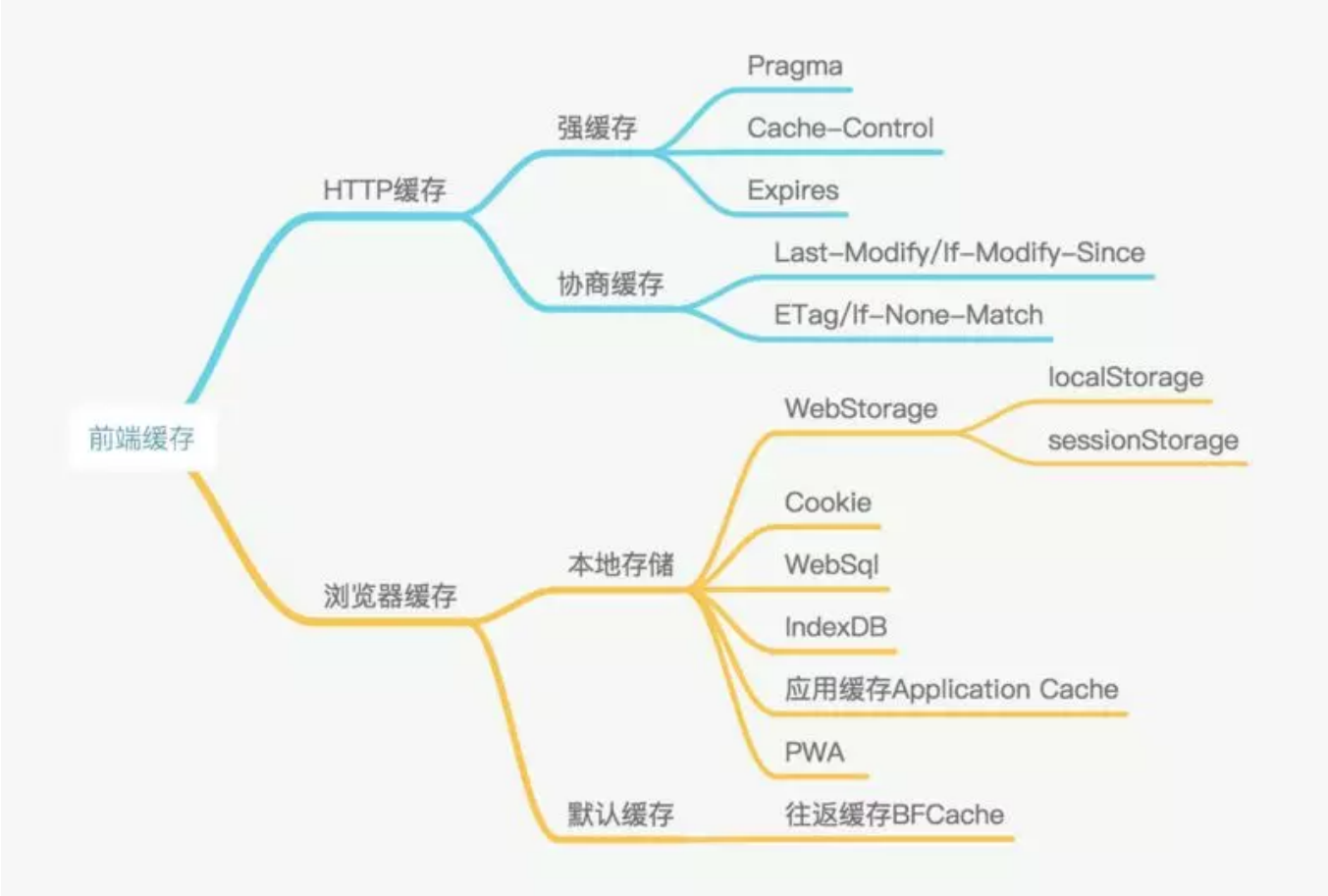
# 一篇文章理解 Web 缓存

JavaScript 今天

> 最近把前端缓存重新整理了一下，从整体的层面上把前端所有能用的缓存方案梳理了一遍。同时，对于http缓存，使用了表格的方案，使得原先晦涩难记的特性变得清晰明了。特记录于此，若有什么欠缺，也望不吝指出。

## 1. 前端缓存概述

前端缓存主要是分为HTTP缓存和浏览器缓存。其中HTTP缓存是在HTTP请求传输时用到的缓存，主要在服务器代码上设置；而浏览器缓存则主要由前端开发在前端js上进行设置。下面会分别具体描述。




## 2. 前端缓存分类

### 2.1 HTTP缓存

> **整体流程**：HTTP缓存都是从**第二次请求**开始的。 第一次请求资源时，服务器返回资源，并在respone header头中回传资源的缓存参数；第二次请求时，浏览器判断这些请求参数，击中强缓存就直接200，否则就把请求参数加到request header头中传给服务器，看是否击中协商缓存，击中则返回304，否则服务器会返回新的资源。

HTTP缓存分为强缓存和协议缓存，它们的区别如下：

> **200 from disk or 200 from memory :** 强缓存的200也有两种情况：200 from disk和200 from memory。现在我没有找到明确的文档来描述这种区别的发生条件。知乎这个问题中提到了一些情景，可以自行取用。

### 2.1.1 强缓存

### 2.1.2 协商缓存

协商缓存都是**成对出现**的。

### 2.1.3 最佳优化策略-消灭304

> 最佳优化策略：因为协商缓存本身也有http请求的损耗，所以最佳优化策略是要尽可能的将静态文件存储为较长的时间，多利用强缓存而不是协商缓存，即消灭304。

但是给文件设置一个很长的Cacha-Control也会带来其他的问题，最主要的问题是静态内容更新时，用户不能及时获得更新的内容。这时候就要**使用hash的方法对文件进行命名**，通过每次更新不同的静态文件名来消除强缓存的影响。

> Hash命名： http://xxx.com/main.5eas34fa.js http://xxx.com/main.js?5eas34fahttp://xxx.com/5eas34fa/main.js

### 2.2 浏览器缓存

### 2.2.1 本地存储小容量

Cookie主要用于用户信息的存储，Cookie的内容可以自动在请求的时候被传递给服务器。 LocalStorage的数据将一直保存在浏览器内，直到用户清除浏览器缓存数据为止。 SessionStorage的其他属性同LocalStorage，只不过它的生命周期同标签页的生命周期，当标签页被关闭时，SessionStorage也会被清除。

| | 容量 | 作用 |
|---|---|---|
| Cookie | 4KB | 请求时传递 |
| LocalStorage | 5MB | 永久存储 |
| SessionStorage | 5MB | 同标签页生命周期，不同页面间交换数据 |

### 2.2.2 本地存储大容量

WebSql和IndexDB主要用在前端有大容量存储需求的页面上，例如，在线编辑浏览器或者网页邮箱。

| | 介绍 | 容量 | 状态 |
| --- | --- | --- | --- |
| WebSql | 关系型数据库 | 不知道，应该也是50M左右 | 被W3C标准废弃 |
| IndexDB | 非关系型数据库 | 50M | 正常使用 |

### 2.2.3 应用缓存与PWA

应用缓存全称为Offline Web Application，它的缓存内容被存在浏览器的Application Cache中。它也是一个**被W3C标准废弃**的功能，主要是通过manifest文件来标注要被缓存的静态文件清单。但是在缓存静态文件的同时，也会默认缓存html文件。这导致页面的更新只能通过manifest文件中的版本号来决定。而且，即使我们更新了version，用户的第一次访问还是会访问到老的页面，只有下一次再访问才能访问到新的页面。所以，应用缓存只适合那种常年不变化的静态网站。如此的不方便，也是被废弃的重要原因。

PWA全称是渐进式网络应用，主要目标是实现web网站的APP式功能和展示。**尽管PWA也有manifest文件，但是与应用缓存却完全不同**。不同于manifest简单的将文件通过是否缓存进行分类，PWA用manifest构建了自己的APP骨架。另外，PWA用Service Worker来控制缓存的使用。这一块的内容较多，在这里就不详细展开了。

### 2.2.4 往返缓存

往返缓存又称为BFCache，是浏览器在前进后退按钮上为了提升历史页面的渲染速度的一种策略。BFCache会缓存所有的DOM结构，但是问题在于，一些页面开始时进行的上报或者请求可能会被影响。这个**问题现在主要会出现在微信h5的开发中**。

去除BFCache有多种方法，但不是本文的重点，想了解的同学可以看《浏览器往返缓存（Back/Forward cache）问题的分析与解决》

## 总结

本文梳理了前端所有可能涉及的缓存，希望能从整体层面建立起系统的缓存知识体系。限于篇幅，每一部分的描述都比较简略，仅起到抛砖引玉之用。如有错误，还望指出。

> 作者：这是你的玩具车吗