

**Name:**Dhairiyash Jain

**Class:** D15B

**Roll No:** 22

## **EXPERIMENT No. 9**

# **Implementing Service Worker Events (Fetch, Sync, Push) for E-Commerce PWA**

Progressive Web Apps (PWAs) are web applications that offer app-like experiences through modern web capabilities. One of the key components of a PWA is the service worker, which enables features like offline access, background sync, and push notifications. In this document, we will explore how to implement service worker events such as fetch, sync, and push in the context of an e-commerce application. Below is a sample implementation.

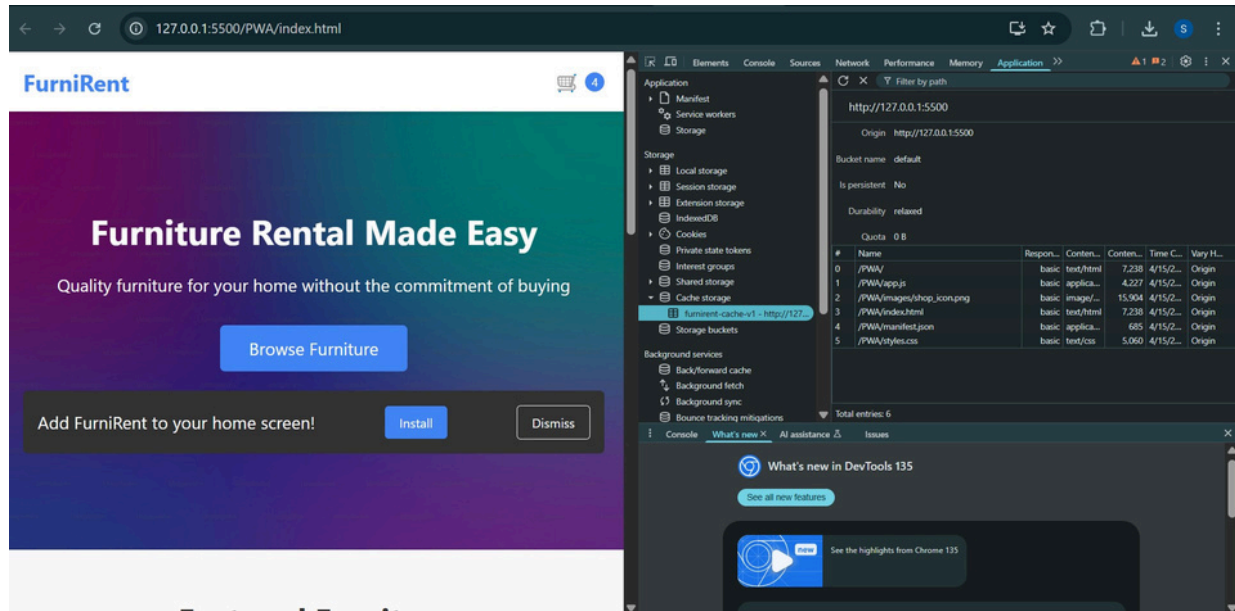
---

## **1. Caching Static Assets Using Install Event**

The install event is triggered when the service worker is installed. During this phase, essential files are cached to enable offline access.

```
const CACHE_NAME = "campquest-v1";
const ASSETS_TO_CACHE = [
  "/",
  "/index.html",
  "/src/main.jsx",
  "/CampQuest.svg",
  "/manifest.json",
];

self.addEventListener("install", (event) => {
  event.waitUntil(
    caches.open(CACHE_NAME).then((cache) => {
      return cache.addAll(ASSETS_TO_CACHE);
    })
  );
});
```



## 2. Handling Fetch Requests

The fetch event intercepts network requests. We use this event to implement a cache-first or network-first strategy depending on the URL path.

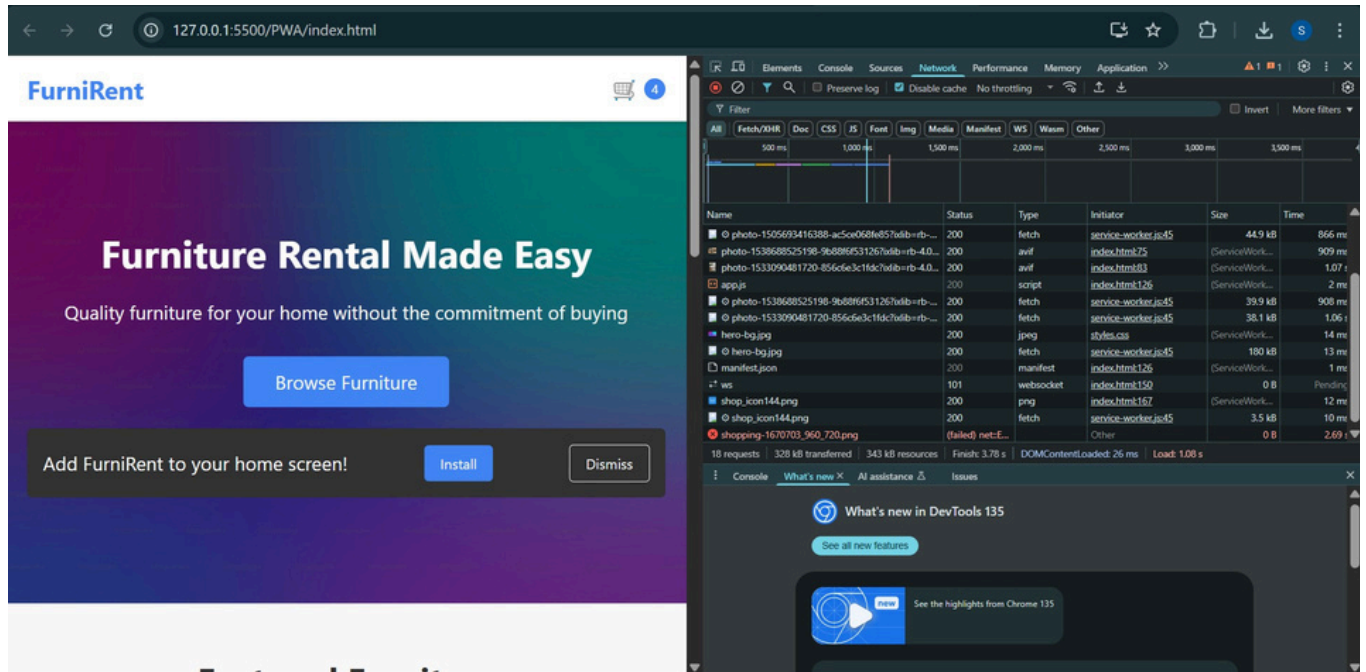
```
self.addEventListener("fetch", (event) => {
  const url = new URL(event.request.url);
  if (url.pathname.startsWith("/campgrounds")) {

    event.respondWith(
      fetch(event.request)
        .then((response) => {
          const responseClone = response.clone();
          caches.open(CACHE_NAME).then((cache) => {
            cache.put(event.request, responseClone);
          });
          return response;
        })
        .catch(() => {
          return caches.match(event.request);
        })
    );
  }
});
```

```

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

```



### 3. Background Sync (Conceptual Example)

The sync event is used to defer actions until the user has stable connectivity. For an e-commerce app, you could use this to sync cart data or orders.

```

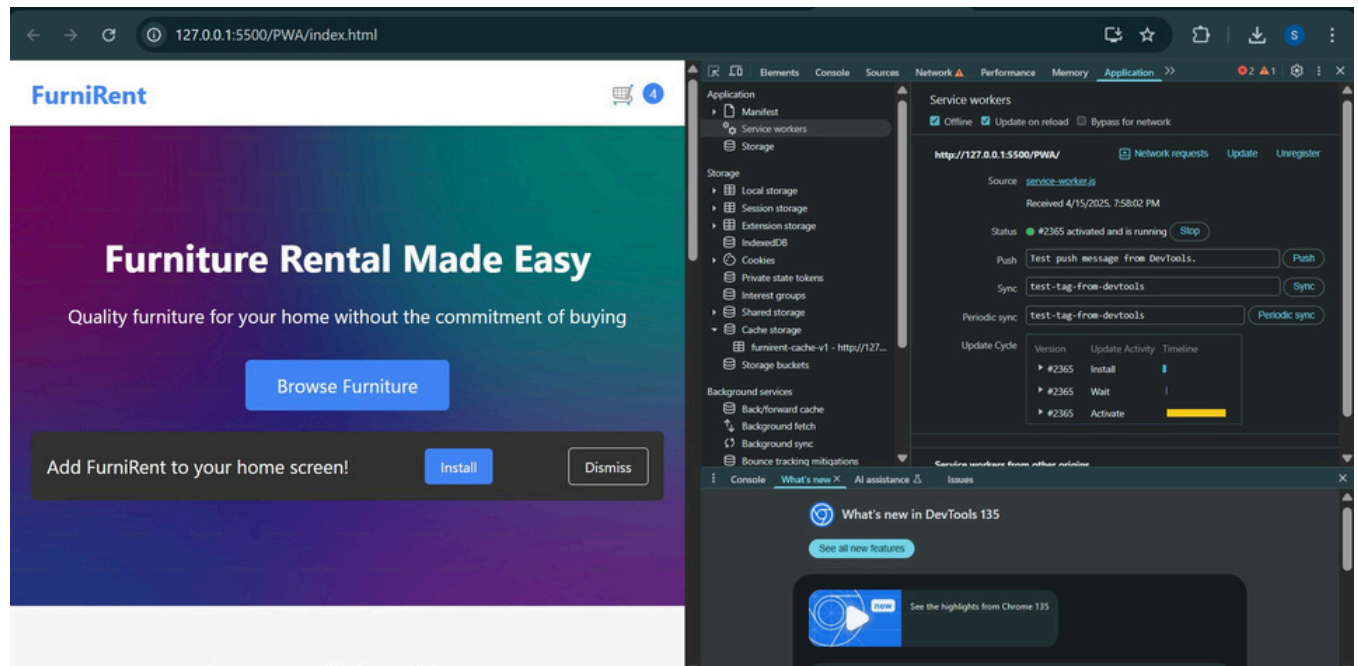
self.addEventListener("sync", (event) => {
  if (event.tag === "sync-cart") {
    event.waitUntil(
      // Logic to sync cart data with server
    );
  }
});

```

## 4. Push Notifications (Conceptual Example)

The push event is triggered when a push message is received. This could be used to notify users of new deals or order status updates.

```
self.addEventListener("push", (event) => {  
  
  const data = event.data.json();  
  const options = {  
    body: data.body,  
    icon: "/icon.png",  
  };  
  event.waitUntil(  
    self.registration.showNotification(data.title, options)  
  );  
});
```



---

## Conclusion

Service workers are powerful tools in building resilient and engaging e-commerce PWAs. By handling install, fetch, sync, and push events effectively, you can create a seamless experience for users, even in offline or low-connectivity scenarios.