

## Practical 8

**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:

#### 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.

What can we do with Service Workers?

- You can dominate Network Traffic

You can manage all network traffic of the page and do any manipulations. For example, when the page requests a CSS file, you can send plain text as a response or when the page requests an HTML file, you can send a png file as a response. You can also send a true response too.

- You can Cache

You can cache any request/response pair with Service Worker and Cache API and you can access these offline content anytime.

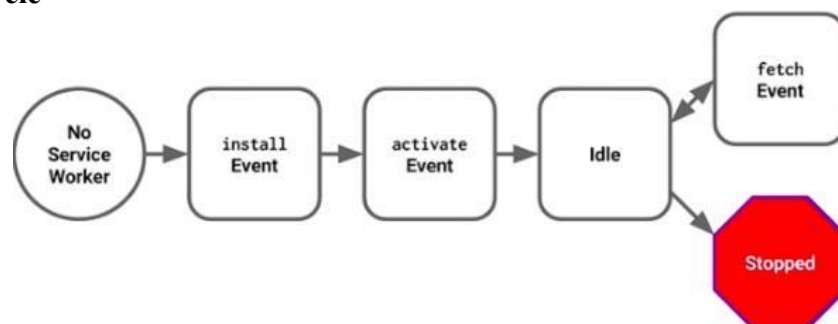
- You can manage Push Notifications

You can manage push notifications with Service Worker and show any information message to the user.

- You can Continue

Although Internet connection is broken, you can start any process with Background Sync of Service Worker.

#### Service Worker Cycle



A service worker goes through three steps in its life cycle:

- Registration
- Installation
- Activation

## **Registration**

To install a service worker, you need to register it in your main JavaScript code. Registration tells the browser where your service worker is located, and to start installing it in the background.

## **Installation**

Once the browser registers a service worker, installation can be attempted. This occurs if the service worker is considered to be new by the browser, either because the site currently doesn't have a registered service worker, or because there is a byte difference between the new service worker and the previously installed one.

A service worker installation triggers an install event in the installing service worker. We can include an install event listener in the service worker to perform some task when the service worker installs. For instance, during the install, service workers can precache parts of a web app so that it loads instantly the next time a user opens it (see caching the application shell). So, after that first load, you're going to benefit from instant repeat loads and your time to interactivity is going to be even better in those cases.

## **Activation**

Once a service worker has successfully installed, it transitions into the activation stage. If there are any open pages controlled by the previous service worker, the new service worker enters a waiting state. The new service worker only activates when there are no longer any pages loaded that are still using the old service worker. This ensures that only one version of the service worker is running at any given time.

When the new service worker activates, an activate event is triggered in the activating service worker. This event listener is a good place to clean up outdated caches

**Code:****//serviceworker.js**

```
var staticCacheName = "pwa";
```

```
self.addEventListener("install", function (e) {  
  e.waitUntil(  
    caches.open(staticCacheName).then(function (cache) {  
      return cache.addAll(["/", "/index.html", "/about.html", "/blog.html", "/contact.html",  
"/services.html", "/img/slider-img4.jpg", "/css/main.css"]);  
    })  
  );  
});
```

```
self.addEventListener("fetch", function (event) {  
  console.log(event.request.url);
```

```
  event.respondWith(  
    caches.match(event.request).then(function (response) {  
      return response || fetch(event.request);  
    })  
  );  
});
```

**JS in index.html:**

```
<script>
```

```
  if ('serviceWorker' in navigator) {  
    window.addEventListener('load', () => {  
      navigator.serviceWorker.register('/serviceworker.js')  
        .then(registration => {  
          console.log('Service Worker registered with scope:', registration.scope);  
        })  
        .catch(error => {  
          console.error('Service Worker registration failed:', error);  
        });  
    });  
  }  
</script>
```

## Output:

Click on the three dots-> More tools-> Developer tools-> Application-> Service workers

The screenshot shows the Chrome DevTools Application panel with the 'Service workers' section selected. The page is 'Heading Two' from 'Glozzome'. The Service workers section shows a worker for 'http://127.0.0.1:5500/' with status 'activated and is running'. The 'Update Cycle' table shows the worker's lifecycle: Install, Wait, and Activate.

Version	Update Activity	Timeline
#2684	Install	
#2684	Wait	
#2684	Activate	

The service worker is active and running.

Go to Cache storage:

The screenshot shows the Chrome DevTools Application panel with the 'Cache storage' section selected. The page is 'Heading Three' from 'Glozzome'. The Cache storage section shows a list of cached resources for 'http://127.0.0.1:5500/'. The table lists resources like /, /about.html, /blog.html, /contact.html, /css/, /css/main.css, /img/slider-img4.jpg, /index.html, and /services.html.

#	Name	Respon...	Conten...	Conten...	Time C...	Vary H...
0	/	basic	text/html	12,929	25/3/2...	Origin
1	/about.html	basic	text/html	7,956	25/3/2...	Origin
2	/blog.html	basic	text/html	8,194	25/3/2...	Origin
3	/contact.html	basic	text/html	6,805	25/3/2...	Origin
4	/css/	basic	text/html	9,720	25/3/2...	Origin
5	/css/main.css	basic	text/css	4,219	25/3/2...	Origin
6	/img/slider-img4.jpg	basic	image/j...	97,426	25/3/2...	Origin
7	/index.html	basic	text/html	12,929	25/3/2...	Origin
8	/services.html	basic	text/html	18,094	25/3/2...	Origin