

Experiment No.9

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

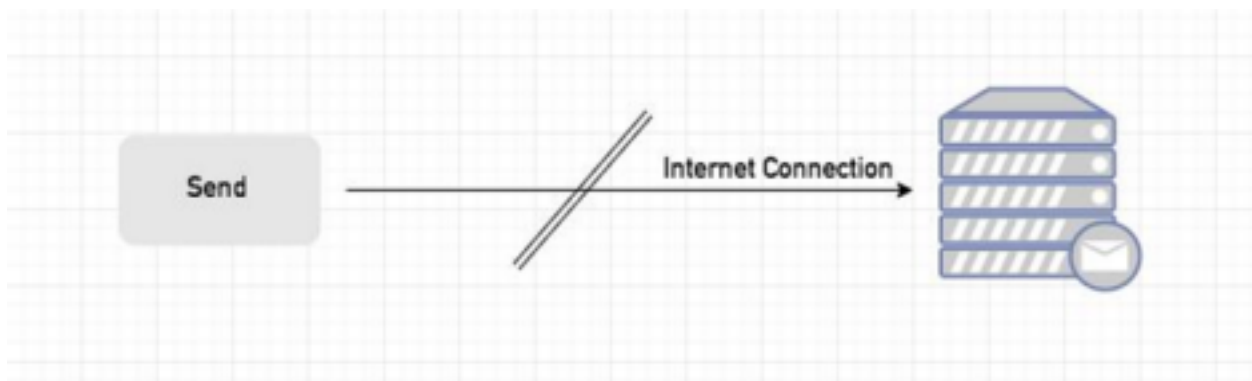
to you. You can return dummy content or information messages to the page.

Sync Event

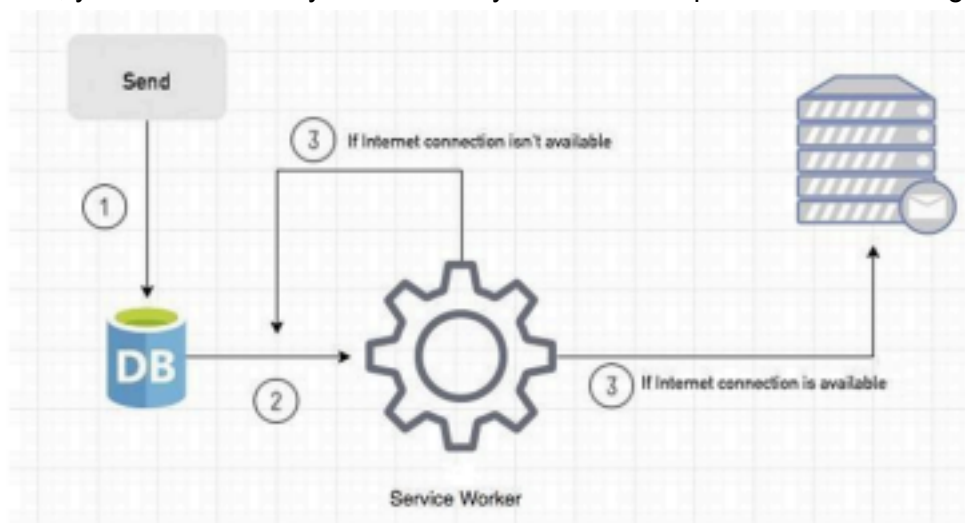
Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are writing e-mail content and we didn't realize it. When completing the writing, we click the send button.

Here is a job for the Background Sync.

The following view shows the classical process of sending email to us. If the Internet Connection is broken, we can't send any content to Mail Server.



Here, you can create any scenario for yourself. A sample is in the following for this case.



1. When we click the "send" button, email content will be saved to IndexedDB.
2. Background Sync registration.
3. **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.

You can see the working process within the following code block.

Push Event

This is the event that handles push notifications that are received from the server. You can apply any method with received data.

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.

Code:

service-worker.js

```
60 self.addEventListener("fetch", function (event) {
61   event.respondWith(checkResponse(event.request).catch(function () {
62     console.log("Fetch from cache successful!");
63     return returnFromCache(event.request);
64   }));
65   console.log("Fetch successful!");
66   event.waitUntil(addToCache(event.request));
67 });
68 self.addEventListener('sync', event => { if (event.tag === 'syncMessage') {
69   console.log("Sync successful!")
70 }
71 });
72 self.addEventListener('push', function(event) {
73   if (event && event.data) {
74     var data = event.data.json();
75     if (data.method == "pushMessage") {
76       console.log("Push notification sent");
77       event.waitUntil(
78         self.registration.showNotification("Anushka", {
79           body: data.message
80         })
81       );
82     }
83   }
84 });
85
```

```
JS script.js M    index.html M    JS service-worker.js U X
pwa > JS service-worker.js > self.addEventListener('push') callback
86   var filesToCache = [ './index.html'
87   ];
88   var preload = function () {
89     return caches.open("offline").then(function (cache) {
90       // caching index and important routes
91       return cache.addAll(filesToCache);
92     });
93   };
94   var checkResponse = function (request) { return new Promise(function (fulfill, reject) {
95     fetch(request).then(function (response) { if (response.status !== 404) {
96       fulfill(response);
97     } else {
98       reject();
99     }
100   }, reject);
101   });
102   };
103   var addToCache = function (request) {
104     return caches.open("offline").then(function (cache) { return fetch(request).then(function
105     (response) {
106       return cache.put(request, response);
107     });
108   });
109   };
110   var returnFromCache = function (request) {
111     return caches.open("offline").then(function (cache) { return
112     cache.match(request).then(function (matching) {
113       if (!matching || matching.status == 404) { return cache.match("offline.html");
114     } else {
115       return matching;
116     }
117   });
118   });
119   };
120
```

Output:

Fetch -

Discover Your Perfect Style - Guaranteed!

At Fashionify, we believe your style should be as unique as you are. If you don't love your new clothing, we'll help you find something that suits you better. That's our style guarantee.

[Find out more](#)

Best Sellers



Sync-
Push notification

#	Name	Response-T...	Content-Type	Content-Le...	Time Cached	Vary Header
0	/assets/css/style.css	basic	text/css	17,184	3/25/2024, ...	Origin
1	/assets/css/style.css?_cacheOverride=1711375038724	basic	text/css	17,162	3/25/2024, ...	Origin
2	/assets/css/style.css?_cacheOverride=1711375040433	basic	text/css	17,162	3/25/2024, ...	Origin
3	/assets/css/style.css?_cacheOverride=1711375047200	basic	text/css	16,607	3/25/2024, ...	Origin
4	/assets/css/style.css?_cacheOverride=1711375049037	basic	text/css	17,083	3/25/2024, ...	Origin
5	/assets/css/style.css?_cacheOverride=1711375051953	basic	text/css	17,162	3/25/2024, ...	Origin
6	/assets/css/style.css?_cacheOverride=1711375066257	basic	text/css	17,052	3/25/2024, ...	Origin
7	/assets/css/style.css?_cacheOverride=1711375069510	basic	text/css	17,050	3/25/2024, ...	Origin
8	/assets/css/style.css?_cacheOverride=1711375101077	basic	text/css	16,288	3/25/2024, ...	Origin
9	/assets/css/style.css?_cacheOverride=1711375112271	basic	text/css	17,050	3/25/2024, ...	Origin
10	/assets/css/style.css?_cacheOverride=1711375119103	basic	text/css	17,162	3/25/2024, ...	Origin

Headers Preview

1 No network manager for request

Total entries: 114

Console Issues +

top Filter Default levels 34 1 hidden

Fetch successful!

[Intervention] Images loaded lazily and replaced with placeholders. Load events are deferred. See <https://go.microsoft.com/fwlink/?linkid=2048113>

7 Fetch successful!

Notification permission granted

Sync successful!

Fetch successful!

Service Worker registered with scope: <http://127.0.0.1:5500/pwa/>

4 Fetch successful!

Conclusion: We have understood and successfully implemented events like fetch , push and sync for our ecommerce Kitter pwa.