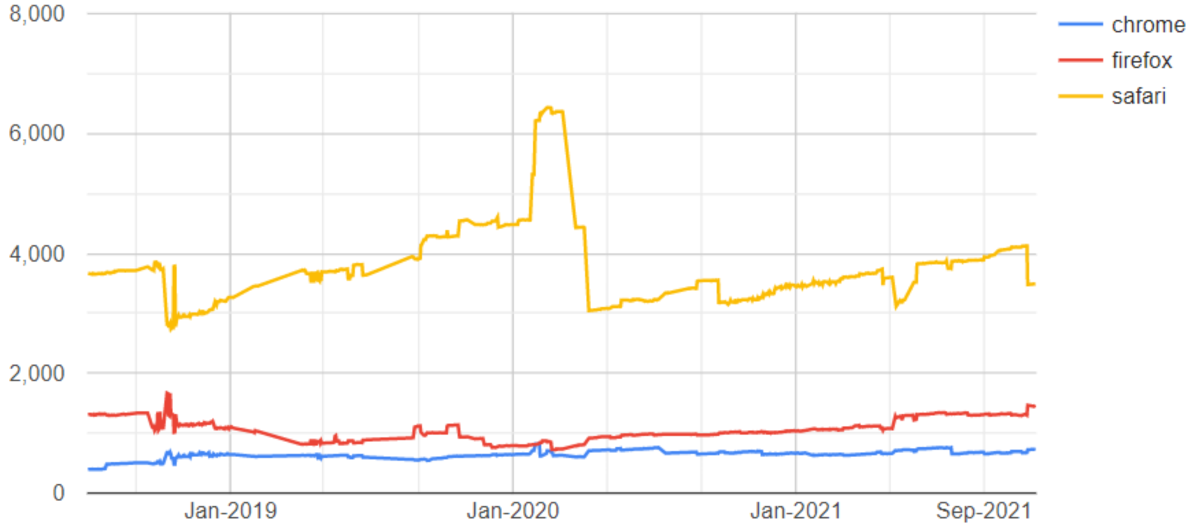
[38] See web-platform-tests dashboard (wpt.fyi)

[39] The Web Platform Test Project is also discussed in blog posts by Alex Russell (Progress Delayed Is Progress Denied - Infrequently Noted) and Tim Perry (Safari isn't protecting the web, it's killing it | HTTP Toolkit). Mozilla submitted that the Web Platform Test Project is useful to gauge interoperability issues, but that it looks at just one facet of how browser engines operate and implement certain web standards.

the basis of the test results, provides assessments of compatibility and feature support of different browsers.

[40] This can broadly be interpreted as instances where the browser is not compatible while the other browsers are. The yellow Safari line (which represents any browser built on WebKit) is substantially and persistently higher than the blue Chrome and red Firefox lines (representing browsers built on Blink and Gecko respectively). This indicates that WebKit has performed significantly worse in terms of compatibility than Blink and Gecko over this period.

Figure F.2: Number of tests which fail in exactly one browser



Source: web-platform-tests dashboard (wpt.fyi)
Note: Graph shows test results based on stable (rather than experimental) version. Graph retrieved 8 November 2021.

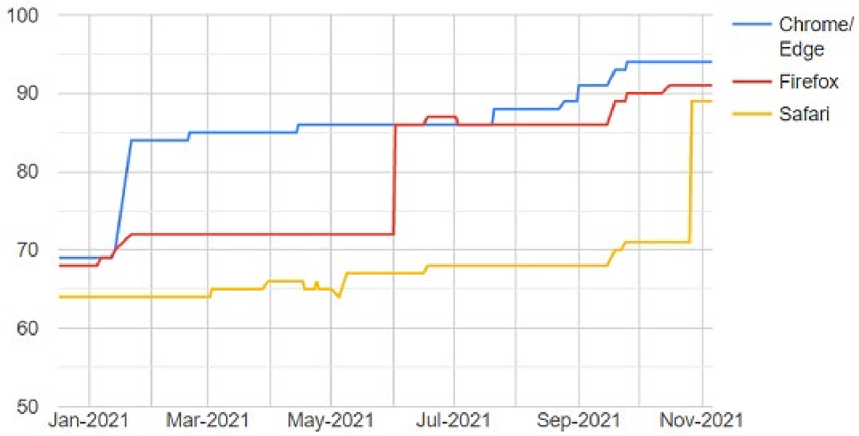
40. Another assessment provided by wpt.fyi is presented in Figure F.3. This assessment focuses on the 2021 Compat Focus Areas, which are five key areas that represent the most painful compatibility bugs (ie a small subset of the features considered in Figure F.2 above). The scores represent how well browser engines are doing on the 2021 Compat Focus Areas (a higher score being better).

41. The yellow Safari line (which represents any browsers built on WebKit) shows that Safari has had a significantly worse compatibility score for most of 2021 than browsers built on Blink or Gecko. We note that the score for Safari has recently improved significantly, following from Apple releasing Safari 15 as

[40] For example, a number of 4,000 for Safari means that there are 4,000 tests which Safari fails but that all other browsers that were considered pass.

part of its iOS 15 release (the compatibility score prior to the jump was based on versions of Safari 14).

Figure F.3: Compat 2021 score

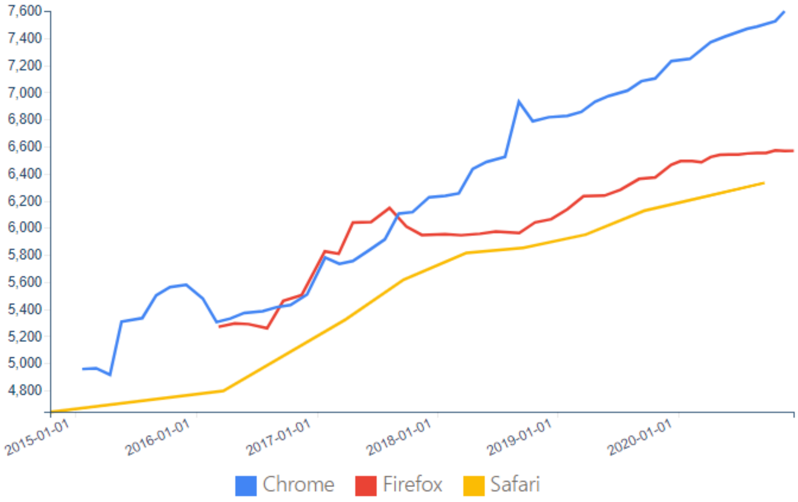


Source: web-platform-tests dashboard (wpt.fyi)
Note: Graph shows test results based on stable (rather than experimental) version. Graph retrieved 8 November 2021.

42. There are several other tests measuring compatibility and feature support to which certain stakeholders referred us, which are set out below.

[41] It shows that the count of APIs available from JavaScript on Safari is lower than on Chrome and Firefox.[42]

Figure F.4: Count of APIs available from JavaScript

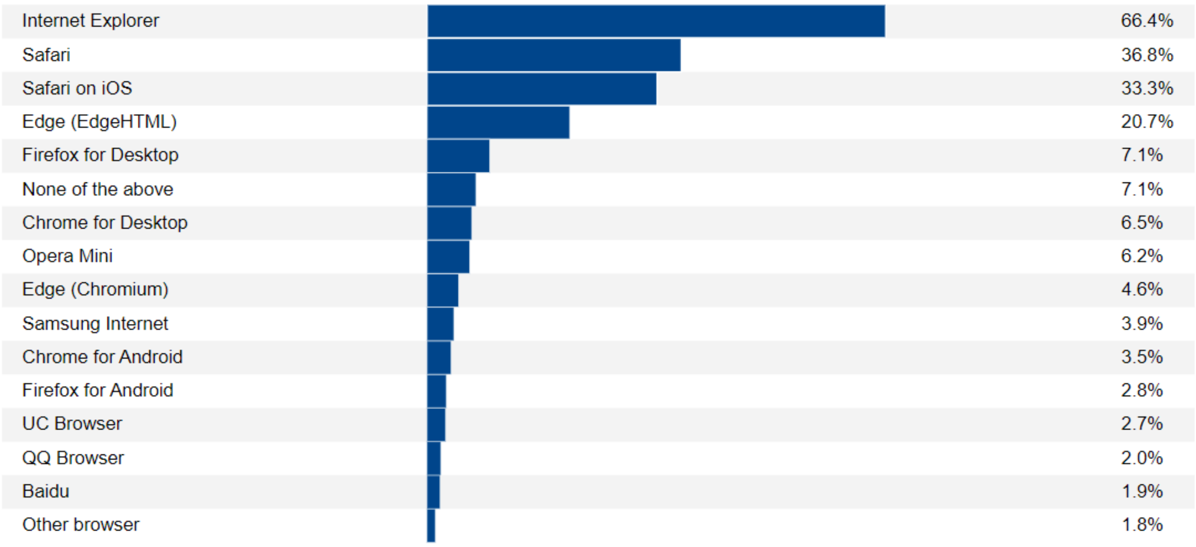


Source: Progress Delayed Is Progress Denied - Infrequently Noted

[41] The data was aggregated by Alex Russel, a Microsoft Partner Program Manager on the Edge team who writes a personal blog called https://infrequently.org/. For disaggregated data, see Web API Confluence Dashboard (web-confluence.appspot.com).

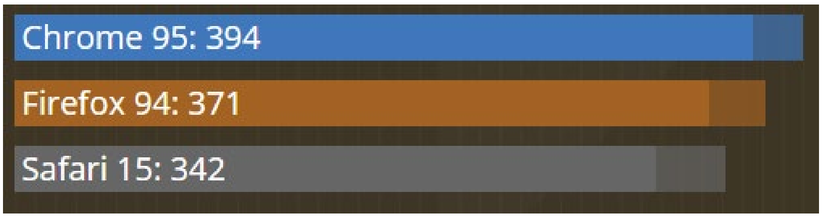
[42] We understand that, again, Safari should be interpreted as any browser built on WebKit while Chrome and Firefox should be interpreted as any browser build on Blink or Gecko, respectively.

Figure F.5: MDN Web Developer Needs Assessment



Source: MDN Web Developer Needs Assessment 2020 - MDN (mozilla.org)

Figure F.6: Can I use browser scores

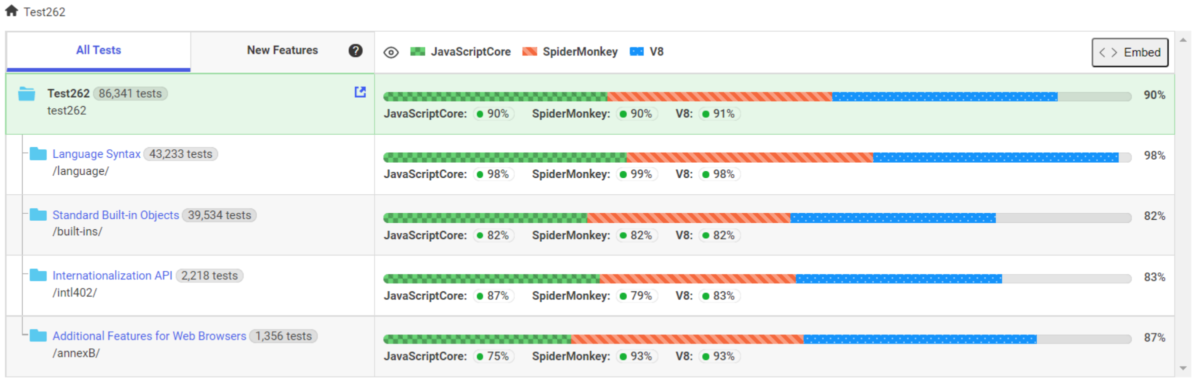


Source: Can I use... Support tables for HTML5, CSS3, etc
Note: Graph shows results for current version. Graph retrieved 8 November 2021.

46. Mozilla submitted that it recommends Test262, which tests standards compliance of different Java Script engines as an example of a way to test browser engines that goes beyond the Web Platform project (Java Script

engines, as discussed in Chapter 5, form part of the browser engine).[43] A group of technical experts told us that this test suite is more limited in use for web developers than wpt.fyi or caniuse because it focuses on only one of the web's four major languages. Figure F.7 below shows that Safari's Java Script engine (called JavaScriptCore) performs similarly to those of Blink (V8) and Gecko (SpiderMonkey) overall, but significantly worse with respect to additional features for web browsers.

Figure F.7: Test 262



Test262 86,341 tests test262	JavaScriptCore: ● 90% SpiderMonkey: ● 90% V8: ● 91%	90%
Language Syntax 43,233 tests /language/	JavaScriptCore: ● 98% SpiderMonkey: ● 99% V8: ● 98%	98%
Standard Built-in Objects 39,534 tests /built-ins/	JavaScriptCore: ● 82% SpiderMonkey: ● 82% V8: ● 82%	82%
Internationalization API 2,218 tests /intl402/	JavaScriptCore: ● 87% SpiderMonkey: ● 79% V8: ● 83%	83%
Additional Features for Web Browsers 1,356 tests /annexB/	JavaScriptCore: ● 75% SpiderMonkey: ● 93% V8: ● 93%	87%

Source: https://test262.report/?engines=javascriptcore%2Cspidermonkey%2Cv8
Note: Graph retrieved 8 November 2021

Stakeholders' views on the test suites

47. With respect to tests in general, Mozilla submitted that it is generally hard to judge the real-world performance of different browser engines from generalised tests, and that the comparisons such tests lead to are 'often subjective, blunt and lack nuance that is meaningful to end users' both when it comes to performance and also web compatibility and interoperability.

48. With respect to the test suites discussed above, and in particular the wpt.fyi assessment on the number of tests which fail in exactly one browser (Figure F.2), Apple submitted the following main criticisms:[44]

- First, Apple submitted that several of the tests discussed above focus on metrics based on the total number of tests run, rather than the importance of those tests. Apple submitted that it therefore does not believe that these metrics are reflective of a browser engine's quality or impact the vast majority of web developers.

[43] Test262 Report – About this project
[44] Apple also submitted that wpt.fyi is a third-party group of test suites that are contributed to by various people, including browser vendors.

- Second, Apple submitted that many tests run by wpt.fyi turn out to be mistaken, or to support single-browser non-standard features, which Apple does not believe is helpful. Apple also noted that, since tests are contributed by third parties, the test quality is inconsistent, and vulnerable to gaming by browser vendors.

- Third, Apple submitted that wpt.fyi is configured in a way that distorts the actual performance of Safari. In this regard, Apple submitted that (i) wpt.fyi compares an old beta version of Safari with newer versions of other browsers,[45] (ii) wpt.fyi runs tests in private browsing mode (which leads to many of the compatibility failures that are being reported) and (iii) a substantial portion of the reported test failures can be attributed to Safari's lack of support for the web specification 'SharedWorkers', for which Apple removed support due to its low adoption.

49. With respect to Apple's first point, Google told us that it considers measures on overall API support (such as the graph on the number of tests which fail in exactly one browser in presented in Figure F.2 and the graph on the count of APIs available from JavaScript presented in Figure F.4) are simple and objective measures. Google also submitted that it considers these tests to be quite comprehensive and cover a wide range of APIs.

50. We also note that one of the assessments (Figure F.3) focuses on the five key areas that represent the most painful compatibility bugs. Even on the basis of this narrower set of features, Safari was performing significantly worse than Blink and Gecko based browsers until very recently and still has a slightly lower score.

51. With respect to Apple's second and third point, Google told us that the tests are developed with and shared with the community. Google also told us that it considers the comparison to be fair and conducted on equivalent terms. In particular, it told us that it does not think that the test results are reliant on enabling a special mode and that it would expect that the graphs would, independent of which mode is selected, show a similar pattern.

52. We also note that, as set out above, the wpt.fyi assessments were endorsed by several stakeholders, which calls into question the quality concerns raised by Apple.

53. Overall, we acknowledge that there are certain limitations to the test suites discussed above. With respect to the wpt.fyi assessment on the number of

[45] In particular, Apple submitted that wpt.fyi uses Safari Technical Preview (STP), a months-old beta version of potential future iterations of Safari for a technical audience.

tests which fail in exactly one browser, we consider that it is a meaningful indicator of the relative feature support of WebKit compared to other browser engines. We also note that a number of other test suites show similar patterns with respect to WebKit's feature support.

Other measures

54. Apple submitted that it uses test providers to test web app responsiveness, JavaScript performance and graphics performance. Apple further submitted that, on these tests, WebKit-based Safari on iOS outperforms competing browsers based on competing engines on Android devices. However, Apple only provided us with the test results for desktop browsers, rather than for browsers on mobile operating systems.

55. Apple further submitted that it believes that power efficiency (particularly in mobile applications), page load speed and memory use are good metrics to compare browser engine performance. While Apple submitted that there are no third-party test suites that measure these metrics across browsers, it submitted internal tests showing that WebKit-based browsers perform better on page loading and power efficiency than competing browsers based on other browser engines.