Practical 9

Name: Sharvari Kishor More
Class: D15B
Roll No.: 35

Aim:

To implement Service worker events like fetch, sync and push for E-commerce PWA.

Theory:

Service workers are powerful scripts that run in the background of Progressive Web Apps (PWAs), enabling features like offline access, background sync, and push notifications. In this experiment, we focused on three key service worker events: fetch, sync, and push.

In this experiment, we enhanced our E-commerce Progressive Web App (PWA) by implementing advanced Service Worker events — **fetch**, **sync**, and **push**.

- The **install** and **activate** events handle caching of files and removal of old caches.
- The **fetch** event intercepts network requests and serves cached resources for better performance and offline support.
- The **sync** event simulates background synchronization when the app regains connectivity.
- The **push** event listens for push notifications and displays them, improving user engagement.

These events help in making PWAs more reliable, fast, and interactive even in low or no network conditions.

Code:

Index.html

```
<script>
    if ("serviceWorker" in navigator) {
      window.addEventListener("load", () => {
        navigator.serviceWorker
          .register("/serviceworker.js")
          .then((registration) => {
            console.log("✅ Service Worker registered! Scope:", registration.scope);

            // 🔔 Request Push Notification Permission
            if ("PushManager" in window) {
              Notification.requestPermission().then((permission) => {
                if (permission === "granted") {
```

```
            console.log("🔔 Push notifications granted.");
          } else {
            console.log("🔕 Push notifications denied.");
          }
        });
      }

      // 🔄 Register Background Sync
      if ("SyncManager" in window) {
        navigator.serviceWorker.ready.then((swReg) => {
          swReg.sync.register("sync-data").then(() => {
            console.log("🔄 Sync registered");
          }).catch((err) => {
            console.log("❌ Sync registration failed:", err);
          });
        });
      }
    })
    .catch((error) => {
      console.log("❌ Service Worker registration failed:", error);
    });
  });
}
</script>
```

Serviceworker.js

```
const CACHE_NAME = 'festival-combo-shop-v1';
const FILES_TO_CACHE = [
  '/',
  '/index.html',
  '/styles.css',
  '/manifest.json',
  '/serviceworker.js',
  '/color.webp',
  '/diwalisweets.webp',
  '/holicolors.webp',
  '/christmas.webp',
];
```

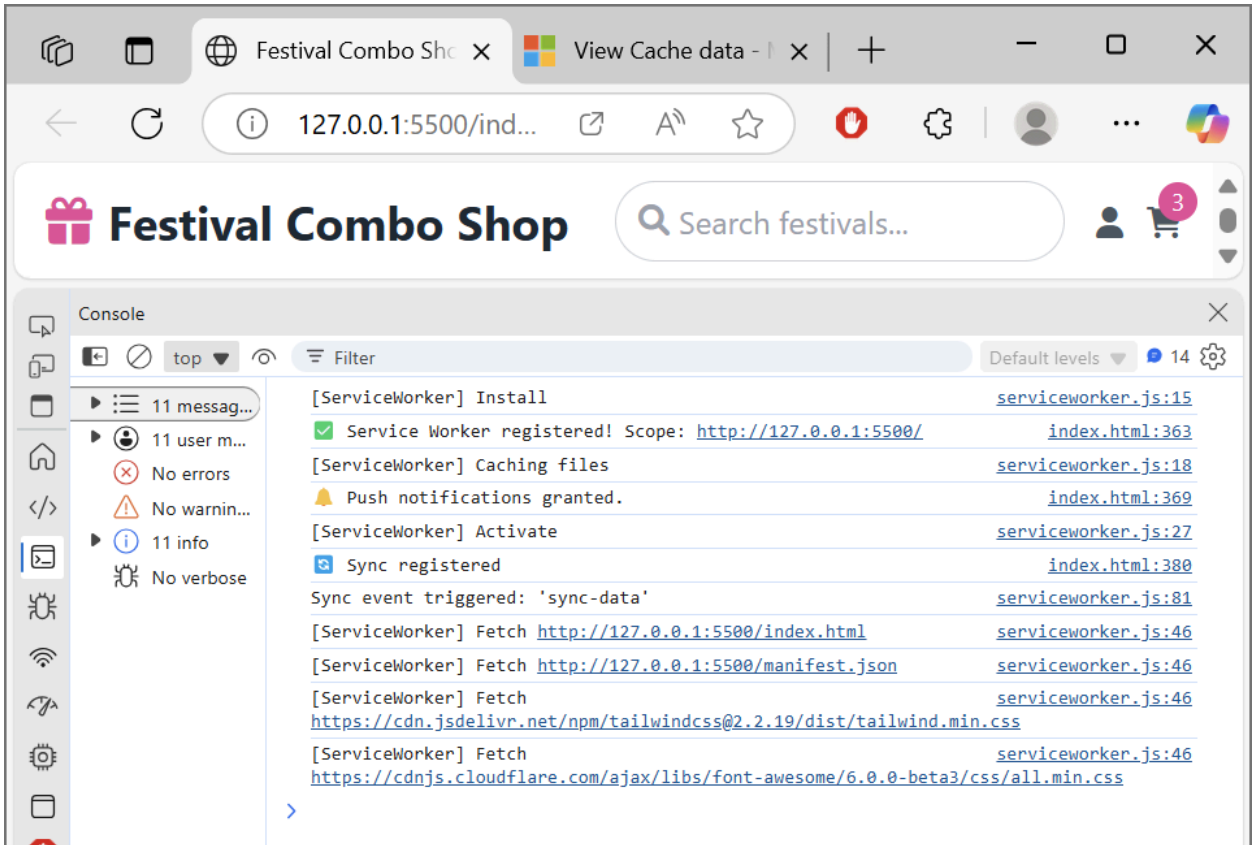
```
// Install Event
self.addEventListener("install", (event) => {
  console.log("[ServiceWorker] Install");
  event.waitUntil(
    caches.open(CACHE_NAME).then((cache) => {
      console.log("[ServiceWorker] Caching files");
      return cache.addAll(FILES_TO_CACHE);
    })
  );
});

// Activate Event
self.addEventListener("activate", (event) => {
  console.log("[ServiceWorker] Activate");
  event.waitUntil(
    caches.keys().then((keyList) =>
      Promise.all(
        keyList.map((key) => {
          if (key !== CACHE_NAME) {
            console.log("[ServiceWorker] Removing old cache", key);
            return caches.delete(key);
          }
        })
      )
    )
  );
  return self.clients.claim();
});

// Enhanced Fetch Event
self.addEventListener("fetch", (event) => {
  console.log("[ServiceWorker] Fetch", event.request.url);
  const requestURL = new URL(event.request.url);

  // If request is same-origin, use Cache First
  if (requestURL.origin === location.origin) {
    event.respondWith(
      caches.match(event.request).then((cachedResponse) => {
```

```
        return ( cachedResponse ||
          fetch(event.request).catch(() => caches.match("offline.html"))
        );
      })
    );
  } else {
    // Else, use Network First
    event.respondWith(
      fetch(event.request)
        .then((response) => {
          return response;
        })
        .catch(() =>
          caches.match(event.request).then((res) => {
            return res || caches.match("offline.html");
          })
        )
    );
  }
});

// Sync Event (simulation)
self.addEventListener("sync", (event) => {
  if (event.tag === "sync-data") {
    event.waitUntil(
      (async () => {
        console.log("Sync event triggered: 'sync-data'");
        // Here you can sync data with server when online
      })()
    );
  }
});

// Push Event
self.addEventListener("push", function (event) {
  if (event && event.data) {
    let data = {};
    try {
```

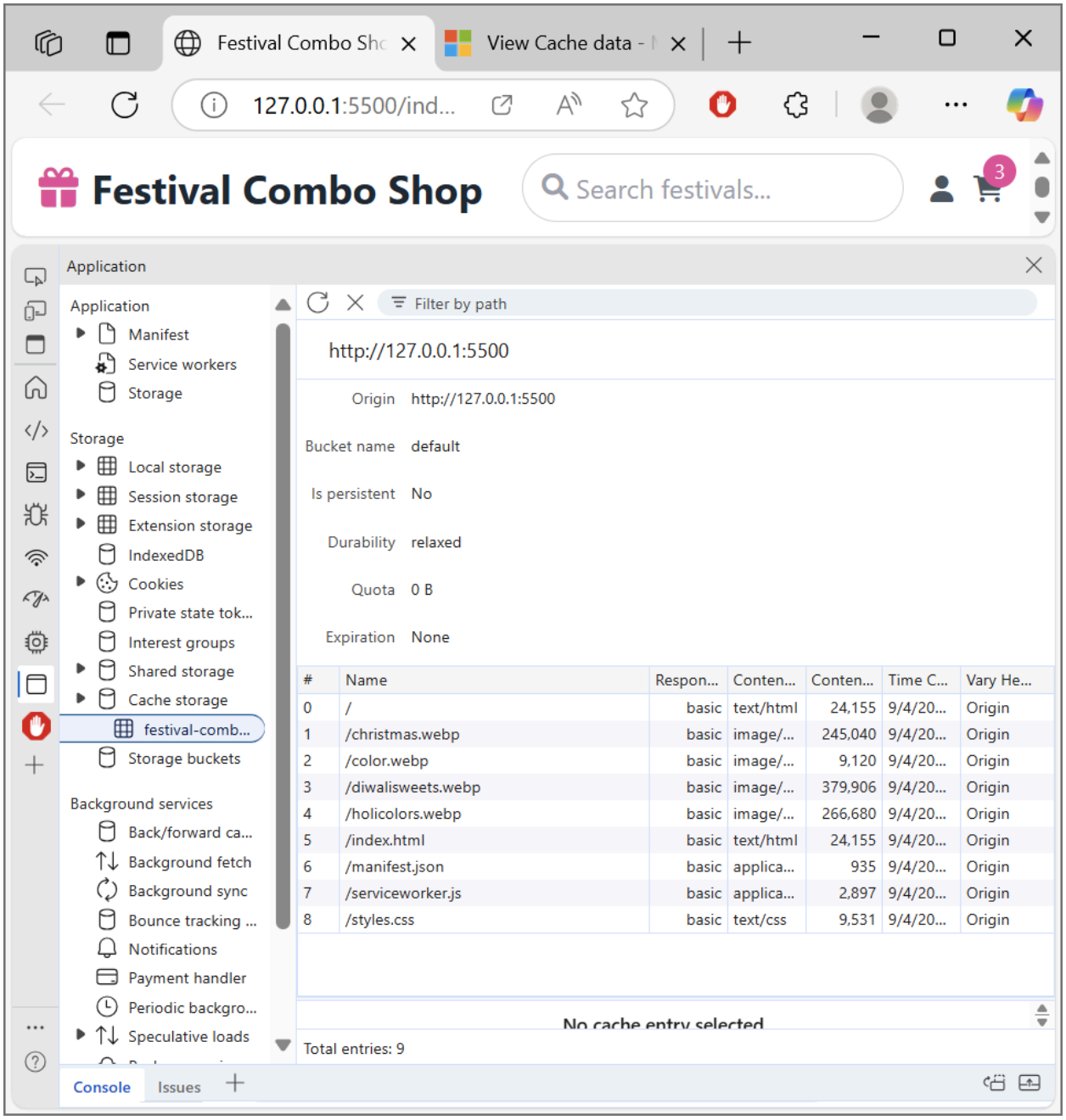
```
      data = event.data.json();
    } catch (e) {
      data = {
        method: "pushMessage",
        message: event.data.text(),
      };
    }

    if (data.method === "pushMessage") {
      console.log("Push notification sent");
      event.waitUntil(
        self.registration.showNotification("Maharashtrian Handloom", {
          body: data.message,
        })
      );
    }
  }
});
```

Output:

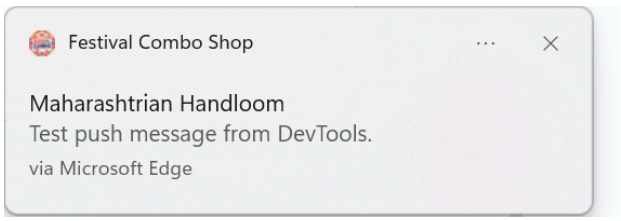
Working of service worker



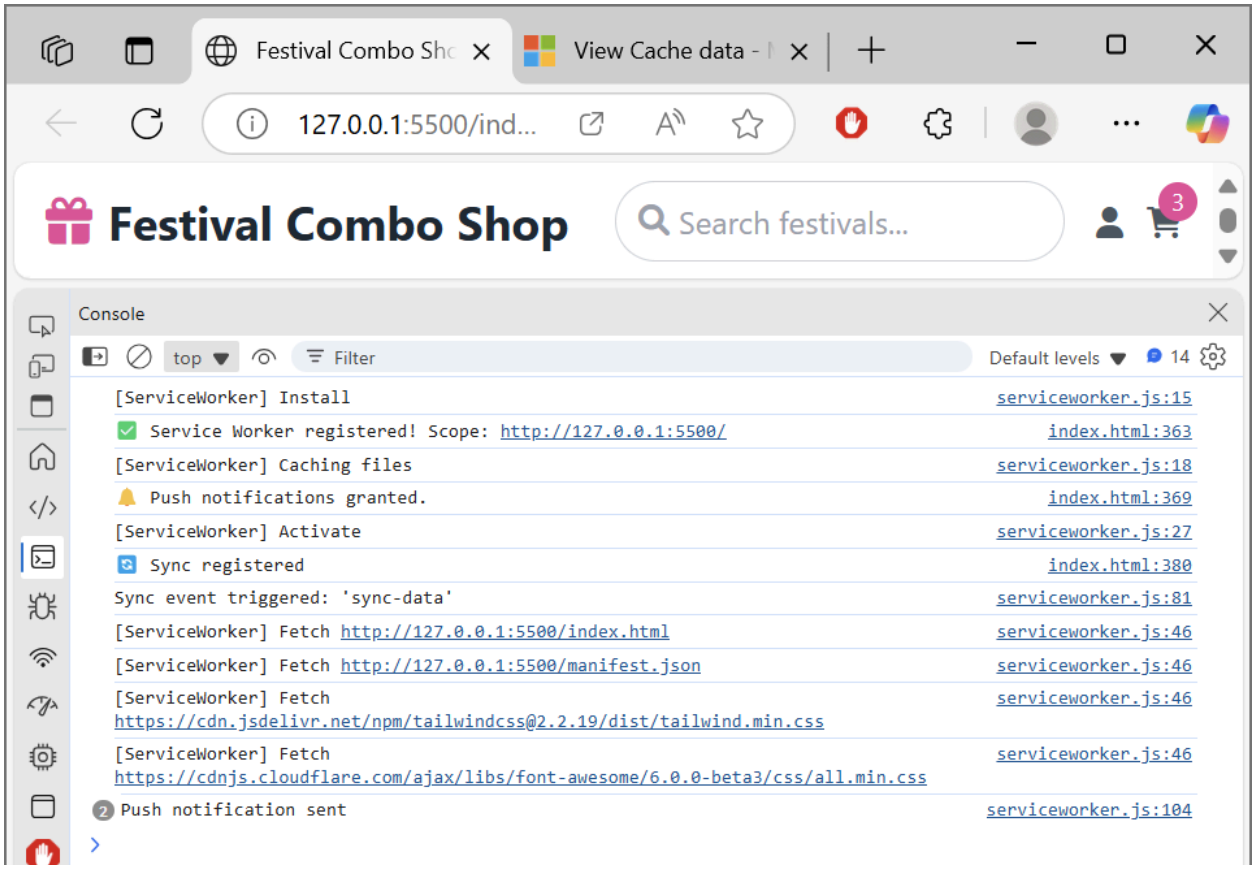
Cache



Sending of push notification.



Test: Push Notification sent



Conclusion:

By implementing fetch, sync, and push events in the Service Worker, we made our E-commerce PWA capable of working offline, syncing data in the background, and sending push notifications. This ensures a better user experience and highlights the power of PWAs in building modern web applications.