atitit.提升性能AppCache

[1.1. 起源 1](#_Toc27523)

[2. 离线存储 1](#_Toc22744)

[3. AppCache 2](#_Toc23623)

[3.1. Manifest文件 2](#_Toc12832)

[3.1.1. CACHE:（必须） 3](#_Toc2258)

[3.2. 页面都离线了，ajax 更新 4](#_Toc20983)

[3.3. 页面的参数如何携带使用js解析 4](#_Toc19722)

[3.4. 离线页面的更新 长尾问题 4](#_Toc23993)

[3.5. 如何更新缓存 6](#_Toc21924)

[3.6. 注意事项 7](#_Toc8428)

## 起源

html5之前的网页，都是无连接，必须联网才能访问，这其实也是web的特色，这其实对于PC是时代问题并不大，但到了移动互联网时代，设备终端位置不再固定，依赖无线信号，网络的可靠性变得降低，比如坐在火车上，过了一个隧道（15分钟），便无法访问网站，这对于web的伤害是很大的，比如对于 《[ecmascript合集](http://yanhaijing.com/es5)》这样的为阅读而生的页面。

html5便引入了cache manifest 文件。那么什么是cache manifest呢，接下来会讲到。

# 离线存储

工作中实际使用的离线缓存有localstorage与Application cache，这两个皆是好东西，一个常用于ajax请求缓存，一个常用于静态资源缓存，这里简单说下我的一些理解

# AppCache

注意： 主页一定会被缓存起来的，因为AppCache主要是用来做离线应用的，如果主页不缓存就无法离线插件了，因此把index.html添加到NETWORK中是不起效果的。

AppCache文件里面儿

 不必明确列出AppCache链接到的页面,因为在默认情况下,任何包含html元素manifest属性的页面都会缓存,这些自动缓存的页面称为主条目,而清单中列出的文件称为详细条目,如果某些文件需要在线访问,,则可以创建一种文件白名单,如果将它们列于NETWORK;之后,这些文件就总可以通过网络加载,这些文件称为网络条目.

使用Application cache可以提升网站载入速度，主要体现在请求传输上，把一些http请求转为本地读取，有效地降低网络延迟，降低http请求，使用简单，还节约流量何乐而不为

另外要做好对manifest文件的管理，由于清单内文件不可访问或manifest更新不及时造成的一些问题。

html新增了一个manifest属性，可以用来指定当前页面的manifest文件

## Manifest文件

接下来详细说说manifest的细节，一个典型的manifest文件代码结构像下面这样：

CACHE MANIFEST#version 1.3CACHE:    test.cssNETWORK:\*

manifest文件，基本格式为三段： CACHE， NETWORK，与 FALLBACK，其中NETWORK和FALLBACK为可选项。

而第一行CACHE MANIFEST为固定格式，必须写在前面。

以#号开头的是注释，一般会在第二行写个版本号，用来在缓存的文件更新时，更改manifest的作用，可以是版本号，时间戳或者md5码等等。

## 自动化工具

manifest文件中的cache部分不能使用通配符，必须手动指定，这实在太让人不可理解，文件一多，就成了体力活了

### CACHE:（必须）

标识出哪些文件需要缓存，可以是相对路径也可以是绝对路径。

a.csshttp://yanhaijing.com/a.css

NETWORK:（可选）

这一部分是要绕过缓存直接读取的文件，可以使用通配符＊。

下面的代码 “login.asp” 永远不会被缓存，且离线时是不可用的：

NETWORK:login.asp

FALLBACK:（可选）

指定了一个后备页面，当资源无法访问时，浏览器会使用该页面。该段落的每条记录都列出两个 URI—第一个表示资源，第二个表示后备页面。两个 URI 都必须使用相对路径并且与清单文件同源。可以使用通配符。

下面的例子中，如果无法建立因特网连接，则用 “404.html” 替代 /html5/ 目录中的所有文件。

FALLBACK:/html5/ /404.html

下面的例子中，则用 “404.html” 替代所有文件。

FALLBACK:\*.html /404.html

## ****页面都离线了，****ajax ****更新****

1. 首先，你可以修改下 manifest 文件来更新这个页面，这个我就不介绍了，但是作为文章内容 页面离线以后，就会存储在本地了，如果你是一篇文章的话，那么这个文章的内容页就被存下来了，你如果以相同的 url 去访问，不管你文章里面的数据更新没有，这个离线下来的页面都不会更新了 ( 除非你更新manifest 文件 ) 。所以，所有的动态数据，你都得用 ajax 方式去获取，就像客户端一样，离线的页面应该是一个没有数据的 空壳，然后通过 ajax 去拉去数据填补这个空壳。然后要注意的是， ajax 的请求地址，要写到 manifest 的 network 中，要不然，你可以试试。

## ****页面的参数如何携带使用js解析****

比如 m.baidu.com/app?a=1&b=2 ，通常我们用一些参数来标记这个页面，通过参数来渲染页面内容，但是 manifest 对于上面的方式，会认为不同的参数表示不同的页面。如果你吧内容页做成一

## 离线页面的更新 长尾问题

网站更新了，如果更新用户本地的离线页面呢？与很多文章中说的一样，先上线你的文件，然后修改一下页面中引入的cache.manifest文件即可，比如修改下注释，修改后，如果再访问页面，就会先去校验manifest 时候有更新，如有更新，再次刷新页面的时候，页面就会更新了。

****A.长尾问题，**** 就像前面说到的一样，如果你的 manifest 文件更新了，你访问页面，需要刷新一次，更新的页面才能 load 进来，那么这样就有一个问题，如果你的后端数据，就是给 js ajax 接口的数据变化了，你对应的 js 也修改了。那么你修改 manifest 上线的时候，第一次开页面，页面就会 bug 了。再刷一次页面，就好了。那么，这个第一次访问的 bug ，是我们不想看到的。而且你不能知道用户什么时候第二次再来访问你的页面，所以你的页面一旦使用 manifest 离线，就像客户端一样，这样就出现了长尾问题。还好， manifest 有一些 js 接口，可以来判断， load 更新文件。

****B.刷新页面****

copy 了 api 一段代码：

1. 1 当前文档对应的 applicationCache 对象
2. window.applicationCache
3. 2 cache.status 属性 , 返回当前离线应用的状态
4. UNCACHED ( 数值 0) ：未启用离线应用
5. IDLE ( 数值 1) ：已开启离线应用，但本地缓存的资源是最新的，并且未标记为废弃资源
6. CHECKING ( 数值 2) ：当前更新缓存的状态为 “ 检查中 ”
7. DOWNLOADING ( 数值 3) ：当前更新缓存的状态为 “ 下载资源中 ”
8. UPDATEREADY ( 数值 4) ：当前更新缓存的状态为 “ 更新完毕 ”
9. OBSOLETE ( 数值 5) ：已开启离线应用，但缓存资源都已标记为废弃

## 如何更新缓存

如下三种方式，可以更新缓存：

更新manifest文件

通过javascript操作

清除浏览器缓存

给manifest添加或删除文件，都可更新缓存，如果我们更改了js，而没有新增或删除，前面例子中注释中的版本号，可以很好的用来更新manifest文件。

html5中引入了js操作离线缓存的方法，下面的js可以手动更新本地缓存。

window.applicationCache.update();

如果用户清除了浏览器缓存（手动或用其他一些工具）都会重新下载文件。

## 注意事项

浏览器对缓存数据的容量限制可能不太一样（某些浏览器设置的限制是每个站点 5MB）。

如果manifest文件，或者内部列举的某一个文件不能正常下载，整个更新过程都将失败，浏览器继续全部使用老的缓存。

引用manifest的html必须与manifest文件同源，在同一个域下。

FALLBACK中的资源必须和manifest文件同源。

当一个资源被缓存后，该浏览器直接请求这个绝对路径也会访问缓存中的资源。

站点中的其他页面即使没有设置manifest属性，请求的资源如果在缓存中也从缓存中访问。

当manifest文件发生改变时，资源请求本身也会触发更新

参考

HTML5 AppCache机制分析 - liqinghua1653的专栏 - 博客频道 - CSDN.NET.html

慎用manifest - 推酷.html