### Word/VS/Edge宿主的“面向字符的.docx/.txt/.html/.css文件宿客”（字符语法主导）

表 6‑20 Word/VS/Edge宿主的“面向字符的.docx/.txt/.html/.css”（字符语法主导）

|  |  |  |
| --- | --- | --- |
| ASP.Net MVC架构的视图V的创新研发：“客户端的Edge浏览器宿主的Blazor宿主的视图V宿客”的C#语言思维  （本表聚焦Word编辑好的.docx另存成为.html/.css文件，然后，VS/Edge Developer Toos可以微调的） | ASP.Net MVC架构的控制C的创新研发：“互动的Kestrel/IIS宿主的控制C宿客”的C#语言思维 | ASP.Net MVC架构的模型M的创新研发：“服务端的SS/ChatGPT/Qwen宿主的模型M宿客”的C#语言思维 |
| 基于XML规范的多媒体应用有HTML、SVG/XAML、X3D等，可形成媒体的整体解决方案，本节将系统叙述XML超文本应用--HTML的基本原理，为超文本的具体应用奠定原理基础。本章主要以代码的方式叙述HTML，建议结合5.2节所述，进行上机实践探究如下。   1. 理解HTML是XML超文本媒体应用。 2. 掌握HTML常用元素的功能和用法 3. 理解HTML文档的布局机制。 4. 理解HTML文档内容与格式，掌握HTML文档CSS样式的基本原理。 5. 理解HTML文档内容、格式与功能，了解HTML结合Jscript、C#等编程。 | | |

1. **HTML****：基于XML标准的网页超文本**

文本是最基本的媒体，文本编辑处理是最基本的媒体技术。文本编辑理论及其实现软件正日益发展并日趋标准化，XML在超文本领域的具体应用--HTML语言则是描述超文本的Web标准（http://www.w3.org/TR/xhtml1/）。HTML超文本从媒体的角度称为超文本或超文本为主体的文档，从Web的角度又可称为Web页、网页，在此根据媒体叙述的特性，主要使用HTML超文本一词。

HTML是eXtensible HyperText Markup Language的首字母缩写，中文意译为可扩展超文本标记语言，是一个描述超文本的标记语言，当前最高版本是1.1。HTML可以将各种媒体有机地Web方式地集成在一起，并以超链接的方式综合表达信息（适合大脑的联想功能）。HTML文本技术拥有常见文本技术（如当前流行的文本处理技术Word等）的基本功能，还具有可交换性、交互性、可扩展性、易维护性等优点，可完美集成各种媒体、脚本语言、程序组件等实现强大的Web应用。已经发展成为了面向文本数据的无可争议的媒体平台（参见**Error! Reference source not found.**节）。

小练习：HTML是HTML的基于XML的发展成果，HTML诞生于XML之先，所以并不完全遵守XML规范，通过网络探究HTML、HTML的发展历史。

HTML和XML在语法、结构等众多方面都具有相似性，只不过HTML是XML在超文本领域的具体应用，而XML是关于HTML、SVG/XAML、X3D等具体应用的元语言，将HTML与XML对比理解，是掌握HTML的捷径。

HTML文档属于互动式二维文本视觉媒体，基本原理可参见**Error! Reference source not found.**中与互动式文本紧密相关的内容，特别是二维布局、时间特性、交互性等。该部分则将以代码的方式叙述HTML，HTML功能强大、元素特性丰富，应该注重掌握HTML的基本原理，并结合Expression Web软件（将在**Error! Reference source not found.**详述）进行实践锻炼。

遵守XML语法，同时具有如下特点：

* **元素是预定义的**
* 主要就是理解预定义的元素的用法。
* **自定义特性规范**
* Data-特性
* **HTML的格式标准：CSS**
* **HTML的编程标准：DOM与JS\C#**
* **HTML代码文件**

1. **HTML文档浏览**

可以浏览HTML文档的软件很多，例如IE，Firefox浏览器等等，当前最常使用的是IE浏览器，**Error! Reference source not found.**节图 6.3‑1就是IE浏览HTML文档的一个示例。IE浏览器右键菜单中还提供丰富的功能，不过IE已广为使用，在此不赘述。

IE可全屏或非全屏浏览HTML文档，可参见**Error! Reference source not found.**节。

1. **HTML文档编辑**

HTML文档是文本文档，所以原则上可以使用任何文本编辑器编辑（如Windows自带的记事本程序），但为了提高编辑效率，一般使用专业的HTML文档编辑软件进行编辑（如Expression Web，将在**Error! Reference source not found.**详述）。

HTML文档包含布局、内容、样式、功能等信息。编辑HTML文档时，一般首先形成整个HTML文档的整体布局，然后根据布局填充文本、图像、动画等内容，通过CSS赋予样式，生成超链接等功能，并结合Jscript、C#等语言获得强大的功能，从而形成强大的HTML应用。

注：HTML文档编辑又大众化地称为网页编辑。本书主要选用HTML文档编辑的说法，必要时也选用HTML文档/网页编辑、网页编辑的说法。

1. **HTML文档基本结构**

HTML本质上是基于XML为规范的标记语言，该语言完全遵守XML语法，可编写并形成HTML超文本文档。一个HTML文档往往对应一个计算机文件，文件扩展名是".htm" 或".html"， MIME类型是 "text/html"。

下面是一个简单HTML文档的示例（参见示例文件HTML\_BasicStructure.htm），反映了HTML文档的基本结构。在浏览器中的运行浏览结果将如图 6.3‑1所示。

|  |
| --- |
| 1. <?xml version="1.0" encoding=”gb2312”?> **<!-- XML声明-->** 2. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.1//EN" 3. "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"> **<!--该语句声明和引用了HTML架构-->** 4. <html xmlns="http://www.w3.org/1999/xhtml"> **<!--根元素<html>及其名称空间，<html>是HTML文档的根元素。浏览器通过<html>判断该文档是一个HTML文档，并使用HTML架构对该文档进行解析。-->** 5. <head>**<!--<head>用来为文档定义一些头部信息，例如引用外部CSS文件等，将在相关部分详述。-->** 6. <title>HTML文档示例</title>**<!--<title>用来为文档指定一个标题。标题将在文档一开始下载时就显示在浏览器窗口的标题栏中，因为标题一般体现了该文档的主要内容，这样浏览者就可快速了解到该文档的主要内容。-->** 7. </head> 8. <body>**<!--<body>是HTML文档的主体部分，用来嵌套其他HTML子元素形成强大的超文本应用。HTML文档中<html>、<head>、<title>、<body>四个元素定义了一个HTML文档的基本结构，<html>、<head>、<title>三个元素的功能和用法不会有太大变化，而<body>则可嵌套大量的其他元素来获得HTML的强大功能，是HTML文档编辑的主要任务。-->** 9. <p>网页信息通过HTML文档进行超文本传播。</p>**<!--该语句生成一段文本-->** 10. </body> 11. </html> |

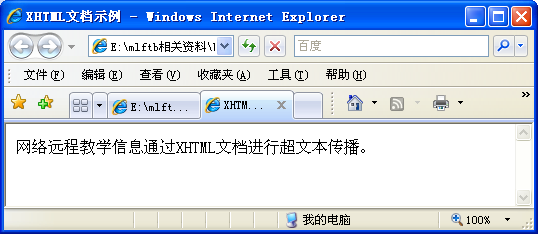


图 6.3‑1 HTML\_BasicStructure.htm在浏览器中的运行浏览结果

1. **HTML文档架构及其元素概述**

HTML是一个Web标准，HTML具有标准的预定义的架构（而XML是自定义架构），掌握HTML的关键就是掌握HTML架构中预定义的元素及其特性的功能和用法， HTML架构中预定义的元素及其特性丰富，功能强大，在此主要掌握其基本原理，然后可基于基本原理，参考HTML架构标准（http://www.w3.org/TR/xhtml1/），自主上机实验，全面掌握HTML。

为了更好地理解HTML元素，在此将HTML所有常用元素归类概述如下，以作大概了解。实际应用时，应该根据正编辑的HTML文档的需求而考虑元素及其属性选用，如果不知道元素的具体用法，则应该参考HTML标准。如果以软件为辅助工具，则更可大大地提高效率，使用Expression Web软件编辑HTML将在**Error! Reference source not found.**详述。

* 1. 文档基本结构（Document Structure）类元素

html、head、title、body。形成HTML文档的基本结构。已在本节进行过叙述。

* 1. 布局（Layout）类元素

ol、ul、li。生成有编号或无编号列表。将在**Error! Reference source not found.**节详述。

p、br。p生成段落式换行、br生成非段落式换行。将在**Error! Reference source not found.**节详述。

div，span。div生成块状区域，span生成内嵌的非块状区域。将在**Error! Reference source not found.**节详述。

table、tr、td。生成表格或表格式布局区域。将在**Error! Reference source not found.**节详述。

XTHML可以结合CSS实现强大的布局、格式、样式，将在**Error! Reference source not found.**节详述。

* 1. 外部媒体引用（External Reference）类元素

img。引用外部图像。将在四、节详述。

frameset、frame。生成框架而引用外部的HTML文档。将在四、节详述。

* 1. 超链接（Hyperlink）类元素

a。生成超链接。将在九、节详述。

* 1. 表单（Form）类元素

form、textarea、select、option、input。生成表单基本结构。将在八、节详述。

1. **HTML文档内容、格式、功能**

HTML文档可理解为内容、格式、功能三类主要信息。内容定义是HTML文档编辑的首要任务，内容定义后可进一步为内容赋予格式，为文档添加功能。HTML文档内容通过元素/标记定义，关键是了解HTML所有元素/标记的语法和基本功能，然后通过软件辅助（例如VS）具体实现。

HTML文档可理解为内容、格式、功能三类主要信息。前面叙述了HTML文档内容定义，并通过CSS格式化HTML文档，本节概述HTML文档功能，例如超链接、通过表单与服务器通信等，此外，HTML文档还可通过JScript，C#，Java等程序语言扩展强大的功能，但涉及系统的编程知识，根据立体化课程观，将在后续课程详述。

1. **HTML的格式标准：CSS**

HTML文档和XML文档一样选用CSS样式技术（参见**Error! Reference source not found.**节）格式化文档，所以HTML文档中可能含有外部CSS引用、内部CSS、嵌入式CSS等内容。将在**Error! Reference source not found.**节详述。

1. **HTML的编程标准：DOM与JS\C#**

上述HTML\_BasicStructure.htm的DOM的逻辑关系将如图 6.3‑2所示。HTML DOM可被程序语言（如Jscript、C#等）动态操作，从而实现强大的HTML应用，将在具体涉及时详述。（这样也就是通过XML初始生成初始内容，然后在运行时动态修改XML）



图 6.3‑2. HTML\_BasicStructure.htm的DOM图解

1. **HTML布局**
2. **布局系统：元素布局与CSS布局**

由于兼容HTML历史的原因，HTML布局以元素/标记和CSS样式共同实现，在此主要叙述了元素/标记相关的布局。CSS布局则将在九、节详述。

1. **文档尺寸、坐标系统、布局规则、元素占位**

HTML文档面向屏幕动态互动式显示，其尺寸随网站访问者的屏幕分辨率的改变而动态改变，还将在**Error! Reference source not found.**节结合上机实践体验。如果HTML签入在其他文档中（典型地如一个HTML文档嵌入在另一个HTML文档中，将在四、节涉及），其尺寸则是嵌入时作为对象而设置的尺寸。

一个复杂的HTML文档将由许多HTML元素组成，HTML元素占位、坐标系统、布局规则(流式、绝对、相对定位)主要通过CSS机制实现，还将在九、节详述。

还必须提及的是，由于兼容历史的原因，HTML一些定义文本的元素（如<p>,<list>,<table>等）具有直接样式化文本的功能，但这些元素的种类是有限的，只能对文档的样式进行微弱的控制，为了对HTML文档格式进行强大的控制，如字体、背景、文本、布局 、定位等等，必须使用CSS。

1. **HTML文本**

HTML是XML在超文本领域的具体应用，所以文本定义是HTML的最基本功能。字符形成段落，段落形成文档，整个文档就是一页，如果一页超过满屏的高度或宽度，将提供滚动条滚动浏览，所以，HTML文档没有分页的概念。

HTML是为定义(超)文本文档而诞生的，所以HTML设计为并不提供专门定义文本的元素，而是作为一些元素的值而实现。文本主要由段落类元素p，列表类元素ol、ul、li，表格类元素table、tr、td，分块元素div，内嵌元素span等格式类元素定义并初步格式化，已在**Error! Reference source not found.**节结合布局进行过详述，在此不赘述。上述元素定义文本后，可进一步通过CSS样式机制实现字体、背景、文本、布局 、定位等强大的格式功能，将在九、节详述。

可以通过表格、div、list元素，也可通过css.

HTML文档属于视觉媒体，视觉媒体布局基本原理参见**Error! Reference source not found.**节。HTML文档以描述文本为主导，所以又具有面向文本的布局特点，最基本的布局思路是字符形成段落，段落形成文档，整个文档就是一页，并可结合列表，表格等形成更复杂的布局，这些都通过相应的元素实现，下面是一个包含了这些元素的HTML文档示例（参见示例文件HTML\_Quiz.htm），描述的是由试题组成的试题库，运行结果将如图 6.3‑3所示。

|  |
| --- |
| 1. <?xml version="1.0" encoding="utf-8"?> 2. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.1//EN" 3. "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"> 4. <html xmlns="http://www.w3.org/1999/xhtml"> 5. <head> 6. <title>HTML文档示例</title> 7. </head> 8. <body> <!--段落类，列表类，表格类元素都只能出现在<body>内--> 9. <table style="border-width:1px;border-style:solid;width:100%"> <!--<table>、<tr>、<td>是表格类元素。可以使用一个<table>嵌套多个<tr>,一个<tr>又嵌套多个<td>来获得表格格式。<table>定义一个表格，<table>中嵌套的每一个<tr>为表格定义一行，<tr>中嵌套的每一个<td>为表格定义一列，行和列交叉得到单元格并可通过<tr>、<td>的属性定义单元格的合并，每个单元格都可指定边框的显示特性。表格可以互相嵌套，单元格又可包含其它元素，从而获得样式整洁的的HTML文档，所以表格类元素是非常重要的元素，几乎任何HTML文档都可以考虑使用表格类元素来定义文本。--> 10. <tr> 11. <td style="border-width:1px;border-style:solid;width:100%"> 12. <p>下面可用作元素定界符的字符是：</p> <!--<p>是段落类元素，<p>的内容自成一段--> 13. <ol> <!--<ul>、<ol>、<li>是列表类元素，可以使用<ul>或<ol>嵌套<li>来获得列表格式。<ul>中的每一列表项（<li>）在显示时将自动加上项目符号，项目符号的类型可选择；<ol>中的每一列表项（<li>）在显示时将自动加上序号，序号的类型可选择。在此，<ol>嵌套<li>定义一个有次序的列表，每个<li>定义一个列表项。功能相似的还有<ul>嵌套<li>可以定义一个无次序的列表--> 14. <li>&lt;</li> 15. <li>@</li> 16. <li>$</li> 17. <li>?</li> 18. </ol> 19. <p>难度系数：0.5</p> 20. <p>1</p> 21. <p>可用于元素定界符的是&lt;和&gt;。</p> 22. </td> 23. </tr> 24. <tr> 25. <td style="border-width:1px;border-style:solid;width:100%"> 26. <p>下面可用于字符引用的字符是：</p> 27. <ol> 28. <li>&amp;</li> 29. <li>)</li> 30. <li>!</li> 31. <li>(</li> 32. </ol> 33. <p>0.6</p> 34. <p>1</p> 35. <p>可用于字符引用的是&amp;。</p> 36. </td> 37. </tr> 38. </table> 39. </body> 40. </html> |

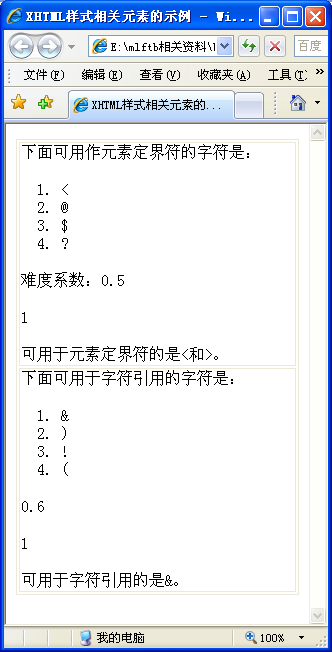


图 6.3‑3 示例文件HTML\_Quiz.htm的运行结果

下面以HTML\_Quiz.htm文档为例概述HTML文档的布局特征及其相关的元素。

1. **HTML图像**

HTML设计为引用外部的图像文件到当前文档中，并布局。

HTML可以直接定义文本，从媒体完整性的视角，还可能需要引用文本（即外部HTML文件）、图像、视频、二维图形动画、三维图形动画、波形音频、MIDI器乐等外部媒体；从功能性的视角，还可能需要引用功能性组件。

HTML可以直接定义文本，同时可以集成外部文本、图像、图形、视频、动画、波形音频、MIDI器乐等媒体和组件。这些媒体和组件格式繁多、变化复杂，有些适合用二进制数据表示，有些适合用文本数据表示，直接使用HTML来生成这些媒体和组件不是最好的解决办法，所以HTML采用了引用的机制来实现这些媒体和组件在HTML文档中的集成。

如果浏览HTML文档的浏览器自身能够直接播放欲引用的媒体，就可通过HTML的相关元素直接引用这些媒体。要知道浏览器能够直接播放哪些媒体，则必须查看所使用的浏览器的功能，并且功能是随浏览器的发展而发展的，例如IE 6.0支持直接播放在HTML文档中所引用的.gif，.jpeg等常见格式的图像、.mp3，.midi等常见格式的音频、外部HTML文档（.htm或.html格式）等。

下面是引用了.gif图像和.mp3背景音乐的HTML文档示例：

|  |
| --- |
| 1. <?xml version="1.0" encoding="gb2312"?> 2. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.1//EN" 3. "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"> 4. <html xmlns="http://www.w3.org/1999/xhtml"> 5. <head> 6. <title>示例</title> 7. <bgsound src="sample.mp3" loop="-1"> <!--<bgsound>引用一个外部声音文件成为背景声音，声音文件的源由src特性指定，loop等特性指定声音如何播放。--> 8. </head> 9. <body> 10. <img src="sampe.gif" width="28" height="28" border="0"> <!--<img>引用一个外部图像文件，图像文件的源由src特性指定，width等特性指定图像的格式。--> 11. </body> 12. </html> |

1. **HTML视频**

HTML设计为引用外部的视频文件到当前文档中，并布局。

.<video src="../\_video/EDT.mp4" controls>

<p>Your browser does not support HTML5 video.</p>

</video>

其中：

1.src指定了节目的源；

2.controls表示带上浏览器自带的播放器控制；

3.<p></p>内的是页面遇到不支持的浏览器时显示的文本内容；

1. **HTML 2D**

HTML设计为引用外部的图像文件到当前文档中，并布局。

引用外部媒体时，如果浏览HTML文档的浏览器自身不能直接播放这些媒体（如在HTML文档中引用.wmv、.rmvb、.mov等流式视频，.swf、.xaml等二维图形动画.x3d三维图形动画），必须首先引用相关的组件，再由组件引用该媒体。下面是引用了.wmv、.swf、.xaml、.x3d的一个HTML文档示例（参见示例文件HTML\_MediaReference.htm）。

|  |
| --- |
| 1. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> 2. <html xmlns="http://www.w3.org/1999/xhtml"> 3. <head> 4. <meta content="text/html; charset=utf-8" http-equiv="Content-Type" /> 5. <title>Untitled 3</title> 6. <style type="text/css"> 7. .style1 { 8. text-align: center; 9. } 10. </style> 11. </head> 12. <body> 13. <p class="style1"> 14. <!--下面<object>元素引用.wmv视频--> 15. <object id="media1" type="video/x-ms-wmv" style="width: 295px; height: 355px"> 16. <param name="filename" value="sample.wmv" /> 17. </object> 18. </p> 19. <p class="style1"> 20. <!--下面<object>元素引用.xaml或其压缩包.xap二维动画--> 21. <object data="data:application/x-silverlight," type="application/x-silverlight-3" width="100%" height="100%"> 22. <param name="source" value="sample.xap"/> 23. <param name="onerror" value="onSilverlightError" /> 24. <param name="background" value="white" /> 25. <param name="minRuntimeVersion" value="$silverlight3sdkversion$" /> 26. <param name="autoUpgrade" value="true" /> 27. <a href="http://go.microsoft.com/fwlink/?LinkID=149156&v=$silverlight3sdkversion$" style="text-decoration: none;"> 28. <img src="http://go.microsoft.com/fwlink/?LinkId=108181" alt="Get Microsoft Silverlight" style="border-style: none"/> 29. </a> 30. </object> 31. </p> 32. <p class="style1"> 33. <!--下面<object>元素引用.swf二维动画--> 34. <object id="flash1" data="sample.swf" height="200" type="application/x-shockwave-flash" width="200"> 35. <param name="movie" value="sample.swf" /> 36. </object> 37. </p> 38. <p class="style1"> 39. <!--下面<object>元素引用.x3d三维动画--> 40. <OBJECT 41. classid="clsid:4B6E3013-6E45-11D0-9309-0020AFE05CC8" 42. codeBase="http://www.bitmanagement.com/download/BS\_Contact/cab-xpi/BS\_Contact\_VRML-X3D.exe#Version=7,0,3,1" 43. width="400" 44. height="300" 45. type="application/x-oleobject" 46. > 47. <PARAM NAME="SRC" VALUE="sample.x3d"/> 48. </OBJECT> 49. </p> 50. </body> 51. </html> |

HTML\_MediaReference.htm的运行结果如图 6.3‑4所示。



图 6.3‑4. HTML\_MediaReference.htm的运行结果

1. **HTML 3D**

HTML设计为引用外部的图像文件到当前文档中，并布局。

1. **HTML表单**

HTML专门提供了元素，实现表单的功能。

HTML文档放置在Web服务器端，由客户端的浏览器装载并运行，为了实现浏览器端和服务器端之间的信息交互，HTML文档必须有向Web服务器端提交信息的机制，这通过表单来实现。

HTML表单由<form>嵌套<input>等称为表单域的元素来实现，下面是含有表单的HTML文档示例，可保存为HTML\_Form.htm：

|  |
| --- |
| 1. <?xml version="1.0" encoding="gb2312"?> 2. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.1//EN" 3. "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"> 4. <html xmlns="http://www.w3.org/1999/xhtml"> 5. <head> 6. <title>表单元素示例</title> 7. </head> 8. <body> 9. <form method="post" action="quizForm.ASPX"> <!-- <form>是一个表单元素，一个表单中的内容作为一个整体提交给服务器端程序。method特性指定表单数据以post方法提交，post是提交大量数据的常用方法;action特性指定服务器端接受客户端表单所提交的数据的文件是quizForm.ASPX，该文件接受表单提交的数据后可进行处理，处理机制参看**Error! Reference source not found.** --> 10. <p>1.下面可用作元素定界符的字符是：</p> 11. <p> 12. <input type="radio" value="V5" name="R1">&lt;</input> 13. <input type="radio" value="V6" name="R1">@</input> 14. <input type="radio" value="V7" name="R1">$</input> 15. <input type="radio" value="V8" name="R1">?</input> <!--<input>是表单域元素，其type属性可以分别赋值submit（提交按钮）,reset（重置按钮）,button（普通按钮）,checkbox（复选框）,radio（单选按钮）,text（文本框）,file（文件上载按钮）,password（口令框）。--> 16. </p> 17. <p> 2.下面可用于字符引用的字符是：</p> 18. <p> <select size="1" name="D1"> 19. <option>请选择</option> 20. <option>&amp;</option> 21. <option>)</option> 22. <option>!</option> 23. <option>)</option> 24. </select> </p> 25. <p>欢迎你对网络测试发表观点：</p> 26. <p><textarea rows="3" name="S1" cols="20">你对网络测试的观点是：</textarea></p> 27. <p> 28. <input type="submit" value="交卷" name="B1"> <!--提交整个表单中各表单域数据给服务器端的按钮--> 29. <input type="reset" value="重做" name="B2"> <!--将整个表单中各表单域的值恢复为初始值的按钮--> 30. </p> 31. </form> 32. </body> 33. </html> |

HTML\_Form.htm的运行结果如图 6.3‑5所示：

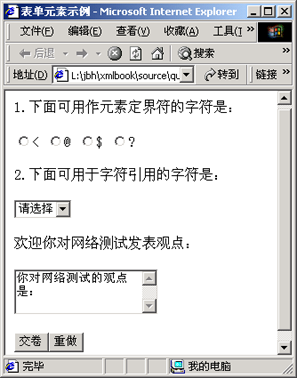


图 6.3‑5. HTML\_Form.htm的运行结果

1. **HTML超链接**
2. **关于超链接**

超链接是HTML的重要特征功能之一，本质上是从一个对象（Anchor：锚）到另一个对象（Target：链接目标）的连接，当单击“锚”时，将跳转到并打开“链接目标”，从而实现全球Web信息的网状互连。由超链接联接在一起的全球Web信息就像是一个世界上最大的网状数据库。

超链接技术诞生之初，“锚”只能是文本，赋予了超链接的文本又称为超文本，随着技术的进步，超链接逐渐发展成为了一种通用的信息跳转技术，当前“锚”除了可以是文本之外，还可以是几乎任何的媒体对象，所以正变换概念为超媒体。

HTML文档中，超链接的“锚”可以是各种对象（例如文本、图形、图像、视频、音频等各种媒体对象），“链接目标”可以是该HTML文档中的任意的预定义位置，也可以是该HTML文档之外的其它文件（可以是任何类型的文件，此时浏览器将根据具体的文件格式运行这些“链接目标”）。

注：打开一个超链接的目标文件时（如超链接的是一个.avi格式的视频文件时），所使用的程序是Windows系统中所注册的关于该格式文件的默认程序，是不完全确定的（如可能注册的是Windows Media Player、Realplayer、QuickTime Player中的一个播放器来播放.avi格式的视频文件），如果想控制具体的程序来打开该格式的文件（如Realplay 10来播放.avi格式的视频文件），必须使用四、节叙述的ActiveX组件的方法。

1. **跳转到其它文件的超链接**

下面是含有跳转到其它文件的超链接的HTML文档示例（参见示例文件HTML\_HyperLink.htm）：

|  |
| --- |
| 1. <?xml version="1.0" encoding="gb2312"?> 2. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.1//EN" 3. "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"> 4. <html xmlns="http://www.w3.org/1999/xhtml"> 5. <head> 6. <title>示例</title> 7. </head> 8. <body> 9. <a href="http://www.w3.org/index.htm">单击这里前往Web联盟网站的index.htm文档</a> <!-- href特性指定了链接目标的URL --> 10. </body> 11. </html> |

1. **跳转到HTML文档（自身或其它）中预定义部分的超链接**

下面是含有跳转到文档的预定义部分的超链接的HTML文档示例（参见示例文件HTML\_HyperLinkName.htm）：

|  |
| --- |
| 1. <?xml version="1.0" encoding="gb2312"?> 2. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.1//EN" 3. "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"> 4. <html xmlns="http://www.w3.org/1999/xhtml"> 5. <head> 6. <title>示例</title> 7. </head> 8. <body> 9. <p><a href="http://www.w3.org/index.htm#xx">单击这里前往web联盟网站index.htm文档中预定义的xx部分</a></p> <!-- href特性指定了链接目标的URL和预定义的xx --> 10. <p><a href="#xx">单击这里跳转到本文档中预定义的xx部分</a> <!-- href特性指定了链接目标是本文档中预定义的xx --> 11. <p/><p/><p/><p/><p/><p/><p/><p/><p/><p/><p/><p/><p/><p/><p/><p/><p/><p/> <p/><p/><p/><p/><p/><p/><p/><p/><p/><p/><p/><p/><p/><p/><p/><p/><p/><p/> <p/> <p/> 12. <p><a name="xx">这里是上面定义的超链接要跳转到的部分</a></p> <!-- name特性指定了预定义名称xx --> 13. </body> 14. </html> |

一个网站中含有多个HTML文件，其中一个是启动文件（一般是default.html，或类似的default.htm、default.aspx等）。通过一个文件启动之后，各个文件之间通过超链接互相跳转链接在一起。所以网站的信息交互机制是面向用户事件互动式驱动的，例如，用户通过鼠标单击超链接(即鼠标事件)、按键盘(即键盘事件)而交互等，而超链接是网站信息交互的基础，HTML文档又面向文本定义，所以HTML从媒体的角度可称为超文本媒体。

1. **HTML引用外部HTML文件**

下面是引用了外部HTML文档的HTML文档示例，可保存为HTML\_IFrame.htm：

|  |
| --- |
| 1. <?xml version="1.0" encoding="gb2312"?> 2. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.1//EN" 3. "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"> 4. <html xmlns="http://www.w3.org/1999/xhtml"> 5. <head> 6. <title>示例</title> 7. </head> 8. <body> 9. <table border="1" " style="border-collapse: collapse" width="100%"> 10. <tr> 11. <td width="100%" colspan="2">网页</td> 12. </tr> 13. <tr> 14. <td width="30%"> 15. <iframe src="HTML\_Contents.htm" width="100%" frameBorder="0"></iframe> <!--<irame>称为在线框架，可用来引用外部的HTML文档，src特性指定被引用的HTML文档源，注意该处的HTML\_Contents.htm和HTML\_IFrame.htm处于同一路径 --> 16. </td> 17. <td width="70%"> 18. <iframe src="HTML\_Main.htm" width="100%" frameBorder="0"></iframe> <!--<irame>称为在线框架，可用来引用外部的HTML文档，src特性指定被引用的HTML文档源，注意该处的HTML\_Main.htm和HTML\_IFrame.htm处于同一路径 --> 19. </td> 20. </tr> 21. </table> 22. </body> 23. </html> |

一个HTML文档引用外部HTML文档后，浏览器窗口被分割为多个框架，每个框架对应于一个HTML文档，每个框架的操作就象操作独立的HTML文档。HTML\_Contents.htm、HTML\_Main.htm、HTML\_IFrame.htm分别被浏览的结果比较如图 6.3‑6所示：

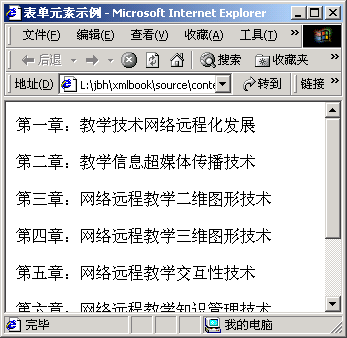
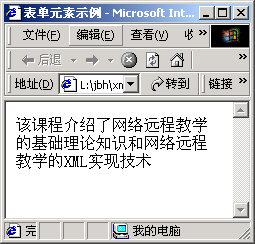
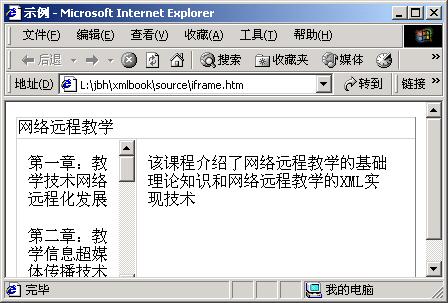
  

图 6.3‑6. HTML\_Contents.htm、HTML\_Main.htm、HTML\_IFrame.htm分别被浏览的结果

1. **CSS：多媒体格式的语言标准**

CSS是本节所述的XML的格式标准，也是本书后述的基于XML的可扩展超媒体标记语言（HTML）、可伸缩二维矢量图形标记语言（SVG）、可扩展三维图形标记语言（X3D）等各种标记语言的格式标准。

1. **CSS代码文件**
2. **CSS定义、引用; 源代码编辑、目标代码编译运行**

还可分别从编辑时、运行时理解XML文档。即时编译运行.

XML文档由前言和元素（集）组成。元素的值体现为XML文档的内容，XML文档内容的呈现方式（如在屏幕上显示的颜色，纸张打印时的排版等）称为该内容的格式（Format）。样式（Style）则可以看作是为内容应用格式的一种机制，即可以将格式以各种方式组合并赋予名称，然后可以按名称调用以获得强大的格式功能。

CSS（Cascading Style Sheets，层级样式单）是Web标准之一（http://www.w3.org/TR/CSS/），是一种最常使用的样式技术，CSS包含大量格式相关的特性[[[1]](#endnote-1)]可用来定义元素的样式。下面是一个CSS样式示例(示例文件XML\_Quiz.css，以.css扩展名标识CSS样式文件）：

|  |
| --- |
| 1. item{ 2. display:block; **/\*指定所有的<item>元素的内容以块状显示\*/** 3. border:2px black double; **/\*指定所有的<item>元素的边界宽度为2像素，黑色，双线， \*/** 4. } 5. problem{ 6. display:block; 7. } 8. content{ 9. display:block; 10. } 11. answers option{ 12. display:list-item; 13. list-style-type: upper-alpha; 14. } 15. difficulty{ 16. display:none; 17. } 18. key{ 19. display:none; 20. } |

一个CSS样式由一系列的样式定义组成。每个样式定义是CSS的基本组成单位。样式定义的基本句法如下：

selector{

Propertyl:valuel;

Property2:value2;

…

}

句法中的selector是一个样式选择器。该选择器的所有样式定义都包含在一对大括号（{}）中，称为样式块。样式块中的每一个样式定义之间以分号（;）分隔，每一个样式以“属性:值”的方式定义，也就是说冒号（:）是赋值符号。XML文档中所有具有与选择器相同名称的元素，即元素与选择器匹配，其内容都将赋予该选择器所对应的样式，即大括号（{}）中所定义的样式。

CSS属性丰富，功能强大，还将在**Error! Reference source not found.**节详述。

小练习：CSS是XML文档样式标准之一，XSL(EXtensible Stylesheet Language)则是XML文档样式定义的另一标准，CSS、XSL具有各自的较适合领域，通过网络探究CSS、XSL的发展历史。

1. **CSS属性的定义与使用**

基于现有的属性，定义想要的属性。

Css中定义。

HTML文档可理解为内容、格式、功能三类主要信息。前面叙述了HTML中定义内容，内容需要进一步赋予格式，HTML文档和XML文档那样，选用CSS样式技术（参见**Error! Reference source not found.**节）格式化文档。CSS是Web标准之一（http://www.w3.org/TR/CSS/）。关键是了解CSS的语法和所有属性的基本功能，然后通过软件辅助（如**Error! Reference source not found.**将叙述的Expression Web）具体实现CSS。

1. **CSS编辑**

CSS代码本质上是文本，所以原则上可以使用任何文本编辑器编辑（如Windows自带的记事本程序），但为了提高编辑效率，而且CSS为HTML赋予格式时，总是和HTML文档紧密联系的，所以一般使用专业的HTML文档编辑软件进行编辑（如Expression Web，将在**Error! Reference source not found.**详述），也就是说编辑HTML文档时，同时编辑CSS。HTML文档引用CSS而获得样式。

1. **HTML文档浏览**

CSS被HTML文档引用并样式化该HTML文档，所以CSS的浏览是因为HTML文档的浏览而随之浏览的。在HTML文档中使用CSS的方式有多种，将在6节详述。在此首先定义一个CSS文件（参见示例文件HTML\_Quiz.css），然后同一路径下的HTML文件（参见示例文件HTML\_Quiz-css.htm）引用该CSS文件来获得样式。引用了HTML\_Quiz.css的HTML\_Quiz-css.htm在浏览器中的运行浏览结果将如图 6.3‑7所示。与**Error! Reference source not found.**节图 6.3‑3比较可发现文档中一些内容格式(如颜色等)的变化。

CSS文件（参见示例文件HTML\_Quiz.css）的代码如下。

|  |
| --- |
| 1. p{ 2. color:rgb(0,255,0); <!--本样式基本原理在一、节XML文档中使用CSS定义格式时已有叙述，其中p是选择器声明，将匹配引用本css文件的HTML文档（在此为同一路径下的HTML\_Quiz.htm）的所有<p>元素的内容。CSS选择器还将在**Error! Reference source not found.**节详述--> 3. } |

HTML文件（参见示例文件HTML\_Quiz-css.htm）的代码如下。

|  |
| --- |
| 1. <?xml version="1.0" encoding="utf-8"?> 2. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.1//EN" 3. "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"> 4. <html xmlns="http://www.w3.org/1999/xhtml"> 5. <head> 6. <title>示例</title> 7. <link rel="styleSheet" href="HTML\_Quiz.css" type="text/css"></link> <!--在XML文档中一般在文档前言通过CSS处理指令来引用CSS文件（参见**Error! Reference source not found.**），而HTML文档为了兼容历史，使用<link>元素来引用CSS文件，href指定了引用同一路径下的CSS文件HTML\_Quiz.css --> 8. </head> 9. <body> 10. <table style="border-width:1px;border-style:solid;width:100%"> 11. <tr> 12. <td style="border-width:1px;border-style:solid;width:100%"> 13. <p>下面可用作元素定界符的字符是：</p> 14. <ol> 15. <li>&lt;</li> 16. <li>@</li> 17. <li>$</li> 18. <li>?</li> 19. </ol> 20. <p>难度系数：0.5</p> 21. <p>1</p> 22. <p>可用于元素定界符的是&lt;和&gt;。</p> 23. </td> 24. </tr> 25. </table> 26. </body> 27. </html> |

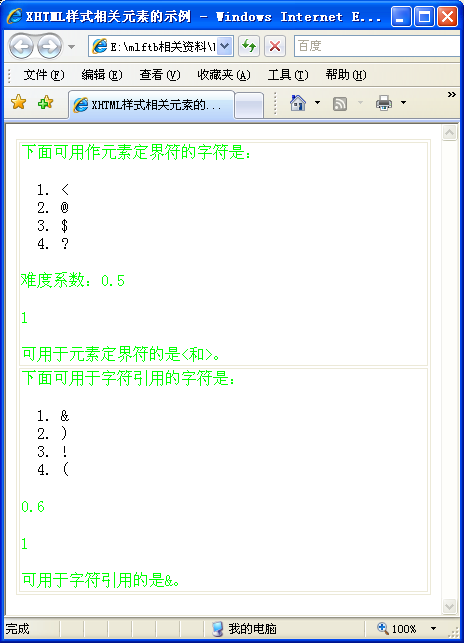


图 6.3‑7 引用了HTML\_Quiz.css的HTML\_Quiz-css.htm的运行结果

在本节的所有CSS示例中，都可将样式定义保存为HTML\_Quiz.css文件，并放置在HTML\_Quiz-css.htm文件同一路径下，浏览该HTML\_Quiz-css.htm文件就可查看应用了HTML\_Quiz.css文件中CSS样式后的结果（**Error! Reference source not found.**节则系统叙述了使用Expression Web软件快捷地编辑CSS，可进行下述所有CSS的实践体验）。

XML文件与CSS文件是两个独立的文件，所以CSS样式必须以一定的方式引用到XML文档中才能起到样式化该XML文档的作用。样式一般在XML文档前言中通过xml-stylesheet处理指令来声明并被XML文档引用。下面是xml-stylesheet处理指令的一个示例：

<?xml-stylesheet href="XML\_Quiz.css" type="text/css"?>

示例声明并引用了一个同一路径下的CSS文件XML\_Quiz.css来获得样式。引用了XML\_Quiz.css样式文件的XML\_Quiz.xml在浏览器中的运行结果如图 6.3‑8所示：

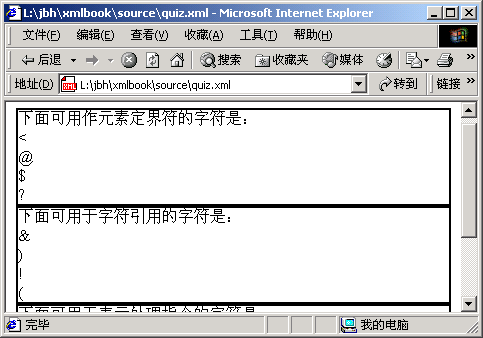


图 6.3‑8. 引用了XML\_Quiz.css样式文件的XML\_Quiz.xml在浏览器中的运行结果

上述示例声明XML\_Quiz.xml文档的样式在同一路径下的XML\_Quiz.css中进行了定义。如果两个文件不是同一路径下，必须使用相对路径或*uri*形式的绝对路径。

1. **CSS代码文件基本结构**

一、节XML文档中使用CSS定义格式时，已叙述CSS基本结构，HTML文档中使用CSS定义格式时的CSS代码基本结构和XML文档中使用CSS定义格式时的CSS代码基本结构是一致的，在此不赘述。

1. **CSS语法**

XML代码必须遵守基本框架保存为.xml格式文件（**Error! Reference source not found.**节），同时，这些代码还必须遵守语法规则，**Error! Reference source not found.**~**Error! Reference source not found.**节将叙述XML语法。

CSS语法可迭代为词法、句法。

* 词即单词，是语言的最小单位，单词之间可以选择空格、换行、回车等多种分隔方式。词法则是关于单词的使用规则。**Error! Reference source not found.**~**Error! Reference source not found.**节将叙述C#词法。
* 句即语句，是一系列单词的组合，并以标记“<></>”分隔。句法则是关于语句的运用规则。**Error! Reference source not found.**~**Error! Reference source not found.**节将叙述XML句法。

1. CSS语法的CSS词法：关键词，非关键词
2. **CSS语法的CSS词法：标识符；名称、ID**
3. **标识符涵义**

标识符是名称、ID的统称。标识符属于非关键词，是用作名称、ID的非关键词。代码中所定义的类型、对象，及其成员（属性、方法等），都是通过标识符进行指代的。例如，**Error! Reference source not found.**中代码“*class MyClass{;}”中*的MyClass是所定义的类的标识符。

标识符用于指代类型、对象等，所以，同一名称空间中不能有重复的标识符，或者说，整个软件中不能存在重复的“名称空间.标识符”，而必须唯一。

思考：关键词能否用作标识符？

1. **标识符、名称、ID规范性**

标识符要尽量体现真实意义，这样便于与他人交流，便于维护。同时，标识符可能需要由多个词复合而成，此时通常遵守如下几种规则中的一种，可结合示例代码自主体会。

* Pascal规则：例如，MyClass这一标识符是遵守Pascal规则的，即，复合词的第一部分My首字母大写，第二部分Class首字母也大写，各部分都首字母大写。名称空间名称、类型名称、类型成员名称（属性名称、方法名称）一般选用Pascal规则。
* Camel规则：例如，myClass这一标识符是遵守Camel规则的，即，复合词的第一部分my首字母小写，第二部分Class首字母大写，随后的各部分都首字母大写。类型实例化出的对象名称一般选用Camel规则。

小练习：VS对象资源管理器中，查看.Net内置类型的代码，思考标识符规范性。

1. **CSS语法的CSS词法：注释**
2. **CSS语法的CSS句法：样式（属性与方法的统一体）**

定义样式

1. **CSS语法的CSS句法：语句、语句块**

HTML是面向元素的语言。C#是面向对象的语言。元素与对象互为映射（元素是特性与行为的统一体；对象是属性与方法的统一体）。整个CSS文件的核心是属性集合。样式名称规则。

CSS定义需要与HTML文档中的元素匹配，以格式化所匹配的元素的内容，所以必须理解CSS与元素匹配的机制，那就是选择器机制。其实前面已述及多种选择器机制，在此系统化概述。

1. **嵌入式CSS与style (样式)特性选择器**

对于嵌入式CSS，是作为一个元素的style(样式)特性值嵌入在该元素中了，所以也就是只能样式化该元素，而不需要过多考虑选择器问题，**Error! Reference source not found.**节已述及嵌入式CSS，在此不赘述。但如果从选择器角度来考虑嵌入式CSS与元素的匹配问题，这样也可将嵌入式CSS选择器、内部CSS选择器、外部CSS选择器保持逻辑一致的思路，那也就可认为嵌入式CSS选择器是style (样式)特性选择器。

1. **element(元素)选择器**

一、节HTML文档中使用CSS定义格式时，CSS选择器就是元素选择器，即HTML文档中所有具有与选择器相同名称的元素，即元素与选择器匹配，其内容都将赋予该选择器所对应的样式。在此将XML文档中的元素迁移到HTML文档中的元素即可，如**Error! Reference source not found.**节所述的样式引用也属于元素选择器。

1. **class(类)特性选择器**

class(类)特性选择器又常简称为class(类)选择器。即任何HTML元素都预定义了一个class(类)特性，class(类)特性值与选择器匹配时就将赋予CSS样式，class(类)选择器又可进一步分为类型类、通用类和假类选择器，**Error! Reference source not found.**节曾述及，在此不赘述。

1. **关于CSS类**

在CSS中，通过选择器来与HTML文档中的元素匹配，匹配的即赋予相应的样式。如p{color: rgb(0,255,0)}将HTML文档中所有的与选择器p匹配的所有<p>赋予相应的样式。

HTML文档样式的重要性而需要更多灵活的CSS机制，所以任何HTML元素都预定义了一个类（class）特性，这个特性专门与CSS类选择器机制相结合，提供更灵活强大的CSS功能。

CSS类选择器有多种，常见的是类型类、通用类和假类。

1. **类型类**

下面是类型类定义的一个示例（参见示例文件HTML\_ClassType.htm），代码注释中含有类型类的解释。

|  |
| --- |
| 1. <?xml version="1.0" encoding="utf-8"?> 2. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.1//EN" 3. "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"> 4. <html xmlns="http://www.w3.org/1999/xhtml"> 5. <head> 6. <title>示例</title> 7. <style type="text/css"> 8. p.size{ <!--类型类方式定义的CSS样式。这时HTML文档应用样式时，不仅要求选择器匹配，还要求HTML文档中元素的class特性与选择器的类匹配--> 9. font-size:30; 10. } 11. </style> 12. </head> 13. <body> 14. <table style="border-width:1px;border-style:solid;width:100%"> 15. <tr> 16. <td style="border-width:1px;border-style:solid;width:100%"> 17. <p class="size">下面可用作元素定界符的字符是：</p> <!--只有选择器和类都匹配的元素才应用类型类方式定义的CSS样式--> 18. <ol> 19. <li>&lt;</li> 20. <li>@</li> 21. <li>$</li> 22. <li class="size">?</li> <!--只有选择器和类都匹配的元素才应用类型类方式定义的CSS样式--> 23. </ol> 24. <p>难度系数：0.5</p> 25. <p>1</p> 26. <p>可用于元素定界符的是&lt;和&gt;。</p> 27. </td> 28. </tr> 29. </table> 30. </body> 31. </html> |

1. **通用类**

下面是通用类定义的一个示例（参见示例文件HTML\_ClassGeneral.htm），代码注释中含有通用类的解释。

|  |
| --- |
| 1. <?xml version="1.0" encoding="utf-8"?> 2. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.1//EN" 3. "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"> 4. <html xmlns="http://www.w3.org/1999/xhtml"> 5. <head> 6. <title>示例</title> 7. <style type="text/css"> 8. \*.size{ 9. font-size:30; 10. } <!--该语句块以通用类方式定义了CSS样式。\*表示选择器是通配的，\*也可以省略，即\*.size变换为.size时功能是一样的，这时HTML文档应用样式时，只要求HTML文档中元素的class特性与类匹配--> 11. </style> 12. </head> 13. <body> 14. <table style="border-width:1px;border-style:solid;width:100%"> 15. <tr> 16. <td style="border-width:1px;border-style:solid;width:100%"> 17. <p class="size">下面可用作元素定界符的字符是：</p> <!--只要类匹配的元素就可应用通用类方式定义的CSS样式--> 18. <ol> 19. <li>&lt;</li> 20. <li>@</li> 21. <li>$</li> 22. <li class="size">?</li> <!--只要类匹配的元素就可应用通用类方式定义的CSS样式--> 23. </ol> 24. <p>难度系数：0.5</p> 25. <p>1</p> 26. <p>可用于元素定界符的是&lt;和&gt;。</p> 27. </td> 28. </tr> 29. </table> 30. </body> 31. </html> |

1. **假类**

CSS样式通常根据HTML文档中的元素的名称、class特性来与样式定义匹配并赋予样式。而假类是一种特殊的类，是根据元素的特征（如交互性、动态性）来与样式定义匹配并赋予样式。所以假类总是针对HTML文档中一些特殊元素的。常用的假类有超链接假类，超链接假类只适用于<a>元素，列举如下：

* 1. a:link 假类：应用于还未被访问的超链接
  2. a:visited假类：应用于已被访问的超链接
  3. a:hover假类：应用于鼠标移动到的超链接

注意假类的定义中选择器与类名称之间是以:而不是以.进行分隔的。

下面是假类的一个示例：

a: hover { text-decoration:none; /\*IE中超链接默认有下划线格式，该CSS指定当鼠标移动到超链接上时该超链接无下划线格式\*/}

1. **id特性选择器**

id(标识)特性选择器又常简称为id选择器。即任何HTML元素都预定义了一个id特性，id特性值与选择器匹配时就将赋予CSS样式，因为id特性用来唯一标识一个元素，所以一个HTML中所有元素的id号不允许重复，因此，id特性选择器意味着只能匹配整个HTML文档中的一个元素。下面是一个示例(参见示例文件“ID选择器.htm”)。

|  |
| --- |
| 1. <html> 2. <head> 3. <title>ID选择器</title> 4. <style type="text/css"> 5. <!-- 6. #one{ <!--id特性选择器以#声明--> 7. font-weight:bold; 8. } 9. #two{ 10. font-size:30px; 11. color:#009900; 12. } 13. --> 14. </style> 15. </head> 16. <body> 17. <p id="one">id选择器1</p><!--将显示#one定义的样式--> 18. <p id="two">id选择器2</p><!--将显示#two定义的样式--> 19. <p id="three">id选择器3</p><!--无匹配的样式--> 20. <p id="one two">id选择器3</p><!--无匹配的样式--> 21. </body> 22. </html> |

1. **CSS继承机制、层级机制；在线样式、文档样式、外部样式**

和面向对象的程序机制那样，CSS支持继承，CSS继承是指后代元素自动“遗传”祖先元素样式的机制，是简化样式生成的捷径。当为一个元素指定了样式时，无需特别声明即可将其样式属性赋予其后代元素。

如果有如下一个CSS定义：

body{font-size:50;}

示例中只定义了<body>的样式，但因为包含在<body>中的所有元素都是<body>的后代，如果这些元素没有专门指定CSS属性，这些元素将继承<body>的CSS属性，即所有包含在<body>中的元素的文本内容都以50像素的大小显示。

注：不是任何CSS属性都能被继承，例如CSS计算值（如百分比值等）一般就不能被继承，CSS属性是否可以被继承不用死记，而应该理解和试验。

前面所有有关HTML文档样式的叙述中，CSS样式定义都是保存为独立的CSS文件，然后HTML通过引用该外部CSS文件来获取样式。其实HTML文档除了通过引用外部CSS(External CSS)文件来获取样式外，还支持内部CSS(Internal CSS)、嵌入式CSS(Inline CSS)，并支持它们的层级机制。下面是一个示例，参见示例文件HTML\_Casscade.htm。

|  |
| --- |
| 1. <?xml version="1.0" encoding="utf-8"?> 2. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.1//EN" 3. "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"> 4. <html xmlns="http://www.w3.org/1999/xhtml"> 5. <head> 6. <title>CSS层级示例</title> 7. <link rel="styleSheet" href="HTML\_Cascade.css" type="text/css"></link> <!--外部CSS定义<body>中所有文本的字体大小为10像素。--> 8. <style type="text/css"> <!--这里是内部CSS定义。HTML有一个预定义的<style>，内部CSS在HTML文档内部通过<style>定义，语法和在外部CSS中定义CSS时完全一样,<style>的位置必须保证在欲使用该样式的元素之前。实际上一般的做法是把所有定义样式的<style>置于<head>中，<head>出现在<body>之前，这样<body>中所有可能需要使用内部CSS的元素都可方便地使用。--> 9. li{ 10. font-size:30; } <!--内部CSS定义所有<li>的文本内容的字体大小为30像素，这与外部CSS的<body>的定义相冲突，由于内部CSS权重大于外部CSS，所以<li>中的文本内容的字体大小为30像素。--> 11. </style> 12. </head> 13. <body> 14. <table style="border-width:1px;border-style:solid;width:100%"> 15. <tr> 16. <td style="border-width:1px;border-style:solid;width:100%"> <!--每个HTML元素都内置有预定义的style特性，嵌入式CSS直接在HTML元素中通过style特性定义，style特性表示该处是CSS属性赋值，style特性值是分号分隔的CSS属性的“名：值”对列表。--> 17. <p>下面可用作元素定界符的字符是：</p> 18. <ol> 19. <li>&lt;</li> 20. <li style="font-size:50">@</li> <!--嵌入式CSS定义该<li>的文本内容的字体大小为50像素，这与外部CSS的<body>的定义和内部CSS<li>的定义相冲突，由于嵌入式CSS的权重最大，所以这个<li>中的文本内容的字体大小为50像素。--> 21. <li>$</li> 22. <li>?</li> 23. </ol> 24. <p>难度系数：0.5</p> <!--<p>等没有样式冲突的元素的文本内容继承<body>的样式，字体大小为10像素。--> 25. <p>1</p> 26. <p>可用于元素定界符的是&lt;和&gt;。</p> 27. </td> 28. </tr> 29. </table> 30. </body> 31. </html> |

一个HTML元素可能会应用到多重的CSS定义，这时通过CSS层级机制来解决样式冲突。CSS 层级机制是当多个CSS样式定义应用到同一个HTML元素时，分配一个权重给每一个样式定义，最高权重的样式定义优先。下面是层级的规则。

* + - 外部CSS、内部CSS、嵌入式CSS三种方式的样式定义的权重依次增大。
    - 同一方式的样式定义发生冲突时，后定义样式的权重大于先定义的样式。

上述示例中，如果HTML\_Cascade.css样式文件的内容如下：

body{ font-size:10;/\*定义<body>中所有文本的字体大小为10像素。\*/}

HTML\_Casscade.htm的运行结果如图 6.3‑9所示。

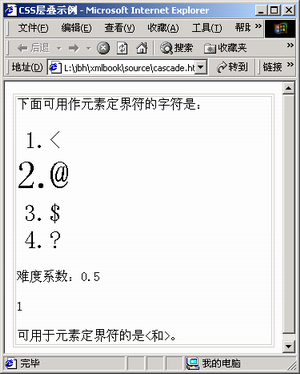


图 6.3‑9. HTML\_Casscade.htm的运行结果

思考：外部CSS、内部CSS、嵌入式可以达到同样的样式结果，试从易用性、易维护性等角度思考在什么情形下使用外部CSS，什么情形下使用内部CSS，什么情形下使用嵌入式CSS。

1. **CSS属性概述**
2. **CSS属性概述**

CSS是一个Web标准，具有标准的预定义属性，掌握CSS的关键就是掌握CSS标准中预定义的属性的用法。CSS预定义的属性丰富，功能强大，在此主要掌握其基本原理，然后可基于基本原理，参考CSS标准（http://www.w3.org/TR/CSS/），自主上机实验，全面掌握CSS。

为了更好地理解CSS属性，在此将CSS所有常用属性归类概述如下，以作大概了解。实际应用时，应该根据正编辑的HTML文档的需求而考虑CSS属性选用，如果不能确定属性的具体用法，则应该参考CSS标准。如果以软件为辅助工具，则更可大大地提高效率，使用Expression Web软件编辑CSS将在**Error! Reference source not found.**详述。

* 1. 颜色（Color）

color

* 1. 背景（Background）

Background，background-attachment，background-color，background-image，background-position，background-positionX，background-positionY，background-repeat，layer-background-color，layer-background-image

* 1. 字体（Font）

Font，font-family，font-size，font-size-adjust，font-stretch，font-style，font-weight，text-decoration，text-underline-position，text-shadowfont-variant，text-transform，line-height，letter-spacing，word-spacing

* 1. 文本（Text）

text-indent，text-overflow，vertical-align，text-align，layout-flow，writing-mode，direction，unicode-bidi，word-break，line-break，white-space，word-wrap，text-autospace，text-kashida-space，text-justify，ruby-align，ruby-position，ruby-overhang，ime-mode，layout-grid，layout-grid-char，layout-grid-char-spacing，layout-grid-line，layout-grid-mode，layout-grid-type

* 1. 用户界面（Interface）

滚动条（Scrollbars）类：scrollbar-3d-light-color，scrollbar-highlight-color，scrollbar-face-color，scrollbar-arrow-color，scrollbar-shadow-color，scrollbar-dark-shadow-color，scrollbar-base-color

其它属性（Classification）类：Behavior，zoom，cursor，filter

* 1. 布局（Layout）

尺寸（Dimensions）类：Height，max-height，min-height，width，max-width，min-width

占位（Occupation）类：clear，float，clip，overflow，overflow-x，overflow-y，display，visibility

定位（Positioning）类：Position，z-index，top，right，bottom，left

边白（Margins）类：Margin，margin-top，margin-right，margin-bottom，margin-left

轮廓（Outlines）类：Outline，outline-color，outline-style，outline-width

边框（Borders）类：Border，border-color，border-style，border-width，border-top，border-top-color，border-top-style，border-top-width，border-right，border-right-color，border-right-style，border-right-width，border-bottom，border-bottom-color，border-bottom-style，border-bottom-width，border-left，border-left-color，border-left-style，border-left-width

衬距（Paddings）类：Padding，padding-top，padding-right，padding-bottom，padding-left

列表项目（Lists）类：list-style，list-style-image，list-style-position，list-style-type，marker-offset

表格（Table）类：border-collapse，border-spacing，caption-side，empty-cells，table-layout，speak-header

下面将根据上述分类，分别在十、节、十一、节、十二、节、十三、节、十四、节、九、节系统叙述CSS。

1. **CSS属性概述：CSS布局**

曾述及<table>与<tr><td>、<div>、<li>与<ol><ul>等元素/标记相关的布局，由于兼容HTML历史的原因，HTML布局以元素/标记和为元素/标记赋予CSS样式共同实现，在此主要叙述CSS相关的布局。在元素/标记布局的基础上，为元素/标记赋予CSS布局属性可进一步用来控制元素的占位大小，绝对定位，相对定位，溢出处理，层，等等。

1. **元素尺寸、填塞、边框、边白**

元素尺寸(size)、填塞(padding)、边框(border)、边白(margin)，元素内容占位尺寸与元素占位尺寸的关系参见**Error! Reference source not found.**节**Error! Reference source not found.**。

* 1. 元素尺寸

width和height属性为一个元素指定元素内容占位的尺寸。

width

width属性指定元素内容的宽度，可赋一个长度数值、百分比、**auto** 值。下面是一个示例：

td{ width:100;/\*指定所有<td>内容的宽度为100像素\*/ }

height

height属性指定元素内容的高度，可赋一个长度数值，百分比，**auto** 值。下面是一个示例：

td{ height:100;/\*指定所有<td>内容的高度为100像素\*/ }

注：HTML文档是面向屏幕动态互动式显示的，不允许指定尺寸(因为<body>代表文档内容的开始，所以也就是说<body>元素的CSS的width和height属性值设置默认不起作用)，其尺寸随网站访问者的屏幕分辨率的改变而改变，或者说也就是随网站访问者浏览器(如IE)窗口尺寸的变化而变化，为文档设置整洁的布局将在**Error! Reference source not found.**节结合上机实验详述。

* 1. 填塞

padding属性指定一个布局了元素的填塞大小，只可赋一个数字值。下面是一个示例：

td{ padding: 10;/\*指定所有<td>的四周的填塞为10像素\*/ }

padding属性是 padding -top, padding-right, padding-bottom和padding-left属性集合的综合表示方法，当集合中的每一属性都赋相同值时，就可这样使用。上例可复杂化为td{padding-top:10; padding-right: 10; padding-bottom:10; padding-left: 10}，这时可以为填塞的上、下、左、右每一个属性赋不同的值。这个使用规律适用于类似的CSS属性。

* 1. 边框

border属性为一个元素指定边框的宽度、颜色和样式。

边框宽度

border-width 属性用来为一个元素设置边框的宽度。下面是设置边框宽度的一个示例：

p{ border-width: 2 } /\*指定所有<p>的边框的宽度为2 像素\*/

1. 边框样式

border-style属性用来为形成边框的线条指定样式(如固体式，双线条式，破折号式等)，下面是一个示例：

p{border-width: 2 ;border-style:dash; /\* 指定所有<p>的边框的边框线条为破折号样式\*/}

1. 边框颜色

border-color属性用来为形成边框的线条指定颜色，下面是一个示例：

p{border-width: 2;border-color:RGB(0,255,0); /\* 指定所有<p>的边框的线条的颜色为绿色\*/}

* 1. 边白

margin 属性用来为一个布局了的元素指定边白。只可赋一个数字值。下面是一个示例：

body{margin:10 /\*指定<body>即整个HTML文档内容的边白为10像素 \*/}

1. **显示**

display和visibility属性用来指定元素的显示方式。

display

display 属性为元素指定显示方式（如block， inline ，list-item 、none 、table）下面是一个示例：

p{display:none;/\* 指定所有<p>不显示（即不再占据任何布局空间），相当于其内容折叠起来了\*/}

visibility

visibility属性为元素指定可见性（如visible、 hidden） 下面是一个示例：

p{visibility:hidden;/\* 指定所有的<p>不可见（但仍然占据了它原本应该占据的布局空间），相当于其内容透明了\*/}

思考：display:none与visibility:hidden的区别。

1. **层**

屏幕是二维的，HTML文档也是二维的，HTML使用层，即Z-Index（Z-索引）满足简单的三维Z轴定位的需求。下面是一个示例。层还将在**Error! Reference source not found.**节上机操作部分专门实践。

div{z-index:2 /\*指定<div>元素内容为第2层，没有显式指定z-index值的默认为0层，所有层根据z-index值的大小排序，值更大的在更上层\*/}

1. **流式定位、绝对定位、相对定位**

HTML面向二维文本，所以默认是二维流式定位机制，也就是说，HTML文档布局的初始坐标原点位于浏览器窗口的左上顶角，X轴正向朝右，Y轴正向朝下，如图 6.3‑10所示。



图 6.3‑10. HTML初始坐标系

注：原理上可以对图 6.3‑10所示的初始坐标系进行变换而获得新的布局坐标系。但HTML文档编辑一般不进行坐标系变换，坐标系变换将在**Error! Reference source not found.**节的XAML坐标变换和0节的X3D坐标变换详述。

一个HTML文档由多个元素组成，所有元素在视区中默认是以流的方式进行定位而布局的。流式定位意味着在视区中显示HTML文档时将HTML文档中的所有元素依照它们在文档中的次序一个紧接着一个罗列排布，这样任何元素之间不会发生互相重叠的现象，当整个文档内容多于视区可容纳的大小时，将使用滚动条机制。当影响布局的因素改变（如字体、填充、边界、边白等布局属性改变）或视区自身大小改变时（如用户可以改变浏览器窗口的大小而因此改变了HTML文档视区的大小），流式定位的元素的内容将重新流布。

在整个HTML文档的布局中，根元素<html>是HTML文档在浏览器窗口中运行时的第一个容器，HTML设计为<body>的布局与<html>是重叠的，所以<body>也就成为了实际意义上的最顶层的定位容器。<body>的左上角坐标是（0，0），占位空间则由其CSS布局属性决定。随着<body>中嵌套的其他元素的显示，形成了整个HTML文档的布局。

HTML面向二维文本，所以默认是二维流式定位机制，当然，HTML也提供了将默认的二维流式定位机制转变为绝对定位、相对定位的机制。下面是一个示例。

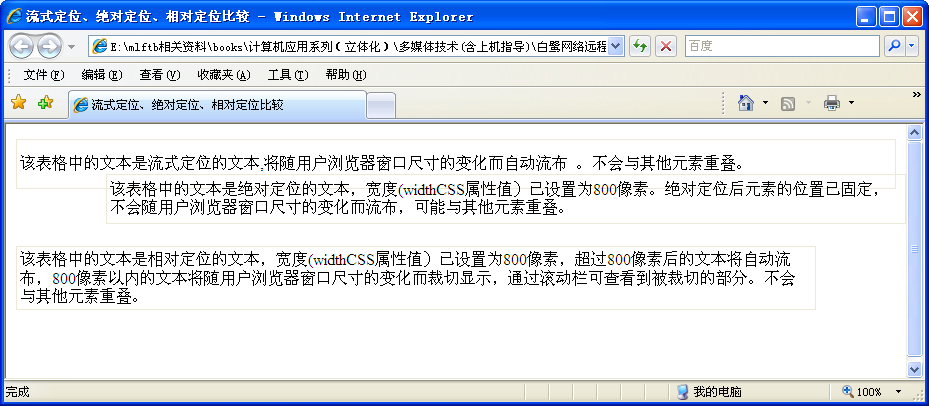
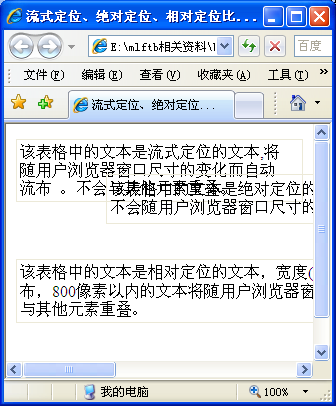
div{position: /\*position赋值空，即默认值，指定<div>元素为默认定位(流式定位)\*/}

div{position:relative /\* position赋值relative，指定<div>元素为relative(相对定位)\*/}

div{position:absolute /\* position赋值absolute，指定<div>元素为absolute(绝对定位)\*/}

流式定位、绝对定位、相对定位各有其适用场合，互为补充，以满足HTML文档面向文本的互动式动态定位的需求，但也正因为互动式动态性，所以也是难点之一，注意实践中思辩。

小练习：IE中打开示例文件position.html，拖动IE边框改变IE窗口的尺寸，观察内容的动态定位，思辩流式定位、绝对定位、相对定位的异同以及各自的适用情形。体会：(1).表格1中的文本是流式定位的文本,将随用户浏览器窗口尺寸的变化而自动流布，不会与其他元素重叠；(2).表格2中的文本是绝对定位的文本，绝对定位后元素的位置已固定，不会随用户浏览器窗口尺寸的变化而流布，可能与其他元素重叠；(3).表格3中的文本是相对定位的文本，将随用户浏览器窗口尺寸的变化而裁切显示，通过滚动栏可查看到被裁切的部分，不会与其他元素重叠。

(a)1024\*768像素的IE窗口 (b)缩小为400\*768像素的IE窗口

图 6.3‑11. 流式定位、绝对定位、相对定位的异同

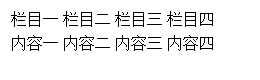
1. **CSS布局思考**

HTML布局以元素/标记和为元素/标记赋予CSS样式交叉实现，其中<table>+CSS与<div>+CSS是两种最流行的机制，试通过网络探究<table>+CSS与<div>+CSS布局机制各自的优缺点。并结合**Error! Reference source not found.**节上机操作部分实践HTML文档布局。**Error! Reference source not found.**节则将叙述XAML布局，面向图形的XAML布局全部单一地通过元素/标记实现，更容易理解，可将元素/标记与CSS交叉实现的HTML布局与其比较理解。

关于网页布局，大体上可以分为两种，一种是使用table布局，另一种使用div布局，很多人说现在主流的网页布局是div+css。那么div和table布局究竟有那些联系和区别呢？我们又该如何进行网页布局呢？

1. div和table布局各自的优缺点：
2. div+css进行布局，可以实现table的页面布局效果，还能减少页面中元素的个数，使得页面夹在较快，而且能更好的被搜索引擎相匹配。
3. table进行布局可以有一个统一的外观，便于控制格式，对于初学者来说是很好的选择。还有一个最大的好处就是兼容性好。
4. div的缺点是兼容性没有table好，而且对于新手来说，div没有table好控制。Table比div更便于可视化编辑网页。
5. table的缺点就是太多的table嵌套会使网页的内容量减少，而且看起来非常臃肿，网页的加载速度变慢。在网页显示的时候，table必须把结束</table>加载完以后才能显示整合网页，而div则是一个小的显示块，加载完以后可以逐个显示。这样div在一定程度上比table显示的要快，性能上更好。
6. 现在有很多网站都有主题更换功能， QQ空间，人人网，百度空间、新浪博客等都是将一系列主题CSS文件放置在一起，供你选择替换，如果是表格的布局就不可能有这种功能？
7. 个人分析感觉，在一些项目中，企业内部使用的管理信息系统，很多多使用的table进行布局，因为企业内部系统一般没有很多的特效，不需要精细的处理，也不是为了吸引人浏览而做的。相对来说企业网络较好，所以使用table布局快速，简洁，方便。而工商银行使用table应该更多的是想到了兼容性。看了一些大公司的网页布局，其中腾讯，网易，中国移动等公司都使用的是div进行的布局。而工商银行则使用table进行布局。
8. div布局和table布局的互相转换

上面的优缺点难以绝对，事实上div布局和table布局可互相转换，那么有什么需求，尽管花时间去检验尝试就可以了。下面是分别使用div布局和table布局实现如下一个布局效果。所以建议在VS中通过table布局快速可视化地编辑布局，需要转化为div布局时就再转化。



1. table布局实现
2. **<html>**
3. **<head>**
4. **</head>**
5. **<body>**
6. **<table>**
7. **<tr>**
8. **<td>**栏目一**</td>**
9. **<td>**栏目二**</td>**
10. **<td>**栏目三**</td>**
11. **<td>**栏目四**</td>**
12. **<tr>**
13. **<tr>**
14. **<td>**内容一**</td>**
15. **<td>**内容二**</td>**
16. **<td>**内容三**</td>**
17. **<td>**内容四**</td>**
18. **<tr>**
19. **</table>**
20. **</table>**
21. **</body>**
22. **</html>**
23. Div布局实现
24. **<html>**
25. **<head>**
26. **<style** type="text/css"**>**
27. ul li{
28. list-style-type:none;
29. display:inline;
30. }
31. **</style>**
32. **</head>**
33. **<body>**
34. **<div>**
35. **<ul>**
36. **<li>**栏目一**</li>**
37. **<li>**栏目二**</li>**
38. **<li>**栏目三**</li>**
39. **<li>**栏目四**</li>**
40. **<ul>**
41. **</div>**
42. **<div>**
43. **<ul>**
44. **<li>**栏目一**</li>**
45. **<li>**栏目二**</li>**
46. **<li>**栏目三**</li>**
47. **<li>**栏目四**</li>**
48. **<ul>**
49. **</div>**
50. **</table>**
51. **</body>**
52. **</html>**

这里几点比较注意的是，ul 和li 可以实现表格的布局，这里每个ul相当于table里面的tr实现一行，每个li相当于table的td，实现一个单元格。

其中需要用到css控制它的显示样式，其中list-style-type:none; 这句话消除li前面的项目符号，display:inline; 这句话让li标签横向显示。这样就能实现表格的效果。

1. **CSS属性概述：CSS颜色**

CSS颜色属性用来为元素的文本内容或背景等指定颜色。下面的示例为元素的文本内容指定红色：

li{ color: rgb(255,0,0) } /\*指定所有<li>的文本内容为红色，rgb(255,0,0)使用的是RGB颜色表示方式来表示红色，该方法可满足任何颜色的表示 \*/

颜色机制参见**Error! Reference source not found.**节。为了方便使用和兼顾兼容性，CSS颜色可以选用如下表示方法：

1. 可以用一个颜色关键词。常用的颜色关键词是: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white和 yellow。如：body{style="color: green" }
2. 一个rgb()函数来表示。rgb()函数表示颜色时的参数由逗号分隔的三个十进制数（或百分数）组成,每个十进制数值介于[0,255]之间。如：body{style="color: rgb(0,255,0) " }
3. #符号后紧跟6位十六进制的数值来表示。如： body{style= "#00FF00 " }
4. **CSS属性概述：CSS背景**

CSS背景属性可以为一个具有布局属性的元素指定背景颜色或以一个背景图像来获得所需要的背景效果。

下面的示例为元素指定红色背景色：

td{ background-color: rgb(0,255,0)} /\*指定所有<td>的背景色为绿色\*/

下面的示例为元素指定一个背景图像:

td{ background-image: url("background.gif") } /\* 指定所有<td>的背景图像, 背景图像通过url("background.gif")函数来获取,其中参数background.gif是一个图像文件的url\*/

当使用背景图像时,背景图像的原始尺寸可能与元素的尺寸不相等,这时可附加background-repeat、repeat、 repeat-x、 repeat-y 、no-repeat等属性来控制背景的重复属性。

注：具有布局属性的元素才能指定背景，具有布局可简单理解为该元素具有确定的长和宽。

思考：为元素使用背景时，为了美观，所引用的原始图像应该考虑哪些因素？

1. **CSS属性概述：CSS字体**

CSS字体类属性可为文本指定字体特征。

1. **font-family**

font-family为文本指定字系，下面是一个示例：

p{

font-family:华文彩云; /\* 为所有<p>的文本指定“华文彩云”字系\*/

}

注：各种语言的字系（如中文的“华文彩云”等）需要浏览器的特定支持。

1. **font-style**

font-style为文本指定字样（如normal 、italic和oblique等），下面是一个示例：

p{

font-family:华文彩云;

font-style:italic; /\* 为所有<p>的文本指定斜体字样\*/

}

1. **font-size**

font-size属性为文本指定字体大小，下面是一个示例：

p{

font-size:14; /\* 为所有<p>的文本指定字体大小为14像素\*/

}

字体大小的默认单位是px，可以使用其他绝对单位或相对单位。

1. **CSS属性概述：CSS文本**

CSS文本属性用来指定文本排布的一些特性，可用来控制文本的字母间隔、词间隔、文本对齐等。

1. **text-align**

text-align属性用来指定一个块中的在线内容怎样横向对齐，（如left, right, center和 justify等 ），下面是一个示例：

td{

text-align:center;/\*指定所有<td>中的内容中间对齐\*/

}

1. **vertical-align**

vertical-align属性用来为一个具有布局属性的元素的内容指定纵向对齐方式（如：baseline 、middle、 subsuper 、top 、bottom ）。下面是一个示例：

p{

vertical-align:sub /\*指定所有<p>的内容的纵向对齐为上标方式\*/

}

1. **text-decoration**

text-decoration属性用来为一个元素的文本增加装饰效果（如none，underline 、overline 、line-through 、blink ），下面是一个示例：

li{

text-decoration:underline /\* 指定所有<li>的内容装饰下划线。\*/

}

1. **letter-spacing**

letter-spacing属性用来指定一个单词中字母之间的间隔（如normal ，[<length>](file:///E:\\books\\web\\ref\\CSSRefer\\syndata.html" \l "value-def-length)），下面是一个示例：

td{

letter-spacing:10 /\* 指定所有<td>中每个单词中的字母之间的间隔为10像素。\*/

}

1. **word-spacing**

word-spacing属性用来指定单词与单词之间的间隔（如normal，[<length>](file:///E:\books\web\ref\CSSRefer\syndata.html#value-def-length)），下面是一个示例：

td{

word-spacing:20 /\*指定所有<td>中每个单词之间的间隔为20像素。\*/

}

1. **CSS属性概述：CSS用户界面**

CSS用户界面属性用来指定用户界面相关的一些特性，其中经常使用到的是cursor属性。cursor属性为鼠标指定一个光标类型（如auto ，crosshair ，default.，pointer ，move， text， wait， help 等）下面是一个示例：

p{

cursor :move /\*指定当鼠标在所有<p>位置上时显示移动方式类型的光标\*/

}

1. []注：Attribute，本书将Attribute翻译为特性Property，而将Property翻译为属性，以示区别。一般XML等标记语言中，元素(Element)与特性(Attribute)结合使用；编程语言中，对象(Object)与属性(Property)结合使用。 [↑](#endnote-ref-1)