WebWorkers:

Async, Runs in new thread

Use- complex activity (Calculation, ….) that can block the main thread is made to run in separate thread using web workers.

It cannot access DOM directly.

Service Workers, Dedicated Workers, Shared Workers

Service Workers – Proxy between server and client

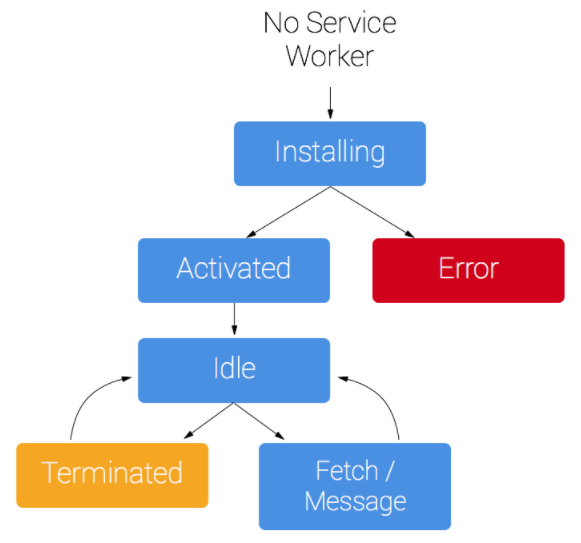
To make api calls and store the data in cache that can be used offline when real api is not available.

It has access to indexedDB API.

Use- To develop PWA (Progressive Web Application), Push Notification, Sync

Lifecycle is below:

1. Registration
2. Installation
3. Activation



**You need HTTPS;**

During development you'll be able to use service worker through localhost, but to deploy it on a site you'll need to have HTTPS setup on your server. (i.e. ssl certified website)

Register:

if ('serviceWorker' in navigator) {  
  window.addEventListener('load', function() {  
    navigator.serviceWorker.register('/sw.js').then(function(registration) {  
      // Registration was successful  
      console.log('ServiceWorker registration successful with scope: ', registration.scope);  
    }, function(err) {  
      // registration failed :(  
      console.log('ServiceWorker registration failed: ', err);  
    });  
  });  
}

## Install a service worker:

1. Open a cache.
2. Cache our files.
3. Confirm whether all the required assets are cached or not.

var CACHE\_NAME = 'my-site-cache-v1';  
var urlsToCache = [  
  '/',  
  '/styles/main.css',  
  '/script/main.js'  
];  
  
self.addEventListener('install', function(event) {  
  // Perform install steps  
  event.waitUntil(  
    caches.open(CACHE\_NAME)  
      .then(function(cache) {  
        console.log('Opened cache');  
        return cache.addAll(urlsToCache);  
      })  
  );  
});

## Cache and return requests:

self.addEventListener('fetch', function(event) {  
  event.respondWith(  
    caches.match(event.request)  
      .then(function(response) {  
        // Cache hit - return response  
        if (response) {  
          return response;  
        }  
  
        return fetch(event.request).then(  
          function(response) {  
            // Check if we received a valid response  
            if(!response || response.status !== 200 || response.type !== 'basic') {  
              return response;  
            }  
  
            // IMPORTANT: Clone the response. A response is a stream  
            // and because we want the browser to consume the response  
            // as well as the cache consuming the response, we need  
            // to clone it so we have two streams.  
            var responseToCache = response.clone();  
  
            caches.open(CACHE\_NAME)  
              .then(function(cache) {  
                cache.put(event.request, responseToCache);  
              });  
  
            return response;  
          }  
        );  
      })  
    );  
});

## Update a service worker:

self.addEventListener('activate', function(event) {  
  
  var cacheAllowlist = ['pages-cache-v1', 'blog-posts-cache-v1'];  
  
  event.waitUntil(  
    caches.keys().then(function(cacheNames) {  
      return Promise.all(  
        cacheNames.map(function(cacheName) {  
          if (cacheAllowlist.indexOf(cacheName) === -1) {  
            return caches.delete(cacheName);  
          }  
        })  
      );  
    })  
  );  
});