Capability Report

PROGRESSIVE WEB APPLICATIONS (PWA)

The flexibility of the web with a seamless user experience.

2019 | MECHANICAL ROCK

Table of Contents

Background

What is a PWA?

Progressive Web Apps (PWAs) are web applications but offer the user functionality traditionally available only to native mobile applications. With features like:

- working offline
- push notifications
- access to native hardware

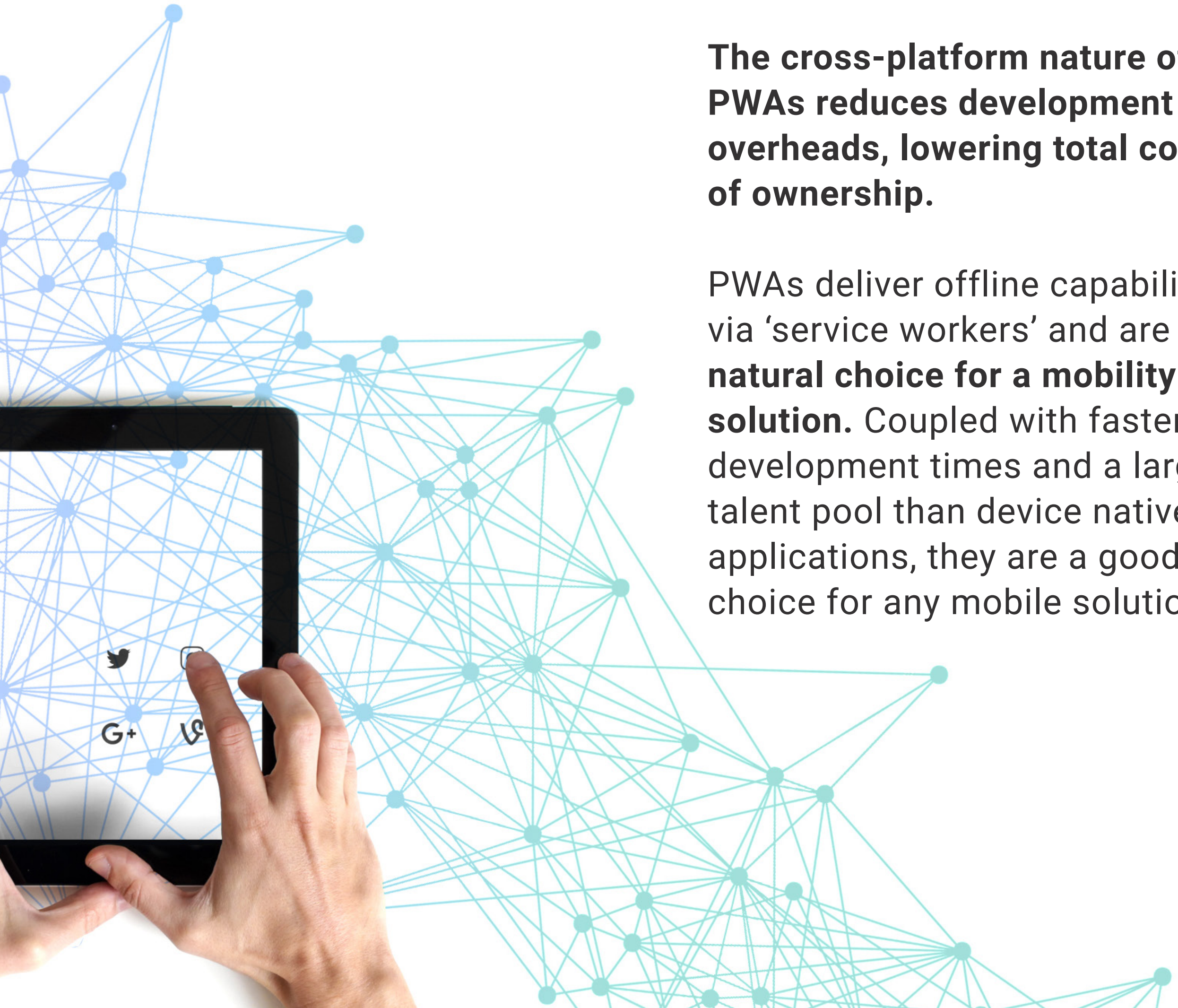
PWAs can provide a rich user experience, equalling that of native applications, and are based on common web standards.

A PWA is simply an advanced web application.

It offers the **frictionless user experience of a native mobile application** with the easy updates of a web application.

PWAs are an emerging technology that combine the open standards of the web offered by modern browsers to provide benefits of a rich mobile experience.
- *Wikipedia*

Background



The cross-platform nature of PWAs reduces development overheads, lowering total cost of ownership.

PWAs deliver offline capabilities via ‘service workers’ and are **a natural choice for a mobility solution.** Coupled with faster development times and a larger talent pool than device native applications, they are a good choice for any mobile solution.

Web standards are continually evolving, with support for many advanced features already available or in experimental preview.

Cutting edge features like **augmented reality** (AR), access to hardware features such as **accelerometer, GPS and Bluetooth** are implemented in some platforms.

However, due to the complexity of the landscape, support differs across platforms.

Why Progressive Web Applications?

- ✓ Native Device Support
- ✓ Multi Platform
- ✓ Instant Updates
- ✓ Offline Support
- ✓ Frictionless Deployment

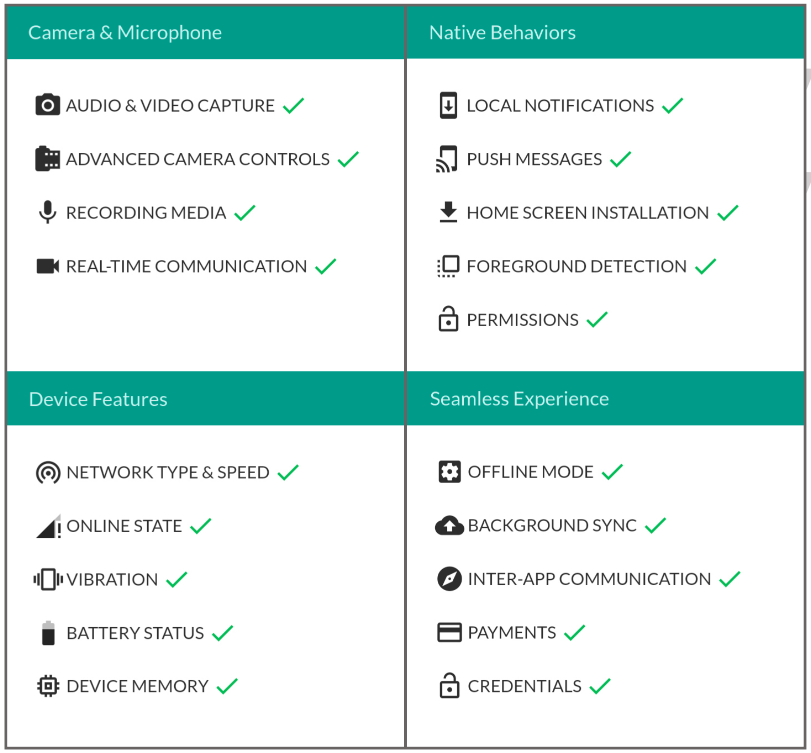
Why Progressive Web Applications? (Continued...)

✓ Native Device Support

Progressive Web Apps running on modern browsers have **access to nearly all of the functions** that were only available to native applications traditionally.

Features like push notifications, geo-location and touch gestures are all available to PWAs.

Access to cameras, sensors, bluetooth and device features is now available on most browsers.



Camera & Microphone	Native Behaviors
AUDIO & VIDEO CAPTURE ✓	LOCAL NOTIFICATIONS ✓
ADVANCED CAMERA CONTROLS ✓	PUSH MESSAGES ✓
RECORDING MEDIA ✓	HOME SCREEN INSTALLATION ✓
REAL-TIME COMMUNICATION ✓	FOREGROUND DETECTION ✓
	PERMISSIONS ✓

Device Features	Seamless Experience
NETWORK TYPE & SPEED ✓	OFFLINE MODE ✓
ONLINE STATE ✓	BACKGROUND SYNC ✓
VIBRATION ✓	INTER-APP COMMUNICATION ✓
BATTERY STATUS ✓	PAYMENTS ✓
DEVICE MEMORY ✓	CREDENTIALS ✓

Diagram 1 - Examples of Native Device Support for Android from https://whatwebcando.today

Why Progressive Web Applications? (Continued...)

✓ Multi Platform

Developers only have to develop and worry about one code base which **decreases the time to build and release updates** (sometimes as much as 3x).

Instead of building, testing and deploying separate code bases for Android, iOS and Web - **there is a single code base to maintain**.

The application can run on any device with a browser, including a desktop, tablet or a mobile.

✓ Instant Updates

Service workers allow new scripts/stylesheets to be used as soon as they are deployed to the server so **the user doesn't even need to press an update button.**

✓ Offline Support

Service workers and local data stores also allow PWAs to **operate beyond the network**. An application can be downloaded once and will **run independently** of network access.

✓ Frictionless Deployment

Deploy through a single channel without having to go through a third party like the Play Store.

With a modern CI/CD pipeline and a PWA, changes and bug fixes can be deployed in a matter of minutes, several times a day, with **no interruption to the user**.

Alternatives to PWAs and Limitations

Native Mobile Applications

The primary mobile operating systems are Android and iOS (after the discontinuation of Windows 10 Mobile in 2017).

Native applications for **each operating system use different languages and tool sets**, meaning separate native applications must be developed for each platform.

Each mobile platform has its own native programming language, SDKs and tools: iOS uses Swift and Xcode; Android promotes Kotlin and Android Studio.

Native applications offer a responsive use experience on a specific device, and can access device specific hardware, but have a number of limitations.

Limitations of Native Mobile Applications

Building and deploying applications is **complicated and slow**, often involving updates pushed through Apple or Android app stores.

Running on multiple mobile platforms means **developing and maintaining separate code bases for each platform.**

Alternatives to PWAs and Limitations

Cross-Platform Programming and Limitations

In an attempt to tackle the challenges around building native applications for multiple platforms, a number of cross platform frameworks have been developed.

Apache Cordova, originally PhoneGap, was one of the first cross platform frameworks, released in 2009 focussed on using CSS, HTML and Javascript built within a native WebView.

Xamarin, owned by Microsoft and founded in 2011, allows developers to build Android and iOS applications using C#.

React Native, released by Facebook in 2015, enables the development of native applications using the React web framework. React Native interprets the Javascript messages to manipulate native views.

Flutter, released by Google in 2015, uses the Dart language and ahead-of-time compilation to build native applications for iOS and Android.

While these systems overcome some of the limitations of native mobile applications they still remain challenging to build, update and deploy continuously.

Comparisons

	PWA	Native	Cross-Platform
Device Compatibility	PWAs currently have some limitations with access to some platform specific capabilities.	Applications have full access to native hardware through native SDKs.	Applications have full access to native hardware through native SDKs.
Consistency	A single codebase supports multiple platforms.	Building apps for each platform means features must be built multiple times.	A shared codebase across mobile platforms improves consistency.
Deployment	Seamless, automatic, upgrade process for end users.	Time to production measured in days, due to complex testing and release management.	Time to production measured in days, due to complex testing and release management.
Technical Accessibility	Javascript is consistently the most popular language of the web, by orders of magnitude. Skills are transferable across front and back end services.	Android and iOS development skills are niche compared to JavaScript.	ReactNative and Apache Cordova are built on JavaScript, so popular. Xamarin is aimed at .NET developer skill sets, again popular. Flutter is new so very niche.
Speed of Development	Mature tool chains provide instant developer feedback. Testing frameworks support fast feedback loops.	Testing requires emulation, with slower feedback loops, or access to a range of hardware devices.	Testing requires emulation, with slower feedback loops, or access to a range of hardware devices.

Compatibility

The Web platform standards, including HTML and Javascript are continuously evolving and adding new capabilities.

Web browsers, developed by different organisations, each support the standards at a different rate.

As such, understanding the compatibility of the wide array of browsers can feel complex and daunting.

If you do not require specialist features of the device, then **PWAs offer a number of other significant advantages.**

PWA are recommended when :

- **You require a flexible platform that allows continuous updates.**
- **You have access to a modern browser, e.g. Chrome.**
- **You want a responsive cross-platform experience.**

An oft-cited limitation of PWAs at the moment is push notifications.

Push notifications are well supported on Android, but are not natively supported on iOS.

If you have control over the hardware platform, Android provides greater flexibility and interoperability.

Compatibility (Continued...)

It is worth noting that the Web platform is a continually evolving standard, with new capabilities being added.

For example, the WebVR and WebXR specifications are currently experimental for developing virtual and augmented reality applications.

PWAs provide a future-proof platform.

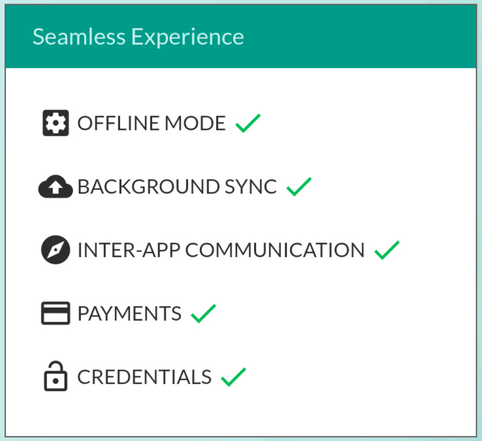
Whilst there are compatability differences between different browsers and platforms, web standards **provide a platform for a consistent experience across multiple devices.**

By comparison, native applications provide full access to all native capabilities supported by the device/SDKs.

Whilst this provides the maximum flexibility, there are additional complexities when considering native applications, including device compatibility, screen resolution and app permissions.

If you want to find out what your web browser is capable of visit https://whatwebcando.today

Note that you will get different results depending on the device and browser you use.



Seamless Experience

- OFFLINE MODE ✓
- BACKGROUND SYNC ✓
- INTER-APP COMMUNICATION ✓
- PAYMENTS ✓
- CREDENTIALS ✓

Try comparing your desktop computer and its default browser with your mobile phone and a browser like Chrome.

Because support varies across platforms, PWAs can gracefully degrade their available functionality to give users a seamless experience.

Performance

Application performance has a key impact on user engagement.

Research from Google shows that 53% of engagements are likely to be abandoned if applications take longer than 3 seconds to load.

In fact, if an application fails to respond in less than 1000 milliseconds (1 second), users lose focus on the task they are performing.

Performance translates directly into user satisfaction and longer engagement within an application, allowing users to complete business tasks easily and without friction.

Mobile web experiences have, historically, been criticised for poor performance.

However, the offline capability of PWAs, coupled with a focus on user experience, can achieve excellent results.

Cloud Native PWAs are deployed on highly scalable, resilient infrastructure, offering fast, low latency connections to minimise load times.

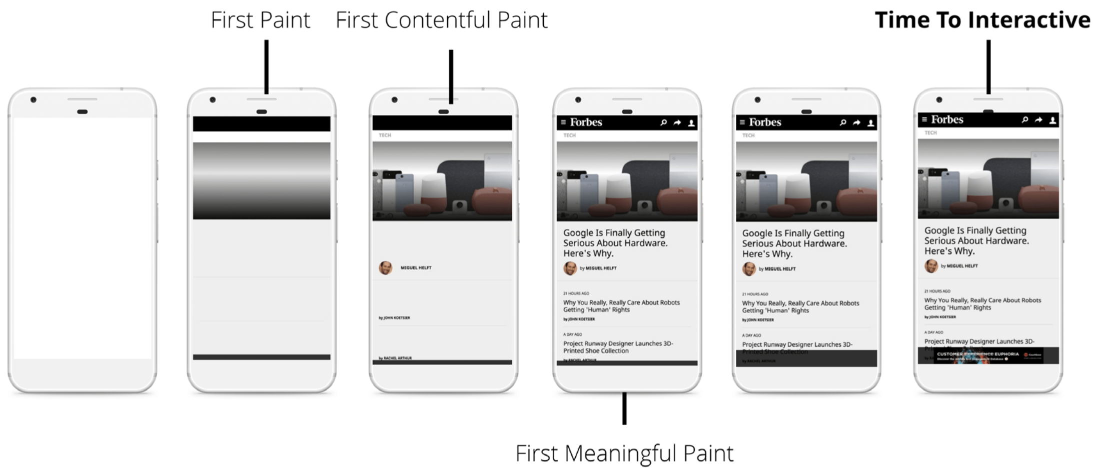
WebAssembly provides an additional mechanism, within the web platform, for native performance aimed at a number of use-cases.

Performance (Continued...)

RAIL is Response, Animation, Idle, Load.

It is a user-centric performance model that breaks down the user's experience into key actions.

RAIL's goals and guidelines aim to help developers and designers ensure a good user experience for each of these actions.
— *developer.google.com*



PWAs can comfortably meet the RAIL requirements:

Responsive:
Process events in under 50ms

Animation:
Produce a frame in 10ms

Idle:
Maximize idle time for background processing

Load:
Deliver content and become interactive in under 5 seconds

Diagram 2 - from https://developers.google.com/web/fundamentals/performance/rail

Discoverability

Progressive web applications are searchable, providing the reach of web, with the performance and capabilities of a native experience.

PWAs are linkable, using the browser's native navigation mechanisms; they can easily transport a user directly to a data-specific page in an app.

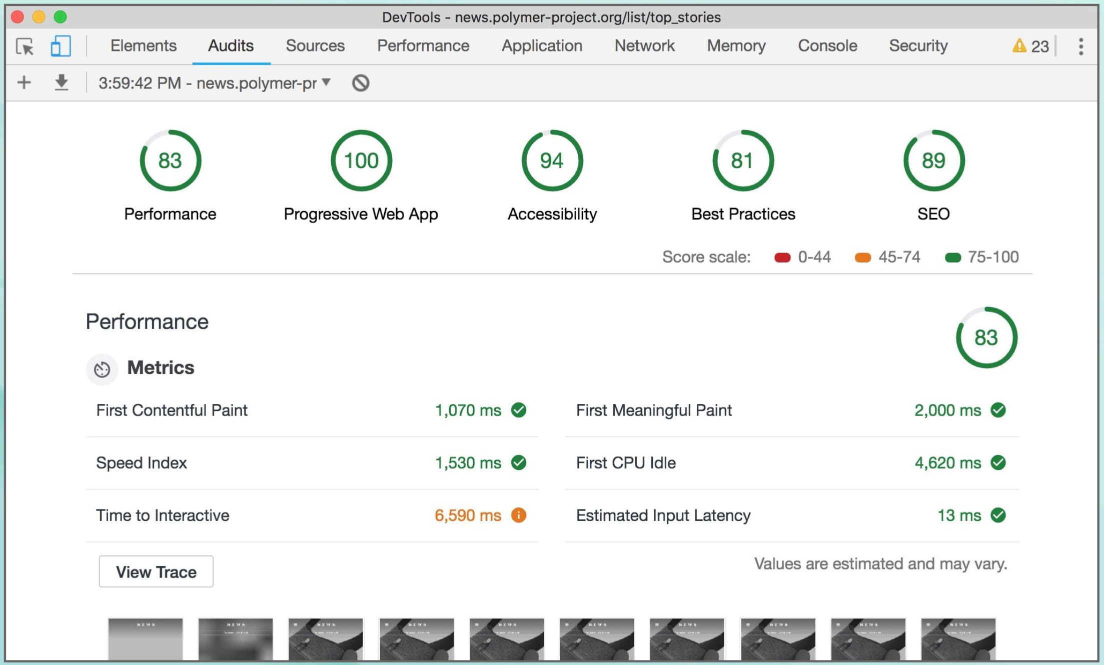
Links can be easily shared with others to extend the reach and utility to other mediums, such as email or instant message.

A key design principle for PWAs is graceful degradation, when features are unavailable in the browser, the application should respond and provide basic functionality and a consistent experience for all users.

Native applications are not inherently searchable or discoverable. To make a native application discoverable you need to build links into the Apple or Android stores and hope that users will follow the redirections.

Progressive web applications are easily discoverable. Whether through public search, such as Google, or through an internal enterprise search.

How Progressive is Your Web App?


DevTools - news.polymer-project.org/list/top_stories

Elements | Audits | Sources | Performance | Application | Network | Memory | Console | Security | ⚠ 23

3:59:42 PM - news.polymer-pr ▾

Performance	Progressive Web App	Accessibility	Best Practices	SEO
83	100	94	81	89

Score scale: 🟥 0-44 🟧 45-74 🟩 75-100

Performance 83

Metrics

First Contentful Paint	1,070 ms ✓	First Meaningful Paint	2,000 ms ✓
Speed Index	1,530 ms ✓	First CPU Idle	4,620 ms ✓
Time to Interactive	6,590 ms !	Estimated Input Latency	13 ms ✓

View Trace

Values are estimated and may vary.

Diagram 3 - An example of a Lighthouse report

Lighthouse is an open source tool released by Google to allow web developers to profile their own applications.

It measures how responsive, accessible, progressive and reliable a web application is.

It can be run inline with the build pipeline for a web application to allow teams to continuously tune the performance and reliability of their web application.

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

Typical PWA Architecture on AWS

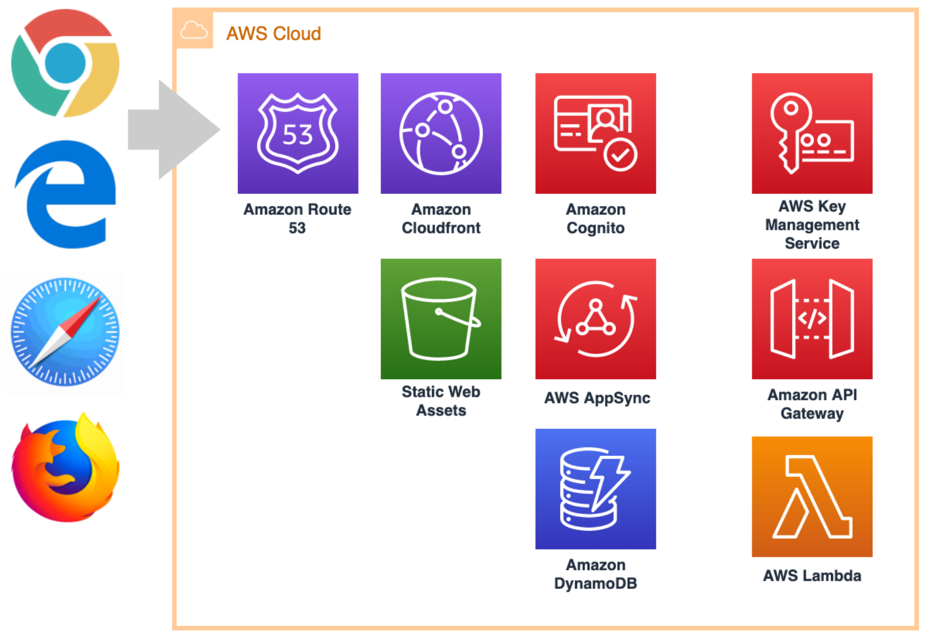


Diagram 4 - This example of a PWA Architecture describes a simple, user-friendly mobile interface to support maintenance technicians in the field.

It allows users to work offline, improve data entry integrity and reduce workforce frustration.

When network connectivity is detected, it will automatically synchronise with a cloud-based backend.

This will store and forward data to enterprise systems and the user will be notified of successful upload, allowing an asynchronous communication model.

Typical PWA Architecture on GCP

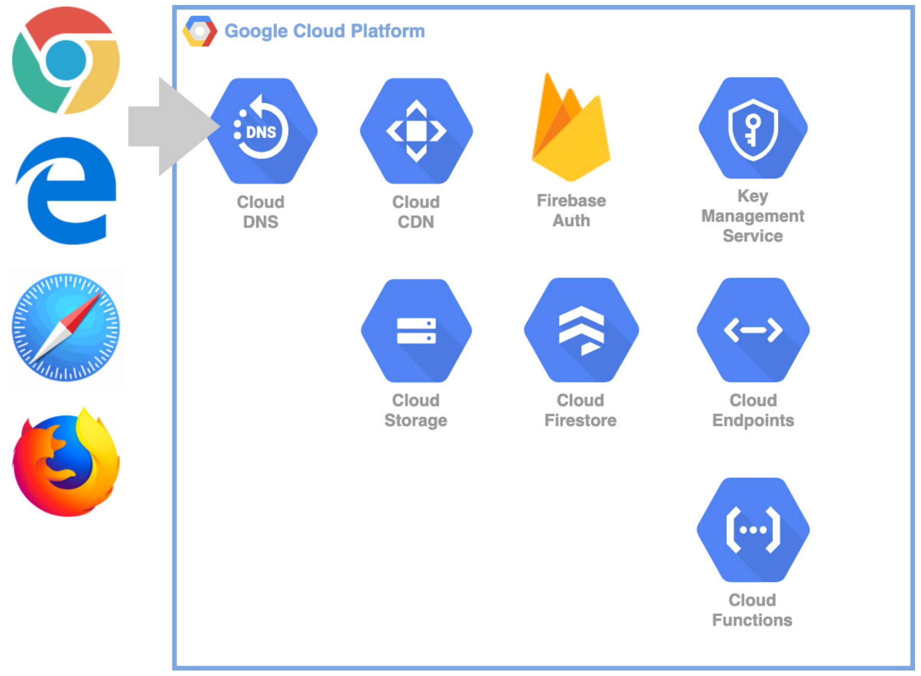


Diagram 5 - Here is an equivalent PWA architecture implemented on Google's Cloud Platform (GCP).

It utilises Google cloud products to provide backing services like DNS, authentication and database, with a compute layer provided by Google Cloud Functions.

This minimises the amount of work developers need to do to provide a reliable, scalable application and allows them to focus on the end-user and business logic.

Final Thoughts

Progressive Web Applications are an appropriate technology to deploy to the majority of mobility use-cases.

The centralised deployment model provides a number of benefits over native application deployments that should not be under-estimated.

Many perceived downsides to PWAs on mobile, such as increased battery usage and UI performance are caused by implementation specifics; comparisons are difficult to draw and examples can be found both for and against.

There are limitations that must be considered when undertaking a new PWA:

- Storage limits are more restrictive than for native applications requiring consideration.
- Support for PWAs does vary across platforms and browsers.

However, the benefits far outweigh the costs for almost all situations.

References

Progressive Web Apps - explained by Google.

WhatWebCanDoToday - to find out what features your browser supports.

Lighthouse - an open source tool to profile your web app for performance and reliability.

11 Examples of Progressive Web Apps - high profile examples of PWAs.

Build a PWA on AWS - an example of how to build a PWA using Amplify and AppSync.

Trivago embraces PWA's for the future - Trivago invest in PWAs to create a better, more stable mobile experience.

Device photography by Charles Deluvio and Hardik Sharma.