

2.3 相对和绝对定位

你可能会注意到在本例中我使用了相对定位。在相对定位和绝对定位中有一个很微妙但确实的区别。

当你将一个对象用绝对定位指令定位时，你实际将其从你的HTML文件流中抽了出来，将其直接根据网页的左上角的位置进行定位。这种情况下各个对象有可能互相重叠在一起。

相对定位并不指根据百分比值或其他某个元素的位置来定位这个元素的位置。它实际占据的就是它在这个HTML文件中的位置，它的定位起点基于HTML页所在的位置，这里是一个例子：

Tim, the hero. Webmonkey editor and resident banjo-picker.

Tim, the hero. Webmonkey editor and resident banjo-picker.

这两个句子中，单词webmonkey都定位在距左边200像素的位置。第1个句子用的是绝对定位，第2个用的是相对定位。注意相对定位的句子中空出了相当于单词webmonkey长度的空间，而绝对定位了的句子中没有这个空间，如果我给两个句子都加一个"top"参数值，则你会看到绝对定位了的句子将飘在文章的顶部，而相对定位的句子则就在它所在的句子的位置之下：

Tim, the hero. Webmonkey editor and resident banjo-picker.

现在我们已经了解了两种定位的不同。我们作一个小练习。我们用CSS-P作一个网页。这里是该网页的[屏幕快照](#)。试着做出这个网页，然后我们将制作更复杂的布局（提示：你可以用这个屏幕快照中的图象作为网页背景图象来帮助你定位。注意将字体采用Comic Sans MS）。

后页
前页
目录
北极星书库

2.4 学习分层技术

你可以注意到上面的那个布局用表格就能很轻易地做到。但是串接样式表比表格有一项独特的优势：分层。

你肯定注意到你无法用表格将几幅图片或文字叠放在一起。一般情况下，如果人们希望实现这种效果只有制作一幅位图，在图片中制作出文字或图象叠放在一起的效果，然后将图片放在网页中。

而利用动态HTML，则可以利用分层顺序将叠放显示各个对象，例：



在这个例子中，Aaron在他的桌子后面，Aaron的图片首先出现在源代码中：

```
<div id="aaron">  
  
  
  
</div>
```

```
<div id="desk">
```

```

```

```
</div>
```

但是如果我们想让Aaron站在桌子前面，我们可以这样写HTML代码：

```
<div id="desk">
```

```

```

```
</div>
```

```
<div id="aaron">
```

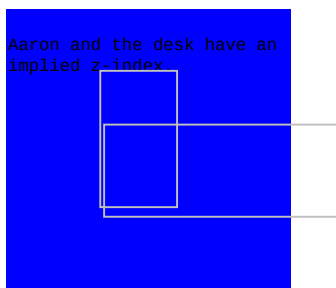
```

```

```
</div>
```

这种类型的分层在HTML暗示编写。所以搭建动态HTML网页的方法之一就是将所有对象按照显示的前后层次顺序排列排在显示底层的对象最先列出，而排在显示最上层的对象在源代码顺序的最后列出。但是这种方式不一定不出问题。很多时候对象在源代码中的顺序和它最后的显示顺序必须没有任何关系。而此时就需要用到z-index CSS属性。

这里是重新制作后的例子。这次的HTML标识和一起一样，但是对象采用了明示的z-index。



两个例子的HTML都是相同的，但CSS不同。

```
<style type="text/css"> #aaron {position:absolute; left:
8px; top: 31px; width: 79px; height: 73px; z-index: 2 }
</style>
```

z-index可以是一个正值或一个负值（带负值的元素将位于母体元素的下面），它所造成的显示效果为：如果一个对象的z-index比另一个元素大，则z-index为1时的视觉效果和z-index为2时的视觉效果的区别相当于1和1000之间的区别。

[后页](#)
[前页](#)
[目录](#)
[北极星书库](#)

2.2 深入

动态HTML中大多数的内容都要用到<DIV>标签作为容器。(在串接样式表和CSS定位的规范草案中规定可以将任何对象定位，但Netscape的浏览器还不支持这项规定。所以你必须用<DIV>标签作为类属容器)，在容器内加入对象然后再定位。

我们先为本教程中的范例生成一个box。

```
<html>
  <head>
    <title>dramatis personae</title>
    <style>
      <!--
        #tim {position: absolute;
          left: 10px;
          top: 10px;
          width: 140px;
          height: 91px;
        }
      -->
    </style>
  </head>
  <body>
    <div id="tim">
    </div>
  </body>
</html>
```

这里是box的显示结果（为了辨认的方便，我给它加了一个红色边框）



现在我们有了一个空的box，其ID名称为tim，其位置在距窗口左边10个像素，距窗口顶部10个像素的位置。现在它还没有体现出动态HTML的魅力，所以我们在这个box中填入Tim的图片，并给它加一点说明：



Tim, the hero. Webmonkey editor and resident banjo-picker.

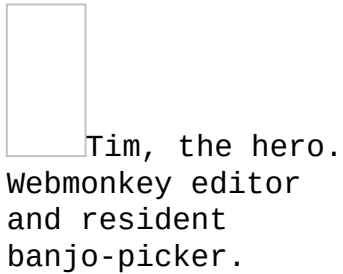
在3.2版本的HTML的网页（即不能支持动态HTML的网页）中，你将会注意到文字一直流到屏幕边缘之外，而Tim的图片朝左浮动。



Tim, the hero. Webmonkey editor and resident banjo-picker.

实际上是它浮动到了本文所在的表格单元的左边。这是一个重要的区别。当你将HTML放在一个已经被定位了的<DIV>之内时，你可以把它看作是放在了一个表格单元中（或者象在其他出版系统之内的制作方法那样）。

在支持HTML 3.2以后的版本的网页中，你可以看到文字自动回行：



现在这个<DIV>标签里已经填充了内容，让我们利用CSS-P的LEFT和TOP选项技将其定位。

```
<html>
  <head>
    <title>dramatis personae</title>
    <style>
      <!--
        #tim {position: absolute;
              left: 300px;
              top: 10px;
              width: 140px;
              height: 91px;
            }
      -->
    </style>
  </head>
  <body>
    <div id="tim">
    </div>
  </body>
</html>
```


显示结果:



Tim, the hero.
Webmonkey editor
and resident
banjo-picker.

[后页](#)
[前页](#)
[目录](#)
[北极星书库](#)

5日学会动态HTML

第一天

·1.1 第一天	·1.2 你需配备的工具
·1.3 这五天的学习计划	

第二天

·2.1 第二天	·2.2 深入
·2.3 相对和绝对定位	·2.4 学习分层技术
·2.5 家庭作业	

第三天

·3.1 第三天	·3.2 微软和网景不同的语法
·3.3 使其在两种浏览器中都可用	·3.4 脚本如何运行
·3.5 访问对象的一个技巧	·3.6 制作互相追赶的动画效果

第四天

·4.1 让我们编写一些有用的东西	·4.2 下拉菜单脚本
·4.3 使菜单初具规模	·4.4 完成菜单条的制作
·4.5 使菜单具备相应的功能	

第五天

·5.1 排错	·5.2 放在底部和右面
·5.3 Netscape的调整大小问题	·5.4 一定要把样式加在标记中吗？
·5.5 关于Event对象	·5.6 课程结束

2.5 家庭作业

这些定位的基本知识是你搭建跨平台动态HTML的第1步。明天我们将学习到动态HTML的精华之处以及目前所受的局限。但是现在，让我们作一个家庭作业。[这里是 act 1, scene 1](#)。下载这些文件，然后将其按照屏幕快照的布局安排布局。明天我们还要将屏幕中的图象制作出动画效果。所以你最好先把今天的作业完成了，因为明天的课程中我们还要它。

明天我们将陶醉在JavaScript的神奇魅力中。

2.3 相对和绝对定位

你可能会注意到在本例中我使用了相对定位。在相对定位和绝对定位中有一个很微妙但确实的区别。

当你将一个对象用绝对定位指令定位时，你实际将其从你的HTML文件流中抽了出来，将其直接根据网页的左上角的位置进行定位。这种情况下各个对象有可能互相重叠在一起。

相对定位并不指根据百分比值或其他某个元素的位置来定位这个元素的位置。它实际占据的就是它在这个HTML文件中的位置，它的定位起点基于HTML页所在的位置，这里是一个例子：

Tim, the hero. Webmonkey editor and resident banjo-picker.

Tim, the hero. Webmonkey editor and resident banjo-picker.

这两个句子中，单词webmonkey都定位在距左边200像素的位置。第1个句子用的是绝对定位，第2个用的是相对定位。注意相对定位的句子中空出了相当于单词webmonkey长度的空间，而绝对定位了的句子中没有这个空间，如果我给两个句子都加一个"top"参数值，则你会看到绝对定位了的句子将飘在文章的顶部，而相对定位的句子则就在它所在的句子的位置之下：

Tim, the hero. Webmonkey editor and resident banjo-picker.

现在我们已经了解了两种定位的不同。我们作一个小练习。我们用CSS-P作一个网页。这里是该网页的[屏幕快照](#)。试着做出这个网页，然后我们将制作更复杂的布局（提示：你可以用这个屏幕快照中的图象作为网页背景图象来帮助你定位。注意将字体采用Comic Sans MS）。

后页
前页
目录
北极星书库

2.1 第二天

在Nadav的DHTML揭密一文中所述，动态HTML网页利用串接样式表布局，并利用Javascript控制网页元素。实际并不复杂或有很多的规定。我们所需要作的只是要给网页加入样式，能使其运动，并且在两种主要的浏览器中都能正确显示。

精通我们将用CSS-P布局网页，这是学习动态HTML的第1步。学习完定位元素之后，我们将接着学习编写使其动态显示的脚本。

大多数用于设计和页面布局的程序或文件格式必须现生成box，它们的宽度和高度可以改变，它们还能使你控制将对象叠放在对象之上。CSS-P的原理与其相似，你先用HTML定义一个容器（为了便于具备跨平台应用的兼容性，你最好实用<DIV>或标签），设置其水平和垂直位置，然后设定哪个对象应该叠放在哪个对象之上。

真的很简单，它能象大多数人（尤其那些利用表格定位和布局的人）希望的那样让HTML发挥功能。你只需以一种不同的思维角度来思考网页的布局，本教程将重点讲述一些基本的布局技巧明天我们将利用Javascript控制你的网页布局。

1.1 第一天

经过微软和Netscape关于浏览器大战新闻发布会以及种种传闻和无休止的辩论之后，你从新发布的这两种4.0版的浏览器中得到了什么呢？

只是一个占据了你的硬盘40-MB空间的庞大怪物？还是可以推动媒体发展的令人兴奋倍受鼓舞的充满活力的新生事物？除了一些有能力制定4.0版本规格的大型站点之外，实际上动态HTML还没有真正广泛应用于互联网。

如果你不了解动态HTML的含义，你应该了解dHTML只能应用于最新版的浏览器，即Internet Explorer 4和 Netscape Navigator 4。

这两种浏览器对于动态HTML的支持并不相同，所以目前缺少一种编写dHTML网页的通用方式。这就是动态HTML不能在互联网中得到广泛应用的原因。网站开发者为了使自己的网页让尽量多的人看到，宁可使网页的编写臃肿不堪也不敢使用动态HTML使自己的网页具备更高的效率。因为用动态HTML编写的网页在这两种广泛使用的浏览器中显示的效果有很大的区别。

我们将学习使用串接样式表编写网页并利用Javascript给网页加入动画。如果某些用户认为在网页中加入满天飞的动画实在是无益的劳动，那么我还要讲授一些只有动态HTML才能做到的非常实用的动态用户界面。我们将充分结合我们大脑两个半球的优势编写动态的事件驱动动画，这种动画可以根据用户的指令进行变化，如果这些还不足够吸引你，我们还将应用结构、数组循环、对象引用以及更多的串接样式表使你拥有编写可以令任何梦想都变成现实的结构化网页编程能力。

后页
前页
目录
北极星书库

1.2 你需配备的工具

坦白的地说，动态HTML不是为初学者准备的教程。要引用动态HTML，你必须熟悉3中不同的技术：HTML、样式表和JavaScript。动态HTML网页实际上就是这3种技术的集成。所以第1天的课程只是一堂复习课。你的作业就是要重温以前所了解的关于这3种技术的知识。不是很难，但你至少要掌握这3种技术的语法。

对于JavaScript, 建议你先阅读Javascript教程。你应该特别注意学习一下第2日教程中关于变量的知识，第3日教程中文件对象模块（DOM）的知识，第4日教程中的循环和数组，以及在表单中通过引用来控制对象的技巧。

关于串接样式表的内容，你可以阅读[5日精通样式表教程](#)熟悉一下样式表的语法。你还应该阅读[CSS定位一文](#)，学习如何利用样式表布局页面。

至于HTML，你应该熟悉<div>和标签。

我并不希望你阅读和记住所有这些内容，你只需浏览一下，知道所需要的知识在什么位置可以找到就可以了。

1.3 这五天的学习计划

这里是本教程所要讲授的内容：

- 第2日：利用CSS定位生成在两种浏览器都可以浏览的网页布局，以及如何将文字和图象叠放在彼此之上。
- 第3日 利用你的布局并制作动画效果。
- 第4日用动态HTML建立用户界面，以及如何隐藏和显示界面的元素。
- 第5日：更深入地了解动态HTML以及常见的问题。
- 明天见。

3.1 第三天

今天我们将开始学习动态HTML最精彩的部分：动态HTML编程。HTML之所以成功的原因之一在于它提供了在互联网上展现网页的统一方式。一旦你标识了一个网页，它在任何地方都可以看到。JavaScript也是如此。及时它的版本会有很多种，但它的核心和语法是统一的，它在支持JavaScript的所有浏览器中都可以应用。CSS也具有统一的语法。在所有的浏览器中它的显示结果也应该是一致的。所以它能帮助你生成跨平台风格一致的动态网页。

但是，还有一些小问题。

在1997中期时，Netscape和Microsoft都推出了自己的动态HTML浏览器，但那时没有一种规范编程语言同网页元素之间的交流的统一规定。但是在那时已经有了JavaScript同网页中的图象、链接和表单元素进行交流的规范。然而这两家公司对于如何控制网页元素存在不同的见解。

Netscape对JavaScript加入了一系列的对象，并引入了<LAYER>标签，伴随产生的便是它的dHTML理念：- 或者采用<LAYER>标签分层的内容分层，或者<DIV>标签的CSS定位，它们可以用JavaScript的对象访问，而且它的属性(left, top, visibility,和background color)可以被控制。

但是微软走得更深入一些。它推出了全新的HTML生成引擎，它不仅能定位任何元素，还能改变对CSS设置的任何选项。（例如，你可以改变字体或你的EM标签），而且，它还完善了文件对象模块（Document Object Model），使其能通过任何IE支持的编程或脚本语言访问。

这两种体制之间的明显区别在于其语法。Netscape 支持树形语法：

```
document.layers['topLayer'].document.layers['subLayer'].document.lay
```

而Internet Explorer将所有的HTML对象都放在一个水平结构上，使你可以修改也是对象：

```
bottom.style.left
```

如果你在这种浏览器中使用了另一种浏览器支持的方法，则浏览器会显示错误信息。但是还有许多技术可以对这种不兼容性作出变通处理。一旦你掌握了其中的诀窍，你就可以左右逢源，无所不能。本课中将学习如何利用最少量的条件访问文件对象模块DOM，然后我们将介绍一个将对象在屏幕中移动的简单Javascript脚本。

[后页](#)
[前页](#)
[目录](#)
[北极星书库](#)

3.2 微软和网景不同的语法

首先我们学习为两种浏览器生成不同的动态HTML。我不打算在此列出两种浏览器各自的动态HTML特点，我只列出两种浏览器所支持的相同的动态HTML功能。

Function	Navigator	Explorer
改变某个区域的内容:	<code>object.document.open();</code> <code>object.document.writeln();</code> <code>object.document.close();</code>	<code>object.innerHTML</code>
识别不同的区域:	<code>object.name</code>	<code>object.id</code> or <code>object.name</code>
根据母体对象的左边定位:	<code>object.left</code>	<code>object.style.left</code>
根据母体对象的顶部定位:	<code>object.top</code>	<code>object.style.top</code>
堆栈顺序:	<code>object.zIndex</code>	<code>object.style.zIndex</code>
对象的可视性:	<code>object.visibility</code>	<code>object.style.visibility</code>
clipping区域:	<code>object.clip</code>	<code>object.style.clip</code>
背景图像:	<code>object.background</code>	<code>object.style.backgroundImage</code>
背景颜色:	<code>object.bgColor</code>	<code>object.style.backgroundColor</code>

你可以看到它们之间没有完全相同的项目。`object.name`似乎相同，但是对对象使用CSS时必须制定ID属性，所以你只能对图象、链接和表单 使用该功能。所以为了编写脚本，你必须设定很多条件。听起来很痛苦，但其实还不象你想象的那么糟。

3.3 使其在两种浏览器中都可用

有很多中办法可以利用JavaScript设定条件。其中办法是确定用户所使用的浏览器的类型和版本，并据此作出评估。但现在我还发现了更好的解决办法。

浏览器类型和版本条件设定的第1个问题是浏览器的改变的问题。如果你将浏览器版本设定为Internet Explorer 4.0和Netscape 4.0，那么等5.0版的浏览器发布时你的网页会发生什么情况？或者这两种新发布的浏览器版本互相兼容，而你的脚本中都没有考虑到这些问题又会如何？所以根据浏览器类型和版本设定条件的方法不可行。

更好的方法是根据功能设定条件。如果你对JavaScript比较熟悉的话，你一定知道JavaScript的"if"语句中测试一个语句返回的是真true (1)还是假false (0)，然后它将根据返回的布尔值执行一个语句。通常情况下它被用来测试某些变量的条件，例如：

```
if (x<=5) setTimeout('doSomething()', 500);
```

但是一个对象是否存在也可以作为被测试的条件。如果某个对象存在，则返回真，否则返回假。所以为了提供一个有效的测试条件，你必须从某个特定角度来测试一个对象的代表性质以便应用动态HTML。

Netscape利用分层运行动态HTML。它的运行方式同图象数组在Navigator中的方式类似。你设定一个分层对象的数组，象对图象的引用那样设定引用方式。所以document.layers['foobar']则是代表被相对定位或是绝对定位了的<DIV>的对象 (在Netscape模式中，它必须是在你的文件NAME或ID属性为"foobar"的一个绝对定位或相对定位了的<DIV>或)。然后你就可以设定上述各种属性。你还可以用document.foobar来代表这个对象。

在Internet Explorer中，则用一种水平的结构来代表对象。所以ID或NAME属性为foobar的<DIV>就是foobar，所以你可以用foobar.style.left来访问或设定这个对象的左边的位置。还有许多种循环访问一系列对象的方法（非常有用）。在Explorer的文件对象模块（Document Object Model）中，对象可以代表所有在该对象之下的对象（object object object）。所以foobar.all包含所有包含在foobar之下的HTML标签。无论什么时候你都可以找到一个很好的类属对象 - document.all，因为所有的网页中都有一个文件（document），因此它必然有一个document.all对象。

后页
前页
目录
北极星书库

3.4 脚本如何运行

所以如果你想控制某个对象的位置，你应该这样编写你的JavaScript：

```
<script language="javascript">

function moveIt() {

if(document.layers) {

document.truck.left -= 5;

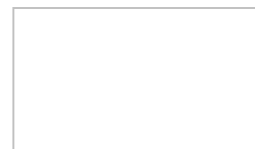
if (document.truck.left<0){
document.truck.left=480}

}

else if (document.all) {
truck.style.left="parseInt(truck.style.left)" 5;
if (parseInt(truck.style.left)<0)
{ truck.style.left="480;" } }

if ((document.layers) || (document.all))
{ setTimeout('moveIt()', 100); } } </script>
```

它的执行结果应该是：



代码的含义如下：

- 如果客户端支持document.layers对象，

- 将名为truck的对象层左移5个位置.
- 如果名为truck的对象层的位置小于0,
 - 则名为truck的对象层的left属性设置为480。
- 但是，如果客户端支持document.all对象，
 - 则将名为truck的HTML对象的样式选项left的整数值减去5。
 - 如果名为truck的HTML对象的样式选项left的整数值小于0，则将其数值设为480。
- 如果客户端支持document.all或document.layers对象中的任何一种，
 - 等候10秒钟，然后执行函数moveIt。

现在你自己作一下这个程序。让拖拉机在页面中移动。注意：你应该采用行内样式，例：`<DIV id="truck" style="position: absolute; left: 20; top: 20">`。

3.5 访问对象的一个技巧

现在你已经可以在页面中移动对象了。但是如果你需要将多个对象按照特定的顺序移动或者执行多个事件触发的行为时，不仅的代码体积会增加（每次访问一个对象时你都必须使用一次if/then 语句），而且每次都必须为某个行为键入`document.truck.left`是一件很烦人的事。在Netscape模块中，当你嵌入DIV时，文件对象模块的层次结构就会增加，如下：

```
<div id="foo">  
  
<div id="bar">  
  
<div id="sna">  
  
</div>  
  
</div>  
  
</div>
```

要访问foo，则必须执行`document.foo`，而要访问bar，则必须执行`document.foo.document.bar`。而要访问sna，则必须执行`document.foo.document.bar.document.sna`。我快要受不了啦啦啦！！！！

所以你必须解决引用对象时造成的代码体积膨胀的问题，还得避免每次移动一个对象时必须设定条件。但你可以用一个技巧解决这个问题。

任何用JavaScript编写过网页的人都知道任何打开一个小窗口：

```
windowID = window.open('name', 'http://blah.com/');
```

这项指令就可打开一个小窗口，但是你还可以通过使用windowID 作一个引用在继续控制追赶窗口。例如windowID.location = 'http://www.taylor.org/'就可以改变窗口资源的定位。"windowID.close()"就可关闭该窗口。你所做的只是对一个Javascript对象设置一个引用。在动态HTML中也可以使用同样的技巧。

你可能会注意到在上一页的图表中，大多数定位属性的语法都很相似。只不过它们被用在了不同的对象上。我们可以用一个JavaScript例程解决上面的问题。

```
<script>

function setup(){

if(document.layers){

daTruck = document.truck;

} else if(document.all) {

daTruck = truck.style;

}

}

}

</script>
```

现在moveIt函数可以被改为

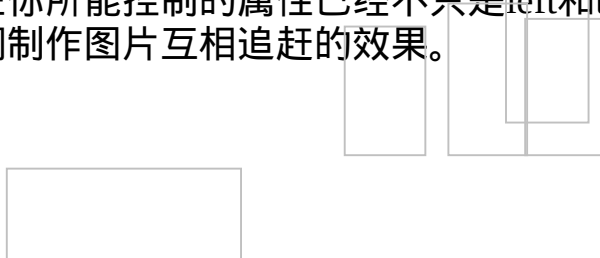
```
function moveIt() {  
  
    daTruck.left = parseInt(daTruck.left) - 5;  
  
    if(parseInt(daTruck.left) < 0){  
  
        daTruck.left = 480;  
  
    }  
  
    setTimeout('moveIt()', 100);  
  
}
```

代码短了一些，对吧？下面我们将使页面这些图象都动起来。

[后页](#)
[前页](#)
[目录](#)
[北极星书库](#)

3.6 制作互相追赶的动画效果

现在你能控制的属性已经不只是left和top了，当然，还有z-index。现在我们制作图片互相追赶的效果。



```
function moveMonkey(monkey, dir) {  
    if (document.all) {  
        var wtMonkey = document.all(monkey).style;  
    } else if (document.layers) {  
        var wtMonkey = document.monkeyContainer.document.layers[monkey];  
    }  
}
```

```
if ((parseInt(wtMonkey.left) <0) || (parseInt(wtMonkey.left)> 150)) {  
  
    dir = dir * -1;  
  
    wtMonkey.zIndex = 5 - dir;  
  
}  
  
wtMonkey.left = parseInt(wtMonkey.left) + dir;  
  
setTimeout('moveMonkey(\" + monkey + '\', ' + dir +')', 100);  
  
}
```

图片的名字被传递给函数，函数则改变图片的左边的定位。然后进行标准的功能检查，为特定的浏览器设定引用参数。然后它改变z-index，如果图片移动到了桌子的任何一端，则函数改变图片移动的方向。

注意这不是编制类属JavaScript移动例程的唯一方法。你可以随意添加、减少或编写自己的例程。下面是今天的家庭作业。将[act I](#), [scene I](#)作出动画效果。你可以使用上面所提供的代码或者编写你自己的代码。当你编写例程时注意Netscape嵌入定位对象的方法。

[后页](#)
[前页](#)
[目录](#)
[北极星书库](#)

4.1 让我们编写一些有用的东西

如果你认真阅读了本教程并做了留给你的家庭作业的话，你心中肯定还存在这样一个疑问：这个动画是很有趣，但是你做的网站仍然是比较朴实无华的那种，所以动态HTML对你来说可能没有太大的价值。

没关系，今天的教程就是针对你这样的问题制作的。今天我们将建立几个用户界面元素，处理用户事件，动态改变背景，自动变换网页内容以及显示或隐藏某些页面对象。所有这些内容都有很强的实用性。

你最好能先了解JavaScript语言中关于链接事件的课程，以便对用户事件具备初步的了解。在今天的课程中将经常用到链接事件。

4.2 下拉菜单脚本

让我们编写一些我们所熟悉的用户界面元素 - - 下拉菜单，子菜单。这些元素和你现在正在浏览的窗口顶部的菜单条很类似。大多数操作系统都有一个类属专用接口工具集，当你需要应用下拉菜单时就可调用该工具集。然后你可以设定被选项目的名称和执行的执行行为。

由于HTML不包含这样的专用接口工具集，所以你必须建立自己的工具集。首先你必须描述它的外观。我们在菜单条上加入一定的颜色以及描述各项功能的名称，例如file、edit、view等等。当你点击该文字时，就会显示该菜单项下的子菜单。在子菜单框中点击相应的菜单项目就可触发某个执行程序的行为。让我们在HTML中编写这样的菜单。

首先，你需要生成一个DIV作为顶部的菜单条：

```
<div id="menuBar"> </div>
```

现在将该菜单条放在页面顶部，加上特定的颜色：

```
<style type="text/css">
#menuBar {position: absolute;
           left: 0;
           top: 0;
width: 100%;
           height: 22px;
           border: 1px solid #99ffff;
           background-color: #99ffff;
           layer-background-color: #99ffff;
}
}
```



```
</style>
```

则生成这样一个DIV：

你可能会对最后一条CSS规范不知所以然，这项是Netscape对CSS的扩展，它指的是绝对定位了的元素的背景色。由于边框设置为border: 1px solid #99ffff;，所以有必要加入这一条使背景色延伸到DIV的边框。否则背景色只能在DIV所包含的内容底下显示。

[后页](#)
[前页](#)
[目录](#)
[北极星书库](#)

4.3 使菜单初具规模

下面我们插入一个菜单目录。我们将标题设置为"Menuitem"。

```
<style type="text/css">

#menuBar {position: absolute;

left: 0;
top: 0;
width: 100%;
height: 22px;
border: 1px solid #99ffff;
background-color: #99ffff;
layer-background-color: #99ffff;
}

.daMenu {position: absolute;

width: 100px;
height: 22px;
border: 1px solid #99ffff;
top: 0px
}

</style>

...

<div id="menuBar"> </div>
<div id="webmonkey" class="daMenu">
```

```
MenuItem </div>
```

显示结果如下：

MenuItem

下面在webmonkey菜单项下加入子菜单：

```
<style>

.moreMenu {position: absolute;

width: 100px;

border: 1px solid red;

background-color: red;

layer-background-color: red;

top: 22px;

}

</style>
```

...

<div id="moreMonkey" class="moreMenu"> Aaron

 Captain Cursor

 Cassandra

 Chris

 Courtney

 Jeff

 Joanne

 Jean Pierre

 Klug

 Kristin

 Nadav

 Taylor

 Thau

 Tim

```
<br> Wendy
```

```
<br>
```

```
</div>
```

同其他菜单项结合后显示结果如下：

MenuItem

Aaron
Captain Cursor
Cassandra
Chris
Courtney
Jeff
Joanne
Jean Pierre
Klug
Kristin
Nadav
Taylor
Thau
Tim
Wendy

下面我们为菜单项设定功能。首先要做的是选择moreMenu类，并将其隐藏。因为通常情况下，如果你不点击菜单项，它不会显示子菜单目录。所以，你应该加入下面的代码：

```
.moreMenu {position: absolute;
```

```
width: 100px;
```

```
border: 1px solid red;

background-color: red;

layer-background-color: red;

visibility: hidden;

top: 22px;

}
```

现在这个DIV仍然在页面中，但浏览器不将它显示出来。

[后页](#)
[前页](#)
[目录](#)
[北极星书库](#)

4.4 完成菜单条的制作

现在给菜单条加入脚本使其能响应鼠标的行为并执行相应的功能。我们利用`visibility`属性隐藏子菜单项目并在菜单项目被点击时显示子菜单。

处理`visibility`时，你会立刻遇到DOM不兼容的问题。Netscape的DOM更多地受其`LAYER`标签和属性的影响。所以即使在DOM中你将一个对象的`visibility`属性设置为`hidden`（隐藏），Netscape也会将其显示出来。如果你加入下面这条：

```
if (daMenu.visibility == 'hidden')
```

你所得到的不是样式表语法的参数值，而是`LAYERS`语法的参数值。

所以如果你设置`document.foo.visibility = 'visible'`，则代码将会按照你预想的那样执行，对象`foo`在屏幕中可以被看到。如果你用`alert(document.foo.visibility)`检查参数值，则返回的数值是`show`。解决办法是设置一些变量，用标准的条件并设定一个名为`visible`的变量用于Netscape中的`'show'`，在Internet Explorer中则设置`'visible'`。

在Netscape中设置一个名为`'hide'`的变量，在Internet Explorer'则用`hidden'`。在相应的字符串位置放入这些变量，则问题就解决了。

```
<script>

if (document.layers) {

    visible = 'show';
```

```
hidden = 'hide';

} else if (document.all) {

visible = 'visible';

hidden = 'hidden';

}

function barTog(menu) {

if (document.layers) {

daMenu = document.layers[menu];

} else if (document.all) {

daMenu = document.all(menu).style;

}

if (daMenu.visibility == visible) {

daMenu.visibility = hidden;

} else {
```



```
daMenu.visibility = visible;

}

lastMenu = daMenu;

}

</script>
```

barTog函数所做的是设置标准的条件语句，如果在被传送的变量菜单中设定的对象是visible，则隐藏该对象，否则就显示该对象。然后将对该对象的引用传递给名为lastMenu的全局变量（这样以来，以后你可以再关闭它）。接下来你要做的就是调用这个函数，所以在菜单条中你将"Webmonkey"设定一个anchor（锚区），并设定被点击是执行的行为。

```
<div id="webmonkey" class="daMenu">

<a href="#" class="itemAnchor"

onclick="javascript: barTog('moreMonkey'); return false;">

Webmonkey

</a>

</div>
```

这段代码执行barTog函数，并将应该被打开或关闭的DIV 的名称传递给它。显示结果如下。

[后页](#)
[前页](#)
[目录](#)
[北极星书库](#)

4.5 使菜单具备相应的功能

下面要做的是给每个菜单条设定一个行为。你可以设定任何一种行为，不一定非得是动态HTML。但由于本教程讲的是动态HTML，而我们的课题是将如何改变layers的visibility(可视性)属性。所以我们为每一个菜单项目设定隐藏和显示包含相应的网猴图象的层(layer)。

```
<script>

function menuItem(item){

if (document.layers) {

daLast = document.layers[lastItem];

daItem = document.layers[item];

} else if (document.all) {

daLast = document.all(lastItem).style;

daItem = document.all(item).style;

}

daLast.visibility = hidden;

daItem.visibility = visible;
```

```
lastMenu.visibility = hidden;

lastItem = item;

}
```

```
var lastItem = 'aaron';
```

```
</script>
```

这段脚本仍然用我们的老朋友 - - 条件语句调用两个对象，现调用item，然后是最后被打开的lastitem(lastItem的缺省设置是'aaron')，并且该语句还要记忆被打开的最后一个菜单（我们在barTog中设置的lastMenu）。设置完所有visibility后还记忆lastItem。

下面需要给每个菜单项目加上相应的anchor调用。

```
<a href="#" class="itemAnchor"

onclick="menuItem('captCursor'); return false">
```

这条代码调用menuItem函数，并指示其显示ID为"captCursor"的DIV。现在将anchor锚定每个菜单项目并将每个DIV指向相应的网猴的名称。现在当你从菜单中选择相应的菜单时就会显示相应的图片。

现在你可以制作你自己的菜单并在其中加入自己的菜单项目，隐藏及显示DIV，并在DIV中加入表单元素。明天的课程中我们将结合前面所学的知识作一个完整的展示。

[后页](#)
[前页](#)
[目录](#)
[北极星书库](#)

5.1 排错

现在，经过四天的动态HTML教程，你已经知道如何制作动画和用户界面。你可以定位对象，在网页中移动它们，用程序改变它们的z-index和可视性。你知道如何条件化不同的DOM和写在Microsoft和Netscape的浏览器中都可用的文档。也许你觉得用手工编码太辛苦，于是你抛弃了文本编辑器，投入所见即所得的编辑工具的怀抱。

但是不管你用哪中工具，你都会遇到一些问题。比如你想真正让导航条一直放在浏览器的右侧。或者用户访问你的站点时不停地调整浏览器的大小，于是Netscape用户会抱怨所有的样式信息突然消失（当你调整窗口尺寸时，Netscape扔掉了所有的样式信息）。或许，你在制作动画，并把所有的演员放在后台，让他们等着进入，但是有的人有一个非常大的高分辨率显示器，能看出页面边缘的演员。你在使用动态HTML时会遇到所有这些问题。教程的最后一天将涉及其中的一些问题。

5.2 放在底部和右面

CSS在定位上的缺点是它只有left和top属性，但是没有bottom或right。这就意味着你只能放一个对象在离浏览器窗口50个像素的位置，但是你不能只用CSS就能把对象放在离浏览器窗口右边50像素的地方。这个问题在CSS2中应该能得到解决，样式表的下一代specification正在出笼；在它被主流浏览器支持之前，你只好求助于JavaScript。

用JavaScript解决这个问题的最好方式是基于文档宽度的计算。两种4.0浏览器都在DOM中包含文档大小的对象。所以定位对象的方式是把窗口的宽度减去对象的宽度。

```
<div id="foo">

your content here

</div>

<script>

if (document.layers) {

document.foo.left = window.innerWidth
                    - document.foo.clip.width;

} else if (document.all) {

foo.style.left = document.body.offsetWidth
                - parseInt(foo.style.width);
```

```
}
```

```
</script>
```

这可以把div定位到屏幕的右边。这种技术也可以把对象定位到窗口底部。

两种浏览器都支持的屏幕对象是：

Feature	Internet Explorer	Netscape Navigator
height of the screen	screen.height	screen.height
width of the screen	screen.width	screen.width
color depth of the screen	screen.colorDepth	screen.colorDepth
height of the window	document.body.offsetHeight*	window.innerHeight
width of the window	document.body.offsetWidth*	window.innerWidth

*从技术上讲，这是文档的高度和宽度，不是窗口的，但是大多数情况下可以把它们看成一件事。

5.3 Netscape的调整大小问题

如果你用的是Netscape，在看这篇教程的时候，可能要调整浏览器窗口的大小以便达到较好的显示效果。这是可能会突然出现一个可怕的闪光，然后所有的定位信息都消失了，你的页面看起来象jodi.org。

这是Netscape 4.0的一个bug。有解决办法吗？较简单的方式是在页面上放一个提示：如果您是Netscape用户，当调整浏览器尺寸时，需要重新下载页面。

或者可以使用程序：

```
<script>
if (document.layers) {
window.onResize = reloadIt;
}
function reloadIt() {
document.location = document.location;
}
</script>
```

这种方法在大多数情况下管用。（但是，Netscape偶尔会进入重新下载文档的死循环。）

北极星书库

5.4 一定要把样式加在标记中吗？

如果你经常访问网猴，你可能知道我们鼓吹在线文档中结构、表现和行为的分离。你保持你的HTML结构，用CSS进行布局，然后用JavaScript做一些事情。尽可能地不要把这三者混在一起。问题是，当你从文档的head部分应用样式规则到对象时，在IE中，整个规划会分离到不同部分。

```
<html>

<head>

<title>DOM example</title>

<style>

#foo {position: absolute; left: 10px; top: 10px}

</style>

<script>

function alertIt() {

alert(foo.style.left);

</script>
```

```
</head>

<body onload="alertIt()">

<div id="foo">This is the sample</div>

</body>

</html>
```

你得到一个空的对话框，是吗？

你的第一个反映可能是假定下载过程中断了。但是实际上是IE把这个整体分离。因为如果你查看文档，对象foo并没有样式在其中，虽然它有一个指向样式表的ID。所以foo的唯一属性是ID。你可以做一个实验，在foo标记中加入：

```
<div id="foo" bar="neat">This is the sample</div>
```

现在alert(foo.bar)会返回"neat"。你看出来在IE中发生什么了吗？标记中的任何属性都在DOM中作为对象的属性出现。但是如果它不在对象内，就在DOM中没有反映。这就是为什么把样式加到标记中。

为了修正这个问题，我们再一次使用JavaScript的[object-as-reference](#)能力。但是不指向HTML对象，我们为按照ID为对象指定样式表规则。页面的结果和在你的脚本中的一致，但是不需要把样式放在标记行中。

```
<script>
```

```
function setup(myId){
if(document.layers){
myObj = document.layers[myID];
} else if(document.all) {
for (ss=0 ; ss < document.styleSheets.length; ss++)
{
for (sr=0 ; sr < document.styleSheets(ss).rules.length;
sr++) {
if (document.styleSheets(ss).rules(sr).selectorText ==
'#' + myId) {
myObj = document.styleSheets(ss).rules(sr).style;
}
}
}
}
}
}
</script>
```

结果是循环经过所有的页面的样式表。如果其中一个与你的对象的ID匹配，它就作为这个对象的别名。你可以象从前一样处理，但是用的HTML代码更整洁。

[后页](#)
[前页](#)
[目录](#)
[北极星书库](#)

5.5 关于Event对象

在你完全掌握dHTML前，你需要了解event对象。两种4.0浏览器都包含event对象。它在事件创立时产生，如点击一个可点击的对象，移动鼠标，或聚焦到一个窗体元素上。Event对象被创建然后传递给处理事件的函数。

下面是event对象属性的描述，以及Netscape和IE处理它们的方式：

描述	Microsoft 属性	Netscape 属性
代表事件类型的字符串	type	type
代表最初发送给对象事件的字符串	srcElement	target
光标横坐标	x	x
光标纵坐标	y	y
相对于页面的横坐标	clientX	pageX
相对于页面的纵坐标	clientY	pageY
相对于屏幕的横坐标	screenX	screenX
相对于屏幕的纵坐标	screenY	screenY
键代码	keyCode	which
Netscape返回键的代码，IE 返回 true或false	altKey ctrlKey shiftKey	modifiers

4.0浏览器增加了一些新事件：

onDb1Click	鼠标双击
onKeyDown	键被按下

onKeyPress	键被按下然后被释放
onKeyUp	键被释放
onMouseDown	鼠标被按下
onMouseMove	鼠标移动
onMouseUp	鼠标被释放
onResize	窗口被调整大小

4.0浏览器也增加了处理事件的新方法，虽然它们（Netscape和IE）的方式不同。Netscape用的是“时间捕捉”，IE用的是“事件气泡”。

事件处理对Netscape处理如mouseMove或keyPress等事件是必须的，它并不隐含指向一个标记或元素。你应该告诉客户注意这些事件，并告诉它用什么函数来处理它们。下面例子用window对象的captureEvents方法来描述正在捕捉的事件：

```
window.captureEvents(Event.MOUSEMOVE);
```

注意到在不用on作为名称一部分的情况下特定事件是如何被指向的。你只是告诉Netscape注意所有发生在窗口中的事件并捕捉它们。然后你需要告诉Netscape用这些被捕捉的时间做些什么。注意on又出现了。

```
window.onMouseMove = handlerFunction;
```

```
function handlerFunction(yourEvent) {
```

```
    alert(yourEvent.screenX);
```

```
}
```

这段代码是一个演示事件过程的烦人的例子。每次你移动鼠标，一个对话框会跳出来告诉你它的横坐标位置。以这种方式处理的事件传递一个指针到event对象。从这儿你可以获取必

要的信息。一旦你厌倦了捕捉事件，你可以象这样释放它们：

```
window.releaseEvents(Event.MOUSEMOVE);
```

于是这种类型的事件不再被捕捉。

IE用不同的方法处理事件，叫做“事件气泡”。在这种方法中，如果你有这样一个结构：

```
<body onclick="bloorf()">
<p onclick="baz()">
<em onclick="bar()">
<strong onclick="foo()">Click on me</strong>
</em>
</p>
</body>
```

如果你点击strong标记内的文本，它接收到一个onClick事件，然后发送onclick事件给标记，处理它然后发送到<p>标记，等等，然后直到窗口。这样每个元素以自己的方式处理点击。但是如果你想停止气泡上传，可以取消气泡。

```
<script>
function foo() {
```



```
        doSomething();
        this.event.cancelBubble = true;
    }
</script>
```

所以如果你不想让某个事件传递到所有它包含的标记，可以象上面那样阻止它。

你可以看到，因为两种浏览器存在相似的事件，每一步都需要大量的条件化工作，而且没有简单的方式把它们映射到一个句法中。

[后页](#)
[前页](#)
[目录](#)
[北极星书库](#)

5.6 课程结束

虽然动态HTML背后有很多要了解的，但是你现在已经可以开始并使dHTML适应不同的浏览器。如果你想了解更多的内容，[Microsoft](#)和[Netscape](#)在它们的网站上都有扩展dHTML文档。

那么你觉得动态HTML怎样？爱它还是恨它？爱它所做的，但是在实现上的差异上又心存顾虑？事实上浏览器公司和[World Wide Web Consortium](#)已经意识到这种差异，所以将来应该有解决办法。

最后，有一项期末作业：

做一个动态HTML主页，让它运动、让它可视和隐藏、让它反应用户事件。

祝你编码和设计快乐！