Progressive Web Applications

Story time...

What is a Progressive Web Application?

A Progressive Web Application is a term used to describe a web app that takes advantage of the latest technologies to combine the best of web and mobile apps.

The term was coined by Frances Berriman and Google Chrome engineer Alex Russell in 2015.

Think of a PWA as a web application built using web technologies but behaving like a native app.

Why Progressive Web Applications?

20%

A study has shown that, on average, a native app loses 20% of its users for every step between the user's first contact with the app and the user starting to use the app. The app requires space on the device, downloading and updating.

When a user finds the PWA they can start using it straight away, no downloading and no installing! And when the user returns to the app, they will be prompted to install the app and upgrade to a full-screen experience.

Native Mobile Apps aren't all bad...

Mobile applications with push notifications achieve up to three times more retention than their counterparts without push, and a user is three times more likely to reopen a mobile application than a website.

In addition, a well-designed mobile application consumes less data and is much faster because some resources reside on the device.

A PWA takes advantage of a mobile app’s characteristics, resulting in improved user retention and performance, without the complications involved in maintaining a mobile application.

What makes a Progressive Web Application?

Progressive - Works for every user, regardless of browser choice because it's built with progressive enhancement as a core tenet.

Responsive - Fits any form factor: desktop, mobile, tablet, or whatever is next.

Connectivity independent - Enhanced with service workers to work offline or on low-quality networks.

App-like - Feels like an app, because the app shell model separates the application functionality from application content .

Fresh - Always up-to-date thanks to the service worker update process.

Safe - Served via HTTPS to prevent snooping and to ensure content hasn't been tampered with.

Discoverable - Is identifiable as an "application" thanks to W3C manifest and service worker registration scope, allowing search engines to find it.

Re-engageable - Makes re-engagement easy through features like push notifications.

Installable - Allows users to add apps they find most useful to their home screen without the hassle of an app store.

Linkable - Easily share the application via URL, does not require complex installation.

What makes a Progressive Web Application?

The 3 technologies of Progressive Web Applications

Web App Manifest

A web app manifest file is a simple JSON file.

It follows the W3C’s specification and provides developers a centralized place to put metadata associated with a web application such as the name of the web app, links to the web app icons or image objects, the preferred URL to launch or open the web app and much more.

In addition, Chrome on Android will proactively suggest that the user install the web app, via a web app install banner.

App Manifest Example

```json
{
    "short_name": "Critic",
    "name": "Food Critic",
    "description": "Progressive web application",
    "icons": [
        {
            "src": "launcher-icon-256.png",
            "sizes": "256x256",
            "type": "image/png"
        }
    ],
    "start_url": "./?utm_source=web_app_manifest",
    "display": "standalone",
    "orientation": "portrait",
    "theme_color": "#29BDBB",
    "background_color": "#29BDBB"
}
```

Service Workers

Service Workers provide such features as offline functionality, push notifications, background content updating, content caching, and much more.

It's purely a script that works behind the scenes, independent of your app, and runs in response to domain-wide events like network requests, push notifications, connectivity changes, and more.

A Service Worker in a sense is a Proxy. It sits between the application and the network.

We can use Service Workers to cache application content and store it in browser based storage such as IndexedDB or WebSQL then retrieve this content if the application has no or poor connectivity.

Alternatively, the Service Work can show the app shell and inform the user that they are not connected to the Internet. Once the user reconnects, we can then retrieve the latest data from the server.

The App Shell

The app's shell is the minimal HTML, CSS, and JavaScript that is required to power the user interface of a progressive web app and is one of the components that ensures reliably good performance.

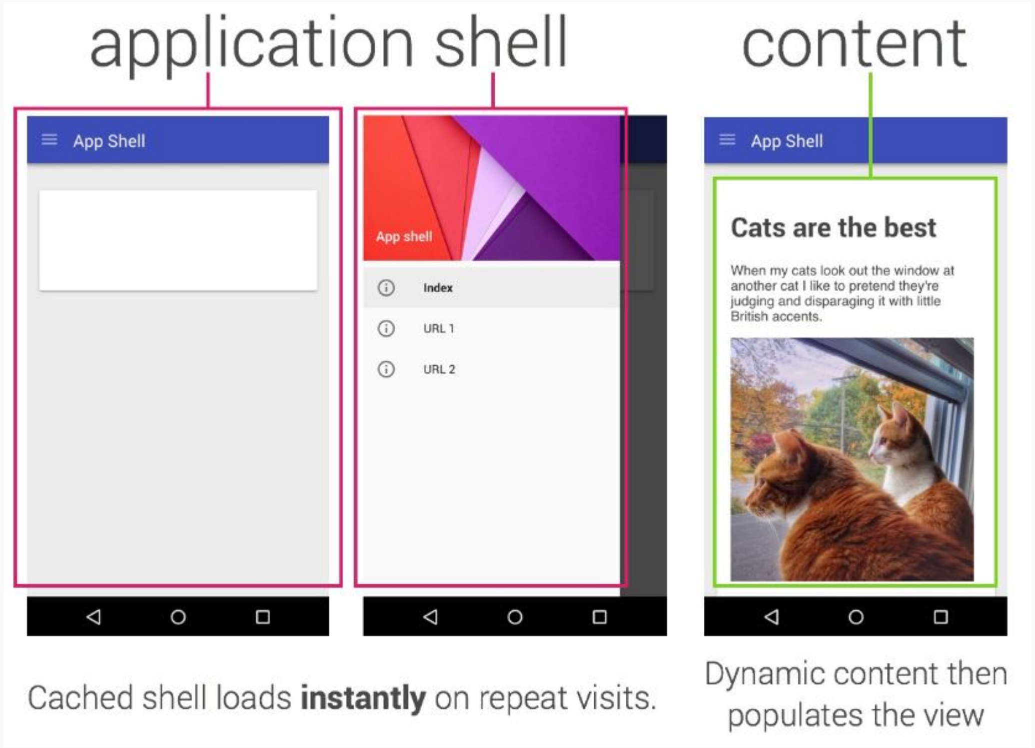
Its first load should be extremely quick and immediately cached.

Every subsequent time that the app is opened, the shell is loaded from the local device's cache, which results in blazing-fast startup times.

Google suggest using an App shell architecture as it separates the core application infrastructure and UI from the data.

Why do Google advise this…as it allows you to focus on speed, giving your Progressive Web App similar properties to native apps: instant loading and regular updates, all without the need of an app store.

The App Shell



Cats are the best

When my cats look out the window at another cat I like to pretend they're judging and disparaging it with little British accents.

Cached shell loads **instantly** on repeat visits.

Dynamic content then populates the view

A couple of demos...

[App Shell]

[PWA Concept]