

## Experiment No. 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.

#### 1. Network Proxy:

- Service workers act as an intermediary between your web page and the network.
- They intercept all outgoing HTTP requests made by your application. They can choose how to handle these requests:
- Serve content from a local cache if available.

#### 2. Offline Capabilities:

- Service workers enable offline functionality by allowing caching of essential application resources (HTML, CSS, JavaScript, images).
- When a user is offline, the service worker can retrieve the requested content from the cache, providing a seamless experience even without an internet connection.

#### 3. HTTPS Requirement:

- Due to security concerns, service workers can only function on HTTPS connections.

- This ensures secure communication between the service worker, your application, and the server.

## 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.

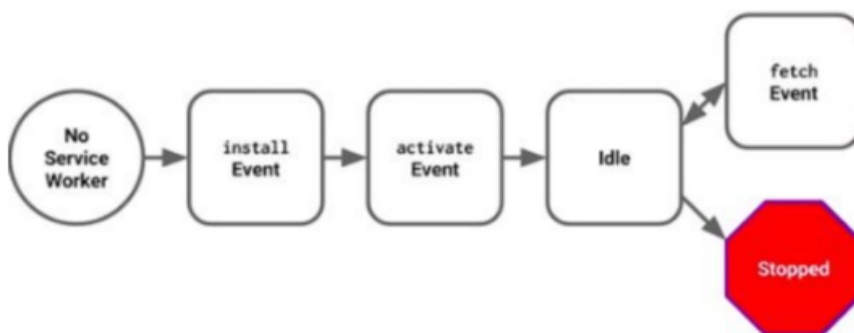
- You can manage Push Notifications

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.

## Service Worker Cycle



## Steps for coding and registering a service worker for your E-commerce PWA completing the install and activation process:

1. Create the Service Worker File (sw.js):

```
JS sw.js > ...
1  self.addEventListener("install", function (event) { event.waitUntil(preLoad());
2  });
3
4  var filesToCache = ['/index.html',
5  ];
6
7  var preLoad = function () {
8  return caches.open("offline").then(function (cache) {
9  // caching index and important routes
10 return cache.addAll(filesToCache);
11 });
12 };
13
14 self.addEventListener("fetch", function (event) { event.respondWith(checkResponse(event.request)).
15 return returnFromCache(event.request);
16 });
17 event.waitUntil(addToCache(event.request));
18 });
19
20 var checkResponse = function (request) { return new Promise(function (fulfill, reject) {
21 fetch(request).then(function (response) { if (response.status !== 404) {
22 fulfill(response);
23
24
25 } else {
26 reject();
27 }
28 } reject);
```

2. Register the Service Worker:

In your main JavaScript file (e.g., main.js or app.js), add the following code:

```
// Check if browser supports service workers
if (navigator.serviceWorker) {
  console.log("Service Worker Supported");

  // Start registration process on page load
  window.addEventListener("load", () => {
    navigator.serviceWorker

    // The register function takes as argument
    // the file path to the worker's file
    .register("./sw.js")

    // Gives us registration object
    .then(reg => console.log("Service Worker Registered"))
    .catch(swErr => console.log(
      `Service Worker Error: ${swErr}`));
  });
}
```

## Output

The screenshot displays a web application titled "Welcome to Cake Shop" and the Chrome DevTools interface showing the Service Workers panel.

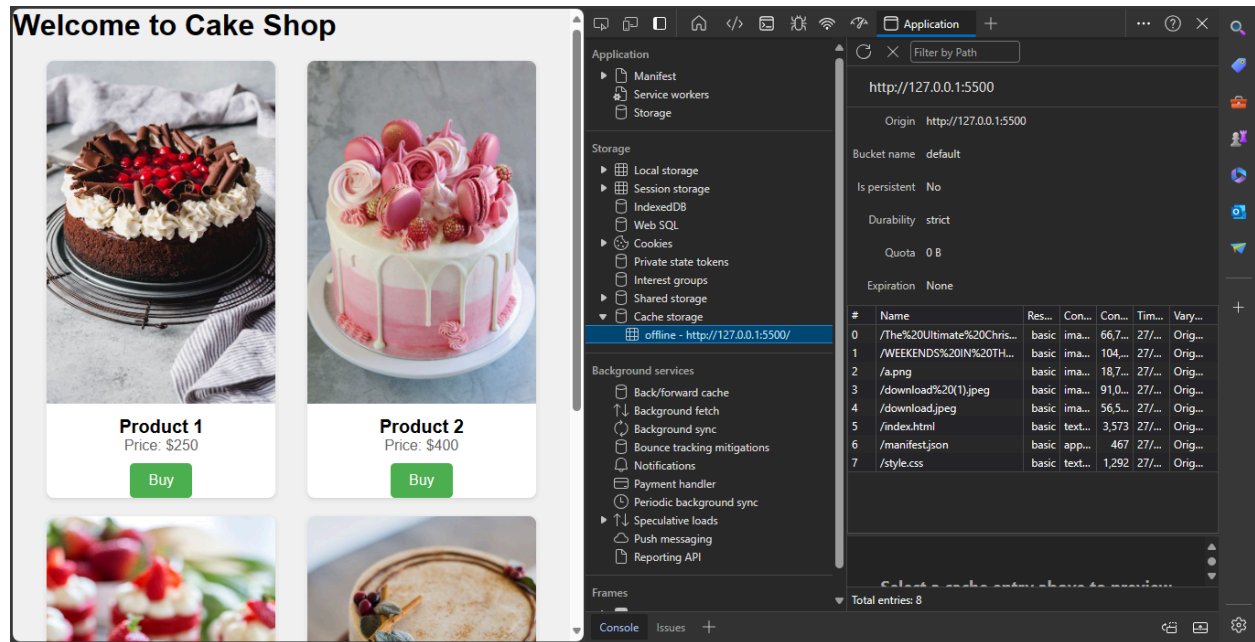
**Web Application:**

- Product 1:** A chocolate cake with white cream and red berries. Price: \$250. [Buy](#)
- Product 2:** A pink cake with white cream and pink macarons. Price: \$400. [Buy](#)

**DevTools - Service Workers Panel:**

- Application:** Manifest, Service workers, Storage.
- Storage:** Local storage, Session storage, IndexedDB, Web SQL, Cookies, Private state tokens, Interest groups, Shared storage, Cache storage, offline - http://127.0.0.1:5500/.
- Background services:** Back/forward cache, Background fetch, Background sync, Bounce tracking mitigations, Notifications, Payment handler, Periodic background sync, Speculative loads, Push messaging, Reporting API.
- Service workers:** ☐ Offline, ☐ Update on reload, ☐ Bypass for network.
- http://127.0.0.1:5500/** [Network requests](#) [Update](#) [Unrec](#)
  - Source: [sw.js](#)
  - Received 27/3/2024, 9:54:23 am
  - Status: ● #74 activated and is running [stop](#)
  - Clients: [http://127.0.0.1:5500/index.html focus](#), [http://127.0.0.1:5500/index.html focus](#)
  - Push: [Test push message from DevTools...](#) [Pk](#)
  - Sync: [test-tag-from-devtools](#) [Sy](#)
  - Periodic Sync: [test-tag-from-devtools](#) [Periodic Sy](#)
  - Update Cycle:

Version	Update Activity	Timeline
▶ #74	Install	
▶ #74	Wait	
▶ #74	Activate	<div style="width: 100%;"></div>
- http://127.0.0.1:5500/ - deleted** [Network requests](#) [Update](#) [Unrec](#)
  - Source: [sw.js](#)



**Conclusion :** I have understood and successfully registered a service worker, and completed the install and activation process for a new service worker for the E-commerce PWA.