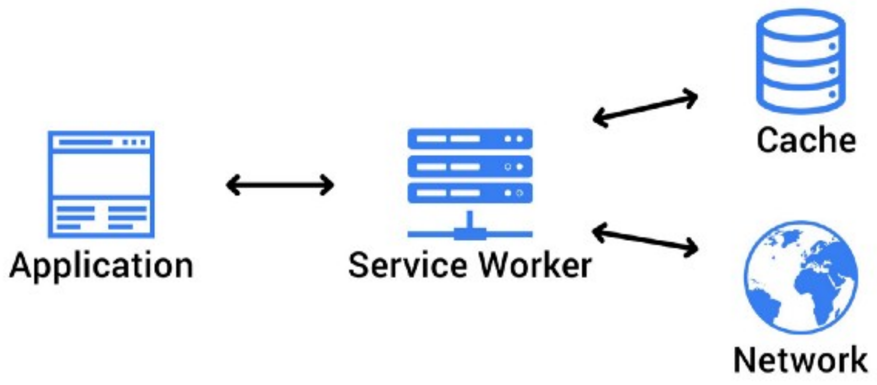
itnext.io

Caching in a PWA: When to use Optimistic vs Pessimistic

Craig Taub

4-5 minutes



One of the first steps for making any application a Progressive Web App is introducing an offline page via ServiceWorkers. ServiceWorkers bring with them lots of additional functionality, a large portion of which is around caching. Whether that is on a page-level, asset-level or even font-level there are strategies open to you for all types and scenarios of caching. Today we are going to talk about just 2 of them; they are optimistic caching and pessimistic caching. We will cover implementation details in follow-up posts so this will just focus on the strategies themselves.

Optimistic caching

You can think about the “network” being the focus of the strategy.

So Optimistic caching is when you check the network initially and then fallback to any cache storage, so it’s a *network-first* approach.

So the process would be:

1. *Does the user have a valid network connection*
2. *IF YES -> request data from network*
3. *IF NO -> request data from cache*

This is useful when you have data which changes frequently, and it will be a negative user experience (and possibly worse) to show stale data to a user.

Some of examples of where this might be useful are:

- User profile page: if a user has updated content and it is not showing correctly at this moment, this can be incredibly frustrating and worrying
- Stock levels: giving an incorrect availability count could lead to very confusing situations wasting time and money for both customer and seller.

A final thing to mention on “optimistic caching” is that it obviously relies on a reliable way to determine IF your user has a network connection, ideally without making them wait for a request to fail first. It is now possible to use the Network Information API from inside a ServiceWorker. With this the check becomes a 1 liner:

```
if (navigator.onLine) {
  // user is online
  // ...
```

Unfortunately support is not great (currently missing on FireFox and iPhone Safari). So you might have to do a feature check first before running this logic.

Pessimistic caching

So if optimistic is using network, pessimistic is not using network and is a *cache-first* approach. The idea for this type of caching is to check the cache for the data and then fallback to the network. If the network still fails then shown an error.

So the process would be:

1. *Does the cache have the required data*
2. *IF YES -> return this data from cache*
3. *IF NO -> request data from network*

This is useful when you have data which does not change often at all, and making a network request for it every time is wasteful and resource expensive.

Some of examples of where this might be useful are:

- Product page: if a products details do not change very often, you can be confident that the same product ID/SKU content will remain the same for a given period of time
- Product family page: if products within a family do not change very often then it makes little sense to request the details every time.
- Blog content: if articles written are unlikely to change often it seems safe to utilise cache

A real world example of where we see pessimistic-caching-like behaviour is in the concept of “Optimistic UI”. Where we make local changes on the assumption a remote operation will succeed (e.g. a user clicks save and a message suggesting the save is complete is shown, even though the network request is not back yet). The Apollo documentation has a great section all about it. This is pessimistic-caching-like as it is cache-first.

To sum up

You can see above that both strategies are useful under different situations so it's all about making the right call for a given page/asset/resource etc. I feel it is great that we have so many options available as it really helps us make an experience on our application the best it can possibly be 💪

If you found this useful please spare a clap, thanks 😁