International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 06 | June-2016 www.irjet.net p-ISSN: 2395-0072

Progressive WEBAPP : Review

Rahul Surendra Mishra

Master of Computer Application
VESIT, Chembur, Mumbai
University of Mumbai

Abstract - *Mobile apps have now found everywhere . Google has proposed a way to have one app be equal on both web and on mobile devices – Progressive Web Apps. Such apps take advantage of the modern web and browser capabilities to provide a full native app experience on any form factor. Progressive apps load quickly even on slow network connections, send push notifications, and have a splash screen and an icon on the home screen. When launched from the home screen, these apps blend into the environment; they're top-level, full-screen, and work offline. Progressive web apps are an interesting forward look into the future of mobile apps.*

Hybrid mobile development with a single Android/iOS codebase using HTML3, CSS3, JavaScript and various frameworks (Cordova/PhoneGap, Ionic, NativeScript, etc.) can aid is reducing some of the associated costs. However, hybrid mobile is not yet mature enough to fully replace native mobile apps. They often don't provide the proper native experience across the entire functional space. This dissonance has caused some to look for a better way. One idea, originating from Google, defines a new take on a mobile app called a "Progressive Web App."

Key Words*: *WebApp, progressive, offline, app install banners, Application shell, Service workers, Flipkart lite.

1. INTRODUCTION

A progressive web app combines the best experiences of the web and an app. They don't require any installation. The word 'progressive' comes from the relationship that the user builds with the app over time. The app loads quickly, even when the user is on bad networks. It can send relevant push notifications to the user and has an icon on the home screen and loads as top-level, full screen experience.

The main characteristics of a progressive web app are:[1]

Progressive – Work for every user on all browsers.
Responsive – Operate seamlessly across all form factors.
Connectivity independent – Work offline or on low quality network connections.
App-like – App-style interactions and navigation.
Fresh – Always up-to-date.
Secure – Served only via HTTPS.
Discoverable – Are identifiable as "applications," allowing search engine discovery.
Re-engageable – Make user re-engagement easy through features like push notifications.
Installable – Allow users to easily "keep" apps they find most useful on their home screen.
Linkable – Easily share via URL with no app store installations required.

2. Background

The mobile Web existed for years as a subset of the World Wide Web that was trimmed down, slow and ugly. The mobile Web existed in WAP and m.website.com pages that would load on limited smartphone and tablet browsers that could not handle the full Web.

For a few years it looked like the old, dirty mobile Web was going to die. Adaptive and responsive design came to make full websites look good on mobile with rich and immersive experiences. The "mobile" bit was going to be stripped out and all we were left with was the Web, in all its glory, from any device we decide to access it.But it now looks like the mobile Web is making a comeback. Instead of breaking down barriers between the mobile Web and the full Web, a group of technology companies is working to try and make the mobile version of the Web faster.

Apps are fast and the mobile websites are slow. In 2015, this particular problem has been one of the prime conversations in Web and app publishing and development. A new architecture is coming that will help bridge the gap between performance of Web and native apps and may finally provide the solution to building apps and websites that are fast and reliable for the mobile age.

In a nut shell, progressive Web apps start out as tabs in Chrome and become progressively more "app" like the more people use them, to the point where they can be pinned on the home screen of a phone or in the app drawer and have

access to app-like properties such as notifications and offline use. Progressive Web apps are linkable with an URL, fully responsive and secure.[3]

3. Why WEBAPP

In September 2015, research firm comScore released an extraordinary survey about how people actually use websites and apps. From an engagement perspective, the Web is a mile wide and an inch deep. Apps are the opposite, an inch wide and a mile deep. Apps are fast and the mobile websites are slow.

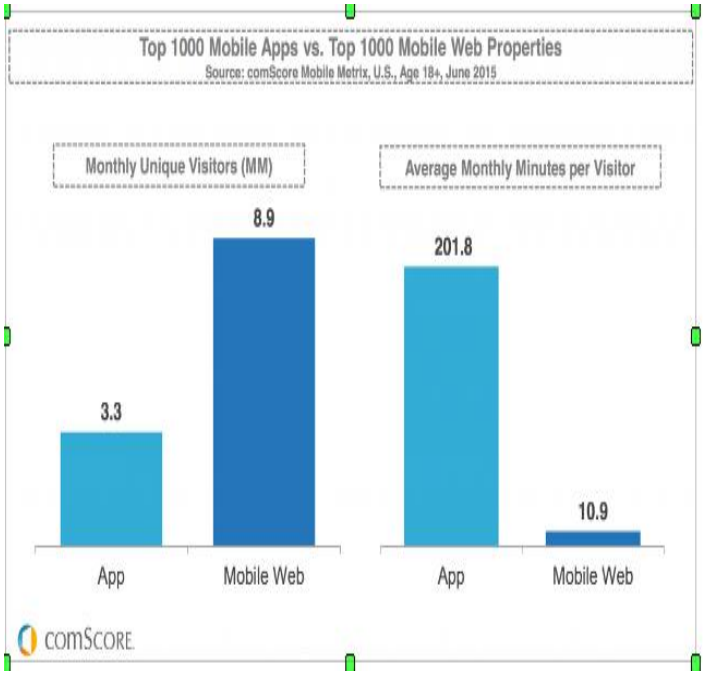

***Chart-1:** App & Mobile Web usage*

The Web reaches wider audiences than apps. But apps dominate in time spent. So there is need of something that combines the best experiences of the web and the app. A new architecture is coming that will help to combine the best experiences of the web and the app and may finally provide the solution to building apps and websites that are fast and reliable. With Progressive Web Apps, Google has seen engagement levels of websites approach nearly that of native apps.[4]

4. Architecture

For speed and functionality, progressive Web apps rely on two functions: Application Shell Architecture and Service Workers.

4.1 Application Shell Architecture:

An application shell is the minimal HTML, CSS, and JavaScript powering a user interface. The application shell should:

- *Load fast*
- *Be cached*
- *Dynamically display content*

An application shell is the secret to reliably good performance. Think of your app's shell like the bundle of code you'd publish to an app store if you were building a native app. It's the load needed to get off the ground, but might not be the whole story. It keeps your UI local and pulls in content dynamically through an API.[5]

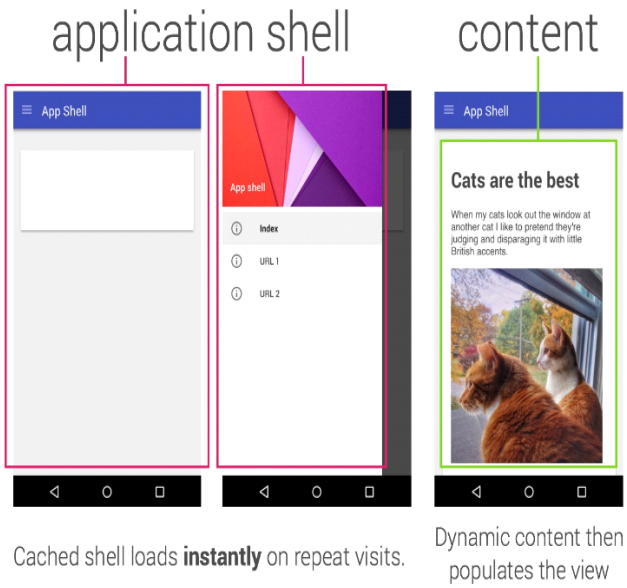

***Fig-1:** Application shell and content [6]*

In general the application shell architecture will:
Prioritize the initial load, but let service worker cache the application shell so repeat visits do not require the shell to be re-fetched from the network.
Lazy-load or background load everything else. One good option is to use read-through caching for dynamic content.
Use service worker tools, such as sw-precache, for example to reliably cache and update the service worker that manages your static content.

4.2 Service Workers:

A service worker is a script that runs in the background, separate from your web page. It responds to events, including network requests made from pages it serves and push notices from your server. A service worker has an intentionally short lifetime. It wakes up when it gets an event and runs only as long as it needs to process it.[6]

Service workers also have a limited set of APIs when compared to JavaScript in a normal browsing context. This is standard for workers on the web. A Service worker can't access the DOM but can access things like the Cache API, and they can make network requests using the Fetch API. The IndexedDB API and postMessage() are also available to use for data persistence and messaging between the service worker and pages it controls. Push events sent from your server can invoke the Notification API to increase user engagement.

A service worker can intercept network requests made from a page (which triggers a fetch event on the service worker) and return a response retrieved from the network, or retrieved from a local cache, or even constructed programmatically. Effectively, it's a programmable proxy in the browser. The neat part is that, regardless of where the response comes from, it looks to the web page as though there were no service worker involvement.

Service Workers are a way to increase Web app performance by helping to cache and deliver content and background functionality (like push notifications). Service workers can make sites work offline or help speed up the content by, "intercepting network requests to deliver programmatic or cached responses."[7]

5. Comparative Study

5.1 Native Applications:

Native applications have codes that are devised specifically for a particular platform, namely Android, iOS and so on. Cross-platform codes are not possible with native apps as the codes written for one platform cannot be used in another. You may use the latest APIs, but you cannot use one platform's code on another one. A native app would always feel right for the user because it has a mature ecosystem containing all the specific guidelines used for the OS it is developed for; ranging from swipes, app defines gestures to centrally aligned headers for iOS and left aligned headers for Android. This makes it easier for the user.

5.2 Web Apps:

Mobile web applications require web browsers to function and they are developed using HTML ,CSS and JavaScript. The program will be stored on a remote server and shows itself when the user asks for it. It is not necessary for web apps to have native codes and they can function on any operating system.

5.3 Hybrid App:

When smartphones were first released into the market, the war between native and web applications took shape and form, but along with the spoils of the war, another category was created – hybrid apps. If you want to get an app out the door as soon as possible and save time and money on developing the app, then you need hybrid apps. Hybrid app development is cross-platform app development and only one source code is used; this would be upgraded and updated to suit the purpose. Thus it combines the benefits of both native apps and web apps.

***Table-1:** Comparative factors*

Factors	Native	Web	Hybrid
Code Portability	Not Possible	Possible, but poses difficulties sometimes	Possible with many codes
Local Storage, Offline Capability	Possible	Possible	Very less possibility
Monetization	Highly Possible	Very Less Possibility	Highly Possible
Cost	Comparatively High	Not very Expensive	Not Expensive
Time to Market	Takes time	Takes very less time	Takes time
User Experience & Interaction	Very Good	Good	Good
Internationalization & Localization	Win-win	Win-win	Win-win

5.4 Comparative factors:

5.4.1 Code Portability:

You can not port native apps from one platform to another. With web apps, you can have a single code base for any major mobile platform. This is not 100% portable and sometimes developers are faced with portability issues. For hybrid apps, you can reuse many of the apps from one platform to another.

5.4.2 Local Storage, Offline Capability:

Offline apps would function even when your user is not online. There is no need for the internet connection to be

constant. Local storage that retains web app data is possible with web apps, and thus it would be ideal to use web apps if you are looking for offline storage of 5MB at a time. With hybrid apps, users cannot enjoy the offline mode as much as they would want. With native apps, it would be possible for users to enjoy the capabilities of offline capability.

5.4.3 Monetization:

Every app developer seeks to come up with a ground-breaking concept when they release their app to the app store. They all expect it to bring in phenomenal success in terms of money. The possibility of monetization with native and hybrid apps would be much higher compared to web apps. The downside for native and hybrid apps is that the app store takes a percentage of all sales that you make. Whereas, for web apps, there are no commissions.

5.4.4 Cost:

Native apps often cost more to develop because they ask for specific language and tooling ecosystems, apart from the customization of code. Cost is often dependant on a number of factors and sometimes even the skill of a developer which could cost more. Web app's functionality is based on JavaScript, CSS and HTML5. Hybrid apps are least expensive of all three.

5.4.5 Time to Market Apps:

App store is quite strict about the apps sold in their store and things get stricter for in app purchases. You have to submit an application and sometimes, wait for months to get their approval. The time to market for web apps is much smoother and simpler, whereas for the other two, you will have to wait.

5.4.6 User Experience and Interaction:

Native app provides much better accessibility features when in a native UI. You have absolute control here. Unfortunately, web apps are hindered by the capability of a web browser. For native apps, you can actually accelerate the UI performance when you enhance the capabilities of the device hardware.

5.4.7 Internationalization & Localization:

Every app development company dreams of surpassing geographical boundaries and creating software that people from anywhere can access. Internationalization & Localization for hybrid, native and web apps is excellent because the software can be designed in such a way that they can be adapted to any language without making engineering changes. These softwares can be localized by making it applicable for a particular region or language.[8]

6. Pros & Cons

6.1 Pros:

There are two big pros when it comes to progressive web apps. The offline support and the app install banners. The offline part comes from the caching that the web does with the service worker. The app install banner is a banner that chrome prompt when it sees a manifest on the site. This will allow the user add an icon on their home screen.[9]

6.2 Cons:

The first con is that is only supported by browsers that support the service worker. At the moment, that is only Chrome. Another big problem is the high barrier to entry. This high barrier is HTTPS. HTTPS gives the site better ranking in Google searches and more privacy. Another thing is the service worker. It's awesome, but it's another thing to learn, this means that developers will not pick it up so likely or poorly. A problem with that is that the service worker is really powerful but when not used properly it can give the developer a lot of headaches. [10]

7. Example

7.1 Flipkart Lite:

Flipkart Lite, a Progressive Web App that combines the best of the web and the best of the Flipkart native app. It leverages new, open web APIs to offer a mobile web experience that loads fast, uses less data than before, and re-engages users in multiple ways. Users visit via their browser and find a fast app-like user experience.[11]

7.1.1 A fast and streamlined site:

With 63% of Flipkart Lite users reaching the site via a 2G network, a fast user experience was essential. To decrease load times, Flipkart added service workers and streamlined the site to help consumers quickly reach the product they are looking for. Users can even continue to browse categories, review previous searches, and view product pages—all while offline.

7.1.2 Taking advantage of the web's low friction:

Reaching a broad set of users is important for Flipkart. With Flipkart Lite, users are one click away from accessing content and many new users are first-time internet users. In addition to easy access, Flipkart Lite requires less data. A key metric for Flipkart is tracking data usage to complete first transaction: when comparing Flipkart Lite to the native app, Flipkart Lite uses 3x less data. Having a strong and engaging mobile website means they're no longer turning

away potential shoppers who don't want to use data or space to download an app."

7.1.3 Bringing users back with home screen icon:

Flipkart wanted to be able to re-engage with mobile web users just as they would with mobile app users. The company implemented an "Add to Home Screen" prompt. Now, 60% of all visits to Flipkart Lite come from people launching the site from the homescreen icon. Add to Home Screen also delivers high-quality visits, with customers converting 70% more than average users. These two activities alone resulted in engagement numbers that were 40% higher than before.

7.1.4 Building for future success for the evolving online shopper:

Flipkart will continue using progressive web technology to reach their evolving online shoppers. Flipkart Lite has enabled us to find some of Flipkart's highest-value customers. Flipkart will continue to expand progressive web app technology across all of their platforms, investing significant resources to maximize the potential scale. They truly believe that this is a new way to experience mobile and they're just getting started."[12]

8. CONCLUSION

Progressive web apps have benefits for everyone involved. The user will be able to instantly install the "app" without a visit to the app store and a large download, which can be an unpleasant experience on a slow connection. Organizations can go back to developing web apps without requiring the requisite separate Android and iOS teams. They can update and "release" their app without going through the app store approval process. Releases and defect fixes can be deployed immediately. Web design elements are immediately picked up by the progressive web app.

A progressive web app is a website that combines the best experiences of the web and an app. They don't require any installation. The app loads quickly, even when the user is on bad networks. It can send relevant push notifications to the user and has an icon on the home screen and loads as top-level, full screen experience.

Application shell architectures comes with several benefits but only makes sense for some classes of applications. The model is still young and it will be worth evaluating the effort and overall performance benefits of this architecture.

Progressive web apps are an interesting forward look into the future of mobile apps. It will become a key factor in the world of apps.

REFERENCES

[1]http://www.ness-ses.com/progressive-web-apps-the-new-future-of-mobile-apps/
[2]http://blog.cloudfour.com/android-instant-apps-progressive-web-apps-and-the-future-of-the-web
[3]https://arc.applause.com/2015/11/30/application-shell-architecture
[4]http://digiday.com/platforms/wtf-progressive-web-apps/
[5]https://developers.google.com/web/updates/2015/11/app-shell?hl=en
[6]https://addyosmani.com/blog/getting-started-with-progressive-web-apps
[7]http://blog.ionic.io/what-is-a-progressive-web-app/
[8]http://www.cabotsolutions.com/native-vs-hybrid-vs-web-comparison-study/
[9]http://developer.telerik.com/featured/what-progressive-web-apps-mean-for-the-web/
[10]http://kasszz.github.io/minor-web-development/performanceMatters/exercise3/index.html
[11]https://mobiforge.com/news-comment/progressive-web-apps-are-future
[12]https://developers.google.com/web/showcase/2016/flipkart#tldr