Vinay Israni 24 **Exp 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:

A service worker is a type of web worker that runs in the background of a web application, separate from the main browser thread. It is written in JavaScript and provides a way to run scripts in the background, independent of the user interface, allowing web applications to support offline functionality, push notifications, background sync, and more.

Here are some key points about service workers:

1. Offline Support: Service workers enable web applications to work offline by caching resources like HTML, CSS, JavaScript, and images. When the user is offline, the service worker can intercept network requests and serve cached content, providing a seamless offline experience.

2. Push Notifications: Service workers can receive push notifications from a server even when the web application is not open in the browser. This allows web apps to send notifications to users, similar to native mobile apps.

3. Background Sync: Service workers can schedule background syncs to send data to a server
when the device is online, even if the web app is not actively being used. This is useful for applications that need to periodically sync data with a server.

4. Improved Performance: By offloading tasks to a separate thread, service workers can improve the performance of web applications. They can handle resource-intensive operations without blocking the main thread, leading to smoother user experiences.

5. Security: Service workers run in a separate context from the main page, which enhances security by preventing malicious scripts from accessing sensitive information or modifying the DOM directly.

A service worker is a type of web worker in the browser that runs JavaScript code in the background, separate from the main web page. Its primary purpose is to enable various functionalities that enhance web applications. Here are the key things that a service worker can do:

1. Offline Support: One of the main tasks of a service worker is to enable offline functionality in web applications. It does this by caching resources such as HTML, CSS, JavaScript, images, and
other assets when the user first visits the website. When the user goes offline or experiences a network interruption, the service worker can intercept network requests and serve the cached content, allowing the web app to continue functioning without an internet connection.

2. Push Notifications: Service workers handle push notifications in web applications. They can receive push messages from a server even when the web app is not actively open in the browser. This enables web apps to send notifications to users, similar to how native mobile apps send notifications.

3. Background Sync: Service workers can schedule and perform background sync operations. This is useful for web applications that need to sync data with a server periodically, even when the app is not actively being used. For example, an email client web app can use a service worker to sync emails in the background.

4. Caching Strategies: Service workers implement caching strategies to optimize performance and resource management. Developers can define caching policies such as cache-first (serve content from the cache if available, otherwise fetch from the network), network-first (try to fetch content from the network first, falling back to the cache if offline), or stale-while-revalidate (serve stale content from the cache while fetching updated content from the network).
5. Performance Improvement: By running tasks in the background, separate from the main thread of the browser, service workers improve the performance of web applications. They can handle resource-intensive operations without blocking the user interface, leading to a smoother and more responsive user experience.

Implementation:
Steps :
1. Create a Service Worker File:
   - Create a new JavaScript file for your service worker, for example, service-worker.js.
   - Define the service worker code inside this file. The code will include event listeners for install and activate events, as well as caching strategies.
2. Register the Service Worker:
In your main HTML file (e.g., index.html), add a script tag to register the service worker.
Place the following code inside your HTML file:

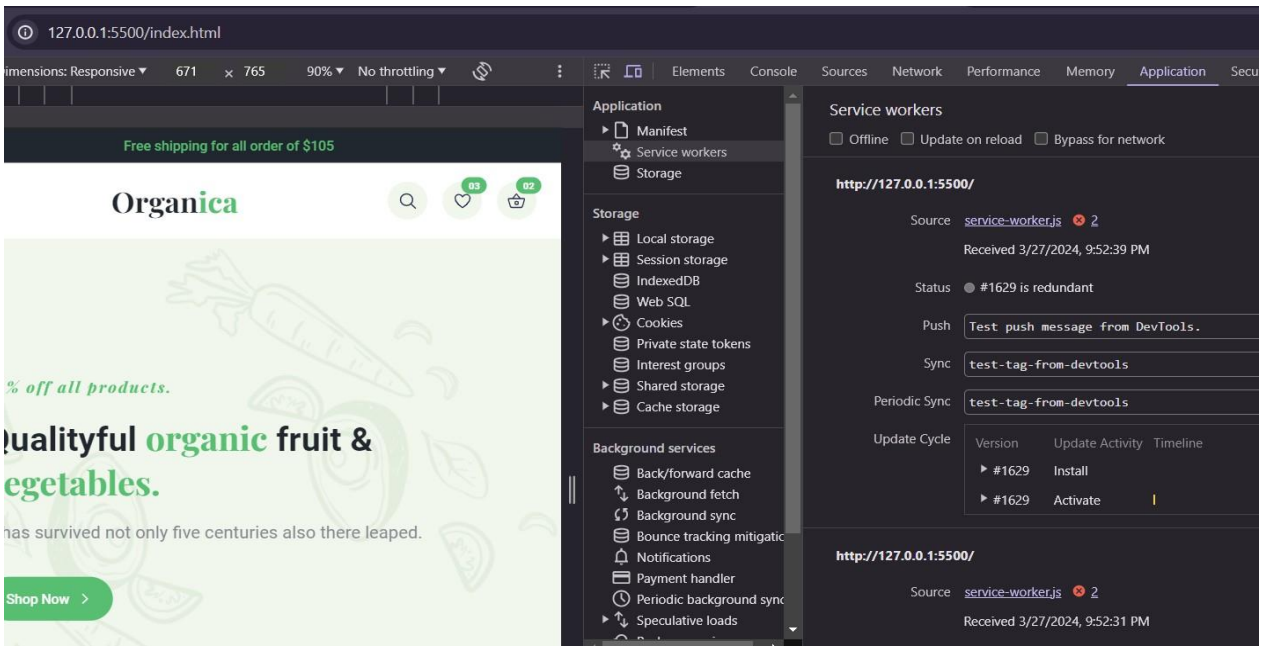
```
<script>
    if ('serviceWorker' in navigator) {
      window.addEventListener('load', function () {

navigator.serviceWorker.register('/service-worker.js').then(function
(registration) {
          console.log('Service Worker registered with scope:',
            registration.scope);
        }).catch(function (error) {
          console.error('Service Worker registration failed:', error);
        });
      });
    }
  </script>
```

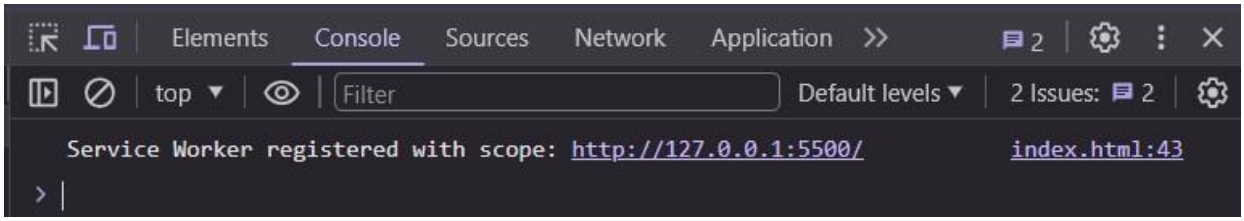
3. Serviceworker code :

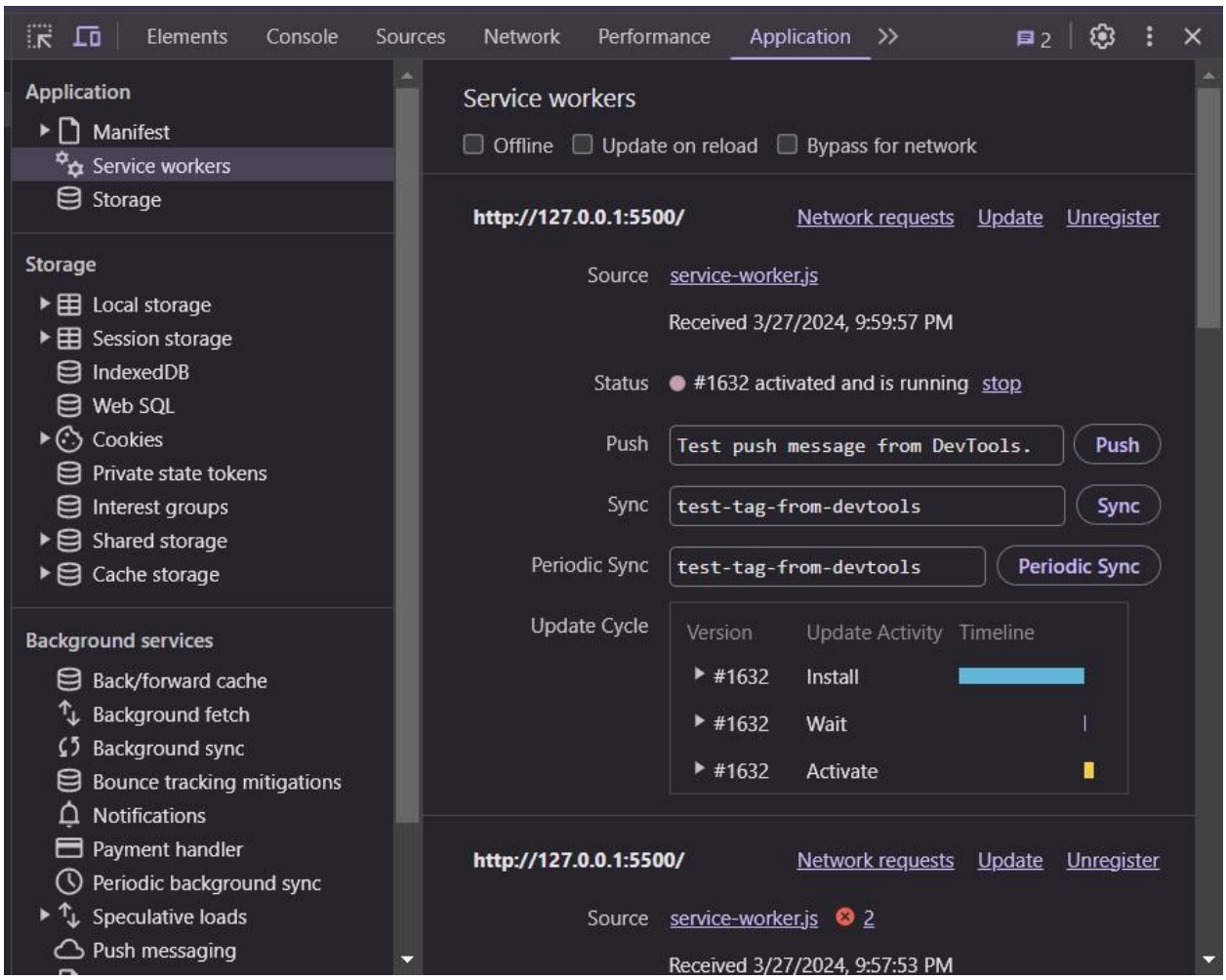
```
const CACHE_NAME = 'ecommerce-pwa-cache-v1';
const urlsToCache = [
    '/',
    '/index.html',
    '/logolaptop.jpeg',
];
```

```
self.addEventListener('install', (event) => {
    event.waitUntil(
        caches.open(CACHE_NAME)
            .then((cache) => {
                return cache.addAll(urlsToCache);
            })
    );
});
self.addEventListener('activate', (event) => {
    event.waitUntil(
        caches.keys()
            .then((cacheNames) => {
                return Promise.all(
                    cacheNames.filter((name) => {
                        return name !== CACHE_NAME;
                    }).map((name) => {
                        return caches.delete(name);
                    })
                );
            })
    );
});
self.addEventListener('fetch', (event) => {
    event.respondWith(
        caches.match(event.request)
            .then((response) => {
                return response || fetch(event.request);
            })
    );
});
```

4. Testing:
Save all your files and load your E-commerce PWA in a browser that supports service workers (e.g., Chrome, Firefox).
Open the developer tools (F12 or Ctrl+Shift+I), go to the Application tab, and check the Service Workers section to see if your service worker is registered and active.
Test the offline functionality by disconnecting from the internet and reloading the page. Ensure that cached content is served when offline.


127.0.0.1:5500/index.html
Free shipping for all order of $105
Organica
% off all products.
ualityful organic fruit &
egetables.
has survived not only five centuries also there leaped.
Shop Now
Elements
Console
Sources
Network
Performance
Memory
Application
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
IndexedDB
Web SQL
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking mitigatic
Notifications
Payment handler
Periodic background sync
Speculative loads
Service workers
Offline
Update on reload
Bypass for network
http://127.0.0.1:5500/
Source service-worker.js 2
Received 3/27/2024, 9:52:39 PM
Status #1629 is redundant
Push Test push message from DevTools.
Sync test-tag-from-devtools
Periodic Sync test-tag-from-devtools
Update Cycle
Version Update Activity Timeline
#1629 Install
#1629 Activate
http://127.0.0.1:5500/
Source service-worker.js 2
Received 3/27/2024, 9:52:31 PM


Elements
Console
Sources
Network
Application
2
top
Filter
Default levels
2 Issues: 2
Service Worker registered with scope: http://127.0.0.1:5500/
index.html:43



Conclusion:

Hence we have understood the working of PWA applications and learnt how to deploy Progressive web applications. Also we have coded and registered a service worker, and completed the installation and activation process for a new service worker for the Laptop E-commerce PWA.