3.5 Working with Offline Page



This section will guide you to:

* Enable offline feature in your app using service worker

This guide has mainly five subsections, namely:

3.5.1 Creating an Angular application

3.5.2 Installing PWA and **http-server**

3.5.3 Building and run the application

3.5.4 Adding service worker for offline access

3.5.5 Pushing the files to your GitHub repositories

**Step 3.5.1:** Creating an Angular application

* Run the below command to create an Angular application:

**ng new angpwa**

**Step 3.5.2:** Installing PWA and **http-server**

* Navigate to the application folder and run the below command to install PWA and **http-server**:

**ng add @angular/pwa**

**npm install -g http-server@0.9.0**

**Step 3.5.3:** Building and running the application

* Run the below command to build your application to the production mode:

**ng build --prod**

* Navigate to the application folder under **dist** folder and run the below command to serve the application on **http-server:**

**cd dist/angpwa**

**http-server -o**

* Open **localhost:8080** in your browser.
* To install or download the application to your local system:
* **For Chrome:** Click on the **+** button on the right corner of the address bar.

**Step 3.5.4:** Add service worker for offline access

* Add the code given below in your home page:

**if** ('serviceWorker' **in** navigator) {

navigator.serviceWorker.register('service-worker.js');

}

* Create **service-worker.js** and add the code given below:

**const** OFFLINE\_VERSION **=** 1;

**const** CACHE\_NAME **=** 'offline';

*// Customize this with a different URL if needed.*

**const** OFFLINE\_URL **=** 'offline.html';

self.addEventListener('install', (event) **=>** {

event.waitUntil((**async** () **=>** {

**const** cache **=** **await** caches.open(CACHE\_NAME);

*// Setting {cache: 'reload'} in the new request will ensure that the response*

*// isn't fulfilled from the HTTP cache; i.e., it will be from the network.*

**await** cache.add(**new** Request(OFFLINE\_URL, {cache: 'reload'}));

})());

});

self.addEventListener('activate', (event) **=>** {

event.waitUntil((**async** () **=>** {

*// Enable navigation preload if it's supported.*

*// See https://developers.google.com/web/updates/2017/02/navigation-preload*

**if** ('navigationPreload' **in** self.registration) {

**await** self.registration.navigationPreload.enable();

}

})());

*// Tell the active service worker to take control of the page immediately.*

self.clients.claim();

});

self.addEventListener('fetch', (event) **=>** {

*// We only want to call event.respondWith() if this is a navigation request*

*// for an HTML page.*

**if** (event.request.mode **===** 'navigate') {

event.respondWith((**async** () **=>** {

**try** {

*// First, try to use the navigation preload response if it's supported.*

**const** preloadResponse **=** **await** event.preloadResponse;

**if** (preloadResponse) {

**return** preloadResponse;

}

**const** networkResponse **=** **await** fetch(event.request);

**return** networkResponse;

} **catch** (error) {

*// catch is only triggered if an exception is thrown, which is likely*

*// due to a network error.*

*// If fetch() returns a valid HTTP response with a response code in*

*// the 4xx or 5xx range, the catch() will NOT be called.*

console.log('Fetch failed; returning offline page instead.', error);

**const** cache **=** **await** caches.open(CACHE\_NAME);

**const** cachedResponse **=** **await** cache.match(OFFLINE\_URL);

**return** cachedResponse;

}

})());

}

*// If our if() condition is false, then this fetch handler won't intercept the*

*// request. If there are any other fetch handlers registered, they will get a*

*// chance to call event.respondWith(). If no fetch handlers call*

*// event.respondWith(), the request will be handled by the browser as if there*

*// were no service worker involvement.*

});

**Step 3.5.5:** Pushing the files to your GitHub repositories

* Open your command prompt and navigate to the folder where you have created your files.

cd <folder path>

* Initialize your repository using the following command:

git init

* Add all the files to your Git repository using the following command:

git add .

* Commit the changes using the following command:

git commit . -m “Changes have been committed.”

* Push the files to the folder you initially created using the following command:

git push -u origin master