Open
Web Advocacy

OWA - Mobile Browsers and Cloud Gaming - Response to Working Papers 1-6

VERSION 1.0

Open Web Advocacy
contactus@open-web-advocacy.org

1. Table of Contents

2. Introduction

We would like to thank the Market Investigation Reference (MIR) team for investigating these critical topics and for the thoughtful analysis that they have published. Both browser and Web App competition is currently being undermined on iOS and Android to the detriment of UK consumers and businesses. This MIR has correctly identified and explored many of these issues and has the opportunity to fix them.

In order of importance we believe the following remedies are justified and necessary to allow browsers and Web Apps to compete fairly on all mobile operating systems. We request that MIR team consider each for inclusion in the final working paper:

1. Remove Apple's ban on third-party browser engines on iOS.
2. Obligate Apple to implement install prompts for Safari/WKWebView based browsers on iOS.
3. Prevent Apple from implementing an effective ban on third party browser engines via unreasonable contracts.
4. Allow browsers access to software and hardware APIs on iOS.
5. Allow browsers to install and manage Web Apps on iOS.
6. Change SFSafariViewController to load the user's chosen default browser.
7. Obligate Google to share WebAPK minting on Android.
8. Obligate in-app browsers on iOS and Android to respect the users choice of default browser.
9. Allow browsers with an entitlement to be installed directly on iOS and Android.
10. Make being set as the default browser should grant the hotseat.
11. Choice Screens for browsers
12. Don't pre-install Safari and Chrome on iOS and Android on new devices, instead users choose via choice screen in setup.
13. Allow Safari and Chrome to be uninstalled on iOS and Android.

3. Apple is preventing Web Apps from Contesting their App Store

In 2011, Philip Schiller internally sent an email to Eddie Cue to discuss the threat of HTML5 to the Apple App Store titled “HTML5 poses a threat to both Flash and the App Store”.

> *“Food for thought: **Do we think our 30/70% split will last forever?** While I am a staunch supporter of the 30/70% split and keeping it simple and consistent across our stores, I don’t think 30/70 will last unchanged forever. **I think someday we will see a challenge** from another platform or **a web based solution** to want to adjust our model”*
>
> Internal Apple Emails
> (emphasis added)

That is, as early as 2011, Apple's management viewed Web Apps as a credible threat to the App Store revenue model. This is perhaps unsurprising as Steve Jobs originally intended Web Apps to be the only way to deliver third party apps on iOS.

Apple, again and again claims that Web Apps are the alternative distribution method for Apps in their mobile ecosystem.

> *“QUESTION: Apple is the sole decision maker as to whether an app is made available to app users though the Apple Store, isn't that correct?*
> *REPLY: If it’s a Native App, yes sir, if it’s a Web App no.”*
>
> Tim Cook - Speaking to US Congress

> *“Web browsers are used not only as a distribution portal, but also as platforms themselves, hosting “progressive web applications” (PWAs) that eliminate the need to download a developer’s app through the App Store (or other means) at all.”*
>
> Apple’s Lawyers - Court Filing in Australia

> *“For everything else there is always the open Internet. If the App Store model and guidelines are not best for your app or business idea that’s okay, we provide Safari for a great web experience too.”*
>
> Apple’s App Store Guidelines

Despite this Apple is undermining Web Apps from contesting their app store in a number of critical ways.

First, Apple has effectively banned third party browsers from competing.

Second, Apple does not invest sufficiently in their own browser leading to significant bugs, particularly for Web Apps that are attempting to provide a native app look and feel.

Third, Apple's browser is missing a number of key features. The most primary one being install prompts. That is, Apple has obscured the method of installing Web Apps to the point that most consumers do not realize they exist on iOS. They have consistently refused to implement functionality that would allow web developers to let consumers easily install their Web Apps via Safari while doing everything they can to make it easy to find and install native apps via Safari.

In mobile ecosystems study the CMA listed two motives for this, protecting Apple's app store from the threat of viable Web Apps and protecting Apple's Google search engine deal from third party browsers that can actually compete with Safari on iOS.

> ***Apple receives significant revenue from Google by setting Google Search as the default search engine on Safari****, and therefore benefits financially from high usage of Safari. [...]* ***The WebKit restriction may help to entrench this position*** *by limiting the scope for other browsers on iOS to differentiate themselves from Safari [...] As a result, it is less likely that users will choose other browsers over Safari, which in turn* ***secures Apple's revenues from Google****.*
> *[...]*
> *Apple generates revenue through its App Store, both by charging developers for access to the App Store and by taking a commission for payments made via Apple IAP. Apple therefore benefits from higher usage of native apps on iOS. By requiring all browsers on iOS to use the WebKit browser engine,* ***Apple is able to exert control over the maximum functionality of all browsers on iOS*** *and, as a consequence, hold up the development and use of web apps. This limits the* ***competitive constraint that web apps pose on native apps****, which in turn protects and benefits Apple's App Store revenues.*
>
> UK CMA - Interim Report into Mobile Ecosystems
> (emphasis added)

These two revenue streams are vast, even for a company of Apple's size. Apple collected $85 billion USD in App Store fees in 2022, of which it keeps approximately 30%. Apple reportedly receives $18-20 billion USD a year from their Google Search engine deal, accounting for 14-16 percent of Apple's annual operating profits.

A third and interesting incentive the CMA does not cite, but which the US's Department of Justice does, is that this behavior greatly weakens the interoperability of Apple's devices, making it harder for consumers to switch or multi-home. It also greatly raises the barriers of entry for new mobile operating system entrants by depriving them of a library of interoperable apps.

> *"**Apple has long understood how middleware can help promote competition** and its myriad benefits, including increased innovation and output, **by increasing scale and interoperability**.*
> *[...]*
> *In the context of smartphones, examples of **middleware include internet browsers**, internet or cloud-based apps, super apps, and smartwatches, among other products and services.*
> *[...]*
> ***Apple has limited the capabilities of third-party iOS web browsers, including by requiring that they use Apple's browser engine, WebKit.***
> *[...]*
> *Apple has sole discretion to review and approve all apps and app updates. **Apple selectively exercises that discretion to its own benefit**, deviating from or changing its guidelines when it suits Apple's interests and allowing Apple executives to control app reviews and decide whether to approve individual apps or updates. Apple often enforces its App Store rules arbitrarily. **And it frequently uses App Store rules and restrictions to penalize and restrict developers that take advantage of technologies that threaten to disrupt, disintermediate, compete with, or erode Apple's monopoly power.**"*
>
> DOJ Complaint against Apple
> (emphasis added)

Interoperability via middleware would reduce lock-in for Apple's devices. Lock-in is a clear reason for Apple to block interoperability, as can be seen in this email exchange where Apple executives dismiss the idea of bringing iMessage to Android.

> *"The #1 most difficult [reason] to leave the Apple universe app is iMessage ... iMessage amounts to serious lock-in"*
>
> Unnamed Apple Employee

> *"iMessage on Android would simply serve to remove [an] obstacle to iPhone families giving their kids Android phones ... moving iMessage to Android will hurt us more than help us, this email illustrates why."*
>
> Craig Federighi - Apple's Senior Vice President of Software Engineering

Put together this leads to a situation where not only is Apple not incentivized to provide the features and stability that Web Apps need to contest their App Store but that they are actively incentivized to prevent this from occurring. Worse, due to their control of the operating system they can prevent any third party browser vendor from doing a better job.

This leads to a situation where without regulatory intervention, Web Apps will never be viable on iOS. This greatly dampens their appeal via interoperability, particularly as iOS is often a key, if not the primary target, for businesses due to the wealth of its users.

3.1. Effectively No Browser Competition on iOS

> *"Businesses that face effective competition dare not raise prices, or cut down on quality standards, for fear of losing customers to their competitors (and so losing money)"*
>
> Dr Michael Grenfell

There is effectively no browser competition on iOS. This is due to the fact that Apple has banned third party browsers from choosing or modifying their own engines.

Apple has done this by rule 2.5.6 of their app store guidelines:

> *"Apps that browse the web must use the appropriate WebKit framework and WebKit Javascript."*

This very innocuous rule in fact forces the entire content area of a browser and the majority of functionality important to Web Apps to be exclusively controlled by Apple. This essentially makes all browsers on iOS reskinned versions of Safari. What Apple is saying with this clause is, *"Throw away the browser you have spent 100,000+ hours working on, and build a 'thin' user interface shell around Safari which we have total control over and that you can't modify".*

> *"**Apple has a browser monopoly on iOS**, which is something Microsoft was never able to achieve with IE"*
>
> Scott Gilbertson - The Register
> (emphasis added)

> *"because **WebKit has literally zero competition on iOS, because Apple doesn't allow competition**, the incentive to make Safari better is much lighter than it could (should) be."*
>
> Chris Coyier - CSS Tricks
> (emphasis added)

> *"Today, you can download what look like "alternative" browsers on iOS, like Chrome and Firefox, **but these browsers are mostly just skins overtop of Apple's Safari engine**. iOS app developers aren't actually allowed to include their own browser engines, so everything uses Safari's WebKit engine, with a new UI and settings and sync features layered on top."*
>
> Ron Amadeo - ArsTechnica
> (emphasis added)

While this does allow some minor features to be implemented such as bookmarks and sync, browsers are unable to compete in speed, stability, functionality, security or privacy in almost all aspects. Consumers are mostly unaware of this and thus Apple has maintained this mirage of competition for over a decade.

The WP1 report helpfully separates this out with this definition:

> *"There are two main elements required for mobile browsing:*
> *(a) a browser engine, which renders websites (or web apps – applications which run in web browsers) that users can see and engage with; and*
>
> *(b) a branded user interface (UI), which is responsible for user-facing functionality (a browser).*
>
> *A mobile browser engine is the core software component of a mobile browser that handles the rendering and display of web content. The browser engine is responsible for processing HTML, CSS, and JavaScript code, and rendering websites into the visual format that users see on their mobile devices. **In practical terms, this means the main reason that websites may look, load and work differently in different browsers is their browser engines.**"*
>
> Browsers and Cloud MIR WP1
> (emphasis added)

This is critically important as both developers and consumers have no recourse if Safari is missing a feature, has a severe bug or security vulnerability. Consumers can not switch to a better browser as all other browsers are forced to use Safari's core, the WebKit WKWebView.

This lack of ability to switch means that there is no real risk of Safari losing users due to bugs, missing features or security issues. As such Apple faces no "effective competition" to improve, beyond the bare minimum not to cause embarrassment. This removes all incentive to heavily invest in Safari to improve features and stability.

Some commentators point to Android as a source of browser competition. The argument being, Android has no such ban on third party browser engines, and as such if a consumer were unhappy with the fact that Apple has effectively banned their favorite browser, they could switch to Android where the browser would be available.

This unfortunately does not work. Mobile ecosystems have tremendous lock-in, consumers have low awareness of the mechanics of how browsers work and even fewer consumers are aware that Apple has effectively banned third party browsers.

This was highlighted in the consumer research done by Verian:

> *"These findings are consistent with qualitative consumer research conducted by Verian for this market investigation, which noted the topic of browsers on users' smartphones was a low salience topic that had rarely been considered, if noticed at all, by respondents. This supports the view that the availability of specific browsers and browser engines on a mobile device and their quality likely plays a limited role in users' decisions to purchase mobile devices (and in driving competition between mobile ecosystems)."*
>
> Browsers and Cloud MIR WP1

The WP1 report correctly assesses this, and states that *"On this basis, it is our emerging thinking that the supply of mobile browsers on iOS and Android should be considered as two separate product markets."*. OWA agrees with this assessment.

3.2. Bugs

The CMA's Mobile Ecosystems Market Study states that one of Apple's motives for the WebKit restriction is to undermine the ability of Web Apps to contest its app store.

> *Apple generates revenue through its App Store, both by charging developers for access to the App Store and by taking a commission for payments made via Apple IAP. Apple therefore benefits from higher usage of native apps on iOS. By requiring all browsers on iOS to use the WebKit browser engine,* ***Apple is able to exert control over the maximum functionality of all browsers on iOS*** *and, as a consequence, hold up the development and use of web apps. This limits the* ***competitive constraint that web apps pose on native apps****, which in turn protects and benefits Apple's App Store revenues.*
>
> UK CMA - Interim Report into Mobile Ecosystems
> (emphasis added)

Many of the bugs in Safari are specific to iOS Safari and occur when attempting to develop native-app-like user interfaces and functionality. This goes some way to explaining why many developers working on standard/ordinary websites referred to as Nine-to-fivers' in Jigsaws report did not report significant issues with iOS as they simply wish to make their websites work on all major browsers.

Additionally many of these bugs are encountered most directly by the developers of frameworks and libraries of widgets. Many nine-to-five developers may be unaware that these bugs' existence lead to performance issues or to particular types of widgets not being available. As discussed in Jigsaw's report many developers rely on pre-built components and frameworks, making this issue significantly more hidden for your typical frontend developer.

Essentially we can think of there being two categories of web developers (although there would be some overlap), those trying to develop traditional websites and those trying to deliver software via the Web that could contest Apple's app store.

With regards to consumers, bugs are largely invisible. Developers of Web Apps spend 100s of hours working around specific issues and this cost and time spent is undetectable by end consumers. In the event bugs make it though, we suspect consumers would be significantly more likely to blame the Web App than the browser powering it. Developers can not even suggest that consumers use a different browser which has fixed critical bugs for a better experience, as all other browsers on iOS are forced to use the same WkWebView which contains the same identical bugs.

It is also worth noting that the Jigsaw Survey is unlikely to pick up companies that have exited or chosen not to enter the Web App market, either due to missing features or bugs.

The key questions that arise are:

1. Can third party browser vendors provide a browser with less bugs for Web App developers on iOS?
2. Is there evidence that significant bugs on iOS Safari make it difficult for Web App developers to provide a stable service.
3. Does this reduce Apple's incentive to heavily invest in Safari to fix these bugs?
4. Does this reduce companies' ability to contest the Apple's app store via Web Apps?
5. Can you regulate to make a browser less buggy?

3.2.1. Can Third-Party Browser Vendors Contest Stability for Web Apps on iOS?

The simple answer to this question is, no.

The report correctly states that the majority of what makes one browser different from another browser is contained within its engine. We agree wholeheartedly with this statement.

If the WkWebView contains a critical security bug, third-party browser vendors have no power to fix it, or more importantly avoid having it in the first place. If the WkWebView breaks a feature used by websites or Web Apps, third-party browser vendors can not patch it from the outside.

In essence all browser competition is cut off at the root. Not only are browser vendors forced to build and maintain a new UI around this webview from scratch, they have very limited options for improving it, having fewer bugs or fixing existing bugs.

Few browser vendors are going to educate consumers on this matter, as they would have to declare that their own WkWebView browser for iOS is subpar relative to their products on other platforms or offers little beyond what Safari offers to consumers. This is not an attractive proposition to browser vendors that are already struggling to differentiate their browsers on iOS.

3.2.2. Evidence of Bugs in iOS Safari for Web Apps

Web Apps provide exceptional advantages over Native Apps (write once, instant installation, no gatekeeper tax) but it is not currently possible to build Web Apps that have equivalent functionality to native apps. This is specifically because of Apple's underinvestment in Safari and their lack of motivation to invest in Web Apps and refusal or slow-walk to fix critical bugs.

These bugs are so extensive and numerous we will only be covering a subset in this response, but we do cover it extensively in our Bringing Competition to Walled Gardens report.

3.2.2.1. Scroll Bugs

The most major set of bugs are related to body scroll (preventing the user from scrolling).

This functionality is absolutely critical for many types of user interfaces used in native apps such as when:

- Displaying any kind of overlay such as a popup, centered dialog, bottom sheet, sidebar navigation etc
- When the user needs to scroll within individual containers and the app has multiple scroll areas
- When implementing any kind of advanced animation that should not trigger body scroll while they are occurring
- While implementing any high-fidelity drag and drop interfaces (such as the ones Apple implement themselves throughout their native UIs)
- When implementing anything that requires its own scroll and zoom functionality (e.g. a map component or a diagram visualizer).

OWA organized a group of 7 experienced front-end software engineers to come together to comprehensively report these bugs to Apple. We sent emails, and wrote an extensive ticket covering the issue and additionally tried to get included in Apple's yearly interop effort. Despite all this, since posting this over 2 years ago (developers have been complaining about these issues for over 10 years), these issues have largely been ignored by Apple and remain unresolved ensuring that many types of native like user interfaces can not be reliably built using the web on iOS.

You can see tickets 240860, 237961 and 240859 have not been altered for years.

3.2.2.2. Gesture Based Animations Bugs

requestAnimationFrame, a tool to update styles and positions of elements is capped at 60 frames per second by design and 30 frames per second when the phone is in low-power mode.

iPhone Pros and iPad Pros have screens running at 120 frames per second. Therefore Apple has deliberately ensured that web animations cannot be as smooth as native animations as they are updated only a quarter or half as many frames per a second.

This is perceived by the user as a slight judder or lack of silky smoothness. The net result is web animations feel less polished and thus inferior to end users. As the cause of this is embedded in the **only** browser engine on iOS, it is not resolvable by developers.

Next, WebKit does not communicate the position of the finger on the screen in sub-pixels, but on full pixels only. This can make touch-based animations on the web less fluid and less precise than in native apps.

This lack of accuracy (in combination with other issues) is perceptible by end users, again making web animations feeling less responsive and polished in all browsers on iOS, providing Apple's native apps from which they derive significant profit a key advantage.

3.2.2.3. On Screen Keyboard Bugs

When the on-screen keyboard gets displayed, the entire view is shifted out of the screen by default, so that the focused input remains visible. This is desirable in some cases, but in most cases it needs to be avoided when building proper apps, because otherwise a big part of the interface is being moved out of the screen, for example a top bar that is supposed to remain visible at all times.

There is a feature to avoid this behavior however it is currently broken whenever the field receiving focus is related to user credentials (i.e. username, email, password etc). This causes these sorts of UIs to break in all browsers on iOS.

In addition, it is essential to know the dimensions (i.e width and height of the area containing the web page) when the on-screen keyboard gets displayed to implement many types of interfaces. When the on-screen keyboard is displayed the screen displaying web content shrinks, and it is essential for the developer to know the exact dimensions of this area so they can adjust the user interface accordingly.

A great example of an interface that needs this type of information are toolbars positioned right above the keyboard (such as an auto-suggest, generated content, corrections or emoticons/stickers).

However in WebKit this information is, in certain situations, communicated late, incorrectly or not at all. This leads to the user interface elements being placed in the wrong position which looks hopelessly buggy and non-professional to end users.

As there is no way for developers to fix this (in all browsers on iOS due to Apple's restrictions), these types of interfaces must simply be abandoned. Due to the complexity of maintaining multiple versions of a UI, this will often lead to them getting abandoned in other browsers on other operating systems (such as Android).

3.2.3. Does this Reduce Apple's Incentive to Invest in Safari to Fix Bugs?

The number of competitors of a market is a key determinant of how competitive that market is. In the report iOS is judged a single separate market for browsers.

We would argue that Apple has, via their engine ban, managed to prevent rival browser vendors from effectively being in the market for browsers on iOS via their browser engine ban which removes almost all ability to differentiate their browsers stability and security.

This combined with being the default browser, difficulty in changing the default browser and a general lack of awareness by the public leads to a situation where Safari faces no effective competition on iOS.

A powerful incentive to fix bugs is fear that developers or users will switch to rival browsers. While developers have powerful incentives to support even buggy browsers that have significant market share over time, less stable browsers lose users in competitive markets.

This fear is absent for iOS Safari.

A good example bug is this one where background audio was broken in Safari (and by extension all browsers on iOS) for over 3 years, essentially breaking any music Web App. This bug was only fixed when it started being cited in US antitrust proceedings.

It seems unimaginable that in the hypothetical situation that background audio broke for their native apps such as Spotify, that Apple would leave it broken for 3 years. Had Safari faced significant competition from third-party browser vendors, fear of losing market share via Web Apps suggesting users install different browsers would cause these sorts of bugs to be fixed much faster.

3.2.4. Does this Reduce Companies' Ability to Contest the Apple's App Store via Web Apps?

Stability is of key concern to companies developing mobile apps. App breaking bugs which remain unfixed for considerable periods of time represent a severe threat.

In order to decide to develop a Web App over a native app, companies need to be confident that their products will work and continue to work. Imagine trying to convince a Spotify competitor to release their music app as a Web App. A bug such as that mentioned in the previous section would be a powerful argument to instead simply develop a native app and agree to share 30% of your revenue with the operating system gatekeeper.

3.2.5. Can you Regulate to make a Browser less Buggy?

Directly regulating levels of bugginess in browsers would be a nightmare for any regulator. The topic by its very nature is highly technical and difficult to measure. All browsers have bugs and all browser vendors must decide where to devote resources. This means certain bugs may remain unfixed in particular browsers for long periods of time or indefinitely.

This makes it exceptionally difficult to enforce a minimum standard.

What is needed is effective competition. Remove the barriers to competition and browser vendors will be incentivized to provide the most stable product, or risk over the long term losing market share.

Currently the market for browsers on iOS is not functioning effectively due to Apple's prohibition on third party browser engines. Browser vendors on iOS are not able to provide a more stable service than Safari as these bugs are almost always contained within the engine that they are forced to use and which they can not modify.

The solution is not to focus on directly regulating bugs but rather make sure that for the market of browsers on iOS that there is healthy and effective competition between them to compete to be the most stable, have the least bugs and to fix bugs the fastest.

3.3. Key Missing Features

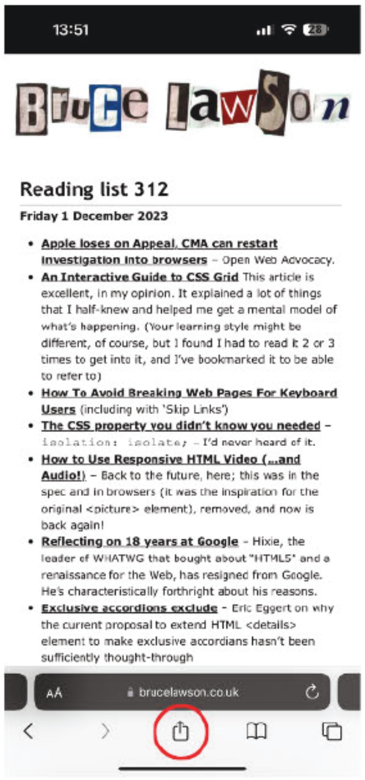
3.3.1. Install Prompts

In order for Web Apps to fairly contest the iOS App Store, they need to be as visible as Native Apps. Apple has progressively added features to iOS Safari to push users towards Native Apps on their app store, however refuses to add the equivalent feature for Web Apps: Install Prompts, despite incredible pressure from developers over many years.

Install Prompts is the essential missing feature for Web Apps, without which Web Apps will not be able to compete with Native Apps.

Currently the installation of Web Apps in Safari is completely hidden away under an awkward multi-step process which the user must access through the "share" menu. The install process for Web Apps are so obscured in Safari that the majority of users are unaware of their existence.

In order for Web Apps to be able to fairly contest Apple's native app ecosystem, and to enable businesses to take over the extensive advantages Web Apps have to offer (including Security/Interoperability/Cost) **Apple must be compelled to provide an effective implementation of Install Prompts from within Safari.** Apple will not provide this without regulatory intervention or pressure.

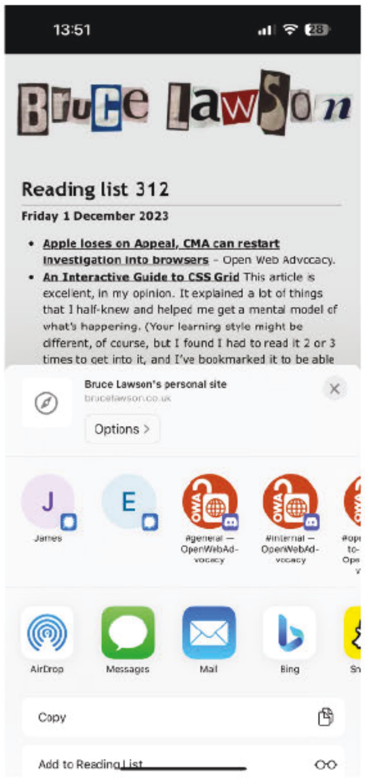


Reading list 312

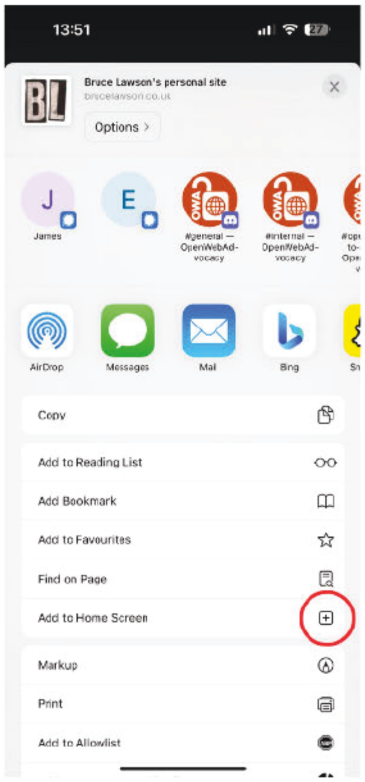
Friday 1 December 2023

- Apple loses on Appeal, CMA can restart investigation into browsers – Open Web Advocacy.
- An Interactive Guide to CSS Grid This article is excellent, in my opinion. It explained a lot of things that I half-knew and helped me get a mental model of what's happening. (Your learning style might be different, of course, but I found I had to read it 2 or 3 times to get into it, and I've bookmarked it to be able to refer to)
- How To Avoid Breaking Web Pages For Keyboard Users (including with 'Skip Links')
- The CSS property you didn't know you needed – `isolation: isolate;` – I'd never heard of it.
- How to Use Responsive HTML Video (...and Audio!) – Back to the future, here; this was in the spec and in browsers (it was the inspiration for the original <picture> element), removed, and now is back again!
- Reflecting on 18 years at Google – Hixie, the leader of WHATWG that bought about "HTML5" and a renaissance for the Web, has resigned from Google. He's characteristically forthright about his reasons.
- Exclusive accordions exclude – Eric Eggert on why the current proposal to extend HTML <details> element to make exclusive accordians hasn't been sufficiently thought-through

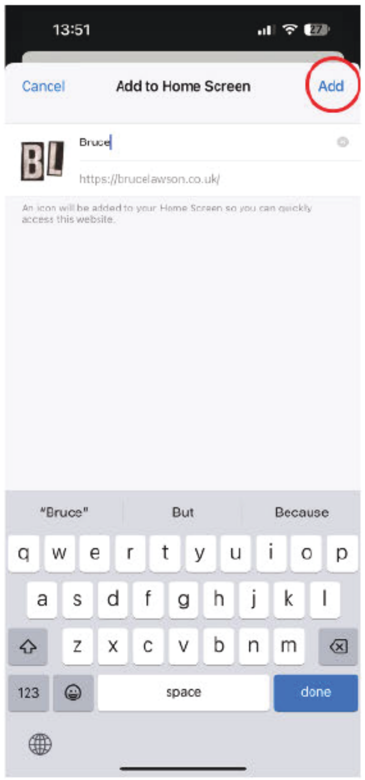
1. Tap the **SHARE** icon



2. Scroll down to the **HIDDEN** part



3. Tap **"Add to Home Screen"**



An icon will be added to your Home Screen so you can quickly access this website.

4. Tap **"Add"**

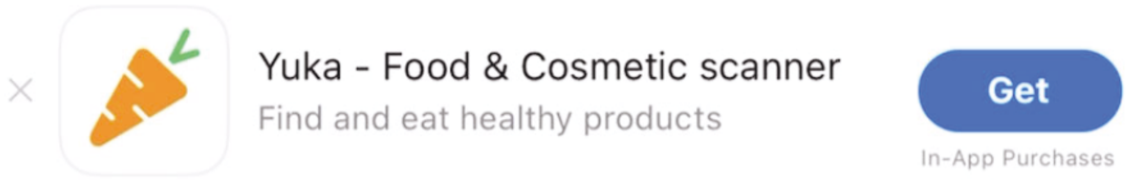
3.3.1.1. Apple makes Native Apps very easy to Install

Apple has gone to extensive efforts to make native apps easy to install via Safari. They provide a wide range of methods that enable users to easily install and use native apps. Each of these methods provides an easy, intuitive and understandable series of steps for the user to be able to install native apps.

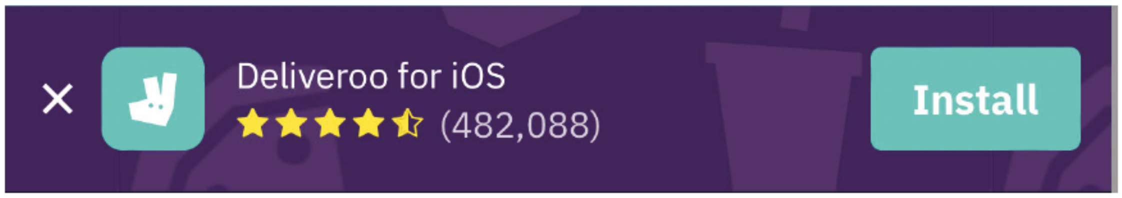
These include:

- Smart Banner (Direct Install from Web Page in Safari)
- Custom Install Banners (Opens AppStore)
- Smart Banners (Opens AppStore)
- App Clips (Direct/Fast Load From Web Page in Safari)
- Download on the App Store Badges

1. **Smart Banner - Direct Install from Web Page in Safari**
   The smart banner visible on a webpage from within Safari allows for a "Get" button which immediately triggers the install interface without even having to visit the AppStore.

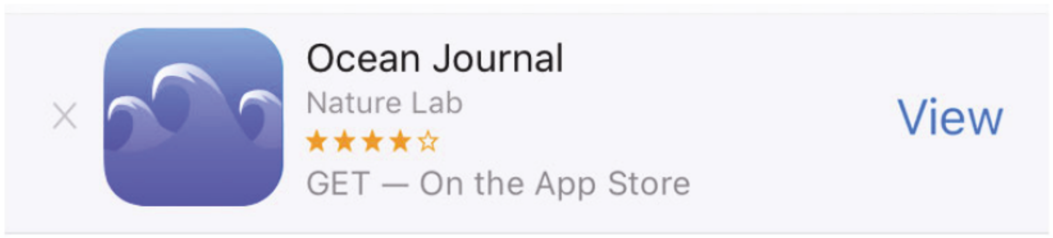


2. **Custom Banner - Takes users directly to the AppStore from Safari**
   Developers can create their own styled custom buttons in Safari by styling a link which takes the user directly to the app's install page on the AppStore.



3. **App Clips - Quickly Load App from a Web Page from Safari**
   App Clips allow the user to directly load an app (without having to install it) from within Safari.

4. **Smart Banner - View in AppStore**
   The Smart Banner can contain a view button which will take the user directly to the app's install page in the AppStore.



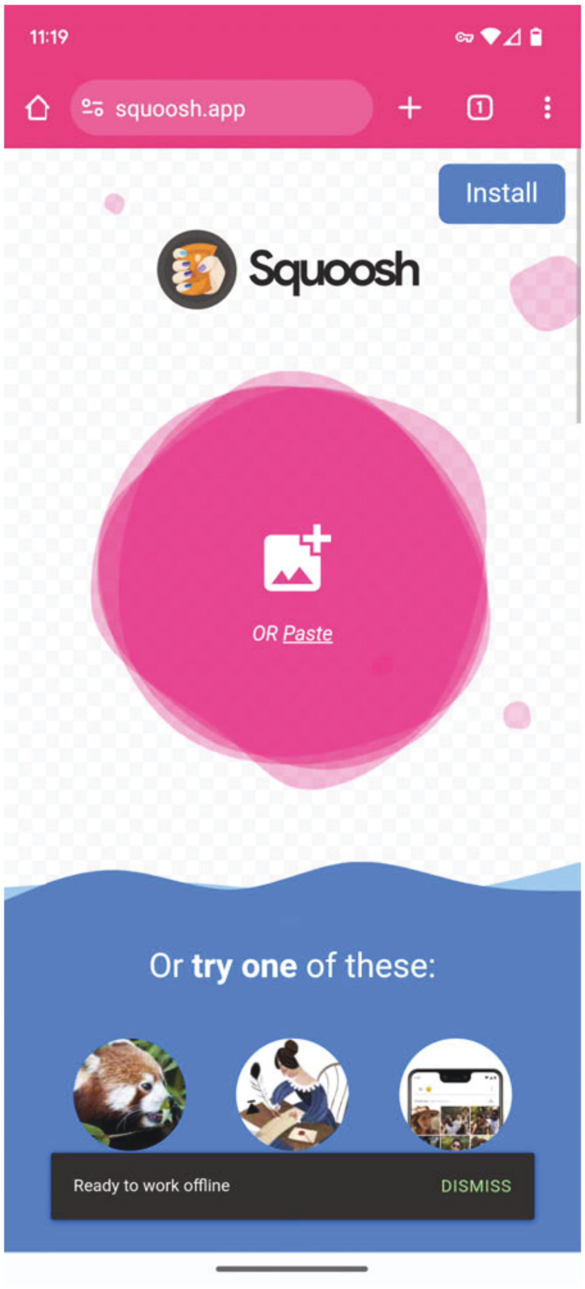
5. **Download on the App Store Badges**
   Identical in implementation to Custom Banners by using a styled link, the "Download on the App Store" badge enables the user to directly open the App Stores install page. Alternatively Apple also provides tools to generate an app icon with equivalent functionality.

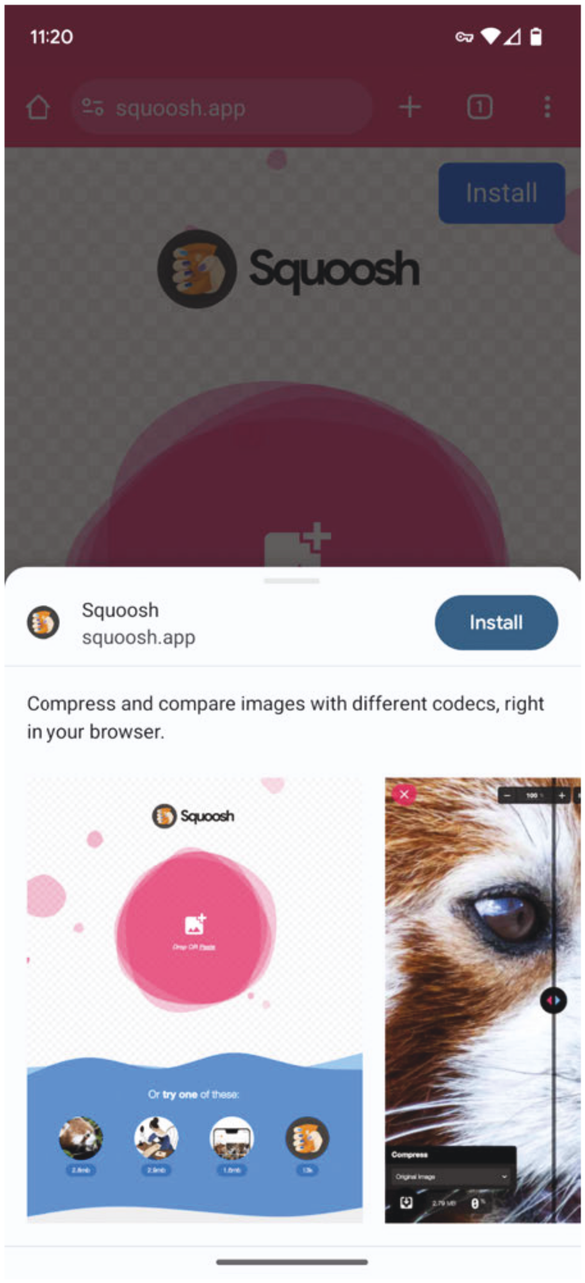
Download on the
App Store

3.3.1.2. Web Apps are Significantly Easier to Install on Android

Unlike on iOS, users on Android can follow a significantly easier path to install and access Web Apps. The two main methods to install web apps on Android are Developer Led and Browser Led install prompts.

3.3.1.2.1. Developer Led Install Prompts



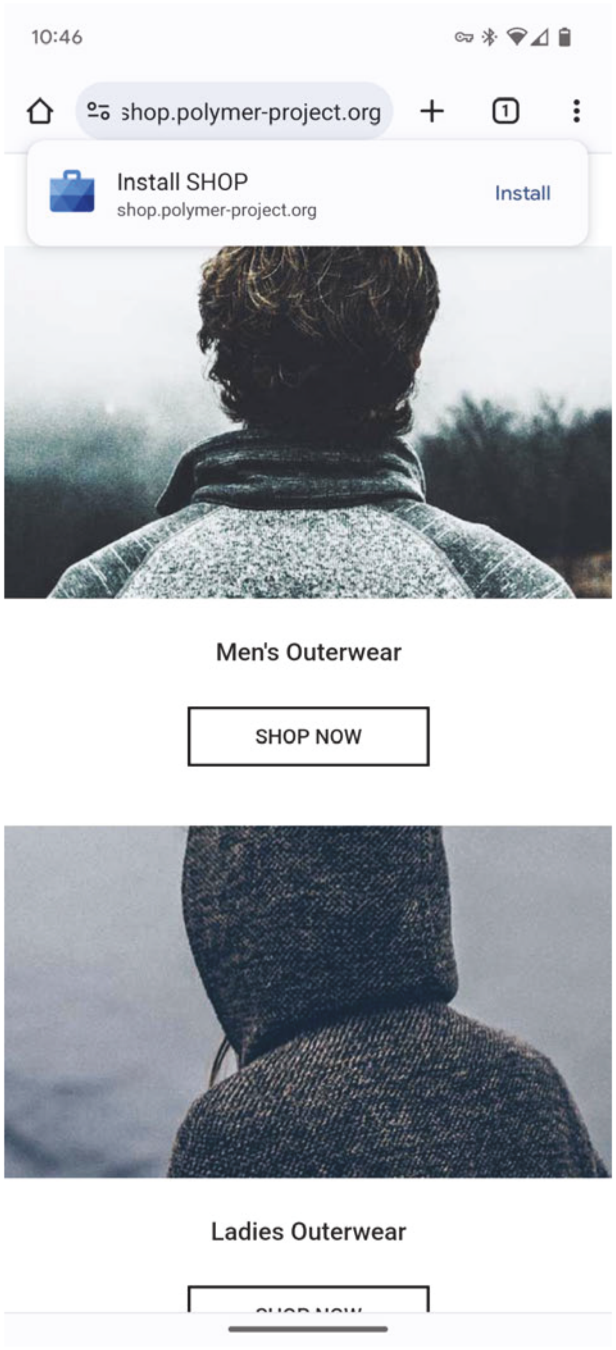


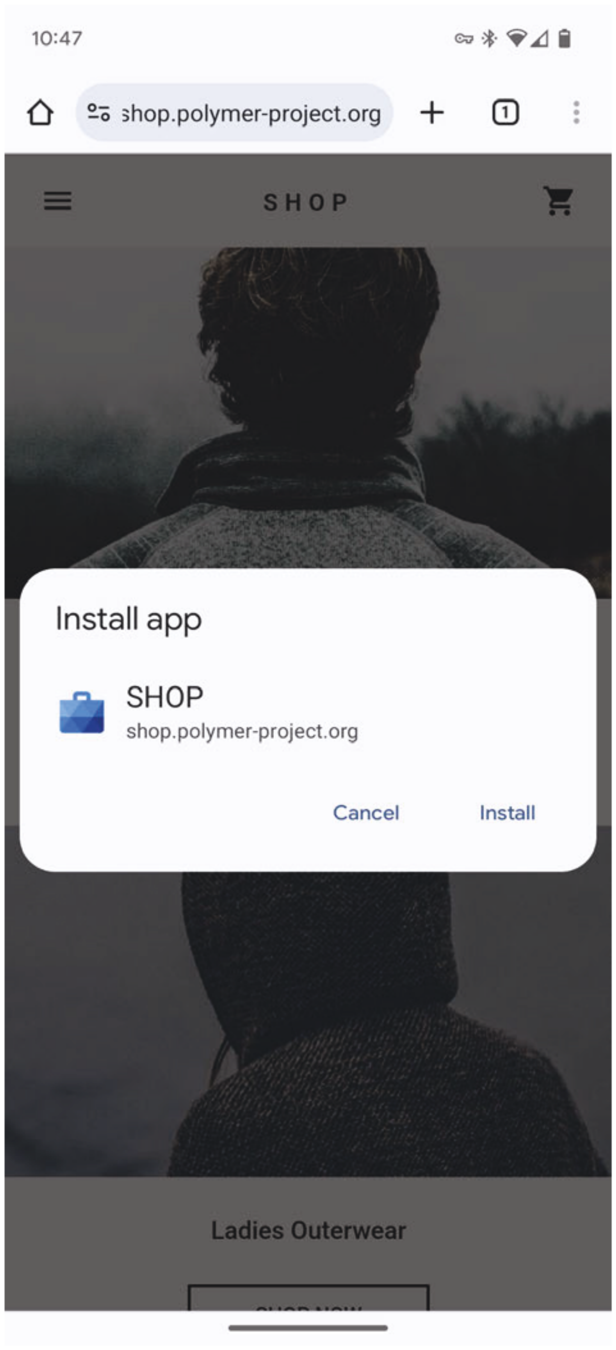
Compress and compare images with different codecs, right in your browser.

Developers can create prominent custom install buttons on Android to install Web Apps, which allows for a “two tap” install of a Web App.

3.3.1.2.2. Browser Led Install Prompts

If the site meets minimum quality criteria (offline support and icons from the manifest), chromium based browsers can display prominent prompts on the developers behalf.



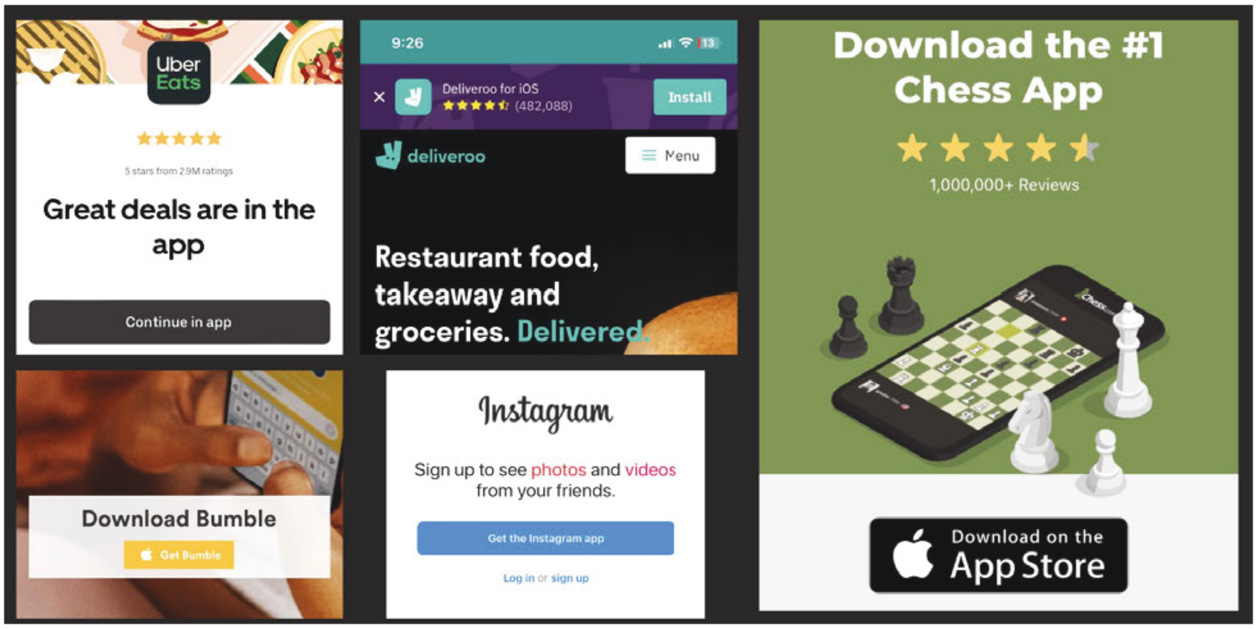


3.3.1.3. Regulatory Intervention

Apple should be obligated to implement install prompts, including both automatic prompts (akin to Smart Banners) and the ability to programmatically display prompts on a button tap from within websites.

Given that Apple has withheld this functionality for almost a decade, we believe that Apple should not be able to delay discussions of the specifications as this could take months or even years. Rather, they should implement the current specification as implemented in chromium browsers (onbeforeinstallprompt) as is currently available in other browsers which support the feature. Only after implementing the existing functionality, should Apple then be able to propose any updates or upgrades to prevent needless delay.

Apple provides users a wide range of methods to easily install Native Apps, it's essential they provide the equivalent to Web Apps.


Uber Eats
5 stars from 2.9M ratings
Great deals are in the app
Continue in app
9:26
Deliveroo for iOS
(482,088)
Install
deliveroo
Menu
Restaurant food, takeaway and groceries. Delivered.
Download the #1 Chess App
1,000,000+ Reviews
Download Bumble
Get Bumble
Instagram
Sign up to see photos and videos from your friends.
Get the Instagram app
Log in or sign up
Download on the App Store

Examples of custom buttons on websites in Safari.
Tapping on these buttons takes users directly to the App Store listing.

3.3.1.4. Effective Implementation

For Install Prompts to be effective they need to be built in a way that does not undermine their effectiveness. A number of success criteria could be written to judge their effectiveness:

No Scare Screens
Apple should not be allowed to downground or be allowed to add additional scare screens or other friction as it is not necessary from a security perspective. The ability to install Web Apps via iOS Safari has been available for almost 15 years without such measures, and Apple has not felt it was necessary to implement such measures. They surely cannot be justified now, particularly after Apple's strong claims regarding the security of its iOS Web App container.

Web Apps are protected by the sandbox of the browser, which Apple themselves state is "*orders of magnitude more stringent than the sandbox for Native Apps*". This is important as it means that Apple does not have any significant unmitigatable security objections to such a change.

Ability to Measure Performance
The ability to measure the performance of different install prompts and pages advertising the app. This is functionality that Apple currently encourages and supports for installation of Native Apps via iOS Safari.

User Acquisition and Engagement
The install rate is the primary metric indicating how effective the prompt is in converting users from browsing to installing the Web App. Safari should not have a significantly worse install rate than other browsers which support installing Web Apps. If they do, it suggests that their design/implementation is inferior or engages in dark patterns to dissuade the user from choosing to install a Web App.

Apple should be obliged to collect and publish to the UK aggregated anonymized data in order to show effective compliance.

Appropriate Language
The wording used in the install prompt should be straightforward and should not be used as a deterrent to prevent users from Web Apps.

The language should be consistent with other operating systems and use the word "Install". Language should not be used to indicate to the user that they are simply adding a "bookmark" as this may undermine user's perceptions in regard to the capabilities of Web Apps which may harm adoption.

We have heard that Google conducted research on language relating to Web App install, and would recommend the UK reach out to see if they could publish or share their findings.

Minimum Required Steps
Install prompts should require no more user interaction than the **minimum** required to meaningfully inform the user regarding the installation and gain their consent for the install. Each additional unnecessary step will increase friction and block Web Apps from contesting and fairly competing against their Native counterparts.

Speed
There should be no degradation in install speed from the current implementation of the "Add to homescreen" functionality which is currently near instant.

3.3.2. Incomplete Support for Push Notifications

Push notifications are an alert from an app. A simple example is that you have received a message from a friend. They were first introduced for native apps on iOS in 2009 and Android in 2010. The feature got progressively more sophisticated and these days apps can now customize and include images in their notifications. Today, billions of push notifications are received every day.

While browsers on Android have had push notifications for years, it was noticeably absent from iOS Safari. Push notifications were a highly requested feature missing from iOS Safari for years. It was our second in our list of most important features for Web Apps (after install prompts).

As covered in this excellent article, while Apple under pressure from regulators has implemented push notifications for iOS Safari, the feature is extremely limited and does not provide a good user experience for most use cases.

> *"In the end, it turns out that Apple shipped an implementation of Web Push that was...*
>
> *... good enough to convince anyone that didn't dig too deep (or did any digging at all) that Web Push is now available on iOS.*
>
> *... incomplete enough to actually be unusable for anything more complex than websites that want to push "breaking news stories" - you know, the sites that have given push notification prompts such a bad name because they ask for the permission on page load.*

Since the initial launch of Web Push on iOS, not much has happened. It's been a year and the list mentioned above is still pretty much accurate. Web Push on iOS is still very much incomplete and, I can't stress this enough, you can not rely on feature detection to prevent a bad user experience.

[...]

I can't imagine anyone actually using Web Push in its current form on iOS. Project teams have probably decided it's not worth to do the extra work to deliver an experience that is, even with the extra work, subpar at best. The chance that a user will install the web app (and knows how to do it) is close to zero. With that in mind, I imagine the Safari team, looking at usage numbers of Web Push on iOS, reaching the conclusion that "Nobody is using it, see?! We should put our efforts into other areas! We need to have high numbers on Interop or risk embarrassment! As long as MDN and Caniuse show all these green checkmarks, we're fine!"

[...]

With actual competition, I think Apple would think twice before they launched another API as incomplete and broken as Web Push."

Webventures - Web Push on iOS - 1 year anniversary
(emphasis added)

There are a range of problems holding back iOS Safari's implementation of push from being a variable substitute for the one provided to native apps on iOS.

3.3.2.1. Strong Barrier to Use - Requires Web App Installation

In Apple's implementation there is a significant barrier to asking permission to provide push notifications, first the Web App must be installed. This is not necessarily an unreasonable policy by itself but when combined with the lack of install prompts discussed in the previous section and how hidden the option to install a Web App is, it significantly weakens the experience and attractiveness of Web Apps.

"[...] adding a website to the home screen remains such a hidden feature that even power users would be forgiven for not knowing about it."

Push - Jeremy Keith

3.3.2.2. Missing Many Important Subfeatures

Push notifications are also missing a great many important subfeatures. These features are needed to provide a good experience to end users which is comparable to that of native apps. While these features are not universal in all other browsers, many popular browsers on Android have implemented them and both Android and iOS have had equivalents available for push notifications from native apps for years.

3.3.2.2.1. Custom Icon

It is not possible for the Web App to set a custom icon on a push notification. Safari will instead only show the app icon of the Web App.

This ability to change the icon is commonly used to show the profile picture of the person who is messaging you in a chat application.

3.3.2.2.2. Image

The Web App can not show a preview of an image in a push notification. So for example if a user in a chat Web App sends a picture, you can not display it in the notification as is standard in native chat apps.

3.3.2.2.3. Custom Timestamp

The Web App can not set a custom timestamp in a push notification. This is useful to set the timestamp to for example the time a message was sent.

3.3.2.2.4. Action Buttons

You can not add custom buttons to a push notification. These buttons can trigger the Web App to perform a specific action, i.e snooze, silence notifications from a particular user etc.

3.3.2.2.5. Update Notification

It is not possible to update a notification that has already been sent. This can be useful for a variety of scenarios, such as a user has edited their message or more information is available.

3.3.2.2.6. Get List of Current Notifications

It is not possible for the Web App to get a list of the current notifications from itself that are being displayed to the user. This is extremely useful in covering edge cases involved in updating or closing notifications.

3.3.2.2.7. Group Notifications

It is not possible for the Web App to group notifications. This is useful for keeping notifications from different chat groups separate and provides a smoother and more pleasant experience for end users.

3.3.2.2.8. Close Notification

It is not possible for the Web App to close a notification, that is remove it from the list being displayed to a user. This is commonly used by native chat apps for when a user views a message in the desktop version, the notification is removed from the mobile app.

3.3.2.3. Subpar for Most Use Cases

The combination of friction in making this functionality available in the first place and the subpar experience it provides to end users, make it deeply unattractive to developers relative to what is possible for native apps. At a technical level there is nothing stopping Apple from providing an equivalent experience for Web Apps as they do for native apps but currently they have chosen not to do so. Worse, due to their browser engine ban and refusal to allow third party browsers to install and manage Web Apps with their own engines, it is impossible for a third party to provide a better service.

To fix Apple would need to implement the above missing sub-features and implement install prompts.

3.3.2.4. Feature Detection does not Work

Feature detection is the process of using automated programmatic checks to see if a browser supports a feature and then adjusting the page accordingly. That is, before a developer uses a feature, the developer can ask the browser if it supports it. If it doesn't then the developer can hide or disable the relevant button or adjust the page in some other user friendly way.

This is best practice for frontend web developers.

Unfortunately on release when Apple released Push Notifications in a single push, they decided to make it appear all these features work, even when they didn't.

This gave the unsupported subfeatures a green tick for each of these features on sites developers use to work out support but made the practice of feature detection not only useless but actually harmful. If a browser lies and says a feature works when it doesn't, this can lead to severe bugs and a poor user experience.

One good example of this getting a list of currently displayed notifications. Safari on release nominally supported this with the standard function getNotifications. It would even return a list. However the list would always and often incorrectly be empty.

It's a little baffling how an otherwise professional team such as the Safari team managed to make this mistake.

3.3.3. Screen Orientation Lock

Orientation lock is a method of indicating to the browser a page would like to be locked in particular orientation. For example horizontal or vertical. This is only available on mobile devices. Orientation lock is very important for computer games that tend to be horizontal. An analogous feature is available for native games on both iOS and Android. Without this many games feel extremely buggy if the viewport jumps orientation every time the user tilts their phone. This feature is missing from Safari making many games feel unpolished or odd due to their vertical orientation.

3.4. Third Party Browsers can not compete in the provision of Web Apps

> *"We all rely on browsers to use the internet on our phones, and t**he engines that make them work have a huge bearing on what we can see and do**. Right now, choice in this space is severely limited and that has real impacts – preventing innovation and **reducing competition from web apps**. We need to give innovative tech firms, many of which are ambitious start-ups, a fair chance to compete."*
> Andrea Coscelli - Chief Executive of the UK's Competition and Markets Authority

As the MIR can not effectively obligate Apple to provide the best or least buggy experience for Web Apps, the only truly effective measure is to allow third-party browsers to compete in providing Web Apps the best or least buggy experience using their own engines.

In fact the very rationale in the EU's Digital Markets Act for enforcing browser engine competition was to prevent gatekeepers from determining and imposing the functionality on *"web software applications"*.

> ***When gatekeepers** operate and **impose web browser engines, they are in a position to determine the functionality and standards that will apply not only to their own web browsers**, but also to **competing web browsers** and, in turn, to **web software applications.** Gatekeepers should therefore not use their position to...*
> Digital Markets Act

Google mechanism for installing Web Apps, WebAPK minting, is built on Android Custom Tabs and already has the ability to support any installing browser. Google however has refused to share this functionality. While third-party browsers can install Web Apps on Android, they do not integrate as well into various operating system surfaces, we cover that in section 5.

Apple however has not built such a mechanism on iOS. Their Web App solution is built on top of WKWebView via a component called web.app and is in essence locked to Safari's engine. Apple will need to update iOS such that it allows third-party browsers the ability to install and manage Web Apps using their own engine.

Simply providing access to web.app is not sufficient as it is locked to the specific version of WebKit that comes with iOS and improvements added to browser vendors browsers or engines would have no impact.

Allowing browser vendors to compete in this fashion does two things.

First, consumers and web developers can bypass Safari and gain a better experience installing a Web App with another browser. While there is a significant barrier to this, developers really don't like to bother consumers, in certain cases for critical functionality developers may convince consumers to switch. As a thought experiment, imagine Web Fortnight only worked well in third-party browsers due to missing or buggy functionality in Safari. Many users might be convinced to make the leap. Cumulatively over time this can lead to an exodus of users.

Second, this style of pressure and fear of losing users leads to vast pressure to fix issues in a way that regulators could never enforce. Safari wins Apple a vast amount of revenue ($200 million USD per year for each 1% market share) which means Apple will be heavily incentivised to stop even small losses of market share.

Between these two effects, and particularly in combination with the other remedies, Web Apps will become not only viable on iOS and Android but an attractive option for businesses. This in turn applies competitive pressure on both the iOS App Store and the Google Play Store. Suddenly both app stores will have a genuine fear of developers abandoning their app store and developing Web Apps. This applies pressure to be reasonable on pricing and policy.

This interoperability which is effectively absent due to current non-viability of Web Apps on iOS will allow businesses significant cost savings and improved quality due to only needing a single code base for both platforms.

> *"**Addressing the complex security and privacy concerns associated with web apps using alternative browser engines would require building an entirely new integration architecture that does not currently exist in iOS** and was not practical to undertake given the other demands of the DMA and the very low user adoption of Home Screen web apps. **And so, to comply with the DMA's requirements, we had to remove the Home Screen web apps feature in the EU.**"*
>
> Apple's statement

(emphasis added)

Apple's response to having to allow third party browsers to install Web Apps using their own engine due to the Digital Markets Act is instructive. Their first instinct was to silently break the feature rather than share it.

Then, under significant pressure, reversed that decision and stated that they will keep the current status-quo, implying that access to the underlying APIs required to run Web Apps would be locked to the WebKit implementation and that third-party browsers using their own engine would not be provided sufficient API access to contest it. That is, third-party browsers would continue to be unable to install and manage Web Apps using their own engine.

> *"This support means Home Screen web apps continue to be built directly on WebKit and its security architecture, and align with the security and privacy model for native apps on iOS."*
>
> Apple's statement

We believe that without regulatory intervention Apple will never allow third-party browsers the ability to compete in the provision of functionality and stability of Web Apps on iOS.

3.5. Harms

3.5.1. This Harms Consumers

The report's survey of consumers highlights something that the development community has long known. The majority of consumers are completely unaware of Apple's restrictions on browsers and it is a choice that has little importance to them. Many users who have downloaded Firefox or Chrome on iOS believe they have the same browser that ships on other operating systems. They are entirely unaware that Apple has forced them to essentially build a rebranded skin around Safari.

The development, maintenance and lost opportunity costs of supporting a buggy browser that misses key features are mostly hidden from them. It is hard for consumers to see a missing feature or an entire Web App that didn't get built (due to poor support in iOS Safari). When they do encounter a bug caused by Safari they are more likely to blame the Web App than the browser. The user may get the impression that the web is buggy, slow and that native apps are better, which then has negative flow on effects for the entire web ecosystem.

As with other industries, the fact that consumers do not understand they are being harmed does not mean that harm is not real. In the same manner that consumers, for example, might not understand the intricacies of an elaborate scheme to extort higher prices via manipulating complex power supply bidding rules, consumers do not need to understand the manner in which Apple has suppressed browser competition. The regulator's role is to protect consumers from such complex and underhanded behavior.

The consumer is harmed in 3 main ways.

First, lack of browser competition on iOS worsens the feature set, stability, security and privacy of browsers on iOS. Fierce browser competition is the key to forcing browser vendors to invest heavily and fight for market share.

Second, this leads to increased costs for developers. These costs must inevitably be in part handed onto consumers. This is hidden from consumers as it will never be clear where the additional cost is coming from.

Finally, this undermines the entire Web App ecosystem. Apple and Google charge an up to 30% commision on all native apps delivered via their app stores. Having a viable, open and interoperable contender would provide both an alternative and a significant competitor driving down prices and improving quality. 90% of the apps on consumers phones could be rewritten as Web Apps today were they given the features and stability that they need to compete fairly. This transition is already well underway on desktop, where the more open model prevents operating system gatekeepers from blocking Web Apps from competing. This means that Apple's behavior is arguably costing consumers billions of pounds in fees that Apple would not have the power to collect on this more open model of app development.

3.5.2. This Harms Businesses

Apple's effective ban of third-party browsers on iOS, bugs and lack of support for Web Apps harms businesses in a number of ways. It even harms businesses that have never built a Web App or considered building a Web Apps.

Apple and Google charge a 30% commision via their ability to block/hinder consumers from installing and using apps from outside their app stores.

Viable Web Apps would allow businesses to have a direct relationship with users without having to pay either Apple or Google. Another key advantage of Web Apps is that they are interoperable between operating systems. That is, developers only need to write a single version of their app for all operating systems.

In many cases iOS is the primary target market for mobile apps due to the wealth of their users. Businesses when deciding what technology to use, often will use what works on iOS as the basis for that decision.

Currently Web Apps are not viable on iOS due to missing key features such as install prompts and, severe and persistent bugs such as with scrolling in more app-like layouts. This lack of support is entrench by Apple's refusal to let third-party browser vendors compete. On Windows, when Internet Explorer did not provide what businesses needed, third-party browsers became available that did. Over the next 15 years Internet Explorer faded into irrelevance. Such a scenario could never play out on iOS under Apple's rules, nor would Apple be driven to add features or fix bugs to avoid such a fate for Safari.

This harms businesses that use Apple's app store, as the lack of viable alternative to reaching consumers on iOS removes any bargaining power they might have to either reduce fees or improve the service that Apple's offers by, for example, making app review less hostile. [redacted]

Were Web Apps viable on iOS this would be different, every business would have the option to deliver their apps directly to customers using Web Apps via competent browsers. The only path towards this future is allowing browsers to be able to compete fairly and effectively on iOS, including the ability to install and manage Web Apps using their own engines.

3.5.3. This Harms Browser Vendors

> *"The evidence shows that the WebKit restriction has significant implications for browser vendors. Given the importance of the browser engine to a browser's features and performance, the inability to use an alternative browser engine limits the ability of browser vendors to innovate and improve their browsers on iOS. Browser vendors are less able to add features and improvements to their browsers on important parameters such as security, privacy, performance, and innovations on iOS relative to less restricted platforms such as Android or desktop. This reduces the features available to consumers and limits effective competition on these parameters.*
>
> *In addition, we have obtained evidence indicating that browser vendors incur additional costs from having to develop and support a version of their browser based on WebKit, which they would not do if the restriction were not in place. Evidence from browser vendors also indicates that Apple is difficult to engage with regarding requests for fixes or the addition of new features to WebKit on iOS."*
>
> Browsers and Cloud MIR - WP2

We agree with the MIR team's preliminary thinking on the matter. We do however believe this conclusion understates the scale of the harm to browser competition, the scale of the loss of revenue and the degree of the additional costs for third party browser vendors.

3.5.3.1. Harm to Browser Vendor Competition

The report correctly states that the majority of what distinguishes a browser takes place within its browser engine. This can either be by both the choice of the browser engine and by modifications that are made to the browser engine by the downstream browser vendor.

For example Microsoft, Opera, Vivaldi and Brave use the Blink engine but then make significant alterations to the browser engine either to add, remove or modify features (for example adding additional privacy features). Firefox uses its own engine Gecko and in general every feature or bug fix is implemented by Mozilla.

Both methods of distinguishing the browser are denied on iOS. Browser vendors can neither choose their own engine nor modify WebKit. Any feature, bug, security or privacy issue that is contained within the WkWebView that Apple forces the browser vendor to use is unchangeable by those browser vendors.

If the WebView is missing a feature or contains a severe bug, browser vendors are powerless to fix or improve on it in their own product. This removes all effective competition.

This is important as it simultaneously removes the ability of third-party browser vendors to contest Safari and removes a powerful incentive on Apple to add features and fix bugs in Safari, fear of losing market share.

3.5.3.2. Harm to Browser Vendor Revenue

Apple via Safari, collects $20 billion annually in advertising revenue from Google via its Google Search deal. Apple is able to collect this sum due to its browser market share on iOS which we would argue a significant percentage of has been gained and maintained by anti-competitive behavior.

On the basis of this deal, this means each 1% browser market share on iOS is worth around $200 million USD per year. As a point of reference Mozilla's entire budget for 2023 was approximately $450 million USD. It seems entirely reasonable that Apple's effective ban of Mozilla's and other browser vendors' browsers, removing their ability to compete on iOS has had a substantial effect on their revenue. We can not know what would have happened had Apple not imposed this restriction for the last 14 years, but it plausibly has halved, quartered or more the budgets of the smaller browser vendors.

Mozilla has wanted to port their browser to iOS for over a decade. Had they been allowed, they would have likely been the first browser on iOS to support extensions and likely grabbed significant market share. This would have then provided them the revenue to continue to invest in their mobile browser and enjoy the mobile to desktop network effects from having a significant mobile market share.

3.5.3.3. Degree of Additional Costs

Regardless of whether the MIR has the power to force Apple to remove this anti-competitive rule globally, browser vendors being able to compete fairly globally should be the reference point in any analysis.

Both the EU and Japan have already passed laws that explicitly ban the practice. Should the UK and other regulators apply and enforce similar remedies, it seems likely that this will be adopted globally. If forced to allow browser competition on iOS in a sufficiently large percentage of the iOS user base, Apple will likely give up and allow browser competition on iOS globally.

With this in mind it will be worthwhile to ask each browser vendor to estimate the annual cost of maintaining an entirely distinct browser built on the uneditable WkWebView vs simply using the same browser it has on every other operating system.

Each browser vendor tends to have entirely dedicated teams just on the iOS version of their browser. While these teams would be smaller (due to an inability to meaningfully add features, fix bugs vs Safar), the costs are likely not insignificant.

3.6. How to Fix?

3.6.1. Remove Apple's Browser Engine Ban

We recommend that the MIR obligates Apple to remove its restriction preventing third-party browsers from using their own engine.

If the MIR is unable to impose this globally then at a minimum it should seek to allow genuine and effective browser competition within the UK. Given the clear anti-competitive nature of such a ban and the harm it inflicts on consumers, developers, businesses use the Web to reach consumers and browser vendors such a remedy is entirely justified.

As an additional proportionality argument, Apple is already legally obligated to remove this rule both in the EU and in Japan. This means any additional cost to Apple for doing so in the UK or globally (not including decreased search engine or app store revenue) should be minimal.

3.6.2. Allow Third-Party Browsers access to needed Software and Hardware APIs

In order to compete with their own engine, browsers need access to a wide range of software and hardware APIs. In many cases on iOS these APIs have not been shared, nor have they been properly documented by Apple.

We believe that the MIR should impose an obligation on Apple to share these APIs with third party browser vendors. Apple should be able to place strictly necessary, proportionate security rules on access to these APIs provided that Apple extensively and publicly justifies each individual rule. These rules should be publicly available in their entirety in a single document. Apple should be obligated to justify the security necessity and proportionality of any rule and any change or addition to these rules well in advance.

In the event that a browser vendor significantly or repeatedly breaches these agreed upon rules, Apple should be able to revoke access to these APIs. Apple should face financial penalties where they can not justify such revocation.

In certain cases where Apple has not provided an API, not publicly documented an API or has designed an API in such a many that it can not be used by third party browsers, Apple will need to update and document such APIs.

Both the EU and Japan have imposed similar conditions on Apple, so much of this work Apple is already obligated to perform over the next few years. This means the additional cost of performing such work for the UK is zero.

3.6.3. Obligate Apple to implement Install Prompts for Safari

As discussed at length in the section, we believe that Apple should be obligated to implement install prompts for iOS for Safari. Install Prompts are essential to ensure Web Apps can contest the app stores of both Apple and Google.

3.6.4. Allow Third-Party Browsers to install and manage Web Apps with their Own Engines

Apple does not currently allow third-party browsers to install and manage Web Apps using their own engine. From statements made in response to the DMA, it appears that Apple never intends to provide this functionality.

We believe it is both justified and proportionate that the MIR team obligates Apple to update iOS (including derivatives such as iPadOS) such that third-party browsers can do so.

3.6.5. Prevent Secret Side-Agreements that undermine Browser Competition

We are concerned that Apple may be undermining browsers ability to compete on iOS by forcing browser vendors to sign secret additional agreements that restrict their ability to compete with Safari.

We believe the MIR team should request copies of all such agreements and prohibit Apple from having such secret agreements with browser vendors.

3.6.6. Prevent Browsers being effectively banned by Unfair or Malicious Contracts

Throughout most of the world, Apple explicitly bans third party browsers from using their own browser engine.

> *"2.5.6 Apps that browse the web must use the appropriate WebKit framework and WebKit JavaScript. You may apply for an entitlement to use an alternative web browser engine in your app. Learn more about these entitlements."*
>
> Apple App Store Guidelines
> (Recently Updated Text)

Recently, Apple has updated the wording of rule 2.5.6. On first glance it might appear they have allowed eligible browsers to globally be updated with their own engines. However they have in actual fact just moved the ban inside their browser engine entitlement contract which makes it explicit that only browsers within the EU are eligible.

This is due to the fact that the app store rules were specifically updated in order to be compliant with the Digital Markets Act which prohibits such a ban. However, as we extensively argue in our paper, they have an effective ban still in place via making the contract for the APIs required by browsers using their own engines so unreasonable, restrictive and scary that no browser vendor would dare sign it for a production browser.

Some non-exhaustive examples of problems with the contract are:
- Vendors must abide by all rules in "Apple Materials", a vague term encompassing thousands of documents that Apple can update at any time with no notice.
- Any breach of the rules (no matter how minor) grants Apple the right to remove all of the apps of the browser vendor from every operating system that Apple controls.
- Browser vendors can not update their existing browser and must ship a new EU only browser, thus losing all their customers.
- Apple can remove the browser vendor's browser for any reason.
- Apple can break or remove any API (including from a single browser vendor) at any time at its sole discretion.
- Browser vendors rebuild their browsers on particular iOS components such as scroll.

What is key here, is that even if the UK mandates that Apple can no longer have such a ban, we are concerned that Apple will try a similar strategy within the UK where they instead switch to a quasi-ban by unreasonable contract terms. The solution is to keep Apple strictly to terms which relate to protecting security and to closely scrutinize the necessity or proportionality of any term. Terms that would severely undermine browser competition should be struck out.

3.6.7. Allow Direct Installation of Browsers

Another important avenue to improve browser competition on iOS and Android is to allow browser vendors to also offer their browsers outside the app stores. Essentially browsers that have been granted the browser entitlement, would be able to offer their browsers directly from their own websites, where users could be prompted to install them. This is already obligated in the EU under the Digital Markets Act. Thus Apple will simply need to turn on this functionality for the UK or globally.

This is important as there is great fear in the industry that Apple wields app store review as a weapon to punish competitors. This fear appears to be well justified.

On March 5th, Spotify submitted an update to Apple that puts links to Spotify's website, along with pricing information for different subscription options, directly in the EU version of its app, without using Apple's payment system. Despite being fined 1.8 billion by the EU on this very topic, Apple refused to let the update pass app review even though more than 2 weeks had passed since the update was submitted.

> *"Moreover, Apple has demonstrated its ability to use its smartphone monopoly to impose fee structures and* ***manipulate app review to inhibit aggrieved parties from taking advantage of regulatory and judicial solutions imposed on Apple*** *that attempt to narrowly remedy harm from its conduct."*
>
> DOJ - Case 2:24-cv-04055
> (emphasis added)

> *"Specifically, Apple sets the conditions for apps it allows on the Apple App Store through its App Store Review Guidelines. Under these guidelines, Apple has sole discretion to review and approve all apps and app updates. Apple selectively exercises that discretion to its own benefit, deviating from or changing its guidelines when it suits Apple's interests and allowing Apple executives to control app reviews and decide whether to approve individual apps or updates.* ***Apple often enforces its App Store rules arbitrarily. And it frequently uses App Store rules and restrictions to penalize and restrict developers that take advantage of technologies that threaten to disrupt, disintermediate, compete with, or erode Apple's monopoly power.****"*
>
> DOJ - Case 2:24-cv-04055
> (emphasis added)

> *“there are endless horror stories around curation of the store. Apps are rejected in arbitrary, capricious, irrational and inconsistent ways, often for breaking completely unwritten rules.”*
>
> Benedict Evans - Technology Writer

> *“There's a lot of talk about the 30% tax that Apple takes from every app on the App Store. The time tax on their developers to deal with this unfriendly behemoth of a system is just as bad if not worse”*
>
> Samantha John - CEO Hopscotch

Given that third-party browsers using their own engine will likely need to comply with extensive security rules in order to be allowed access to the relevant APIs, there is no security justification for any additional scare screens or warnings.

Currently, only a few browser vendors offer their browser through the macOS app store. It seems plausible that browsers will want to abandon the iOS app store at some point in the future to avoid having to deal with any unreasonable app store guidelines that Apple might impose.

4. Browser Choice Architecture

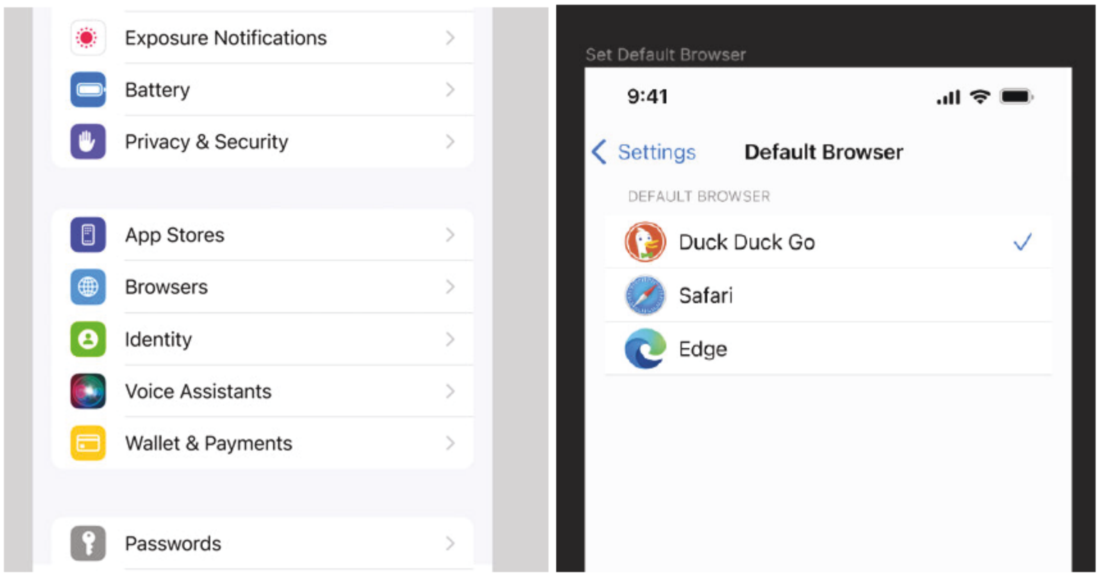
4.1. No Special Location for Safari

On the settings page for iOS, pre-installed Apple apps are not placed with the other apps. Instead they are given a special, far more prominent location, in the settings. Other third-party apps are shown in a separate location further down the settings page. This divide suggests to users that these are official apps they should be using (which come pre-installed) and other apps are “alternative apps”.

It indicates to the user that these apps are special and that, while it may be mechanically possible for some of these apps to be replaced by third party apps, the non-neutral nature of the interface arguably seeks to dissuade users from changing the default.

In particular the fact that Safari has a separate and elevated position from other browsers. We believe that this undermines competition for browsers on iOS and should be fixed.

Apple could place the current default browser in a centralized location in a neutral manner.



DEFAULT BROWSER

[✓ Duck Duck Go]
Safari
Edge

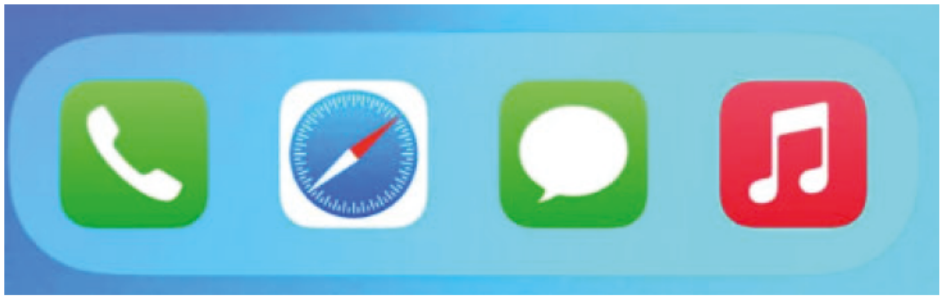
Possible Default Location and Default selection page for browsers

The goal of this remedy is to indicate to the user that while these apps are pre-installed by Apple, they are replaceable and have no elevated privileges over other third party apps. This will encourage the contestability of these services.

4.2. Being Default Browsers should grant the Hotseat

> *"2.11 Placement of browsers refers to the positioning of a browser on the mobile device, typically on the 'default home screen' of the device, and in many cases in the 'hotseat' on the home screen (centrally in the row of apps placed at the bottom of the home screen). Apps located in the 'hotseat' remain visible even when the user moves away from their default home screen to another screen on their device.*
>
> *2.12 Visual salience can be an important aspect of UI design, especially for user engagement. For example, positional bias in search results can influence how many clicks a result receives, even if the result is less relevant, with users primarily focussing their attention on the top three search results in a list.20 A 2022 report on Amazon consumer behaviour indicated that more than 30% of users frequently buy the first product listed in a search.*
>
> *2.13 Similarly, placement on the default home screen can focus user attention and minimise user effort to access applications they use frequently, requiring less navigation and creating inertia through the UI. Therefore, browsers that are placed on the home screen are likely to be more visually salient and accessible, influencing the users' likelihood of using the browser app.*
>
> *2.14 As with pre-installation, users may believe that the browser that is placed most prominently on their mobile device is endorsed or recommended by the mobile device manufacturer, and additionally they may be influenced by the status-quo effect and defer to the browser placed in the hotseat without ever taking an active decision."*

WP5 - The role of choice architecture on competition in the supply of mobile browsers



The hotseat on iOS's homescreen

The **hotseat** is any of that collection of app locations on the dock on the base of iOS homescreen. They have the advantage of always being shown regardless of which homescreen the user is on. Apple has set Safari to be in the hotseat by default, Chrome is set to be in hotseat by default on some but not all Android phones.

While users can change this setting manually by dragging the item out and another in, Apple and Google have added significant friction by not making this automatic.

> *"Only about half (52%) of people understand that their default browser is opened when they, for example, click on a link in an email or document.*
> *[..]*
> *over half (53%) also erroneously believed that their default browser would automatically be pinned to their task-bar."*

Mozilla - "Can browser choice screens be effective?" paper

Being selected as the default browser does not grant the hotseat. This means it is entirely possible, and in fact likely that many existing users who have set a third party browser as their default browser will still have Safari or Chrome in the hotseat.

We believe that setting the default browser should automatically place it in the hotseat if the system default set by the gatekeeper is in the hotseat. At a minimum the user should be prompted when they set a new default browser, if they would like to place it in the hotseat.

4.3. Operatings Systems should not pre-install Browsers

> *"Apple and Google's use of pre-installations and placement of browser apps on the device home screen means that users may be less aware of alternative browsers and less likely to make active choices between browsers. "*

Browsers and Cloud MIR - WP5

The working paper expresses concern that Apple and Google are undermining browser competition on iOS and Android via their pre-installation and prominent placement of Safari and Chrome.

We broadly agree with this assessment. User's choice of which browser they wish to use is the cornerstone of browser competition. We believe the ability of large browser vendors to either automatically set the default browser or to place their own browsers in prominent positions undermines this choice. This choice should be an active choice.

As discussed in the paper, the ways in which Apple and Google claim this market share without active user consent however is different.

Apple manages this by simple virtue of controlling the operating system, its app store and the default factory setup. As we have extensively argued, Apple also prevents third party browsers from porting their browsers to iOS via their 2.5.6 WebKit rule banning other browser engines, in effect forcing third party browser vendors to build shells around Safari's core.

On Android the picture is more complex but still heavily favors Chrome. On some phones, Google controls the factory setup. For other phones/tablets Android is used by OEMs (Original Equipment Manufacturer). Here Google uses a different tactic where it pays the OEMs to both preinstall Chrome and to prominently place it, though a complex set of Placement agreements (PAs) and Revenue sharing agreements (RSAs). Google paid Android manufacturers approximately £100-200 million in Search and Search/Chrome Activation Payments under PAs covering the UK in 2021. This behavior is extensively covered in the CMA's Mobile Ecosystems Studies - Appendix E: Google's agreements with device manufacturers and app developers.

> *"Google has in place, with certain Android manufacturers in respect of UK devices, voluntary commercial agreements. For example, it has PAs in place with certain manufacturers regarding the placement of the Google Search app and Chrome on Android devices and RSAs for respecting a number of placement and promotion requirements with respect to certain Google apps, including Google Search, Google Assistant and in some cases the Play Store. Both the PAs and the RSAs are only available to EMADA partners. More specifically:*
>
> - *Under the PAs, Google pays manufacturers 'activation payments' for each device on which they pre-install the Google Search or Google Search and Chrome apps and satisfy certain placement obligations for either (i) the Google Search app, or (ii) the Google Search and Chrome apps. If manufacturers pre-install and comply with the placement requirements in respect of Google Chrome in addition to Google Search they earn a substantially larger payment per device. Google told us that the placement obligations in the Placement Agreements are non-exclusive, and do not prevent rivals from being pre-installed or displayed prominently on the device.*
>
> - *Under its RSAs, Google pays some manufacturers a proportion of its net ad revenue from specific search access points on their devices in return for meeting a number of placement and promotion requirements, such as setting the Google Search app as the default search engine on all preloaded manufacturer browsers."*

These behaviors by both Apple and Google undermine mobile browser competition. Users are heavily pushed into accepting either Safari or Chrome as their default or most used browser without having made an active choice to select it as their primary browser. Most users rarely change browsers and their significant barriers to entry to browsers with low market share.

According to data in the WK1 on Android (in the UK) Chrome has a 77% market share and Samsung (also a factory default) has a 13% market share. On iOS (in the UK), Safari has an 88% market share.

What is needed is a mechanism that would prevent this behavior for at least new users of iOS and Android. This mechanism is simple. Operating system vendors should not be able to pre-install browsers. Rather on new devices the initial browser should exclusively be chosen though a choice screen.

This realigns choice back to the users and away from either the operating system or those with the biggest budget to pay to have their browser placed prominently. Browser vendors will instead have to compete to convince users to download and use their browser. This realigns incentives back to producing the browser with the best features, performance, stability, security and privacy.

While this remedy is limited in that it will only impact brand new users of Android or iOS, it will over time still have a significant impact.

Critics will likely argue that having a phone which in its initial setup lacks a browser undermines the users ability to use their phone without a 200MB download and that certain setup interfaces need to use the Web to function. They will also argue this is confusing to users and unfriendly. Finally, they may point to the fact we did not argue for this remedy in our previous paper to the DMA.

All of these claims are however without merit.

First, it is typical for devices to require regular system updates in the 100MBs.

Second, for setup devices can continue to make use of either SFSafariViewControl, WkWebView or the Android WebView if they need to display web content.

Third, this will not be confusing to users. The browser can simply be replaced with a neutral browser icon. During device setup, the user can be presented with an unskippable choice screen to select their browser. The chosen browser is then set as the user's default browser, placed in the hotseat (replacing the neutral icon) and queued to download. The users do not need to wait for it to download, they simply need to make their choice.

It is typical for users to be on WIFI when setting up and significant use of mobile internet as opposed to WIFI during setup is an edge case in the UK. It is worth mentioning that mobile internet is also excellent these days and downloading a browser over mobile internet will take an average of 50 seconds and will be 1.3% of low end user's monthly mobile data plan in the UK and between 0% and .2% of a typical monthly mobile data plan in the UK. This is only in the rare event that the user does not have access to WIFI.

The median mobile speed in the UK is 33 Mbps. This covers a broad range from the slowest speed was West Devon at 6.37 Mbps to the fastest of Wokingham at 267.4 Mbps. This means that the time to download a browser on mobile internet would be between 6 seconds and 4.2 minutes, with a median time of 50 seconds. Note that most users would be downloading on WIFI which is typically twice as fast. Mobile data plans appear to range from 15GB per month to unlimited, with a typical plan being either unlimited or 75GB.

Gatekeepers may reasonably argue: "what about users with exceptionally poor internet, who need to perform some vital task on the internet immediately on their phone." While this may be rare, even a 1/10,000 chance becomes a significant number of users when talking about new users for these platforms for the entirety of the UK. For these users, after they have selected their default browser on the choice screen, they may not have WIFI or sufficient internet to download a browser. In this case, a temporary unbranded browser could be provided by the operating system. To avoid granting the operating system any ability to undermine browser choice, the browser should have no branding, could simply be called browser, have a neutral browser icon and upon the user gaining sufficient internet to download their chosen browser, will be uninstalled.

It is worth stressing this is an edge case and few users will ever see or interact with this browser. For most users the new browser will be chosen and downloaded within less than 25 seconds.

Finally, we did not submit this proposal to the EU Commission to implement under the DMA, as the DMA has a provision stating that pre-installation shall not be for the purpose of the act viewed as self-preferencing. As such we were unable to suggest our preferred solution. The UK MIR team is however under no such restriction and has the power to deliver the most effective solution.

This remedy is proportionate is that it is the only remedy that removes the ability of large browser vendors to impose their browsers on users without active consent or to otherwise present their browsers in a non-neutral manner to end users.

4.4. Browser Choice Screen

The previous remedy requires a browser choice screen for new users of Android and iOS in order to be viable.

In addition to showing a browser choice screen to all new users, it is proportionate to show a browser choice screen to existing users upon purchasing a new device or device sync. The purpose of the choice screen is to minimize the advantage companies can gain by setting the default browser by factory settings.

Such a choice screen is already obligated by the Digital Markets Act, meaning the additional cost to activate it for the UK is minimal. Selecting a browser on this choice screen would install it and set it as default.

4.5. Browsers should be able to detect that they are the Default Browser on iOS

> *"However, several browser vendors have reported that they do not have visibility on iOS as to whether their browser is set as default and that this had significantly hindered their ability to effectively target prompts to the right users at the right time and makes it difficult for browser vendors to measure whether their prompts are effective."*
>
> Browsers and Cloud MIR - WP5

Working paper 5 also cites Apple as stating that they do not provide the ability to detect whether Safari is the default. While this is possibly true, it is also irrelevant. Apple has pre-installed Safari and pre-set it as the default. Safari has little need of such functionality and making it absent from all browsers including Safari serves to undermine users ability to switch browsers.

There is a clear argument that the inability of browser vendors to detect whether they are the default on iOS introduces friction in encouraging users to change default and thus allows Safari more ability to maintain its already considerable market share on iOS.

As such, Apple should provide an API which allows browsers to detect whether they are the default browser on iOS. This will allow third party browsers to have reasonable flows to prompt users to switch default browsers without bothering users which have already set them as the default.

4.6. Single Click Switch Default Browser Prompt

Safari does not provide a prompt to change the default browser. Nor as mentioned in the previous section does it allow browsers to detect whether they are the default browser.

On Android such a prompt is provided. However even this one is flawed in that it offers users a super choice in that rather than simply asking the user if they would like to make the just installed browser their default, it requires the user to once again select it again from a list. We would consider this unnecessary additional friction designed to reduce the number of users who change away from the default browser.

Browsers on iOS can link to their own settings page and provide a description on how to change browsers, but this is an awkward process. Additionally browsers do not know whether to display this to the user as they do not know if the user has already set them as their default browser.

Apple should provide a system prompt to browsers (with an option to never ask again, as is usual for all permission prompts) that allows browsers to prompt the user to one-click set it as the new default browser. This is standard on most operating systems.

Google should update the system prompt for Android to be simpler. An example replacement text upon installing say Vivaldi could be:

> *"Do you wish to set Vivaldi as your default browser?*
>
> *[Button 1]: Yes*
> *[Button 2]: Keep existing default browser"*

Example Prompt

This would be instead of the existing setup where the user must manually select Vivaldi out of the list of currently installed browsers.

4.7. Prompts from Non-Browser Apps or Websites

> *"Prompts can minimise user effort because they offer an easier route for switching browsers. However, by interrupting the user and nudging them to either switch or try an alternative browser, prompts can increase the burden on users and reverse a decision a user has made previously. Prompts may require users to take immediate action (known as 'forced action'). As a result, prompts may adversely impact the users' browsing experience and may lead them to accidentally making less effective choices.*

> *[...]*
>
> *On iOS, Google runs marketing campaigns to promote Chrome on Google's owned and operated native apps and websites. Google submitted that as Chrome is not set as the default on non-Android platforms, it engages in 'standard marketing practices' aimed at encouraging users to switch to Chrome."*
>
> Browsers and Cloud MIR - WP5

We view Google's tactics of prompting the user to install Chrome via Google Maps, Gmail and Google via custom functionality only available to Google as negative to browser competition. That is, owners of particularly massive platforms, including operating systems, app stores, maps, search engines and social media should not be using their control of these platforms to push users to use their own browser particularly via use of user interface designs that ignore or undermine the users choice of default browser. This serves simply to concentrate the browser market in the largest tech giants hands. As such the MIR team should consider prohibiting such behavior on Android and iOS.

4.8. Safari and Chrome should be Uninstallable

> *"The inability to uninstall Safari and Chrome further limits user control and choice over the customisation of their device, and could appear to create an implicit endorsement and self-preference Safari on iOS and Chrome on Android in comparison to other browsers."*
>
> Browsers and Cloud MIR - WP5

We agree with the assessment that making particular browsers impossible to uninstall signals to users that these are the preferred browsers for the operating system.

The WKWebView, SFSafariViewController, Android WebView and Android Custom Tabs should be treated as system components, and it should not be possible to uninstall them. These components are used in a wide variety of native apps. Many native apps use the webviews as a convenient way to render first/second party content and any remedy which would break these apps would be unreasonable, disproportionate and counter-productive. Other apps use SFSafariViewController or Android Custom Tabs when the user clicks on an external http/https link and should be allowed to continue to do so, provided Apple commits to making SFSafariViewController respect the user's choice of default browser.

We would also support it not being possible to uninstall the default browser until a new default browser has been selected. An appropriate and neutral error message should be displayed if the user attempts to do so.

As such both Safari on iOS and Chrome on Android should be uninstallable.

5. WebAPK Minting

> *"Overall, the evidence available to date indicates that Google engages in self-preferencing less, in respect of access to functionalities on Android compared to Apple's approach on iOS. Lack of access to WebAPKs, which is essential for installing PWAs, is the main issue highlighted by third parties (see paragraphs 4.6 to 4.7). Whilst Google has acknowledged this restriction, its latest submission to the CMA indicates that it is working to resolve it. Google has in some cases provided justifications for lack of access to functionality being provided or noted that it is working towards providing equal access."*
>
> Browsers and Cloud MIR - WP3

In 2015, Google shipped the first version of Web App Default Install Prompts. The initial implementation was limited and did not integrate well with Android. A follow-up implementation in 2017 fixed these flaws, allowing for deeper system integration and improved installation success rates. This system, known as "WebAPKs", relies on Google's infrastructure and Android system components and since launch has been restricted for use by Chrome, despite requests by other vendors including Mozilla for equal access. This caps the potential of Web Applications on Android and prevents competing browsers from benefiting from the engagement that comes from Web Applications, harming their chances in the market.

A WebAPK is a thin wrapper Native App that provides a splash screen, system launcher integration, and system settings configuration points. When launched, a WebAPK essentially starts a tab in the browser which installed the WebAPK, loading the specific URL the Web App developer configures.

All Native Apps on Android are distributed as APKs, either via the Google Play Store or via sideloading. WebAPKs allow Web Apps to be integrated into the OS for the purposes of discoverability, permissions management, shortcut creation, and uninstall.

Android's security model is built around signed native APKs. In order for Web Apps to integrate properly on Android without significant architectural changes, Web Apps need to be wrapped in a signed native APK. This allowed Google to support Web Apps across existing versions of Android without having to introduce a new architecture to support Web Apps and wait for years for it to be updated.

This strategy allows the operating system to treat Web Apps as Native Apps because, as far as the operating system is concerned, they are. WebAPKs minted by Google's servers are signed by the same keys as those of Native Apps distributed through the Play store. These keys are pre-trusted by the device as a condition of Google's licensing of Play. This also ensures that no side-loading warning messages are displayed.

It is worth noting, one large difference between the iOS and Android ecosystems is the relative prevalence of OS updates. While recent versions of Android from certain vendors have received slightly longer security patch periods, most Android users do not receive new versions of Android within a year of their release, and most Android devices only ever receive a single OS upgrade.

In essence, Google cannot upgrade Android devices that have already been sold, by adding new features to the next version of the OS. Google must rely on other mechanisms – like Play Services, also known as "GMS Core", to introduce new capabilities to the 2 billion+ Android devices in service around the world.

This is important as due to significant issues with Androids distribution via OEMs (original equipment manufacturers) resulting in the inability to patch Android, Google has been using GMS Core to deliver Android OS APIs to devices that would otherwise be un-updatable.

To install richer Web Apps, Chrome requests a package from a Google-run server using a Play Services API that is restricted to Chrome. The server then sends an Android Package (APK) for the Web App and Play Services installs it. This process is known as "WebAPK Minting", but since no other browser can access the minting API, the ability to properly install Web Apps is kept exclusive to Chrome (except Samsung Browser on Samsung devices).

At the bottom of this article (written by Google) it mentions the following:

> ***"I am a developer of another browser on Android, can I have this seamless install process?***
>
> *We are working on it. We are committed to making this available to all browsers on Android and we will have more details soon."*

This article was written May 21st 2017 **(more than 7 years ago)** but there has been no update from Google.

We are aware that multiple browser vendors requested WebAPK Minting but were rebuffed, including Samsung, Opera, Mozilla, Edge, Brave, and Kiwi Browser.

A more recent bug in the Chromium bug tracker, likewise, has remained open for multiple years without commitment from Google despite multiple browser vendors (Brave, Microsoft, Kiwi) commenting that they would benefit from access.

WebAPKs remaining exclusive to Chrome is anti-competitive, restricts browser competition and damages the viability of Web Apps because:

1. Other browsers can not compete to provide better functionality for Web Apps, for enabling better implementations of backup and restore (see below) or introducing Web App Stores.

2. Provides an unfair advantage to Chrome on Android through increased engagement from Web Apps.

3. Damages adoption of Web Apps by making them not work properly in other browsers.

Google publicly stated 7 years ago that they were working on sharing this functionality with other browsers but there has been no progress. We believe that they should be forced to share this functionality with other browsers. Google should be able to set strictly necessary, proportionate and justified security conditions attached to obtaining access to this API.

It is essential that Google allows competing browsers to properly install Web Apps. The simple, timely solution to this is for Google to make the existing Play Services API available to third-party browsers, allowing them to use the same WebAPK minting service that Chrome enjoys access to.

Google is already obligated under the Digital Markets Act to make this functionality available to third-party browsers. As such there is no additional cost for them to make such a change within the UK.

While we are pleased that Google has indicated to the MIR that they are working on sharing this functionality, given that this has been their posture of 7 years with no progress, we recommend that the MIR either force them to share this functionality or obtain legally binding commitments with set time-frames to do so.

Importantly, given this service is provided by Google Play Services, Google should be able to provide this on all existing Android devices (including ones not on the latest version of

Android). We believe it is reasonable that Google complete this change within the next 6 months.

6. In-App Browsers

We would like to thank the MIR team for investigating this difficult and technical topic.

6.1. Android

> *5.9 Google's policy on remote tabs does not appear to have a clear impact on competition between browsers because it is not preventing rivals from offering competing products. Further, while we note that Google linking its Search app with Chrome Custom Tabs may be resulting in advantages in terms of the latter's usage, we recognise there are benefits in allowing app developers such as Google to have some degree of choice over the IABs in their app. Therefore, based on the evidence we have seen so far, we do not consider this policy on remote tab IABs is likely to limit competition among mobile browsers on Android.*
>
> Browsers and Cloud MIR - WP4

Google's decision to override Android Custom Tabs standard behavior to invoke the default browser and instead to lock it to Chrome not only for google.com but for all subsequent urls is a clear undermining of not only the users choice of default browser but all browser competition on Android. The Android Google Search App is one of Android users primary entry points into the web which is now directed away from the browser users have selected as the default to browse the web. We believe Google should be compelled to respect the users choice of default browser and no longer lock the Android Google Search App to Chrome.

We know this is technically possible and we have observed versions of Android Google Search App in the wild that have been programmed to use the users default browser, no doubt in anticipation that regulators might compel such a change.

> *5.34 Overall, users have limited choice and control in relation to which browser is used for IAB implementations in native apps that they use and IAB in general because:*
> *(a) If a user's default browser is not Chrome, any settings they have for browsing in their dedicated browser might not be applied to in-app browsing on Android.*
> *(b) The in-app browsing visual interfaces on Android mimic the 'actual' Chrome browsing experience with some visual differences between the different IAB*

> *implementations. Consequently, users often struggle to distinguish whether they are using their default browser or an in-app browser.*
> *(c) Related to this, consumer research showed very low levels of awareness and limited understanding of in-app browsing among users, who frequently did not realise they were using an IAB instead of their chosen browser. This lack of awareness extends to the features they prefer, such as privacy and security settings.*
> *(d) Ultimately, users are generally not provided with the same level of controls over their in-app browsing experience, which can vary across different apps (eg the presence of an 'IAB menu'). Additionally, at the device setting level, users often cannot turn off IAB on Android.*
>
> Browsers and Cloud MIR - WP4

While there are multiple high profile exceptions such as various Meta Apps, the Android Google Search App and TikTok, we feel that it needs to be highlighted that for a significant percentage of in-app browsing on Android **does** respect the users choice of default browser via Android Custom Tabs default settings. That is for most apps in-app browsing does not negatively impact browser competition and does respect user choice.

What we would like to see is for all apps on Android to follow this model for third-party non-cooperating content. Simply compelling the biggest offenders Google, Meta and TikTok to respect the users choice of default browser would go a long way to resolving the issue on Android.

6.2. iOS

> ***4.14 Apple prevents all rival browser vendors from offering remote tab IABs on iOS.***
> *This means native apps cannot call on a browser other than SFSafariViewController for a remote tab implementation of in-app browsing. This is likely to be limiting rival browsers' ability to compete against Safari on iOS because they lack the functionality of displaying web content within an app. We understand browser vendors would be interested in offering a remote tab IAB on iOS to improve the quality of their offering and to better support their users.*
>
> Browsers and Cloud MIR - WP4

We agree that browsers on iOS should be able to support in-app browsing functionality. We believe that the user's choice of default browser should be respected in all contexts where the user is exploring the wider web.

We agree that browsers are currently being blocked from providing this functionality. We would go further and say the user's choice of browser is also being ignored. With this in mind we believe that SFSafariViewController should be updated to respect the users choice of default browser and for supporting browsers should use the users chosen default browser to power the remote tab IAB. This is already the case in most apps on Android via Android Custom Tabs, so the technicalities of implementing such a system are already well known.

> ***4.54 Overall, users have limited choice and control in relation to which browser is used for IAB implementations in native apps that they use and IAB in general because:***
>
> *(a) Where a user's default browser is not Safari, any default that a user has set for dedicated browsing is not applied for in-app browsing on iOS.*
>
> *(b) Apple's control over the design of IAB implementations, impacts how the visual interface is designed and configured on iOS (eg mimics 'actual' Safari browsing experience). This may result in users finding it difficult to understand when they are in their default browser and when they are in an in-app browser.*
>
> *(c) Linked to this, consumer research indicates that there is low awareness of IAB, in that users will not often be aware that they are in an IAB, or that their default browser, which they may have actively chosen because it offers certain features, is not being used for that purpose. As a result, users might not realise which browser features are not carried over (eg privacy and security features, or 'saved' password information). (d) Finally, users are often not offered the same levels of control over their in-app browsing experience. This can vary across apps (eg 'IAB choice menu') and also across device level settings (eg users mostly cannot turn off IAB on iOS devices).*
>
> Browsers and Cloud MIR - WP4

We agree with the MIR team that the user's choice of default browser is being ignored in the context of in-app browsers. We also agree that the design of the user interface of in-app browsers makes it hard for users to be aware of this. Finally we agree users are offered very little control of their in-app browser experience and can not turn off IAB in iOS, or we might add opt to lock it to using their default browser.

Importantly, unlike Android, iOS does not have an equivalent of Android Custom Tabs which invokes the user's chosen default browser. We believe the solution here is simple, compel Apple to update SFSafariViewController to (if supported by the users default browser) invoke the users chosen default browser when users browse the web in an in-app browser.

6.3. Our Paper

In our document *"OWA - DMA Interventions - In-App Browsers"* (directed at the EU's Digital Markets Act, but relevant here) we proposed the following remedies to fix the issue:

1. Designated Core Platform Services should respect the users choice of default browser.

2. App store rules must mandate non-browser apps use the user's chosen default browser for http/https links to third-party websites/Web Apps.

3. Apple must update SFSafariViewController to respect the user's choice of default browser.

4. Third-party businesses must be provided an explicit and effective technical opt-out from non-default in-app browsers.

5. OSes must provide a global user opt-out. Apps must also request explicit permission from users.

6. Google must remove the ability to override the users choice of default browser via Android Custom Tabs.

Users' choice of default browser and the consequent competition is only effective if that choice is respected.

7. Other Jurisdictions

7.1. Apple's response to the EU's DMA

The EU Digital Markets Act came into force earlier this year. The act is broad and have many requirements but the two most important ones for the Web are:

> ***The gatekeeper shall not require end users to use, or business users to use***, *to offer, or to interoperate with, an identification service,* ***a web browser engine*** *or a payment service, or technical services that support the provision of payment services, such as payment systems for in-app purchases,* ***of that gatekeeper in the context of services provided by the business users using that gatekeeper's core platform services****.*
>
> Digital Markets Act - Article 5(7)

> ***The gatekeeper shall allow providers of services*** *and providers of hardware,* ***free of charge****,* ***effective interoperability with, and access for the purposes of interoperability to, the same hardware and software features accessed or controlled via the operating system*** *or virtual assistant listed in the designation decision pursuant to Article 3(9)* ***as are available to services or hardware provided by the gatekeeper****. Furthermore, the gatekeeper shall allow business users and alternative providers of services provided together with, or in support of, core platform services, free of charge, effective interoperability with, and access for the purposes of interoperability to, the same operating system, hardware or software features, regardless of whether those features are part of the operating system, as are available to, or used by, that gatekeeper when providing such services.*
>
> *The gatekeeper shall not be prevented from* ***taking strictly necessary and proportionate measures to ensure that interoperability does not compromise the integrity of the operating system****, virtual assistant, hardware or software features provided by the gatekeeper,* ***provided that such measures are duly justified by the gatekeeper****.*
>
> Digital Markets Act - Article 6(7)

Interestingly the specific reason cited in the act to allow browser engine competition is to prevent gatekeepers from dictating the speed, stability, compatibility and feature set of "web software applications".

[redacted] The number of ways that Apple is not complying is so myriad that we have attempted to exhaustively catalog them.

We have requested the commision open a proceeding into Apple to investigate this alleged non-compliance.

Specifically we have asked for the following remedies:

- Restricting Apple's API contract for browsers down to strictly necessary, proportionate and justified security measures.
- Make clear what the security measures are for third party browsers using their own engine by publishing them in a single up-to-date document.
- Removing any App Store rule that would prevent third party browsers from competing fairly.
- Allow browser vendors to keep their existing EU consumers when switching to use their own engine.
- Removing the special placement of Safari.
- Making Safari uninstallable.
- Implementing Install Prompts in iOS Safari for Web Apps.
- Allowing Browser Vendors and Developers to be able to test their browsers and web software outside the EU.
- Allowing Browsers using their own engine to install and manage Web Apps.
- Make notarization a fast and automatic process, as on macOS.
- Allow direct browser installation independently from Apple’s app store.
- Allow users to switch to different distribution methods of a native app and allow developers to promote that option to the user.
- Don't break third party browsers for EU residents who are traveling.
- Opt-Into Rights contract should be removed.

- Core Technology Fee should be removed.
- Apple should publish a new more detailed compliance plan.

Apple is obligated under Articles 5(7), 6(3), 6(4) and 6(7) to fix each of the above issues. Apple has failed to achieve effective compliance with these obligations contrary to Article 13(3). Further Apple has taken numerous and significant steps that obstruct and undermine it in contravention of Article 13(4). Apple has introduced conditions and restrictions on DMA-conferred rights that have no legal basis in the DMA and gone far beyond the restrictions that the DMA does allow by introducing rules that have no basis in security or that are not justified, strictly necessary or proportionate.

We have urged the Commission to enforce the DMA and obligate Apple to allow browsers and Web Apps to compete fairly and effectively on their mobile ecosystem. This will unlock contestability, fairness and interoperability. Companies will then have to compete for users on merit, not via lock-in or control over operating systems. Consumers will benefit from choice, better quality and cheaper software, interoperability, and the genuine ability to multihome across devices and operating systems offered by different companies.

We believe that Apple's interactions with the DMA in relation to browsers and Web Apps should be of great interest to the MIR team. [redacted]

It is also important that any remedy allows browser vendors to ship a single browser with their own engine that works in both the UK and the EU.

We have written about this extensively in our paper "Apple DMA Review".

7.2. Japan

Japan has also explicitly in law prohibited large operating systems vendors from banning browser vendors from using their own engine. This is relevant to the UK in that it means that there are now two large jurisdictions in which Apple's behavior is now prohibited.

Given the unlikelihood of Apple withdrawing from those markets, this means that Apple will need to comply with this legislation. This means that Apple faces no additional costs for complying in the UK. This is important in any proportionally argument that the MIR team might face in compelling Apple to take particular measures, as the net of many of those measures is zero.

One concern is that Apple might attempt to diverge compliance strategies such that browser vendors are forced to ship one browser using the WkWebView, one browser for the EU, one browser for Japan and one browser for the UK. This will raise costs considerably and cause immense user confusion.

Ideally Apple should be forced to allow browsers to compete globally on iOS with their own engines. If that is not possible regulators should seek to allow browser vendors to ship a single version of their browser that works in all jurisdictions that Apple has been forced to allow browser vendors to use their own engines. This would allow other regulators around the world to simply compel Apple to add their country to that list.

7.3. Australia

Two years ago the Competition & Consumer Commission (ACCC) published a report outlining competition issues in digital markets. The report contains a number of considered remedies including:

> *"The code of conduct for mobile OS services could require Designated Digital Platforms to allow third-party browser engines to be used on their mobile OS. This could allow third-party providers of browsers and web apps to compete on their merits."*
>
> *"We also consider that the additional competition measures should include the ability to provide third-party providers of apps and services with reasonable and equivalent access to hardware, software and functionality through their mobile OS."*
> Digital Platform Services Inquiry - September 2022 Interim Report

Currently the Australian government is drafting legislation. The laws will be based on the ACCC recommendations in their 5th report. It specifically calls for meaningful browser choice, including engine competition.

7.4. DOJ vs Apple

The United States Department of Justice (DOJ) is taking Apple to court. In its lengthy complaint, the role of middleware, such as browsers, and interoperability is brought up frequently. Apple's undermining of Web Apps via its prohibition on third-party browser engines is explicitly acknowledged. While the DOJs case is broad, we believe that browsers and Web Apps fit the DOJs case perfectly. The case is likely to take the next 3-5 years. As such any report or intervention by the CMA, the EU Commision or Japan's JTC will likely be impactful. Having browser competition be thriving in other countries on iOS but not in the USA will be a compelling argument to the US courts to force Apple to change its behavior in the US, its home and largest market.

> *"**Apple has long understood how middleware can help promote competition** and its myriad benefits, including increased innovation and output, **by increasing scale and interoperability**.*
> *[...]*
> *In the context of smartphones, examples of **middleware include internet browsers**, internet or cloud-based apps, super apps, and smartwatches, among other products and services.*
> *[...]*
> ***Apple has limited the capabilities of third-party iOS web browsers, including by requiring that they use Apple's browser engine, WebKit.***
> *[...]*
> *Apple has sole discretion to review and approve all apps and app updates. **Apple selectively exercises that discretion to its own benefit**, deviating from or changing its guidelines when it suits Apple's interests and allowing Apple executives to control app reviews and decide whether to approve individual apps or updates. Apple often enforces its App Store rules arbitrarily. **And it frequently uses App Store rules and restrictions to penalize and restrict developers that take advantage of technologies that threaten to disrupt, disintermediate, compete with, or erode Apple's monopoly power.**"*
>
> DOJ Complaint against Apple

8. Toward A Brighter Future

OWA believes that the Web's unmatched track record of safely providing frictionless access to information and services has demonstrated that it can enable a more vibrant digital ecosystem. The web's open, interoperable, standards-based nature creates an inclusive environment that fosters competition, delivering the benefits of technology to users more effectively and reliably than any closed ecosystem.

OWA's goal is to ensure that browser competition is carried out under fair terms, that user choice in browsers matters, and that web applications are provided equal access and rights necessary to safely contest the market for digital services.

The MIR team has a critical opportunity to fix key issues that have undermined both browser and Web App competition for over a decade to the benefit of both UK consumers and UK businesses. This will improve interoperability, contestability, and fairness leading to lower priced and higher quality apps— not only for the UK but for the entire world.

OWA believes competition, not walled gardens, leads to the brightest future for consumers, businesses, and the digital ecosystem.

9. Open Web Advocacy

Open Web Advocacy is a not-for-profit organization made up of a loose group of software engineers from all over the world, who work for many different companies and have come together to fight for the future of the open web by providing regulators, legislators and policy makers the intricate technical details that they need to understand the major anti-competitive issues in our industry and potential ways to solve them.

It should be noted that all the authors and reviewers of this document are software engineers and not economists, lawyers or regulatory experts. The aim is to explain the current situation, outline the specific problems, how this affects consumers and suggest potential regulatory remedies.

This is a grassroots effort by software engineers as individuals and not on behalf of their employers or any of the browser vendors.

We are available to regulators, legislators and policy makers for presentations/Q&A and we can provide expert technical analysis on topics in this area.

For those who would like to help or join us in fighting for a free and open future for the web, please contact us at:

Email	contactus@open-web-advocacy.org
Web / Web	https://open-web-advocacy.org
Mastodon	@owa@mastodon.social
Twitter / X	@OpenWebAdvocacy
LinkedIn	https://www.linkedin.com/company/open-web-advocacy