Services(DI) and Workers(Service and web)

Ng new serviceworker –service-worker

Enables service worker to true

Service worker becomes available when we build in –prod

Ng build –prod

Check in browser > Application

Service worker will be seen

Dist folder can be seen

Npm install Http-server –g

Cd dist

http-server

Chekin browser

U can see in service broker

U will see the script that is activated and is running

<https://medium.com/codingthesmartway-com-blog/angular-5-service-worker-b722e571e306>

Adding a **service worker** to an **Angular** application is one of the steps for turning an application into a Progressive Web App (also known as a PWA). At its simplest, a **service worker** is a script that runs in the web browser and manages caching for an application. **Service workers** function as a network proxy.

 They intercept all outgoing HTTP requests made by the application and can choose how to respond to them. For example, they can query a local cache and deliver a cached response if one is available.

Unlike the other scripts that make up an application, such as the Angular app bundle, the service worker is preserved after the user closes the tab. The next time that browser loads the application, the service worker loads first, and can intercept every request for resources to load the application. If the service worker is designed to do so, it can completely satisfy the loading of the application, without the need for the network.

Angular's service worker is designed to optimize the end user experience of using an application over a slow or unreliable network connection, while also minimizing the risks of serving outdated content.

The Angular service worker's behavior follows that design goal:

* Caching an application is like installing a native application. The application is cached as one unit, and all files update together.
* A running application continues to run with the same version of all files. It does not suddenly start receiving cached files from a newer version, which are likely incompatible.
* When users refresh the application, they see the latest fully cached version. New tabs load the latest cached code.
* Updates happen in the background, relatively quickly after changes are published. The previous version of the application is served until an update is installed and ready.
* The service worker conserves bandwidth when possible. Resources are only downloaded if they've changed.

To support these behaviors, the Angular service worker loads a *manifest* file from the server. The manifest describes the resources to cache and includes hashes of every file's contents. When an update to the application is deployed, the contents of the manifest change, informing the service worker that a new version of the application should be downloaded and cached. This manifest is generated from a CLI-generated configuration file called ngsw-config.json.

Installing the Angular service worker is as simple as including an [NgModule](https://angular.io/api/core/NgModule). In addition to registering the Angular service worker with the browser, this also makes a few services available for injection which interact with the service worker and can be used to control it. For example, an application can ask to be notified when a new update becomes available, or an application can ask the service worker to check the server for available updates.

What's being cached?

Notice that all of the files the browser needs to render this application are cached. The ngsw-config.json boilerplate configuration is set up to cache the specific resources used by the CLI:

* index.html.
* favicon.ico.
* Build artifacts (JS and CSS bundles).
* Anything under assets.
* Images and fonts directly under the configured outputPath (by default ./dist/<project-name>/) or resourcesOutputPath. See [ng build](https://angular.io/cli/build) for more information about these options.