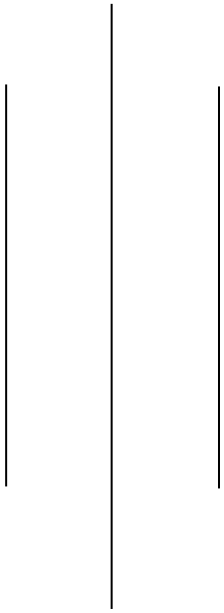
National College of Computer Studies

Paknajol, Kathmandu



Report on
Progressive Web Applications

Submitted By:

Siddhartha Shakya

Programme: BIM 1st Semester

Section: B

Roll No: 24

Email: sidd.shakya123@gmail.com

Submitted To:

Dadhi Ram Ghimire

TU Lecturer

Submitted On: 02/25/2024

Abstract

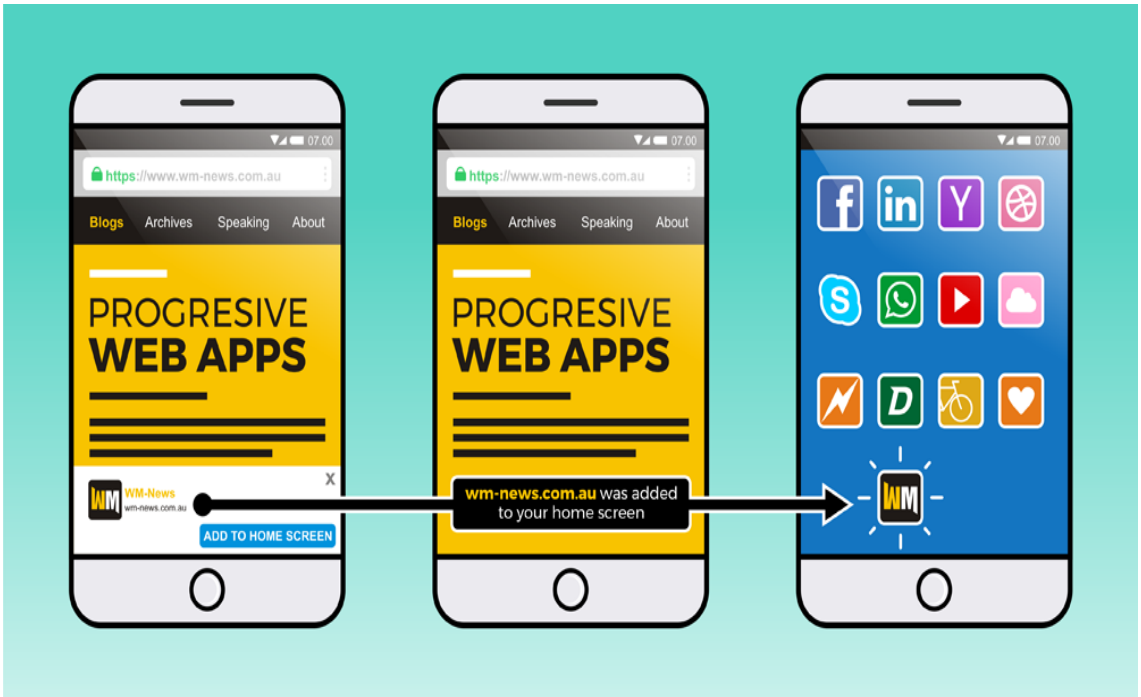
Progressive Web Apps (PWAs) are examined in great detail in this paper, which also includes information on their architecture, underlying technologies, benefits, and future trends. PWAs, which combine the best features of native and online applications, are a significant breakthrough that offer improved performance, strong offline functionality, and a richer user experience. The paper provides a clear and concise illustration of the practical benefits and extensive implementation of PWAs, based on real-world case studies from a variety of businesses. Additionally, it provides insightful analysis of cutting-edge technologies that have the potential to significantly influence how web applications develop in the future. This thorough yet comprehensive analysis highlights PWAs' significant role in fostering innovation within contemporary web development paradigms and gives readers a deep understanding of them.

Table of Contents

Chapter 1
Introduction

A progressive web app (PWA) is an app that uses web platform technology but delivers a platform-specific user experience.

A PWA, like a website, can run on many platforms and devices while using the same codebase. It acts in the background and offline, integrates with the device and other existing apps, and can be loaded on the device, much like a platform-specific app. [1]



1.1 Fig: Progressive Web App

1.1 History

Beginning in the early 2010s, dynamic web pages enabled web technologies to be leveraged to build interactive web apps. Responsive web design, and the screen-size flexibility it offers, have made PWA development more accessible. Continued developments to HTML, CSS, and JavaScript enabled online apps to incorporate higher levels of interactivity, allowing for native-like experiences on websites.[2]

In 2015, designer Frances Berriman and Google Chrome engineer Alex Russell coined the term "progressive web apps"[3] to describe apps that take advantage of new features supported by modern browsers, such as service workers and web app manifests, allowing users to upgrade web apps to progressive web apps in their native operating system. Google

then dedicated significant resources to supporting PWA development for Android.[4][5] Firefox added support for service workers in 2016, and Microsoft Edge and Apple Safari followed suit in 2018,[6][7] bringing service workers to all major operating systems.

By 2019, most desktop browsers, including Microsoft Edge (on Windows) and Google Chrome, supported PWAs.

In December 2020, Firefox for desktop abandoned the implementation of PWAs (specifically, removed the prototype "site-specific browser" configuration that had been available as an experimental feature). A Firefox architect noted: "The signal I hope we are sending is that PWA support is not coming to desktop Firefox anytime soon."[8] Mozilla still plans to support PWAs on Android.

1.2 Characteristics

Progressive web apps are all designed to function with any browser that meets the required web standards. The goal, like that of other cross-platform solutions, is to make it easier for developers to create cross-platform apps than native apps.[9] Progressive web apps use the progressive improvement web development philosophy.

Some progressive web applications employ an architectural approach known as the App Shell Model.[10] In this paradigm, service workers save the responsive web design web application's Basic User Interface (or "shell") in the browser's offline cache. This architecture enables PWAs to function similarly to native applications with or without web connectivity. This can shorten loading time by providing an initial static frame, framework, or architecture into which content can be loaded both gradually and dynamically.[11]

Chapter 2
Technologies

There are numerous technologies used to construct progressive web apps. A web application is deemed a PWA if it meets the installation requirements, works offline, and can be put to the device's home screen. To achieve this criterion, all PWAs must include at least one service worker and a manifest.

2.1 Installation Criteria

The technical baseline conditions for a site to be regarded a progressive web app and hence capable of being installed by browsers were detailed by Russell in a follow-up post [12] and modified since:[13][14]

- Originate from a secure location. Served over TLS, with no active mixed content. Progressive web apps must be served via HTTPS to protect user privacy, security, and content validity.
- Set up a service worker with a fetch handler. To develop programmable content caches, progressive web applications must leverage service workers. Unlike regular HTTP web caches, which cache content after the first use and then rely on heuristics to determine when it is no longer required, programmable caches can explicitly prefetch content before it is used for the first time and explicitly discard it when it is no longer required.[15] This requirement allows websites to be accessed offline or over low-quality networks.
- Refer to a web application manifest. The manifest must include at least five important properties: name or short name, start URL, display (with values of standalone, full screen, or minimal-ui), and icons (in 192px and 512px sizes). The information in the manifest makes PWAs easy to share via a URL, discoverable by search engines, and simplifies installation procedures.[16] Furthermore, PWAs enable native app-style interactions and navigation, such as adding to the home screen and displaying splash screens.

2.2 Applied Technologies

A. Manifest

The web app manifest [17] is a W3C specification that defines a JSON-based manifest (often named manifest.json)[18] to give developers with a centralized place to post metadata related with a web application, including:

- Name of the web application
- Links to web app icons or image assets.
- The ideal URL for launching or opening the web application
- The web app setup data.
- The default orientation of the web app
- The option to set the display mode, such as full screen.

B. IOS Support

iOS Safari partially implements manifests outside of the European Economic Area [19], although the majority of PWA metadata can be defined via Apple-specific meta tag extensions. Developers can use these elements to enable full-screen display, define icons and splash screens, and name the application.[20][21]

C. Web Assembly

WebAssembly enables precompiled code to run in web browsers at near-native speeds.[22] As a result, web apps can include libraries written in languages such as C.

D. Data Storage

Progressive web apps must keep the majority of their long-term internal state (user data, dynamically loaded application resources) in one of the following ways, while execution contexts are unloaded whenever possible.

E. Web Storage

Web Storage is a W3C-standard API that allows key-value storage in modern browsers. The API is made up of two objects: sessionStorage (which permits session-only storage that is deleted when the browser session ends) and localStorage (which allows storage to persist beyond sessions).[23]

F. Indexed Database API

Indexed Database API is a W3C-standard database API that is supported by all major browsers. Modern browsers implement the API, which allows for the storage of JSON objects and other structures that may be represented as strings.[24] The Indexed Database API can be used in conjunction with a wrapper library that adds additional functionality.

G. Service Workers

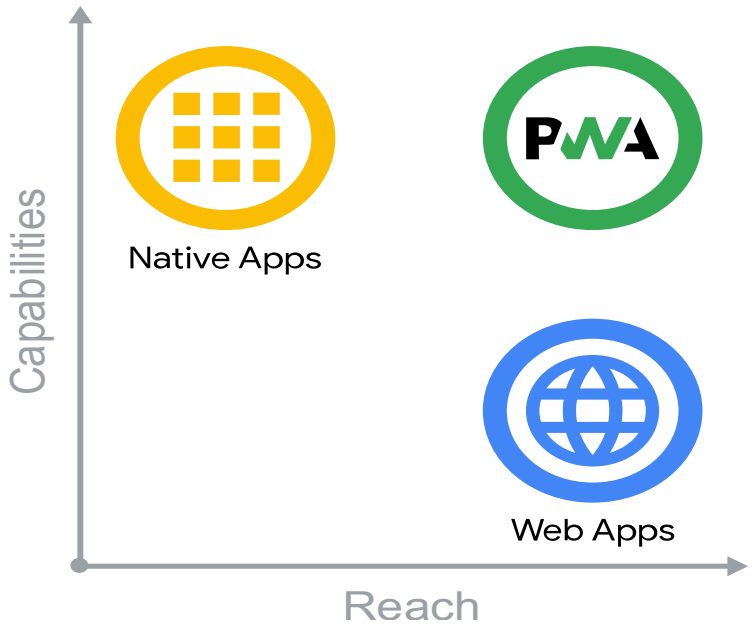
A service worker is a web worker that acts as a programmable network proxy, responding to web/HTTP requests from the main content. It can check the availability of a remote server, cache material when it is available, and serve that content later in the document. Service workers, like any other web worker, operate independently of the primary document context. Service workers may handle push notifications and synchronize data in the background, cache or retrieve resource requests, intercept network requests, and receive centralised updates regardless of whether the page that registered them is loaded.[25]

Service workers go through three steps: registration, installation, and activation. Registration entails informing the browser of the service worker's location in advance of installation. Installation takes place when the web app's service worker is not already installed in the browser or when the service worker is updated. Activation occurs when all of the PWA's pages are closed, ensuring that no conflict exists between the prior and updated versions. Because only one service worker can be active for a domain at a time, the lifecycle also aids in maintaining consistency when transitioning between service worker versions.[26]

Chapter 3
Comparison with native apps and web apps

The web is an amazing platform. The web is a unique platform for developing software due to its ubiquity across devices and operating systems, its user-centered security paradigm, and the fact that neither its specification nor its implementation is controlled by a single business. With its intrinsic linkability, you may search it and share your findings with anyone, anywhere. When you visit a website, it is always current, and your interaction with it can be as brief or as long as you like. Web apps can reach anybody, anywhere, and on any device using a single codebase.

Platform-specific applications are known for being incredibly rich and reliable. They're ever-present, on home screens, docks, and taskbars. They work regardless of network connection. They launch in their own standalone experience. They can read and write files from the local file system, access hardware connected via USB, serial port, or bluetooth, and even interact with data stored on your device, such as contacts and calendar events. In these applications, you can do things such as take pictures, see playing songs listed on the home screen, or control song playback while in another app. Platform-specific applications feel like *part* of the device they run on.



1.2 Fig: Comparison with native apps and web apps

When it comes to functionality and reach, platform-specific apps outperform web apps. So, where do progressive web apps fit in?

Progressive Web Apps (PWA) are created and extended using current APIs to provide improved capabilities, dependability, and installability while reaching anyone, anywhere, and on any device using a single codebase.

3.1 The three app pillars

Progressive Web Apps are web applications that are intended to be capable, reliable, and installable. These three pillars combine to create an experience that feels like it was designed for a certain platform.

- **Capable**

The web is quite capable on its own right now. You can use WebRTC, geolocation, and push notifications to create a hyper-local video chat app. You can make that app installable and virtualize its conversations using WebGL and WebVR. WebAssembly allows developers to leverage other ecosystems such as C, C++, and Rust, bringing decades of work and capabilities to the web. Squoosh.app, for example, employs this for advanced image compression.

Until recently, only platform-specific programmes could legitimately claim these characteristics. While some functions remain outside the web's reach, current and forthcoming APIs aim to change that by introducing functionality such as file system access, media controls, app badging, and complete clipboard support. All of these features are built on the web's secure, user-centric authorization paradigm, which ensures that visiting a website is never frightening for users.

- **Reliable**

A reliable Progressive Web App feels speedy and dependable regardless of network. Speed is crucial for encouraging users to use your experience. In fact, when page load times climb from one to ten seconds, the likelihood of a visitor bouncing rises by 123%. Performance continues after the onload event. Users should never worry if their interaction, such as clicking a button, was recorded or not. Scrolling and animations should be fluid. Performance influences the overall user experience, from how they perceive your application to how well it operates.

Finally, reliable programmes must be usable independent of the network connection. Users expect apps to launch on poor or flaky network connections, as well as when they are offline. They expect the most recent stuff they've dealt with, such as media tracks, tickets, and itineraries, to be available and usable, even if submitting a request to your server is difficult. When a request cannot be fulfilled, they want to be informed of the problem rather than failing or crashing silently.

- **Installable**

Installed Progressive Web Apps run in a separate window rather than a browser tab. They can be launched from the home screen, dock, taskbar, or shelf. You can search for them on your device and move between them using the app switcher, making them feel like they're a part of the device.

New features become available after a web app is installed. Keyboard shortcuts, which are typically reserved when running in the browser, become available. Progressive Web Apps can be configured to accept material from other applications or to be the default programme for handling various sorts of files.

When a Progressive Web App transitions from a tab to a standalone app window, it changes how users think about and interact with it.

3.2 The best of both worlds

Progressive online Apps are essentially just online applications. Modern browsers gain additional capabilities through progressive enhancement. Using service workers and a web app manifest, your web application becomes dependable and installable. If the new features are not available, users will still receive the basic experience.

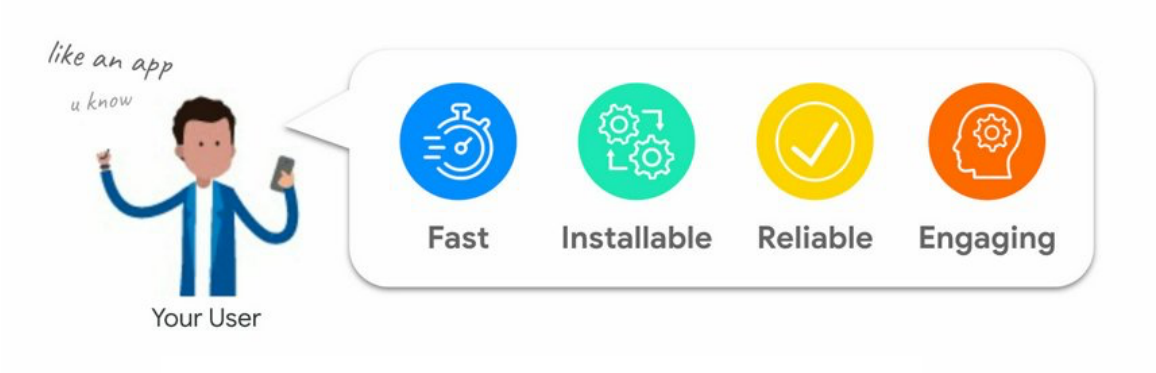
The numbers do not lie! Companies that have implemented Progressive Web Apps have achieved outstanding outcomes. Twitter, for example, witnessed a 65% boost in pages per session, 75% more Tweets, and a 20% decrease in bounce rate, all while lowering app size by more than 97%. Nikkei experienced 2.3 times more organic traffic, 58% more subscriptions, and 49% more daily active users after implementing a PWA. Hulu increased return visits by 27% after replacing their platform-specific desktop experience with a Progressive Web App.

Chapter 4
Business Success with PWA

Progressive Web Apps are on many firms' roadmaps to modernize their websites and meet customers' changing expectations. PWAs, like many technologies, bring questions: is it what my customers want, how much will it increase my business, and what is technically feasible?

4.1 PWAs satisfy client needs

According to Google, one rule they want to follow at Google when developing products is "focus on the user and everything else will follow." Think user-first: what are my customers' needs, and how might a PWA meet them?



1.3 Fig Identify customer needs

When conducting user research, we discovered several intriguing patterns:

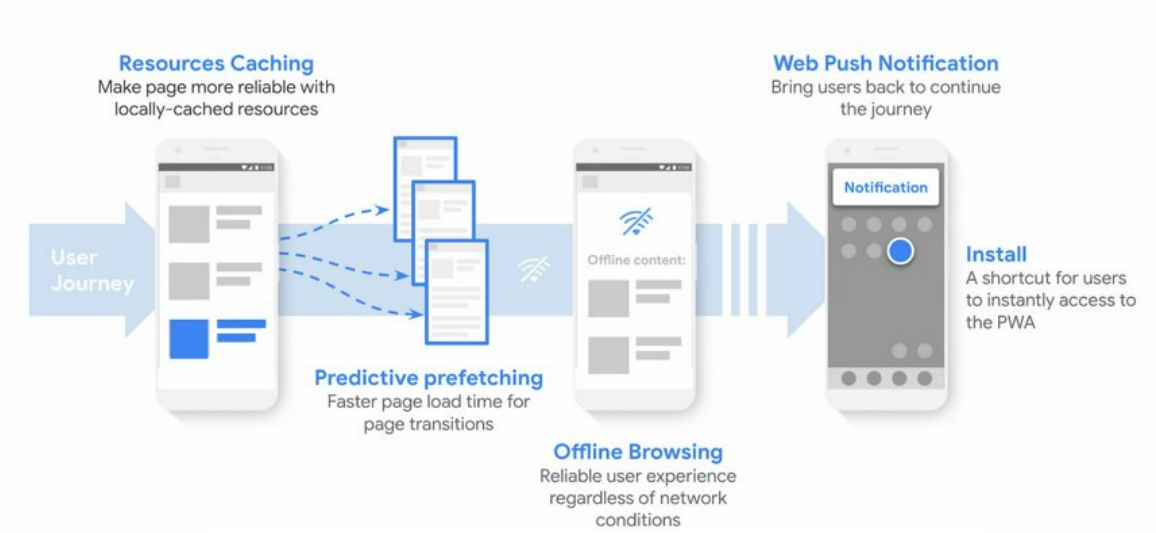
- Users despise delays and unreliability on mobile. The amount of tension generated by mobile delays is similar to viewing a horror film.
- Fifty percent of smartphone users prefer to browse or shop on a company's mobile site rather than download an app.
- One of the primary reasons for deleting an app is insufficient storage. (An installed PWA often requires less than 1MB).
- Smartphone users are more inclined to make purchases from mobile websites that provide relevant product recommendations, and 85% of smartphone users find mobile alerts useful.

According to our observations, users choose experiences that are fast, installable, dependable, and entertaining.

4.2 PWAs utilise contemporary online features

PWAs offer a collection of best practices and contemporary web APIs with the goal of satisfying the expectations of your users by making your website dependable, fast, installable, and visually appealing.

For instance, your website will load faster and be more dependable if you use a service worker to **cache your resources** [27] and do **predictive prefetching** [28]. Giving your website installability gives your users a simple method to access it straight from their home screen or app launcher. Additionally, using APIs like **Web Push Notifications** [29] makes it simpler to re-engage people with tailored content that fosters loyalty.



1.4 Fig: Enhancing the online experience for users.

4.3 Recognize the effects on business

Depending on what you do, the concept of business success can take many different forms:

- More users are spending time on your service
- A decrease in lead bounce rates
- increased conversion rates
- A higher number of repeat visitors

A higher mobile conversion rate is achieved by the majority of PWA initiatives. The many **PWA case studies** [30] have more information. It's totally acceptable to give more weight to the PWA features that make the most sense for your company, depending on your goals. It is possible to launch PWA features individually and selectively.

Conclusion

To sum up, Progressive online Apps (PWAs) bridge the gap between native mobile applications and regular websites, so representing a transformational approach to online development. Modern web technologies and principles are used by PWAs to provide users with a smooth and interesting experience on a variety of devices and network situations. PWAs have several benefits, as shown by their increasing acceptance by users and organizations alike. These benefits include enhanced user engagement, offline functionality, and better performance. Nevertheless, issues like discoverability and browser compatibility still exist and need constant attention from stakeholders and developers. However, PWAs have the unquestionable ability to completely change the digital world by ushering in a time when web experiences will be more dependable, approachable, and user-focused. As technological advancements persist, it is crucial for companies to adopt PWAs as a tactical advantage in their digital strategy, opening up fresh doors for expansion and creativity in the dynamic digital landscape.

References

1. https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps
2. Marcotte, Ethan (25 May 2010). "Responsive Web Design". *A List Apart*.
3. Russell, Alex. "Progressive Web Apps: Escaping Tabs Without Losing Our Soul".
4. Evans, Jonny (26 January 2018). "Apple goes back to the future with web apps". *Computerworld*.
5. Ladage, Aaron (17 April 2018). "Progressive Web Apps Are Here and They're Changing Everything".
6. "Can I use... Support tables for HTML5, CSS3, etc". *caniuse.com*.
7. Evans, Jonny (26 January 2018). "Apple goes back to the future with web apps". *Computerworld*.
8. Newman, Jared (2021-01-26). "Firefox just walked away from a key piece of the open web". *Fast Company*.
9. Evans, Jonny (26 January 2018). "Apple goes back to the future with web apps".
10. "The App Shell Model".
11. Osmani, Addi. "The App Shell Model | Web Fundamentals". *Google Developers*.
12. Russell, Alex. "What, Exactly, Makes a Progressive Web App".
13. "What does it take to be installable?". *web.dev*.
14. web.dev. "Progressive Web App".
15. "Service worker caching and HTTP caching". *web.dev*.
16. W3C "Web App Manifest", Working Draft
17. "Web Manifest Docs on MDN". *MDN Web Docs*.
18. W3C "Web App Manifest", Working Draft
19. Miller, Chance (2024-02-15). "Apple confirms iOS 17.4 removes Home Screen web apps in the EU, here's why". *9to5Mac*.
20. "What's new on iOS 12.2 for Progressive Web Apps". *Medium*.
21. "Configuring Web Applications". *Safari Web Content Guide*.
22. "WebAssembly Concepts". *MDN*.
23. "Web Storage API". *MDN*.
24. "Concepts behind IndexedDB". *MDN*.
25. "Introduction to Service Worker | Web". *Google Developers*. 1 May 2019.
26. "Introduction to Service Worker | Web". *Google Developers*. 1 May 2019.
27. https://web.dev/articles/service-workers-cache-storage

28. https://web.dev/articles/precache-with-workbox

[Chapter 3] “Comparison with native apps and web apps” retrieved from: https://web.dev/articles/what-are-pwas

[Chapter 4] “Business Success with PWA” retrieved from: https://web.dev/articles/drive-business-success

29. https://developer.mozilla.org/en-US/docs/Web/API/Push_API/Best_Practices

30. https://www.pwastats.com/