Experiment No. 9

Name : Kunal Pal **Div/Roll : D15B/50**
Subject : PWA LAB

Aim:- To implement Service worker events like fetch, sync and push for 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.

- ➢ Service workers make extensive use of promises, so if you're new to promises, then you should stop reading this and check out Promises, an introduction.

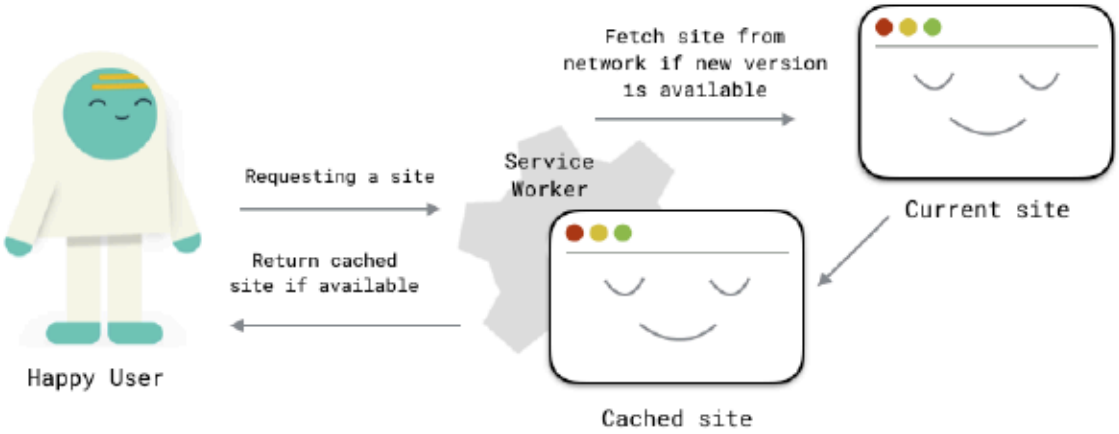
Fetch Event

You can track and manage page network traffic with this event. You can check existing cache, manage “cache first” and “network first” requests and return a response that you want.

Of course, you can use many different methods but you can find in the following example a “cache first” and “network first” approach. In this example, if the request’s and current location’s origin are the same (Static content is requested.), this is called “cacheFirst” but if you request a targeted external URL, this is called “networkFirst”.

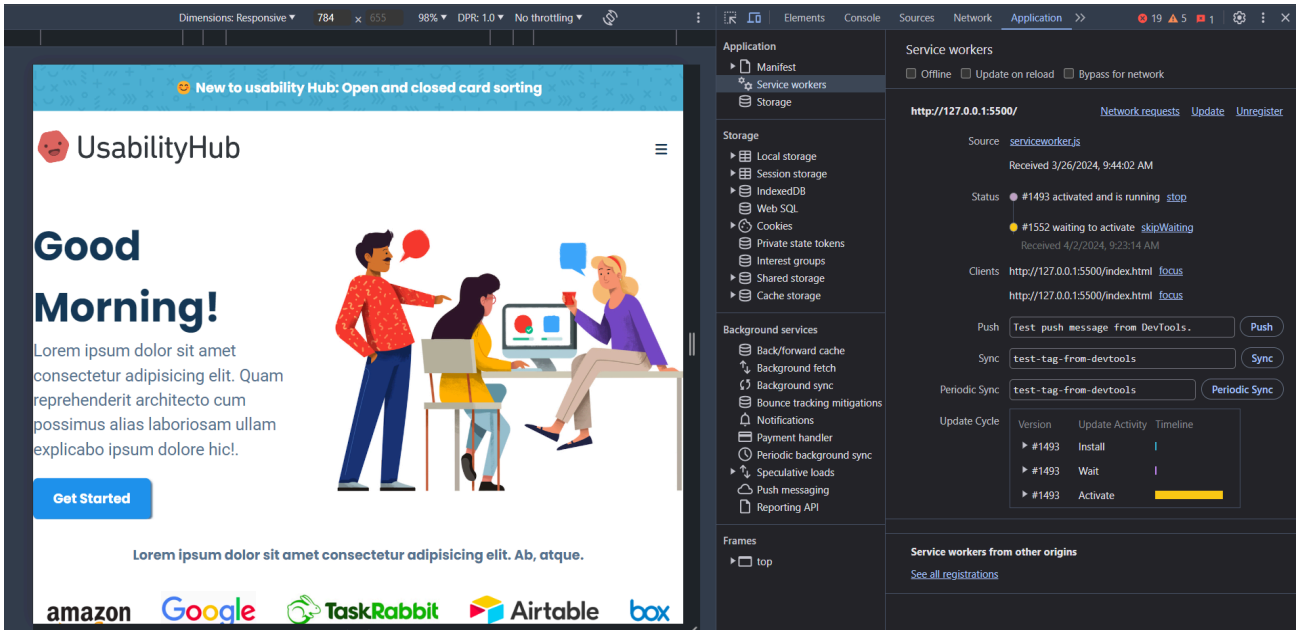
CacheFirst - In this function, if the received request has cached before, the cached response is returned to the page. But if not, a new response requested from the network.

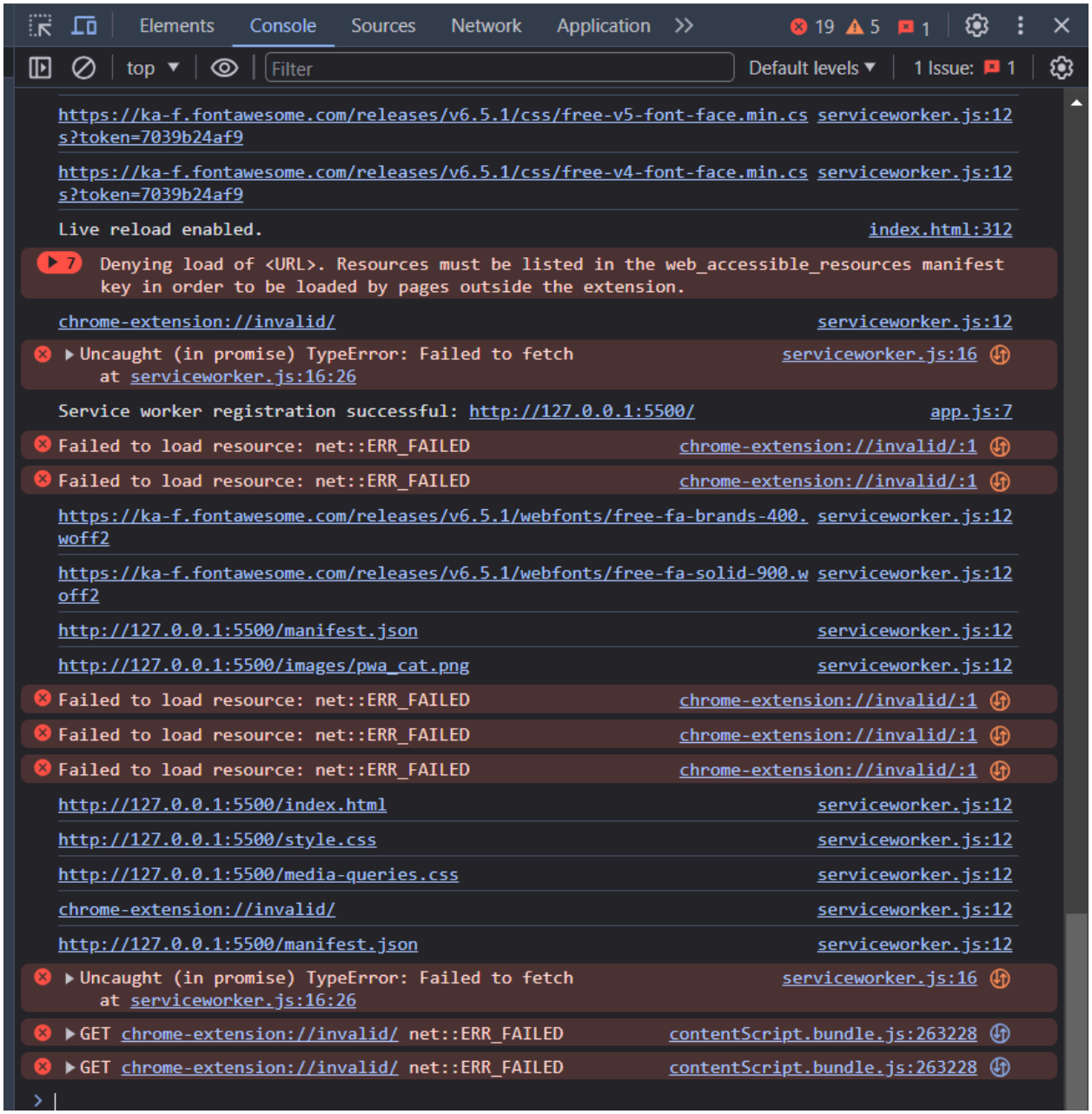
NetworkFirst - In this function, firstly we can try getting an updated response from the network, if this process completed successfully, the new response will be cached and returned. But if this process fails, we check whether the request has been cached before or not. If a cache exists, it is returned to the page, but if not, this is up to you. You can return dummy content or information messages to the page.



Code :

```
self.addEventListener('fetch', function(event) {
  onsole.log('fetch event triggered');
event.respondWith(
caches.match(event.request).then(function(response) {
return response || fetch(event.request);
})
);
});
```


Dimensions: Responsive
784
655
98%
DPR: 1.0
No throttling
New to usability Hub: Open and closed card sorting
UsabilityHub
Good Morning!
Lorem ipsum dolor sit amet consectetur adipisicing elit. Quam reprehenderit architecto cum possimus alias laboriosam ullam explicabo ipsum dolore hic!.
Get Started
Lorem ipsum dolor sit amet consectetur adipisicing elit. Ab, atque.
amazon
Google
TaskRabbit
Airtable
box
Elements
Console
Sources
Network
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 mitigations
Notifications
Payment handler
Periodic background sync
Speculative loads
Push messaging
Reporting API
Frames
top
Service workers
Offline
Update on reload
Bypass for network
http://127.0.0.1:5500/
Network requests
Update
Unregister
Source
serviceworker.js
Received 3/26/2024, 9:44:02 AM
Status
#1493 activated and is running
stop
#1552 waiting to activate
skipWaiting
Received 4/2/2024, 9:23:14 AM
Clients
http://127.0.0.1:5500/index.html
focus
http://127.0.0.1:5500/index.html
focus
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devtools
Sync
Periodic Sync
test-tag-from-devtools
Periodic Sync
Update Cycle
Version
Update Activity
Timeline
#1493
Install
#1493
Wait
#1493
Activate
Service workers from other origins
See all registrations



Sync Event: This event is triggered when the service worker receives a sync event from the browser.

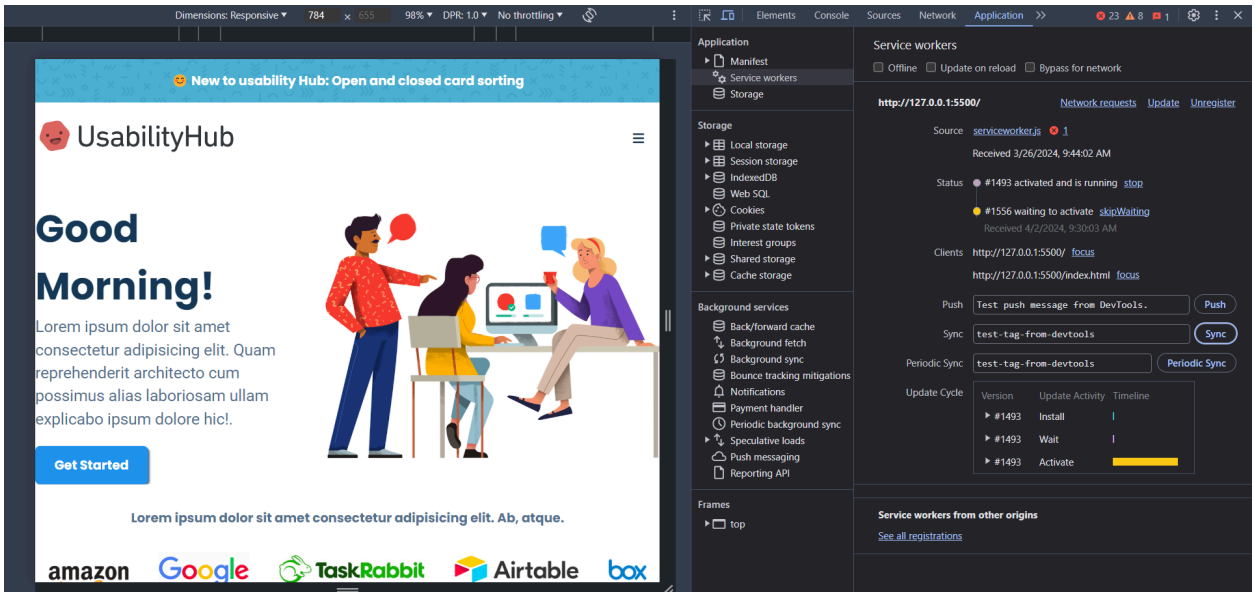
When we click the “send” button, email content will be saved to IndexedDB.Background Sync registration.
If the Internet connection is available, all email content will be read and sent to

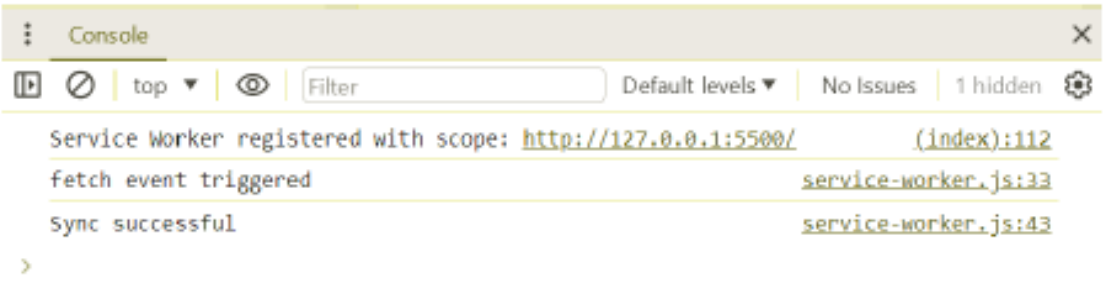
Mail Server.

If the Internet connection is unavailable, the service worker waits until the connection is available even though the window is closed. When it is available, email content will be sent to Mail Server.

Code :

```
self.addEventListener('sync', function(event) {
  console.log('Sync successful');
  if (event.tag === 'syncData') {
  event.waitUntil(syncDataWithServer());
  }
  });
```





Push Event: This event is triggered when a push notification is received. We can check in the following example.
“Notification.requestPermission();” is the necessary line to show notification to the user. If you don’t want to show any notification, you don’t need this line.

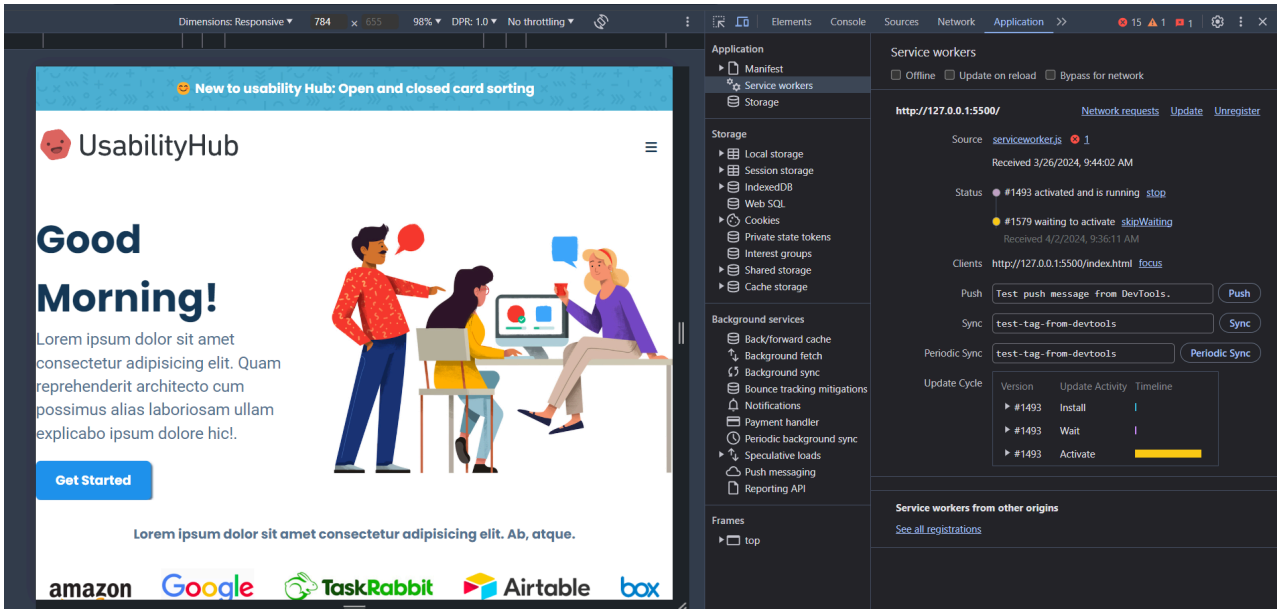
In the following code block is in sw.js file. You can handle push notifications with this event. In this example, I kept it simple. We send an object that has “method” and “message” properties. If the method value is “pushMessage”, we open the information notification with the “message” property.

index.js

```
Notification.requestPermission().then(function(permission) {
    if (permission === 'granted') {
    // Permission granted, you can now show notifications
    }
    });
```

servicework.js

```
self.addEventListener('push', function(event) {
  const options = {
  body: 'Your wishlisted HP laptop was restocked!',
  icon: 'pwa_img.png'
  };
  event.waitUntil(
  self.registration.showNotification('New notification', options)
  );
  })
```


Dimensions: Responsive
784
655
98%
DPR: 1.0
No throttling
Elements
Console
Sources
Network
Application
New to usability Hub: Open and closed card sorting
UsabilityHub
Good Morning!
Lorem ipsum dolor sit amet consectetur adipisicing elit. Quam reprehenderit architecto cum possimus alias laboriosam ullam explicabo ipsum dolore hic!.
Get Started
Lorem ipsum dolor sit amet consectetur adipisicing elit. Ab, atque.
amazon
Google
TaskRabbit
Airtable
box
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 mitigations
Notifications
Payment handler
Periodic background sync
Speculative loads
Push messaging
Reporting API
Frames
top
Service workers
Offline
Update on reload
Bypass for network
http://127.0.0.1:5500/
Network requests
Update
Unregister
Source
serviceworker.js
Received 3/26/2024, 9:44:02 AM
Status
#1493 activated and is running
stop
#1579 waiting to activate
skipWaiting
Received 4/2/2024, 9:36:11 AM
Clients
http://127.0.0.1:5500/index.html
focus
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devtools
Sync
Periodic Sync
test-tag-from-devtools
Periodic Sync
Update Cycle
Version
Update Activity
Timeline
#1493
Install
#1493
Wait
#1493
Activate
Service workers from other origins
See all registrations

Conclusion : Hence we have understood and implemented the working of service worker event listeners such as ‘fetch’, ‘sync’, and ‘push’ notifications in E-commerce Progressive web application.