# 【JavaScript】Cookie and Web Storage

这一块自己学习了有一阵子了，但是今天看面试题的时候，让说一下cookie的弊端以及web storage与cookie的区别，竟然还是不知道从何说起，所以，还是要借助大神的技术支持认真的梳理一遍。

支持离线的Web应用开发，是HTML5的一个重点。离线Web应用，就是在设备不能上网的时候依然可以运行的应用。

**开发离线Web应用需要几个步骤：**

1、确保应用知道设备能否上网，以便下一步执行正确的操作。可以使用HTML5定义的navigator.onLine属性来检测。

2、应用必须在离线的时候能够访问一定的资源（图像，js，css等），只有这样才能正常工作。

3、必须有一块本地空间用于保存数据，无论能否上网都不影响读写。

**下面介绍一下数据存储的两种方式：Cookie 和Web Storage**

### 1、Cookie

Cookie的作用我想大家都知道，这个作用就像你去超市购物时，第一次给你办张购物卡，这个购物卡里存放了一些你的个人信息，下次你再来这个连锁超市时，超市会识别你的购物卡，下次直接购物就好了。通俗地说，就是当一个用户通过 HTTP 协议访问一个服务器的时候，这个服务器会将一些 Key/Value 键值对返回给客户端浏览器，并给这些数据加上一些限制条件，在条件符合时这个用户下次访问这个服务器的时候，数据又被完整地带回给服务器。

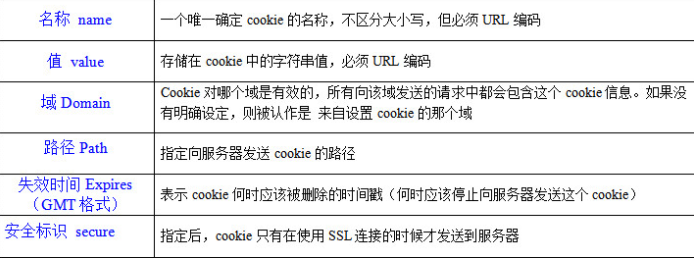
Cookie最初是在客户端用于存储会话信息的，其要求服务器对任意HTTP请求发送Set-CookieHTTP头作为响应的一部分。cookie

以name为名称，以value为值，名和值在传送时都必须是URL编码的。浏览器会存储这样的会话信息，在这之后，通过为每个请求添加Cookie

HTTP头将信息发送回服务器。

Cookie在性质上是绑定在特定的域名下的，当设置了一个cookie后，再给创建它的域名发送请求时，都会包含这个cookie，这确保了储存在cookie中的信息智能让批准的接收者访问，而不能被其他域访问。可以通过document.cookie属性来设置cookie

Cookie的构成



Cookie的操作方法

var CookieUtil = {

get: function (name){

var cookieName = encodeURIComponent(name) + "=",

cookieStart = document.cookie.indexOf(cookieName),

cookieValue = null,

cookieEnd;

if (cookieStart > -1){

cookieEnd = document.cookie.indexOf(";", cookieStart);

if (cookieEnd == -1){

cookieEnd = document.cookie.length;

}

cookieValue = decodeURIComponent(document.cookie.substring(cookieStart + cookieName.length, cookieEnd));

}

return cookieValue;

},

set: function (name, value, expires, path, domain, secure) {

var cookieText = encodeURIComponent(name) + "=" + encodeURIComponent(value);

if (expires instanceof Date) {

cookieText += "; expires=" + expires.toGMTString();

}

if (path) {

cookieText += "; path=" + path;

}

if (domain) {

cookieText += "; domain=" + domain;

}

if (secure) {

cookieText += "; secure";

}

document.cookie = cookieText;

},

unset: function (name, path, domain, secure){

this.set(name, "", new Date(0), path, domain, secure);

}

};

Cookie的优点

可配置到期规则： Cookie 可以在浏览器会话结束时到期，或者可以在客户端计算机上无限期存在，这取决于客户端的到期规则。

不需要任何服务器资源： Cookie 存储在客户端并在发送后由服务器读取。

简单性： Cookie 是一种基于文本的轻量结构，包含简单的键值对。

数据持久性： 虽然客户端计算机上 Cookie 的持续时间取决于客户端上的 Cookie 过期处理和用户干预，但Cookie 通常是客户端上持续时间最长的数据保留形式。

Cookie的弊端

cookie虽然在持久保存客户端数据方面提供了方便，分担了服务器存储的负担，但还是有很多局限性的。

Cookie数量和长度的限制：每个域的cookie总数是有限的，IE6或更低版本最多20个cookie；IE7和之后的版本最后可以有50个；Firefox最多50个；chrome和Safari没有做硬性限制。cookie的长度也有限制，最好将cookie控制在4095B以内。否则会被截掉。

安全性问题： Cookie 把所有要保存的数据通过 HTTP 协议的头部从客户端传递到服务端，又从服务端再传回到客户端，所有的数据都存储在客户端的浏览器里，所以这些 Cookie 数据可以被访问到，如果cookie被人拦截了，那人就可以取得所有的信息。即使加密也与事无补，因为拦截者并不需要知道cookie的意义，他只要原样转发cookie就可以达到目的了。

性能问题：由于所有cookie都会由浏览器作为请求头发送，所以在cookie中存储大量信息会影响到特定域的请求性能

cookie与session的区别

cookie数据存放在客户的浏览器上，session数据放在服务器上。

cookie不是很安全，别人可以分析存放在本地的cookie，并进行cookie欺骗，考虑到安全应当使用session。

session会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能。考虑到减轻服务器性能方面，应当使用cookie。

单个cookie保存的数据不能超过4K，很多浏览器都限制一个站点最多保存20个cookie。

所以个人建议： 将登陆信息等重要信息存放为session, 其他信息如果需要保留，可以放在cookie中

**2、Web Storage**

Web Storage 的目的是克服由cookie带来的一些限制，当数据需要被严格控制在客户端时，无需持续的将数据发回服务器。其主要目的在于：

提供一种在cookie之外存储会话数据的途径；

提供一种存储大量可以跨会话存在的数据的机制。

Storage类型

Storage类型提供最大的存储空间（因浏览器而异）来存储名值对儿。其只能存储字符串，非字符串数据会在存储之前被转换成字符串。

方法：

clear（）;//删除所有值，Firefox未实现；

getItem（name）;//根据指定的name获取对应的值

key(index);//获得index位置处的值的名字

removeItem(name);//删除由name指定的名值对

setItem(name,value);//为指定的name设置value值

sessionStorage对象

存储特定于某个会话的数据，该数据只保持到浏览器关闭。

存储在sessionStorage中的数据可以跨越页面刷新而存在；

sessionStorage对象主要用于仅针对会话的小段数据的存储。所以不详细介绍

globalStorage对象

目的：跨越会话存储数据。要使用globalStorage对象，首先要指定哪些域可以访问该数据，通过方括号标记来实现：

//保存数据

globalStorage["wrox.com"].name = "Vicky";

//获取数据

var name = globalStorage["wrox.com"].name;

在使用globalStorage对象时最好要指定域名，如果事先不能确定域名，那么使用location.host作为属性名比较安全。

如果不使用removeItem（）或者delete删除，或者用户未清除浏览器缓存，存储在globalStorage属性中的数据会一直保存在磁盘上，因此globalStorage很适合在客户端存储文档或者长期保存用户偏好设置。

localStorage对象

localStorage对象是HTML5中取代globalStorage的持久保存客户端数据的方案。要访问同一个localStorage对象，页面必须来自同一个域名，使用那个同一种协议，在同一个端口上。这相当于globalStorage[location.host].

保存在localStorage和globalStorage中的数据一样，数据保留到通过JavaScript删除或者是用户清除浏览器缓存

//客户端数据存储,持久保存客户端数据的方案，适用于长期保存用户偏好设置！

function getLocalStorage () {

if (typeof localStorage == "object") {

return localStorage;

} else if (typeof globalStorage == "object") {

return globalStorage;

} else {

throw new Error ("Local storage not available")

}

}

总结一下：

在较高版本的浏览器中，js提供了sessionStorage和globalStorage。在HTML5中提供了localStorage来取代globalStorage。

html5中的Web Storage包括了两种存储方式：sessionStorage和localStorage。

sessionStorage用于本地存储一个会话（session）中的数据，这些数据只有在同一个会话中的页面才能访问并且当会话结束后数据也随之销毁。因此sessionStorage不是一种持久化的本地存储，仅仅是会话级别的存储。

而localStorage用于持久化的本地存储，除非主动删除数据，否则数据是永远不会过期的。

Web storage和cookie的区别

1、Web Storage的概念和cookie相似，区别是它是为了更大容量存储设计的。Cookie的大小是受限的，并且每次你请求一个新的页面的时候Cookie都会被发送过去，这样无形中浪费了带宽，另外cookie还需要指定作用域，不可以跨域调用。

2、除此之外，Web Storage拥有setItem,getItem,removeItem,clear等方法，不像cookie需要前端开发者自己封装setCookie，getCookie。

3、但是cookie也是不可以或缺的：cookie的作用是与服务器进行交互，作为HTTP规范的一部分而存在 ，而Web Storage仅仅是为了在本地“存储”数据而生。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 姓名 | 学号 | 班级 | 任务 | 权重 |
| 陈若韬 | 3140327 | 软件工程1402 | 总结撰写技术报告 | 1.0 |

项目自评等级:4

项目说明：

1. 翻译GitHub上技术分享者有关WebCookie and Web Storage的技术要点，并加以内化总结，编辑撰写JS技术报告；