Progressive Web Application

Migrating Web Application to a Progressive Web Application

Kristianto Karim
Computer Science/Informatics
School of Electrical Engineering and Informatics
Institut Teknologi Bandung, Jl. Ganesha No.10, Bandung, Indonesia
kristiantokarim@gmail.com

***Abstract*—Native application has good user experience but large file size and it needs installer. On the other hand, user experience of the web application is so unfriendly but it does not need installer and has a very small size. In this paper, progressive web application will be explained, including the way to implement it on web application. Even though it has not been supported by many browsers yet, I believe progressive web application can solve the problem stated above because of its fast, progressive, responsive, and native application-alike characteristic.**

Keywords—PWA; Progressive Web Apps; Responsive; Progressive; Migration; Web; User Experience; JavaScript; Manifest; Service Worker; Application Shell

I. INTRODUCTION

The increasing usage of mobile devices and internet has made these two products of nowadays' development a necessity for all of us. Through the mobile devices, we can download various application from the internet for our purposes. We can search for information, communicate through social media, and many other things anywhere, using the application we have downloaded on our mobile devices.

However, as time goes by, application size is getting larger and larger because of its increasing features. It results in more and more storage space is needed for the application. Indirectly, this may lead to increasing amount of electronic waste as the old storage space will be replaced by new larger storage space. Furthermore, the use of native application becomes not instant as it has to be downloaded and installed to our mobile device. Weaknesses in native application can be overcome by using web application which is more instant and takes less storage space because we just need to access it through URL. Still, web application also has big weakness, that is the lack of user experience when using it compared to native application.

The problem stated above can be solved using progressive web application (hereinafter referred to as PWA). PWA is a web-based application that utilizes the features from modern browser in order to act as if it were a native application.

II. THEORITICAL BASIS

A. *Progressive Web App*

Progressive Web Apps is a term for web-based application that uses the most up-to-date web technologies. PWA is actually a regular web-based application, but it makes great use of the modern browser features in order to appear as though it were a native application.

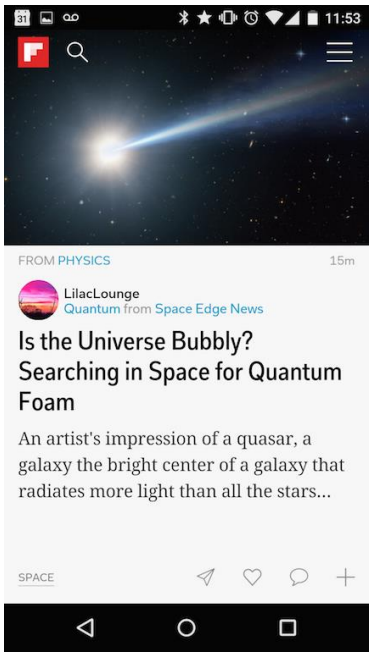

Image 1: Example of PWA View (Source: https://cdn-images-1.medium.com/max/800/0*sfgFRKgJTDNgIK_J.png)

Basically, PWA does not look different from the regular web application because it consists of HTML, CSS, JavaScript and accessed through internet browser. What distinguishes it is because PWA qualifies as follows:

1. **Responsive:** Can adjust to various screen sizes.

2. **Progressive:** Can be run by all users because the content is built progressively.
3. **App-like:** Gives user experience to the user as in native application through its interaction and navigation.
4. **Fresh:** Always up-to-date.
5. **Safe:** Secured with HTTPS.
6. **Discoverable:** Identified as an application (W3C *manifest*) and can be found by search engines.
7. **Re-engageable:** Has a feature that actively includes users such as push notification.
8. **Installable:** Users can save it as an application without having to install from AppStore.
9. **Linkable:** Easy to share through URL without having to go through a complicated installation.
10. **Connectivity-Independent:** Service Workers technology allows PWA to run without internet connection or through bad connection.

Note that PWA is different from the technologies such as Cordova, React Native, NativeScript and Electron. Those technologies create web-based application and wrap it into an APK or EXE that need to be installed while in PWA, it is not wrapped into an APK or EXE.

B. Progressive Web Apps Standard

A web-based application can be regarded as progressive web application if it meets the following requirements:

1. Site is served over HTTPS.
2. The page is responsive if it is opened on a mobile device.
3. Main page URL (*index*) can be accessed offline.
4. There is metadata to add to the Home Screen.
5. Can be accessed quickly for the first time despite using 3G.
6. Sites can be run on various browser (which supports JavaScript).
7. Page transition does not feel like blocked by the network.
8. Each page has a URL.

The requirements above are the basic requirements for a site to be categorized as progressive web application. We can verify the conditions above easily through the Lighthouse application created by Google.

C. Concepts and Technologies used by Progressive Web Application

PWA is built using the following concepts and technologies:

1. **Manifest:** A place to store metadata from web application from W3C.
2. **Service Worker:** Provides a layer between networks with the device that function to perform cache mechanisms for web applications as if they can be accessed offline.
3. **Application Shell:** Serves to provide a temporary container from the web application view. Progressively, the application will load its content to the container. The goal is that the users do not feel like loading an old web page so that the temporary content is displayed first.

D. Disadvantages of Progressive Web Application

Here are the disadvantages of PWA:

1. Not all browsers support it yet.
2. Limited to mobile devices with Android operating system.
3. The access to sensors and components of mobile devices (camera, GPS, and others) are not as free as in native application.
4. No central repository.

IV. MIGRATION PROCESS

In this paper, the migration process of a regular web application into PWA will be implemented.

E. Creating The Manifest File

Create a file named manifest.json in *root* directory containing JSON in the following format:

```
{
  "name": "Your apps name",
  "short_name": "Short apps name",
  "start_url": "/",
  "display": "standalone",
  "background_color": "#fff",
  "description": "Your site description",
  "icons": [{
    "src": "images/touch/homescreen48.png",
    "sizes": "48x48",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen72.png",
    "sizes": "72x72",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen96.png",
    "sizes": "96x96",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen144.png",
    "sizes": "144x144",
    "type": "image/png"
```

```
  }, {
    "src": "images/touch/homescreen168.png",
    "sizes": "168x168",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen192.png",
    "sizes": "192x192",
    "type": "image/png"
  }],
  "related_applications": [{
    "platform": "web"
  }, {
    "platform": "play",
    "url":
"https://play.google.com/store/apps/details?
id=cheeaun.hackerweb"
  }]
}
```

Then, add the following line to the *head* section of the entire page from web application that will be migrated.

```
<link rel="manifest" href="/manifest.json"
/>
```

This is the manifest.json file that will function to manage how web application appear as native application. Note that *"display" : "standalone"* property will change the web interface display so that its user experience feels like native.

F. Service Worker

Add an empty file named sw.js to the *root* directory. Then add the following code to register the Service Worker.

```
navigator.serviceWorker                    &&
navigator.serviceWorker.register('/sw.js').t
hen(function(registration) {

    console.log('Excellent, registered with
scope: ', registration.scope);

});
```

Next, to allow the web application to run offline, cache mechanism needs to be added on service worker. The cache mechanism can be implemented by adding the following code in the sw.js code file.

```
self.addEventListener('install', function(e)
{
  e.waitUntil(
    caches.open('the-magic-
cache').then(function(cache) {
      return cache.addAll([
        '/',
        '/put.js',
        '/all.css',
        '/your page and.html',
        '/your assets.jpeg',
        '/here.json'
      ]);
    })
  );
});
```

The above code only stores the cache storage. Next we should respond to users who are doing HTTP requests with the cache we have stored. It can be done by adding the following code to sw.js.

```
self.addEventListener('fetch',
function(event) {
  event.respondWith(

caches.match(event.request).then(function(re
sponse) {
      return           response           ||
fetch(event.request);
    })
  );
});
```

G. Application Shell

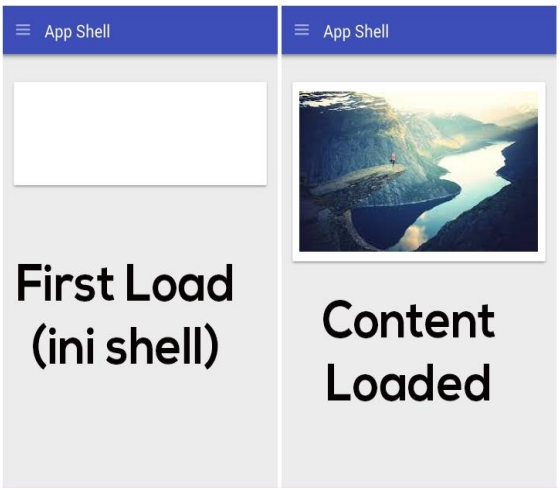

Image 2: *Application Shell* (Source: https://developers.google.com/web/updates/images/2015/11/appshell/appshell-1.jpg)

Application shell is the minimal HTML, CSS and JavaScript required so that PWA can be loaded directly and quickly. Application Shell acts as shell or temporary container that will be replaced with the actual content.

To design the Application shell, there are several things to be considered such as:

1. What should be displayed directly on the screen?
2. What are the important components to our application?
3. What are the resources needed by application shell?

After the first three steps implemented, we also need to do some modifications to our web apps to satisfy all the basic requirements of PWA. The usual modifications are switching to HTTPS supported web server, lightweighting the initial load so the web apps can be accessed in poor connection (application shells should be well designed because application shells is loaded first), etc.

V. CONCLUSION

Although PWA still has weaknesses, PWA can be used to solve the storage space and installation time problem in native application as well as resolve the issues of lack of interaction and user experience from web application. It is shown by the instant nature of PWA like a web application, responsive, and interactive as in native application.

REFERENCES

[1] Fern, D. (2016, October 13). *Progressive Web Applications, Part 2: Pros, Cons, and Looking Ahead*. Retrieve from https://www.digitalgov.gov

[2] Queppelin.(2016, November 22). *Understanding Progressive Web Applications (PWA)*. Retrieve from http://www.queppelin.com

[3] LePage P. (2017, April 10). *Your First Progressive Web App*. Retrieve from https://developers.google.com

[4] Farrugia, K. (2016, August 11). *A Beginner's Guide To Progressive Web Apps.* Retrieve from https://www.smashingmagazine.com

[5] *Migrate your site to a Progressive Web App*. (n.d.). Retrieve from https://codelabs.developers.google.com/

[6] *PWA Tutorial.*(n.d.). Retrieve from https://github.com/IncredibleWeb/pwa-tutorial

[7] Taradaev, S. (2016, December 16). *BUILDING PROGRESSIVE WEB APPS IN 5 SIMPLE STEPS.* Retrieve from https://waverleysoftware.com

[8] Hilmarsson, H. (2017, January 19). *We built a PWA from scratch – This is what we learned*. Retrieved from https://14islands.com

[9] *Buidling a Progressive Web App in Polymer from Scratch.*(n.d). Retrieve from https://codelabs.developers.google.com/

[10] Russel, A. (2015, June 15). *Progressive Web Apps: Escaping Tabs Without Losing Our Soul*. Retrieve from https://infrequently.org

STATEMENT

I hereby declare that the paper I am writing is my own work, not an adaption, or a translation of someone else's paper, and not plagiarism.

Bandung, 5 May 2017

Kristianto Karim - 13514075