Day 08---BOM操作

一、BOM概述

> BOM: Browser Object Model 浏览器模型

1.1 什么是BOM

> 浏览器对象模型(BOM)是JavaScript的组成之一，他提供了独立与内容与浏览器窗口进行交互的对象，使用浏览器对象模型可以实现与HTML的交互。
>
> ECMAScript是 JavaScript的核心，但如果要在 Web中使用 JavaScript，那么 BOM（浏览器对象模型）则无疑才是真正的核心。BOM 提供了很多对象，用于访问浏览器的功能，这些功能与任何网页内容无关。

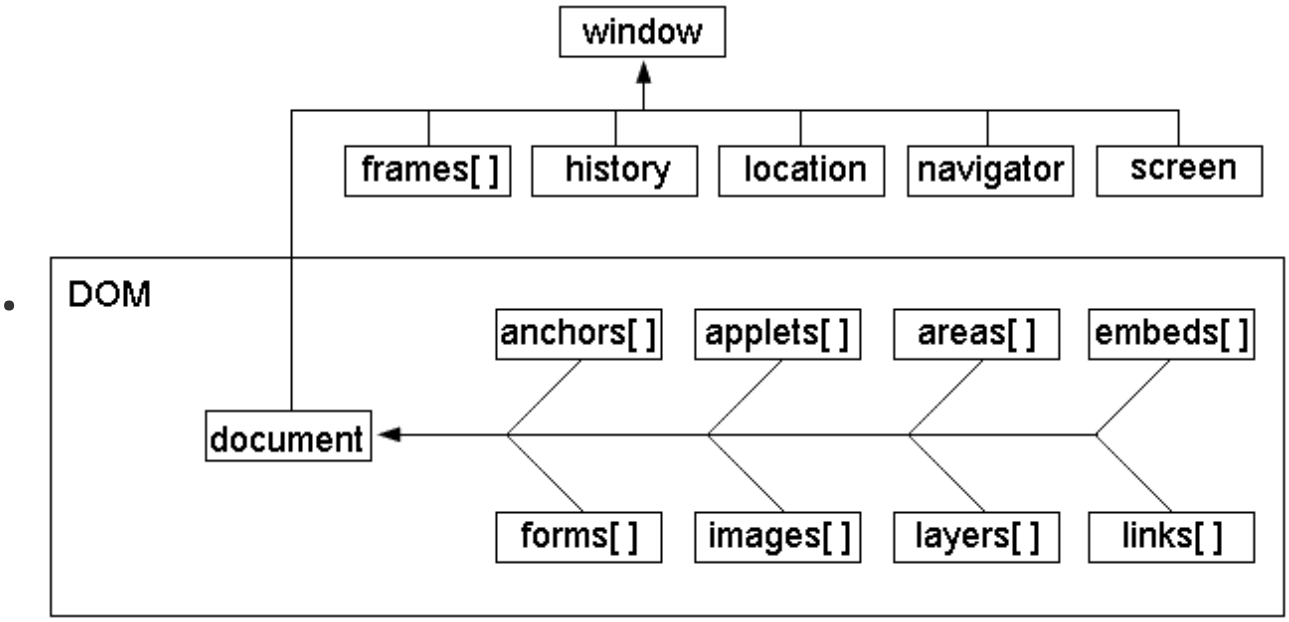
1.2 BOM的作用

> 将相关的元素组织包装起来，提供给程序设计人员使用，从而降低开发人员的工作量，提供设计Web页面的能力.BOM是一个分层结构。

二、BOM核心---window对象

2.1 window对象

> BOM 的核心对象是 window，它表示浏览器的一个实例。在浏览器中， window 对象有双重角色，它既是通过 JavaScript 访问浏览器窗口的一个接口，又是 ECMAScript 规定的 Global 对象。
>
> 通过window可以操作整个浏览器。
>
> BOM图谱：



> window对象是BOM的顶层(核心)对象，所有对象都是通过它延伸出来的，也可以称为window的子对象。

2.1.1 全局作用域

> 由于 window 对象同时扮演着 ECMAScript中 Global 对象的角色，因此所有在全局作用域中声明的变量、函数都会变成 window 对象的属性和方法。

```
<script type="text/javascript">
    var age = 29;    //声明一个全局变量
    function sayAge() { //声明一个全局函数
        alert(this.age);//this表示，调用这个函数时使用的对象。将来是通过window调用的，所以this指代的window
    }
    // 全局变量成为了window对象的属性。 所以可以通过window.age访问
    alert(window.age); //29
    sayAge(); //29
    // 全局函数成为了window对象的方法。 所以可以通过window.sagAge() 来访问。
    window.sayAge(); //29
</script>
//提示：虽然可以通过window对象来调用，但是实际开发的时，一般都省略window。
```

2.1.2 window窗口大小

> A. 有三种方式获取浏览器的窗口大小：(浏览器的视口，不包括工具栏和滚动条)

1. 在Internet Explorer(9+)、Chrome、Firefox、Opera 以及 Safari：
    - window.innerHeight - 浏览器窗口的内部高度
    - window.innerWidth - 浏览器窗口的内部宽度

```
var msg = "窗口宽度：" + window.innerHeight + "\n窗口高度：" + window.innerWidth;
alert(msg );
```

2. 对于 Internet Explorer 8、7、6、5：(Chrome和Firefox也支持)

- document.documentElement.clientHeight
- document.documentElement.clientWidth

```
var msg = "窗口宽度：" + document.documentElement.clientWidth + "\n窗口高度：" +
document.documentElement.clientHeight;
alert(msg);
```

3. 为了兼容浏览器的版本可以使用下面的代码(可以涵盖所有的浏览器)

```
var w = window.innerWidth || document.documentElement.clientWidth || document.body.clientWidth;
var h = window.innerHeight || document.documentElement.clientHeight ||
document.body.clientHeight;
alert("窗口宽度：" + w + "\n窗口高度：" + h);
```

> B. 调整窗口大小：
>
> - window.resizeTo(w, h)：调整到指定的大小
> - resizeBy(deltW, deltH)：增加指定值的窗口的宽和高
>
> 一些浏览器已经禁用了这个方法(Chrome和Firfox均禁用了这两个方法, ie8可用)

```
<body>
    <button onclick="to();">调整到指定大小</button>
    <br/>
    <button onclick="by();">宽高增加10个像素</button>

    <script type="text/javascript">
        function to () {
            alert("我要变小了");
            window.resizeTo(200, 300);
        }
        function by () {
            alert("靠，变大了");
            window.resizeBy(10, 10);
        }
    </script>
</body>
```

2.1.3 打开窗口

> open(URL,name,features) 方法用于打开一个新的浏览器窗口或查找一个已命名的窗口。

參數名称	描述	说　明
URL	一个可选的字符串，声明了要在新窗口中显示的文档的 URL。如果省略了这个参数，或者它的值是空字符串，那么新窗口就不会显示任何文档。	
name	一个可选的字符串，该字符串是一个由逗号分隔的特征列表，其中包括数字、字母和下划线，该字符声明了新窗口的名称。这个名称可以用作标记 和 的属性 target 的值。如果该参数指定了一个已经存在的窗口，那么 open() 方法就不再创建一个新窗口，而只是返回对指定窗口的引用。在这种情况下，features 将被忽略。	
features	一个可选的字符串，声明了新窗口要显示的标准浏览器的特征。如果省略该参数，新窗口将具有所有标准特征。在""窗口特征""这个表格中，我们对该字符串的格式进行了详细的说明。	

备注：如果不用到frame框架，则只需要传入第1个参数和第2个参数即可

窗口特征(常用)：

属性名称	说　明
height、width	窗口文档显示区的高度、宽度。以像素计。
left、top	窗口的x坐标、y坐标。以像素计
toolbar=yes | no |1 | 0	是否显示浏览器的工具栏。默认是yes。
scrollbars=yes | no |1 | 0	是否显示滚动条。默认是yes。
location=yes | no |1 | 0	是否显示地址地段。默认是yes。
status=yes | no |1 | 0	是否添加状态栏。默认是yes。
menubar=yes | no |1 | 0	是否显示菜单栏。默认是yes。
resizable=yes | no |1 | 0	窗口是否可调节尺寸。默认是yes。
titlebar=yes | no |1 | 0	是否显示标题栏。默认是yes。
fullscreen=yes | no |1 | 0	是否使用全屏模式显示浏览器。默认是no。

- 打开新的窗口，窗口特性全部使用默认值

```
<body>
    <button onclick="openAnother();">打开新窗口</button>

    <script type="text/javascript">
    function openAnother () {
        window.open("http://www.yztcedu.com"); //打开新窗口，窗口特性全部使用默认值。
    }
    </script>
</body>
```

- 打开新的窗口，设置窗口特性

```
<body>
    <button onclick="openAnother();">打开新窗口</button>

    <script type="text/javascript">
      var win;
      function openAnother () {
          win = window.open("http://www.yztcedu.com", "", "width=400, height=400, toolbar = no,
menubar = no,left = 200, right = 200");
      }
    </script>
</body>
```

- open方法返回打开的那个窗口的window对象，可以调用close方法关闭新打开的窗口。

```
win.close();
//也可以调整新开窗口的大小位置等，但是大部分浏览器禁用了这个功能。
```

2.1.4 window中的定时器(超时调用和周期调用)(重点)

> window对象提供了两个类似定时器功能的方法：超时调用和周期调用(间隙调用)
>
> JavaScript 是单线程语言，但它允许通过设置超时值和间歇时间值来调度代码在特定的时刻执行。
> 前者是在指定的时间过后执行代码，而后者则是每隔指定的时间就执行一次代码。

1. window.setTimeout()--超时调用

```
/*
    setTimeout(code,millisec)
    参数1：要执行代码。一般传入一个函数。(当然也可是字符串形式的代码，但是不建议使用)
    参数2：多长时间后执行参数1中的代码。   单位毫秒
*/
<script type="text/javascript">
    //传入函数的时，函数名不要加括号。(因为方法不是我们调用，是引擎帮助我们调用)
    // setTimeout方法会返回一个值，表示超时调用的id，可以在任务执行前取消任务。
    var timeOutId = window.setTimeout(go, 3000);  // 3秒中之后执行函数go中的代码
    function go () {
        window.open("http://www.yztcedu.com")
    }
    timeoutId.cancel();  //取消这个超时调用，如果超时调用已经执行完毕，就什么也不会发生。
</script>
```

2. setInterval--周期调用(间隙调用)

```
/*
    setInterval(code,millisec)
    参数1：每隔一段时间执行一次的代码。   一般是一个函数
    参数2：周期性执行的时间间隔。   单位毫秒

*/
<body>
    <h1 id="time"></h1>
    <script type="text/javascript">
        //显示时间，每秒钟变化一次。
        window.setInterval(function() {
            var timeElement = document.getElementById("time");  //找到h1标签
            var msg = new Date().toLocaleString();
            timeElement.innerHTML = msg;     //设置h1标签中的值
        }, 1000);
    </script>
</body>
```

2.1.5 系统对话框

1. alert()对话框

2. confirm()对话框.

方法用于显示一个带有指定消息和 确定 及 取消 按钮的对话框。

说明：如果用户点击确定按钮，则 confirm() 返回 true。如果点击取消按钮，则 confirm() 返回 false。

```
<script type="text/javascript">
    var isOk = window.confirm("你确定要抛弃我吗？");
    //根据用户的不同相动作做出不同响应
    if(isOk){   //用户按下  确定
        window.alert("祝你好运");
    }else { //用户按下  取消
        window.alert("我们就再玩会吧");
    }
</script>
```

3. prompt(text, default) 这是一个提示框，用于提示用户输入一些文本。提示框中除了会有 确定和取消，还有一个可以让用户输入文本的文本框。方法的返回值是用户输入的具体的文本。

- 参数1：提示语句
- 参数2：输入文本框默认已经输入的字符，如果没有默认输入，不用传递
- 返回值：如果用户按下 确定，则返回具体的文本内容，如果按下 取消 返回 null

```
<script type="text/javascript">
    var msg = prompt("请输入你的姓名：");
    if(msg != null && msg.length > 0){
        alert("欢迎：" + msg);
    }
</script>
```

2.2 location对象

属性	描述

> location对象其实是window对象的子对象。可以 这样使用：window.location,但是使用的时候一般省略window。
>
> location 是最有用的 BOM对象之一，它提供了与当前窗口中加载的文档有关的信息，还提供了一些导航功能。
>
> 应用：window.location 对象用于获得当前页面的地址 (URL)，并把浏览器重定向到新的页面。

2.2.1 location对象的常用属性

属性	描述
host	设置或返回主机名和当前 URL 的端口号。
hostname	设置或返回当前 URL 的主机名。
href	设置或返回完整的 URL。
pathname	设置或返回当前 URL 的路径部分。
port	设置或返回当前 URL 的端口号。
protocol	设置或返回当前 URL 的协议。
search	设置或返回从问号 (?) 开始的 URL（查询部分）。

```
<script type="text/javascript">
    //  http://192.168.1.100:8080/JsTest/html/demo2.html?user=lisi&pwd=aaa
    document.write("protocol: " + location.protocol + "<br />");     // 协议
    document.write("hostname: " + location.hostname + "<br />");         //主机名
    document.write("host: " + location.host +  "<br />");  //主机和端口
    document.write("port: " + location.port + "<br />");     //端口
    document.write("pathname: " + location.pathname + "<br />");     // 路径名
    document.write("search: " + location.search + "<br />");     // 查询参数
    document.write("hre: " + location.href + "<br />"); //完整url
</script>
```

- protocol: http:
  hostname: 192.168.1.100
  host: 192.168.1.100:8080
  port: 8080
  pathname: /JsTest/html/demo2.html
  search: ?user=lisi&pwd=aaa
  hre: http://192.168.1.100:8080/JsTest/html/demo2.html?user=lisi&pwd=aaa

2.2.2 location常用方法

属性	描述
assign(url)	加载新的文档。
reload(reforce)	重新加载当前文档。
replace(newURL)	用新的文档替换当前文档。

1. assign(newUrl): 加载新的文档。

```
<button onclick="openNewUrl();">打开育知同创首页</button>
<script type="text/javascript">
    function openNewUrl () {
        location.assign("http://www.yztcedu.com");
    }
</script>
```

2. reload(reforce): 方法不会在 History 对象中生成一个新的记录。当使用该方法时，新的 URL 将覆盖 History 对象中的当前记录。按下回退键的不会返回到刚才的网页。

```
<button onclick="replaceDoc();">替换当前网页</button>
<script type="text/javascript">
    function replaceDoc () {
        location.replace("http://www.yztcedu.com");
    }
</script>
```

3. reload([force]) 方法用于重新加载当前文档。
    - 参数force可选，可以为true。如果该方法没有规定参数，或者参数是 false，它就会用 HTTP 头 If-Modified-Since 来检测服务器上的文档是否已改变。如果文档已改变，reload() 会再次下载该文档。如果文档未改变，则该方法将从缓存中装载文档。如果把该方法的参数设置为true，那么无论文档的最后修改日期是什么，它都会绕过缓存，从服务器上重新下载该文档。

    总结：有参为true时，从服务器重新下载，无参从缓存中重新加载

    注意：reload()调用之后其后的代码有可能执行有可能不执行，所以一般放在最后

```
<button onclick="reloadCurrrent();">重写加载当前网页</button>
<script type="text/javascript">
    function reloadCurrrent () {
        location.reload(true);
    }
</script>  lei
```

2.3 history对象(了解)

History 对象包含用户（在浏览器窗口中）访问过的 URL。

History 对象是 window 对象的一部分，可通过 window.history 属性对其进行访问。

2.3.1 常用属性

属性	描述	描述
length	返回浏览器历史列表中的 URL 数量。(可以点击多少次回退)	

```
<button onclick="urlCount();">历史列表中url数量</button>
<script type="text/javascript">
    function urlCount () {
        alert("历史列表中url数量：" + history.length);
    }
</script>
```

2.3.2 常用方法

方法	描述
back()	加载 history 列表中的前一个 URL。
forward()	加载 history 列表中的下一个 URL。
go()	加载 history 列表中的某个具体页面。

```
<button onclick="myForward();">向前</button>
<button onclick="myBack();">向后</button>
<button onclick="myGo(-2);">后退2个页面</button>
<script type="text/javascript">
    function myForward () {
        history.forward();
    }
    function myBack () {
        history.back();
    }
    function myGo (index) { // 负代表后退， 正代表前进
        history.go(index);
    }
</script>
```

2.4 navigator对象(了解)

> Navigator 对象包含有关浏览器的信息。
>
> Navigator 对象包含的属性描述了正在使用的浏览器。可以使用这些属性进行平台专用的配置。

属性	说　明　　说　明
appCodeName	浏览器代码名的字符串表示
appName	官方浏览器名的字符串表示
appVersion	浏览器版本信息的字符串表示
systemLanguage	操作系统的语音
language	浏览器的主语言
onLine	是否联了因特网
cookieEnabled	如果启用cookie返回true，否则返回false
platform	浏览器所在计算机平台的字符串表示
plugins	安装在浏览器中的插件数组
userAgent	用户代理头的字符串表示

```
<script type="text/javascript">
    document.write("appCodeName: " + navigator.appCodeName + "<br/>");
    document.write("appName: " + navigator.appName + "<br/>");
    document.write("appVersion: " + navigator.appVersion + "<br/>");
    document.write("systemLanguage : " + navigator.systemLanguage + "<br/>");
    document.write("language: " + navigator.language  + "<br/>");
    document.write("onLine: " + navigator.onLine     + "<br/>");
    document.write("cookieEnabled : " + navigator.cookieEnabled     + "<br/>");
    document.write("platform: " + navigator.platform + "<br/>");
    document.write("userAgent: " + navigator.userAgent     + "<br/>");
    document.write("plugins: " + navigator.plugins     + "<br/>");
</script>
```

2.5 screen对象(了解)

> screen对象：包含有关用户屏幕的信息
>
> 在JavaScript中用处不大，只是用来表面客户端的能力，其中包括浏览器窗口外部的显示器的信息，如像素宽度和高度等，各浏览器的screen对象都包含不相同的属性，罗列一些共同的.

属性	说　明
height	屏幕的像素高度
width	屏幕的像素宽度
colorDepth	用于表现颜色的位数
availWidth	屏幕的像素宽度减系统部件宽度之后的值(减去任务栏之后的宽度)
availHeight	部屏幕的像素高度减系统件的高度之后的值（减去任务栏之后的高度）

```
<script type="text/javascript">
    document.write("height : " + screen.height + "<br/>");
    document.write("width : " + screen.width + "<br/>");
    document.write("colorDepth : " + screen.colorDepth + "<br/>");
    document.write("availWidth : " + screen.availWidth + "<br/>");
    document.write("availHeight " + screen.availHeight + "<br/>");
</script>
```

- ```
  height : 768
  width : 1366
  colorDepth : 24
  availWidth : 1366
  availHeight 738
  ```

三、window中常用事件

3.1 onload---加载事件

> 当整个页面加载完成的时会触发该事件。
>
> 语法：window.onload = function(){} 或者 window.onload = 方法名
>
> 注意：一般把事件注册的代码放在head中。

```
<script type="text/javascript">
    window.onload = function () {
        alert("页面加载完毕");
    }
</script>
```

3.2 onscroll---滚动事件

> 当窗口发生滚动会触发该事件
>
> 语法：window.onscroll = function(){} 或者 window.onscroll = 方法名;

```
<script type="text/javascript">
    window.onscroll = function() {
        console.log("开始滚动...")
        //获取滚动距离。
        //document.body.scrollTop:火狐和ie不支持， document.documentElement.scrollTopchrome不支持
        //这样就可以跨浏览器了。
        console.log(document.documentElement.scrollTop | document.body.scrollTop);
    }
</script>
```
```

四、使用JavaScript操作cookie

4.1 什么是cookie

Cookie，有时也用其复数形式 Cookies，指某些网站为了辨别用户身份、进行 session 跟踪而储存在用户本地终端上的数据（通常经过加密）。

Cookie 是在 HTTP 协议下，服务器或脚本可以维护客户工作站上信息的一种方式。Cookie 是由 Web 服务器保存在用户浏览器（客户端）上的小文本文件，它可以包含有关用户的信息。无论何时用户链接到服务器，Web 站点都可以访问 Cookie 信息。

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。 而且不同的浏览器之间cookie不能共享。

> cookie的本质就是用键值对存储在用户本地的一些数据，这些数据不同的网站，不同的浏览器是不能共享的

4.2 cookie的用处

cookie机制将信息存储于用户硬盘，因此可以作为全局变量，这是它最大的一个优点。它可以用于以下几种场合。

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。

4.3 cookie的缺陷

1. 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；
2. 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；
3. 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；
4. 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。

4.4 cookie的构成

> cookie 由浏览器保存的以下几块信息构成。

1. 名称：一个唯一确定 cookie 的名称。cookie 名称是不区分大小写的，所以 myCookie 和 MyCookie被认为是同一个 cookie。然而，实践中最好将 cookie 名称看作是区分大小写的，因为某些服务器会这样处理 cookie。cookie 的名称必须是经过 URL 编码的。
2. 值：储存在 cookie 中的字符串值。值必须被 URL 编码。
3. 域：cookie 对于哪个域是有效的。所有向该域发送的请求中都会包含这个 cookie 信息。这个值可以包含子域（subdomain，如 www.wrox.com ），也可以不包含它（如. wrox.com ，则对于wrox.com的所有子域都有效）。如果没有明确设定，那么这个域会被认作来自设置 cookie 的那个域。
4. 路径：对于指定域中的那个路径，应该向服务器发送 cookie。例如，你可以指定 cookie 只有从 http://www.wrox.com/books/ 中才能访问，那么 http://www.wrox.com 的页面就不会发送 cookie 信息，即使请求都是来自同一个域的。
5. 失效时间：表示 cookie 何时应该被删除的时间戳（也就是，何时应该停止向服务器发送这个cookie）。默认情况下，浏览器会话结束时即将所有 cookie 删除；不过也可以自己设置删除时间。这个值是个 GMT 格式的日期（Wdy, DD-Mon-YYYY HH:MM:SS GMT），用于指定应该删除cookie 的准确时间。因此，cookie 可在浏览器关闭后依然保存在用户的机器上。如果你设置的失效日期是个以前的时间，则 cookie 会被立刻删除。
6. 安全标志：指定后，cookie 只有在使用 SSL 连接的时候才发送到服务器。例如，cookie 信息只能发送给 https://www.wrox.com ，而 http://www.wrox.com 的请求则不能发送 cookie。

4.5 存储cookie

> 有2种途径去存储cookie.

1. 服务器端通过http响应头Set-Cookie来通知浏览器存储cookie。例如：(这个暂时不研究，仅了解)。浏览器收到这个头响应头之后会自动存储这个cookie

```
HTTP/1.1 200 OK
Content-type: text/html
Set-Cookie: name=value; expires=Mon, 22-Jan-07 07:10:24 GMT; domain=.wrox.com
Other-header: other-header-value
```

2. 客户端(浏览器端)通过JavaScript去存储cookie(重点研究)。Javascript操作cookie需要通过document的cookie属性来完成。

```
//存储cookie，并设置超时时间。  domaim和secure使用默认的情况
<script type="text/javascript">
    /*
        在存储cookie的时候，name和value是必须的
        参数1：cookie的name
        参数2：cookie的值
        参数3：存储时间 单位天

        只要name不同，就可以存储多个cookie
    */
    function saveCookie(name, value, expiredays){

        var date = new Date();
        alert(date.getDate())
        date.setDate(date.getDate() + expiredays);
        alert(date.toGMTString())
        //cookie的内容其实就是一串纯文本。 对字符最好进行url编码，否则如果有中文就回出现问题。
        var cookieText = encodeURIComponent(name) + "=" + encodeURIComponent(value) + ";
expires=" + date.toGMTString();
        document.cookie = cookieText;
    }
    saveCookie("name", "张三", 1);
</script>
```

4.5 读取cookie

> 读取cookie仍然是通过document的cookie属性，不过字符串需要我们自己解析。
>
> 只能说，JavaScript的这个API设计的真烂

```
<script type="text/javascript">
    function getCookie (name) {
        //先拿到cookie字符串
        var cookieText = document.cookie;
        //获取要查找的 cookie的 name所在的下标
        var index = cookieText.indexOf(name + "=");  // name=李四; ...
        //如果下标不为-1，证明找到了
        if(index != -1){
            var endIndex = cookieText.indexOf(";", index);  //查找指定的cookie的结束为止
            // 如果等于-1，证明没有找到;号，则把末尾设置为字符串的末尾
            endIndex = (endIndex == -1 ? cookieText.length : endIndex);
            //把想要的cookie的value截取出来
            var value = cookieText.substring(index + (name + "=").length, endIndex);
            //因为存储的时候使用了url编码，所以查到的东西需要url解码
            return decodeURIComponent(value);
        }
    }
    var value = getCookie("name");
    if(value){
        alert("欢迎你: " + value);
    }

</script>
```

4.6 删除cookie

> 删除cookie的原理非常简单，只要把超时时间设置为0，就回立即删除了。

```
function saveCookie(name, value, expiredays){

        var date = new Date();
        alert(date.getDate())
        date.setDate(date.getDate() + expiredays);
        alert(date.toGMTString())
        //cookie的内容其实就是一串纯文本。 对字符最好进行url编码，否则如果有中午就回出现问题。
        var cookieText = encodeURIComponent(name) + "=" + encodeURIComponent(value) + ";
expires=" + date.toGMTString();
        document.cookie = cookieText;
}
    saveCookie("name", "", 0);
```