Progressive Web App (PWA): Introduction

Kabui, Charles

2025-04-23

Table of Contents

What Are Progressive Web Apps (PWAs)? . . . 3
PWAs vs. Other Cross-Platform Approaches . . . 3
    PWAs vs. React Native . . . 3
    PWAs vs. WebView-Wrapped Websites . . . 4
Key Capabilities of PWAs . . . 5
When to Start with a PWA Approach . . . 5
    1. Limited Development Resources . . . 5
    2. Rapid Time-to-Market Requirements . . . 5
    3. Rapid Prototyping and MVPs . . . 5
    4. Content-Centric Applications . . . 6
    5. SEO and Discoverability Priorities . . . 6
    6. Focus on Web Presence . . . 6
    7. Need for Frequent Updates . . . 6
When PWAs Can Completely Replace Native Apps . . . 6
    E-commerce Platforms . . . 6
    Media and Publishing . . . 7
    Online Magazines and News Portals . . . 7
    Business Tools and Enterprise Applications . . . 7
    Service-Based Applications . . . 7
    Basic Productivity Tools . . . 7
    Restaurant Ordering and Menu Apps . . . 7
    Event Information and Ticketing Platforms . . . 7
    Lightweight Social Media Clients . . . 8
Main Components of a Progressive Web App . . . 8
    Service Workers . . . 8
        Workbox . . . 8
    Web App Manifest . . . 9
Getting PWAs into App Stores . . . 9

Real-World PWA Success Stories 10
Twitter Lite 10
Starbucks 10
Pinterest 10
Uber 10
Limitations and When to Consider Native Apps 11
Hardware-Intensive Applications 11
Platform-Specific Features 11
App Store Presence Limitations 11
The Future of PWAs 11
Project Fugu 11
Desktop PWAs 11
WebAssembly Integration 12
Growing Browser Support 12
PWA-Ready Frameworks 12
Conclusion: The Pragmatic Approach to Cross-Platform Development 12

Figure 1: Progressive Web App (PWA)

Click to listen to the audio summary

Individuals and businesses face a crucial decision when developing applications: build separate native apps for each platform or pursue a unified approach? Progressive Web Apps (PWAs)

promise a "**build once, run anywhere**" methodology that can reduce development costs while delivering near-native experiences. This article explains what PWAs are, their benefits and challenges, and when to use or avoid them.

What Are Progressive Web Apps (PWAs)?

Progressive Web Apps [1] are normal web applications that use modern web capabilities to deliver app-like experiences to users. Coined by Google Chrome engineer Alex Russell in 2015 [2], PWAs combine the best of web and mobile apps:

- **Reliability**: Load instantly and function offline through service workers
- **Performance**: Fast response and smooth animations
- **Engagement**: Immersive full-screen experiences with push notifications

At their core, PWAs are built with standard web technologies (HTML, CSS, JavaScript) but leverage advanced APIs and development patterns to function like native applications, essentially bridging the gap between web and native experiences.

Progressive Web Apps follow the principle of progressive enhancement, functioning on basic browsers while providing enhanced experiences on capable devices. This ensures your application reaches the widest possible audience without exclusion.

PWAs vs. Other Cross-Platform Approaches

Before diving deeper into PWAs, it's important to understand how they differ from other cross-platform solutions:

PWAs vs. React Native

While both aim to solve the cross-platform development challenge, they differ fundamentally:

- **React Native** compiles to native code and uses native UI components, resulting in truly native apps that must be installed through app stores. It uses a JavaScript bridge to communicate between JS code and native components. From a single codebase, the developer is able to compile code to target the Web, iOS, and Android.
- **PWAs** run directly in browsers using web technologies and can be added to home screens, but they render using web views rather than native UI components. From a single codebase, the developer only targets the Web, and the web browser is responsible for rendering the code to target iOS and Android.

[1] Progressive web apps

[2] In 2015, designer Frances Berriman and Google Chrome engineer Alex Russell coined the term "progressive web apps"

Aspect	React Native	Progressive Web Apps (PWAs)
Build and Deployment	Requires separate builds and app store submissions for each platform	Single deployment with no app store approval processes
Performance	Typically offers better native performance for complex UI interactions	Relies on web technologies, may have slightly lower performance
Reach	Limited to users who download the app from app stores	Broader reach through web discoverability and instant access
Discoverability	App store-dependent, limited SEO benefits	Discoverable via search engines, leveraging standard web SEO
User Access	Requires installation from app stores	Accessible instantly via a browser, with optional home screen installation

PWAs vs. WebView-Wrapped Websites

Some developers create "hybrid apps" by simply wrapping a standard website in a native WebView container:

- **WebView Wrapping** takes an existing website and packages it inside a native app shell using tools like Cordova or Capacitor, with minimal additional native functionality.
- **PWAs** are specifically designed with app-like features (offline capability, home screen installation, push notifications) from the ground up.

Aspect	WebView-Wrapped Websites	Progressive Web Apps (PWAs)
Performance	Often suboptimal without PWA-specific optimizations	Optimized with service workers and caching
Offline Functionality	Limited or non-existent	Enabled through service workers
App Store Approval	Required for distribution	Not required, accessible directly via browsers
User Installation	Mandatory for access	Optional, can be added to home screens
Architecture	Standard website wrapped in a native container	App-shell architecture designed for speed and reliability

Key Capabilities of PWAs

PWAs have evolved dramatically since their introduction. Today's PWAs can:

- **Work offline** using service workers and caching strategies
- **Install on home screens** without app store gatekeeping
- **Send push notifications** even when the app isn't active
- **Access device hardware** including camera, microphone, geolocation, and more
- **Function across all modern browsers and devices**
- **Update automatically** in the background without user intervention
- **Provide secure experiences** via HTTPS
- **Adapt to any screen size** with responsive design

When to Start with a PWA Approach

1. Limited Development Resources

For startups and small teams with constrained budgets and technical resources, PWAs offer an efficient entry point. Rather than developing and maintaining separate codebases for iOS, Android, and web, a PWA allows you to serve all platforms from a single codebase.

2. Rapid Time-to-Market Requirements

When market timing is critical, PWAs enable faster deployment. Native app development typically requires: - Separate development tracks for each platform - App store approval processes (often taking days) - User-initiated updates

In contrast, PWAs: - Deploy instantly to all platforms simultaneously - Bypass app store reviews - Update automatically when users access the application

Basically, any website can be a PWA, including a static website.

3. Rapid Prototyping and MVPs

If you need to quickly validate an idea or launch a Minimum Viable Product with limited resources, a PWA offers a faster development cycle and broader initial reach compared to native apps.

In this age of "*vibe coding*" and using AI to create applications, its faster to create, deploy and maintain a PWA without alot of developer experience than a native application.

4. Content-Centric Applications

Applications where content consumption is the primary activity benefit significantly from PWAs. News sites, blogs, and documentation platforms can deliver excellent experiences without native implementation.

5. SEO and Discoverability Priorities

Unlike native apps buried in app stores, PWAs are discoverable through search engines. For businesses where organic discovery drives significant value, PWAs provide immediate advantages through standard web SEO practices.

Content in PWAs is indexed by search engines, allowing users to discover your application through search results.

6. Focus on Web Presence

If your core business or service is primarily web-based, extending that experience with app-like features through a PWA can be a natural and efficient progression.

7. Need for Frequent Updates

PWAs allow for easier and faster updates and deployments compared to the often lengthy process of releasing new versions on app stores. This facilitates quicker iteration based on user feedback.

When PWAs Can Completely Replace Native Apps

In several scenarios, PWAs can entirely eliminate the need for native implementations:

E-commerce Platforms

Retail giants like Alibaba and Flipkart have demonstrated that PWAs can deliver conversion rates comparable to or better than native apps. Alibaba's PWA increased conversions by 76% and engagement by 30%, while reducing development resources [3]

[3] Comprehensive Analysis of PWAs in E-commerce for 2025

Media and Publishing

For content-focused businesses, PWAs excel at delivering articles, videos, and interactive media. The Washington Post's PWA loads in under a second and has increased reader engagement by 500% [4].

Online Magazines and News Portals

Delivering articles, videos, and other content with offline access and push notifications makes PWAs ideal for publishers.

Business Tools and Enterprise Applications

Internal tools, CRM systems, and enterprise applications that don't require deep hardware integration can be perfectly served through PWAs, with the added benefit of centralized deployment and instant updates.

Service-Based Applications

Booking services, scheduling applications, and other service-based platforms where the primary interaction is form submission and information retrieval work exceptionally well as PWAs.

Basic Productivity Tools

Simple note-taking apps, checklists, and calculators that don't require extensive device hardware interaction are perfect PWA candidates.

Restaurant Ordering and Menu Apps

Allowing users to browse menus, place orders, and receive updates through a web link or home screen icon.

Event Information and Ticketing Platforms

Providing event details, schedules, maps, and ticket purchasing capabilities without requiring a dedicated app download.

[4] Why does The Washington Post's Progressive Web App increase engagement on iOS?

Lightweight Social Media Clients

Offering core social networking features without the storage overhead of native apps, as proven by Twitter Lite.

Main Components of a Progressive Web App

PWAs leverage two main components to deliver their capabilities:

Service Workers

The backbone of PWAs, service workers are JavaScript files that run separately from the main browser thread, intercepting network requests and enabling: - Offline functionality through strategic caching - Background syncing when connectivity is restored - Push notifications for re-engagement

```
// Simple service worker registration
if ('serviceWorker' in navigator) {
  window.addEventListener('load', () => {
    navigator.serviceWorker.register('/sw.js')
      .then(registration => {
        console.log('ServiceWorker registered with scope:', registration.scope);
      })
      .catch(error => {
        console.error('ServiceWorker registration failed:', error);
      });
  });
}
```

Workbox

Google's Workbox library simplifies service worker implementation with ready-made strategies for caching and offline functionality:

```
// Using Workbox for cache strategies
importScripts('https://storage.googleapis.com/workbox-cdn/releases/6.2.0/workbox-sw.js');

workbox.routing.registerRoute(
  ({request}) => request.destination === 'image',
  new workbox.strategies.CacheFirst({
    cacheName: 'images',
```

```
    plugins: [
      new workbox.expiration.ExpirationPlugin({
        maxEntries: 50,
        maxAgeSeconds: 30 * 24 * 60 * 60, // 30 Days
      }),
    ],
  })
);
```

Web App Manifest

The manifest.json file enables "add to home screen" functionality and controls how the app appears when launched:

```
{
  "name": "My PWA Application",
  "short_name": "MyPWA",
  "start_url": "/index.html",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#2196f3",
  "icons": [
    {
      "src": "icons/icon-192x192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "icons/icon-512x512.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ]
}
```

Getting PWAs into App Stores

While PWAs are inherently web-based, businesses can still maintain an app store presence by converting the Progressive Web App into an actual installable app. Developers can manually convert a PWA into an app through their favourite build tools, and non-developers can use

online tools like PWABuilder to generate a full app from just a link to a Progressive Web App. The generated app is ready to publish to **Google Play Store** or **iOS App Store** without further modifications.

This approach differs from traditional WebView wrapping as it's specifically optimized for PWAs, leveraging their existing capabilities rather than simply displaying a website in a native container.

Real-World PWA Success Stories

Twitter Lite

Twitter's PWA (mobile.twitter.com) reduced data usage by 70% and increased pages per session by 65% [5]

Starbucks

The Starbucks PWA is 99.84% smaller (233KB) than their iOS app, yet offers core functionality including the store locator, menu browsing, and ordering capabilities [6]. Their web orders increased after PWA implementation

Pinterest

After rebuilding as a PWA, Pinterest saw a 60% increase in core engagements and a 44% increase in user-generated ad revenue. Time spent on the mobile web increased by 40% [7] [8]

Uber

Uber's PWA loads in under 3 seconds on 2G networks and provides the core booking experience in less than 50KB of gzipped code, making it accessible even on low-end devices in emerging markets [9]

[5] Twitter Lite PWA Significantly Increases Engagement and Reduces Data Usage
[6] Starbucks ordering and store locator progressive web appStarbucks ordering and store locator progressive web app
[7] A Pinterest Progressive Web App Performance Case Study
[8] Pinterest PWA
[9] Building m.uber: Engineering a High-Performance Web App for the Global Market

Limitations and When to Consider Native Apps

While PWAs solve many cross-platform challenges, they do have limitations:

Hardware-Intensive Applications

Applications requiring intensive hardware access or low-level system integration may still need native implementations. This includes: - Advanced AR/VR applications - High-performance games - Applications requiring specialized hardware access

Platform-Specific Features

Some features remain exclusive to specific platforms or are better implemented natively: - iOS-specific design patterns and interactions - Deep integration with platform-specific services - Background processing with complex requirements

App Store Presence Limitations

Some business models rely on app store discovery and monetization. While PWAs can be listed in Google Play and Microsoft Store using tools like PWABuilder, Apple's App Store has more restrictions around web-based applications [10]

The Future of PWAs

The PWA ecosystem continues to evolve rapidly:

Project Fugu

Google's Project Fugu is closing the gap between web and native capabilities by introducing APIs for: - File system access - Contact picker - Shape detection - Web Bluetooth and USB - Web NFC - Bluetooth access - Shape detection API and more

Desktop PWAs

Major browsers now support installing PWAs on desktop operating systems, blurring the line between web and desktop applications. Companies like Microsoft are embracing PWAs for cross-platform desktop apps.

[10] Your app should include features, content, and UI that elevate it beyond a repackaged website. If your app is not particularly useful, unique, or "app-like," it doesn't belong on the App Store

WebAssembly Integration

The combination of PWAs with WebAssembly (WASM) enables high-performance computing directly in the browser, expanding the types of applications that can be delivered as PWAs. WebAssembly is compilling non-javascript code to run in the browser, such as C# or Go.

Growing Browser Support

Browser vendors continue to expand support for advanced web APIs, gradually eliminating the few remaining advantages of native applications.

PWA-Ready Frameworks

Most modern JavaScript frameworks offer PWA capabilities built-in or through plugins: - React with Create React App - Vue with the PWA plugin - Angular with Angular Service Worker - Next.js with next-pwa

Conclusion: The Pragmatic Approach to Cross-Platform Development

Progressive Web Apps represent a pragmatic solution to the cross-platform development challenge. While not a universal replacement for native apps in every scenario, PWAs offer compelling advantages:

- **Cost efficiency** through unified codebases
- **Broad accessibility** across devices and networks
- **Immediate updates** without app store approval
- **SEO benefits** native apps can't match
- **Reduced user friction** with no installation requirements
- **App store presence**

As web capabilities continue to expand and browser support improves, the gap between PWAs and alternative approaches like React Native or WebView-wrapped sites will continue to narrow. For many businesses, starting with a PWA approach and selectively implementing native components only where necessary represents the most efficient path to reaching users across the digital ecosystem.

The question is no longer whether to build a PWA, but whether and where you need anything beyond one. To get a feeling of what PWA can do, you can check out "What PWA Can Do Today".

For completeness sake, let's consider the following example:

Imagine you're building a taxi service. You'll need two separate apps—one for drivers and one for customers—because the driver's app has extra responsibilities. It handles cost calculations, constantly tracks GPS, enforces location access, and guards against fraud by verifying timestamps and positions. It also has to work reliably across many device models and operating system versions, which often break or change with updates. In contrast, the customer app only needs to let people request rides, so it can be a simple Progressive Web App. That means anyone with an internet-connected device—no matter how old or what OS—can book a taxi without requiring them to install or update a native app. It's far easier to ask drivers (a small, controlled group) to keep their phones upgraded than to make every potential rider do the same.
