HTML5

Web Applications

Web Storage

目录

一、 Web Applications 3

1.1 Web Applications起源 3

1.2 什么是Cache Manifest 3

1.3 Manifest的特点 3

1.4 浏览器支持情况 4

1.5如何使用 4

1.5.1 Manifest文件例子 4

1.5.2 Manifest基本格式 5

1.6 如何更新缓存 6

1.7 注意事项 6

二、 Web Storage 7

2.1 简介 7

2.2 浏览器支持情况 8

2.3 使用Web Storag 9

2.4 两种Web Storage不同点 10

2.5 Web Storage缺点 11

# Web Applications

## 1.1 Web Applications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。

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

## 1.2 什么是Cache Manifest

首先manifest是一个后缀名为minifest的文件，在文件中定义那些需要缓存的文件，支持manifest的浏览器，会将按照manifest文件的规则，像文件保存在本地，从而在没有网络链接的情况下，也能访问页面。

当我们第一次正确配置app cache后，当我们再次访问该应用时，浏览器会首先检查manifest文件是否有变动，如果有变动就会把相应的变得跟新下来，同时改变浏览器里面的app cache，如果没有变动，就会直接把app cache的资源返回，基本流程是这样的。

## 1.3 Manifest的特点

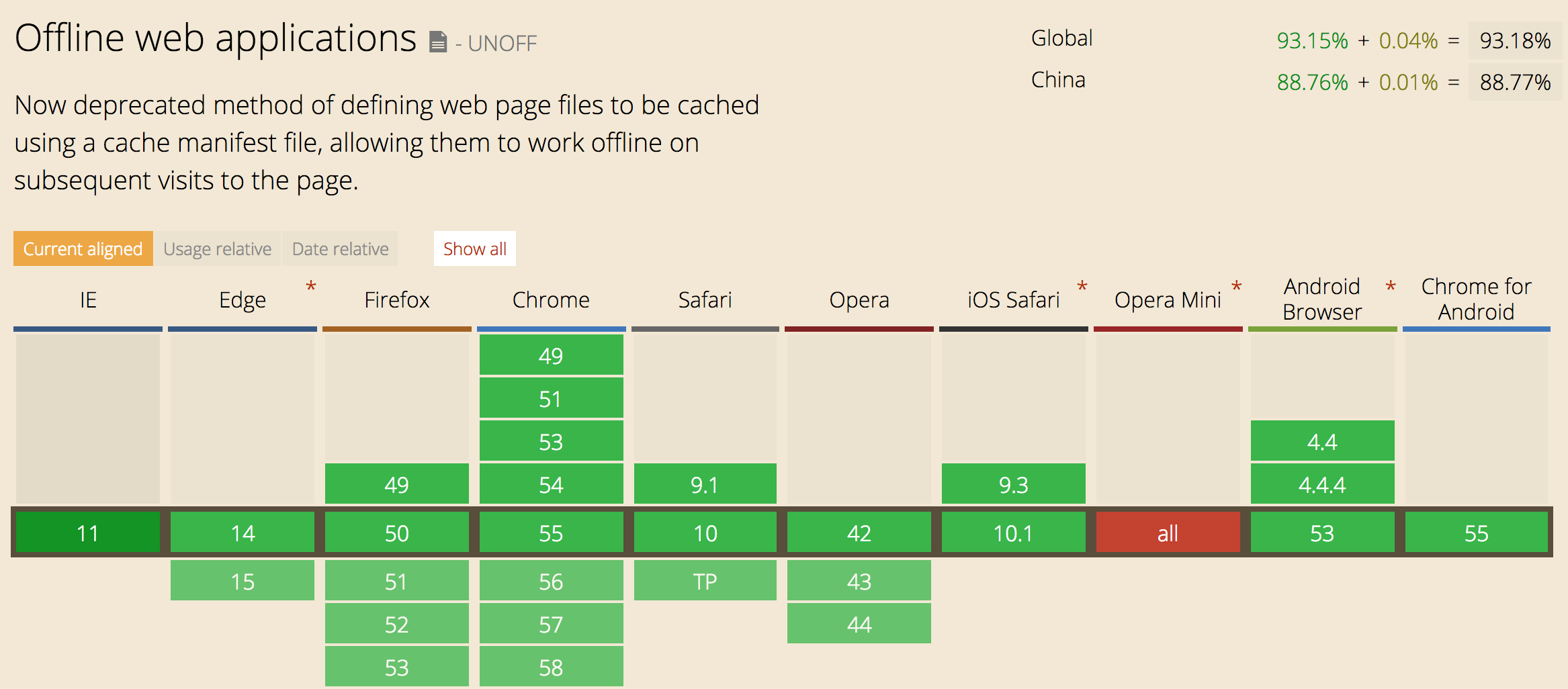
离线浏览: 用户可以在离线状态下浏览网站内容。

更快的速度: 因为数据被存储在本地，所以速度会更快.

减轻服务器的负载: 浏览器只会下载在服务器上发生改变的资源。

## 1.4 浏览器支持情况

所有主流浏览器均支持应用程序缓存，除了 Internet Explorer。caniuse给出的答案如下图所示。



## 1.5如何使用

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

创建一个和html同名的manifest文件，比如页面为index.html，那么可以建一个index.manifest的文件，然后给index.html的html标签添加如下属性即可：

<html lang="en" manifest="index.manifest">

### 1.5.1 Manifest文件例子

一个典型的manifest文件代码结构像下面这样：

CACHE MANIFEST

#version 1.3

CACHE:

test.css

NETWORK:

\*

### 1.5.2 Manifest基本格式

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。

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。

1. css
2. http://www.a.com/123.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

可以使用星号来指示所有其他资源/文件都需要因特网连接：

NETWORK:\*

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

## 1.6 如何更新缓存

如下三种方式，可以更新缓存：

1. 更新manifest文件

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。

1. 通过javascript操作

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();

3.清除浏览器缓存

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。

## 1.7 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。

# Web Storage

## 2.1 简介

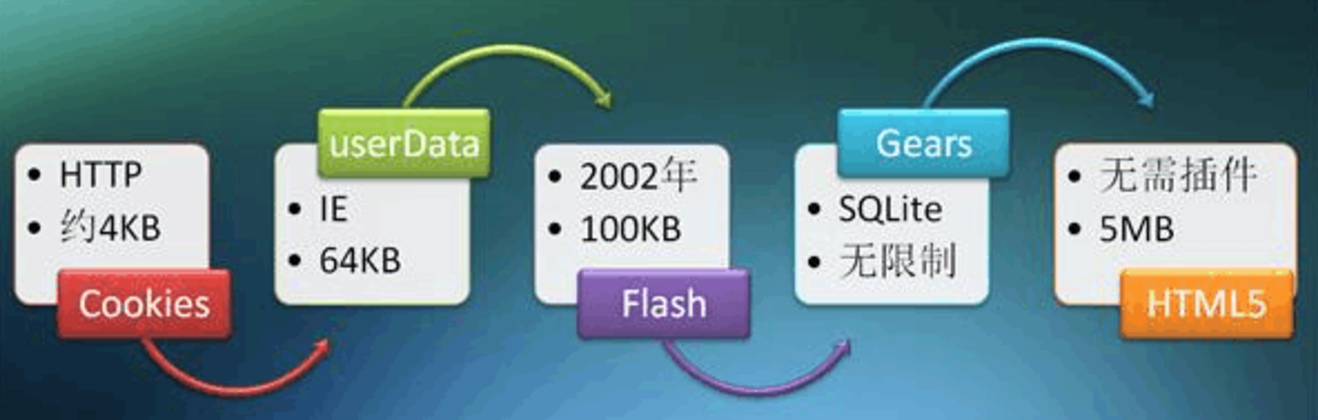
在HTML5本地存储之前，如果我们想在客户端保存持久化数据，有这么几个选择：

1 HTTP cookie。HTTP cookie的缺点很明显，最多只能存储4KB的数据，每个HTTP请求都会被传送回服务器，明文传输（除非你使用SSL）。

2 IE userData。userData是微软在上世纪90年代的浏览器大战时推出的本地存储方案，借助DHTML的behaviour属性来存储本地数据， 允许每个页面最多存储64K数据，每个站点最多640K数据，userData的缺点显而易见，它不是Web标准的一部分，除非你的程序只需要支持IE， 否则它基本没什么用处。

3 Flash cookie。Flash cookie的名字有些误导，它实际上和HTTP cookie并不是一回事，或许它的名字应该叫做"Flash本地存储”，Flash cookie默认允许每个站点存储不超过100K的数据，如果超出了，Flash会自动向用户请求更大的存储空间，借助Flash的 ExternalInterface接口，你可以很轻松地通过Javascript操作Flash的本地存储。Flash的问题很简单，就是因为它是 Flash。

4 Google Gears。Gears是Google在07年发布的一个开源浏览器插件，旨在改进各大浏览器的兼容性，Gears内置了一个基于SQLite的嵌入式 SQL数据库，并提供了统一API对数据库进行访问，在取得用户授权之后，每个站点可以在SQL数据库中存储不限大小的数据，Gears的问题就是 Google自己都已经不用它了。



## 2.2 浏览器支持情况

所有主流浏览器均支持应用程序缓存，除了 Internet Explorer。caniuse给出的答案如下图所示。



## 2.3 使用Web Storag

set方法：

localStorage.setItem("bar", foo);

get方法：

var foo = localStorage.getItem("bar");

快捷方法：

var foo = localStorage["bar"];

localStorage["bar"] = foo;

如果要将某个key从存储空间删除，可以调用removeItem：

localStorage.removeItem('foo');

遍历存储的所有键值对象：

for(var i=0; ivar key = localStorage.key(i);

console.log(key + ":" + localStorage[key]);

}

如果你的程序需要在不同页面访问同一个值，你可能需要了解这个值是否已经被其他页面改变了，这可以通过向浏览器注册storage事件来实现：

代码

1. window.addEventListener('storage', function(e) {

2.     console.log(e.key + "'s value is changed from '" +

3.         e.oldValue + "' to '" + e.newValue + "' by " + e.url);

4. }, false);

5. //A页面

6. localStorage['foo'] = 'bar';

7. //B页面

8. localStorage['foo'] = 'newBar';

window.addEventListener('storage', function(e) {

console.log(e.key + "'s value is changed from '" +

e.oldValue + "' to '" + e.newValue + "' by " + e.url);

}, false);

//A页面

localStorage['foo'] = 'bar';

//B页面

localStorage['foo'] = 'newBar';

这时你应该会在A页面的Console中看到：

foo’s value is changed from ‘bar’ to ‘newbar’ by http://localhost/test.html

要注意的是，storage事件仅仅只是通知你某个键对应的值已经发生了改变，你没有办法在回调中阻止这个改变发生。

## 2.4 两种Web Storage不同点

数据有效期：

sessionstorage仅在当前浏览器窗口关闭前有效

localstorage始终有效，窗口或浏览器关闭也一直保存，删除缓存失效

作用域不同：

sessionstorage 不在不同的浏览器窗口中共享，即使是同一个页面

localstorage 在所有同源窗口中都是共享的

## 2.5 Web Storage缺点

唯一的问题就是它默认的QUOTA只有5MB，并且你没办法通过程序自行或是提示用户来增加存储空间。唯一的办法就是用户自己打开 浏览器的设置，并手动修改QUOTA的大小，如果超出了5MB的限制，你将会遇到一个“QUOTA\_EXCEEDED\_ERR”的错误。