Universidad Tecnológica de Tijuana
Ingeniería En Desarrollo Y Gestión De Software
Aplicaciones Web Progresivas

Cache management strategies in PWA's

Mercado Juarez Angel Hayr
0319124541@ut-tijuana.edu.mx
0319124541

Teacher
Dr. Ray Brunett Parra

Tijuana B.C, México
15 March 2024

Contents

Chapter 1

Cache management strategies in PWA's

1.1 Introduction

Progressive Web Apps (PWAs) are a new generation of web applications that offer an experience similar to that of a native application. To achieve this, PWAs use a variety of techniques, including caching web resources. Caching can improve the performance and reliability of a PWA by reducing the need to download resources from the network.

1.2 Cache Management Strategies

There are several different strategies for managing the cache in a PWA. Some of the most common strategies include:

- Cache-first: This strategy always tries to serve resources from the cache. If the resource is not in the cache, it is downloaded from the network and added to the cache.
- Network-first: This strategy first tries to serve resources from the network. If the resource is not available on the network, it is attempted to be served from the cache.
- Cache-only: This strategy only serves resources from the cache. If the resource is not in the cache, it is not returned.
- Stale-while-revalidate: This strategy serves the resource from the cache, even if it is outdated. At the same time, an updated version of the resource is downloaded from the network and added to the cache.

(Developers, 2023)

1.3 Implementation

To implement a Service Worker, you need to register a JavaScript file in the browser. This file defines the behavior of the Service Worker, including which events to intercept and how to respond to them.(?, ?)

- Registering the Service Worker: You need to register a JavaScript file with the browser using the navigator.serviceWorker.register() method. This file defines the behavior of the Service Worker, including which events to intercept and how to respond to them.
- Installing the Service Worker: Once registered, the Service Worker is downloaded and installed by the browser. This process happens in the background and does not require any user interaction.
- Activating the Service Worker: Once installed, the Service Worker is activated and takes control of all network requests for the pages that are within its scope. The scope is defined in the Service Worker registration file and can include specific URLs or entire directories.
- Handling Network Requests: The Service Worker can intercept and modify network requests made by the pages within its scope. This allows you to implement various caching strategies, perform offline fallback, or modify request headers before they are sent to the server.
- Responding to Push Notifications: Service Workers can also be used to receive push notifications from a server. When a push notification is received, the Service Worker can handle it by displaying a notification to the user or performing other actions.

The best strategy for a particular PWA will depend on the specific needs of the application. (Archibald, 2014)

1.4 Cache-First Strategy

The cache-first strategy is a good choice for PWAs that need to be available offline. This strategy ensures that users can access the PWA even if they do not have a network connection. However, the cache-first strategy can lead to users seeing outdated content.

1.5 Network-First Strategy

The network-first strategy is a good choice for PWAs that need to show the most up-to-date content. This strategy ensures that users always see the latest version of the PWA. However, the network-first strategy can lead to a poor user experience if the network is slow or unreliable.

1.6 Cache-Only Strategy

The cache-only strategy is a good choice for PWAs that do not need to be updated frequently. This strategy can improve the performance of the PWA by reducing the number of network requests. However, the cache-only strategy can lead to users seeing outdated content.

1.7 Stale-While-Revalidate Strategy

The stale-while-revalidate strategy is a good choice for PWAs that need to balance the need for up-to-date content with the need for performance. This strategy ensures that users always see the latest version of the PWA, but it also allows them to access the PWA offline. (Network, 2023)

1.8 Progressive Web App (PWA) Cache Management Example

Scenario:

Imagine creating a PWA that acts as a product catalog for an online store. The PWA should display product images, basic information (name, price, description), and allow users to add products to a shopping cart. The PWA should function offline, enabling users to browse the catalog and add products to the cart even without an internet connection.

Cache Strategy:

We will employ the stale-while-revalidate strategy for cache management. This strategy lets users access PWA content offline while updating the content in the background.

Implementation:

1. Service Worker Registration: A Service Worker will be registered in the PWA's index.html file. The Service Worker is responsible for intercepting resource requests and managing the cache. HTML if ('serviceWorker' in navigator) navigator.serviceWorker.register('service-worker.js');

2. Request Handling: The Service Worker intercepts resource requests and checks if the resource is cached. If found, it's delivered to the user. If not, it's downloaded from the network and added to the cache. JavaScript self.addEventListener('fetch', (event) =¿ event.respondWith(caches.match(event.request).then((cacheRespons =¿ return cacheResponse —— fetch(event.request).then((networkResponse) =¿ return caches.open('my-cache').then((cache) =¿ cache.put(event.request, networkResponse.clone()); return networkResponse;););)););

3. Cache Updates: The Service Worker handles background cache updates. JavaScript self.addEventListener('sync', (event) =¿ if (event.tag === 'sync-cache') event.waitUntil(caches.open('my-cache').then((cache) =¿ return cache.keys().then((cacheKeys) =¿ return Promise.all(cacheKeys.map((cacheKey)

=¿ return fetch(cacheKey).then((networkResponse) =¿ return cache.put(cacheKey, networkResponse););)););)););

4. Resources to Cache: It's recommended to cache the following resources: - Product Images - Basic Product Information (name, price, description) - PWA JavaScript Code

5. Cache Expiration: Defining a cache expiration policy is crucial. This allows removing unused resources and freeing up storage space.

Benefits: - The PWA functions offline, allowing users to access content even without an internet connection. - PWA performance improves as resources are loaded from the local cache. - Mobile data consumption reduces because resources aren't downloaded from the network on every PWA access.

Limitations: - The PWA might display outdated content if the cache isn't updated frequently. - Implementing cache management requires additional development effort.

1.9 Conclusion

Caching is an important part of building a fast and reliable PWA. By choosing the right cache management strategy, developers can improve the user experience and make their PWA more engaging for users.

References

Archibald, J. (2014). *The offline cookbook.* Retrieved from `https://jakearchibald.com/2014/offline-cookbook/`

Developers, G. (2023). *Progressive web apps.* Retrieved from `https://developers.google.com/web/progressive-web-apps/`

Network, M. D. (2023). *The cache api.* Retrieved from `https://developer.mozilla.org/en-US/docs/Web/API/Cache`