API caching in clients

An overview of techniques for caching at a web API level from a client perspective.

About me: Arindam Pradhan

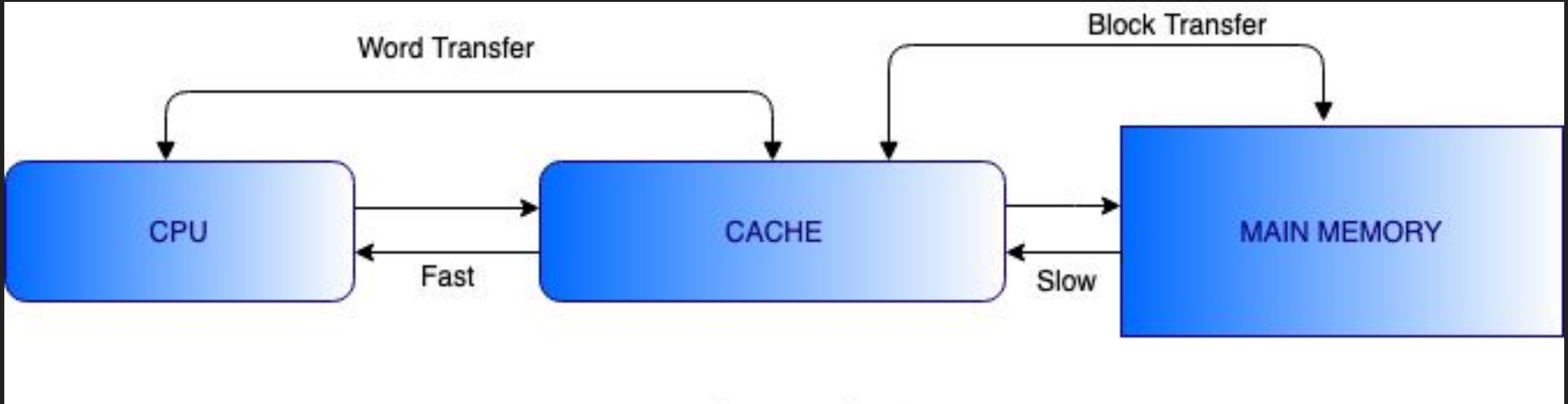
- Senior Frontend Developer @meesho
- Team: UI
- Github: arindampradhan
- Hobbies: trekking, bike riding

Topics:

1. Cache and it’s type
2. Cache control settings and it’s http protocol
3. Request (axios) implementation with a programmable cache (explicit way)
4. Workbox implementation with a programmable network (implicit way)
5. Caching data in a store layer (redux)

What is a cache?

Caches are usually very near to cpu also called CPU memory.



Single Cache

Cache types

Where does clients store the data ?

Chrome: Disk Cache, memory Cache, Localstorage, Indexdb, Cachestorage.

Android: SQlite, PouchDB, AsyncStorage class

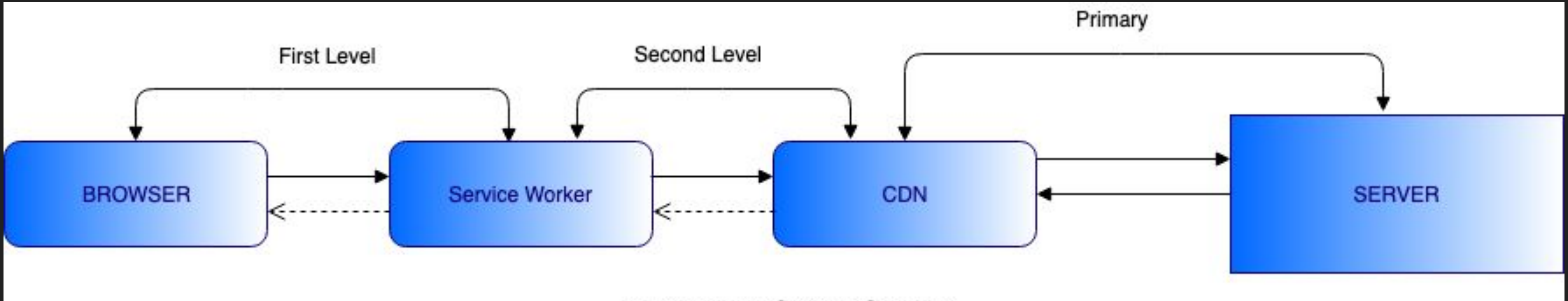
Client side cache vs server side cache

.

Provide offline experience.	Cannot provide offline experience
No need to transfer anything over network.	Has to go over the network again (expensive).
Reduce server overhead as each cache is specific to client.	Reduces server overhead but with a caching server in the mid of server and client.
Avoid transferring the same data over the network repeatedly.	Used to avoid making expensive database operations repeatedly.

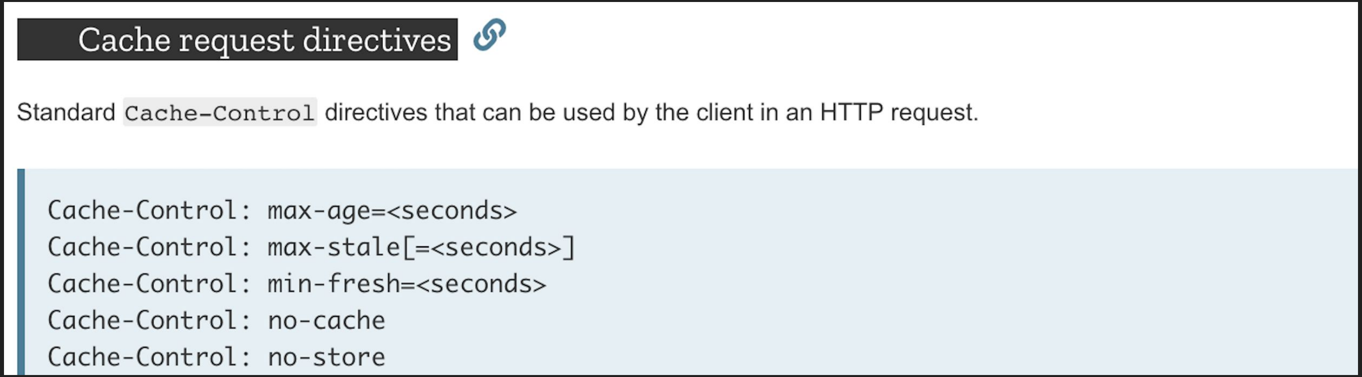
How it should be done ?

Most well engineered application actually use both.



Multi level Client Cache

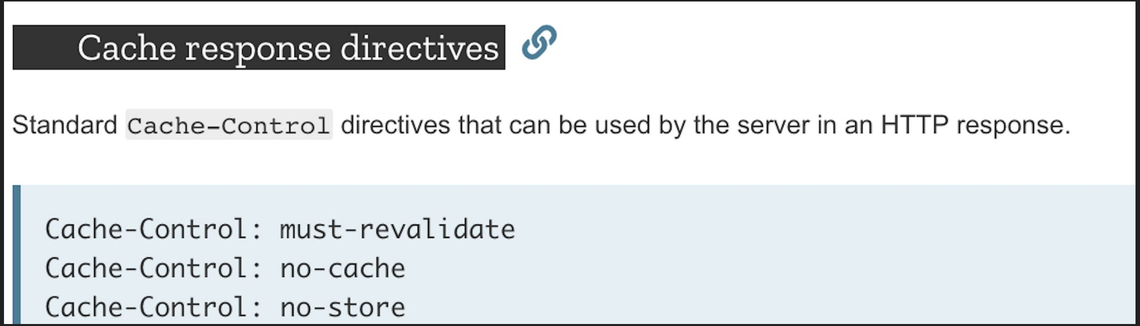
Cache settings



Cache request directives

Standard `Cache-Control` directives that can be used by the client in an HTTP request.

```
Cache-Control: max-age=<seconds>
Cache-Control: max-stale[=<seconds>]
Cache-Control: min-fresh=<seconds>
Cache-Control: no-cache
Cache-Control: no-store
```

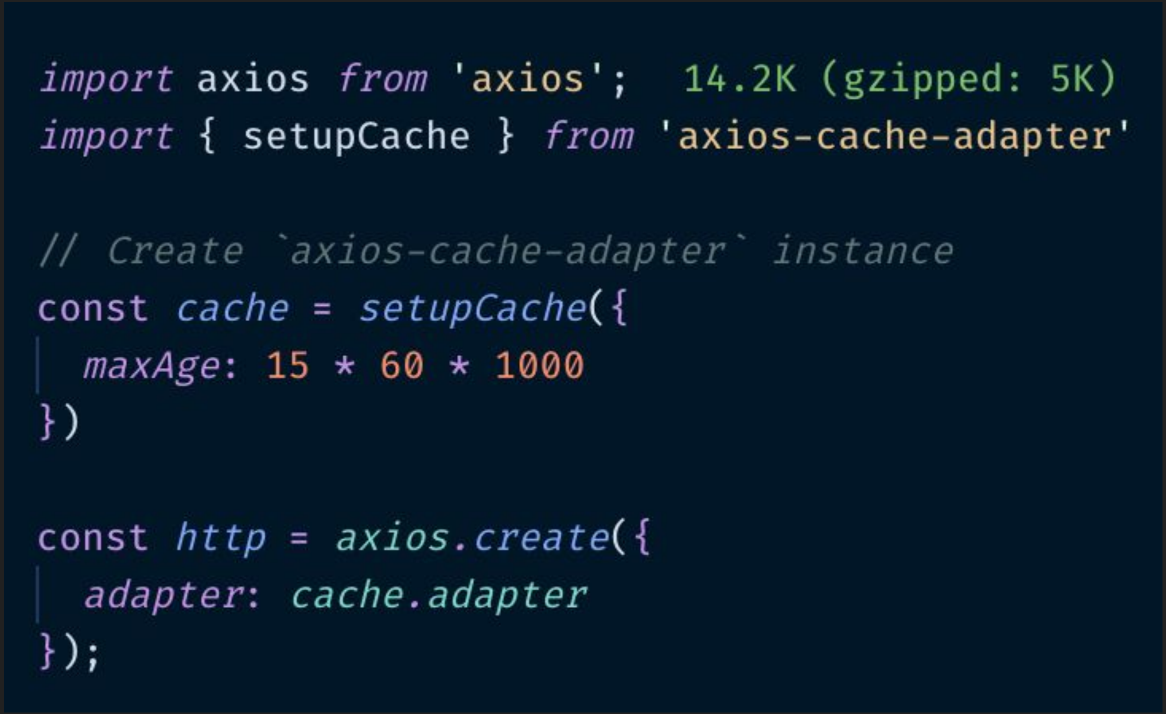


Cache response directives

Standard `Cache-Control` directives that can be used by the server in an HTTP response.

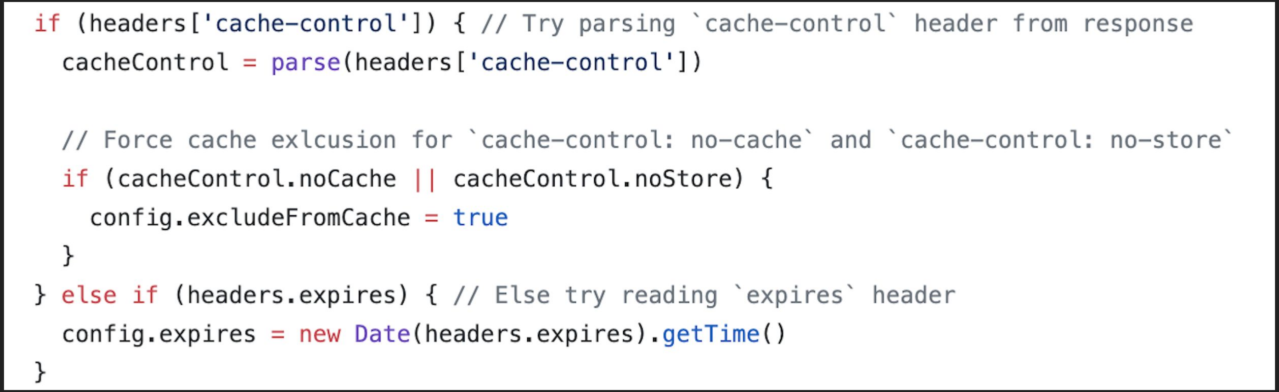
```
Cache-Control: must-revalidate
Cache-Control: no-cache
Cache-Control: no-store
```

axios(client) implementation

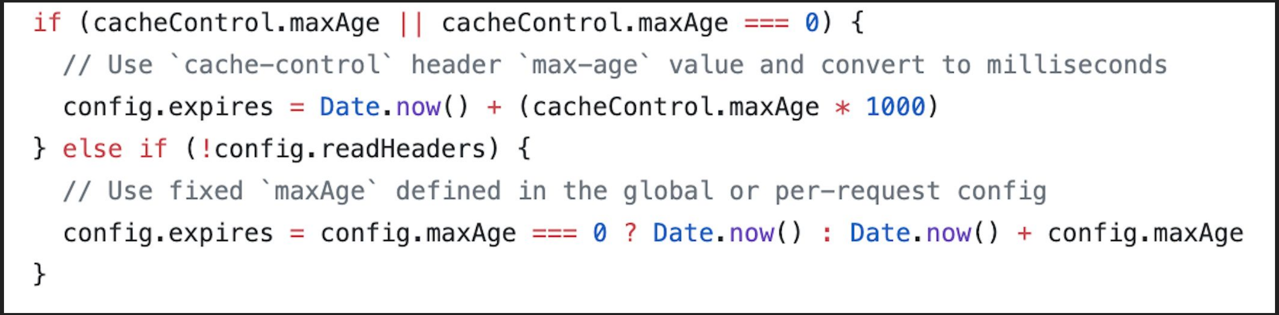


```
import axios from 'axios';  14.2K (gzipped: 5K)
import { setupCache } from 'axios-cache-adapter'

// Create `axios-cache-adapter` instance
const cache = setupCache({
  maxAge: 15 * 60 * 1000
})

const http = axios.create({
  adapter: cache.adapter
});
```

Parsing headers of cache-request directives

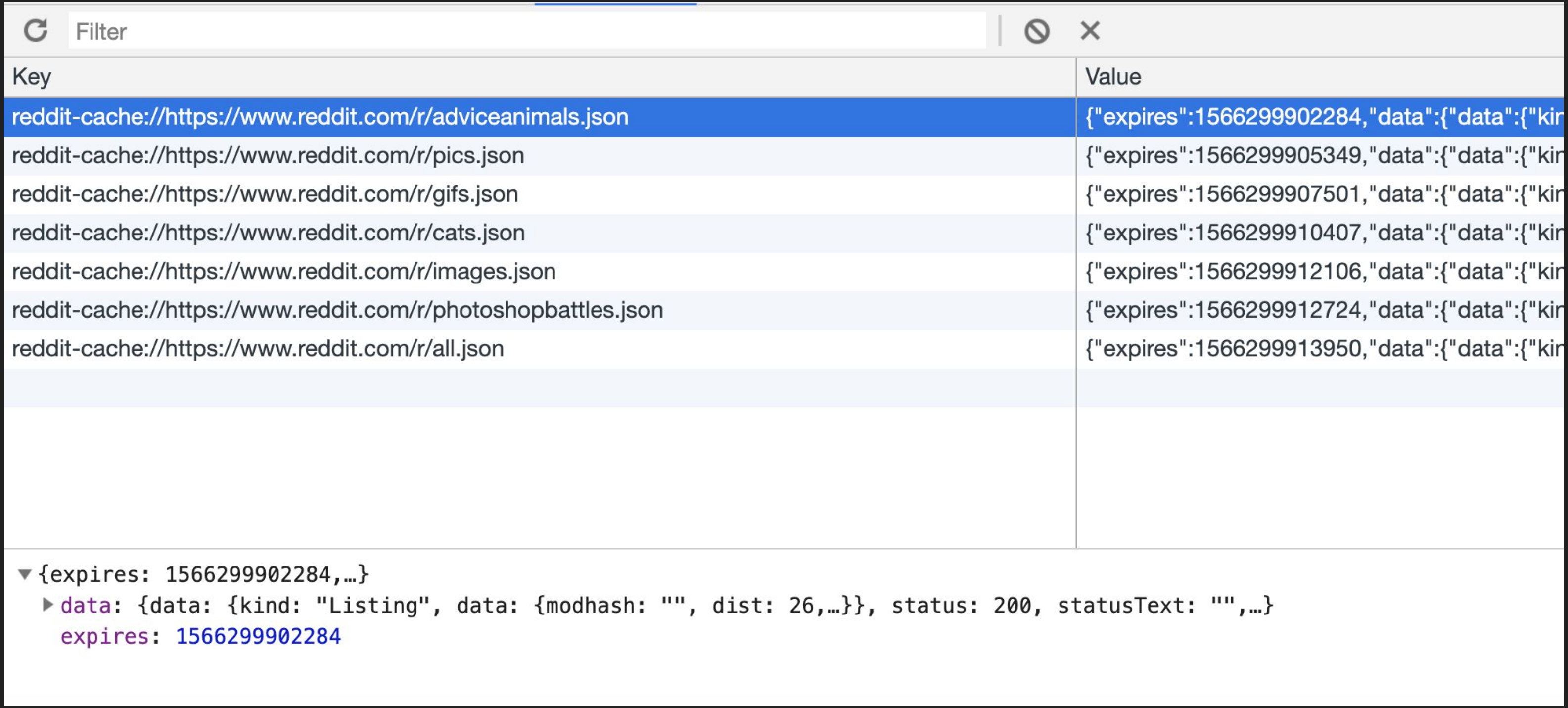


```
if (headers['cache-control']) { // Try parsing `cache-control` header from response
  cacheControl = parse(headers['cache-control'])

  // Force cache exlcusion for `cache-control: no-cache` and `cache-control: no-store`
  if (cacheControl.noCache || cacheControl.noStore) {
    config.excludeFromCache = true
  }
} else if (headers.expires) { // Else try reading `expires` header
  config.expires = new Date(headers.expires).getTime()
}
```



```
if (cacheControl.maxAge || cacheControl.maxAge === 0) {
  // Use `cache-control` header `max-age` value and convert to milliseconds
  config.expires = Date.now() + (cacheControl.maxAge * 1000)
} else if (!config.readHeaders) {
  // Use fixed `maxAge` defined in the global or per-request config
  config.expires = config.maxAge === 0 ? Date.now() : Date.now() + config.maxAge
}
```

axios implementation | result using localstorage as cache layer



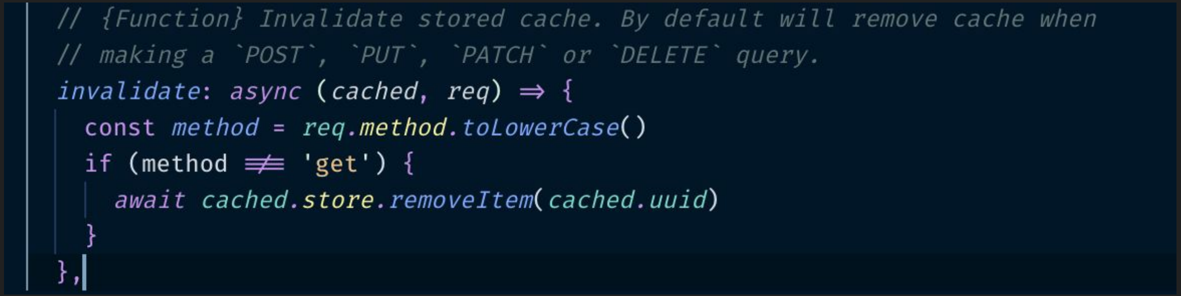
Filter

Key	Value
reddit-cache://https://www.reddit.com/r/adviceanimals.json	{"expires":1566299902284,"data":{"data":{"kir
reddit-cache://https://www.reddit.com/r/pics.json	{"expires":1566299905349,"data":{"data":{"kir
reddit-cache://https://www.reddit.com/r/gifs.json	{"expires":1566299907501,"data":{"data":{"kir
reddit-cache://https://www.reddit.com/r/cats.json	{"expires":1566299910407,"data":{"data":{"kir
reddit-cache://https://www.reddit.com/r/images.json	{"expires":1566299912106,"data":{"data":{"kir
reddit-cache://https://www.reddit.com/r/photoshopbattles.json	{"expires":1566299912724,"data":{"data":{"kir
reddit-cache://https://www.reddit.com/r/all.json	{"expires":1566299913950,"data":{"data":{"kir

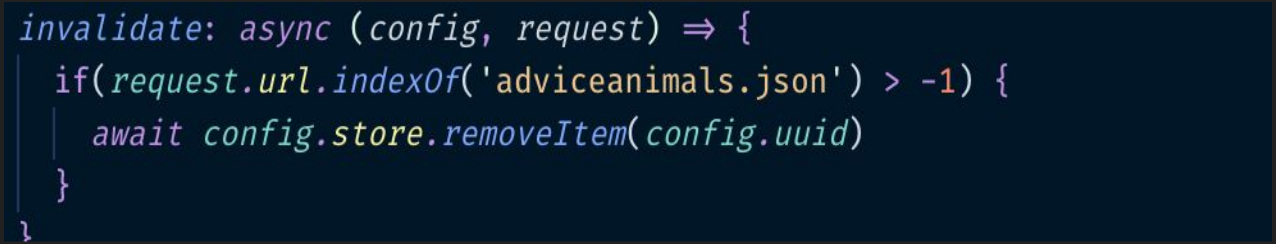
```
▼{expires: 1566299902284,…}
  ▶data: {data: {kind: "Listing", data: {modhash: "", dist: 26,…}}, status: 200, statusText: "",…}
   expires: 1566299902284
```

invalidate cache options

The idea is to have a programmable cache in client. Invalidate a cache based on conditions, whenever a post call goes to a particular api we can remove the cache.



```
// {Function} Invalidate stored cache. By default will remove cache when
// making a `POST`, `PUT`, `PATCH` or `DELETE` query.
invalidate: async (cached, req) => {
  const method = req.method.toLowerCase()
  if (method !== 'get') {
    await cached.store.removeItem(cached.uuid)
  }
},
```

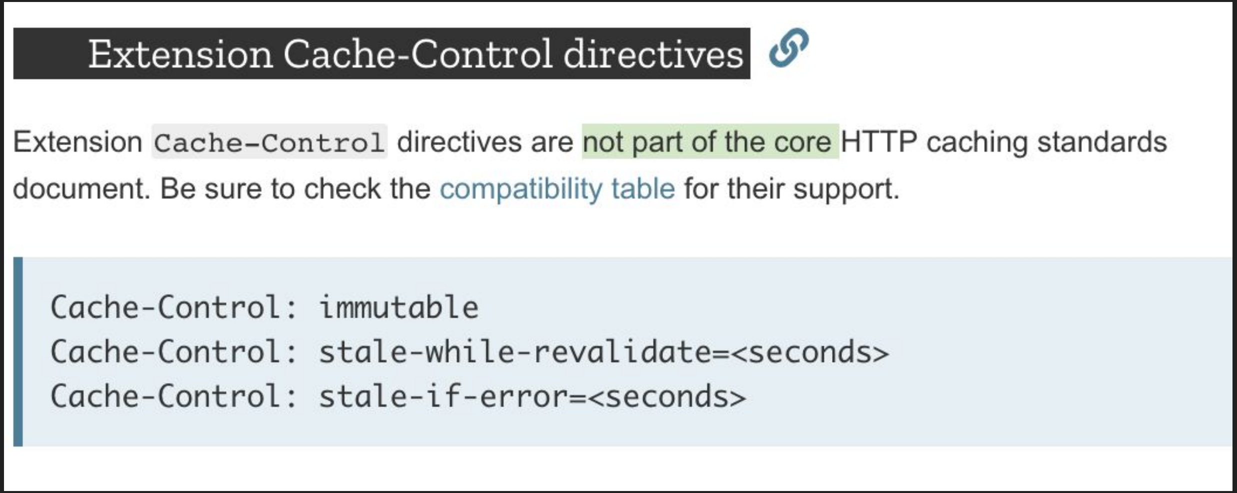


```
invalidate: async (config, request) => {
  if(request.url.indexOf('adviceanimals.json') > -1) {
    await config.store.removeItem(config.uuid)
  }
}
```

stale and revalidate | for content that update frequently

Example:

Cache-Control: max-age=600, stale-while-revalidate=30



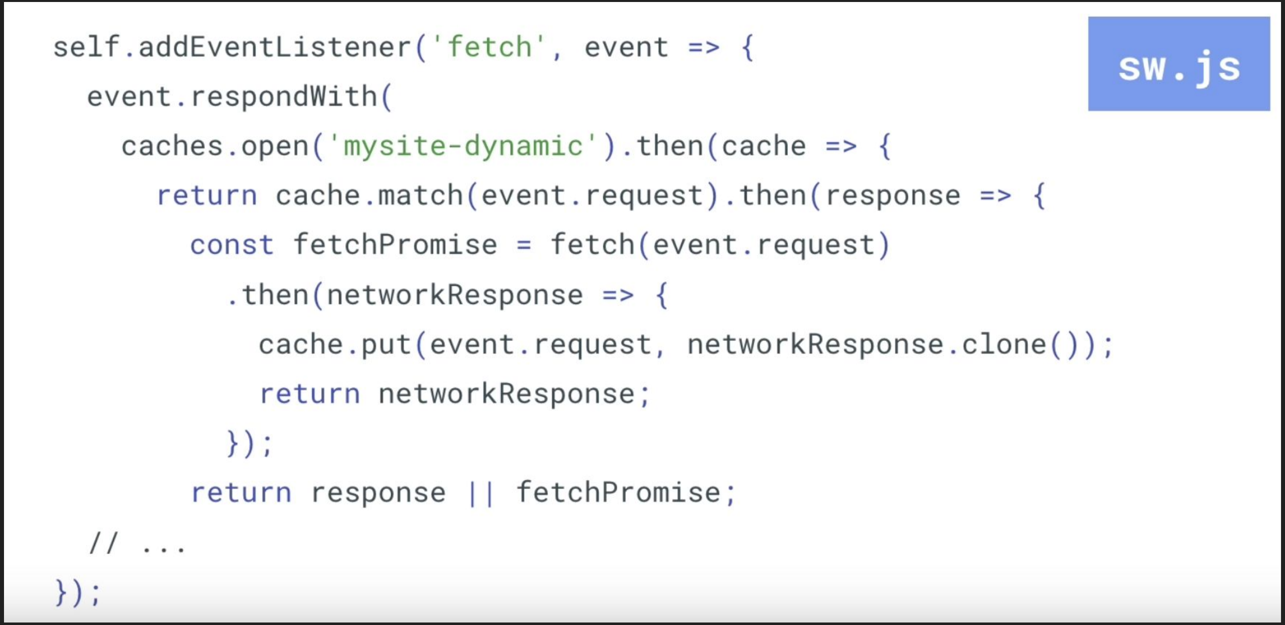
Extension Cache-Control directives

Extension `Cache-Control` directives are not part of the core HTTP caching standards document. Be sure to check the compatibility table for their support.

```
Cache-Control: immutable
Cache-Control: stale-while-revalidate=<seconds>
Cache-Control: stale-if-error=<seconds>
```

Stale while revalidate

Respond to the request as quickly as possible with a cached response if available.



sw.js

```
self.addEventListener('fetch', event => {
  event.respondWith(
    caches.open('mysite-dynamic').then(cache => {
      return cache.match(event.request).then(response => {
        const fetchPromise = fetch(event.request)
          .then(networkResponse => {
            cache.put(event.request, networkResponse.clone());
            return networkResponse;
          });
        return response || fetchPromise;
  // ...
});
```

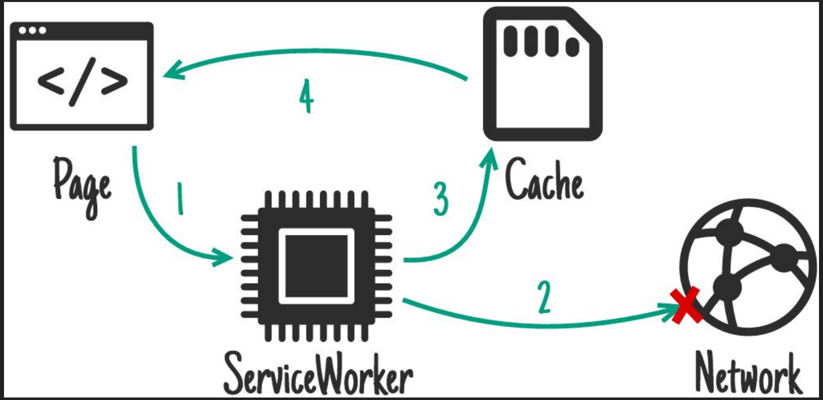
Service Worker | Workbox implementation | PWA

Service Worker is a client side programmable network proxy. It’s a type of web worker.

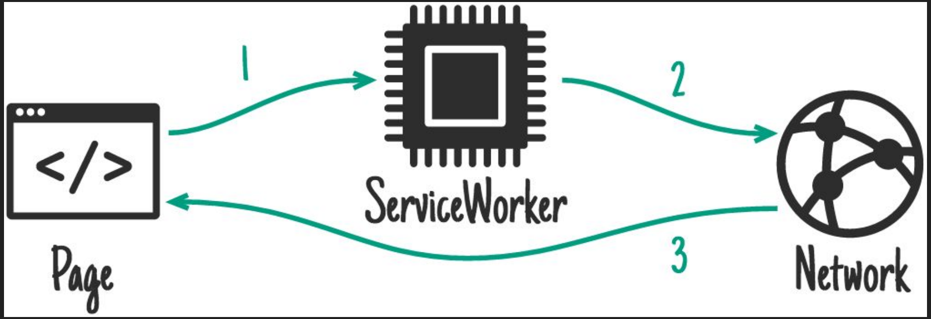
Caching strategies:

- Stale-While-Revalidate *
- Network First (Network Falling Back to Cache) *
- Cache First (Cache Falling Back to Network) *
- Network Only - analytics ping which has no offline equivalent
- Cache Only

Invalidation Strategies

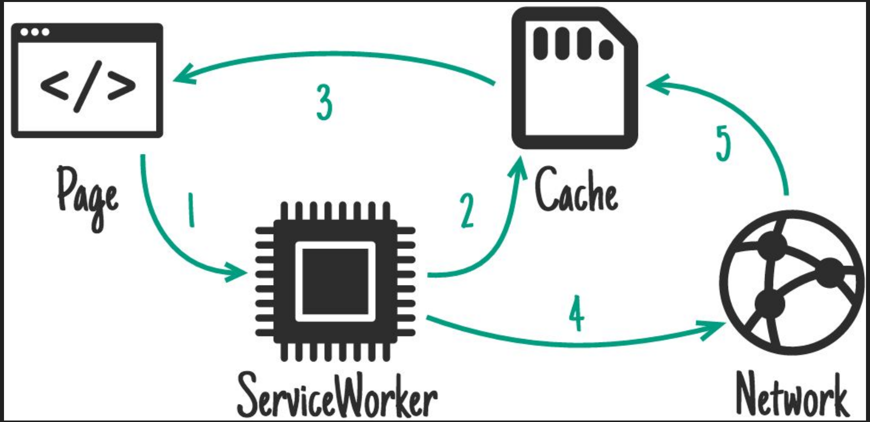


Network first

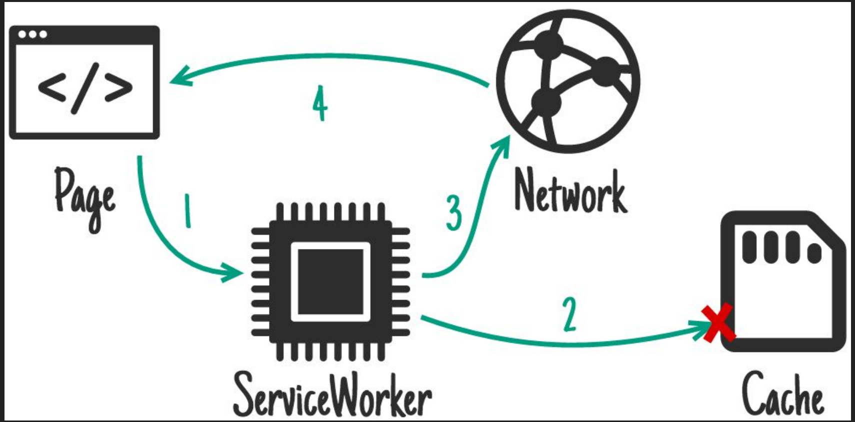


Network only

Invalidation Strategies

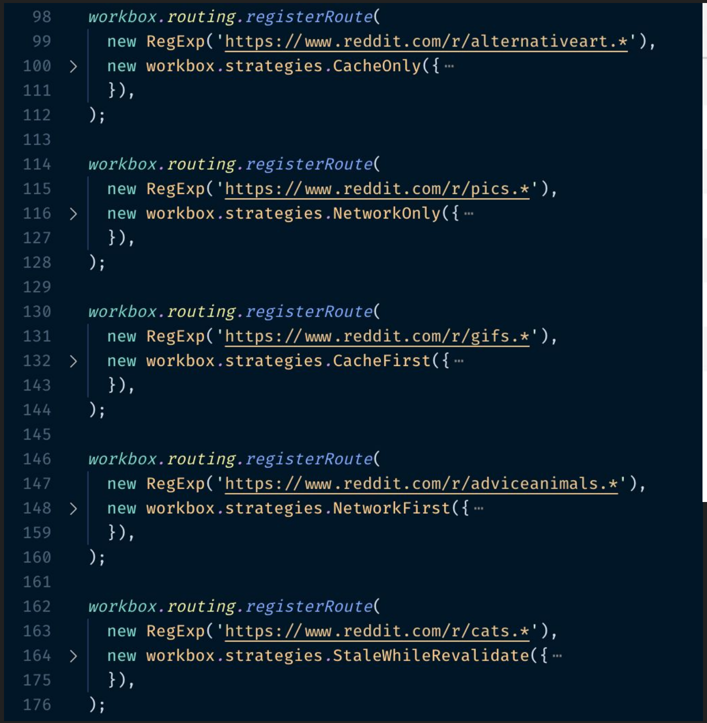


Stale while revalidate

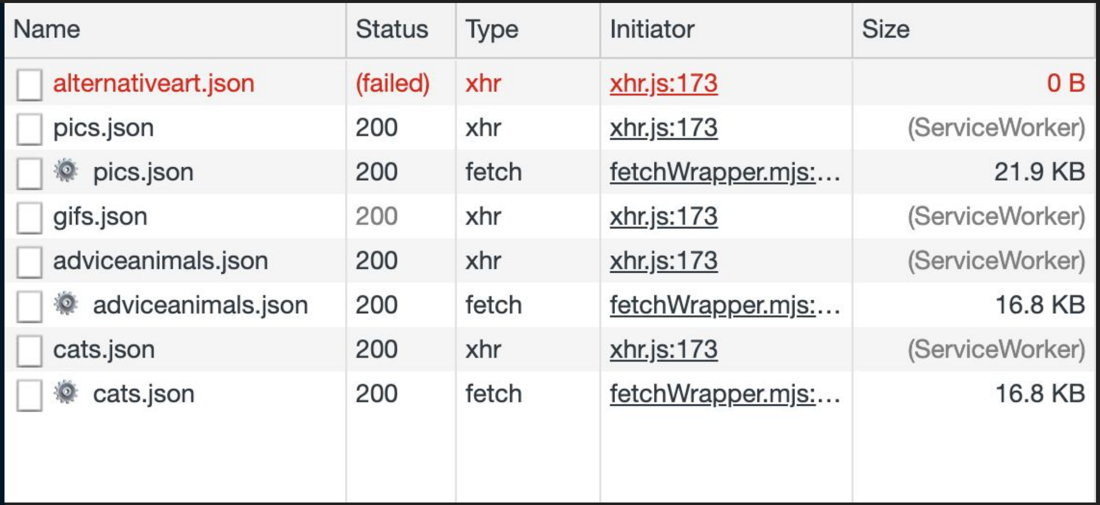


Cache first

Implementation and behaviour | workbox



```
workbox.routing.registerRoute(
  new RegExp('https://www.reddit.com/r/alternativeart.*'),
  new workbox.strategies.CacheOnly({⋯
  }),
);

workbox.routing.registerRoute(
  new RegExp('https://www.reddit.com/r/pics.*'),
  new workbox.strategies.NetworkOnly({⋯
  }),
);

workbox.routing.registerRoute(
  new RegExp('https://www.reddit.com/r/gifs.*'),
  new workbox.strategies.CacheFirst({⋯
  }),
);

workbox.routing.registerRoute(
  new RegExp('https://www.reddit.com/r/adviceanimals.*'),
  new workbox.strategies.NetworkFirst({⋯
  }),
);

workbox.routing.registerRoute(
  new RegExp('https://www.reddit.com/r/cats.*'),
  new workbox.strategies.StaleWhileRevalidate({⋯
  }),
);
```



Name	Status	Type	Initiator	Size
alternativeart.json	(failed)	xhr	xhr.js:173	0 B
pics.json	200	xhr	xhr.js:173	(ServiceWorker)
⚙ pics.json	200	fetch	fetchWrapper.mjs:…	21.9 KB
gifs.json	200	xhr	xhr.js:173	(ServiceWorker)
adviceanimals.json	200	xhr	xhr.js:173	(ServiceWorker)
⚙ adviceanimals.json	200	fetch	fetchWrapper.mjs:…	16.8 KB
cats.json	200	xhr	xhr.js:173	(ServiceWorker)
⚙ cats.json	200	fetch	fetchWrapper.mjs:…	16.8 KB

sw.js

Caching at store layer | redux-persist

- Persist and rehydrate a redux store
- State Reconciler
- Transform: redux-persist-expire
- Storage Engines: AsyncStorage, redux-persist-filesystem-storage, localforage

Demo: https://github.com/arindam-meesho/reddit-beat

Thank you

We are hiring