Shravani Pore
D15A_45
Batch B

MAD PWA Lab 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.

Things we can 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.

Things we can't do with service worker:

- 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.

Code:

Service-worker.js:

```
var cacheName = "Headphone";
self.addEventListener("install", (event) => {
  event.waitUntil(
    caches.open(cacheName).then((cache) => {
return cache.addAll([
  "/",
  "/index.html",
  // "/index.js",
  "css/grid.css",
  "/product-detail.html",
  "/products.html",
]);
    })
  );
});
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);
          })
      );
```

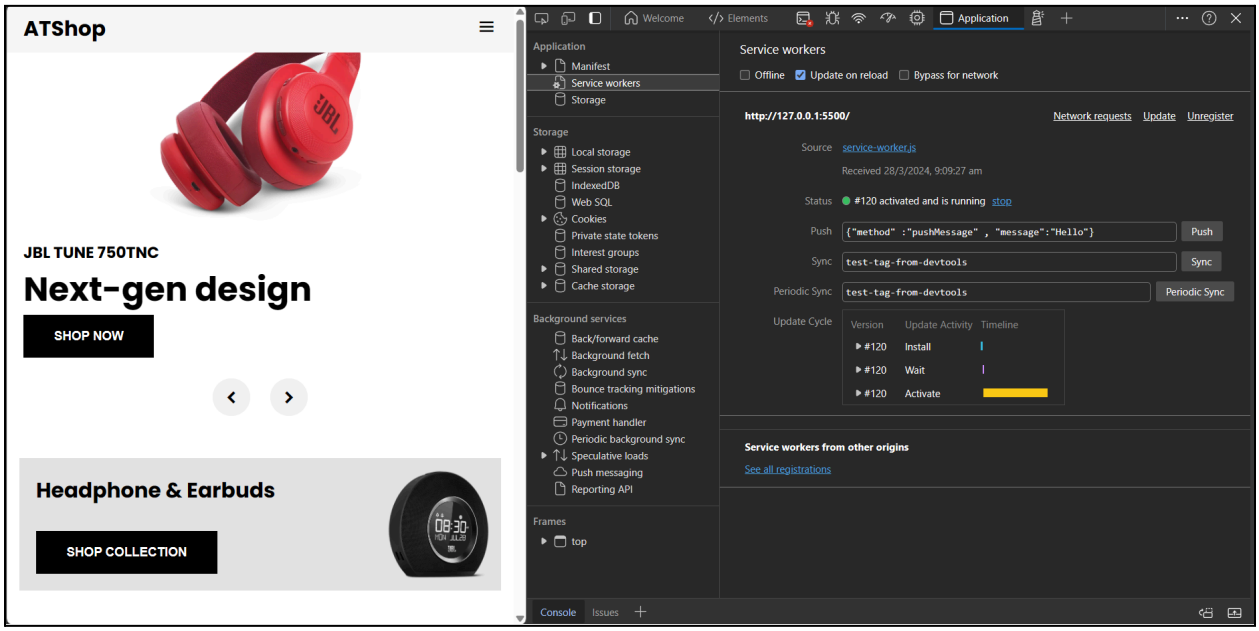
```
    })
  );
});
```

App.js:

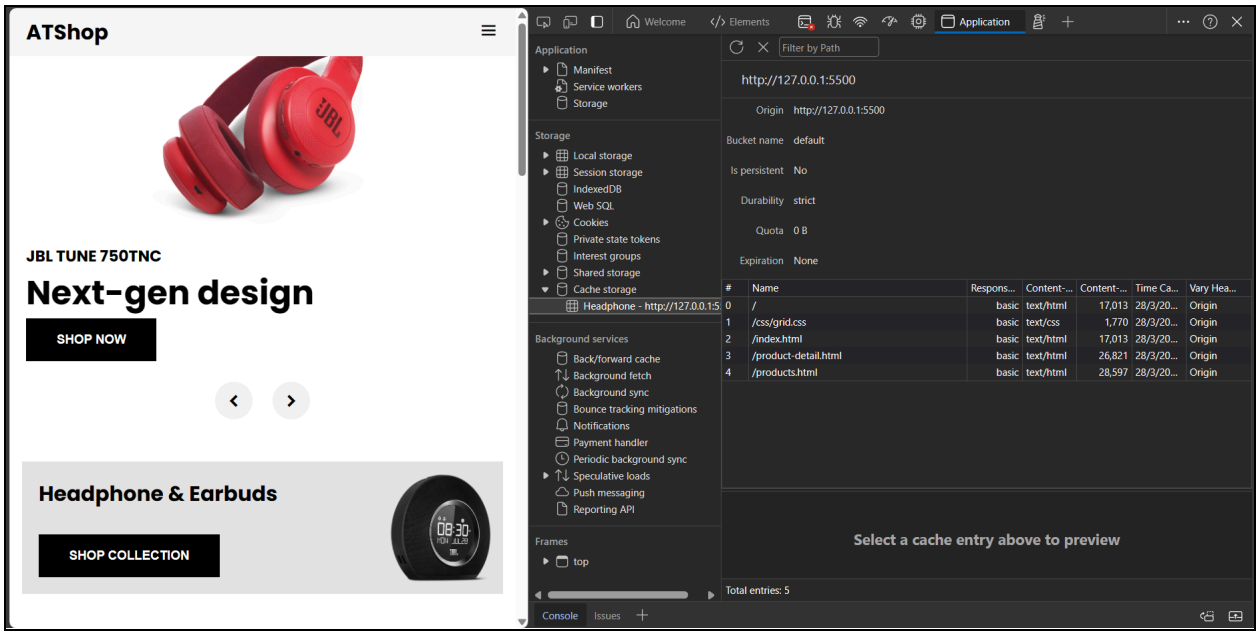
```
if ("serviceWorker" in navigator) {
  window.addEventListener("load", () => {
    navigator.serviceWorker
      .register("/service-worker.js")
      .then((registration) => {
        console.log(
          "Service Worker registered with scope:",
          registration.scope
        );
      })
      .catch((error) => {
        console.error("Service Worker registration failed:", error);
      });
  });
}
if ("Notification" in window) {
  Notification.requestPermission().then(function (permission) {
    if (permission === "granted") {
      console.log("Notification permission granted.");
    } else {
      console.warn("Notification permission denied.");
    }
  });
}
```

Output:

Three dots on right corner ->more tools -> developer tools -> Applications -> Service workers :



Three dots on right corner ->more tools -> developer tools -> Applications -> Cache storage :



Conclusion:

Successfully implemented a code and register a service worker, and completed the installation and activation process for a new service worker for the E-commerce PWA.