Name:Gaurang Mapuskar Class:D15B
Roll. No:35

Experiment 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 Workers are powerful scripts that operate in the background of a browser, providing developers with significant control over network requests and enabling the creation of robust web applications. Think of them as programmable network proxies that allow you to manage network traffic, implement caching strategies, handle push notifications, and develop "offline first" web experiences using the Cache API.

Key Concepts and Features:

1. Network Traffic Control:
Service Workers act as intermediaries between your web app and the network. They give you the ability to intercept and manipulate network requests made by your web pages. For instance, you can modify responses, serve cached content, or handle requests based on specific conditions.

2. Caching Capabilities:

One of the most significant advantages of Service Workers is their ability to cache resources. Using the Cache API, you can store and retrieve request/response pairs, allowing your web app to function seamlessly even when offline. This enables faster loading times and a smoother user experience.

3. Push Notification Management:

Service Workers empower developers to implement push notifications in web applications. This feature enables you to send relevant and timely updates to users, enhancing engagement and user interaction.

4. Background Processes with Background Sync:

Even in scenarios where the internet connection is unreliable or unavailable, Service Workers can initiate processes using Background Sync. This ensures that critical tasks, such as data synchronization or updates, can continue seamlessly.

Limitations of Service Workers:

1. No Access to the Window Object:
Service Workers operate in a separate thread from the main browser window, meaning they do not have direct access to the DOM (Document Object Model). As a result, you cannot directly manipulate DOM elements within a Service Worker. However, communication with the window is possible through postMessage, allowing you to manage certain processes indirectly.

2. HTTPS Requirement:

For security reasons, Service Workers are limited to running on HTTPS websites. This is crucial to prevent "man-in-the-middle" attacks, where malicious actors could intercept and manipulate network requests.

3. Cannot Work on Port 80:

Service Workers are designed to work exclusively over HTTPS. During development, you can utilize localhost for testing purposes. However, in production, your website must be served over HTTPS to enable Service Worker functionality.

Service Worker Life Cycle:

1. Registration:
To start using a Service Worker, you must first register it in your main JavaScript code. This registration process informs the browser about the Service Worker's location and begins the installation in the background.

Name:Gaurang Mapuskar
Roll. No:35
Class:D15B

```
async function registerSW() {
  if ('serviceWorker' in navigator) {
    try {
      await navigator
            .serviceWorker
            .register('serviceworker.js');
    }
    catch (e) {
      console.log('SW registration failed');
    }
  }
}
```

2. Installation:
Once registered, the browser will attempt to install the Service Worker. This occurs when the Service Worker is considered new or updated compared to the previously installed version. An 'install' event is triggered during this stage, allowing developers to perform tasks such as caching resources for offline use.

```
self.addEventListener("install", function (e) {
  e.waitUntil(
    caches.open(staticCacheName).then(function (cache) {
      return cache.addAll(["/"]);
    })
  );
});
```

3. Activation:
After successful installation, the Service Worker transitions into the activation stage. During activation, the new Service Worker takes control, replacing the previous version. An 'activate' event is triggered, providing an opportunity to clean up old caches or perform other necessary tasks.

Name:Gaurang Mapuskar Class:D15B
Roll. No:35

```
self.addEventListener('activate', function(event) {
    event.waitUntil(
        // Perform cleanup tasks or cache management here
        // For example, deleting outdated caches
        caches.keys().then(function(cacheNames) {
            return Promise.all(
                cacheNames.filter(function(cacheName) {
                    // Check if the cache name is outdated and needs to be deleted
                    // For example, you might compare cache names with the current cache version
                }).map(function(cacheName) {
                    // Delete the outdated cache
                    return caches.delete(cacheName);
                })
            );
        })
    );
});
```

Implementation:

serviceworker.js

```
var staticCacheName = "GAU";

self.addEventListener("install", function (e) {
  e.waitUntil(
    caches.open(staticCacheName).then(function (cache) {
      return cache.addAll(["/"]);
    })
  );
});

self.addEventListener("fetch", function (event) {
  console.log(event.request.url);

  event.respondWith(
    caches.match(event.request).then(function (response) {
      return response || fetch(event.request);
    })
  );
});

self.addEventListener('install', function(event) {
```

```
  // Perform some task
  });


  self.addEventListener('activate', function(event) {
    event.waitUntil(
      // Perform cleanup tasks or cache management here
      // For example, deleting outdated caches
      caches.keys().then(function(cacheNames) {
        return Promise.all(
          cacheNames.filter(function(cacheName) {
            // Check if the cache name is outdated and needs to be deleted
            // For example, you might compare cache names with the current cache
version
          }).map(function(cacheName) {
            // Delete the outdated cache
            return caches.delete(cacheName);
          })
        );
      })
    );
  });
```

sw.js

```
self.addEventListener('install', function(event) {
  event.waitUntil(
    caches.open('offline')
      .then(function(cache) {
        return cache.addAll([
          '/',
          '/index.html',
          // Add other essential static assets (CSS, JavaScript, images)
        ]);
      })
  );
});
```

```
self.addEventListener('fetch', function(event) {
  event.respondWith(
    fetch(event.request)
      .catch(function() {
        return caches.match(event.request)
          .then(function(matching) {
            return matching || caches.match('offline.html');
          });
      })
  );
});
```

Name:Gaurang Mapuskar
Roll. No:35

Class:D15B

Output: