RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

Department of Master of Computer Applications (MCA)

Mobile Application Development (MCA221IA)

Hand Notes

Unit - 2	**Topic:**Service Worker, Registering and Updating
Editor: 1RV24MC080 – R Dinesh	

List of Questions

1. **What is a Service Worker in PWAs?(2 Marks)**
2. **How do you register a Service Worker in a PWA?(4 Marks)**
3. **Explain how a Service Worker works in PWA?(6 Marks)**
4. **Describe the lifecycle and updating process of a Service Worker(8 Marks)**
5. **Explain Service Worker in detail,including registration and update process with Suitable example and code (10 Marks)**

02 Marks Question:

1.What is a Service Worker in PWAs?
Answer:
A Service Worker is a script that the browser runs in the background, separate from the web page. It enables features such as offline access, background sync, and push notifications in Progressive Web Applications (PWAs).
Example :
Caching assets for offline usage.

04 Marks Question:

2.How do you register a Service Worker in a PWA?
Answer:
To register a Service Worker, use JavaScript in your main JS file (typically index.js or main.js). The browser checks if Service Workers are supported, then registers the worker.
Code:

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

Example :
When user opens the app for the first time, service-worker.js gets registered.

06 Marks Question:

3.Explain how a Service Worker works in PWA?
Answer:
A Service Worker runs independently of the web page. It intercepts network requests, manages cache, and serves offline content. It has a lifecycle:
1. Install – Occurs when the service worker is first registered.
2. Activate – Cleans up old caches.
3. Fetch – Intercepts network requests and responds with cached or network data.
Code:

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

self.addEventListener('fetch', event => {
  event.respondWith(
    caches.match(event.request)
      .then(response => response || fetch(event.request))
  );
});
```

Diagram:

```
+-----------+     +-----------+     +----------+
|  Browser  | --> | Service   | --> | Network  |
|           | <-- |  Worker   | <-- |/Cache    |
+-----------+     +-----------+     +----------+
```

Example:
Offline-first apps like weather apps or news apps using cached data.

08 Marks Question

4.Describe the lifecycle and updating process of a Service Worker
Answer:
The lifecycle of a Service Worker includes:
1. Installation: Caches static resources.
2. Activation: Removes outdated caches.
3. Fetch Handling: Responds to fetch events from the page.

Updating Process:
- When a new version of the service worker is detected, it goes through the install phase again.
- It won't activate until all tabs using the old service worker are closed or reloaded.
- Developers can use skipWaiting() and clients.claim() to force activation.
Code:

```
self.addEventListener('install', event => {
  self.skipWaiting(); // Forces waiting SW to become active
});

self.addEventListener('activate', event => {
  event.waitUntil(clients.claim()); // Allows SW to control clients immediately
});
```

Old SW Active → New SW Installed → Waiting → Activated
                         (skipWaiting + clients.claim)

Example:
A PWA weather app updates cached weather data with new API response by replacing old cache with new cache on activation.

10 Marks Question

5.Explain Service Worker in detail, including registration and update process with suitable example and code
Answer:
A Service Worker is a background script that helps build reliable, performant web apps. It acts as a network proxy, managing requests and caching. Key roles:
- Offline support
- Push notifications
- Background sync

Registration Process:
Done using navigator.serviceWorker.register. It checks for browser support and links the service worker file.

Updating Process:
Happens automatically when the browser detects changes in the service worker script. Developers can customize update behavior with lifecycle events.
Code:

```
if ('serviceWorker' in navigator) {
  navigator.serviceWorker.register('/sw.js').then(reg => {
    console.log('SW registered:', reg);
  });
}

// Inside sw.js
self.addEventListener('install', event => {
  console.log('Installing SW...');
  self.skipWaiting();
});

self.addEventListener('activate', event => {
  console.log('Activating SW...');
  event.waitUntil(clients.claim());
});

self.addEventListener('fetch', event => {
  event.respondWith(
    caches.match(event.request).then(response => {
      return response || fetch(event.request);
    })
  );
});
```

(Lifecycle):

$$\begin{tikzcd}
\text{Client} \arrow[r] & \text{Register} \arrow[r] \arrow[dr] & \text{Install} \arrow[r] & \text{Activate} \arrow[r] & \text{Fetch} \\
 & & \text{Update} \arrow[r] & \text{Reinstall} \arrow[r] & \text{Activate}
\end{tikzcd}$$

Example:
A news app uses Service Workers to cache articles for offline reading. When new articles are available, the updated service worker activates and caches the latest content.