**第29章 Cookie与存储**

学习要点：

1. cookie
2. cookie局限性
3. 其他存储

随着Web越来越复杂，开发者急切的需要能够本地化存储的脚本功能。这个时候，第一个出现的方案：cookie诞生了。cookie的意图是：在本地的客户端的磁盘上以很小的文件形式保存数据。

1. **Cookie**

cookie也叫HTTP Cookie，最初是客户端与服务器端进行会话使用的。比如，会员登录，下次回访网站时无须登录了；或者是购物车，购买的商品没有及时付款，过两天发现购物车里还有之前的商品列表。

HTTP Cookie要求服务器对任意HTTP请求发送Set-Cookie，因此，Cookie的处理原则上需要在服务器环境下进行。当然，现在大部分浏览器在客户端也能实现Cookie的生成和获取。(目前Chrome不可以在客户端操作，其他浏览器均可)

cookie的组成

cookie由名/值对形式的文本组成：name=value。完整格式为：

name=value; [expires=date]; [path=path]; [domain=somewhere.com]; [secure]

中括号是可选，name=value是必选。

document.cookie = 'user=' + encodeURIComponent('张三'); //编码写入

alert(decodeURIComponent(document.cookie)); //解码读取

expires=date 失效时间，如果没有声明，则为浏览器关闭后即失效。声明了失效时间，那么时间到期后方能失效。

var date = new Date(); //创建一个

date.setDate(date.getDate() + 7);

document.cookie = "user= " + encodeURIComponent('张三') +";expires=" + date;

PS：可以通过Firefox浏览器查看和验证失效时间。如果要提前删除cookie也非常简单，只要重新创建cookie把时间设置当前时间之前即可：date.getDate() - 1或new Date(0)。

path=path 访问路径，当设置了路径，那么只有设置的那个路径文件才可以访问cookie。

var path = '/E:/%E5%A4%87%E8%AF%BE%E7%AC%94%E8%AE%B0/JS1/29/demo';

document.cookie = "user= " + encodeURIComponent('张三') + ";path=" + path;

PS：为了操作方便，我直接把路径复制下来，并且增加了一个目录以强调效果。

domain=domain 访问域名，用于限制只有设置的域名才可以访问，那么没有设置，会默认限制为创建cookie的域名。

var domain = 'pythonblog.com';

document.cookie = "user= " + encodeURIComponent('张三') + ";domain=" + domain;

PS：如果定义了pythonblog.com，那么在这个域名下的任何网页都可访问，如果定义了v.yc60.com，那么只能在这个二级域名访问该cookie，而主域名和其他子域名则不能访问。

PS：设置域名，必须在当前域名绑定的服务器上设置，如果在yc60.com服务器上随意设置其他域名，则会无法创建cookie。

secure 安全设置，指明必须通过安全的通信通道来传输(HTTPS)才能获取cookie。

document.cookie = "user= " + encodeURIComponent('李炎恢') + ";secure";

PS：https安全通信链接需要单独配置。

JavaScript设置、读取和删除并不是特别的直观方便，我们可以封装成函数来方便调用。

//创建cookie

function setCookie(name, value, expires, path, domain, secure) {

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

if (expires instanceof Date) {

cookieText += '; expires=' + expires;

}

if (path) {

cookieText += '; expires=' + expires;

}

if (domain) {

cookieText += '; domain=' + domain;

}

if (secure) {

cookieText += '; secure';

}

document.cookie = cookieText;

}

//获取cookie

function getCookie(name) {

var cookieName = encodeURIComponent(name) + '=';

var cookieStart = document.cookie.indexOf(cookieName);

var cookieValue = null;

if (cookieStart > -1) {

var cookieEnd = document.cookie.indexOf(';', cookieStart);

if (cookieEnd == -1) {

cookieEnd = document.cookie.length;

}

cookieValue = decodeURIComponent(

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

}

return cookieValue;

}

//删除cookie

function unsetCookie(name) {

document.cookie = name + "= ; expires=" + new Date(0);

}

//失效天数，直接传一个天数即可

function setCookieDate(day) {

if (typeof day == 'number' && day > 0) {

var date = new Date();

date.setDate(date.getDate() + day);

} else {

throw new Error('传递的day必须是一个天数，必须比0大');

}

return date;

}

**二．cookie局限性**

cookie虽然在持久保存客户端用户数据提供了方便，分担了服务器存储的负担。但是还有很多局限性的。

第一：每个特定的域名下最多生成20个cookie（根据不同的浏览器有所区别）。

1. IE6或更低版本最多20个cookie
2. IE7和之后的版本最多可以50个cookie。IE7最初也只能20个，之后因被升级不定后增加了。
3. Firefox最多50个cookie
4. Opera最多30个cookie
5. Safari和Chrome没有做硬性限制。

PS：为了更好的兼容性，所以按照最低的要求来，也就是最多不得超过20个cookie。当超过指定的 cookie时，浏览器会清理掉早期的cookie。IE和Opera会清理近期最少使用的cookie，Firefox会随机清理cookie。

第二：cookie的最大大约为4096字节(4k)，为了更好的兼容性，一般不能超过4095字节即可。

第三：cookie存储在客户端的文本文件，所以特别重要和敏感的数据是不建议保存在cookie的。比如银行卡号，用户密码等。

1. **其他存储**

IE提供了一种存储可以持久化用户数据，叫做userData，从IE5.0就开始支持。每个数据最多128K，每个域名下最多1M。这个持久化数据存放在缓存中，如果缓存没有清理，那么会一直存在。

<div style="behavior:url(#default#userData)" id="box"></div>

addEvent(window, 'load', function () {

var box = document.getElementById('box');

box.setAttribute('name', encodeURIComponent('jack'));

box.save('bookinfo');

//box.removeAttribute('name'); //删除userDate

//box.save('bookinfo');

box.load('bookinfo');

alert(decodeURIComponent(box.getAttribute('name')));

});

PS：这个数据文件也是保存在cookie目录中，只要清除cookie即可。如果指定过期日期，则到期后自动删除，如果没有指定就是永久保存。

Web存储

在比较高版本的浏览器，JavaScript提供了sessionStorage和globalStorage。在HTML5中提供了localStorage来取代globalStorage。而浏览器最低版本为：IE8+、Firefox3.5+、Chrome 4+和Opera10.5+。

PS：由于这三个对浏览器版本要求较高，我们就只简单的在Firefox了解一下，有兴趣的可以通过关键字搜索查询。

//通过方法存储和获取

sessionStorage.setItem('name', 'jack');

alert(sessionStorage.getItem('name'));

//通过属性存储和获取

sessionStorage.book = 'jack';

alert(sessionStorage.book);

//删除存储

sessionStorage.removeItem('name');

PS：由于localStorage代替了globalStorage，所以在Firefox、Opera和Chrome目前的最新版本已不支持。

//通过方法存储和获取

localStorage.setItem('name', 'jack');

alert(localStorage.getItem('jack'));

//通过属性存储和获取

localStorage.book = 'jack';

alert(localStorage.book);

//删除存储

localStorage.removeItem('name');

PS：这三个对象都是永久保存的，保存在缓存里，只有手工删除或者清理浏览器缓存方可失效。在容量上也有一些限制，主要看浏览器的差异，Firefox3+、IE8+、Opera为5M，，Chrome和Safari为2.5M。