

Experiment 8

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.

Theory:

Service Worker

Service Worker is a script that works on browser background without user interaction independently. Also, It resembles a proxy that works on the user side. With this script, you can track network traffic of the page, manage push notifications and develop “offline first” web applications with Cache API.

Things to note about Service Worker:

- A service worker is a programmable network proxy that lets you control how network requests from your page are handled.
- Service workers only run over HTTPS. Because service workers can intercept network requests and modify responses, "man-in-the-middle" attacks could be very bad.
- The service worker becomes idle when not in use and restarts when it's next needed. You cannot rely on a global state persisting between events. If there is information that you need to persist and reuse across restarts, you can use IndexedDB databases.

What can we do with Service Workers?

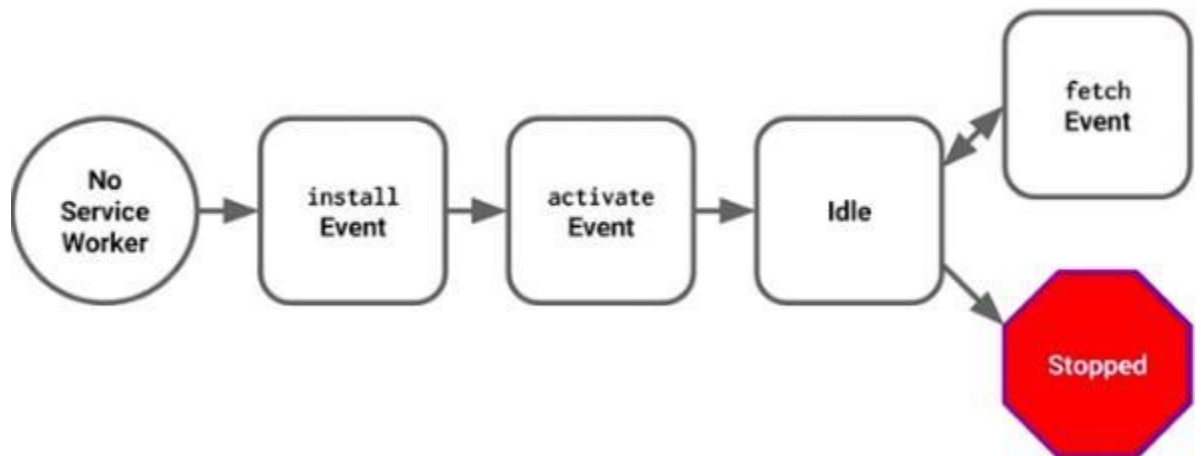
- You can dominate **Network Traffic**
You can manage all network traffic of the page and do any manipulations. For example, when the page requests a CSS file, you can send plain text as a response or when the page requests an HTML file, you can send a png file as a response. You can also send a true response too.
- You can **Cache**
You can cache any request/response pair with Service Worker and Cache API and you can access these offline content anytime.
- You can manage **Push Notifications**
You can manage push notifications with Service Worker and show any information message to the user.

- You can **Continue**
Although Internet connection is broken, you can start any process with Background Sync of Service Worker.

What can't we do with Service Workers?

- You can't access the **Window**
You can't access the window, therefore, You can't manipulate DOM elements. But, you can communicate to the window through post Message and manage processes that you want.
- You can't work it on **80 Port**
Service Worker just can work on HTTPS protocol. But you can work on localhost during development.

Service Worker Cycle



A service worker goes through three steps in its life cycle:

- Registration
- Installation
- Activation

Registration

To install a service worker, you need to register it in your main JavaScript code. Registration tells the browser where your service worker is located, and to start installing it in the background. Let's look at an example:

main.js

```
if ('serviceWorker' in navigator) {  
  navigator.serviceWorker.register('/service-worker.js')  
  .then(function(registration) {  
    console.log('Registration successful, scope is:', registration.scope);  
  })  
  .catch(function(error) {  
    console.log('Service worker registration failed, error:', error);  
  });  
}
```

This code starts by checking for browser support by examining **navigator.serviceWorker**. The service worker is then registered with `navigator.serviceWorker.register`, which returns a promise that resolves when the service worker has been successfully registered. The scope of the service worker is then logged with `registration.scope`. If the service worker is already installed, `navigator.serviceWorker.register` returns the registration object of the currently active service worker.

The scope of the service worker determines which files the service worker controls, in other words, from which path the service worker will intercept requests. The default scope is the location of the service worker file, and extends to all directories below. So if `service-worker.js` is located in the root directory, the service worker will control requests from all files at this domain.

You can also set an arbitrary scope by passing in an additional parameter when registering. For example:

ServiceWorker.js

```
var staticCacheName = "pwa-v1"; // Update cache name with versioning  
  
self.addEventListener("install", function (e) {  
  
  e.waitUntil(  
  
    caches.open(staticCacheName).then(function (cache) {  
  
      return cache.addAll([  
  
        '/', // Add your homepage and other important routes here  
  
        '/index.html',  
  
        '/assets/img/testimonial.png',  
  
      ]);  
    })  
  );  
});
```

```
});

self.addEventListener('activate', function(event) {

  event.waitUntil(

    caches.keys().then(function(cacheNames) {

      return Promise.all(

        cacheNames.filter(function(cacheName) {

          // Return true if you want to remove this cache, false otherwise

          return cacheName.startsWith('pwa-') && cacheName !== staticCacheName;

        }).map(function(cacheName) {

          return caches.delete(cacheName);

        })

      );

    })

  );

});

self.addEventListener("fetch", function (event) {

  event.respondWith(

    caches.match(event.request).then(function (response) {

      return response || fetch(event.request);

    })

  );

});
```

Add this in index.html

```
<script>

  window.addEventListener('load', () => {

    registerSW();

  });

  async function registerSW() {

    if ('serviceWorker' in navigator) {

      try {

        await navigator.serviceWorker.register('/serviceworker.js', { scope: '/' });

        console.log('SW registration successful');

      } catch (error) {

        console.log('SW registration failed:', error);

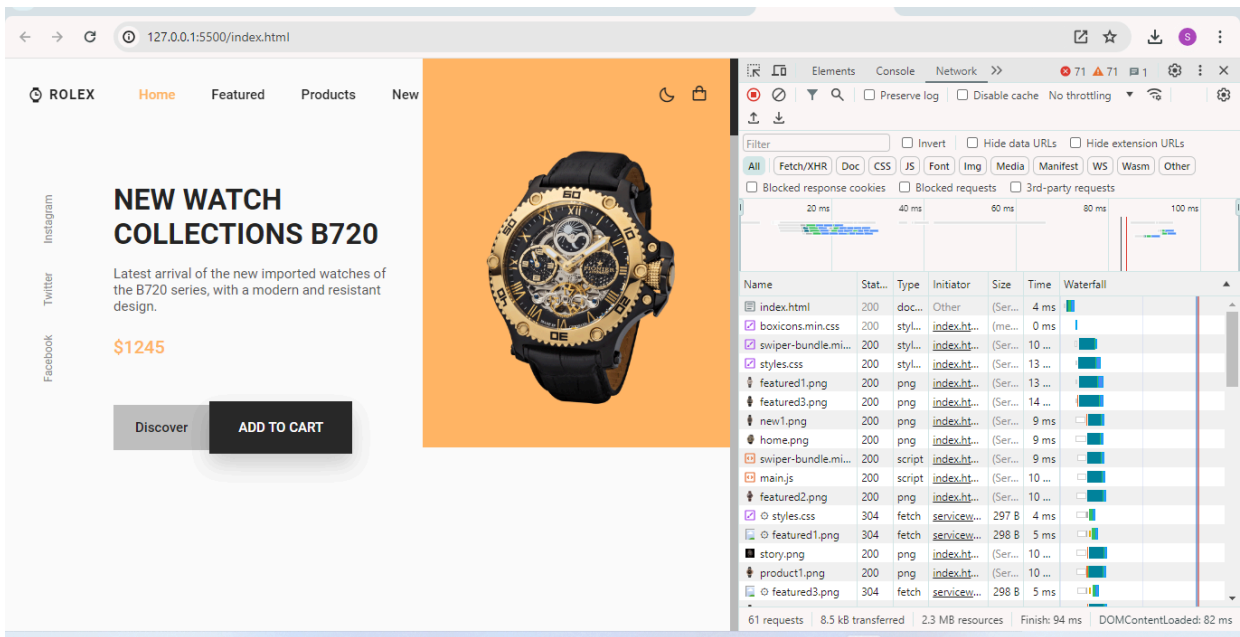
      }

    } else {

      console.log('Service Worker is not supported in this browser.');
```

Output:

1.Cache



2.Service Worker

 [Rolex](#)

- [Home](#)
- [Featured](#)
- [Products](#)
- [New](#)

×



×

My Cart

Jazzmaster

\$1050

− 1 +



Rose Gold

\$850

− 1 +



Longines Rose

\$980

− 1 +



3 items \$2880

[Facebook](#) [Twitter](#) [Instagram](#)

March 27, 2021

Raul Zaman Director of a company

→
←



Conclusion: In conclusion, testing the offline functionality of a Progressive Web App (PWA) is crucial to ensure a reliable user experience, especially in scenarios where internet connectivity may be unreliable or unavailable. By simulating an offline environment and thoroughly testing the behavior of the app, including service worker registration, caching of resources, and progressive enhancement, developers can ensure that the app continues to function seamlessly even without an internet connection.