Progressive Web Apps

Progressive web applications (PWA) are web applications that are regular web pages or websites, but can appear to the user like traditional applications or native mobile applications.

The application type attempts to combine features offered by most modern browsers with the benefits of a mobile experience. [[1]](#footnote-1)

Historically (circa 2007- 2015) native apps provided a better user experience both through speed and computational ability.

Starting in 2010 continued advancements in HTML, CSS and JavaScript, along with more capable browsers and processers made hybrid-apps a viable alternative.

In 2015, Frances Berriman and Alex Russell coined the term Progressive Web Apps to describe apps taking advantage of new features supported by modern browsers.

## The characteristics that make up a progressive web app include:

* Progressive - Work for every user, regardless of browser choice because they’re built with [progressive enhancement](https://en.wikipedia.org/wiki/Progressive_enhancement) as a core tenet.
* Responsive - Fit any form factor: desktop, mobile, tablet, or forms yet to emerge.
* Connectivity independent - [Service workers](https://en.wikipedia.org/wiki/Progressive_Web_Apps#Service_Workers) allow work offline, or on low quality networks.
* App-like - Feel like an app to the user with app-style interactions and navigation.
* Fresh - Always up-to-date thanks to the service worker update process.
* Safe - Served via HTTPS to prevent snooping and ensure content hasn’t been tampered with.
* Discoverable - Are identifiable as “applications” thanks to W3C manifests[[6]](https://en.wikipedia.org/wiki/Progressive_Web_Apps#cite_note-w3cmanifest-6) and service worker registration scope allowing search engines to find them.
* Re-engageable - Make re-engagement easy through features like [push notifications](https://en.wikipedia.org/wiki/Push_technology).
* Installable - Allow users to “keep” apps they find most useful on their home screen without the hassle of an app store.
* Linkable - Easily shared via a URL and do not require complex installation.

## What’s the difference between a PWA and a standard mobile website?

* Originate from a secure origin. Served over TLS and green padlock displays (no active mixed content).
* Load while offline (even if only a custom offline page). By implication, this means that progressive web apps require service workers.
* Reference a web app manifest with at least the four key properties: name, short\_name, start\_url, and display (with a value of standalone or fullscreen)
* An icon at least 144×144 large in png format. E.g.: "icons": [ { "src": "/images/icon-144.png", "sizes": "144x144", "type": "image/png" } ]

## Commonly used technologies

### Manifest

The web app manifest is a [W3C](https://en.wikipedia.org/wiki/World_Wide_Web_Consortium) specification defining a [JSON](https://en.wikipedia.org/wiki/JSON)-based manifest[[6]](https://en.wikipedia.org/wiki/Progressive_Web_Apps#cite_note-w3cmanifest-6) to provide developers a centralized place to put [metadata](https://en.wikipedia.org/wiki/Metadata) associated with a web application including:

* The name of the web application
* Links to the web app icons or image objects
* The preferred URL to launch or open the web app
* The web app configuration data for a number of characteristics
* Declaration for default orientation of the web app
* Enables to set the display mode e.g. full screen

By setting and manipulating the metadata for the web manifest file, developers enable user agents to create seamless native-like mobile experiences through the progressive web app.

## Reliable connectivity with Service Workers

## What is a service worker?

A script that your browser runs in the background, separate from a web page, opening the door to features that don’t need a web page or user interaction. [[2]](#footnote-2)

## Some Examples

* Push Notifications  
  Web push notifications allow users to opt-in to timely updates from sites they love and allow you to effectively re-engage them with customized, relevant content.[[3]](#footnote-3)  
  **Note:**Unless you're using localhost, the Push API requires HTTPS.

* Background Sync  
  Background sync is a new web API that lets you defer actions until the user has stable connectivity. This is useful for ensuring that whatever the user wants to send, is actually sent.[[4]](#footnote-4)

Technically, service workers provide a scriptable network proxy in the web browser to manage the web/HTTP requests programmatically. The service workers lie between the network and device to supply the content. They are capable of using the cache mechanisms efficiently and allow error-free behavior during offline periods.

* Properties of service workers
* Trigger and keep alive by the relationship to the events, not by the documents
* Generic in nature
* Event-driven with time limit scripting contexts and running at the origin
* With natural endpoints for a wide range of runtime services
* Have a state
* With a script URL
* Containing registration
* Allocated ID or UUID
* With lifecycle events
* Have script resource map
* Can skip waiting for the flags

### Benefits of service workers

* Capable of handling the push notification easily
* Synchronize data in the background
* Capable of responding to the resource requests originate elsewhere
* Receive centralized updates

### Application shell architecture

Some progressive web apps use an architectural approach called the App Shell Model.

For rapid loading, service workers store the Basic User Interface or "shell" of the [responsive web design](https://en.wikipedia.org/wiki/Responsive_web_design) web application. This shell provides an initial static frame, a layout or architecture into which content can be loaded progressively as well as dynamically, allowing users to engage with the app despite varying degrees of web connectivity.

Technically, the shell is a code bundle stored locally in the browser cache of the mobile device. [[5]](#footnote-5)

**A common problem for the web:** Latency when trying to achieve a specific goal from a web app or website.

Service workers solve the page loading part by letting you serve content from a cache.

But what about when the page needs to send something to the server?

Background sync allows for communication to the server to happen in the background.

## Service Worker Life Cycle

Lifecycle is separate from your web page

* Register it with JavaScript
* Typically cache static assets
* Activation step – a moment to deal with old caches
* After activation, the service worker will control all pages in scope

Must have HTTPS

Some examples here: <https://developers.google.com/web/fundamentals/primers/service-workers/>

## PWA Performance

sites loading within 5 seconds had 70% longer sessions, 35% lower bounce rates, and 25% higher ad viewability than sites taking nearly four times longer at 19 seconds

An effective method of building high performance applications is to audit what you send to users. [[6]](#footnote-6)

## A few suggestions to increase performance

What resources are you sending?

* Do you need Bootstrap?
* Is there a smaller alternative to jQuery?
* Not all websites need to be Single page applications – JavaScript is typically the most expensive resource, as it has to be downloaded, parsed, compiled, and executed.

How are you sending resources?

* Migrate to HTTP/2[[7]](#footnote-7)
* Expedite the delivery of resources – rel=preload[[8]](#footnote-8)

How much data are you sending?

* Minify text
* Configure server to compress resources – GZIP
* Optimize Images
* Consider WebP[[9]](#footnote-9) or JPEG XR[[10]](#footnote-10)
* Deliver images responsively[[11]](#footnote-11)
* Utilize client hints[[12]](#footnote-12)
* Utilize Network Information API[[13]](#footnote-13)

<https://developers.google.com/web/progressive-web-apps/>

1. <https://en.wikipedia.org/wiki/Progressive_Web_Apps> [↑](#footnote-ref-1)
2. <https://developers.google.com/web/fundamentals/primers/service-workers/> [↑](#footnote-ref-2)
3. <https://developers.google.com/web/fundamentals/push-notifications/> [↑](#footnote-ref-3)
4. <https://developers.google.com/web/updates/2015/12/background-sync> [↑](#footnote-ref-4)
5. All from Wikipedia Page cited above [↑](#footnote-ref-5)
6. <https://developers.google.com/web/fundamentals/performance/why-performance-matters/> [↑](#footnote-ref-6)
7. <https://developers.google.com/web/fundamentals/performance/http2/> [↑](#footnote-ref-7)
8. <https://www.igvita.com/2015/08/17/eliminating-roundtrips-with-preconnect/> [↑](#footnote-ref-8)
9. <https://developers.google.com/speed/webp/> [↑](#footnote-ref-9)
10. <https://jpeg.org/jpegxr/index.html> [↑](#footnote-ref-10)
11. <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/img#attr-srcset> [↑](#footnote-ref-11)
12. <http://httpwg.org/http-extensions/client-hints.html> [↑](#footnote-ref-12)
13. <https://developer.mozilla.org/en-US/docs/Web/API/NetworkInformation> [↑](#footnote-ref-13)