Service Worker

What is Service Worker?

A service worker is 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. Today, they already include features like push notifications and background sync.

With the **Angular Service Worker** and the Angular CLI built-in PWA support, it's now simpler than ever to make our web application downloadable and installable, *just like a native mobile application*

Here we will discuss topics:

- Step 1 - Scaffolding an Angular PWA Application using the Angular CLI
- Step 2 - Understanding How To Add Angular PWA Support Manually
- Step 3 - Understanding the Angular Service Worker runtime caching mechanism
- Step 4 - Running and Understanding the PWA Production Build
- Step 5 - One-Click Install with the App Manifest
- Sample images

Step 1:- Create Angular PWA Application Using Angular

For this first We need to install latest version of Angular Cli

Command :- `npm install -g @angular/cli@latest`

Command :- `npm install -g @angular/cli@next`

Command to Create Pwa Project:- `ng new angular-pwa-app --service-worker`

Step 2 :- Adding Angular Service Worker support manually

```
ng add @angular/pwa --project <name of project as in angular.json>
```

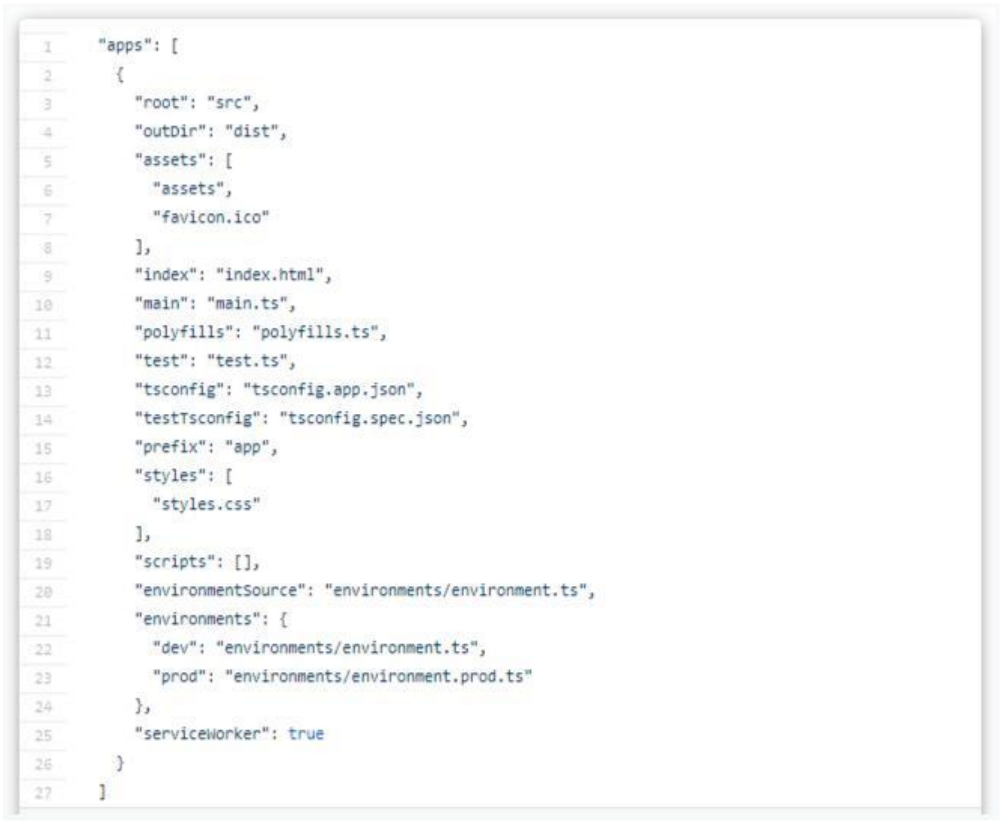
In Our Project we have a angular.json

It contain a flag serviceworker this flag will cause the production build to include a couple of extra files in the output dist folder:

- The Angular Service Worker file ngsw-worker.js
- The runtime configuration of the Angular Service Worker ngsw.json

`Step 3:-` Understanding the Angular Service Worker runtime caching mechanism

The Angular Service Worker can cache all sorts of content in the browser Cache Storage.
This is a Javascript-based key/value caching mechanism that is not related to the standard browser Cache-Control mechanism, and the two mechanisms can be used separately.



```
"apps": [
  {
    "root": "src",
    "outDir": "dist",
    "assets": [
      "assets",
      "favicon.ico"
    ],
    "index": "index.html",
    "main": "main.ts",
    "polyfills": "polyfills.ts",
    "test": "test.ts",
    "tsconfig": "tsconfig.app.json",
    "testTsconfig": "tsconfig.spec.json",
    "prefix": "app",
    "styles": [
      "styles.css"
    ],
    "scripts": [],
    "environmentSource": "environments/environment.ts",
    "environments": {
      "dev": "environments/environment.ts",
      "prod": "environments/environment.prod.ts"
    },
    "serviceWorker": true
  }
]
```

All the configuration related to service worker is available in **`ngsw-config.json`**

.

There are two cache configuration entries

1. App (For single page app it cache index.html,css,js)
2. Assets (other required assets like Images)

Step 4 :- Running and Understanding the PWA Production Build

Angular Service Worker will only be available in production mode, so let's first do a production build of our application

```
ng build --prod
```

It will create our production build into the dist folder this folder will contain two files related to Service worker

1) Ngsw.json (Runtime configuration file)
2) Ngsw-worker.js (Service Worker file)

What is ngsw.json file

This is runtime configuration file, That the angular service worker will use. This file contains the information that what needs to be chached.

Sample of ngsw.json file

```
          {
  "configVersion": 1,
  "timestamp": 1557252409455,
  "index": "/index.html",
  "assetGroups": [
    {
      "name": "app",
      "installMode": "prefetch",
      "updateMode": "prefetch",
      "urls": [
        "/es2015-polyfills.c5dd28b362270c767b34.js",
        "/favicon.ico",
        "/index.html",
        "/main.8e7c35a51d8df1f29679.js",
        "/polyfills.8bbb231b43165d65d357.js",
        "/runtime.26209474bfa8dc87a77c.js",
        "/styles.ae876c74e5091f736943.css"
      ],
      "patterns": []
    },
    {
      "name": "assets",
      "installMode": "lazy",
      "updateMode": "prefetch",
      "urls": [
        "/assets/icons/icon-128x128.png",
        "/assets/icons/icon-144x144.png",
        "/assets/icons/icon-152x152.png",
        "/assets/icons/icon-192x192.png",
        "/assets/icons/icon-384x384.png",
        "/assets/icons/icon-512x512.png",
        "/assets/icons/icon-72x72.png",
        "/assets/icons/icon-96x96.png"
      ],
      "patterns": []
    }
  ],
  "dataGroups": [],
  "hashTable": {
    "/assets/icons/icon-128x128.png": "dae3b6ed49bdaf4327b92531d4b5b4a5d30c7532",
    "/assets/icons/icon-144x144.png": "b0bd89982e08f9bd2b642928f5391915b74799a7",
    "/assets/icons/icon-152x152.png": "7479a9477815dfd9668d60f8b3b2fba709b91310",
    "/assets/icons/icon-192x192.png": "1abd80d431a237a853ce38147d8c63752f10933b",
    "/assets/icons/icon-384x384.png": "329749cd6393768d3131ed6304c136b1ca05f2fd",
    "/assets/icons/icon-512x512.png": "559d9c4318b45a1f2b10596bbb4c960fe521dbcc",
    "/assets/icons/icon-72x72.png": "c457e56089a36952cd67156f9996bc4ce54a5ed9",
    "/assets/icons/icon-96x96.png": "3914125a4b445bf111c5627875fc190f560daa41",
    "/es2015-polyfills.c5dd28b362270c767b34.js": "b7a81c61d32b0b12737f0c13f45973ba391cd7f9",
    "/favicon.ico": "84161b857f5c547e3699ddfbffc6d8d737542e01",
    "/index.html": "a035c43eee902064648aa2905497e82636d53368",
    "/main.8e7c35a51d8df1f29679.js": "64e652749ce3a9cfb0c70b2e094223569115a6fc",
    "/polyfills.8bbb231b43165d65d357.js": "ffa449995bf2eba2ccd0a002417214f7eb75e7b5",
    "/runtime.26209474bfa8dc87a77c.js": "b62956c2192bfe5516d6374e753773901ed50ec5",
    "/styles.ae876c74e5091f736943.css": "fd1162acb81455cfd6308dfcd433695dddf07c6b"
  },
  "navigationUrls": [
    {
      "positive": true,
      "regex": "^\\/.*$"
    },
    {
      "positive": false,
      "regex": "^\\/(?:.+\\/)?[^/]*\\.[^/]*$"
    },
    {
      "positive": false,
      "regex": "^\\/(?:.+\\/)?[^/]*__[^/]*$"
    },
    {
      "positive": false,
      "regex": "^\\/(?:.+\\/)?[^/]*__[^/]*\\/.*$"
    }
  ]
}
```

Through this **hashTable** entries service worker will know which files needs to be cached. So it download those files and in background.

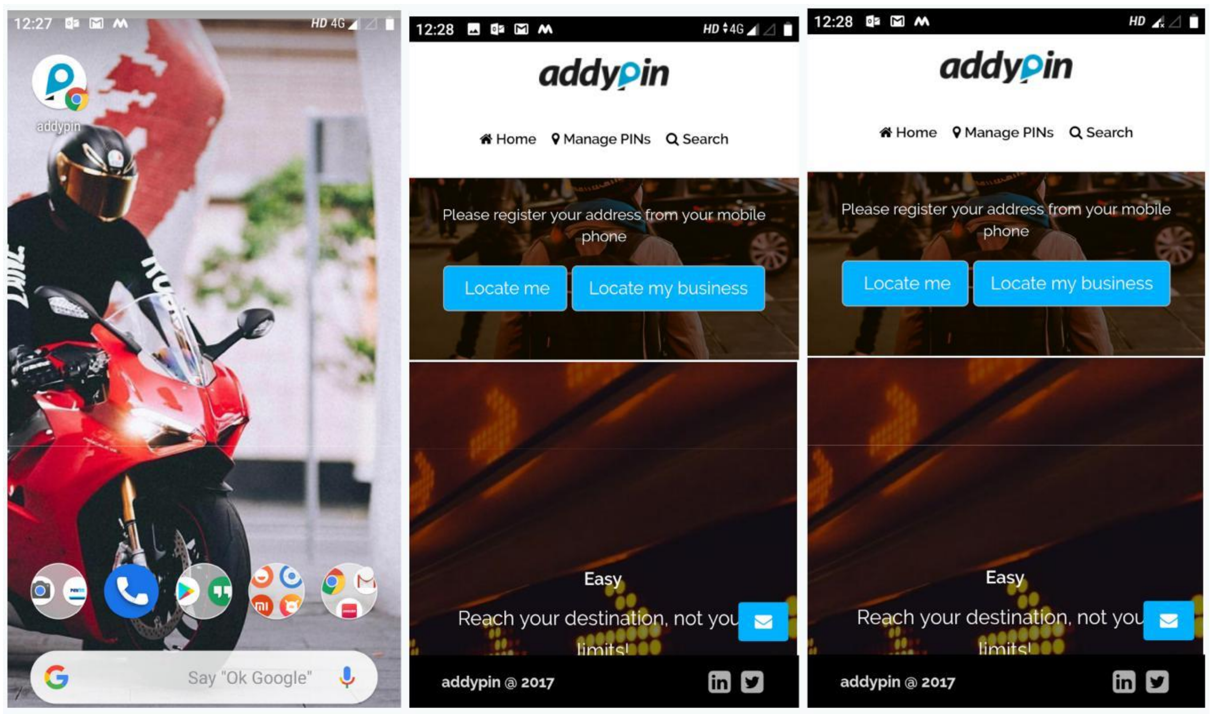
Step 5 - One-Click Install with the App Manifest.

We can make it one click installable through manifest.json file after adding this manifest file to our website we can add our website to home screen and then we can use website as an application.

```json
{
  "name": "angular-pwa-app",
  "short_name": "angular-pwa-app",
  "theme_color": "#1976d2",
  "background_color": "#fafafa",
  "display": "standalone",
  "scope": "/",
  "start_url": "/",
  "icons": [
    {
      "src": "assets/icons/icon-72x72.png",
      "sizes": "72x72",
      "type": "image/png"
    },
    {
      "src": "assets/icons/icon-96x96.png",
      "sizes": "96x96",
      "type": "image/png"
    },
    {
      "src": "assets/icons/icon-128x128.png",
      "sizes": "128x128",
      "type": "image/png"
    },
    {
      "src": "assets/icons/icon-144x144.png",
      "sizes": "144x144",
      "type": "image/png"
    },
    {
      "src": "assets/icons/icon-152x152.png",
      "sizes": "152x152",
      "type": "image/png"
    },
    {
      "src": "assets/icons/icon-192x192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "assets/icons/icon-384x384.png",
      "sizes": "384x384",
      "type": "image/png"
    },
    {
      "src": "assets/icons/icon-512x512.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ]
}
```

Once all this complete we have to create production build through angular cli and deploy build on server then it will work as PWA

Application home screen sample:- Image



I have added sample image how it works on prod env.

1. In first image we can see a clickable icon of PWA app.
2. In second image we can see how website work on mobiles as an Application its not running in browser.
3. In third image we can see internet connection is down but still it show web page through Cache.