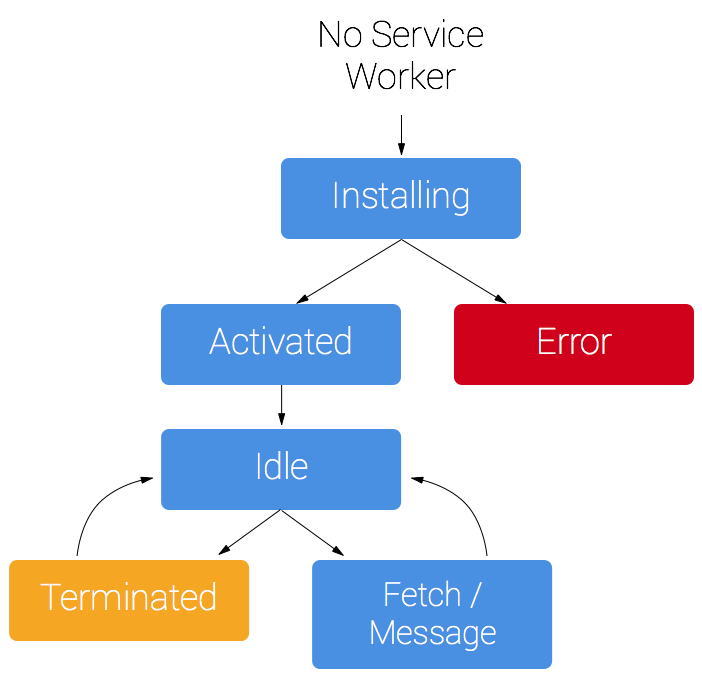
Service workers are JavaScript web workers that runs on browser in background independent of browser’s main thread. Service workers doesn’t need browser to be active. They include some features like **push notification**, **background sync**, **offline caching** etc.

Service workers can’t access DOM directly as they run independently of web page. However, they can post message to a web page using **PostMessage** Interface.

**Service Worker Life Cycle**



Life cycle of service worker is independent of a web page

1. To install a service worker, it needs to be registered first which is done mostly in web page. Once it gets registered, it starts running in background.
2. Most of the static assets are cached during installation process. Once the files are cached successfully only then the service worker activates successfully. if it fails then service worker will not be installed and activated. But it will continue to try installation process until it is successful.
3. Once it is installed successfully, all the static files that are cached during the install process and saved in browser cache can be fetched using fetch event from the web request.
4. When the worker is up and running, it can carry out all the supported tasks like handling push notifications, background sync etc without the need of web browser’s active state.
5. When the service worker will go to idle state, it terminates itself to save memory, which can be reinstalled again by registering it in a web page. They can also be terminated by user manually.

**Prerequisites**

* **Browser Support**

Service workers are not supported by all browsers. Only browsers like chrome, Firefox etc will have built-in service worker support.

* **Need of HTTPS**

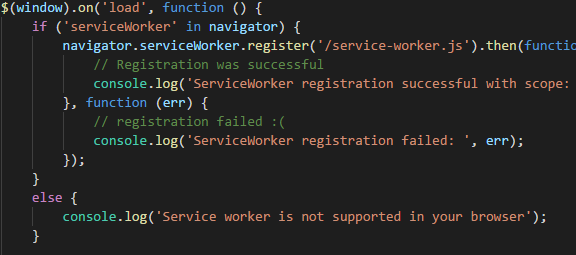
Service workers can only be registered over https protocol for security reasons however we can run them on localhost during development.

**Sample Demo**

Here in this demo, we create a sample web application which fetches all the static resources from cache when the user is online.

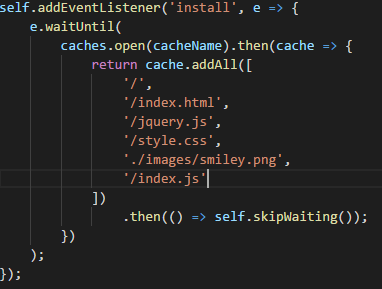
1. **Registering the service worker**

We created a js file called as index.js and we **register** the service worker on window load event as follows,

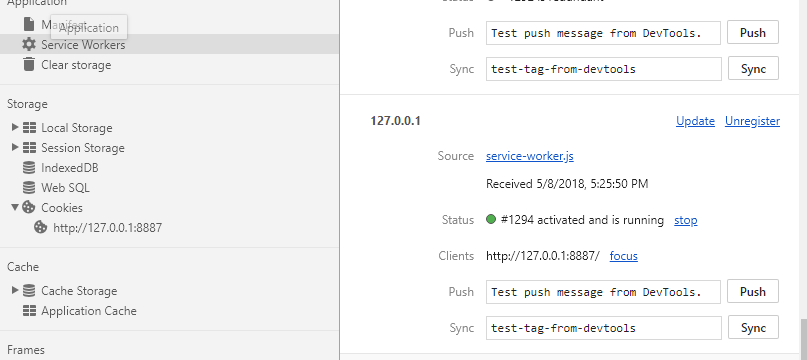


1. **Caching the resources in install event of service worker**

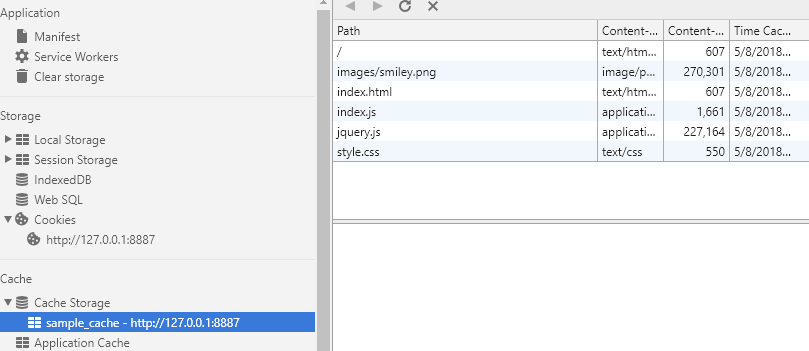
Once the service worker is registered, it caches all the mentioned files in **install event** as follows,



Once the service worker is successfully installed, we can see the service worker activated and running in service worker section of chrome browser as follows,



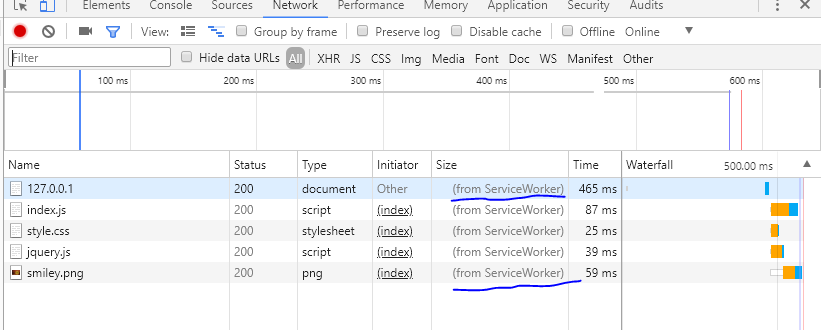
Also, we can notice that all the necessary files will be cached under cache storage as follows if everything goes right,



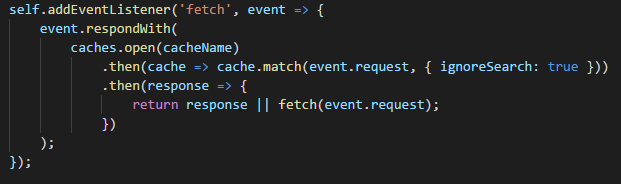
1. **Fetching the resources from cache instead of server**

**When Online**

Once the SW is successfully installed and cached all the resources, we can see in network tab showing that all the resources are loaded from service worker cache instead of server even when the user is online.

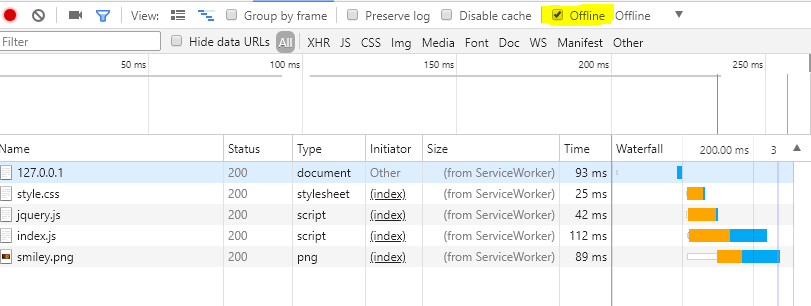


This is to ensure better performance for the web page. So as per our observations we can clearly say that all the **resources will be loaded from server only for the first time**. After they are cached successfully by service worker, they’ll be loaded from cache from next requests using service worker’s fetch event as follows.



**When Offline**

When the user is offline, all the resources are fetched from cache as per below image,



We can clearly see that I’ve turned on offline mode and all the resources are fetched from service worker.

**Service workers support for Ionic Cordova Android Application**

As per [this](https://stackoverflow.com/questions/44877656/how-to-use-service-workers-in-cordova-android-app) answer on stackoverflow, Cordova android app doesn’t support service workers (**Need to do more R&D on this**) as they block service workers from HTTP protocol and also ionic forum doesn’t mention on their blog post about service workers. Ionic will support service workers only when they are published as PWA (progressive web app) not as native Cordova android apps as they serve resources via file protocol not over https

**References:**

* [**https://ionicframework.com/docs/developer-resources/service-worker/**](https://ionicframework.com/docs/developer-resources/service-worker/)
* [**https://developers.google.com/web/fundamentals/primers/service-workers/**](https://developers.google.com/web/fundamentals/primers/service-workers/)
* [**https://stackoverflow.com/questions/44877656/how-to-use-service-workers-in-cordova-android-app**](https://stackoverflow.com/questions/44877656/how-to-use-service-workers-in-cordova-android-app)
* [**https://caniuse.com/#search=service%20workers**](https://caniuse.com/%23search=service%20workers)