

**Name : Nilanchala Panda**  
**Div . : 41**

**Batch : B**  
**Roll No . : 41**

## **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.

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?

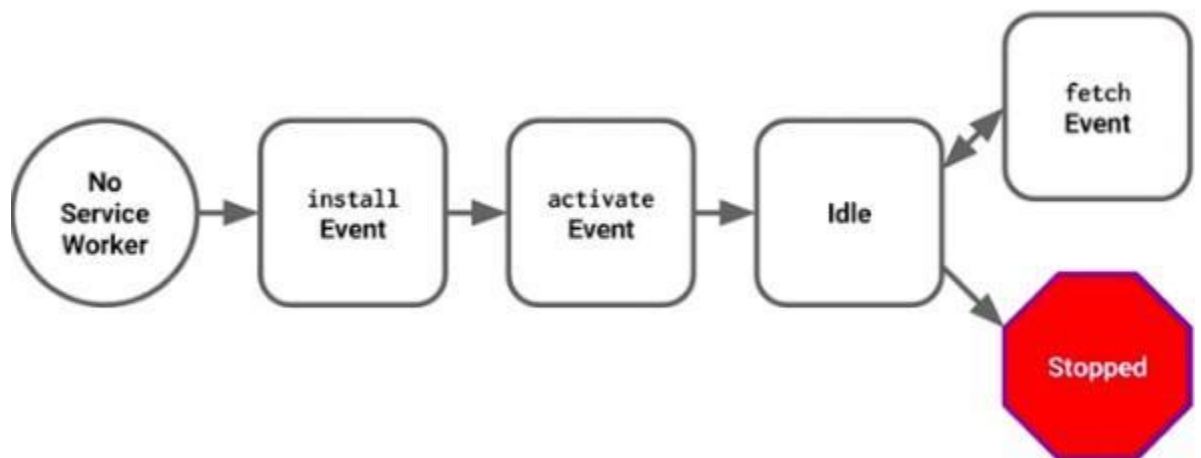
- You can dominate Network Traffic  
You can manage all network traffic of the page and do any manipulations. For example, when the page requests a CSS file, you can send plain text as a response or when the page requests an HTML file, you can send a png file as a response. You can also send a true response 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

**CODE -**

**index.html -**

```

<!DOCTYPE html>
<html lang="en">

```

```

<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
  <title>E-commerce PWA</title>
  <link rel="stylesheet" href="main.css" />
</head>
<body>
  <h1>Welcome to Our E-commerce Store</h1>
  <!-- E-commerce content goes here -->
  <script src="app.js"></script>
</body>
</html>

```

### service-worker.js -

```

const cacheName = "ecommerce-pwa-v1";
const assetsToCache = [
  "/",
  "/index.html",
  "/app.js",
];

self.addEventListener("install", (event) => {
  event.waitUntil(
    caches.open(cacheName).then((cache) => {
      return cache.addAll(assetsToCache);
    })
  );
});

self.addEventListener("activate", (event) => {
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames
          .filter((name) => {
            return name !== cacheName;
          })
          .map((name) => {
            return caches.delete(name);
          })
      );
    })
  );
});

```

```

    })
  );
})
);
});

```

## app.js -

```

if ('serviceWorker' in navigator) {
  window.addEventListener('load', () => {
    navigator.serviceWorker.register('/service-worker.js')
      .then(registration => {
        console.log('Service Worker registered with scope:',
registration.scope);
      })
      .catch(error => {
        console.error('Service Worker registration failed:', error);
      });
  });
}

```

## OUTPUT -

The screenshot shows a web browser at the address `127.0.0.1:5500` displaying the text "Welcome to Our E-commerce Store". The Chrome DevTools interface is open, with the "Service workers" panel selected. The panel shows two service workers for the URL `http://127.0.0.1:5500/`, both sourced from `service-worker.js` and received on 3/27/2024 at 11:14:30 AM. The status for both is "#190 is redundant". The "Update Cycle" table shows the following data:

Version	Update Activity	Timeline
#190	Install	
#190	Activate	

The left sidebar of DevTools shows the "Application" tab with sections for "Storage" (Local storage, Session storage, IndexedDB, Web SQL, Cookies, Private state tokens, Interest groups, Shared storage, Cache storage) and "Background services" (Back/forward cache, Background fetch, Background sync, Bounce tracking mitig., Notifications, Payment handler, Periodic background sync, Cooperative loading).