互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

第2章 在HTML中使用JavaScript

只要一提到把JavaScript放到网页中，就不得不涉及Web的核心语言——HTML。在当初开发JavaScript的时候，Netscape要解决的一个重要问题就是如何做到让JavaScript既能与HTML页面共存，又不影响那些页面在其他浏览器中的呈现效果。经过尝试、纠错和争论，最终的决定就是为Web增加统一的脚本支持。而Web诞生早期的很多做法也都保留了下来，并被正式纳入HTML规范当中。

2.1 <script>元素

向HTML页面中插入JavaScript的主要方法，就是使用<script>元素。这个元素由Netscape创造并在Netscape Navigator 2中首先实现。后来，这个元素被加入到正式的HTML规范中。HTML 4.01为<script>定义了下列5个属性。

- charset：可选。表示通过src属性指定的代码的字符集。由于大多数浏览器会忽略它的值，因此这个属性很少有人用。
- defer：可选。表示脚本可以延迟到文档完全被解析和显示之后再执行。
- language：已废弃。原来用于表示编写代码使用的脚本语言（如JavaScript、JavaScript1.2或VBScript）。大多数浏览器会忽略这个属性，因此也没有必要再用了。
- src：可选。表示包含要执行代码的外部文件。
- type：必需。可以看成是language的替代属性；表示编写代码使用的脚本语言的内容类型（也称为MIME类型）。虽然text/javascript和text/ecmascript都已经不被推荐使用，但人们一直以来使用的都还是text/javascript。实际上，服务器在传送JavaScript文件时使用的MIME类型通常是application/x-javascript，但在type中设置这个值却可能导致脚本被忽略。另外，在非IE浏览器中还可以使用以下值：application/javascript和application/ecmascript。考虑到约定俗成和最大限度的浏览器兼容性，目前type属性的值依旧还是text/javascript。

使用<script>元素的方式有两种：直接在页面中嵌入JavaScript代码和包含外部JavaScript文件。

在使用<script>元素嵌入JavaScript代码时，只须为<script>指定type属性。然后，像下面这样把JavaScript代码直接放在元素内部即可：

```
<script type="text/javascript">
    function sayHi(){
        alert("Hi!");
    }
</script>
```

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

包含在<script>元素内部的JavaScript代码将被从上至下依次解释。就拿前面这个例子来说，解释器会解释到一个函数的定义，然后将该定义保存在自己的环境当中。在解释器对<script>元素内部的所有代码求值完毕以前，页面中的其余内容都不会被浏览器加载或显示。

在使用<script>嵌入JavaScript代码时，记住不要在代码中的任何地方出现"</script>"字符串。例如，浏览器在加载下面所示的代码时就会产生一个错误：

```
<script type="text/javascript">
    function sayScript(){
        alert("</script>");
    }
</script>
```

因为按照解析嵌入式代码的规则，当浏览器遇到字符串"</script>"时，就会认为那是结束的</script>标签。而通过把这个字符串分隔为两部分可以解决这个问题，例如：

```
<script type="text/javascript">
    function sayScript(){
        alert("</scr" + "ipt>");
    }
</script>
```

像这样分成两部分来写就不会造成浏览器的误解，因而也就不会导致错误了。

如果要通过<script>元素来包含外部JavaScript文件，那么src属性就是必需的。这个属性的值是一个指向外部JavaScript文件的链接，例如：

```
<script type="text/javascript" src="example.js"></script>
```

在这个例子中，外部文件example.js将被加载到当前页面中。外部文件只须包含通常要放在开始的<script>和结束的</script>之间的那些JavaScript代码即可。与解析嵌入式JavaScript代码一样，在解析外部JavaScript文件（包括下载该文件）时，页面的处理也会暂时停止。如果是在XHTML文档中，也可以省略前面示例代码中结束的</script>标签，例如：

```
<script type="text/javascript" src="example.js"/>
```

但是，不能在 HTML 文档使用这种语法。原因是这种语法不符合 HTML 规范，而且也得不到某些浏览器——尤其是 IE——的正确解析。

> 按照惯例，外部 JavaScript 文件带有.js 扩展名。但这个扩展名不是必需的，因为浏览器不会检查包含 JavaScript 的文件的扩展名。这样一来，使用 JSP、PHP 或其他服务器端语言动态生成 JavaScript 代码也就成为了可能。

需要注意的是，带有src属性的<script>元素不应该在其<script>和</script>标签之间再包含额外的JavaScript代码。

另外，通过<script>元素的src属性还可以包含来自外部域的JavaScript文件。这一点既使<script>元素倍显强大，又让它备受争议。在这一点上，<script>与<img>元素非常相似，即它的src属性可以是指向当前HTML页面所在域之外的某个域中的URL，例如：

```
<script type="text/javascript" src="http://www.somewhere.com/afile.js"></script>
```

这样，位于外部域中的代码也会被加载和解析，就像这些代码位于加载它们的页面中一样。利用这一点就可以在必要时通过不同的域来提供 JavaScript 文件。不过，在访问自己不能控制的服务器上的 JavaScript 文件时则要多加小心。如果不幸遇到了怀有恶意的程序员，那他们随时都可能替换该文件中的代码。因此，如果想包含来自不同域的代码，则要么你是那个域的所有者，要么那个域的所有

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

者值得信赖。

无论如何包含代码，浏览器都会按照<script>元素在页面中出现的先后顺序对它们依次进行解析。换句话说，在第一个<script>元素包含的代码解析完成后，第二个<script>包含的代码才会被解析，然后才是第三个、第四个……。

2.1.1 标签的位置

按照惯例，所有<script>元素都应该放在页面的<head>元素中，例如：

```
<html>
  <head>
    <title>Example HTML Page</title>
    <script type="text/javascript" src="example1.js"></script>
    <script type="text/javascript" src="example2.js"></script>
  </head>
  <body>
    <!— 这里放内容 -->
  </body>
</html>
```

这种做法的目的就是把所有外部文件（包括CSS文件和JavaScript文件）的引用都放在相同的地方。可是，在文档的<head>元素中包含所有JavaScript文件，意味着必须等到全部JavaScript代码都被下载、解析和执行完成以后，才能开始呈现页面的内容（浏览器在遇到<body>标签时才开始呈现内容）。对于那些需要很多JavaScript代码的页面来说，这无疑会导致浏览器在呈现页面时出现明显的延迟，而延迟期间的浏览器窗口中将是一片空白。为了避免这个问题，现代Web应用程序一般都把全部JavaScript引用放在<body>元素中，放在页面的内容后面，如下例所示：

```
<html>
  <head>
    <title>Example HTML Page</title>
  </head>
  <body>
    <!-- <!— 这里放内容 -->
    <script type="text/javascript" src="example1.js"></script>
    <script type="text/javascript" src="example2.js"></script>
  </body>
</html>
```

这样，在解析包含的 JavaScript 代码之前，页面的内容将完全呈现在浏览器中。而用户也会因为浏览器窗口显示空白页面的时间缩短而感到打开页面的速度加快了。

2.1.2 延迟脚本

HTML4.01 为<script>标签定义了defer属性。这个属性的用途是表明脚本在执行时不会影响页面的构造。也就是说，脚本会被延迟到整个页面都解析完毕后再运行。因此，在<script>元素中设置defer属性（如下面的例子所示），实际上与上一节中介绍的把<script>元素放在页面最底部的效果是一样的。

```
<html>
  <head>
    <title>Example HTML Page</title>
    <script type="text/javascript" defer="defer" src="example1.js"></script>
    <script type="text/javascript" defer="defer" src="example2.js"></script>
  </head>
  <body>
```

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章


```
    <!— 这里放内容 -->
  </body>
</html>
```

在这个例子中，虽然我们把<script>元素放在了文档的<head>元素中，但其中包含的脚本将延迟到浏览器遇到</html>标签后再执行。

不过，问题是并非所有浏览器都支持defer属性。IE和Firefox 3.1 是目前唯一支持defer属性的主流浏览器。其他浏览器则会忽略这个属性，不延迟脚本的执行。

2.1.3　在 XHTML 中的用法

可扩展超文本标记语言，即XHTML（Extensible HyperText Markup Language），是将HTML作为XML的应用而重新定义的一个标准。编写XHTML代码的规则要比编写HTML严格得多，而且直接影响能否在嵌入JavaScript代码时使用<script/>标签。以下面的代码块为例，虽然它们在HTML中是有效的，但在XHTML中则是无效的：

```
<script type="text/javascript">
    function compare(a, b) {
        if (a < b) {
            alert("A is less than B");
        } else if (a > b) {
            alert("A is greater than B");
        } else {
            alert("A is equal to B");
        }
    }
</script>
```

在HTML中，有特殊的规则用以确定<script>元素中的哪些内容可以被解析，但这些特殊的规则在XHTML中不适用。这里比较语句a < b中的小于号（<）在XHTML中将被当作开始一个新标签来解析。但是作为标签来讲，小于号后面不能跟空格，因此就会导致语法错误。

避免在XHTML中出现类似语法错误的方法有两个。一是用相应的HTML实体（<）替换代码中所有的小于号（<），替换后的代码类似如下所示：

```
<script type="text/javascript">
    function compare(a, b) {
        if (a &lt; b) {
            alert("A is less than B");
        } else if (a > b) {
            alert("A is greater than B");
        } else {
            alert("A is equal to B");
        }
    }
</script>
```

虽然这样可以让代码在 XHTML 中正常运行，但却导致代码不好理解了。为此，我们可以考虑采用另一个方法。

保证让相同代码在 XHTML 中正常运行的第二个方法，就是用一个 CData 片段来包含 JavaScript 代码。在 XHTML（XML）中，CData 片段是文档中的一个特殊区域，这个区域中可以包含不需要解析的任意格式的文本内容。因此，在 CData 片段中就可以使用任意字符——小于号当然也没有问题，而且不会导致语法错误。引入 CData 片段后的 JavaScript 代码块如下所示：

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章


```
<script type="text/javascript"><![CDATA[
    function compare(a, b) {
        if (a < b) {
            alert("A is less than B");
        } else if (a > b) {
            alert("A is greater than B");
        } else {
            alert("A is equal to B");
        }
    }
]]></script>
```

在兼容 XHTML 的浏览器中，这个方法可以解决问题。但实际上，还有不少浏览器不兼容 XHTML，因而不支持 CData 片段。怎么办呢？再使用 JavaScript 注释将 CData 标记注释掉就可以了：

```
<script type="text/javascript">
//<![CDATA[
    function compare(a, b) {
        if (a < b) {
            alert("A is less than B");
        } else if (a > b) {
            alert("A is greater than B");
        } else {
            alert("A is equal to B");
        }
    }
//]]>
</script>
```

这种格式在所有现代浏览器中都可以正常使用。虽然有几分 hack 的味道，但它能通过 XHTML 验证，而且对 XHTML 之前的浏览器也会平稳退化。

2.1.4 不推荐使用的语法

在最早引入<script>元素的时候，该元素与传统HTML的解析规则是有冲突的。由于要对这个元素应用特殊的解析规则，因此在那些不支持JavaScript的浏览器——最典型的是Mosaic——中就会导致问题。具体来说，不支持JavaScript的浏览器会把<script>元素的内容直接输出到页面中，因而会破坏页面的布局和外观。

Netscape 与 Mosaic 协商并提出了一个解决方案，让不支持<script>元素的浏览器能够隐藏嵌入的 JavaScript 代码。这个方案就是把 JavaScript 代码包含在一个 HTML 注释中，像下面这样：

```
<script><!--
    function sayHi(){
        alert("Hi!");
    }
//--></script>
```

给脚本加上HTML注释后，Mosaic等浏览器就会忽略<script>标签中的内容；而那些支持JavaScript的浏览器在遇到这种情况时，则必须进一步确认其中是否包含需要解析的JavaScript代码。

虽然这种注释 JavaScript 代码的格式得到了所有浏览器的认可，也能被正确解释，但由于所有浏览器都已经支持 JavaScript，因此也就没有必要再使用这种格式了。

2.1.5 嵌入代码与外部文件

在 HTML 中嵌入 JavaScript 代码虽然没有问题，但一般认为最好的做法还是尽可能使用外部文件

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

来包含 JavaScript 代码。不过，并不存在必须使用外部文件的硬性规定，但支持使用外部文件的人多会强调如下优点。

可维护性——遍及不同HTML页面的JavaScript会造成维护问题。但把所有JavaScript文件都放在一个文件夹中，维护起来就轻松多了。而且开发人员因此也能够在不触及HTML标记的情况下，集中精力编辑JavaScript代码。

可缓存——浏览器能够根据具体的设置缓存链接的所有外部JavaScript文件。也就是说，如果有两个页面都使用同一个文件，那么这个文件只须下载一次。因此，最终结果就是能够加快页面加载的速度。

可适应未来——通过外部文件来包含JavaScript无须使用前面提到XHTML或注释hack。HTML和XHTML包含外部文件的语法是相同的。

2.2 文档模式

IE5.5 引入了文档模式的概念，而这个概念是通过使用文档类型（doctype）切换实现的。最初的两种文档模式是：**混杂模式**（quirks mode）[①]和**标准模式**（standards mode）。混杂模式会让IE的行为与（包含非标准特性的）IE5 相同，而标准模式则让IE的行为更接近标准行为。虽然这两种模式主要影响CSS内容的呈现，但在某些情况下也会影响到JavaScript的解释执行。本书将在必要时再讨论这些因文档模式而影响JavaScript执行的情况。

在IE引入文档模式的概念后，其他浏览器也纷纷效仿。在此之后，IE又提出一种所谓的**准标准模式**（almost standards mode）。这种模式下的浏览器特性有很多都是符合标准的，但也不尽然。不标准的地方主要体现在处理图片间隙的时候（在表格中使用图片时问题最明显）。

如果在文档开始处没有发现文档类型声明，则所有浏览器都会默认开启混杂模式。但采用混杂模式不是什么值得推荐的做法，因为不同浏览器在这种模式下的行为差异非常大，如果不使用某些 hack 技术，跨浏览器的行为根本就没有一致性可言。

对于标准模式，可以通过使用下面任何一种文档类型来开启：

```
<!-- HTML 4.01 严格型 -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">

<!-- XHTML 1.0 严格型 -->
<!DOCTYPE html PUBLIC
"-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

而对于准标准模式，则可以通过使用过渡型（transitional）或框架型（frameset）文档类型来触发，如下所示：

```
<!-- HTML 4.01 过渡型 -->
<!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<!-- HTML 4.01 框架集型 -->
<!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML 4.01 Frameset//EN"
```

① 这里quirks mode的译法源自Firefox 3.5.5 中文版。——译者注

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章


```
"http://www.w3.org/TR/html4/frameset.dtd">

<!-- XHTML 1.0 过渡型 -->
<!DOCTYPE html PUBLIC
"-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- XHTML 1.0 框架集型 -->
< !DOCTYPE html PUBLIC
"-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

准标准模式与标准模式非常接近，它们的差异几乎可以忽略不计。因此，当有人提到“标准模式”时，有可能是指这两种模式中的任何一种。而且，检测文档模式（本书后面将会讨论）时也不会发现什么不同。

IE8 又引入了一种新的文档模式，叫做**超级标准模式**。超级文档模式可以让IE以其所有版本中最符合标准的方式来解释网页内容。总的来看，混杂模式让IE像IE5，标准模式使用IE7 的呈现引擎，而超级标准模式则是IE8 的默认文档模式——不过，在IE8 中使用下面这个特殊的<meta>值可以关闭其默认文档模式：

```
<meta http-equiv="X-UA-Compatible" content="IE=7" />
```

其中，content 属性中 IE 的值用于指定使用哪个版本的呈现引擎来呈现页面。设计这个值的目的就是为了向后兼容那些专门为老版本的 IE 设计的站点和页面。

与准标准模式类似，超级标准模式与标准模式一般也不会有什么差别。因此，本书在提到**标准模式**时，指的是除混杂模式之外的其他所有模式。

2.3 <noscript>元素

早期浏览器都面临一个特殊的问题，即当浏览器不支持JavaScript时如何让页面平稳地退化。对这个问题的最终解决方案就是创造一个<noscript>元素，用以在不支持JavaScript的浏览器中显示替代的内容。这个元素可以包含能够出现在文档<body>中的任何HTML元素——<script>元素除外。包含在<noscript>元素中的内容只有在下列情况下才会显示出来：

- 浏览器不支持脚本；
- 浏览器支持脚本，但脚本被禁用。

符合上述任何一个条件，浏览器都会显示<noscript>中的内容。而在除此之外的其他情况下，浏览器不会呈现<noscript>中的内容。

请看下面这个简单的例子：

```
<html>
  <head>
    <title>Example HTML Page</title>
    <script type="text/javascript" defer="defer" src="example1.js"></script>
    <script type="text/javascript" defer="defer" src="example2.js"></script>
  </head>
  <body>
    <noscript>
      <p>本页面需要浏览器支持（启用）JavaScript。
    </noscript>
  </body>
</html>
```

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

这个页面会在脚本无效的情况下向用户显示一条消息。而在启用了脚本的浏览器中，用户永远也不会看到它——尽管它是页面的一部分。

2.4 小结

把JavaScript插入到HTML页面中要使用<script>元素。使用这个元素可以把JavaScript嵌入到HTML页面中，让脚本与标记混合在一起；也可以包含外部的JavaScript文件。而我们需要注意的地方有：

- 这两种使用方式都要求把type属性设置为text/javascript，以表明使用的脚本语言是JavasScript。
- 在包含外部JavaScript文件时，必须将src属性设置为指向相应文件的URL。而这个文件既可以是与包含它的页面位于同一个服务器上的文件，也可以是其他任何域中的文件。
- 所有<script>元素会按照它们在页面中出现的先后顺序依次被解析。只有在解析完前面<script>元素中的代码之后，才会开始解析后面<script>元素中的代码。
- 浏览器在呈现后面的页面内容之前，必须先解析完前面<script>元素中的代码。为此，一般要把<script>元素放在页面的末尾，放在页面内容之后和结束的</body>标签之前。
- 在IE中，可以通过设置defer属性让浏览器在呈现完文档之后再执行脚本。虽然defer属性是HTML 4.01 中规定的，但只有IE支持该属性。

另外，使用<noscript>元素可以指定在不支持脚本的浏览器中显示的替代内容。但在启用了脚本的情况下，浏览器不会显示<noscript>元素中的任何内容。