# 本地存储

1. Cookie

1.cookie简介

cookie是浏览器提供的一种机制，它将document 对象的cookie属性提供给JavaScript。可以由JavaScript对其进行控制，而并不是JavaScript本身的性质。cookie是存于用户硬盘的一个文件，这个文件通常对应于一个域名，当浏览器再次访问这个域名时，便使这个cookie可用。因此，cookie可以跨越一个域名下的多个网页，但不能跨越多个域名使用。

2.Cookie使用的场合：

1. 保存用户登录状态。例如将用户id存储于一个cookie内，这样当用户下次访问该页面时就不需要重新登录了，现在很多论坛和社区都提供这样的功能。 cookie还可以设置过期时间，当超过时间期限后，cookie就会自动消失。因此，系统往往可以提示用户保持登录状态的时间：常见选项有一个月、三个 月、一年等。
2. 跟踪用户行为。例如一个天气预报网站，能够根据用户选择的地区显示当地的天气情况。如果每次都需要选择所在地是烦琐的，当利用了 cookie后就会显得很人性化了，系统能够记住上一次访问的地区，当下次再打开该页面时，它就会自动显示上次用户所在地区的天气情况。因为一切都是在后 台完成，所以这样的页面就像为某个用户所定制的一样，使用起来非常方便
3. 定制页面。如果网站提供了换肤或更换布局的功能，那么可以使用cookie来记录用户的选项，例如：背景色、分辨率等。当用户下次访问时，仍然可以保存上一次访问的界面风格。
4. 创建购物车。正如在前面的例子中使用cookie来记录用户需要购买的商品一样，在结账的时候可以统一提交。例如淘宝网就使用cookie记录了用户曾经浏览过的商品，方便随时进行比较。
5. cookie的缺点
6. cookie可能被禁用。当用户非常注重个人隐私保护时，他很可能禁用浏览器的cookie功能；
7. cookie是与浏览器相关的。这意味着即使访问的是同一个页面，不同浏览器之间所保存的cookie也是不能互相访问的；
8. cookie可能被删除。因为每个cookie都是硬盘上的一个文件，因此很有可能被用户删除；
9. cookie安全性不够高。所有的cookie都是以纯文本的形式记录于文件中，因此如果要保存用户名密码等信息时，最好事先经过加密处理。
10. cookie的安全性还体现在，每一次http请求都会携带cookie，这使得在网络上存在风险，同时也增加了请求的时间。
11. cookie的大小限制，官网声明cookie仅能存储4k大小的文件。
12. Cookie设置

每个cookie都是一个名/值对，可以把下面这样一个字符串赋值给document.cookie：   
document.cookie="userId=828";

如果要一次存储多个名/值对，可以使用分号加空格（; ）隔开，例如：   
document.cookie="userId=828; userName=hulk";

在cookie 的名或值中不能使用分号（;）、逗号（,）、等号（=）以及空格。在cookie的名中做到这点很容易，但要保存的值是不确定的。如何来存储这些值呢？方 法是用escape()函数进行编码，它能将一些特殊符号使用十六进制表示，例如空格将会编码为“20%”，从而可以存储于cookie值中，而且使用此 种方案还可以避免中文乱码的出现。例如：

document.cookie="str="+escape("I love ajax");

相当于：

document.cookie="str=I%20love%20ajax";

当使用escape()编码后，在取出值以后需要使用unescape()进行解码才能得到原来的cookie值，这在前面已经介绍过。   
  
尽管document.cookie看上去就像一个属性，可以赋不同的值。但它和一般的属性不一样，改变它的赋值并不意味着丢失原来的值，例如连续执行下面两条语句：

document.cookie="userId=828";   
 document.cookie="userName=hulk";

这时浏览器将维护两个cookie，分别是userId和userName，因此给document.cookie赋值更像执行类似这样的语句：

document.addCookie("userId=828");   
 document.addCookie("userName=hulk");

事实上，浏览器就是按照这样的方式来设置cookie的，如果要改变一个cookie的值，只需重新赋值，例如：   
  
document.cookie="userId=929";   
这样就将名为userId的cookie值设置为了929。

1. Cookie获取

下面介绍如何获取cookie的值。cookie的值可以由document.cookie直接获得：   
  
var strCookie=document.cookie;   
这将获得以分号隔开的多个名/值对所组成的字符串，这些名/值对包括了该域名下的所有cookie。例如：

<script language="JavaScript" type="text/javascript">   
 <!--   
 document.cookie="userId=828";   
 document.cookie="userName=hulk";   
 var strCookie=document.cookie;   
 alert(strCookie);   
 //-->   
 </script>

从输出可知，只能够一次获取所有的cookie值，而不能指定cookie名称来获得指定的值，这正是处理cookie值最麻 烦的一部分。用户必须自己分析这个字符串，来获取指定的cookie值，例如，要获取userId的值，可以这样实现：

<script language="JavaScript" type="text/javascript">   
 <!--   
 //设置两个cookie   
 document.cookie="userId=828";   
 document.cookie="userName=hulk";   
 //获取cookie字符串   
 var strCookie=document.cookie;   
 //将多cookie切割为多个名/值对   
 var arrCookie=strCookie.split("; ");   
 var userId;   
 //遍历cookie数组，处理每个cookie对   
 for(var i=0;i<arrCookie.length;i++){   
 var arr=arrCookie[i].split("=");   
 //找到名称为userId的cookie，并返回它的值   
 if("userId"==arr[0]){   
 userId=arr[1];   
 break;   
 }   
 }   
 alert(userId);   
 //-->   
 </script>

这样就得到了单个cookie的值。   
  
用类似的方法，可以获取一个或多个cookie的值，其主要的技巧仍然是字符串和数组的相关操作。

1. Cookie 有效期

到现在为止，所有的cookie都是单会话cookie，即浏览器关闭后这些cookie将会丢失，事实上这些cookie仅仅是存储在内存中，而没有建立相应的硬盘文件。   
  
在实际开发中，cookie常常需要长期保存，例如保存用户登录的状态。这可以用下面的选项来实现：   
  
document.cookie="userId=828; expiress=GMT\_String";   
其中GMT\_String是以GMT格式表示的时间字符串，这条语句就是将userId这个cookie设置为GMT\_String表示的过期时间，超过这个时间，cookie将消失，不可访问。例如：如果要将cookie设置为10天后过期，可以这样实现：

<script language="JavaScript" type="text/javascript">   
 <!--   
 //获取当前时间   
 var date=new Date();   
 var expiresDays=10;   
 //将date设置为10天以后的时间   
 date.setTime(date.getTime()+expiresDays\*24\*3600\*1000);   
 //将userId和userName两个cookie设置为10天后过期   
 document.cookie="userId=828; userName=hulk;

expires="+date.toGMTString();   
 //-->   
</script>

1. 删除cookie

为了删除一个cookie，可以将其过期时间设定为一个过去的时间，例如：

<script language="JavaScript" type="text/javascript">   
 <!--   
 //获取当前时间   
 var date=new Date();   
 //将date设置为过去的时间   
 date.setTime(date.getTime()-10000);   
 //将userId这个cookie删除   
 document.cookie="userId=828; expires="+date.toGMTString();   
 //-->   
 </script>

1. Cookie的封装

**1．添加一个cookie：addCookie(name,value,expiresHours)** 该函数接收3个参数：cookie名称，cookie值，以及在多少小时后过期。这里约定expiresHours为0时不设定过期时间，即当浏览器关闭时cookie自动消失。该函数实现如下：

复制代码代码如下:

<script language="JavaScript" type="text/javascript">   
<!--   
function addCookie(name,value,expiresHours){   
var cookieString=name+"="+escape(value);   
//判断是否设置过期时间   
if(expiresHours>0){   
var date=new Date();   
date.setTime(date.getTime+expiresHours\*3600\*1000);   
cookieString=cookieString+"; expires="+date.toGMTString();   
}   
document.cookie=cookieString;   
}   
//-->   
</script>

**2．获取指定名称的cookie值：getCookie(name)**   
该函数返回名称为name的cookie值，如果不存在则返回空，其实现如下：

复制代码代码如下:

<script language="JavaScript" type="text/javascript">   
<!--   
function getCookie(name){   
var strCookie=document.cookie;   
var arrCookie=strCookie.split("; ");   
for(var i=0;i<arrCookie.length;i++){   
var arr=arrCookie[i].split("=");   
if(arr[0]==name)return arr[1];   
}   
return "";   
}   
//-->   
</script>

**3．删除指定名称的cookie：deleteCookie(name)**   
该函数可以删除指定名称的cookie，其实现如下：

复制代码代码如下:

<script language="JavaScript" type="text/javascript">   
<!--   
function deleteCookie(name){   
var date=new Date();   
date.setTime(date.getTime()-10000);   
document.cookie=name+"=v; expires="+date.toGMTString();   
}   
//-->   
</script>

1. Userdata

**仅支持IE6、7、8，这里不做过多介绍**

1. Localstorage

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。

1、web storage和cookie的区别  
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。  
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。  
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

2、html5 web storage的浏览器支持情况  
浏览器的支持除了IE７及以下不支持外，其他标准浏览器都完全支持(ie及FF需在web服务器里运行)，值得一提的是IE总是办好事，例如IE7、IE6中的UserData其实就是javascript本地存储的解决方案。通过简单的代码封装可以统一到所有的浏览器都支持web storage。  
要判断浏览器是否支持localStorage可以使用下面的代码：

if(window.localStorage){  
 alert("浏览支持localStorage")   
 }  
 else  
 {   
 alert("浏览暂不支持localStorage")   
 }   
 //或者 if(typeof window.localStorage == 'undefined'){ alert("浏览暂不支持localStorage") }

1. localStorage和sessionStorage操作  
   localStorage和sessionStorage都具有相同的操作方法，例如setItem、getItem和removeItem等

localStorage和sessionStorage的方法：  
 setItem存储value  
 用途：将value存储到key字段  
 用法：.setItem( key, value)  
 代码示例：

sessionStorage.setItem("key", "value");  
 localStorage.setItem("site", "js8.in");

getItem获取value  
 用途：获取指定key本地存储的值  
 用法：.getItem(key)  
 代码示例：

var value = sessionStorage.getItem("key");   
 var site = localStorage.getItem("site");

removeItem删除key  
 用途：删除指定key本地存储的值  
 用法：.removeItem(key)  
 代码示例：

sessionStorage.removeItem("key");   
 localStorage.removeItem("site");

clear清除所有的key/value  
 用途：清除所有的key/value  
 用法：.clear()  
 代码示例：

sessionStorage.clear();   
 localStorage.clear();

4、其他操作方法：点操作和[ ]  
web Storage不但可以用自身的setItem,getItem等方便存取，也可以像普通对象一样用点(.)操作符，及[]的方式进行数据存储，像如下的代码:  
var storage = window.localStorage;

storage.key1 = "hello";   
 storage["key2"] = "world";   
 console.log(storage.key1);   
 console.log(storage["key2"]);

5、localStorage和sessionStorage的key和length属性实现遍历  
sessionStorage和localStorage提供的key()和length可以方便的实现存储的数据遍历，例如下面的代码：

var storage = window.localStorage;   
for (var i=0, len = storage.length; i < len; i++)  
{  
 var key = storage.key(i);   
 var value = storage.getItem(key);   
 console.log(key + "=" + value);   
}

6、storage事件  
storage还提供了storage事件，当键值改变或者clear的时候，就可以触发storage事件，如下面的代码就添加了一个storage事件改变的监听：

if(window.addEventListener){   
 window.addEventListener("storage",handle\_storage,false);   
 }  
 else if(window.attachEvent)  
 {   
 window.attachEvent("onstorage",handle\_storage);   
 }   
 function handle\_storage(e){  
 if(!e){e=window.event;}   
 }

storage事件对象的具体属性如下表：

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| key | String | The named key that was added, removed, or moddified |
| oldValue | Any | The previous value(now overwritten), or null if a new item was added |
| newValue | Any | The new value, or null if an item was added |
| url/uri | String | The page that called the method that triggered this change |

1. Appliction cache
2. IndexedDB 和web sql