St. Francis Institute of Technology, Mumbai-400 103
Department of Information Technology

A.Y. 2024-2025
Class: TE-ITA/B, Semester: VI

Subject: **MAD & PWA LAB**

Experiment – 9: Registration, Installation and Activation of Service worker for Ecommerce PWA

1. **Aim:** To code and register a service worker and complete the install and activation process for a new service worker for the E-commerce PWA.

2. **Objectives:** After study of this experiment, the student will be able to
   - Develop responsive web applications by combining AJAX development techniques with the jQuery JavaScript library.
   - Understand how service workers operate.

3. **Outcomes:** After study of this experiment, the student will be able to
   - Understand various PWA frameworks and their requirements.
   - Design and Develop a responsive User Interface by applying PWA Design techniques.

4. **Prerequisite:** HTML/ CSS/ JavaScript.

5. **Requirements:** Visual Studio Code, Bootstrap framework, Internet Connection.

6. **Pre-Experiment Exercise:**
   Brief Theory:

   Service Worker

   A service worker is a script that runs in the background of your web application. It doesn't need the DOM and in fact doesn't even have access to the DOM. Service workers run in a separate thread from the UI, so they don't block or freeze the UI while they process. They only work on HTTPS. The service worker acts as an intermediary between your app and the Internet. It then performs whatever function you've set it up to perform, and finally communicates some result back to your app by passing messages. Figure shows architecture of service worker.

   A service worker accomplishes:
   1. Caching assets like images, scripts, or styles
   2. Caching entire pages
   3. Syncing an app that was offline once its Internet connection comes back to life
   4. Push notifications

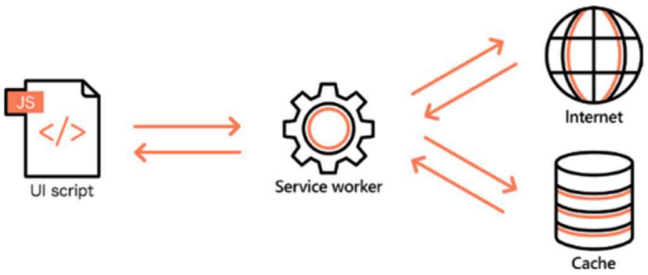


Figure: Service worker architecture.

Service Worker Lifecycle

The main parts of the service worker lifecycle.

1. Registration and downloading
2. Installation
3. Waiting (sometimes)
4. Activation
5. Updating

The first thing you do to create your first service worker is register it. This will download your service worker script. Then in a script called service-worker.js, you listen for the install and activate events, the other two parts of the life cycle. Finally, you just need to reference your script in an index.html file.

There are cases where this path takes a few detours when you update the service worker.

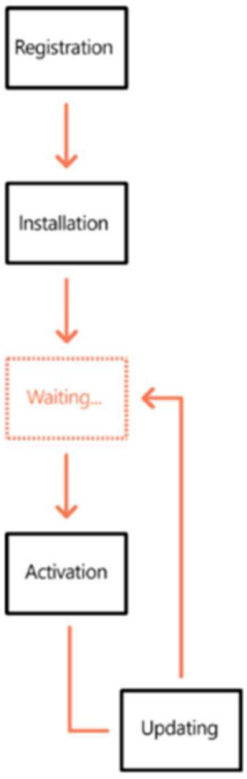


Figure: Service worker architecture.

7. Laboratory Exercise

A. Program

1. Register a service worker and complete the install and activation process for a new service worker for the E-commerce PWA.

B. Result/Observation

1. Print out of program code and output.

8. Post-Experimental Exercise

A. Questions:

1. Explain how the caching mechanism gets implemented with service workers.

B. Conclusion:

1. Write what you have learnt in the experiment.

C. References:

1. https://flaviocopes.com/service-workers/
2. https://developers.google.com/web/ilt/pwa/e-commerce-lab-1-create-a-service-worker
3. Building Progressive Web Apps, O'Reilly 2017.

Name: Vishal Rajesh Mahajan EXP 09

Class: TE IT A Roll No: 56

public/sw.js

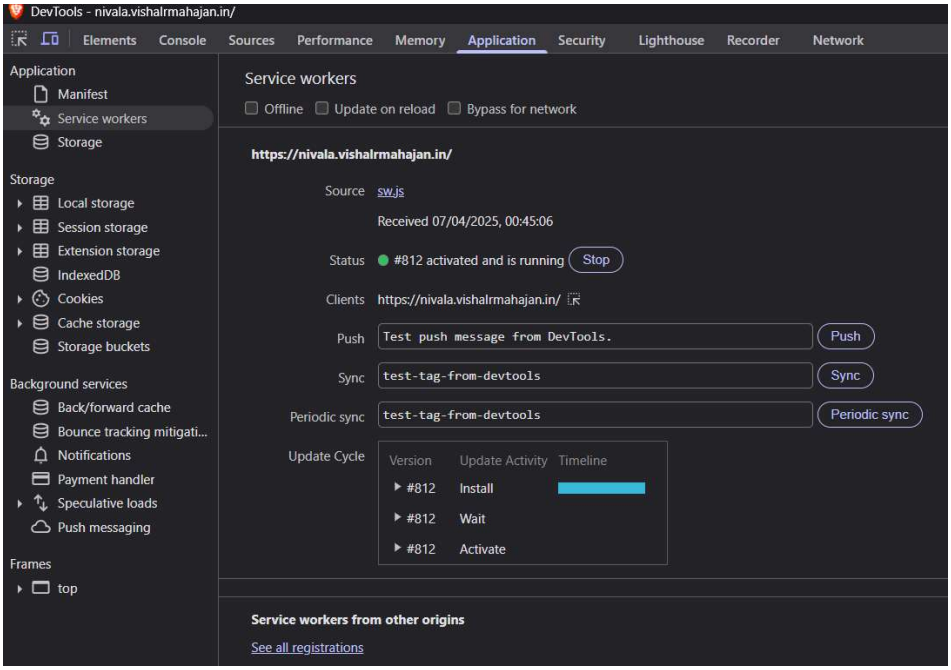
```
const CACHE_NAME = "nivala-cache-v1";
const urlsToCache = [
  "/",
  "../index.html",
  "../src/main.jsx",
  "../src/index.css",
];

self.addEventListener("install", (event) => {
  event.waitUntil(
    caches.open(CACHE_NAME).then((cache) => cache.addAll(urlsToCache))
  );
  self.skipWaiting();
});

self.addEventListener("activate", (event) => {
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames.map((cache) => {
          if (cache !== CACHE_NAME) return caches.delete(cache);
        })
      );
    })
  );
  self.clients.claim();
});

self.addEventListener("fetch", (event) => {
  event.respondWith(
    caches
      .match(event.request)
      .then((response) => {
        return response || fetch(event.request);
      })
      .catch(() => {
        return caches.match("/"); // fallback to home
      })
  );
})
```

src/main.jsx

```
if ("serviceWorker" in navigator) {
  window.addEventListener("load", () => {
    navigator.serviceWorker
      .register("/sw.js")
      .then((registration) => {
        console.log("Service Worker registered:", registration.scope);
      })
      .catch((err) => {
        console.error("Service Worker registration failed:", err);
      });
  });
}
```

Output:

Service Worker Registered



Service Worker Activated