**Topic:PWA**

**-Steps to make pwa :**

1) add pwa using (ng add @angular/pwa --project projectName)

2) npm install -g http-server-spa

3) ng build --prod --source-map

4) http-server-spa dist/rojectName / index.html 8080

**1) What is PWA?**

-A Progressive Web App (PWA) is an app that uses modern Web capabilities to offer users a very similar (if not better) experience to a native app.

- progressive web apps are a hybrid between regular web pages and mobile applications

**2) How does PWA work?**

-A PWA is a web application that can be “installed” on your system. It works offline when you don't have an internet connection, leveraging data cached during your last interactions with the app.If you launch the apps, they will run in Chrome rather than having their own icon.

-Progressive Web Apps, aka PWAs, are the best way for developers to make their webapps load faster and more performant.

**3)Features of PWA:**

- 1)[Secure contexts](https://developer.mozilla.org/en-US/docs/Web/Security/Secure_Contexts) ([HTTPS](https://developer.mozilla.org/en-US/docs/Glossary/https)), one or more (2) [Service Workers](https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API), and a (3)[manifest file](https://developer.mozilla.org/en-US/docs/Web/Manifest).

[Secure contexts](https://developer.mozilla.org/en-US/docs/Web/Security/Secure_Contexts) ([HTTPS](https://developer.mozilla.org/en-US/docs/Glossary/https))

The web application must be served over a secure network. Being a secure site is not only a best practice, but it also establishes your web application as a trusted site especially if users need to make secure transactions. Most of the features related to a PWA such as geolocation and even service workers are available only once the app has been loaded using HTTPS.

[Service workers](https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API)

A service worker is a script that allows intercepting and control of how a web browser handles its network requests and asset caching. With service workers, web developers can create reliably fast web pages and offline experiences.

[Manifest file](https://developer.mozilla.org/en-US/docs/Web/Manifest)

A [JSON](https://developer.mozilla.org/en-US/docs/Glossary/JSON) file that controls how your app appears to the user and ensures that progressive web apps are discoverable. It describes the name of the app, the start URL, icons, and all of the other details necessary to transform the website into an app-like format.

**4) What makes a good Progressive Web App?**

### -Starts fast, stays fast

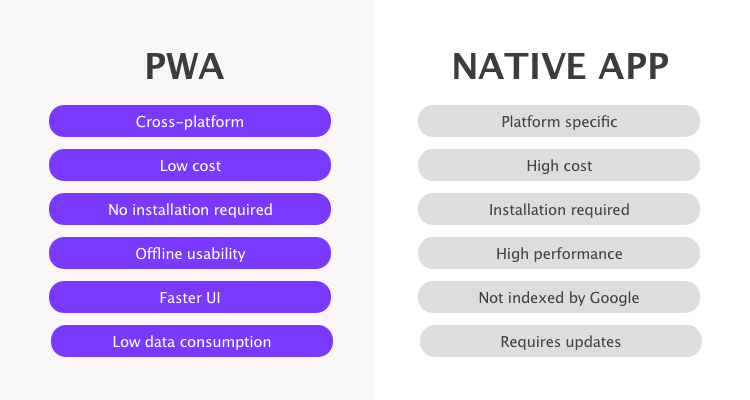
### -Works in any browser

### -Responsive to any screen size

### -Provides a custom offline page

### -Is installable

**5)PWA VS Native App:**



**6) Progressive Web Apps Examples**

Starbucks.

Housing.com.

Digikala.

Flipboard.

Soundslice.

## **7) Service worker lifecycle:**

A service worker goes through three steps in its lifecycle:

* Registration
* Installation
* Activation

## **8)** [**Why Should You Build a PWA?**](https://scotch.io/tutorials/the-ultimate-guide-to-progressive-web-applications#toc-why-should-you-build-a-pwa-)

You should make your webapp into a PWA because it’ll reduce the time it takes for your app to load and it’ll give your users a better experience. Having it load over HTTPS is a good security practice and adding icons (using a web app manifest) is something you’d do anyway. Having a cache-first service worker strategy will allow your app to work offline (if the user has already loaded data), alleviating one of the biggest issues with webapps.

There are a number of other performance recommendations you can implement in your webapp. While the following list is not required for PWAs, many PWAs employ these elements:

* Implement the [PRPL pattern](https://synd.co/2sTNeWw):
  + **Push** critical resources for the initial URL route.
  + **Render** initial route.
  + **Pre-cache** remaining routes.
  + **Lazy-load** and create remaining routes on demand.
* Use <link rel=”preload”> to tell your browser to load a resource you know you’ll eventually need. This is a [W3C Standard specification](https://synd.co/2r2tweS).
* Use HTTP/2 and server push to “push” assets to the browser without the user having to ask for them.
* Use code-splitting and lazy-loading for granular loading of application pages/features.