TOPIC:

Cache with network an mood offline

BY:

Salas Diaz Guillermo

GROUP:

10B

CLASS:

PWA

PROFESSOR:

Dr. Ray Brunett Parra Galavíz

Tijuana, Baja California, 21 de March del 2024

Introduction

The "Cache with network and offline mode" strategy is a powerful approach used in web development to enhance the performance and reliability of web applications. By utilizing the Service Worker API, developers can cache resources locally, allowing the application to load faster and provide offline functionality. This strategy is particularly beneficial for Progressive Web Apps (PWAs) and other web applications that require reliable performance even in challenging network conditions.

Índex

Table of ilustrations

1.What is a Cache whit network and mood offline?

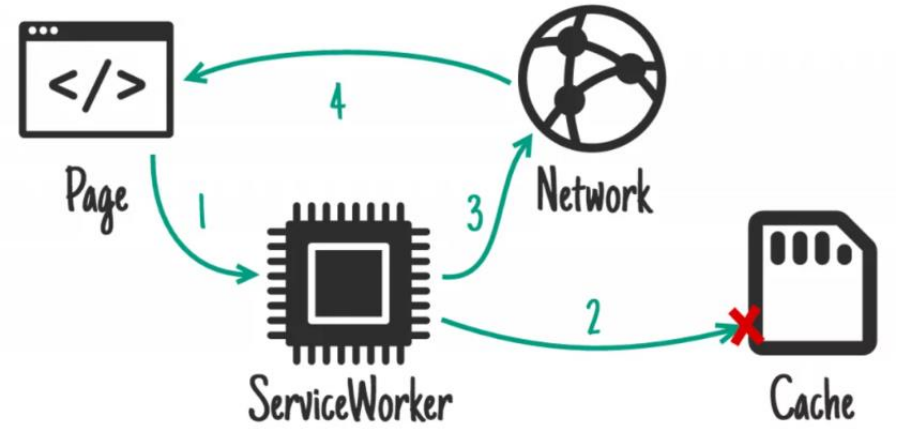
This approach refers to a caching strategy in web development where resources are first fetched from a cache (if available) and then, if the resource is not in the cache or the user is online, fetched from the network. This strategy allows the application to work offline by serving cached resources when the network is unavailable.

Cache strategies and offline mode

By combining the use of fetch API and cache API, we can create different caching strategies for our service workers. Some of the most common are detailed below.

Cache first

This strategy responds to requests with the version of the resource that is cached in the Cache Storage. If it's the first time and it doesn't find the cached resource, it will return the resource to us from the network and cache it for the next time we query it.

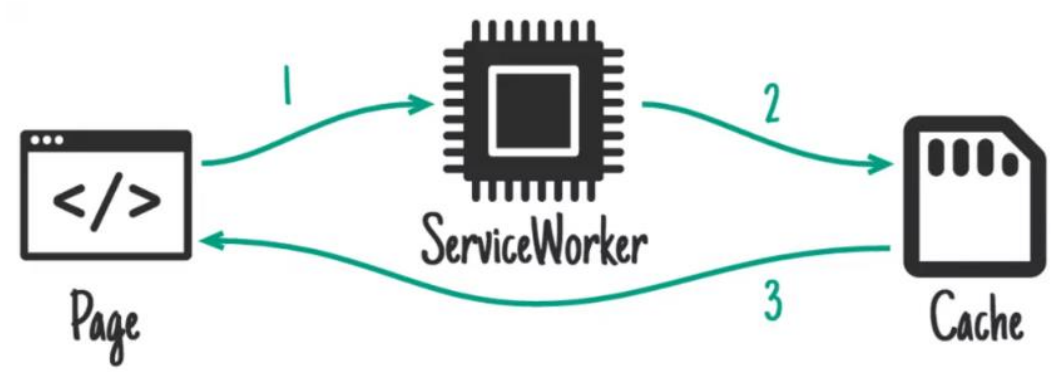


Normally, this strategy for medium resources, such as images, videos, etc., since they are heavier resources and take longer to load.

```
const cacheFirst = (event) => {
 event.respondWith(
   caches.match(event.request).then((cacheResponse) => {
     return cacheResponse || fetch(event.request).then((networkResponse) => {
       return caches.open(currentCache).then((cache) => {
         cache.put(event.request, networkResponse.clone());
         return networkResponse;
       })
     })
   })
 )
};
```

Cache only

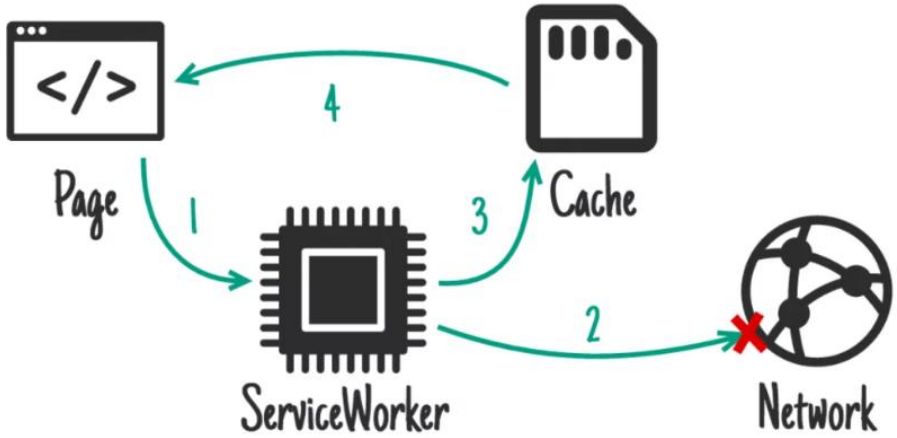
This strategy responds to requests directly with the cached version of the resource. If the cached version does not exist, an error will be returned.




```
const cacheOnly = (event) => {
 event.respondWith(caches.match(event.request));
};
```

Network first

This strategy prioritizes the most up-to-date version of the resource, always trying to obtain it first over the network, even if a cached version already exists. If the network response is successful, it will refresh the cache. If there is any error in the network response it will return the resource directly from cache, if any.




```
const networkFirst = (event) => {
 event.respondWith(
   fetch(event.request)
     .then((networkResponse) => {
       return caches.open(currentCache).then((cache) => {
         cache.put(event.request, networkResponse.clone());
         return networkResponse;
       })
     })
     .catch(() => {
       return caches.match(event.request);
     })
 )
};
```

I normally use it in API calls, where the frequency of content change is very low and it is not critical to return the latest version of the content. With this, I prioritize new content and make room for offline mode, where we will see the version of the data from our last online session.

Advantages:

- Improved performance: By caching resources locally, the application can load faster, especially for repeat visits or when the network connection is slow.
- Offline functionality: Users can continue using the application even when they are offline, as cached resources can be served instead of relying on a network connection.
- Reduced server load: Caching resources locally reduces the number of requests sent to the server, reducing server load and potentially saving bandwidth costs.
- Better user experience: With offline functionality and faster loading times, users experience a smoother and more reliable application, leading to higher satisfaction and engagement.
- Network resilience: The application can gracefully handle network failures or intermittent connectivity, providing a more robust user experience.
- Control over cache: Developers have control over what resources are cached and for how long, allowing them to optimize the cache for performance and reliability.
- Support for progressive web apps (PWAs): Implementing this caching strategy is a key component of creating PWAs, which are web applications that provide a native app-like experience, including offline functionality.

Disadvantages:

- Increased complexity: Implementing service workers and managing caching strategies adds complexity to the application development process, requiring additional time and effort.

- Cache management: Developers need to carefully manage the cache to ensure it does not grow too large or become stale, which can lead to unexpected behavior or performance issues.

- Cache invalidation: It can be challenging to invalidate the cache or ensure that users receive the latest version of resources, especially for frequently updated content.

- Storage limitations: Service workers have limited storage capacity, so caching large amounts of data or resources may not be feasible.

- Security concerns: Caching sensitive data or resources could pose security risks, as cached data may be accessible to other applications or users.

- Debugging and troubleshooting: Debugging issues related to service workers and caching can be challenging, as they operate in the background and are not always easy to inspect or debug.

- Compatibility issues: Not all browsers support service workers and the caching strategies required for offline functionality, which can limit the reach of applications using this approach.

- Performance trade-offs: While caching can improve performance in many cases, it can also lead to increased memory and CPU usage, especially if not implemented efficiently.

Applications:

- Progressive Web Apps (PWAs): PWAs are web applications that provide a native app-like experience, including offline functionality. Implementing the cache with network and offline mode is essential for PWAs to work reliably offline and offer fast loading times.

- E-commerce websites: E-commerce websites can benefit from caching product images, descriptions, and other resources to improve performance and allow users to browse products even when they are offline.

- News websites: News websites can use caching to store articles, images, and other content, allowing users to access the latest news even when they are offline or have a slow network connection.

- Productivity tools: Tools such as task managers, note-taking apps, and calendars can benefit from caching to ensure that users can access their data and perform essential tasks even when they are offline.

- Learning platforms: Platforms that offer online courses, tutorials, or educational content can use caching to allow users to access course materials even when they are offline or have limited connectivity.

- Maps and navigation apps: Maps and navigation apps can use caching to store map tiles, route information, and other data, allowing users to navigate even when they are offline or in areas with poor network coverage.

- Collaboration tools: Tools that facilitate collaboration, such as project management apps or document editors, can use caching to ensure that users can access and edit their documents even when they are offline.

- Gaming websites: Online gaming websites can use caching to store game assets and data, allowing users to play games even when they are offline or have a slow network connection.

Conclution

"Cache with network and offline mode" strategy offers several advantages, including improved performance, offline functionality, and better user experience. While it comes with some challenges, such as increased complexity and cache management issues, the benefits outweigh the drawbacks for many web applications. By carefully implementing and managing caching strategies, developers can create web applications that are faster, more reliable, and provide a seamless user experience, even in offline scenarios.

Bibliografía

docs, m. w. (2022 de Diciembre de 05). *Developer Mozilla*. Recuperado el 02 de February de 2024, de Org: https://developer.mozilla.org/es/docs/Web

Facilito, C. (2023 de 07 de 08). *Código Facilito*. Obtenido de https://codigofacilito.com/articulos/

MSEEdgeTeam, M. H. (03 de 04 de 2023). *Microsoft*. Obtenido de Learn: https://learn.microsoft.com/es-es/microsoft-edge/