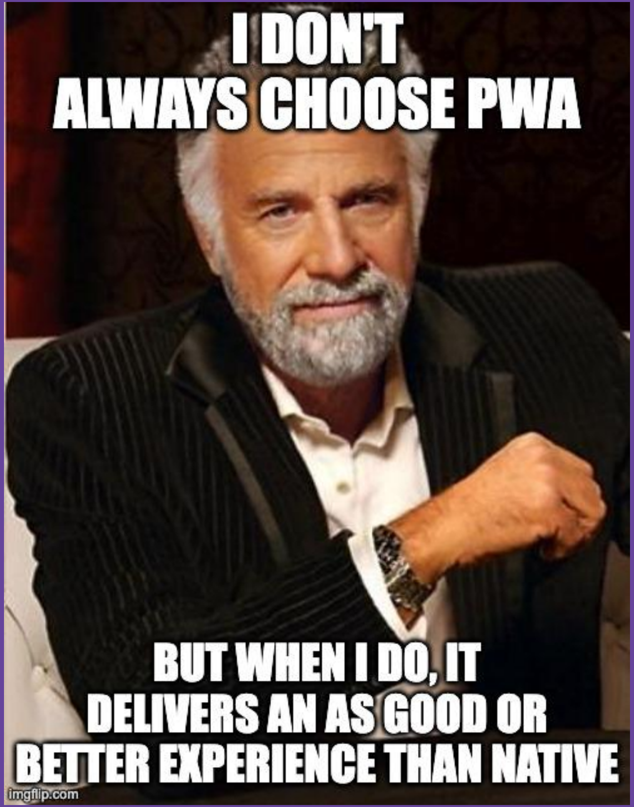
Cross-Platform Apps Powered by Drupal for Everyone

Alex Borsody - Moonraft a UST Company



Obligatory meme from 2010

Disclaimer

- For people familiar with Drupal and working with Drupal’s theme layer (there are a lot) for those looking to get the most out of their coupled architecture and also for those who are interested in learning more about decoupled (off the Drupal island)

- Focused on technologies running in the **web browser** Chromium/Webkit (Chrome/Safari).

- PWA is different technologies that come together focused on the web browser to create native app experience.

- This particular talk is from an architect’s perspective, choosing and implementing various technologies that come together to make a great PWA.

- This is a survey, any of the slides could be their own talk, there will be code samples and repositories provided as well as a companion blog post.

Where are we going?

- PWA module as a starting point.
- PWA Builder and getting your into the app stores.
- Choosing decoupled or standard Drupal theme layer.
- Everything you need to know about making your PWA truly feel like native app. (which no one will talk about right now but Google)

What is a PWA? What is not a PWA?

PWA - Progressive Web App

Essentially an effort led by Google and Microsoft to make the web browser a runtime environment on par with a native app experience. Truly write code once for Android and iOS, plus get an optimized website for performance and SEO.

By Definition

- Service worker: Offline functionality
- Web manifest: Add to home screen and icons.
- Security: Must use https (service worker and session cookies)

PWAs Can Also

- Be discoverable in app stores via TWA or custom wrapper for iOS
- Have access to native device features whatwebcando.today
- Web technologies “just work”

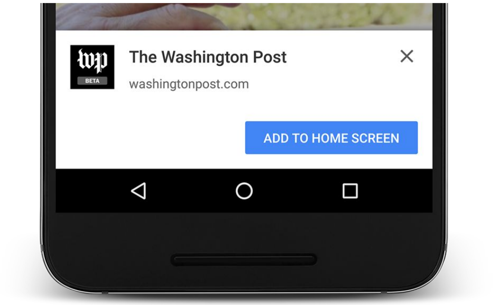
Cross-Platform-App

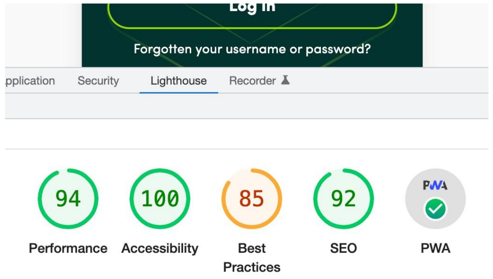
Cross platform App is a general term that means the app can run on multiple platforms from a single codebase and so could be a PWA or any of these other technologies.

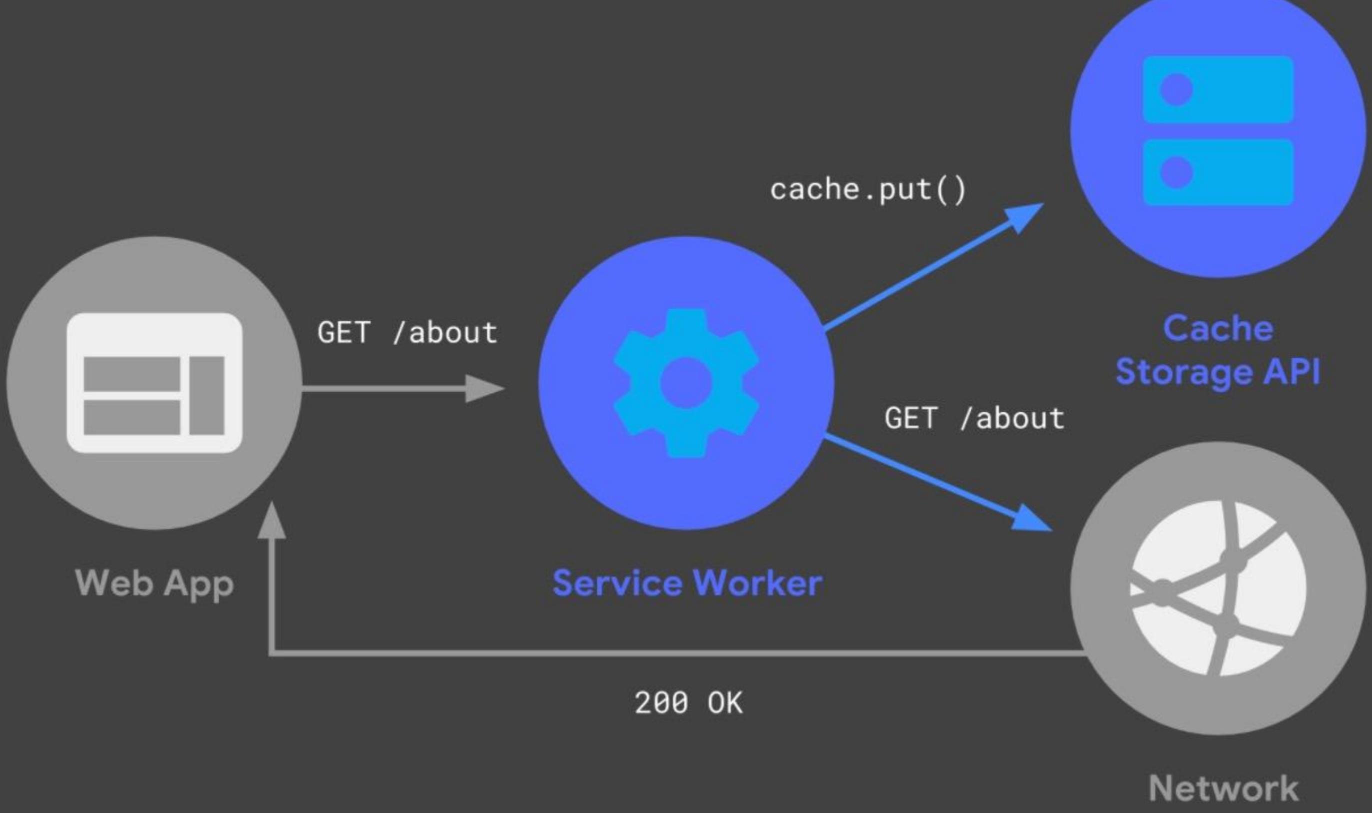
- React Native
- Flutter
- Ionic/Cordova/Phonegap
- In general more technologies involved beyond those familiar with web developers, more labor intensive. Potentially more margin for error.

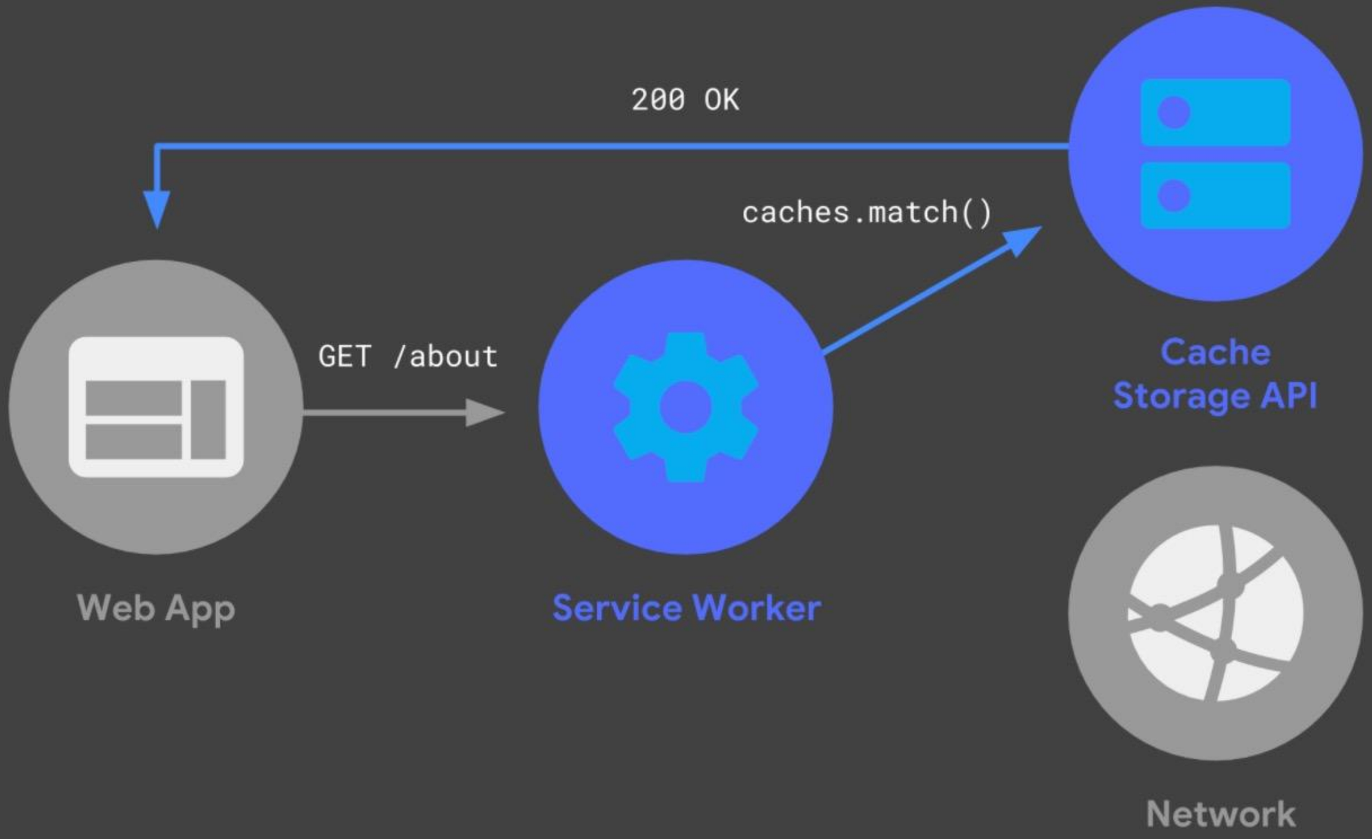
Drupal PWA Module 🔗

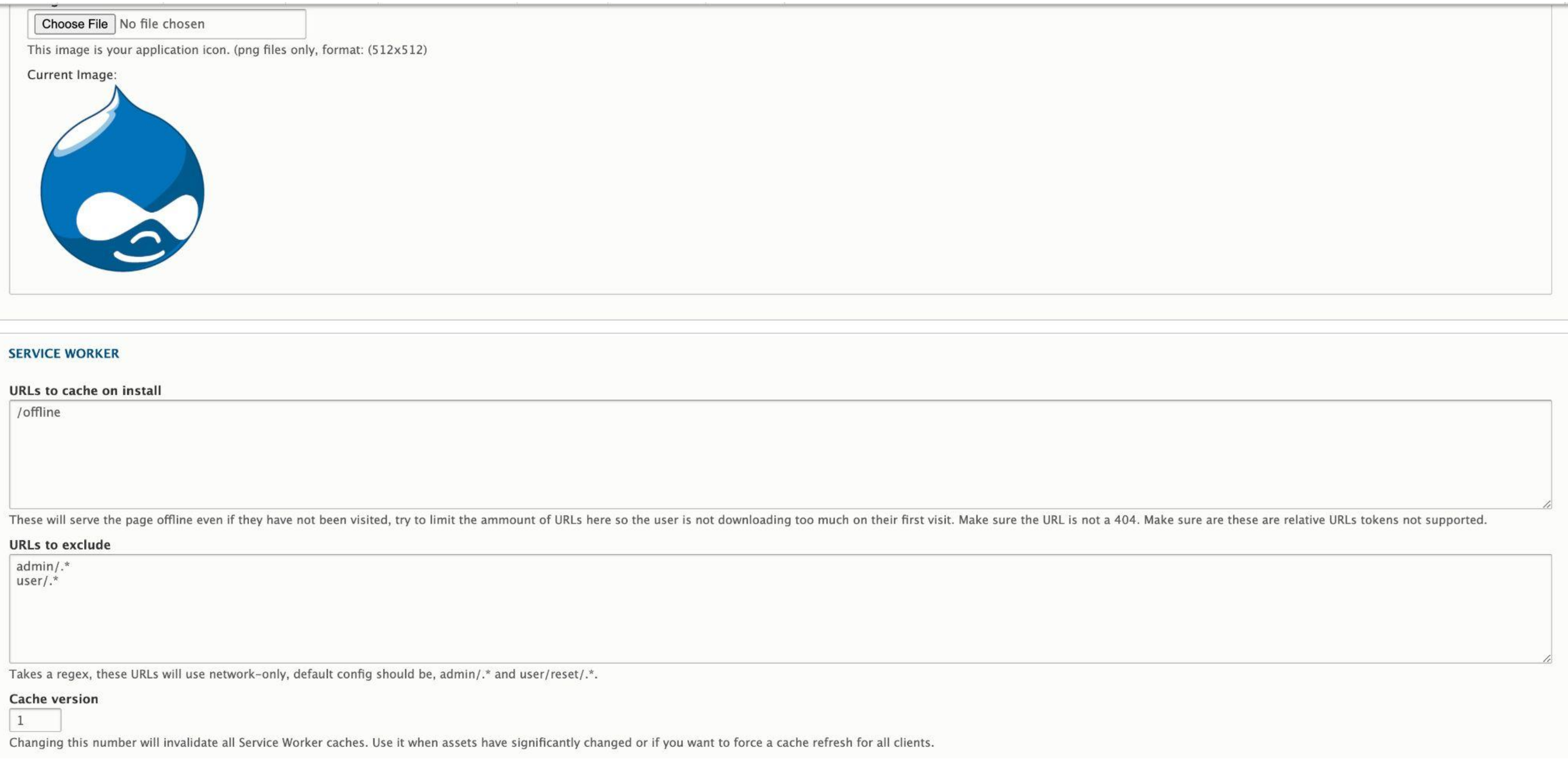
- Offline caching provided by a service worker.
- Manifest + PWA Extras
- Improve lighthouse score, performance and SEO.






cache.put()
GET /about
Cache
Storage API
Web App
Service Worker
GET /about
200 OK
Network


200 OK
caches.match()
GET /about
Cache Storage API
Web App
Service Worker
Network



Choose File No file chosen

This image is your application icon. (png files only, format: (512x512)

Current Image:

SERVICE WORKER

URLs to cache on install

```
/offline
```

These will serve the page offline even if they have not been visited, try to limit the ammount of URLs here so the user is not downloading too much on their first visit. Make sure the URL is not a 404. Make sure are these are relative URLs tokens not supported.

URLs to exclude

```
admin/.*
user/.*
```

Takes a regex, these URLs will use network-only, default config should be, admin/.* and user/reset/.*.

Cache version

```
1
```

Changing this number will invalidate all Service Worker caches. Use it when assets have significantly changed or if you want to force a cache refresh for all clients.

PWA Module Demo

Workbox + PWA Module

Effort made to port the D8 module to Workbox (now done in D7)

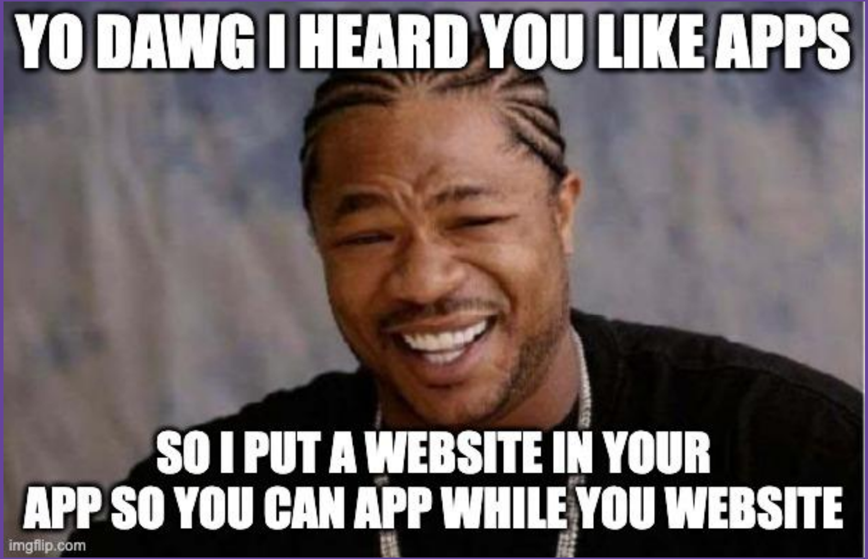
- **Precaching** - Using precache manifest
- **Runtime caching strategies** - cache first, network first, stale-while-revalidate
- **Request routing** - Different strategies for different assets and URLS (cache first for images)
- **Background sync** - Post request fails come back online they retry

Cool... but I want to be *discoverable in the app stores*

TWA, WKWebview and PWA Builder

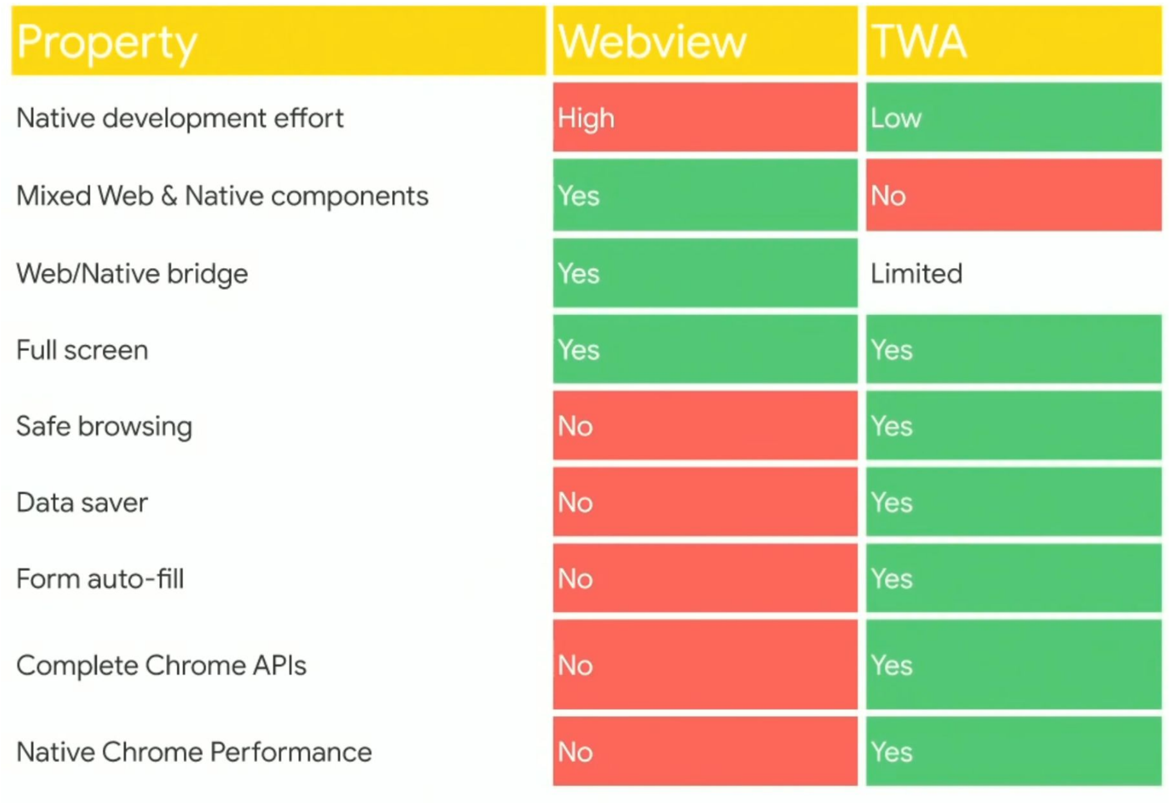
- The PWA module provides everything you need to run through PWA Builder which will…
  - Create a lightweight .apk/.aap to submit to the Play Store 800kb
  - Wrap your website in WKWebView to submit to the App Store
  - Provides you the original source code to make modifications if you want

Demo

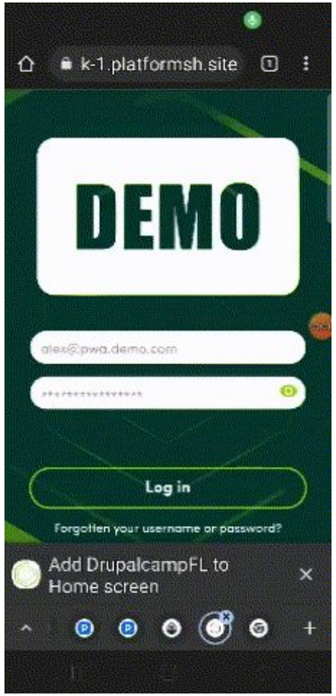

YO DAWG I HEARD YOU LIKE APPS
SO I PUT A WEBSITE IN YOUR APP SO YOU CAN APP WHILE YOU WEBSITE
imgflip.com

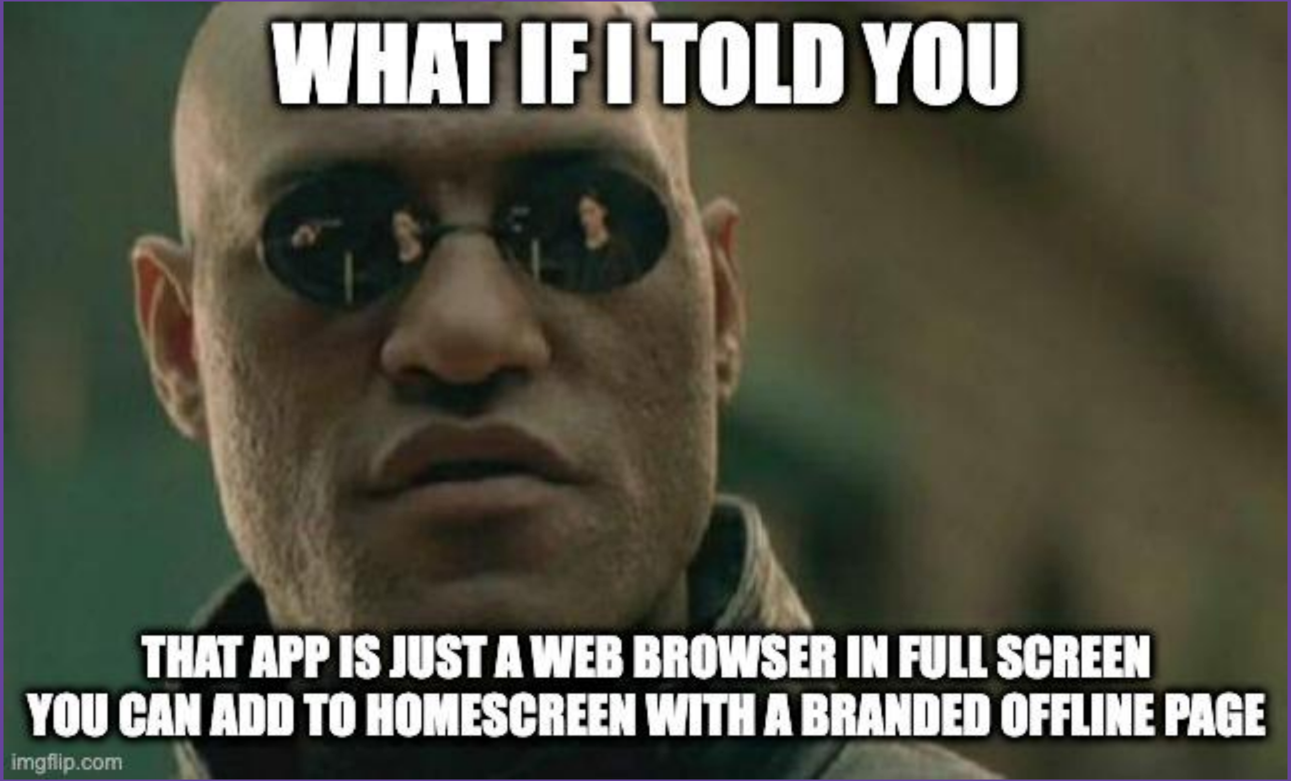
Google TWA

in Chrome Browser



Property	Webview	TWA
Native development effort	High	Low
Mixed Web & Native components	Yes	No
Web/Native bridge	Yes	Limited
Full screen	Yes	Yes
Safe browsing	No	Yes
Data saver	No	Yes
Form auto-fill	No	Yes
Complete Chrome APIs	No	Yes
Native Chrome Performance	No	Yes


k-1.platformsh.site
DEMO
alex@pwa.demo.com
Log in
Forgotten your username or password?
Add DrupalcampFL to Home screen


WHAT IF I TOLD YOU
THAT APP IS JUST A WEB BROWSER IN FULL SCREEN
YOU CAN ADD TO HOMESCREEN WITH A BRANDED OFFLINE PAGE
imgflip.com

iOS Wrapper - WebKit - WKWebView

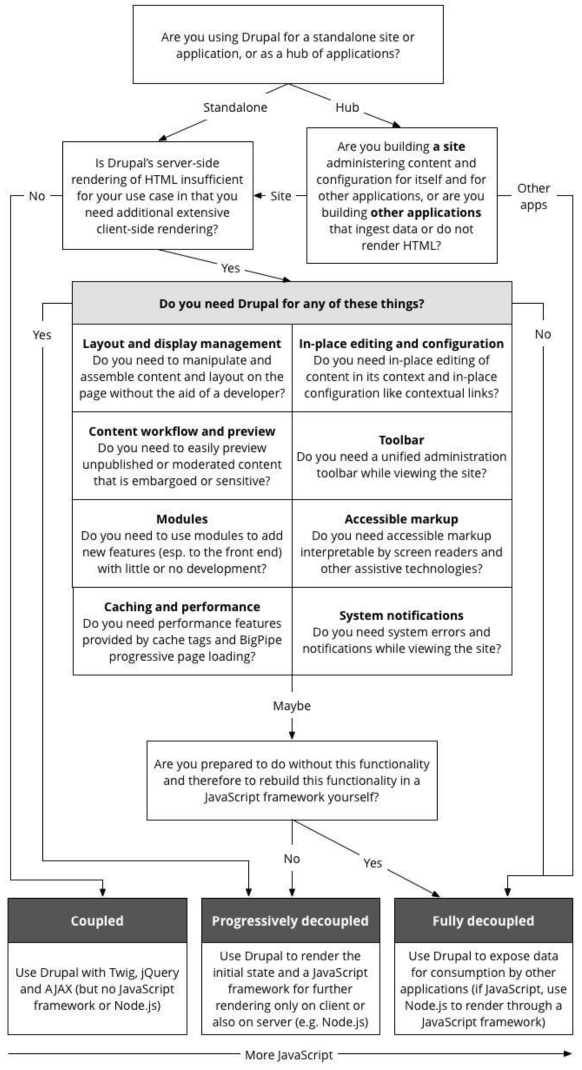
Custom functionality possible in WKWebView

i. Apple minimal use guidelines and keeping up to date with Safari: Firtman
ii. No bounce
iii. Setting cookie from start_url
iv. Adding biometric to WKWebView (check url and don’t show on user/login)
v. Adding push to WKWebView iOS workaround with cookies difficult to do
vi. Associated domain
vii. Handling login:
   1. Cookie expiration GC and Auto Logout module **session.gc_maxlifetime, session.cookie_lifetime**
   2. Biometric login.

Making your Drupal website “App like”

Fully Decoupled vs Progressive; Decision Making Flowchart

- A fully Decoupled Architecture enables the Front end team to have the complete freedom to showcase creativity with Javascript MVC Frameworks.
- The strategy of Fully Decoupling limits Drupal's capabilities like, layout and display management, content previews, user interface (UI) localisation, form display, accessibility, authentication, security features such as XSS & CSRF and performance.
- There are certain Risks and Rewards of Fully Decoupling Drupal.
- The flow chart helps with the decision making to go with a Fully decoupled vs Progressively Decoupled Architecture.



Are you using Drupal for a standalone site or application, or as a hub of applications?
Standalone
Hub
Is Drupal's server-side rendering of HTML insufficient for your use case in that you need additional extensive client-side rendering?
Are you building **a site** administering content and configuration for itself and for other applications, or are you building **other applications** that ingest data or do not render HTML?
No
Site
Other apps
Yes
Do you need Drupal for any of these things?
Yes
No
Layout and display management Do you need to manipulate and assemble content and layout on the page without the aid of a developer?
In-place editing and configuration Do you need in-place editing of content in its context and in-place configuration like contextual links?
Content workflow and preview Do you need to easily preview unpublished or moderated content that is embargoed or sensitive?
Toolbar Do you need a unified administration toolbar while viewing the site?
Modules Do you need to use modules to add new features (esp. to the front end) with little or no development?
Accessible markup Do you need accessible markup interpretable by screen readers and other assistive technologies?
Caching and performance Do you need performance features provided by cache tags and BigPipe progressive page loading?
System notifications Do you need system errors and notifications while viewing the site?
Maybe
Are you prepared to do without this functionality and therefore to rebuild this functionality in a JavaScript framework yourself?
No
Yes
Coupled
Use Drupal with Twig, jQuery and AJAX (but no JavaScript framework or Node.js)
Progressively decoupled
Use Drupal to render the initial state and a JavaScript framework for further rendering only on client or also on server (e.g. Node.js)
Fully decoupled
Use Drupal to expose data for consumption by other applications (if JavaScript, use Node.js to render through a JavaScript framework)
More JavaScript

Reference:
https://dri.es/how-should-you-decouple-drupal

A lot can be done while using Drupal’s theme layer

Use all web technologies available to feel “app-like”

UX and frontend dev experience is extremely important to making your website feel like an app and is accessible to all web developers.

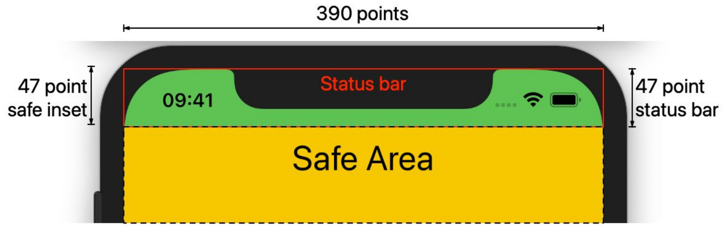
Keeping all CSS/HTML/JS updated and compatible with all browsers takes a lot of research but once all adjustments are made, changes only need to happen occasionally when a new version of Safari comes out or a new iPhone screen size.

This is about taking Google’s work on PWAs one step further, considering and combining *all web technologies available* to make the web look and feel like an app.

The Chrome team is constantly improving the browser experience, other browsers are following suit.

CSS, HTML, JS and UX Considerations

- Javascript touch events: disable pinch zoom and swipe gestures
- Viewport meta tag
- CSS:
  - Minify/optimize CSS don’t have flashes of unstyled content, Lighthouse
  - “App-like” input/font sizes and making sure everything fits in viewport make it *visually look* like an app
- Preloaders everywhere!
- HTML attributes
  - autocomplete="username"
  - autocomplete="current-password"
  - autocomplete="one-time-code" WebOTP API
  - input type="tel"
- IPhone specific
  - Status bar on iPhone X plus safe area
  - Meta tags and pwa_extras submodule
- Ajax API (Drupal specific)



iPhone 12

LOADING ICONS
LOADING ICONS EVERYWHERE
LIGHTYEAR
imgflip.com

The Weather Company
An IBM Business

AliExpress

airberlin

The Washington Post

Mobify

Geo

NFL

letras

5miles

konga.com

Geo News

Walgreens
at the corner of happy & healthy

suumo
スーモ

BABE

タウンワーク
TOWN WORK

Flipkart

PURE FORMULAS

snapdeal

Booking.com

KASKUS

JUMIA

JalanTikus