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 No: 02	Topic: Promises
Editor: 1RV24MC061 – LEELANJALI P	

List of Questions

Question 1 (02 Marks)
Question 2 (02 Marks)
Question 3 (04 Marks)
Question 4 (04 Marks)
Question 5 (06 Marks)
Question 6 (08 Marks)
Question 7 (10 Marks)

02 Marks Questions:

Q: What is a Service Worker in a Progressive Web App (PWA)?

Answer:
A Service Worker is a script that runs in the background of a PWA, enabling features like offline access, background sync, and push notifications.

Code:

```
self.addEventListener('install', event => {
  console.log('Service Worker installing.');
});
```

Example:

Caching static files for offline usage using a service worker.

Q: What is a Promise in JavaScript?

Answer:
A Promise is an object in JavaScript that represents the eventual completion (or failure) of an asynchronous operation and its resulting value.

Code**:**

```
let promise = new Promise(function(resolve, reject) {
  resolve("Success!");
});
```

Example:
Fetching data from an API using .then() and .catch().

---

04 Marks Questions:

Q: List any four features of Service Workers.

Answer:

1. Offline support via caching.
2. Background data sync.
3. Push notifications.
4. Interception of network requests.

Code:

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

Example:
Using a service worker to cache HTML and CSS files for offline use.

Q: How do Promises work in Service Workers?

Answer:
Service Workers use Promises to handle asynchronous tasks such as caching files or fetching network resources. Promises ensure that the service worker completes tasks like installation or fetch before proceeding.

Code:

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

Example:
Using event.waitUntil() with a Promise to cache files during install.

---

06 Marks Questions:

Q: Explain the lifecycle of a Service Worker.

Answer:
The Service Worker lifecycle has three main stages:

1. **Install:** Caches required resources.
2. **Activate:** Cleans up old caches and gets control.
3. **Fetch:** Intercepts network requests to serve cached responses or fetch new ones.

Code:

```
self.addEventListener('activate', event => {
  console.log('Service Worker activated');
});
```

Diagram:
Install → Activate → Fetch (with arrows showing flow)

Example:
Service Worker installing and caching assets, then handling fetch requests offline.

---

08 Marks Questions:

Q: Describe how Service Workers use Promises to enable offline functionality in PWAs.

Answer:
Service Workers use Promises in install and fetch events to manage caching and retrieving resources. During install, assets are cached using a Promise inside event.waitUntil(). During fetch, a Promise is used to respond with either cached data or data fetched from the network.

Code:

```
self.addEventListener('install', event => {
  event.waitUntil(
    caches.open('static-cache').then(cache => {
      return cache.addAll(['/offline.html', '/styles.css']);
    })
  );
});
self.addEventListener('fetch', event => {
  event.respondWith(
    caches.match(event.request).then(response => {
      return response || fetch(event.request);
    }).catch(() => caches.match('/offline.html'))
  );
});
```

Diagram:
Network → Service Worker → Cache → Response

Example:
Accessing a website offline after it has been cached by the service worker.

10 Marks Questions:

Q: Discuss in detail the role of Promises in Service Worker operations with examples.

Answer:
Promises play a crucial role in handling asynchronous tasks in Service Workers. They are used in install, activate, and fetch events.

- **Install:** Uses Promises to cache files and ensures completion before activation.
- **Activate:** Promises help in cleaning up old caches.
- **Fetch:** Promises are used to respond with cached resources or fetch from the network. The event.waitUntil() and event.respondWith() methods rely on Promises to keep the Service Worker alive during these operations.

Code:

```
self.addEventListener('install', event => {
  event.waitUntil(
    caches.open('app-cache').then(cache => {
      return cache.addAll(['/index.html', '/main.js']);
    })
  );
});
self.addEventListener('fetch', event => {
  event.respondWith(
    caches.match(event.request).then(response => {
      return response || fetch(event.request);
    })
  );
});
```

Example:
A PWA that loads previously cached assets using Promises even when the device is offline.