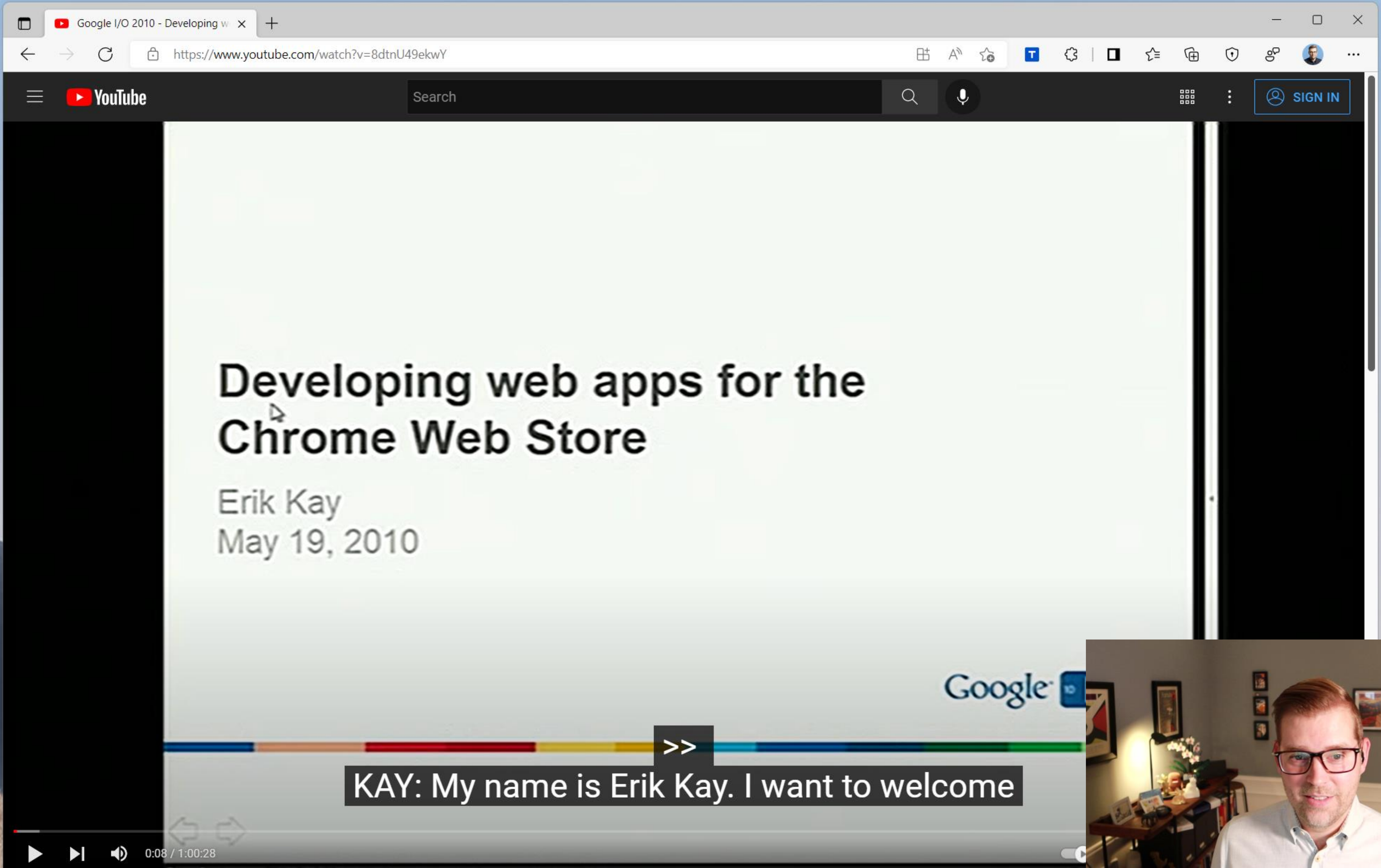
Microsoft Edge

Microsoft

PWAs & MiniApps

Alex Russell <alexrussell@microsoft.com>
infrequently.org
@slightlylate

PWA Dev Day, August 2022
https://pwadev.io/



Google I/O 2010 - Developing w

https://www.youtube.com/watch?v=8dtnU49ekwY

Developing web apps for the Chrome Web Store

Erik Kay
May 19, 2010

KAY: My name is Erik Kay. I want to welcome

APACHE
CORDOVA™

A Gnawing Question:

What Was The Web Missing?

- Offline
- Device APIs
- Push Notifications
- High-power compute
- Access to storage & files
- First-class UI & OS integration
- …?

Radical Distribution Simplicity:

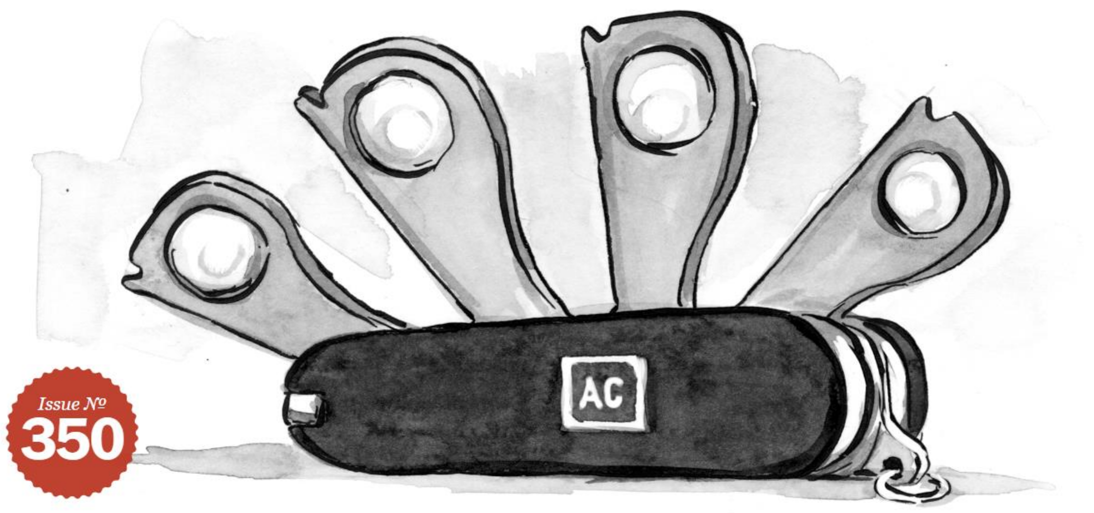
```
<a href="https://...">...</a>
```

Application Cache is a Doucheba

https://alistapart.com/article/application-cache-is-a-douchebag/

ARTICLES EVENTS TOPICS WRITE FOR US LANGUAGE (ENGLISH)

A LIST APART



Application Cache is a Douchebag

by **Jake Archibald** · May 08, 2012

Published in Application Development, HTML, JavaScript

Good morning! Over in "castle Lanyrd" we recently launched our mobile site, which caches data on events you're attending for viewing offline. I've boiled the offline bits down to a simple demo and posted all the code on Github. But before we delve into the code, let me tell you a true story. Totally true.

Northwestern
INFORMATION DESIGN AND STRATEGY
School of Professional Studies

Northwestern's Online MS in

I was at a party, one where the guests were mostly strangers to one another. I was part of a little huddle that was awkwardly trying to make introductions. A rather pretty lady turned to one of the shyer members of the group, introduced herself as "Dev," and asked "So, what do you do then?"

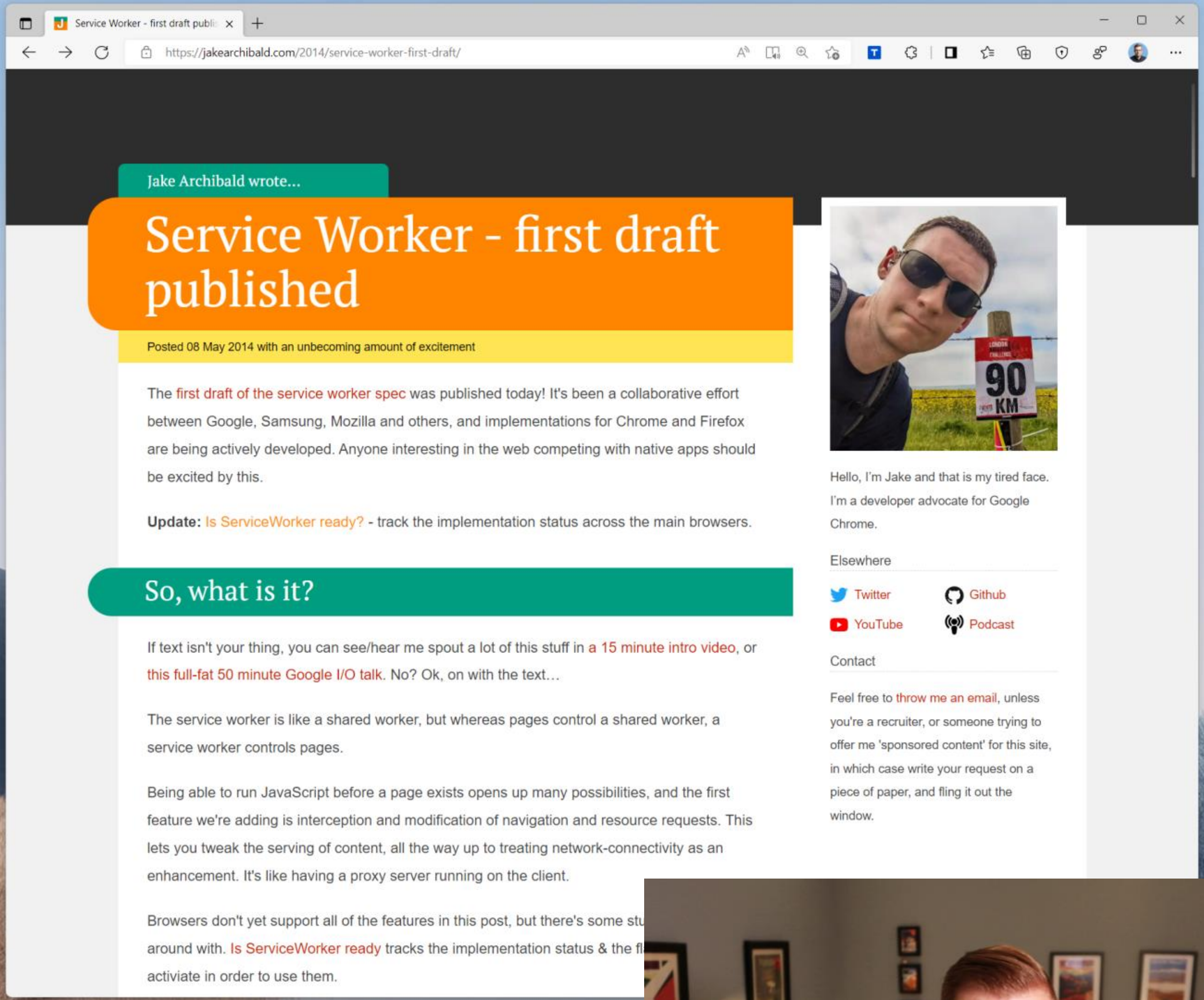
48 Comments

Share this:

Microsoft

Fast Forward: 2014

- ☑ Developer interest & feeback
- ☑ Cross-browser collaboration
- ☑ Working implementations



Service Worker - first draft publi

https://jakearchibald.com/2014/service-worker-first-draft/

Jake Archibald wrote…

Service Worker - first draft published

Posted 08 May 2014 with an unbecoming amount of excitement

The first draft of the service worker spec was published today! It's been a collaborative effort between Google, Samsung, Mozilla and others, and implementations for Chrome and Firefox are being actively developed. Anyone interesting in the web competing with native apps should be excited by this.

Update: Is ServiceWorker ready? - track the implementation status across the main browsers.

So, what is it?

If text isn't your thing, you can see/hear me spout a lot of this stuff in a 15 minute intro video, or this full-fat 50 minute Google I/O talk. No? Ok, on with the text…

The service worker is like a shared worker, but whereas pages control a shared worker, a service worker controls pages.

Being able to run JavaScript before a page exists opens up many possibilities, and the first feature we're adding is interception and modification of navigation and resource requests. This lets you tweak the serving of content, all the way up to treating network-connectivity as an enhancement. It's like having a proxy server running on the client.

Browsers don't yet support all of the features in this post, but there's some stu

around with. Is ServiceWorker ready tracks the implementation status & the fl

activiate in order to use them.

Hello, I'm Jake and that is my tired face. I'm a developer advocate for Google Chrome.

Elsewhere

Twitter Github

YouTube Podcast

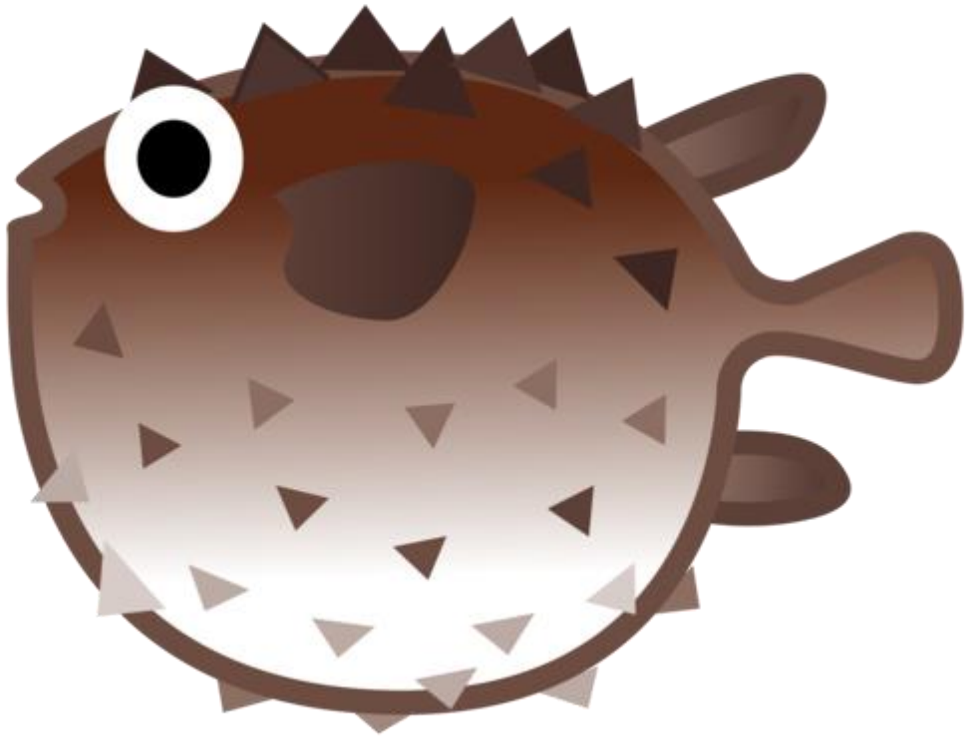
Contact

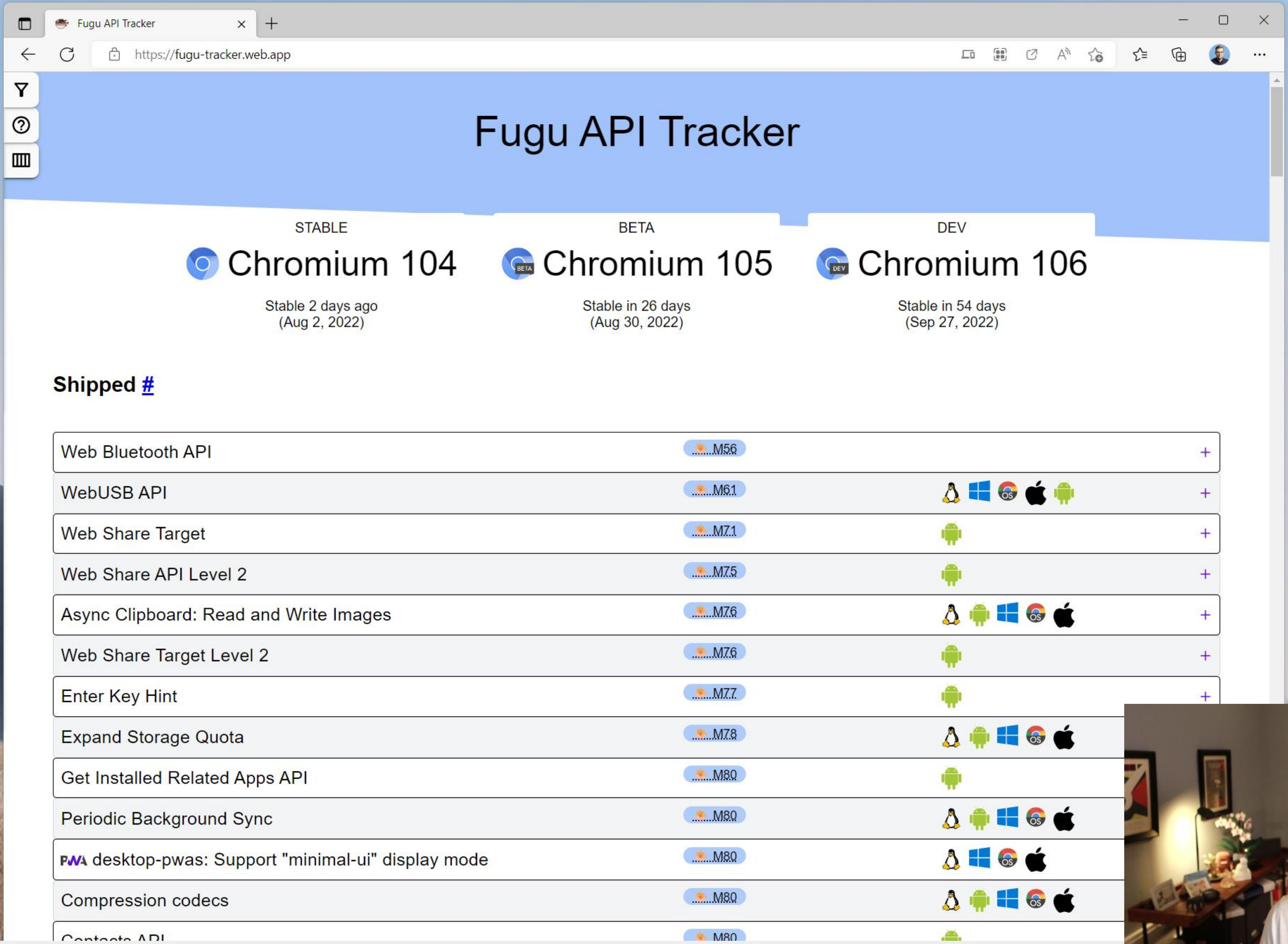
Feel free to throw me an email, unless you're a recruiter, or someone trying to offer me 'sponsored content' for this site, in which case write your request on a piece of paper, and fling it out the window.

PWA

What Was The Web Missing?

- ~~Offline~~
- Device APIs
- ~~Push Notifications~~
- High-power compute
- Access to storage & files
- ~~First-class UI & OS integration~~
- ...?





Fugu API Tracker

https://fugu-tracker.web.app

Fugu API Tracker

STABLE	BETA	DEV
Chromium 104	Chromium 105	Chromium 106
Stable 2 days ago (Aug 2, 2022)	Stable in 26 days (Aug 30, 2022)	Stable in 54 days (Sep 27, 2022)

Shipped

Web Bluetooth API	M56		+
WebUSB API	M61	🐧 ⊞ ◉ 🍎 🤖	+
Web Share Target	M71	🤖	+
Web Share API Level 2	M75	🤖	+
Async Clipboard: Read and Write Images	M76	🐧 🤖 ⊞ ◉ 🍎	+
Web Share Target Level 2	M76	🤖	+
Enter Key Hint	M77	🤖	+
Expand Storage Quota	M78	🐧 🤖 ⊞ ◉ 🍎	
Get Installed Related Apps API	M80	🤖	
Periodic Background Sync	M80	🐧 🤖 ⊞ ◉ 🍎	
PWA desktop-pwas: Support "minimal-ui" display mode	M80	🐧 ⊞ ◉ 🍎	
Compression codecs	M80	🐧 🤖 ⊞ ◉ 🍎	
Contacts API	M80	🤖	

PWAs & MiniApps:

What Is The Web Missing?

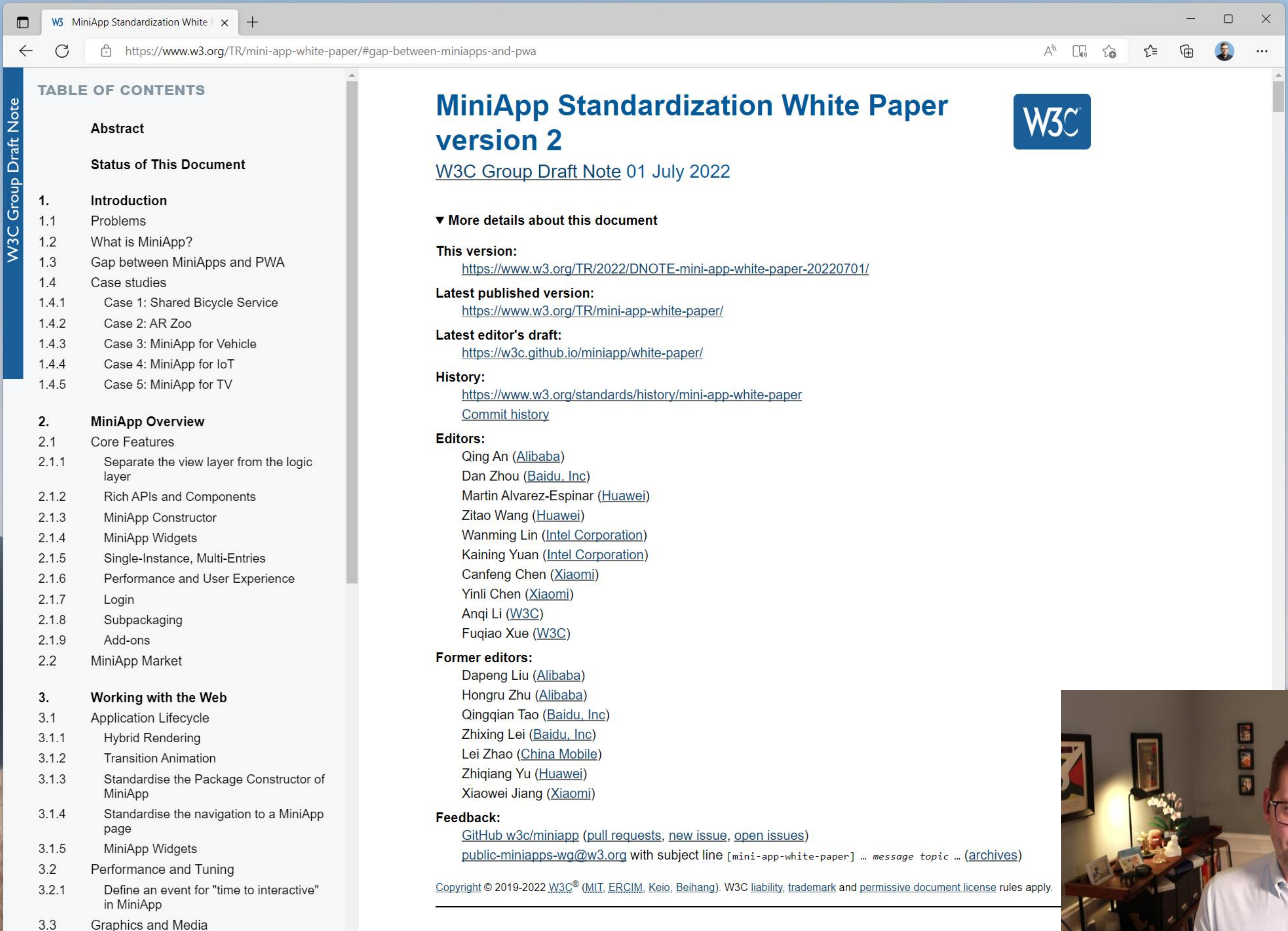

MiniApp Standardization White
https://www.w3.org/TR/mini-app-white-paper/#gap-between-miniapps-and-pwa
W3C Group Draft Note

TABLE OF CONTENTS

MiniApp Standardization White Paper version 2

W3C Group Draft Note 01 July 2022

▼ More details about this document

This version:
https://www.w3.org/TR/2022/DNOTE-mini-app-white-paper-20220701/

Latest published version:
https://www.w3.org/TR/mini-app-white-paper/

Latest editor's draft:
https://w3c.github.io/miniapp/white-paper/

History:
https://www.w3.org/standards/history/mini-app-white-paper
Commit history

Editors:
Qing An (Alibaba)
Dan Zhou (Baidu, Inc)
Martin Alvarez-Espinar (Huawei)
Zitao Wang (Huawei)
Wanming Lin (Intel Corporation)
Kaining Yuan (Intel Corporation)
Canfeng Chen (Xiaomi)
Yinli Chen (Xiaomi)
Anqi Li (W3C)
Fuqiao Xue (W3C)

Former editors:
Dapeng Liu (Alibaba)
Hongru Zhu (Alibaba)
Qingqian Tao (Baidu, Inc)
Zhixing Lei (Baidu, Inc)
Lei Zhao (China Mobile)
Zhiqiang Yu (Huawei)
Xiaowei Jiang (Xiaomi)

Feedback:
GitHub w3c/miniapp (pull requests, new issue, open issues)
public-miniapps-wg@w3.org with subject line `[mini-app-white-paper]` … *message topic* … (archives)

MiniApp Standardization White

https://www.w3.org/TR/mini-app-white-paper/#gap-between-miniapps-and-pwa

W3C Group Draft Note

TABLE OF CONTENTS

reminders using the device's native alarm and calendar features, perform phone calls and trigger performance warnings. Although both technologies have similar APIs and services, there is a significant gap between the API specifications in each application type. PWAs rely on standard Web APIs, while MiniApps implement non-standard APIs to maximize the platform's capabilities, such as device-specific features and vendor-exclusive services.

Depending on the implementation, a MiniApp user agent could be an operating system, a super app, or any other hosting platform based on different and various rendering engines and WebViews. The architecture of a MiniApp user agent differs from PWA user agents, as we can see in the following picture.

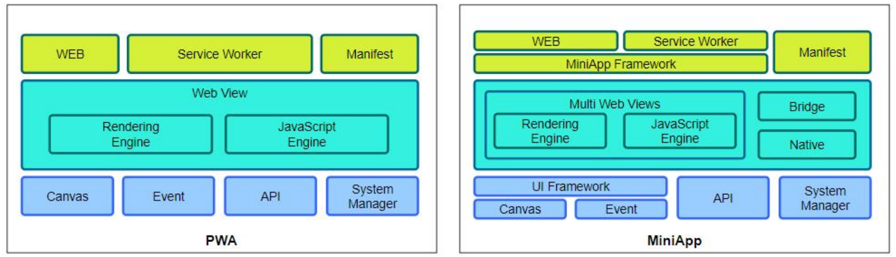
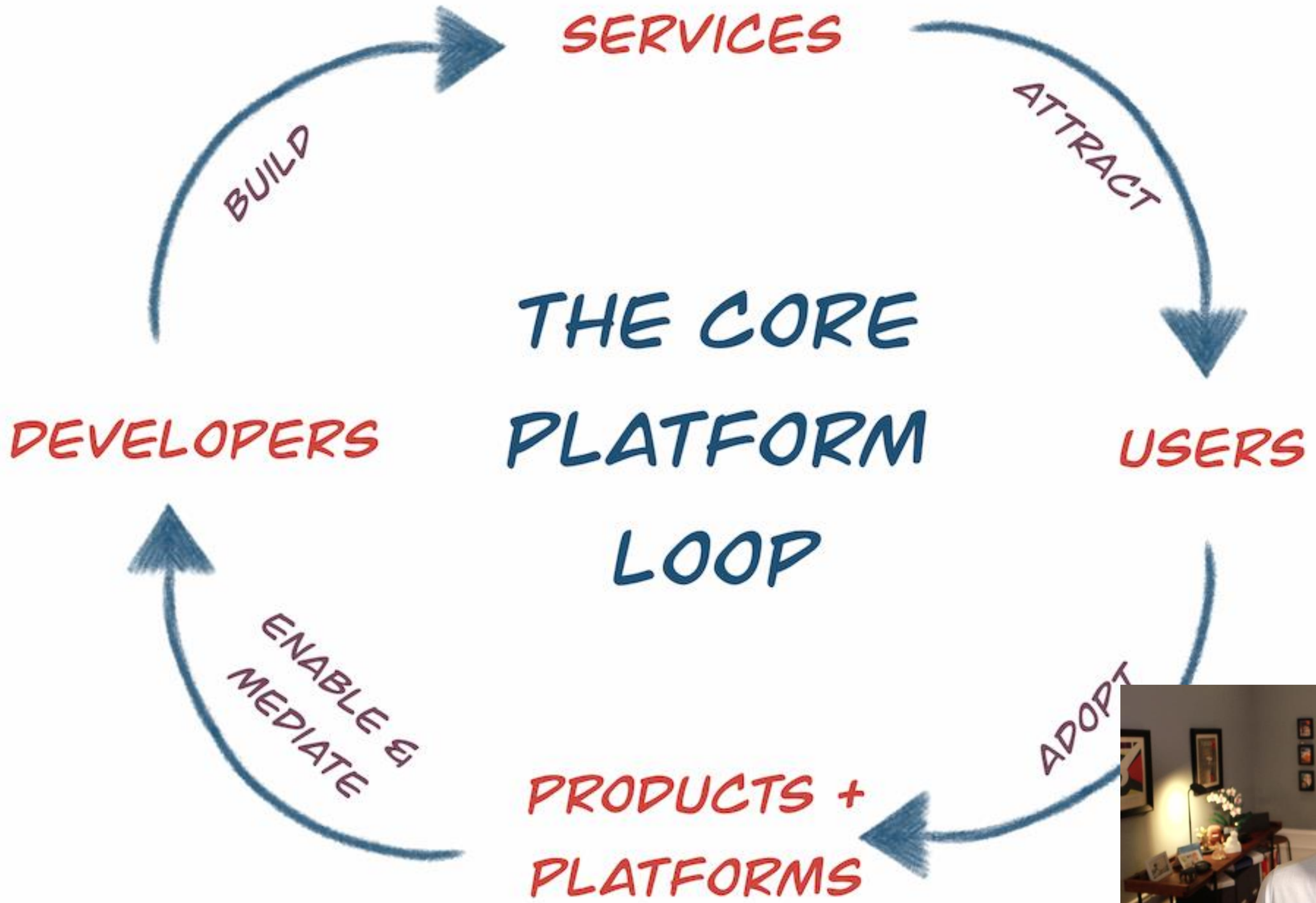


Figure 1 The architecture of MiniApps and PWA

The comparison table in the annex identifies and highlights the differences between Progressive Web Applications (PWA) and various MiniApp implementations. Some of these differences are summarized in the following table.

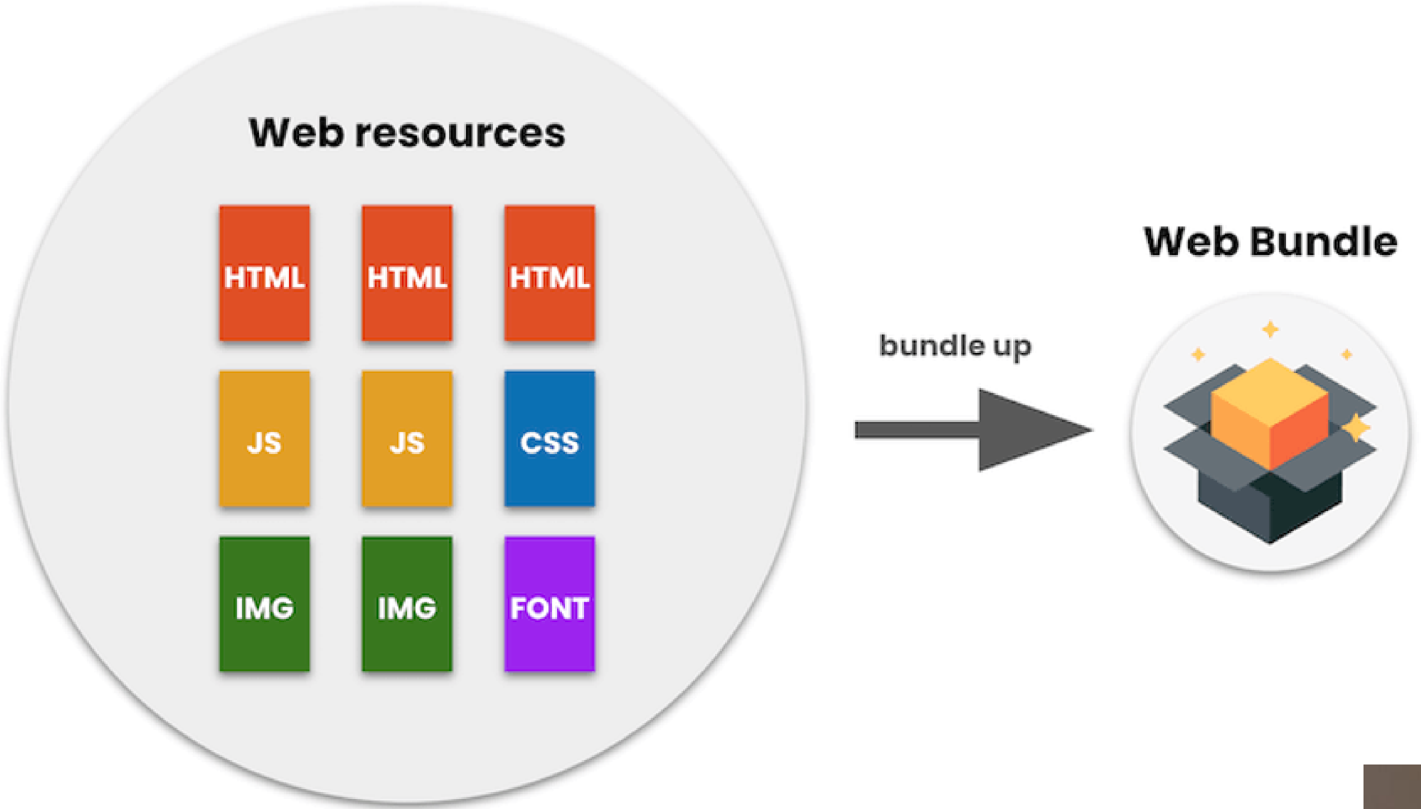
Feature	Progressive Web App	MiniApp
Source code	Standard markup languages (HTML), stylesheets (CSS), and scripts (JavaScript).	Non-standard dialects of HTML, CSS and JavaScript
Deployed Format	Web resources (mainly: HTML, CSS, JavaScript code, and WebAssembly modules)	HTML, CSS, JavaScript, and other resources packed in a ZIP container.
Packaging	No. Resources linked on the Web.	Yes. Different package formats per vendor.
Needs to host files on Web server	Yes	No
Installation-free usage	Yes, running in the browser.	Running in a super app or on the OS.
Installation with standalone icon	From the browser or app marketplace (optional)	No
Services	Access to Web APIs	Access to non-standard Web APIs, including some system native APIs


SERVICES
ATTRACT
USERS
ADOPT
PRODUCTS + PLATFORMS
ENABLE & MEDIATE
DEVELOPERS
BUILD
THE CORE PLATFORM LOOP

A Gnawing Question:

What Is The Web Missing?

- Packaging & Signing
- Super-App Service Integration
- Upgrades to HTML & CSS
- API access


Web resources
HTML
HTML
HTML
JS
JS
CSS
IMG
IMG
FONT
bundle up
Web Bundle

Thank You.

Alex Russell <alexrussell@microsoft.com>
infrequently.org
@slightlylate