Progressive Web Apps

An overview of their capabilities and introduction to associated technologies

Part 1: An introduction to PWAs

The problem space

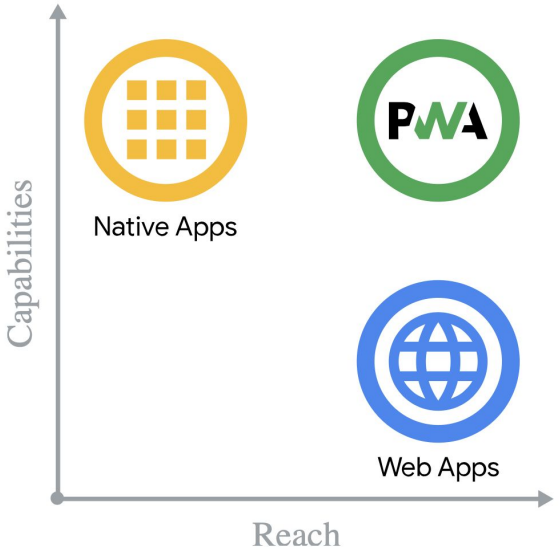
- Building, maintaining and distributing multiple native apps - for both mobile and desktop - is expensive and difficult
- Web apps don't have access to some of the 'nice to have' features enjoyed by native

Put simply: native is capable but expensive with limited reach; web is less capable, cheaper and with much better reach.

Hence the rise of compile to native tools

*"..but those experiences all rely on technologies that **won't also work in the browser**. PWA's goal is to build directly on the web. So that means you can maintain one code base that's going to work for users in the browser and your installed users."*

PJ McLachlan, Chrome Dev Summit 2019 See: https://youtu.be/Hp_dQvQyYEI?t=241



What is a PWA

“A Progressive Web App (PWA) is a web app that leverages special browser capabilities that enable the app to ***act more like a native or mobile app*** *when running on capable browsers. That’s it, that’s all that needs to be said - …”*

Learning Progressive Web Apps by by John M. Wargo (Addison Wesley 2020)

The community-driven logo for PWA

The PWA advantage

“With the additional superpowers they introduce, progressive web apps fulfil many of the expectations we have from native apps [including]: availability regardless of connection; fast load times; push notifications; homescreen shortcut; native look”

Building Progressive Web Apps, by Tal Ater (O’Reilly)

What does ‘act like a native app’ mean?

Here are ***some*** key things PWAs can do:

- be **installed**, either via the browser or via an app store.
  - Note: Once installed they can look and feel much like a native app (with an icon and accessible via familiar discovery and launch processes for the platform etc.)
- **load instantly**
- work **offline**
- **run outside the browser**
- **re-engage users when the app is closed**, via:
  - accepting **push messages** (incl. background data sync)
  - **raising notifications** (with badges etc.)

Note: what you can do with a PWA will differ by the user’s platform - hence the emphasis on ‘progressive’ in learning materials associated with PWA.

Plus PWAs have access to *some* native APIS, including:

- Push API (allowing us to send notifications to users when the app is closed)
- Web Share API
- Web Bluetooth API
- Sensors (accelerometer) and geolocation
- MediaStream Recording API
- Payment Request API
- Gamepad API
- Web RTC (which supports things like peer-to-peer teleconferencing)
- WebAssembly (a low-level language that provides languages like C++ and Rust with a compilation target for the web)
- WebGL (2D and 3D graphics)

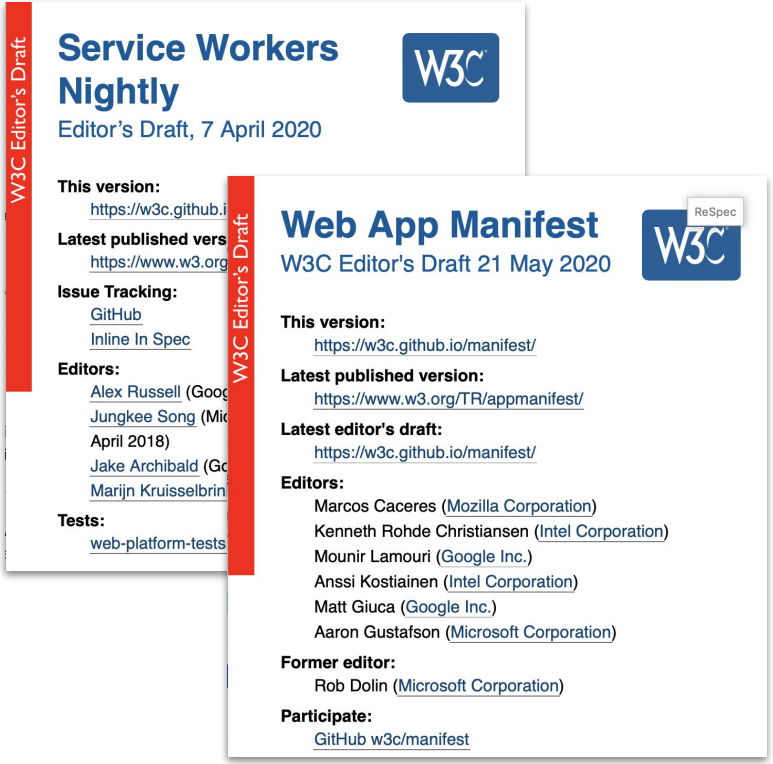
Just remember: a progressive enhancement approach will be essential when building these capabilities into PWA, because not all these APIs, capabilities or sensors will be available for all the places your app might run

How does a PWA achieve this?

PWA is a design pattern that combines discrete standard specifications

"There's no standard or standards body for PWAs; a PWA is just a web app built to act a certain way. Use as much or as little PWA functionality as you want in your web apps. You can have web apps that use some PWA capabilities but aren't PWAs and PWAs that use only some PWA capabilities."

Learning Progressive Web Apps by by John M. Wargo (Addison Wesley 2020)



The three necessary components for your web app to be a PWA

The three components of a PWA

But, in order to be installable, the PWA must meet also the browser's **installability criteria**. Only then will users be able to install it.

PWA installability criteria

In addition to *being* a PWA, to be installable it must meet the browser's installability criteria. Many browsers today will analyse your web app to determine if it's a PWA and - if it meets ***their*** installability criteria - will:

- Indicate to the user that it is installable
- Provide a `beforeinstallprompt` event we can hook into

You can see the current installability criteria for the Chrome browser here: https://web.dev/install-criteria/.

Note also, once you put a PWA online, it might just appear in app stores. For example, the Bing web crawler automatically packages PWAs for installation on Windows 10 and puts them in the web store

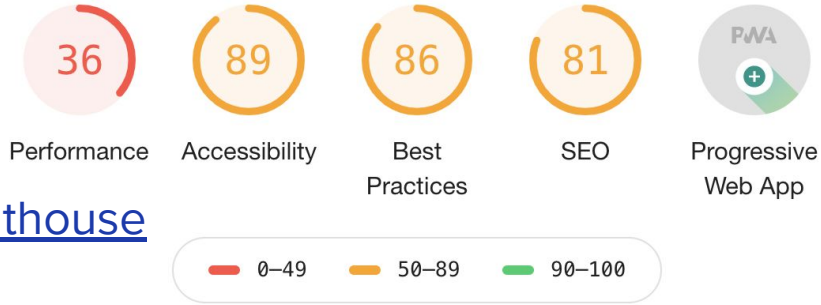
Google Lighthouse for PWA ‘audits’

The Lighthouse tool (which can be accessed via the Chrome Developer Tools ***or*** as a command line utility is capable of performing PWA ***audits***.

Note: unlike other audits (such as those for accessibility), it’s perfectly OK not to get a ‘100 score’ for a PWA since the decision to opt in to PWA features, and the extent of that opt in, are to be determined by the team.

Read more about Lighthouse here:

https://developers.google.com/web/tools/lighthouse



Installation process

The steps to install **vary by platform**. Some prompt users to install, others don’t.

But once installed, the PWA is released from the browser and behaves like a native app - including through familiar discovery and launch patterns

Some important UX considerations for PWAs

"... most importantly, the word install signals to users that **the experience was meant to be great on their device**" https://youtu.be/Hp_dQvQyYEI?t=241

"... there is a bit of an install funnel. This is the same as native apps or e-commerce conversions. Most strategies for optimization apply here as well. You don't want to push the user to convert too soon or they'll leave your site running. ***You should only promote install to users who are frequent users or who will actually benefit from your services****."* https://youtu.be/Hp_dQvQyYEI?t=326

So, in summary, PWAs are

- PWA === Web app + ServiceWorker + distribution model (i.e. manifest)
- *‘write once, run anywhere’* combining the capability of native with reach of web
- Installable via:
  - the browser if they meet the browser’s installability criteria
  - app stores. Some stores will package PWAs their crawlers encounter
- Once installed they behave like native apps and are accessed via familiar discovery and launch capabilities
- A careful progressive enhancement approach is vital because some APIs might not be available on all platforms