PWA EXPT-08

Name: Swarali Dhobale
Class:D15A-13

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.

What can’t we do with Service Workers?

- You can’t access the **Window**
  You can’t access the window, therefore, You can’t manipulate DOM elements. But, you can communicate to the window through post Message and manage processes that you want.

- You can’t work it on **80 Port**
  Service Worker just can work on HTTPS protocol. But you can work on localhost during development.

Code and Output:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Wildlife Monitoring</title>
   <style>
          body {
         font-family: Arial, sans-serif;
         margin: 0;
         padding: 0;
         background:
url("https://media.istockphoto.com/id/537389352/photo/tropical-rainforest.webp?b=1&s=170667a&w=0&k=20&c=rPxzpapBr16QF47PWJ474qVXE1SjaRImJvt6pClavew=") no-repeat fixed;
         background-size: cover;
      }

      header {
         background-color: #2f292f;
         color: white;
         padding: 9px;
         text-align: center;
      }
      nav ul {
         list-style-type: none;
         padding: 0;
      }
      nav li {
         display: inline;
         margin-right: 20px;
```

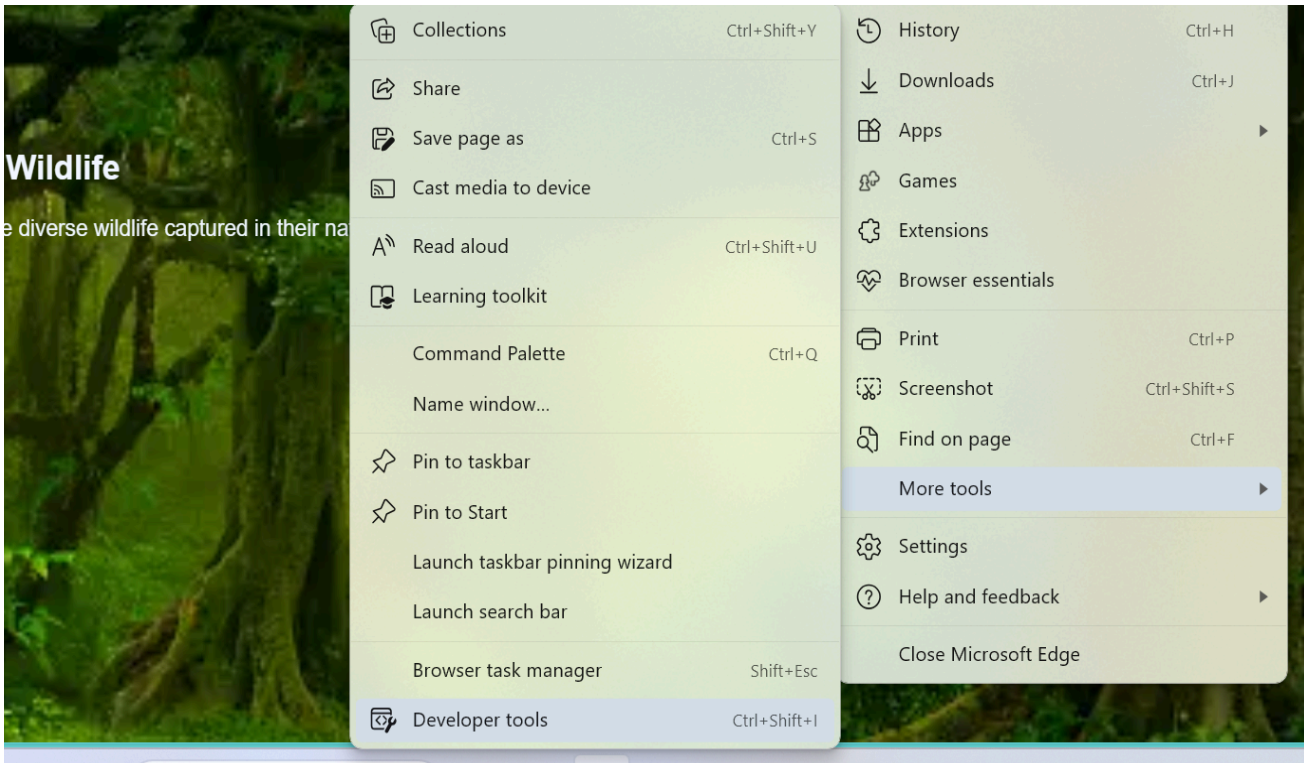
```
}
.main-content {
    max-width: 800px;
    margin: 120px auto;
    padding: 20px;
    color: aliceblue;
    border-radius: 10px;
}

.predicted-image {
    border: 2px solid red;
    max-width: 100%;
}

footer {
    background-color: #2f292f;
    color: white;
    text-align: center;
    padding: 10px;
    position: fixed;
    bottom: 0;
    width: 100%;
}

nav a {
    text-decoration: none;
    padding: 8px 12px;
    border: 1px solid #4CAF50;
    color: #4CAF50;
    border-radius: 5px;
    transition: background-color 0.3s;
}

nav a:hover {
    background-color: #4CAF50;
    color: white;
}

.loading-message-text {
    color: yellow;
    font-size: 18px;
}
@media (max-width: 600px) {
    nav ul {
```

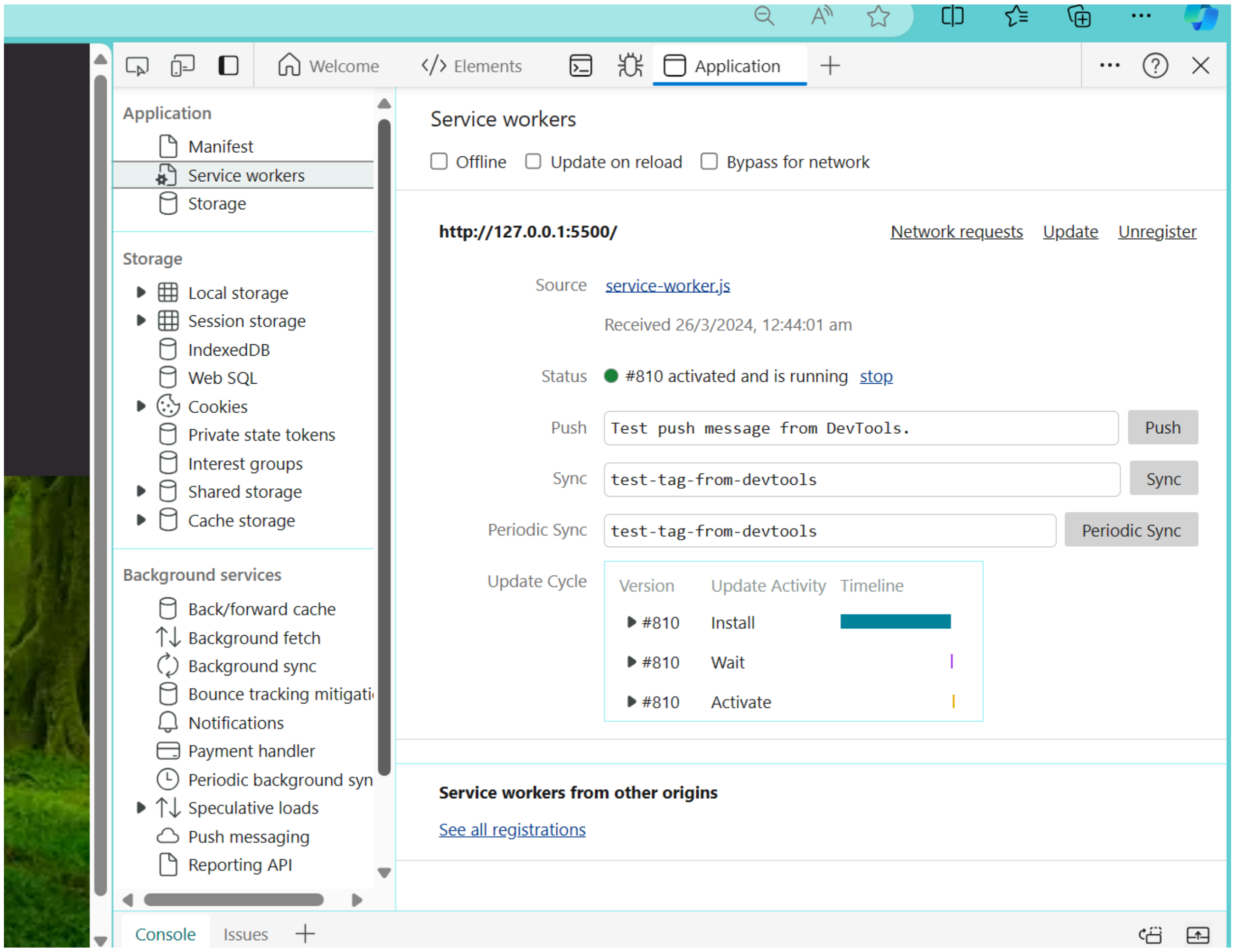
```
                text-align: center;
            }

            nav li {
                display: block;
                margin: 30px 0;
            }
        }
    </style>
<script>
    document.addEventListener("DOMContentLoaded", function () {
        var imageResultsBtn = document.getElementById("image-results-btn");
        var loadingMessage = document.getElementById("loading-message");

        imageResultsBtn.addEventListener("click", function () {
            loadingMessage.innerHTML = '<p class="loading-message-text">Hang on! Your content
is being loaded...</p>';
            setTimeout(function () {
                loadingMessage.innerHTML = '';

            }, 30000);
        });
    });
</script>
</head>
<body>
<header>
    <h1>Welcome to Wildlife Monitoring</h1>
    <nav>
        <ul>
            <li><a id="image-results-btn" href="/results">Image Results</a></li>
            <li><a href="/human">Human Interaction</a></li>
            <li><a href="/contact">Contact Us</a></li>
            <li><a href="/blogs">Blogs</a></li>
        </ul>
    </nav>
</header>
<section class="main-content">
    <div id="loading-message"></div>
    <h2>About Wildlife</h2>
    <p>Explore the diverse wildlife captured in their natural habitats.</p></section>
<script src="app.js"></script>
</body>
```

app.js

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

Output:





Application
- Manifest
- Service workers
- Storage

Storage
- Local storage
- Session storage
- IndexedDB
- Web SQL
- Cookies
- Private state tokens
- Interest groups
- Shared storage
- Cache storage

Background services
- Back/forward cache
- Background fetch
- Background sync
- Bounce tracking mitigati
- Notifications
- Payment handler
- Periodic background syn
- Speculative loads
- Push messaging
- Reporting API

Service workers

☐ Offline ☐ Update on reload ☐ Bypass for network

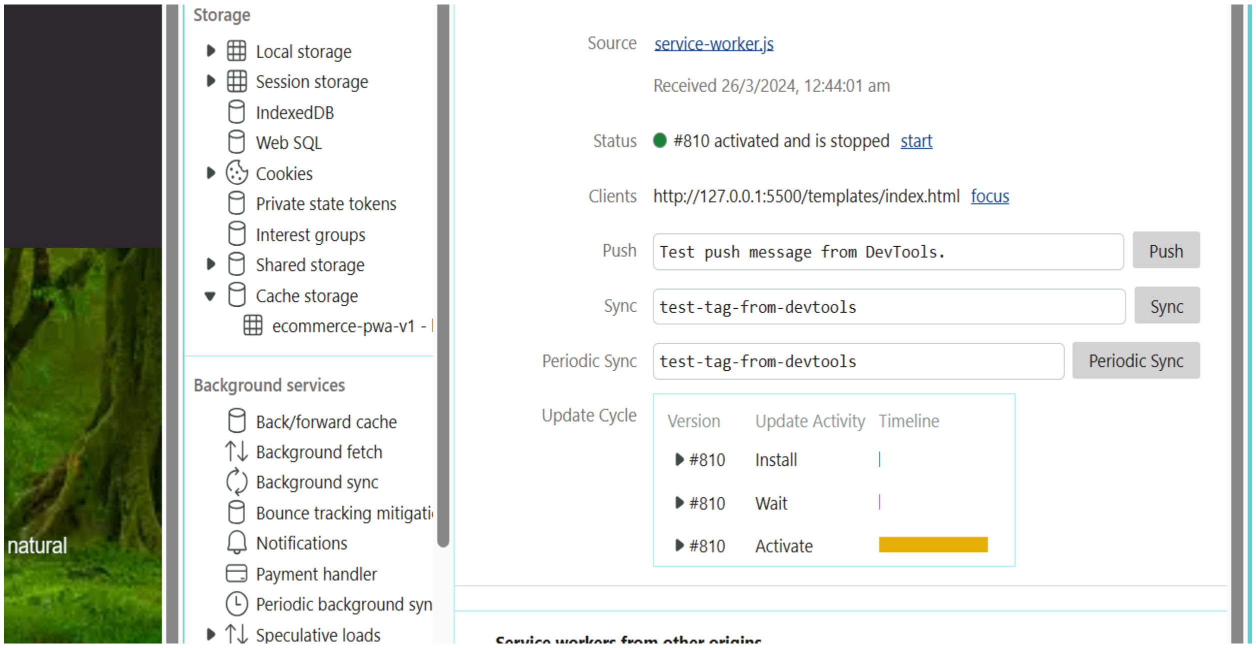
http://127.0.0.1:5500/ Network requests Update Unregister

Source	service-worker.js
	Received 26/3/2024, 12:44:01 am
Status	● #810 activated and is running stop
Push	Test push message from DevTools. [Push]
Sync	test-tag-from-devtools [Sync]
Periodic Sync	test-tag-from-devtools [Periodic Sync]
Update Cycle	

Version	Update Activity	Timeline
#810	Install	▬
#810	Wait	|
#810	Activate	|

Service workers from other origins

See all registrations

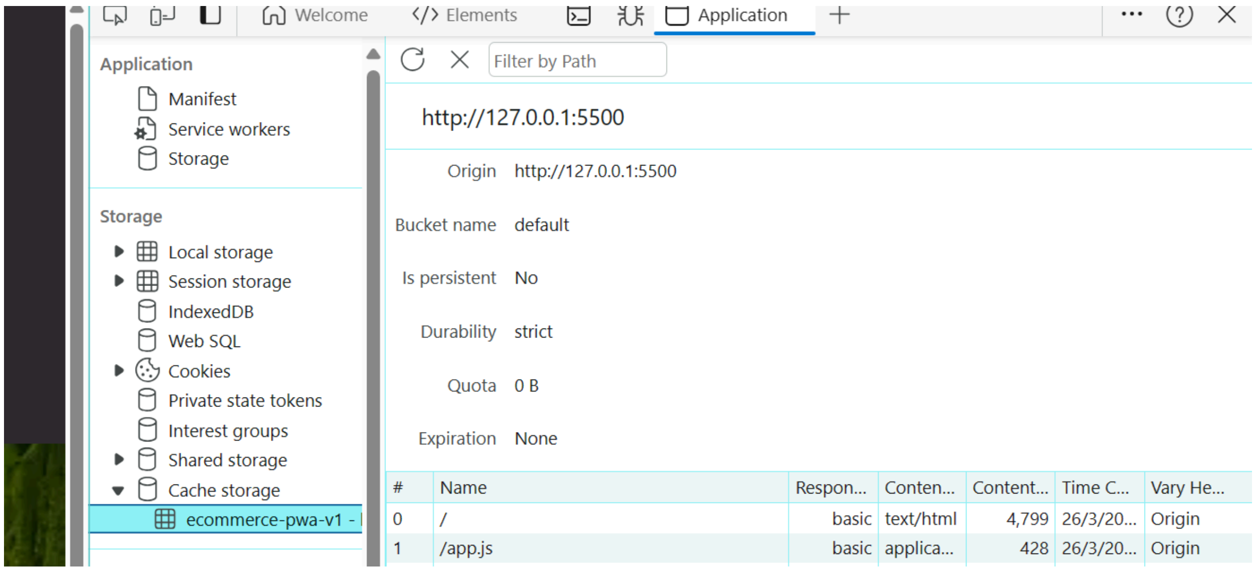


Storage
- Local storage
- Session storage
- IndexedDB
- Web SQL
- Cookies
- Private state tokens
- Interest groups
- Shared storage
- Cache storage
  - ecommerce-pwa-v1 -

Background services
- Back/forward cache
- Background fetch
- Background sync
- Bounce tracking mitigati
- Notifications
- Payment handler
- Periodic background syn
- Speculative loads

Source	service-worker.js
	Received 26/3/2024, 12:44:01 am
Status	● #810 activated and is stopped start
Clients	http://127.0.0.1:5500/templates/index.html focus
Push	Test push message from DevTools. [Push]
Sync	test-tag-from-devtools [Sync]
Periodic Sync	test-tag-from-devtools [Periodic Sync]
Update Cycle	

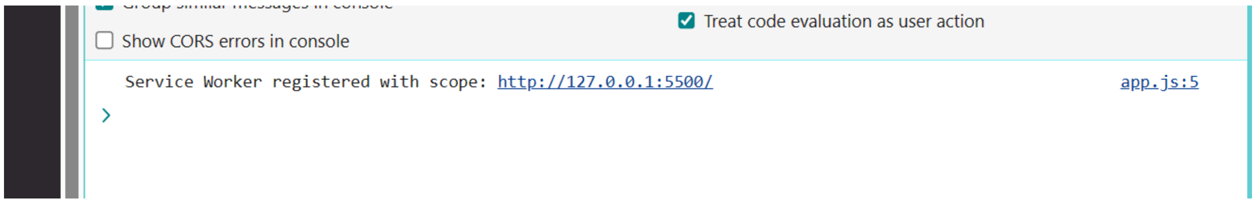
Version	Update Activity	Timeline
#810	Install	|
#810	Wait	|
#810	Activate	▬



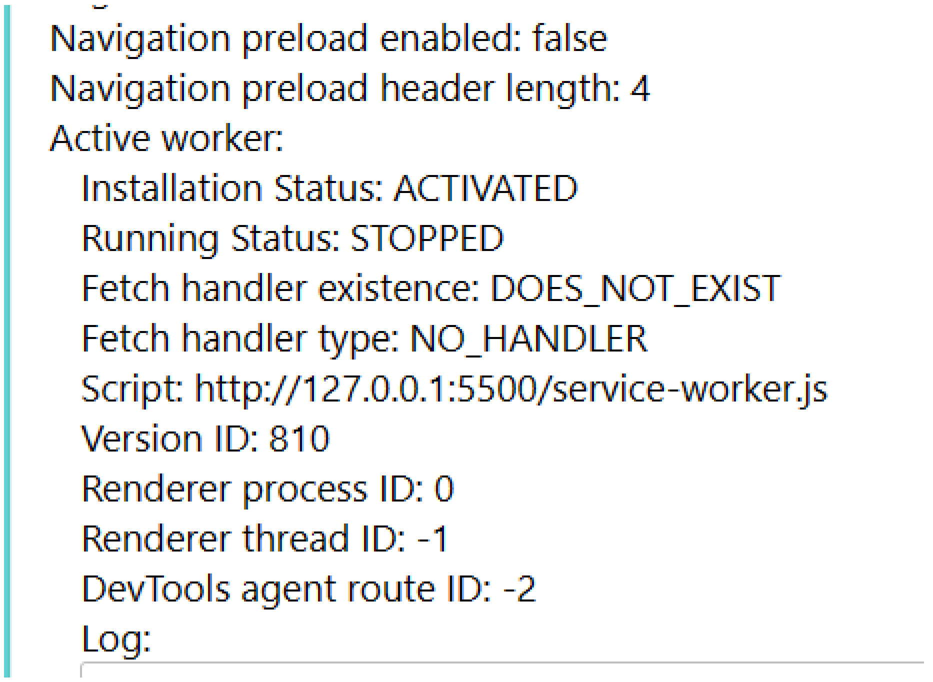
http://127.0.0.1:5500

Origin	http://127.0.0.1:5500
Bucket name	default
Is persistent	No
Durability	strict
Quota	0 B
Expiration	None

#	Name	Respon...	Conten...	Content...	Time C...	Vary He...
0	/	basic	text/html	4,799	26/3/20...	Origin
1	/app.js	basic	applica...	428	26/3/20...	Origin


Show CORS errors in console
Treat code evaluation as user action

```
Service Worker registered with scope: http://127.0.0.1:5500/                app.js:5
```



```
Navigation preload enabled: false
Navigation preload header length: 4
Active worker:
  Installation Status: ACTIVATED
  Running Status: STOPPED
  Fetch handler existence: DOES_NOT_EXIST
  Fetch handler type: NO_HANDLER
  Script: http://127.0.0.1:5500/service-worker.js
  Version ID: 810
  Renderer process ID: 0
  Renderer thread ID: -1
  DevTools agent route ID: -2
  Log:
```

<u>Conclusion</u>: Registered a service worker, and completed the installation and activation process for a new service worker for the PWA.