Name - Gaurang A Raorane | Div - D15A

Roll no - 49 | Batch - C

Experiment - 9

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

Theory -
Service Worker is a powerful script that operates in the background of a browser independently, similar to a proxy working on the user's side. It allows developers to track network traffic, manage push notifications, and develop "offline-first" web applications using the Cache API. Here are some important aspects to note about Service Workers:

- Programmable Network Proxy: Service Worker acts as a programmable network proxy, giving developers control over how network requests from their web page are handled.

- HTTPS Requirement: Service Workers only function over HTTPS due to security reasons. They can intercept network requests and modify responses, so running them over insecure connections could pose significant security risks.

- Idle State and Restart: When not in use, a Service Worker becomes idle and restarts when needed next. Developers can't rely on a global state persisting between events. IndexedDB databases can be used to persist and reuse information across restarts.

- Fetch Event: This event allows tracking and managing of page network traffic. Developers can implement strategies such as "cache first" and "network first" approaches. For example:
  - CacheFirst: If the requested resource is cached, return the cached response; otherwise, request a new response from the network.
  - NetworkFirst: Attempt to get an updated response from the network. If successful, cache and return the new response. If unsuccessful, check if the request is cached and return it if available.

- Sync Event: Background Sync API delays a process until the internet connection is stable. For example, in an email client application, if the internet connection is lost while composing an email, Background Sync can ensure the email is sent once the connection is restored.

- Push Event: Handles push notifications received from the server. Developers can apply various actions based on received data. For instance, displaying a notification to the user. The `Notification.requestPermission();` line is necessary to show notifications to the user.

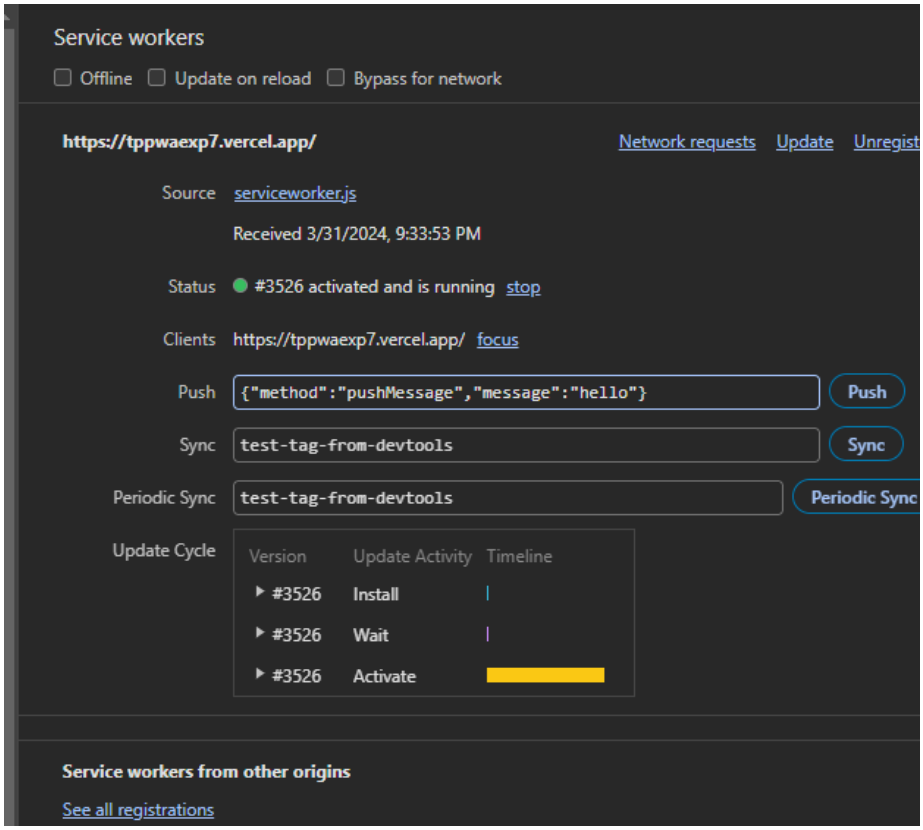
Service Workers offer developers extensive capabilities for enhancing web applications, including offline functionality, network traffic management, and push notifications. However, they require careful implementation due to security considerations and their asynchronous nature.

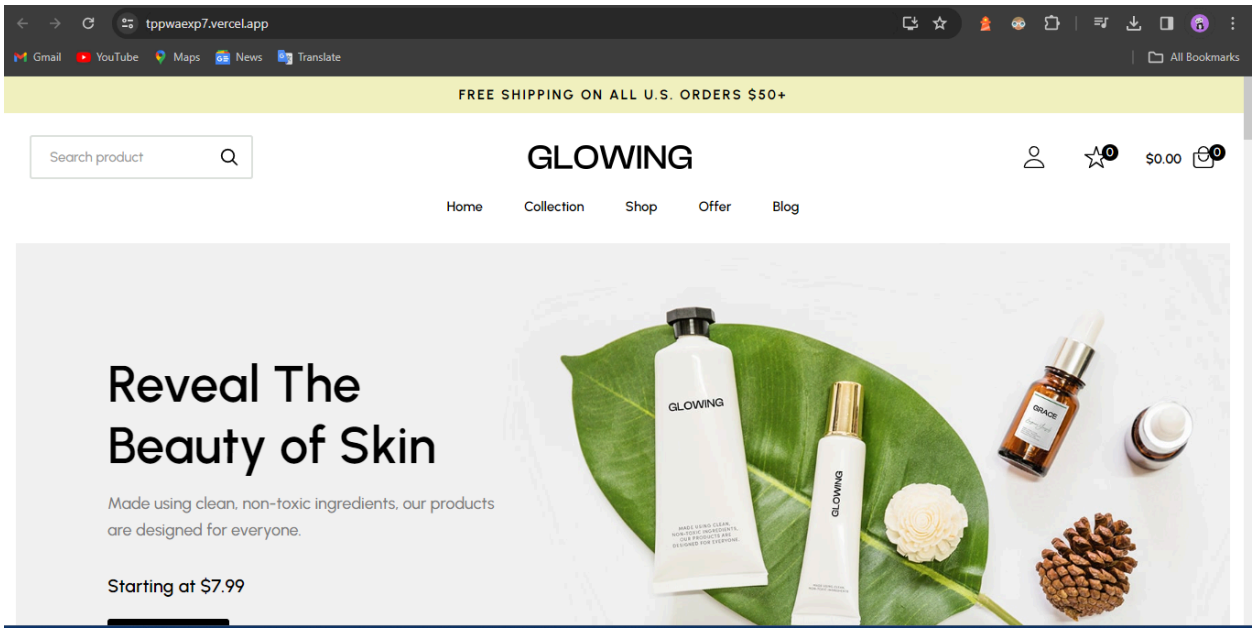
Code -
Service worker.js

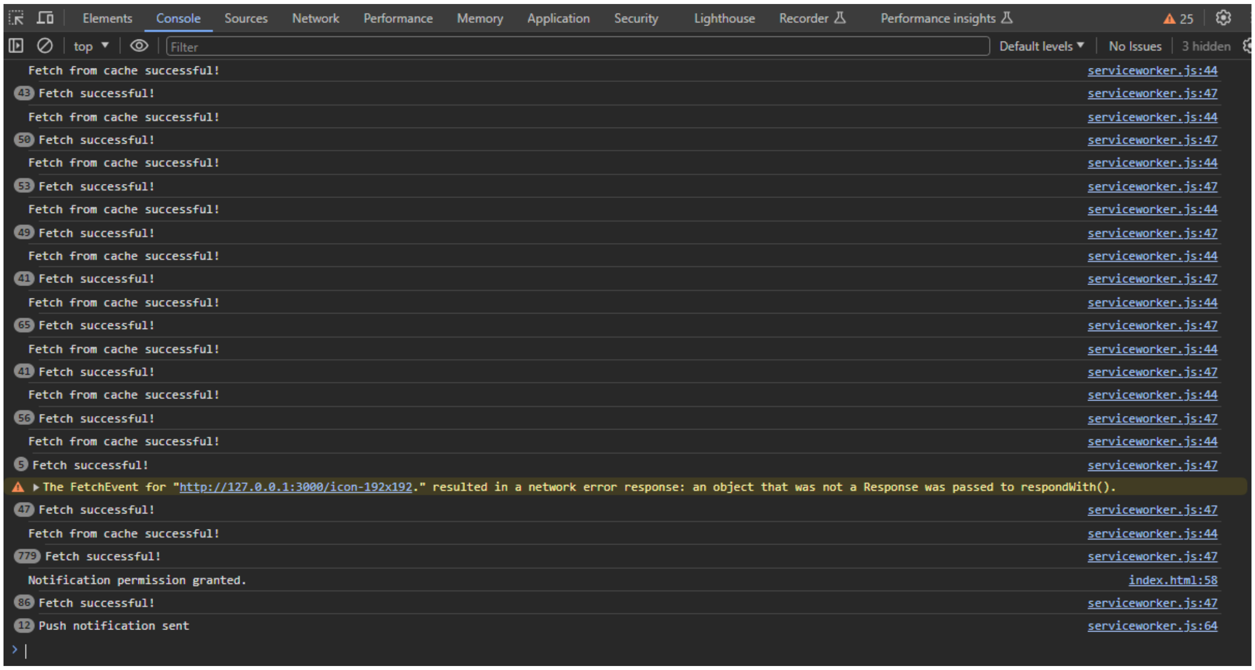
```
const CACHE_NAME = 'my-ecommerce-app-cache-v1';
const urlsToCache = [
  '/',
  'index.html',
  'assets/js/script.js',
  'assets/css/style.css',
  'serviceworker.js',
  'manifest.json',
  'offline.html'
  // Add more files to cache as needed
];

self.addEventListener('install', function(event) {
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then(function(cache) {
        console.log('Opened cache');
        return cache.addAll(urlsToCache)
          .catch(function(error) {
            console.error('Cache.addAll error:', error);
          });
      })
  );
});

self.addEventListener('activate', function(event) {
  // Perform activation steps
  event.waitUntil(
    caches.keys().then(function(cacheNames) {
      return Promise.all(
        cacheNames.map(function(cacheName) {
          if (cacheName !== CACHE_NAME) {
            return caches.delete(cacheName);
          }
        })
      );
    })
  );
});
```

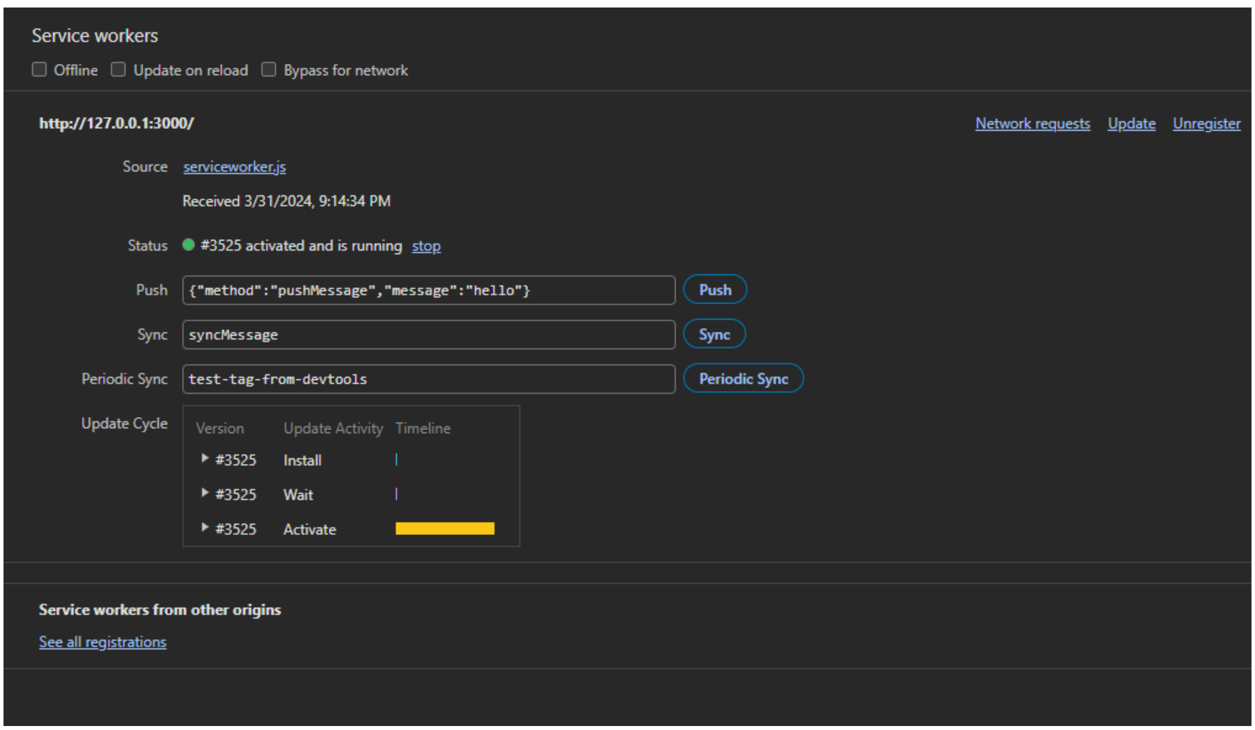
```
// Fetch event listener
self.addEventListener("fetch", function (event) {
  event.respondWith(checkResponse(event.request).catch(function () {
    console.log("Fetch from cache successful!");
    return returnFromCache(event.request);
  }));
  console.log("Fetch successful!");
  event.waitUntil(addToCache(event.request));
});

// Sync event listener
self.addEventListener('sync', function(event) {
  if (event.tag === 'syncMessage') {
    console.log("Sync successful!");
  }
});

// Push event listener
self.addEventListener("push", function (event) {
  if (event && event.data) {
    try {
      var data = event.data.json();
      if (data && data.method === "pushMessage") {
        console.log("Push notification sent");
        self.registration.showNotification("Ecommerce website", { body: data.message });
      }
    } catch (error) {
      console.error("Error parsing push data:", error);
    }
  }
});

self.addEventListener('activate', async () => {
  if (Notification.permission !== 'granted') {
    try {
      const permission = await Notification.requestPermission();
      if (permission === 'granted') {
        console.log('Notification permission granted.');
      } else {
        console.warn('Notification permission denied.');
      }
    } catch (error) {
      console.error('Failed to request notification permission:', error);
```

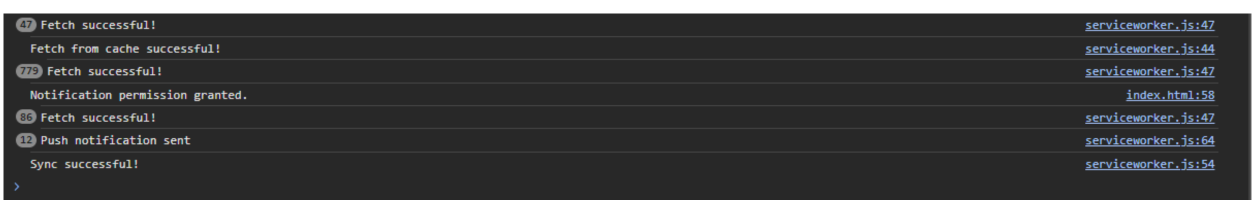
```
    }
  }
});

var checkResponse = function (request) {
  return new Promise(function (fulfill, reject) {
    fetch(request)
      .then(function (response) {
        if (response.status !== 404) {
          fulfill(response);
        } else {
          reject(new Error("Response not found"));
        }
      })
      .catch(function (error) {
        reject(error);
      });
  });
};

var returnFromCache = function (request) {
  return caches.open("offline").then(function (cache) {
    return cache.match(request).then(function (matching) {
      if (!matching || matching.status == 404) {
        return cache.match("offline.html");
      } else {
        return matching;
      }
    });
  });
};

var addToCache = function (request) {
  return caches.open("offline").then(function (cache) {
    return fetch(request).then(function (response) {
      return cache.put(request, response.clone()).then(function () {
        return response;
      });
    });
  });
};
```


Service workers
Offline
Update on reload
Bypass for network
https://tppwaexp7.vercel.app/
Network requests
Update
Unregist
Source
serviceworker.js
Received 3/31/2024, 9:33:53 PM
Status
#3526 activated and is running
stop
Clients
https://tppwaexp7.vercel.app/
focus
Push
{"method":"pushMessage","message":"hello"}
Push
Sync
test-tag-from-devtools
Sync
Periodic Sync
test-tag-from-devtools
Periodic Sync
Update Cycle
Version
Update Activity
Timeline
#3526
Install
#3526
Wait
#3526
Activate
Service workers from other origins
See all registrations


tppwaexp7.vercel.app
Gmail
YouTube
Maps
News
Translate
All Bookmarks
FREE SHIPPING ON ALL U.S. ORDERS $50+
Search product
GLOWING
$0.00
Home
Collection
Shop
Offer
Blog
Reveal The Beauty of Skin
Made using clean, non-toxic ingredients, our products are designed for everyone.
Starting at $7.99
GLOWING
GLOWING
GRACE







Conclusion -

In this experiment, we have successfully implemented service worker events like fetch, sync and push for E-commerce PWA and found out output for above implementation.