EXPERIMENT 08

NAME :Mahek Taneja
CLASS : D15B
ROLL_NO : 58

Aim: To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA.

Theory:
A **Service Worker** is a core component of a Progressive Web App (PWA). It runs separately from the web page, intercepting network requests and enabling features like offline caching, background sync, and push notifications.
To use a service worker in a web app, it must be:

1. **Registered** from the main JavaScript file.
2. **Installed** through the install event.
3. **Activated** through the activate event.

1. Registering a Service Worker
This is done in the main JavaScript file (e.g., index.js or main.js) as:
javascript
CopyEdit

```
if ('serviceWorker' in navigator) {
  navigator.serviceWorker.register('/service-worker.js')
    .then(reg => console.log('Service Worker registered:', reg))
    .catch(err => console.error('Registration failed:', err));
}
```

2. Install Event
This event is triggered when the browser installs the service worker. It's typically used to cache essential static assets:
javascript
CopyEdit

```
self.addEventListener('install', event => {
  event.waitUntil(
    caches.open('static-v1').then(cache => {
      return cache.addAll([
        '/',
        '/index.html',
        '/styles.css',
        '/app.js',
        '/images/logo.png'
      ]);
    })
  );
});
```

3. Activate Event

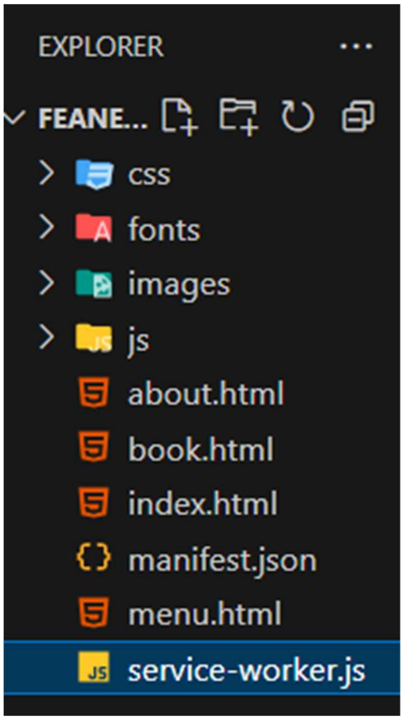
This event is used to clear old caches and take control of the client immediately:

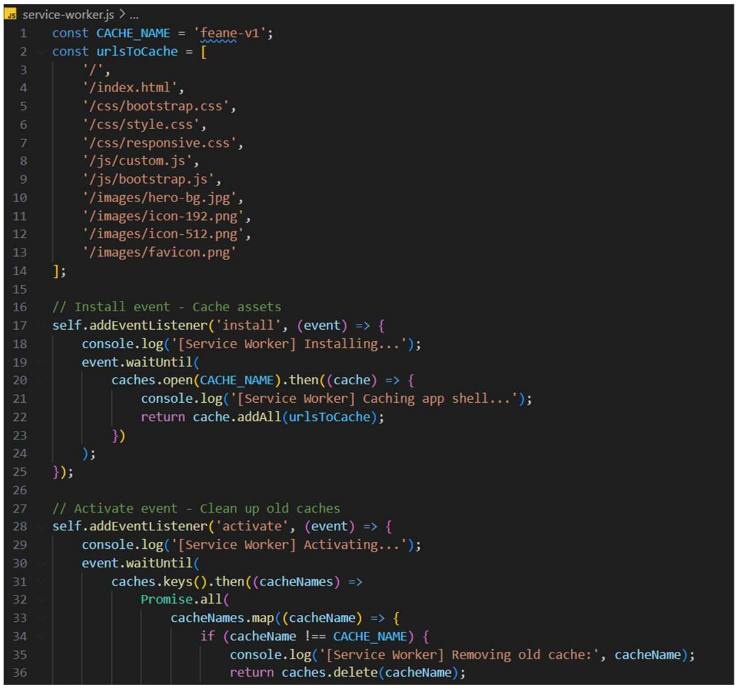
javascript

CopyEdit

```
self.addEventListener('activate', event => {
  event.waitUntil(
    caches.keys().then(keys => {
      return Promise.all(
        keys.filter(key => key !== 'static-v1')
            .map(key => caches.delete(key))
      );
    })
  );
});
```

These steps ensure that the service worker is properly set up to cache assets and manage updates effectively.



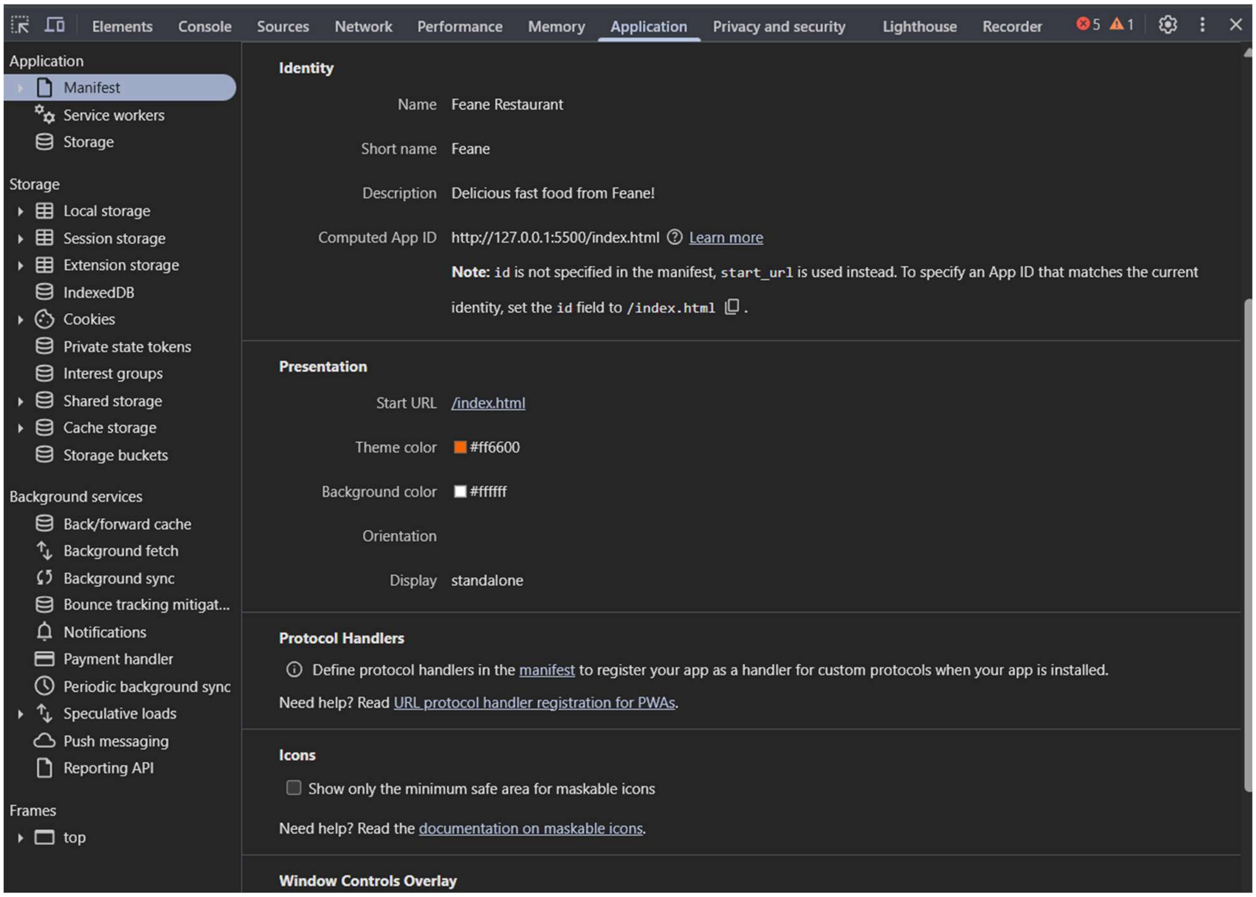


service-worker.js > ...

```javascript
const CACHE_NAME = 'feane-v1';
const urlsToCache = [
    '/',
    '/index.html',
    '/css/bootstrap.css',
    '/css/style.css',
    '/css/responsive.css',
    '/js/custom.js',
    '/js/bootstrap.js',
    '/images/hero-bg.jpg',
    '/images/icon-192.png',
    '/images/icon-512.png',
    '/images/favicon.png'
];

// Install event - Cache assets
self.addEventListener('install', (event) => {
    console.log('[Service Worker] Installing...');
    event.waitUntil(
        caches.open(CACHE_NAME).then((cache) => {
            console.log('[Service Worker] Caching app shell...');
            return cache.addAll(urlsToCache);
        })
    );
});

// Activate event - Clean up old caches
self.addEventListener('activate', (event) => {
    console.log('[Service Worker] Activating...');
    event.waitUntil(
        caches.keys().then((cacheNames) =>
            Promise.all(
                cacheNames.map((cacheName) => {
                    if (cacheName !== CACHE_NAME) {
                        console.log('[Service Worker] Removing old cache:', cacheName);
                        return caches.delete(cacheName);
```



Identity

Name	Feane Restaurant
Short name	Feane
Description	Delicious fast food from Feane!
Computed App ID	http://127.0.0.1:5500/index.html ⓘ Learn more

Note: `id` is not specified in the manifest, `start_url` is used instead. To specify an App ID that matches the current identity, set the `id` field to `/index.html`.

Presentation

Start URL	/index.html
Theme color	🟧 #ff6600
Background color	⬜ #ffffff
Orientation	
Display	standalone

Protocol Handlers

ⓘ Define protocol handlers in the manifest to register your app as a handler for custom protocols when your app is installed.

Need help? Read URL protocol handler registration for PWAs.

Icons

□ Show only the minimum safe area for maskable icons

Need help? Read the documentation on maskable icons.

Window Controls Overlay

Conclusion:

By coding and registering a service worker and handling its install and activate events, we prepare the E-commerce PWA for offline support and future enhancements like caching and background processes. The install event allows pre-caching of essential assets, while the activate event ensures outdated caches are cleaned up. This foundational setup is crucial for building a responsive, reliable, and user-friendly PWA experience, especially for users with unstable or limited internet access.