前端缓存调研

一 动机

在当前项目开发中，会出现许多接口重复调用的情况，产生不必要的资源消耗。组内讨论后，决定在前端添加数据缓存

二 背景

前端常用的缓存分为同步缓存和异步缓存。常用的缓存方式
有 `localStorage`、`sessionStorage`、`cookie`、`indexedDB`、`service Worker`

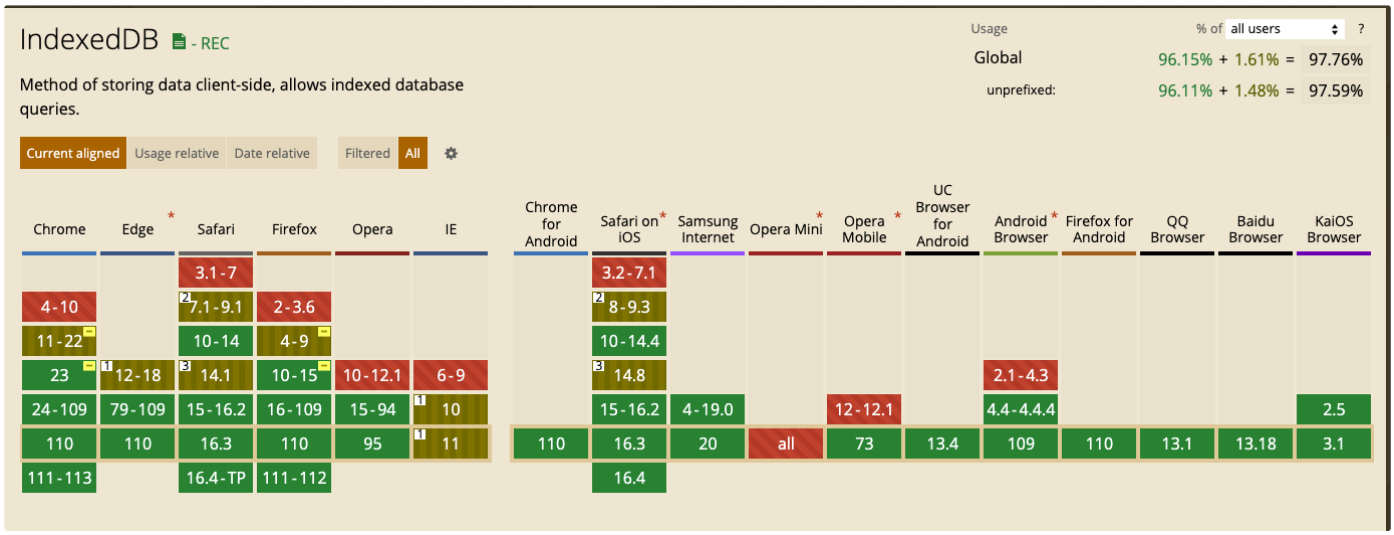
storage缓存

简单的缓存方式有cookie，localStorage、sessionStorage，但最大存储容量只有**5MB**，如果缓存的数据较多时，容易出现内存泄漏

前端数据库

前端数据库有WebSql和IndexDB，其中WebSql被规范废弃，他们都有大约50MB的最大容量，可以理解为localStorage的加强版

indexDB兼容性



indexDB优缺点

优点

1. 异步缓存
2. 存储容量50MB，可以满足接口缓存要求
3. 兼容性良好
4. 可支持存储对象、数组等数据类型
5. 存储数据的时候不需要进行数据序列话

缺点

1. 异步缓存，数据需要在.then中获取
2. 不同浏览器中indexDB版本有差异化

解决方案

1. 使用 localforage 插件，解决indexDB兼容问题，并且支持优雅降级，在不支持indexDB的情况下，会使用localStorage存储

使项目的类型相似，也存在一些细微的差异。

应用缓存

浏览器端常用的应用缓存有 service worker

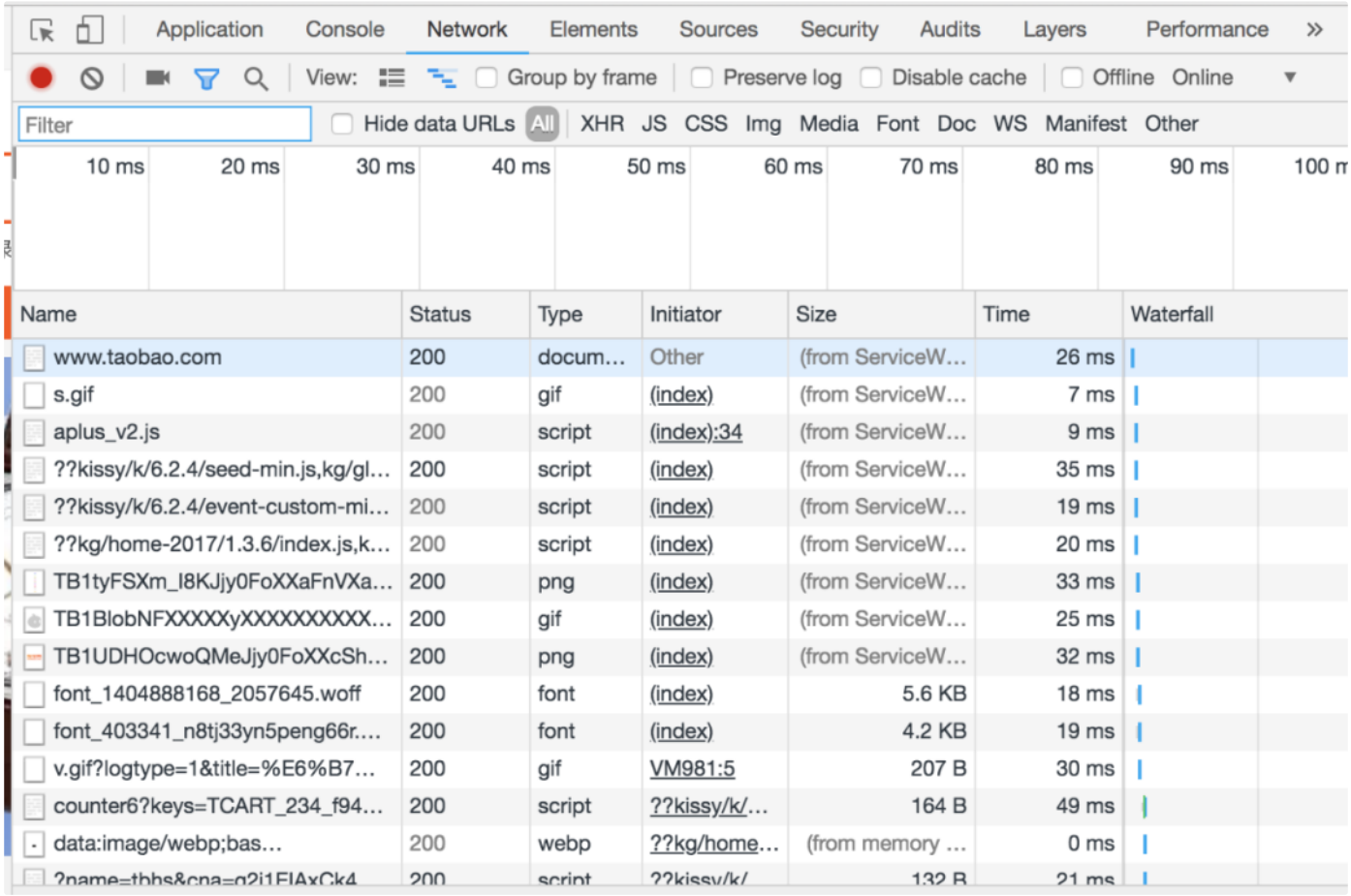
实现原理

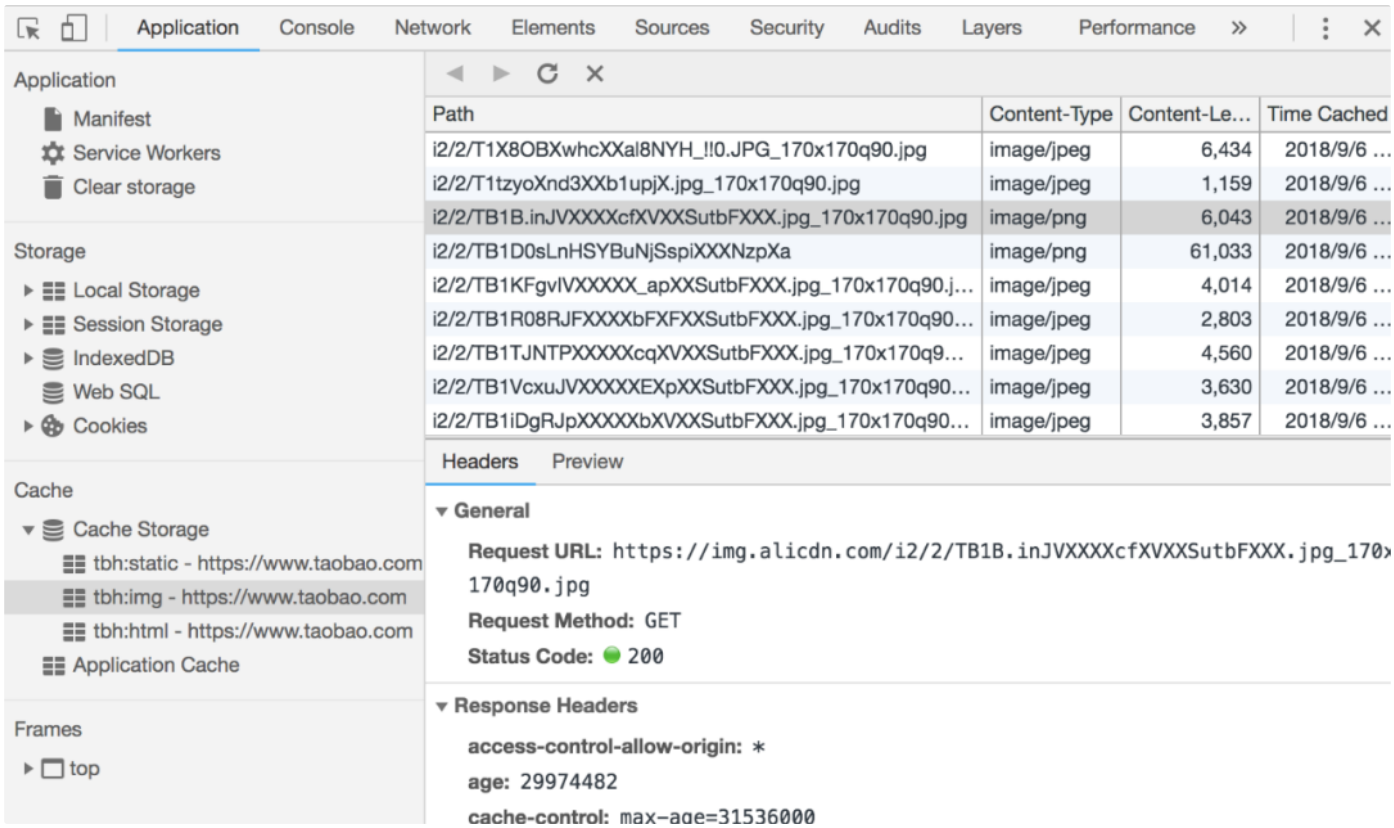
1. 后台线程：独立于当前网页线程
2. 网络代理：在网页发起请求时代理，来缓存文件

兼容性

成熟度

1. 淘宝
2. 网易新闻
3. 考拉


Application
Console
Network
Elements
Sources
Security
Audits
Layers
Performance
View:
Group by frame
Preserve log
Disable cache
Offline
Online
Filter
Hide data URLs
All
XHR
JS
CSS
Img
Media
Font
Doc
WS
Manifest
Other
10 ms
20 ms
30 ms
40 ms
50 ms
60 ms
70 ms
80 ms
90 ms
Name
Status
Type
Initiator
Size
Time
Waterfall
www.taobao.com 200 docum... Other (from ServiceW... 26 ms
s.gif 200 gif (index) (from ServiceW... 7 ms
aplus_v2.js 200 script (index):34 (from ServiceW... 9 ms
??kissy/k/6.2.4/seed-min.js,kg/gl... 200 script (index) (from ServiceW... 35 ms
??kissy/k/6.2.4/event-custom-mi... 200 script (index) (from ServiceW... 19 ms
??kg/home-2017/1.3.6/index.js,k... 200 script (index) (from ServiceW... 20 ms
TB1tyFSXm_I8KJjy0FoXXaFnVXa... 200 png (index) (from ServiceW... 33 ms
TB1BlobNFXXXXXyXXXXXXXXXX... 200 gif (index) (from ServiceW... 25 ms
TB1UDHOcwoQMeJjy0FoXXcSh... 200 png (index) (from ServiceW... 32 ms
font_1404888168_2057645.woff 200 font (index) 5.6 KB 18 ms
font_403341_n8tj33yn5peng66r.... 200 font (index) 4.2 KB 19 ms
v.gif?logtype=1&title=%E6%B7... 200 gif VM981:5 207 B 30 ms
counter6?keys=TCART_234_f94... 200 script ??kissy/k/... 164 B 49 ms
data:image/webp;bas... 200 webp ??kg/home... (from memory ... 0 ms


Application
Console
Network
Elements
Sources
Security
Audits
Layers
Performance
Application
Manifest
Service Workers
Clear storage
Storage
Local Storage
Session Storage
IndexedDB
Web SQL
Cookies
Cache
Cache Storage
tbh:static - https://www.taobao.com
tbh:img - https://www.taobao.com
tbh:html - https://www.taobao.com
Application Cache
Frames
top
Path
Content-Type
Content-Le...
Time Cached
i2/2/T1X8OBXwhcXXal8NYH_!!0.JPG_170x170q90.jpg image/jpeg 6,434 2018/9/6 ...
i2/2/T1tzyoXnd3XXb1upjX.jpg_170x170q90.jpg image/jpeg 1,159 2018/9/6 ...
i2/2/TB1B.inJVXXXXcfXVXXSutbFXXX.jpg_170x170q90.jpg image/png 6,043 2018/9/6 ...
i2/2/TB1D0sLnHSYBuNjSspiXXXNzpXa image/png 61,033 2018/9/6 ...
i2/2/TB1KFgvIVXXXXX_apXXSutbFXXX.jpg_170x170q90.j... image/jpeg 4,014 2018/9/6 ...
i2/2/TB1R08RJFXXXXbFXFXXSutbFXXX.jpg_170x170q90... image/jpeg 2,803 2018/9/6 ...
i2/2/TB1TJNTPXXXXXcqXVXXSutbFXXX.jpg_170x170q9... image/jpeg 4,560 2018/9/6 ...
i2/2/TB1VcxuJVXXXXXEXpXXSutbFXXX.jpg_170x170q90... image/jpeg 3,630 2018/9/6 ...
i2/2/TB1iDgRJpXXXXXbXVXXSutbFXXX.jpg_170x170q90... image/jpeg 3,857 2018/9/6 ...
Headers
Preview
General
Request URL: https://img.alicdn.com/i2/2/TB1B.inJVXXXXcfXVXXSutbFXXX.jpg_170x
170q90.jpg
Request Method: GET
Status Code: 200
Response Headers
access-control-allow-origin: *
age: 29974482
cache-control: max-age=31536000

使用条件

sw 是基于 **HTTPS** 的，因为Service Worker中涉及到请求拦截，所以必须使用HTTPS协议来保障安全。如果是本地调试的话，localhost是可以的。

三 方案选择

方案对比

1. service worker多用于脚本、图片资源等大资源缓存，当前做的前端缓存主要是用于web浏览器接口数据缓存，用 localforage更适合当前的场景
2. service worker需要先进行注册，包含多个生命周期，使用相对复杂， localforage主要使用的api包含三个，setItem, getItem, removeItem

综上，最后选择 localforage进行数据缓存

缓存要求

1. 支持区分不同用户接口缓存
2. 请求某个接口，支持覆盖更新缓存和清除非当前用户的该接口缓存数据
3. 调用接口时，先判断缓存数据，无指定缓存才请求接口

实现思路

封装request方法

二次封装 request请求方法 generateDataCache方法，支持是否缓存，缓存时间操作

```
import request from "@/utils/request"
import {
  getCacheItem,
  generateReqKey,
  setCacheItem,
  getNowTime,
  cleanCacheItem
} from "../commonFuns"

/**
 * 请求拦截
 * @param config
 * @returns {Promise<*>}
 */
export async function requestInterceptor(config) {
  // 开启缓存则保存请求结果
  if (config.cache) {
    let data = await getCacheItem(generateReqKey(config))
    // 记录缓存时间
    let setExpireTime
    if (config.setExpireTime) {
      setExpireTime = config.setExpireTime
    }
```

```
    // 判断缓存数据是否存在 存在的话 是否过期 没过期就返回
    if (data && getNowTime() - data.expire < setExpireTime) {
      return data.data
    }
  }
}

/**
 * 响应拦截
 * @param config
 * @param res
 */
export function responseInterceptor(config, res) {
  // 返回的code === 200 时候才会缓存下来
  if (config && config.cache && res.code === 20000) {
    let data = {expire: getNowTime(), data: res}
    let key = generateReqKey(config)
    setCacheItem(key, data)
    // todo:清除非当前用户的相同接口的数据
    cleanCacheItem(config)
  }
}

/**
 * 生成缓存数据
 * @param {Object} requestConfig 请求数据配置
 * @returns {Promise<unknown>}
 */
export async function generateDataCache(requestConfig) {
  let cacheData = await requestInterceptor(requestConfig)
  // 有缓存数据，且未过期，直接走缓存
  if (cacheData) {
    return cacheData
  }
  let res = await request(requestConfig)
  // 缓存数据
  responseInterceptor(requestConfig, res)
  // 返回接口数据
  return res
}
```

调用缓存方法

```
export function getArticle(data) {
  return generateDataCache({
    url: '/article/list',
    method: 'get',
    params: data,
    cache: true,
    setExpireTime: 300000
  })
}
```

工具类方法

公共类方法抽离

依赖资源

1. localforage插件 [indexDB封装插件]
2. Qs插件 [将对象或者数组序列化成 URL 的格式(a=b&c=d),生成唯一key]

```
# 若localforage安装失败，可以直接引入min.js资源文件
npm i localforage Qs -save
```

四 参考链接

1. service worker使用教程
2. localforage官网