Vivaldi Technologies AS, Mølleparken 6, Oslo, Norway

28 August 2024

Comments on Mobile Browsers and Cloud Gaming Market Investigation WP7: Potential Remedies

Contents:

The following is Vivaldi's commentary on the WP7: Potential Remedies paper dated 8 August 2024. **Please contact us as usual if you wish to quote some of it so we can redact any company confidential information.**

Vivaldi, launched in 2016, is a powerful, personal & private web browser (for desktop, mobile and in-car) that adapts to its users and offers more features than any other modern browser.

Vivaldi's has two ground rules: privacy is a default, and everything's an option. In practice, this means building software that protects users' privacy but also does not track how they use it. Vivaldi believes private and secure software should be the rule, not the exception.

Vivaldi is headquartered in Norway, with satellite offices in Iceland and USA. It has no external investors and is co-owned by its approximately 50 employees.

There are currently 3,100,000 active users world-wide, [REDACTED] of whom are in UK (as are [REDACTED] employees).

Comments on section 5 (browser engine restrictions)

Vivaldi exists to give users access to the Web in a way that they control, while protecting them as far as possible from surveillance and other bad actors.

Leaders in the organisation have many years of experience in ensuring that the web service is available across devices, operating systems, constrained networks and hardware.

Except in some very small niches (some gaming, systems very tightly coupled to specific hardware features) we believe the web, built on mature open technologies, should be the delivery mechanism for software, rather than some vendor-controlled proprietary technology. Consumers should be free to access that software on whichever browser and device they prefer.

This informs our comments. We welcome the CMA's proposed remedies but do not believe they go far enough to ensure that Web Applications can be a viable competitor to single-platform "native" apps, especially on iOS.

Apple must allow browser engines access to all device features

CMA wrote in the opening statements of the Market Investigation Reference into Browsers and Cloud Gaming https://www.gov.uk/government/news/cma-plans-market-investigation-into-mobile-browsers-and-cloud-gaming

> “We all rely on browsers to use the internet on our phones, and the 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.”

Allowing competing browser engines, like Firefox’s Gecko and Chromium (in Vivaldi, Chrome, Edge, Opera et al) is welcome, and vital. But that is not the goal; the goal is making web apps competitive with “native” apps, and the CMA does not go far enough because it only plans to mandate that Apple “grant equivalent access to APIs used by WebKit and Safari to browsers using alternative browser engines”.

The problem here is that Apple has long denied Safari access to some system features that “native” apps on iOS can access. For years, for example, native apps could send push notifications but Safari couldn’t. That recently changed, although the web push implementation is still reportedly buggy (https://webventures.rejh.nl/blog/2024/web-push-ios-one-year/). But still on iOS, Apple refuses to allow the web to access Bluetooth, USB etc, even though “native” apps can access these device capabilities.

Apple says that this is for security, but we note that Apple themselves told the CMA (https://assets.publishing.service.gov.uk/media/62277271d3bf7f158779fe39/Apple_11.3.22.pdf) that

> “WebKit’s sandbox profile on iOS is orders of magnitude more stringent than the sandbox for native iOS apps”

So Apple’s web browser, according to Apple, is more secure than Apple’s own iOS native sandbox. Yet Apple has a worse track record than Firefox or Google in shipping patches for browser vulnerabilities (https://googleprojectzero.blogspot.com/2022/02/a-walk-through-project-zero-metrics.html).

The CMA wrote (https://assets.publishing.service.gov.uk/media/667d2f0caec8650b100900c0/WP2_-_The_requirement_for_browsers_operating_on_iOS_devices_to_use_Apple_s_WebKit_browser_engine_1.pdf)

> “it is not clear from the evidence available to date that WebKit has better security outcomes compared to other browser engines”

If Apple itself does not believe that native iOS apps are more secure than browser sandboxes, why do they keep some device features away from

Safari? It's fair to think that Apple wants to force businesses to make native iOS apps instead of Web Apps, so that distribution and monetisation is therefore controlled by Apple.

The missing remedy

To ensure that Apple cannot artificially hamstring all third party browsers, the CMA should mandate that all APIs and device integrations available to native iOS Apps, Apple's own apps and services must be available to third party browser engines.

Apple must implement Web App install prompts

Apple likes to tell regulators that web apps are an alternative to native iOS apps in its App Store. In a submission to the Australian regulator (https://www.accc.gov.au/system/files/Apple%20Pty%20Limited%20%2810%20February%202021%29.pdf) Apple wrote

> 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. PWAs are increasingly available for and through mobile-based browsers and devices, including on iOS. PWAs are apps that are built using common web technology like HTML 5, but have the look, feel and functionality of a native app.

Apple's statement is not entirely accurate. Here's an example of a PWA, Stuart Langridge's Farmbound game(https://www.kryogenix.org/farmbound/), rendered by Chrome on Android, the OS manufacturer's provided browser (the lower part of the screen is cropped):

Chrome has seen that Farmbound is an installable Progressive Web App and generated the install banner seen ‘floating’ over the game (and which we’ve highlighted with a red box). Tapping it adds the game’s icon to your Android homescreen. The author didn’t need to do anything other than code the game correctly, and add a line in his HTML pointing to a manifest file:

```
<link rel="manifest" href=“manifest.json">
```

By contrast, Safari, the default browser on iOS, does not prompt a user to install a Progressive Web App. Instead, the user must perform the following ritual:

- Press the share button. In this case, to install; no sharing involved here.
- Scroll down below the share menu’s fold to discover the ‘add to homescreen’ option, assuming a user knows that this exists
- Click ‘add to homescreen’
- Decide to accept the app’s name, or replace it
- Click ‘add’

However, Apple does offer an analogous one-tap installation mechanism for apps, **but it only works for native iOS apps that are in the Apple App Store**.

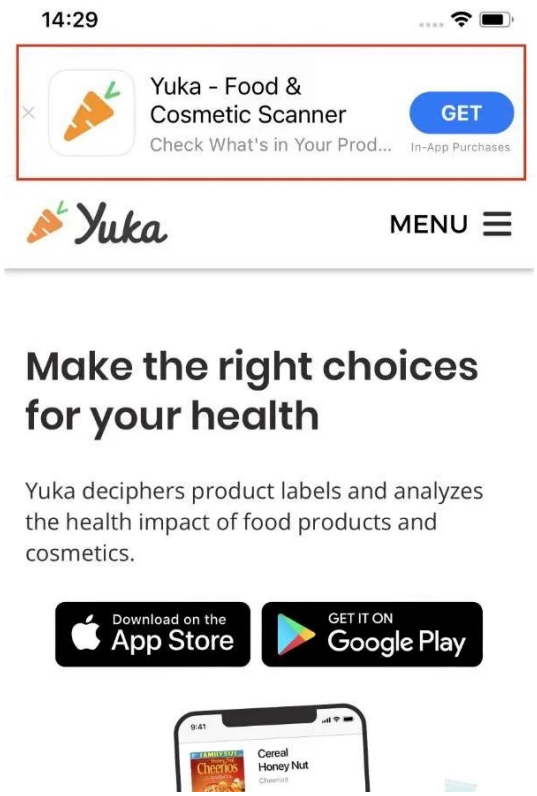
A web site owner can add a single line of HTML, much like the one linking to a PWA manifest, but this one pointing to the app in the App Store associated with the website:

```
<meta name="apple-itunes-app" content="app-
id=myAppStoreID, app-argument=myURL">
```

Apple writes that its proprietary Smart App Banners (https://developer.apple.com/documentation/webkit/promoting_apps_with_smart_app_banners) allow site owners to

> “create a banner to promote your app on the App Store from a website”.

Here's an example from yuka.io (https://yuka.io/), with the Safari-generated banner highlighted by us in a red box (the lower portion of the screenshot is cropped):



Apple says

> "If the app is already installed on a user's device, the Smart App Banner intelligently changes its action, and tapping the banner simply opens the app. If the user doesn't have your app on their device, tapping the banner takes them to the app's entry in the App Store … Smart App Banners automatically determine whether the user's device supports your app. If it doesn't, or if your app is unavailable in the user's location, the banner doesn't appear."

This is very different from the clunky precess to install a PWA on iOS, and cast doubts on the accuracy Apple's claim to the Australian regulator that PWAs "have the look, feel and functionality of a native app".

Close inspection of the Smart App Banner in the yuka.io screenshot above reveals that below the 'Get' button, in a small grey font against a white background, the words "In-App Purchases".

Apple makes it easy to install a native iOS App from a webpage; a native app in the Apple App Store, from which Apple taxes 30% of in-app purchases. It earns nothing from PWAs because they are distributed from the owners' sites, not an intermediary gatekeeper's App Store.

The missing remedy

Apple should not be allowed to continue self-preferencing native Apps over Web Apps.

5.32 of CMA WP7 says

> "Apple would be required to eliminate its use of private APIs for WebKit and Safari without degrading currently available functionality made available for WebKit and Safari"

We do not believe Smart App Banners should be removed, but comparable functionality should be *added* to Safari for PWAs.

Vivaldi recommends removing any doubt, and asks that CMA explicitly requires Apple to **implement install prompts for PWAs in Safari/iOS**. It doesn't have to be implemented in them the same way as Chrome, just with the same outcome.

Web Apps must open in the browser that installed them

The current suggested remedies allow for third-party browser engines on iOS, and those will be able to download PWAs. However, none of the remedies explicitly requires that those PWAs will subsequently run in the engine that downloaded them.

Apple could therefore argue that it could fulfil CMA's remedies by allowing browsers to use their own engine and providing them access to the share menu to install Apple's WebKit implementation of Web Apps.

This is actually the current state of affairs in the EU; Apple tried to completely kill PWAs (which it calls 'Homescreen Apps') in Safari, so that it wouldn't have to allow them in other engines. They backed down after a campaign by EU web developers (https://open-web-advocacy.org/blog/apple-backs-off-killing-web-apps/), but (so far, at least) all PWAs on iOS will continue to be hamstrung by running in Webkit. Apple said (https://developer.apple.com/support/dma-and-apps-in-the-eu#8)

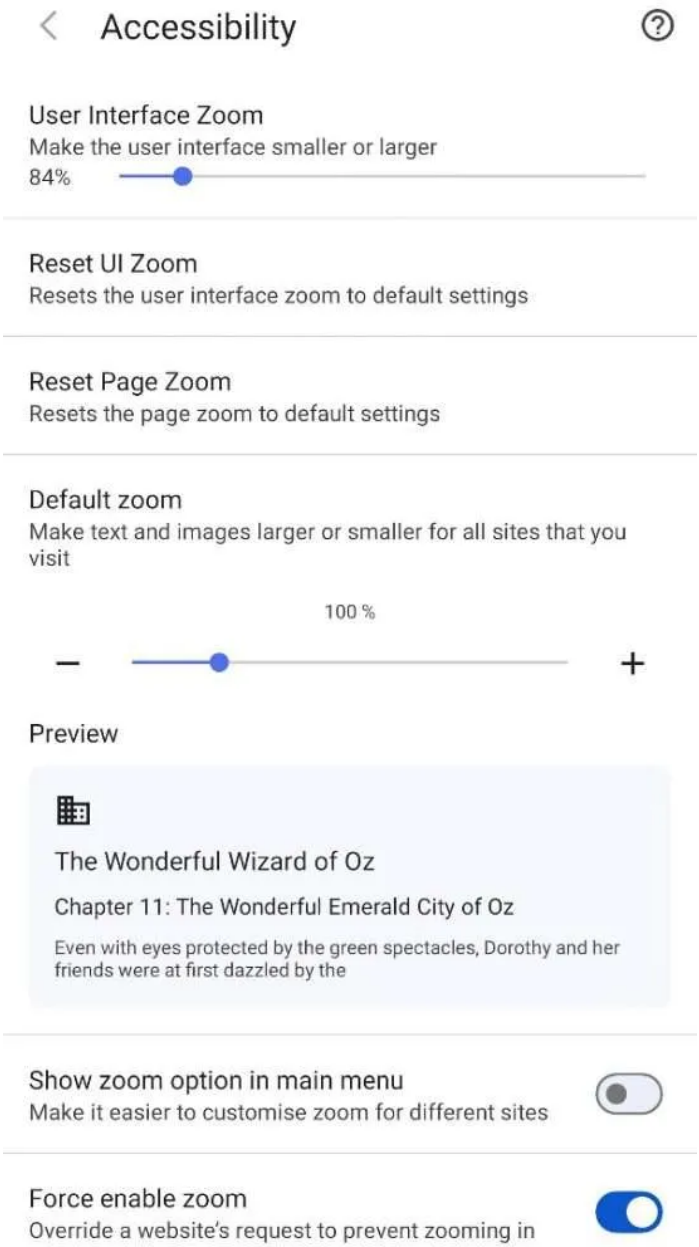
> “we will continue to offer the existing Home Screen web apps capability in the EU. This support means Home Screen web apps continue to be built directly on WebKit and its security architecture, and align with the security and privacy model for native apps on iOS.”

CMA is aware of this problem. In the *interim report in connection with our market study into mobile ecosystems* (https://www.gov.uk/government/publications/mobile-ecosystems-market-study-interim-report), they wrote

> “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.”

Allowing third-party browsers to install PWAs but not subsequently run them is user-hostile. The browser that installs a PWA would almost certainly be the user’s default browser, and (on a personal device rather than, say, a locked-down corporate laptop) people choose their default browser for a reason.

A disabled user of Vivaldi browser prefers to browse with some accessibility settings that they can tweak as necessary according to the severity of symptoms of a disability:



Another person will use something else, with different settings. But whatever their choice is, we believe it should be respected if you open up a PWA, and believe that the CMA should make this explicit.

It would be simply weird and user-hostile if default browser settings worked fine while someone played Farmbound on its website, but didn't work if they opened the Farmbound PWA because, *unknown to them*, it was running in WebKit.

The missing remedy

The CMA should require that PWAs downloaded in one browser are subsequently run in that browser, running its own engine, with its user settings.

Otherwise, we could find ourselves in a situation where Vivaldi could ship Vivaldi on iOS, powered by Chromium; therefore Chromium could be someone's default in-app browser, but we couldn't implement Web Bluetooth in Vivaldi (because Safari doesn't have Bluetooth access, so Vivaldi couldn't have it either) and the user couldn't have add a Web App to Home Screen that opens with their chosen accessibility settings, because Apple might/would mandate that add to Home Screen would add it as Safari).

We do not believe this is a future that CMA or consumers want.

Comments on section 7 (Apple's and Google's choice architecture practices)

Vivaldi believes that the browser choice screen should be shown at device set-up and when new versions of operating systems are installed. This is when consumers generally expect to be asked questions about the environment.

We think this is a more natural time for a customer to be asked such questions (they are already allocating time for setting up/ upgrading their device, and expect configuration questions).

The alternative approach taken in EU (of asking users on first run of the default browser) is more intrusive to the user, because they are attempting to browse the web, potentially urgently. The user will also believe they have completed device setup so feel their task at hand is being interrupted and are therefore more likely to 'skip' the choice by confirming the default.

A 'choice screen at first browse' scenario also gives the incumbent an unfair advantage, because it cements the idea that the manufacturer has provided a default that will therefore be "good enough".

Option C5

When the new regulatory regime comes into effect, existing smartphone users will need to see the choice screen too. Because the changes required

in iOS and Android will necessitate the manufacturers to release a new version of the software, the operating system update process should display the choice screen.

Option C6

Vivaldi agrees with the suggested remedy of single centralised location in the settings for changing default browser, regardless of what browser is currently set as default and that the user journey for changing default browser should be identical regardless of which browser is set as default.

Option C7

A requirement for Apple and Google to share user data on default browsers settings with browser vendors.
It's imperative that each vendor receives timely information from Apple and Google that, at a minimum, allows the vendor to determine what proportion of choice screen views resulted in selection of its product. The data must be sufficiently granular (ideally, daily) so that trends can be identified (for example, to cross-reference with marketing campaigns etc)

Assuming the choice screen randomises the position/ order of the various browsers, the statistics should be broken down so that vendors can satisfy themselves that the placements are truly random.

If this is offered via an API, it should be well-documented.

Option C8

Vivaldi agrees with the CMA that "limiting the frequency of prompts would seek to level the playing field for other providers, ensuring that neither Apple nor Google can leverage their control of their respective operating systems to self-preference in relation to prompts and notifications regarding default browser status", and also that "using prompts is an important tool for third-party vendors as it is one of the main mechanisms through which they can obtain a foothold in the market.

Option C9

"A requirement for Apple and Google to allow users to uninstall Safari browser app on iOS and Chrome on Android devices"

Vivaldi agrees that users must be able to uninstall browsers they no longer want. Users on low-specification, low-storage devices could be put off from installing a third-party browser if they knew they were further curtailing their

storage space by causing an unused, unwanted app to remain on their device.

Storage space remains a problem in the world of mobile, especially for the lower-end devices that make up the long tail of the market:

> "All in all, we didn't stand a chance as we were fighting with both our competitors and other apps for a few more MB of room inside people's phones"
>
> *Inside Birdly: Why you shouldn't bother creating a mobile app. https://medium.com/inside-birdly/why-you-shouldn-t-bother-creating-a-mobile-app-328af62fe0e5#.ufoave1x4*

Candidate selection for choice screen

Vivaldi believes that the selection of candidates for the choice screen is of paramount importance.

1. Candidates for the choice screen should be general-purpose web browsers aimed at end-users.

   We believe that Apple has attempted to erode trust in the EU's choice screen and exclude other competitors by deliberately including browsers that aren't useful to end-users, [REDACTED]

2. Cross-platform browsers should take priority. If it is a browser that is available on all platforms, it is more likely to be a major competitor.

3. Browsers that contain their own compiled code should take priority as a candidate. If their vendors compile the code themselves (rather than simply wrap a third party's core), they are more likely to receive quick security and privacy updates.

4. Browsers that are updated frequently should take priority, as they are more likely to receive quick security and privacy updates.

5. In the case of Android, the only OEM browser that should be a candidate is that of the manufacturer of the device.

Design and functionality of choice screen

We also believe that the design and functionality of the choice screen is of vital importance in reducing Apple and Google's ability as controller of the

Operating System to self-preference their own browsers. Therefore, the order of display should be randomised each time the screen is displayed.

Selection should only be possible once the user has scrolled through all the choices, so that users whose font size is large for accessibility reasons are aware of all the options. The choice screen should be very obviously scrollable if there are options “below the fold”.

While not part of a choice screen, an additional ‘choice’ if a user installs a third-party general-purpose browser (e.g., a choice screen candidate as listed above) but has not set it as default replacing Safari on iOS, or Chrome on Android would educate users and offer more competitive opportunity.

So if, for example, a user installs Vivaldi on iOS but did not know how to set it as default, when default Safari is activated, a system message saying “You downloaded Vivaldi; do you want to set that as default?” with a yes/ no button could be shown (with ‘do not ask again’ choice).

END