

## Experiment No: 08

**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 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.

## What can't we do with Service Workers?

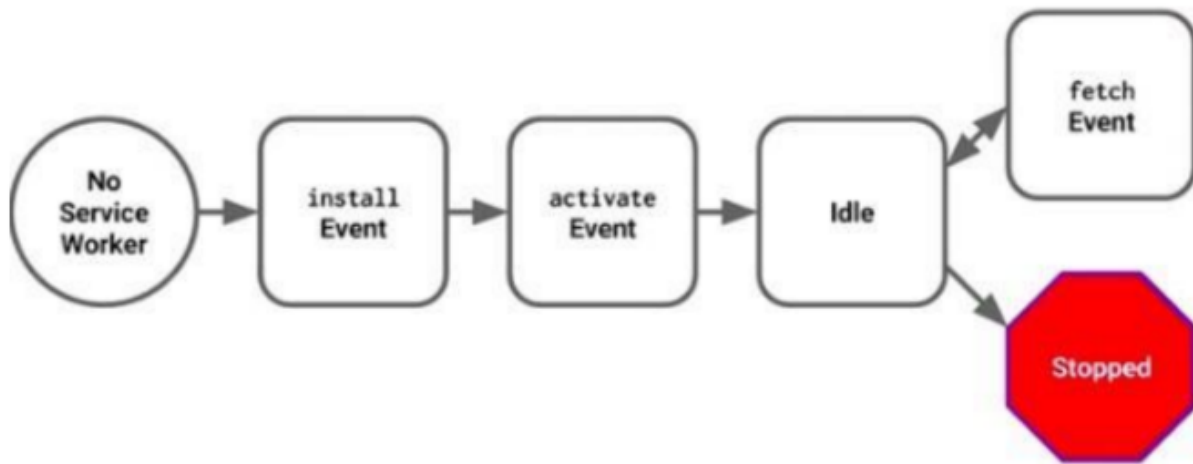
- You can't access the **Window**

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.

## Service Worker Cycle



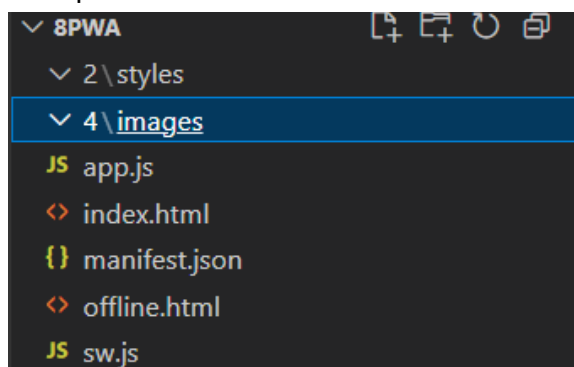
A service worker goes through three steps in its life cycle:

- 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. Let's look at an example:



## app.js

```
// Service Worker Registration
if ('serviceWorker' in navigator) {
  window.addEventListener('load', () => {
    navigator.serviceWorker.register('/sw.js')
      .then(registration => {
        console.log('SW registered: ', registration);
      })
      .catch(registrationError => {
        console.log('SW registration failed: ', registrationError);
      });
  });
}

// Listen for controller changes (updates)
navigator.serviceWorker.addEventListener('controllerchange', () => {
  console.log('New service worker activated!');
  window.location.reload();
});
```

## sw.js

```
const CACHE_NAME = 'ecommerce-pwa-v1';
const OFFLINE_URL = '/offline.html';
const ASSETS_TO_CACHE = [
  '/',
  '/index.html',
  '/styles/main.css',
  '/scripts/app.js',
  '/images/logo.png',
  OFFLINE_URL,
  // Add other critical assets for your e-commerce site
];

// Install event - caching essential assets
self.addEventListener('install', event => {
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then(cache => {
        console.log('Opened cache');
        return cache.addAll(ASSETS_TO_CACHE);
      })
      .then(() => self.skipWaiting()) // Force the waiting service worker to become
active
  );
});

// Activate event - clean up old caches
self.addEventListener('activate', event => {
  const cacheWhitelist = [CACHE_NAME];
  event.waitUntil(
    caches.keys().then(cacheNames => {
      return Promise.all(
        cacheNames.map(cacheName => {
          if (cacheWhitelist.indexOf(cacheName) === -1) {
            return caches.delete(cacheName);
          }
        })
      );
    })
  );
});
```

```

    })
  );
})
.then(() => self.clients.claim()) // Take control of all clients immediately
);
});
// Fetch event - network with cache fallback
self.addEventListener('fetch', event => {
  // Skip cross-origin requests
  if (!event.request.url.startsWith(self.location.origin)) {
    return;
  }
  // For API calls, use network-first strategy
  if (event.request.url.includes('/api/')) {
    event.respondWith(
      fetch(event.request)
        .then(response => {
          // Cache the API response if successful
          const responseToCache = response.clone();
          caches.open(CACHE_NAME)
            .then(cache => cache.put(event.request, responseToCache));
          return response;
        })
        .catch(() => {
          // If network fails, try to get from cache
          return caches.match(event.request);
        })
    );
  } else {
    // For static assets, use cache-first strategy
    event.respondWith(
      caches.match(event.request)
        .then(cachedResponse => {
          // Return cached response if found
          if (cachedResponse) {
            return cachedResponse;
          }
          // Otherwise fetch from network
          return fetch(event.request)
            .then(response => {
              // Check if we received a valid response
              if (!response || response.status !== 200 || response.type !== 'basic')
            {
              return response;
            }
            // Clone the response
            const responseToCache = response.clone();

            caches.open(CACHE_NAME)
              .then(cache => {
                cache.put(event.request, responseToCache);
              });

            return response;
          }
        })
    );
  }
}

```

```

    })
    .catch(() => {
      // If both fail, show offline page for HTML requests
      if (event.request.headers.get('accept').includes('text/html')) {
        return caches.match(OFFLINE_URL);
      }
    });
  })
);
});
// Push notification event handling
self.addEventListener('push', event => {
  const data = event.data.json();
  const title = data.title || 'New update from our store!';
  const options = {
    body: data.body || 'Check out our latest products and offers!',
    icon: '/images/icon-192x192.png',
    badge: '/images/badge-72x72.png'
  };
  event.waitUntil(
    self.registration.showNotification(title, options)
  );
});
self.addEventListener('notificationclick', event => {
  event.notification.close();
  event.waitUntil(
    clients.openWindow('https://your-ecommerce-site.com/latest-offers')
  );
});

```

## Index.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="theme-color" content="#ffffff" />
  <title>Simple E-commerce PWA</title>
  <link rel="manifest" href="/manifest.json" />
  <link rel="icon" href="https://via.placeholder.com/192" type="image/png" />
  <link rel="stylesheet" href="/styles/main.css" />
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 20px;
      background: #f0f0f0;
    }

    h1 {
      text-align: center;
    }
  </style>

```

```
.product-grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
  gap: 20px;
  max-width: 1200px;
  margin: 0 auto;
}

.product-card {
  background: white;
  padding: 15px;
  border-radius: 8px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  text-align: center;
}

.product-card img {
  max-width: 100%;
  height: 150px;
  object-fit: cover;
}

button {
  background: #007bff;
  color: white;
  border: none;
  padding: 10px 20px;
  border-radius: 5px;
  cursor: pointer;
}

button:hover {
  background: #0056b3;
}

#notification {
  position: fixed;
  bottom: 20px;
  right: 20px;
  background: #28a745;
  color: white;
  padding: 10px 20px;
  border-radius: 5px;
  display: none;
}

</style>
</head>
<body>
  <h1>Welcome to Our E-commerce Store</h1>
  <div class="product-grid" id="products"></div>
  <div id="notification">Item added to cart!</div>

  <script>
    // Sample product data
```

```

const products = [
  { id: 1, name: "T-Shirt", price: 19.99, image:
"https://via.placeholder.com/200" },
  { id: 2, name: "Jeans", price: 39.99, image: "https://via.placeholder.com/200"
},
  { id: 3, name: "Shoes", price: 59.99, image: "https://via.placeholder.com/200"
}
];

// Render products
function renderProducts() {
  const grid = document.getElementById('products');
  grid.innerHTML = products.map(product => `
    <div class="product-card">
      
      <h3>${product.name}</h3>
      <p>${product.price}</p>
      <button onclick="addToCart(${product.id})">Add to Cart</button>
    </div>
  `).join('');
}

// Add to cart simulation
function addToCart(productId) {
  const notification = document.getElementById('notification');
  notification.style.display = 'block';
  setTimeout(() => notification.style.display = 'none', 2000);
}

// Register service worker
if ('serviceWorker' in navigator) {
  window.addEventListener('load', () => {
    navigator.serviceWorker.register('/sw.js')
      .then(reg => {
        console.log('Service Worker registered', reg);
      })
      .catch(err => {
        console.log('Service Worker registration failed:', err);
      });
  });
}

renderProducts();
</script>
</body>
</html>

```

## offline.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Offline | Your E-commerce Store</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
      padding: 50px;
    }
    h1 {
      color: #333;
    }
    p {
      color: #666;
    }
  </style>
</head>
<body>
  <h1>You're Offline</h1>
  <p>It looks like you've lost your internet connection. Some features may not
be available.</p>
  <p>We'll automatically show you the latest version when you're back
online.</p>
  <button id="reload">Try Again</button>

  <script>

    document.getElementById('reload').addEventListener('click', () => {

      window.location.reload();

    });

  </script>
</body>
</html>
```




## manifest.js


```
{
  "name": "E-commerce Store",
  "short_name": "EcomStore",
  "start_url": "/",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#4285f4",
  "icons": [
    {
      "src": "/images/icon-72x72.png",
      "sizes": "72x72",
      "type": "image/png"
    },
    {
      "src": "/images/icon-96x96.png",
      "sizes": "96x96",
      "type": "image/png"
    },
    {
      "src": "/images/icon-128x128.png",
      "sizes": "128x128",
      "type": "image/png"
    },
    {
      "src": "/images/icon-144x144.png",
      "sizes": "144x144",
      "type": "image/png"
    },
    {
      "src": "/images/icon-152x152.png",
      "sizes": "152x152",
      "type": "image/png"
    },
    {
      "src": "/images/icon-192x192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "/images/icon-384x384.png",
      "sizes": "384x384",
      "type": "image/png"
    },
    {
      "src": "/images/icon-512x512.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ]
}
```

127.0.0.15500


Welcome to Our E-commerce Store



T-Shirt  
\$19.99  
Add to Cart



Jeans  
\$39.99  
Add to Cart



Shoes  
\$59.99  
Add to Cart

Item added to cart!

ElementsConsoleSourcesNetworkApplication

ServiceWorker registration successful with scope: /127.0.0.1-5500/?sw.js:1s

EiriusConsoutedLayoutEventtlragersStorage

1270.0.1:5500/

ScopeRegisteredManifestScope 127.0.0.5500/Service WorkersActivated activated runningStorage