**本章内容：**

传统技术：document.write 和 innerHTML.

深入剖析DOM 方法：createElement、createTextNode、appendChild 和 insertBefore.

**一些传统方法：**

**document.write:**

document 对象的write() 方法可以方便快捷地把字符串插入到文档里。

即使把document.write() 语句挪到外部函数中，也需要在标记的<body>部分使用<script> 标签才能调用那个函数。

MIME 类型 application/xhtml+xml 与 document.write 不兼容。

尽量避免使用 document.write 方法。

**innerHTML 属性：**

可以用来读、写某给定元素里的HTML 内容。

使用innerHTML 读取：



使用 innerHTML 写入：  
 ***function wInnerHTML(){***

***var testdiv = document.getElementById("testdiv");***

***testdiv.innerHTML = "<p>This is <em>another</em> content.</p>";***

***}***

一旦使用innerHTML 写入，该部分内容将全部被替换。

**DOM 方法：**

createElement 方法：

语法：

***document.createElement(nodeName);***

***document.createElement(“p”);***

appendChild 方法：

语法：

***parent.appendChild(child);***

createTextNode 方法：

使用此方法将一些文本放入创建的元素节点。

语法：

***document.createTextNode(text);***

***document.createTextNode(“Hello world”);***

**在已有元素前面插入一个元素：**

insertBefore() 方法：

语法：

***parentElement.insertBefore(newElement, targetElement);***

**parentElement: 目标元素的父元素；**

**newElement: 想要插入的新元素；**

**targetElement: 目标元素，想要插到哪个元素之前。**

**在现有元素后插入一个元素：**

DOM 并没有 insertAfter() 方法。

自己编写：

***function insertAfter(newElement, targetElement){***

***var parent = targetElement.parentNode;***

***if (parent.lastChild == targetElement){***

***parent.appendChild(newElement);***

***}else{***

***parent.insertBefore(newElement, targetElement.nextSibling);***

***}***

***}***

如果目标元素是 其父元素的最后一个子元素，即使用appendChild 方法，插入父元素；如果不是，则将新元素插入目标元素下一个兄弟元素之前，下一个兄弟元素即是目标元素的 nextSibling 属性。

**Ajax**

Ajax 用于概括异步加载页面内容的技术。主要优势就是对页面的请求以异步方式发送到服务器。而服务器不会用整个页面来响应请求，它会在后台处理请求，与此同时用户还能继续浏览页面并与页面交互。脚本可以按需加载和创建页面内容，而不会打断用户的浏览体验。

Ajax 依赖于JavaScript，所以可能会有浏览器不支持它，而搜索引擎的爬虫程序也不会抓取到有关内容。

**XMLHttpRequest 对象：**

Ajax 技术的核心就是XMLHttpRequest 对象。这个对象充当着浏览器中的脚本（客户端）与服务器之间的中间人的角色。

不同浏览器实现XMLHttpRequest 对象的方式不太一样，为了保证跨浏览器，你不得不为同样的事情写不同的代码分支。

XMLHttpRequest 对象有许多方法。其中最有用的是 open 方法，用来指定服务器上将要访问的文件，指定请求类型：GET、POST 或 SEND。这个方法的第三个参数用于指定请求是否以异步方式发送和处理。

服务器在向XMLHttpRequest 对象发回相应时，该对象有许多属性可用，浏览器会在不同阶段更新 readyState 属性的值，它有5个可能的值：

**0表示未初始化；**

**1表示正在加载；**

**2表示加载完毕；**

**3表示正在交互；**

**4表示完成。**

只要readyState 属性值变成了4， 就可以访问服务器发送回来的数据了。

访问服务器发送回来的数据要通过两个属性完成：

一个是 **responseText** 属性，这个属性用于保存文本字符串形式的数据。

另一个是 **responseXML** 属性，用于保存Content-Type 头部中指定为 “text/xml” 的数据，其实是一个 DocumentFragment 对象。可以使用各种DOM 方法处理这个对象。

Ajax 应用主要依赖于服务器端处理，而非客户端处理。

小结：

使用这些方法的关键是将Web 文档视作节点树。

用createElement 或 createTextNode 方法刚刚创建的节点只是JavaScript 世界里的孤儿。要利用 appendChild 或 insertBefore 方法，把这些 DocumentFragment 对象插入某个文档的节点树，让它们在浏览器窗口呈现。