Amit Nayak 46/D15B

Experiment No. 10

Aim : To study and implement deployment of Ecommerce PWA to GitHub Pages.

Theory :
GitHub Pages
Public web pages are freely hosted and easily published. Public webpages hosted directly from your GitHub repository. Just edit, push, and your changes are live.
GitHub Pages provides the following key features:
1. Blogging with Jekyll
2. Custom URL
3. Automatic Page Generator

Reasons for favoring this over Firebase:
1. Free to use
2. Right out of github
3. Quick to set up

GitHub Pages is used by Lyft, CircleCI, and HubSpot.
GitHub Pages is listed in 775 company stacks and 4401 developer stacks.

Pros
1. Very familiar interface if you are already using GitHub for your projects.
2. Easy to set up. Just push your static website to the gh-pages branch and your website is ready.
3. Supports Jekyll out of the box.
4. Supports custom domains. Just add a file called CNAME to the root of your site, add A record in the site's DNS configuration, and you are done.

Cons
1. The code of your website will be public, unless you pay for a private repository.
2. Currently, there is no support for HTTPS for custom domains. It's probably coming soon
though.
3. Although Jekyll is supported, plug-in support is rather spotty

Firebase
The Realtime App Platform. Firebase is a cloud service designed to power real-time, collaborative applications. Simply add the Firebase library to your application to gain access to a shared data structure; any changes you make to that data are automatically synchronized with the Firebase cloud and with other clients within milliseconds.

Some of the features offered by Firebase are:
1. Add the Firebase library to your app and get access to a shared data structure. Any changes made to that data are automatically synchronized with the Firebase cloud and with other clients within milliseconds.
2. Firebase apps can be written entirely with client-side code, update in real-time out-of-the-box, interoperate well with existing services, scale automatically, and provide
strong data security.
3. Data Accessibility- Data is stored as JSON in Firebase. Every piece of data has its own URL which can be used in Firebase's client libraries and as a REST endpoint. These URLs can also be entered into a browser to view the data and watch it update in real-time.

Reasons for favoring over GitHub Pages:
1. Realtime backend made easy
2. Fast and responsive
Instacart, 9GAG, and Twitch are some of the popular companies that use Firebase
Firebase has a broader approval, being mentioned in 1215 company stacks & 4651 developers stacks

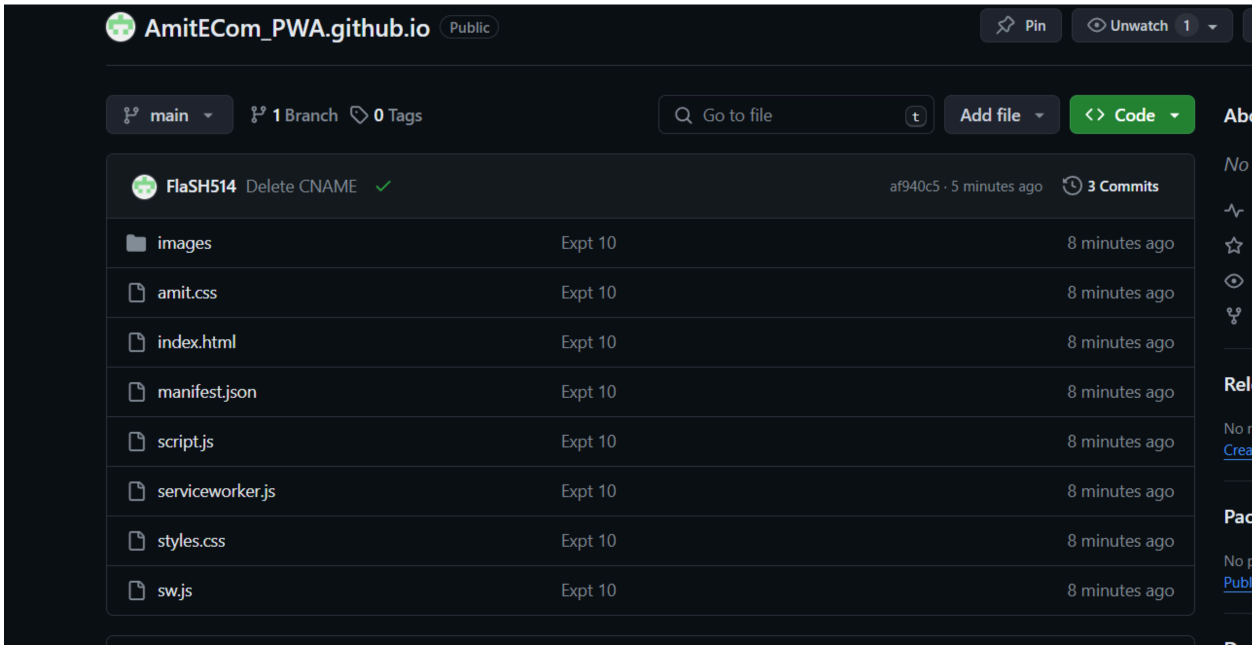
Pros
1. Hosted by Google. Enough said.
2. Authentication, Cloud Messaging, and a whole lot of other handy services will be available to you.
3. A real-time database will be available to you, which can store 1 GB of data.
4. You’ll also have access to a blob store, which can store another 1 GB of data.
5. Support for HTTPS. A free certificate will be provisioned for your custom domain within 24 hours.

Cons
1. Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if

you use a CDN or AMP.
2. Command-line interface only.
3. No in-built support for any static site generator

Link to Github repo : https://github.com/FlaSH514/AmitECom_PWA.github.io
Hosted website Link : https://flash514.github.io/AmitECom_PWA.github.io/

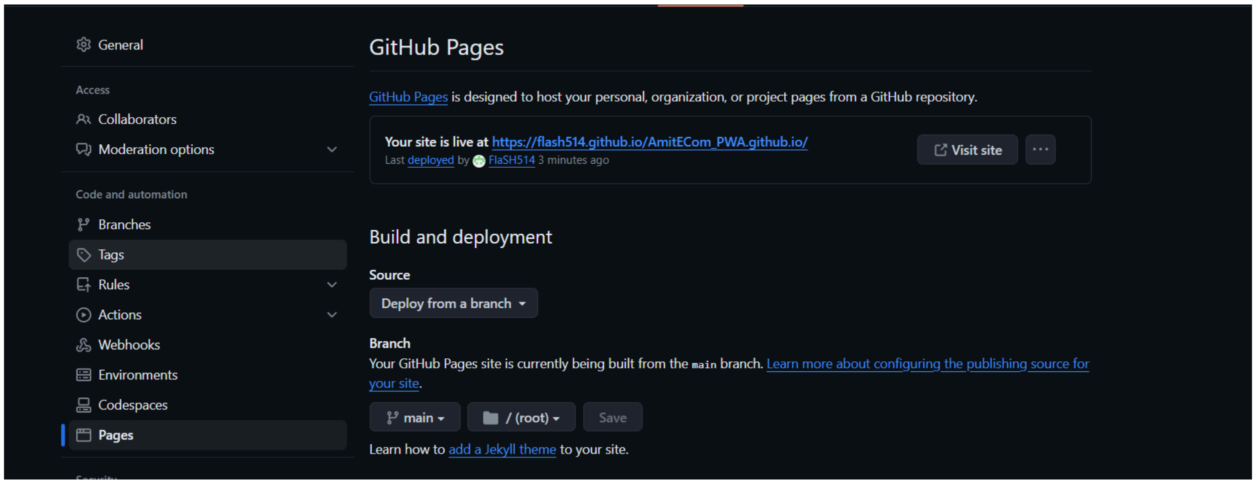
GitHub Project:


AmitECom_PWA.github.io Public

main · 1 Branch · 0 Tags

FlaSH514 Delete CNAME ✓ af940c5 · 5 minutes ago 3 Commits

images	Expt 10	8 minutes ago
amit.css	Expt 10	8 minutes ago
index.html	Expt 10	8 minutes ago
manifest.json	Expt 10	8 minutes ago
script.js	Expt 10	8 minutes ago
serviceworker.js	Expt 10	8 minutes ago
styles.css	Expt 10	8 minutes ago
sw.js	Expt 10	8 minutes ago


GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Your site is live at https://flash514.github.io/AmitECom_PWA.github.io/
Last deployed by FlaSH514 3 minutes ago

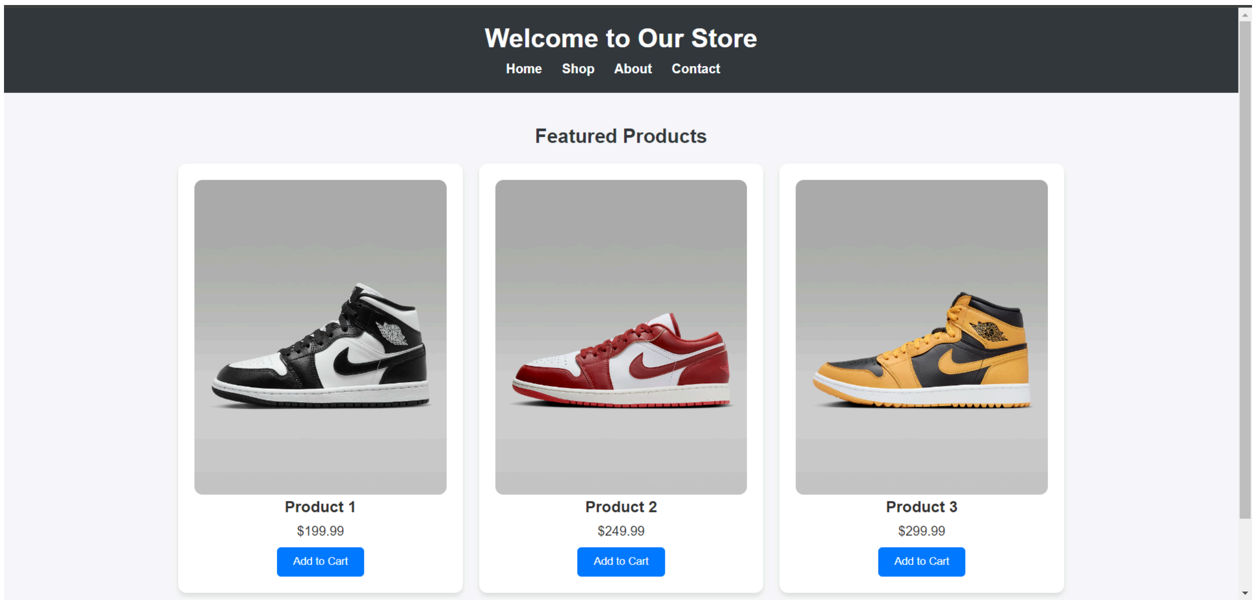
Build and deployment

Source: Deploy from a branch

Branch
Your GitHub Pages site is currently being built from the `main` branch. Learn more about configuring the publishing source for your site.

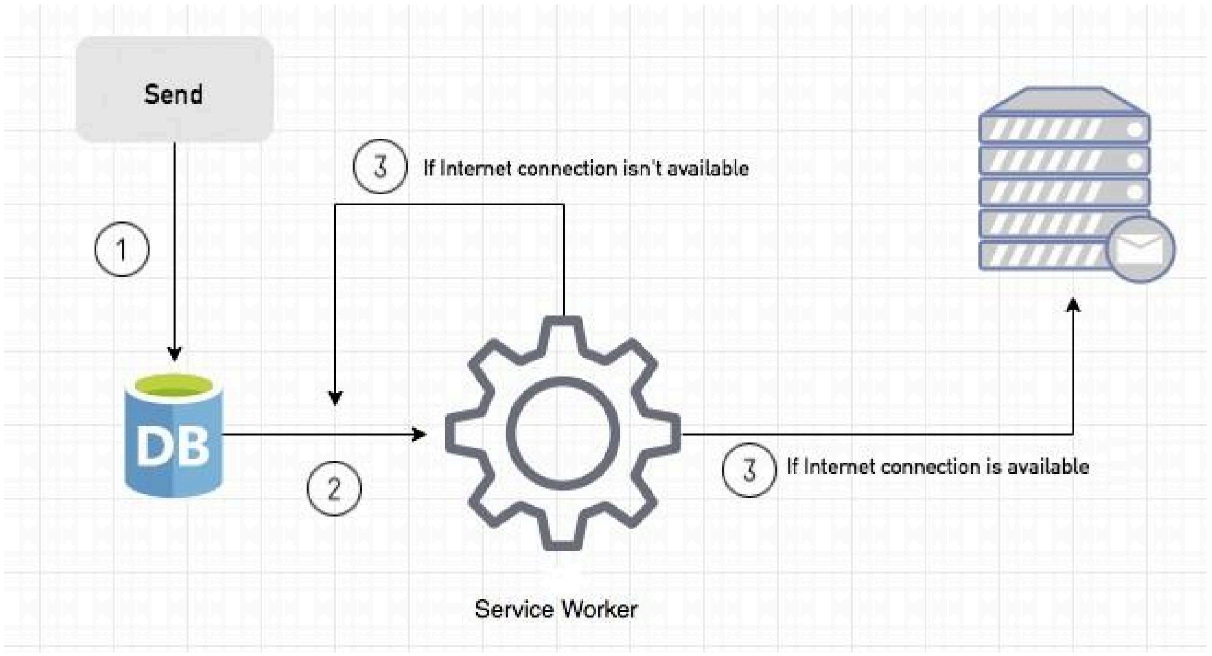
main / (root) Save

Learn how to add a Jekyll theme to your site.



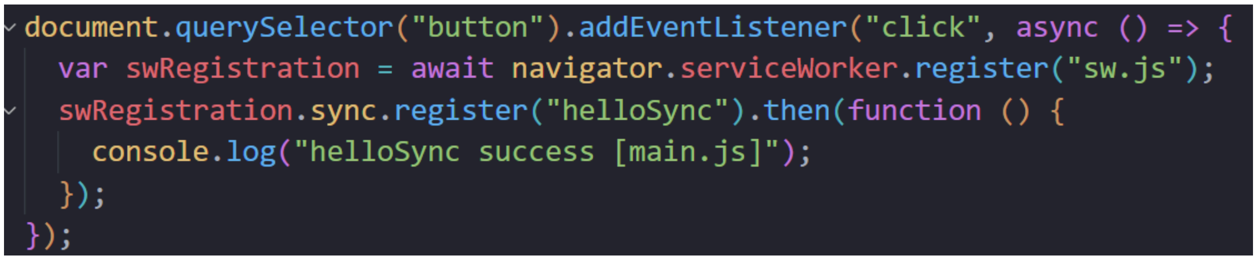
Conclusion : We have deployed our Ecommerce Pwa via GitHub pages and understood the working of Github pages.

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


Send
1
3 If Internet connection isn't available
DB
2
Service Worker
3 If Internet connection is available

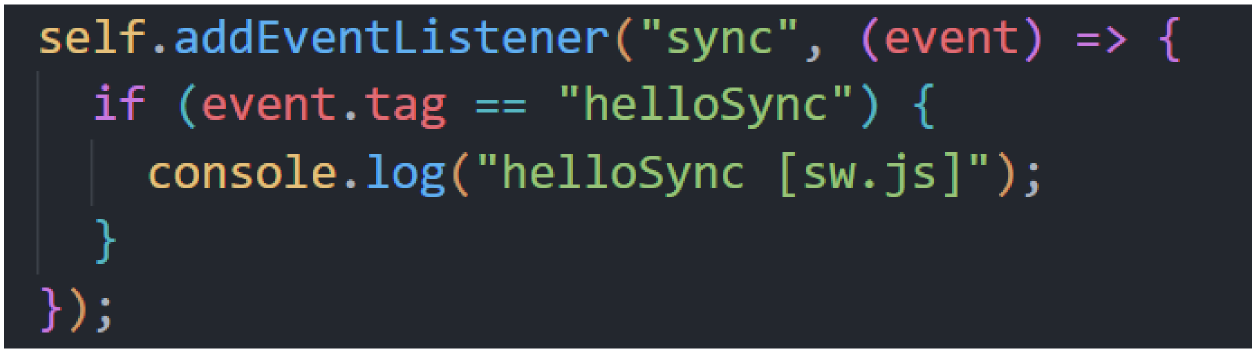
- When we click the “send” button, email content will be saved to IndexedDB.
- Background Sync registration.
- If the Internet connection is available, all email content will be read and sent to Mail Server.
- If the Internet connection is unavailable, the service worker waits until the connection is available even though the window is closed. When it is available, email content will be sent to Mail Server.
- You can see the working process within the following code block.

Event Listener for Background Sync Registration:



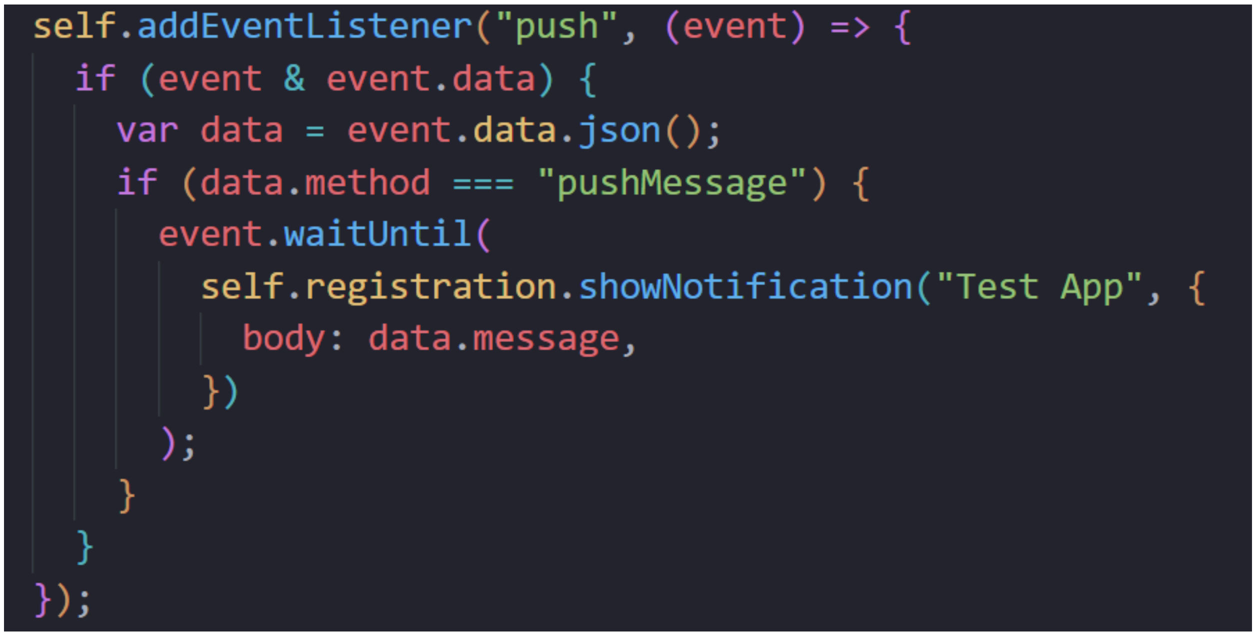
```
document.querySelector("button").addEventListener("click", async () => {
  var swRegistration = await navigator.serviceWorker.register("sw.js");
  swRegistration.sync.register("helloSync").then(function () {
    console.log("helloSync success [main.js]");
  });
});
```

Event Listener for sw.js



```
self.addEventListener("sync", (event) => {
  if (event.tag == "helloSync") {
    console.log("helloSync [sw.js]");
  }
});
```

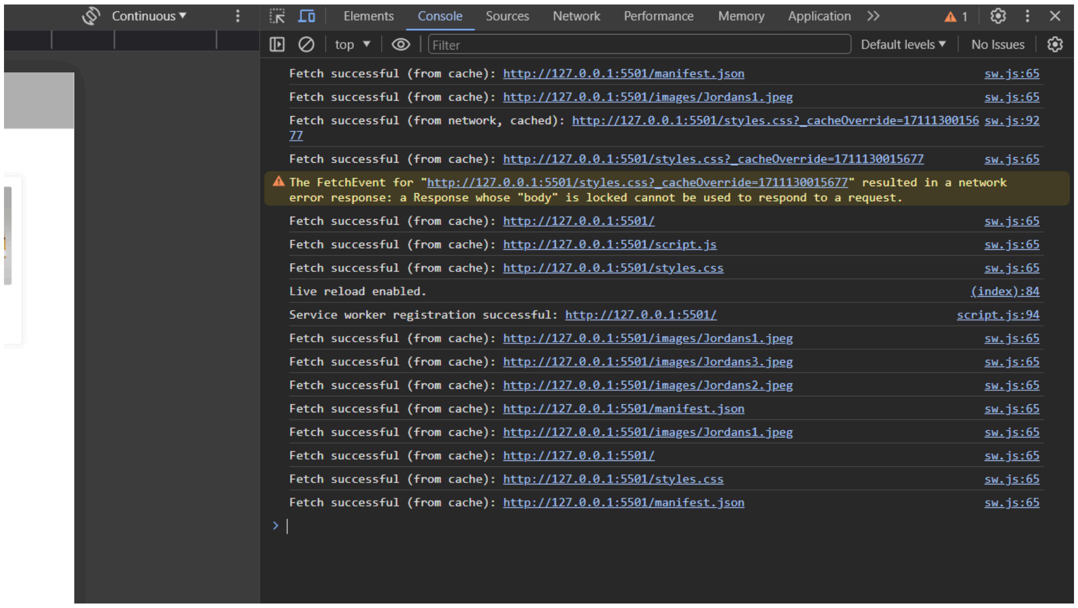
Push Event
The Push Event is triggered when a push notification is received from the server. It allows developers to handle the incoming data and perform actions based on the received notification.
For example, in a service worker (sw.js), you can use the Push Event to display a notification to the user. First, you would request permission to show notifications using Notification.requestPermission(). Then, when a push notification is received, you can extract the message from the notification and display it to the user using the showNotification() method.



```
self.addEventListener("push", (event) => {
  if (event & event.data) {
    var data = event.data.json();
    if (data.method === "pushMessage") {
      event.waitUntil(
        self.registration.showNotification("Test App", {
          body: data.message,
        })
      );
    }
  }
});
```

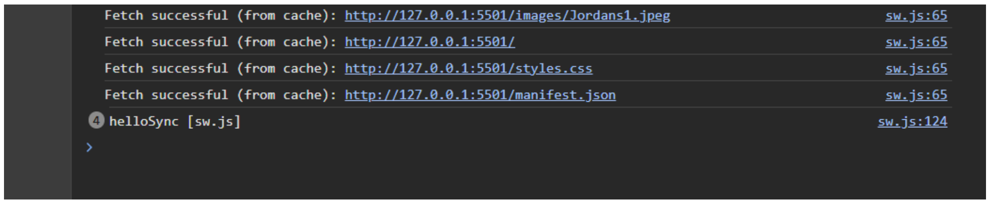
sw.js:
```
self.addEventListener("install", function (event) { event.waitUntil(preLoad());
});

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));
});

self.addEventListener('sync', event => { if (event.tag === 'syncMessage') {

console.log("Sync successful!")
}
});

self.addEventListener('push', function   (event)   { if (event && event.data) {
var data = event.data.json();
if (data.method == "pushMessage") { console.log("Push notification sent");
event.waitUntil(self.registration.showNotification("Omkar Sweets Corner", { body: data.message
}))
}
}
})



var filesToCache = [ '/',
'/menu', '/contactUs', '/offline.html',
];

var preLoad = function () {
return   caches.open("offline").then(function   (cache)   {
// caching index and important routes
return cache.addAll(filesToCache);
});
};

var checkResponse =   function   (request)   { return new Promise(function (fulfill, reject) {
fetch(request).then(function    (response)    { if (response.status !== 404) {
fulfill(response);
} else {
reject();
}
}, reject);
});
};

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

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;
}
});
});
};
```

Output:
FetchEvent



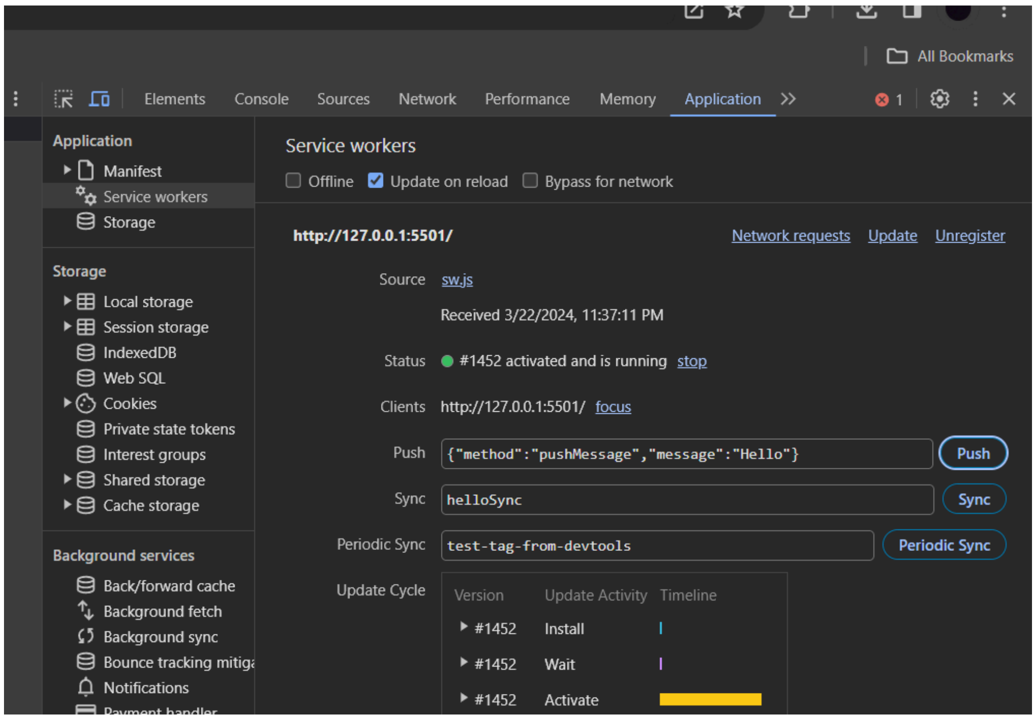
```
Fetch successful (from cache): http://127.0.0.1:5501/manifest.json                                    sw.js:65
Fetch successful (from cache): http://127.0.0.1:5501/images/Jordans1.jpeg                             sw.js:65
Fetch successful (from network, cached): http://127.0.0.1:5501/styles.css?_cacheOverride=17111300156 sw.js:92
77
Fetch successful (from cache): http://127.0.0.1:5501/styles.css?_cacheOverride=1711130015677          sw.js:65
⚠ The FetchEvent for "http://127.0.0.1:5501/styles.css?_cacheOverride=1711130015677" resulted in a network
error response: a Response whose "body" is locked cannot be used to respond to a request.
Fetch successful (from cache): http://127.0.0.1:5501/                                                 sw.js:65
Fetch successful (from cache): http://127.0.0.1:5501/script.js                                        sw.js:65
Fetch successful (from cache): http://127.0.0.1:5501/styles.css                                       sw.js:65
Live reload enabled.                                                                               (index):84
Service worker registration successful: http://127.0.0.1:5501/                                    script.js:94
Fetch successful (from cache): http://127.0.0.1:5501/images/Jordans1.jpeg                             sw.js:65
Fetch successful (from cache): http://127.0.0.1:5501/images/Jordans3.jpeg                             sw.js:65
Fetch successful (from cache): http://127.0.0.1:5501/images/Jordans2.jpeg                             sw.js:65
Fetch successful (from cache): http://127.0.0.1:5501/manifest.json                                    sw.js:65
Fetch successful (from cache): http://127.0.0.1:5501/images/Jordans1.jpeg                             sw.js:65
Fetch successful (from cache): http://127.0.0.1:5501/                                                 sw.js:65
Fetch successful (from cache): http://127.0.0.1:5501/styles.css                                       sw.js:65
Fetch successful (from cache): http://127.0.0.1:5501/manifest.json                                    sw.js:65
>
```

Sync event



```
Fetch successful (from cache): http://127.0.0.1:5501/images/Jordans1.jpeg        sw.js:65
Fetch successful (from cache): http://127.0.0.1:5501/                            sw.js:65
Fetch successful (from cache): http://127.0.0.1:5501/styles.css                  sw.js:65
Fetch successful (from cache): http://127.0.0.1:5501/manifest.json               sw.js:65
4 helloSync [sw.js]                                                             sw.js:124
>
```

Push event



Service workers

Offline [✓ Update on reload] Bypass for network

http://127.0.0.1:5501/ Network requests Update Unregister

Source sw.js

Received 3/22/2024, 11:37:11 PM

Status ● #1452 activated and is running stop

Clients http://127.0.0.1:5501/ focus

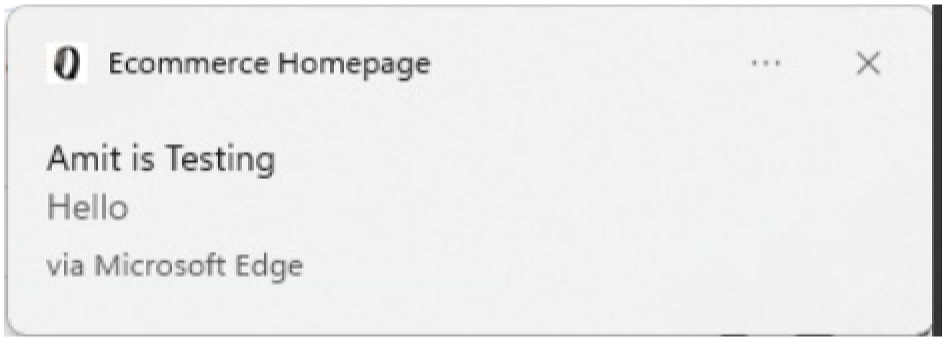
Push {"method":"pushMessage","message":"Hello"} Push

Sync helloSync Sync

Periodic Sync test-tag-from-devtools Periodic Sync

Update Cycle

Version	Update Activity	Timeline
#1452	Install	
#1452	Wait	
#1452	Activate	


Ecommerce Homepage

Amit is Testing

Hello

via Microsoft Edge

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

Conclusion : I have understood and successfully registered a service worker, and completed the install and activation process for a new service worker for the E-commerce PWA.