# 第1章 HTML 简介

## 1.1 什么是HTML？

HTML 是用来描述网页的一种语言。

1. HTML 指的是超文本标记语言 (Hyper Text Markup Language)
2. HTML 不是一种编程语言，而是一种标记语言 (markup language)
3. 标记语言是一套标记标签 (markup tag)
4. HTML 使用标记标签来描述网页

## 1.2 HTML 标签

HTML 标记标签通常被称为 HTML 标签 (HTML tag)。

1. HTML 标签是由尖括号包围的关键词，比如 <html>
2. HTML 标签通常是成对出现的，比如 <b> 和 </b>
3. 标签对中的第一个标签是开始标签，第二个标签是结束标签
4. 开始和结束标签也被称为开放标签和闭合标签

## 1.3 HTML 文档 = 网页

HTML 文档描述网页

HTML 文档包含 HTML 标签和纯文本

HTML 文档也被称为网页

Web 浏览器的作用是读取 HTML 文档，并以网页的形式显示出它们。浏览器不会显示 HTML 标签，而是使用标签来解释页面的内容：

<html>

<body>

<h1>我的第一个标题</h1>

<p>我的第一个段落。</p>

</body>

</html>

例子解释

<html> 与 </html> 之间的文本描述网页

<body> 与 </body> 之间的文本是可见的页面内容

<h1> 与 </h1> 之间的文本被显示为标题

<p> 与 </p> 之间的文本被显示为段落

# 第2章 HTML 编辑器

## 2.1 使用 Notepad 或 TextEdit 来编写 HTML

可以使用专业的 HTML 编辑器来编辑 HTML：

Adobe Dreamweaver

Microsoft Expression Web

CoffeeCup HTML Editor

不过，我们同时推荐使用文本编辑器来学习 HTML，比如 Notepad (PC) 或 TextEdit (Mac)。我们相信，使用一款简单的文本编辑器是学习 HTML 的好方法。

通过记事本，依照以下四步来创建您的第一张网页。

步骤一：启动记事本

如何启动记事本：

开始

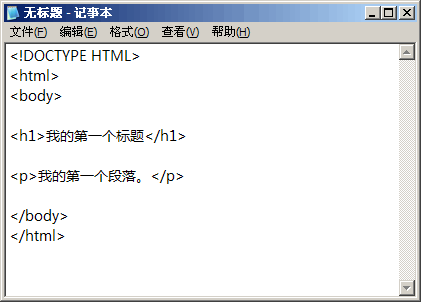
所有程序

附件

记事本

步骤二：用记事本来编辑 HTML

在记事本中键入 HTML 代码：



记事本

步骤三：保存 HTML

在记事本的文件菜单选择“另存为”。

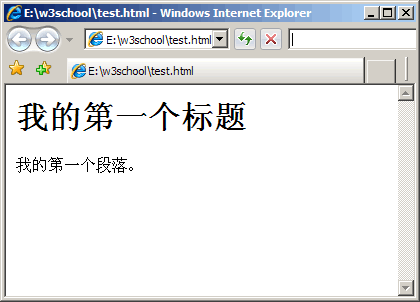
当您保存 HTML 文件时，既可以使用 .htm 也可以使用 .html 扩展名。两者没有区别，完全根据您的喜好。

在一个容易记忆的文件夹中保存这个文件，比如 w3school。

步骤四：在浏览器中运行这个 HTML 文件

启动您的浏览器，然后选择“文件”菜单的“打开文件”命令，或者直接在文件夹中双击您的 HTML 文件。

结果应该类似这样：



在浏览器中查看

# 第3章 基本的 HTML 标签 - 四个实例

## 3.1 HTML 标题

HTML 标题（Heading）是通过 <h1> - <h6> 等标签进行定义的。

<html>

<body>

<h1>This is heading 1</h1>

<h2>This is heading 2</h2>

<h3>This is heading 3</h3>

<h4>This is heading 4</h4>

<h5>This is heading 5</h5>

<h6>This is heading 6</h6>

<p>请仅仅把标题标签用于标题文本。不要仅仅为了产生粗体文本而使用它们。请使用其它标签或 CSS 代替。</p>

</body>

</html>

## 3.2 HTML 段落

HTML 段落是通过 <p> 标签进行定义的。

<html>

<body>

<p>这是段落。</p>

<p>这是段落。</p>

<p>这是段落。</p>

<p>段落元素由 p 标签定义。</p>

</body>

</html>

## 3.3 HTML 链接

HTML 链接是通过 <a> 标签进行定义的。

<html>

<body>

<a href="http://www.w3school.com.cn">

This is a link</a>

</body>

</html>

注释：在 href 属性中指定链接的地址。

## 3.4 HTML 图像

HTML 图像是通过 <img> 标签进行定义的。

<html>

<body>

<img src="/i/eg\_w3school.gif" width="300" height="120" />

</body>

</html>

注释：图像的名称和尺寸是以属性的形式提供的。

# 第4章 HTML 元素

HTML 文档是由 HTML 元素定义的。

## 4.1 HTML 元素

HTML 元素指的是从开始标签（start tag）到结束标签（end tag）的所有代码。

|  |  |  |
| --- | --- | --- |
| 开始标签 | 元素内容 | 结束标签 |
| <p> | This is a paragraph | </p> |
| <a href="default.htm" > | This is a link | </a> |
| <br /> |  |  |

注释：开始标签常被称为开放标签（opening tag），结束标签常称为闭合标签（closing tag）。

## 4.2 HTML 元素语法

HTML 元素以开始标签起始

HTML 元素以结束标签终止

元素的内容是开始标签与结束标签之间的内容

某些 HTML 元素具有空内容（empty content）

空元素在开始标签中进行关闭（以开始标签的结束而结束）

大多数 HTML 元素可拥有属性

提示：您将在本教程的下一章中学习更多有关属性的内容。

## 4.3 嵌套的 HTML 元素

大多数 HTML 元素可以嵌套（可以包含其他 HTML 元素）。

HTML 文档由嵌套的 HTML 元素构成。

HTML 文档实例

<html>

<body>

<p>This is my first paragraph.</p>

</body>

</html>

上面的例子包含三个 HTML 元素。

## 4.4 HTML 实例解释

### 4.4.1 <p> 元素：

<p>This is my first paragraph.</p>

这个 <p> 元素定义了 HTML 文档中的一个段落。

这个元素拥有一个开始标签 <p>，以及一个结束标签 </p>。

元素内容是：This is my first paragraph。

### 4.4.2 <body> 元素：

<body>

<p>This is my first paragraph.</p>

</body>

<body> 元素定义了 HTML 文档的主体。

这个元素拥有一个开始标签 <body>，以及一个结束标签 </body>。

元素内容是另一个 HTML 元素（p 元素）。

### 4.4.3 <html> 元素：

<html>

<body>

<p>This is my first paragraph.</p>

</body>

</html>

<html> 元素定义了整个 HTML 文档。

这个元素拥有一个开始标签 <html>，以及一个结束标签 </html>。

元素内容是另一个 HTML 元素（body 元素）。

## 4.5 不要忘记结束标签

即使您忘记了使用结束标签，大多数浏览器也会正确地显示 HTML：

<p>This is a paragraph

<p>This is a paragraph

上面的例子在大多数浏览器中都没问题，但不要依赖这种做法。忘记使用结束标签会产生不可预料的结果或错误。

注释：未来的 HTML 版本不允许省略结束标签。

## 4.6 空的 HTML 元素

没有内容的 HTML 元素被称为空元素。空元素是在开始标签中关闭的。

<br> 就是没有关闭标签的空元素（<br> 标签定义换行）。

在 XHTML、XML 以及未来版本的 HTML 中，所有元素都必须被关闭。

在开始标签中添加斜杠，比如 <br />，是关闭空元素的正确方法，HTML、XHTML 和 XML 都接受这种方式。

即使 <br> 在所有浏览器中都是有效的，但使用 <br /> 其实是更长远的保障。

## 4.7 HTML 提示：使用小写标签

HTML 标签对大小写不敏感：<P> 等同于 <p>。许多网站都使用大写的 HTML 标签。

W3School 使用的是小写标签，因为万维网联盟（W3C）在 HTML 4 中推荐使用小写，而在未来 (X)HTML 版本中强制使用小写。

# 第5章 HTML 属性

性为 HTML 元素提供附加信息。

## 5.1 HTML 属性

HTML 标签可以拥有属性。属性提供了有关 HTML 元素的更多的信息。

属性总是以名称/值对的形式出现，比如：name="value"。

属性总是在 HTML 元素的开始标签中规定。

## 5.2 属性实例

HTML 链接由 <a> 标签定义。链接的地址在 href 属性中指定：

<html>

<body>

<a href="http://www.w3school.com.cn">This is a link</a>

</body>

</html>

## 5.3 更多 HTML 属性实例

### 5.3.1 属性例子 1:

<h1> 定义标题的开始。

<h1 align="center"> 拥有关于对齐方式的附加信息。

TIY : 居中排列标题

### 5.3.2 属性例子 2:

<body> 定义 HTML 文档的主体。

<body bgcolor="yellow"> 拥有关于背景颜色的附加信息。

TIY : 背景颜色

### 5.3.3 属性例子 3:

<table> 定义 HTML 表格。（您将在稍后的章节学习到更多有关 HTML 表格的内容）

<table border="1"> 拥有关于表格边框的附加信息。

## 5.4 HTML 提示：使用小写属性

属性和属性值对大小写不敏感。

不过，万维网联盟在其 HTML 4 推荐标准中推荐小写的属性/属性值。

而新版本的 (X)HTML 要求使用小写属性。

## 5.5 始终为属性值加引号

属性值应该始终被包括在引号内。双引号是最常用的，不过使用单引号也没有问题。

在某些个别的情况下，比如属性值本身就含有双引号，那么您必须使用单引号，例如：

name='Bill "HelloWorld" Gates'

## 5.6 HTML 属性参考手册

我们的完整的 HTML 参考手册提供了每个 HTML 元素可使用的合法属性的完整列表：

|  |  |  |
| --- | --- | --- |
| 属性 | 值 | 描述 |
| class | classname | 规定元素的类名（classname） |
| id | id | 规定元素的唯一 id |
| style | style\_definition | 规定元素的行内样式（inline style） |
| title | text | 规定元素的额外信息（可在工具提示中显示） |

# 第6章 HTML 标题

## 6.1 HTML 标题

标题（Heading）是通过 <h1> - <h6> 等标签进行定义的。

<h1> 定义最大的标题。<h6> 定义最小的标题。

实例

<html>

<body>

<h1>This is heading 1</h1>

<h2>This is heading 2</h2>

<h3>This is heading 3</h3>

<h4>This is heading 4</h4>

<h5>This is heading 5</h5>

<h6>This is heading 6</h6>

<p>请仅仅把标题标签用于标题文本。不要仅仅为了产生粗体文本而使用它们。请使用其它标签或 CSS 代替。</p>

</body>

</html>

注释：浏览器会自动地在标题的前后添加空行。

注释：默认情况下，HTML 会自动地在块级元素前后添加一个额外的空行，比如段落、标题元素前后。

## 6.2 标题很重要

请确保将 HTML heading 标签只用于标题。不要仅仅是为了产生粗体或大号的文本而使用标题。

搜索引擎使用标题为您的网页的结构和内容编制索引。

因为用户可以通过标题来快速浏览您的网页，所以用标题来呈现文档结构是很重要的。

应该将 h1 用作主标题（最重要的），其后是 h2（次重要的），再其次是 h3，以此类推。

## 6.3 HTML 水平线

<hr /> 标签在 HTML 页面中创建水平线。

hr 元素可用于分隔内容。

<html>

<body>

<p>hr 标签定义水平线：</p>

<hr />

<p>这是段落。</p>

<hr />

<p>这是段落。</p>

<hr />

<p>这是段落。</p>

</body>

</html>

提示：使用水平线 (<hr> 标签) 来分隔文章中的小节是一个办法（但并不是唯一的办法）。

## 6.4 HTML 注释

可以将注释插入 HTML 代码中，这样可以提高其可读性，使代码更易被人理解。浏览器会忽略注释，也不会显示它们。

注释是这样写的：

实例

<!-- This is a comment -->

注释：开始括号之后（左边的括号）需要紧跟一个叹号，结束括号之前（右边的括号）不需要。

提示：合理地使用注释可以对未来的代码编辑工作产生帮助。

## 6.5 HTML 提示 - 如何查看源代码

您一定曾经在看到某个网页时惊叹道 “WOW! 这是如何实现的？”

如果您想找到其中的奥秘，只需要单击右键，然后选择“查看源文件”（IE）或“查看页面源代码”（Firefox），其他浏览器的做法也是类似的。这么做会打开一个包含页面 HTML 代码的窗口。

## 6.6 HTML 标签参考手册

W3School 的标签参考手册提供了有关这些标题及其属性的更多信息。

您将在本教程下面的章节中学到更多有关 HTML 标签和属性的知识。

|  |  |
| --- | --- |
| 标签 | 描述 |
| <html> | 定义 HTML 文档。 |
| <body> | 定义文档的主体。 |
| <h1> to <h6> | 定义 HTML 标题 |
| <hr> | 定义水平线。 |
| <!--...--> | 定义注释。 |

# 第7章 HTML 段落

可以把 HTML 文档分割为若干段落。

## 7.1 HTML 段落

段落是通过 <p> 标签定义的。

<html>

<body>

<p>这是段落。</p>

<p>这是段落。</p>

<p>这是段落。</p>

<p>段落元素由 p 标签定义。</p>

</body>

</html>

注释：浏览器会自动地在段落的前后添加空行。（<p> 是块级元素）

提示：使用空的段落标记 <p></p> 去插入一个空行是个坏习惯。用 <br /> 标签代替它！（但是不要用 <br /> 标签去创建列表。不要着急，您将在稍后的篇幅学习到 HTML 列表。）

## 7.2 不要忘记结束标签

即使忘了使用结束标签，大多数浏览器也会正确地将 HTML 显示出来：

<html>

<body>

<p>This is a paragraph.

<p>This is a paragraph.

<p>This is a paragraph.

<p>不要忘记关闭你的 HTML 标签！</p>

</body>

</html>

上面的例子在大多数浏览器中都没问题，但不要依赖这种做法。忘记使用结束标签会产生意想不到的结果和错误。

注释：在未来的 HTML 版本中，不允许省略结束标签。

提示：通过结束标签来关闭 HTML 是一种经得起未来考验的 HTML 编写方法。清楚地标记某个元素在何处开始，并在何处结束，不论对您还是对浏览器来说，都会使代码更容易理解。

## 7.3 HTML 折行

如果您希望在不产生一个新段落的情况下进行换行（新行），请使用 <br /> 标签：

<html>

<body>

<p>

To break<br />lines<br />in a<br />paragraph,<br />use the br tag.

</p>

</body>

</html>

<br /> 元素是一个空的 HTML 元素。由于关闭标签没有任何意义，因此它没有结束标签。

## 7.4 <br> 还是 <br />

您也许发现 <br> 与 <br /> 很相似。

在 XHTML、XML 以及未来的 HTML 版本中，不允许使用没有结束标签（闭合标签）的 HTML 元素。

即使 <br> 在所有浏览器中的显示都没有问题，使用 <br /> 也是更长远的保障。

## 7.5 HTML 输出 - 有用的提示

我们无法确定 HTML 被显示的确切效果。屏幕的大小，以及对窗口的调整都可能导致不同的结果。

对于 HTML，您无法通过在 HTML 代码中添加额外的空格或换行来改变输出的效果。

当显示页面时，浏览器会移除源代码中多余的空格和空行。所有连续的空格或空行都会被算作一个空格。需要注意的是，HTML 代码中的所有连续的空行（换行）也被显示为一个空格。

<html>

<body>

<h1>春晓</h1>

<p>

春眠不觉晓，

处处闻啼鸟。

夜来风雨声，

花落知多少。

</p>

<p>注意，浏览器忽略了源代码中的排版（省略了多余的空格和换行）。</p>

</body>

</html>

## 7.6 HTML 标签参考手册

W3School 的标签参考手册提供了有关 HTML 元素及其属性的更多信息。

|  |  |
| --- | --- |
| 标签 | 描述 |
| <p> | 定义段落。 |
| <br /> | 插入单个折行（换行）。 |

# 第8章 HTML 样式

style 属性用于改变 HTML 元素的样式。

<html>

<body style="background-color:PowderBlue;">

<h1>Look! Styles and colors</h1>

<p style="font-family:verdana;color:red">

This text is in Verdana and red</p>

<p style="font-family:times;color:green">

This text is in Times and green</p>

<p style="font-size:30px">This text is 30 pixels high</p>

</body>

</html>

## 8.1 HTML 的 style 属性

style 属性的作用：

提供了一种改变所有 HTML 元素的样式的通用方法。

样式是 HTML 4 引入的，它是一种新的首选的改变 HTML 元素样式的方式。通过 HTML 样式，能够通过使用 style 属性直接将样式添加到 HTML 元素，或者间接地在独立的样式表中（CSS 文件）进行定义。

您可以在我们的 CSS 教程中学习关于样式和 CSS 的所有知识。

在我们的 HTML 教程中，我们将使用 style 属性向您讲解 HTML 样式。

## 8.2 不赞成使用的标签和属性

在 HTML 4 中，有若干的标签和属性是被废弃的。被废弃（Deprecated）的意思是在未来版本的 HTML 和 XHTML 中将不支持这些标签和属性。

这里传达的信息很明确：请避免使用这些被废弃的标签和属性！

应该避免使用下面这些标签和属性：

|  |  |
| --- | --- |
| 标签 | 描述 |
| <center> | 定义居中的内容。 |
| <font> 和 <basefont> | 定义 HTML 字体。 |
| <s> 和 <strike> | 定义删除线文本 |
| <u> | 定义下划线文本 |
| 属性 | 描述 |
| align | 定义文本的对齐方式 |
| bgcolor | 定义背景颜色 |
| color | 定义文本颜色 |

对于以上这些标签和属性：请使用样式代替！

## 8.2 HTML 样式实例 - 背景颜色

background-color 属性为元素定义了背景颜色

<html>

<body style="background-color:yellow">

<h2 style="background-color:red">This is a heading</h2>

<p style="background-color:green">This is a paragraph.</p>

</body>

</html>

style 属性淘汰了“旧的” bgcolor 属性。

## 8.3 HTML 样式实例 - 字体、颜色和尺寸

font-family、color 以及 font-size 属性分别定义元素中文本的字体系列、颜色和字体尺寸：

<html>

<body>

<h1 style="font-family:verdana">A heading</h1>

<p style="font-family:arial;color:red;font-size:20px;">A paragraph.</p>

</body>

</html>

style 属性淘汰了旧的 <font> 标签。

<html>

<body>

<h1><font face="verdana">A heading</font></h1>

<p><font size="5" face="arial" color="red">A paragraph.</font></p>

</body>

</html>

## 8.4 HTML 样式实例 - 文本对齐

text-align 属性规定了元素中文本的水平对齐方式：

<html>

<body>

<h1 style="text-align:center">This is a heading</h1>

<p>上面的标题相对于页面居中对齐。</p>

</body>

</html>

style 属性淘汰了旧的 "align" 属性。

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

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

<html>

<body>

<h1 align="center">This is heading 1</h1>

<p>上面的标题在页面中进行了居中排列。上面的标题在页面中进行了居中排列。上面的标题在页面中进行了居中排列。</p>

</body>

</html>

# 第9章 HTML 文本格式化

HTML 可定义很多供格式化输出的元素，比如粗体和斜体字。

下面有很多例子，您可以亲自试试：

文本格式化标签：

|  |  |
| --- | --- |
| 标签 | 描述 |
| <b> | 定义粗体文本。 |
| <big> | 定义大号字。 |
| <em> | 定义着重文字。 |
| <i> | 定义斜体字。 |
| <small> | 定义小号字。 |
| <strong> | 定义加重语气。 |
| <sub> | 定义下标字。 |
| <sup> | 定义上标字。 |
| <ins> | 定义插入字。 |
| <del> | 定义删除字。 |
| <s> | 不赞成使用。使用 <del> 代替。 |
| <strike> | 不赞成使用。使用 <del> 代替。 |
| <u> | 不赞成使用。使用样式（style）代替。 |

“计算机输出”标签

|  |  |
| --- | --- |
| 标签 | 描述 |
| <code> | 定义计算机代码。 |
| <kbd> | 定义键盘码。 |
| <samp> | 定义计算机代码样本。 |
| <tt> | 定义打字机代码。 |
| <var> | 定义变量。 |
| <pre> | 定义预格式文本。 |
| <listing> | 不赞成使用。使用 <pre> 代替。 |
| <plaintext> | 不赞成使用。使用 <pre> 代替。 |
| <xmp> | 不赞成使用。使用 <pre> 代替。 |

引用、引用和术语定义

|  |  |
| --- | --- |
| 标签 | 描述 |
| <abbr> | 定义缩写。 |
| <acronym> | 定义首字母缩写。 |
| <address> | 定义地址。 |
| <bdo> | 定义文字方向。 |
| <blockquote> | 定义长的引用。 |
| <q> | 定义短的引用语。 |
| <cite> | 定义引用、引证。 |
| <dfn> | 定义一个定义项目。 |

改变文本的外观和含义：<https://www.w3school.com.cn/html/html_style.asp>第10章 HTML 引用

## 10.1 引用（Quotation）

这是摘自 WWF 网站的引文：

五十年来，WWF 一直致力于保护自然界的未来。 世界领先的环保组织，WWF 工作于 100 个国家，并得到美国一百二十万会员及全球近五百万会员的支持。

## 10.2 HTML <q> 用于短的引用

HTML <q> 元素定义短的引用。

浏览器通常会为 <q> 元素包围引号。

<!DOCTYPE html>

<html>

<body>

<p>浏览器通常会在 q 元素周围包围引号。</p>

<p>WWF 的目标是 <q>构建人与自然和谐相处的世界。</q></p>

</body>

</html>

## 10.3 用于长引用的 HTML <blockquote>

HTML <blockquote> 元素定义被引用的节。

浏览器通常会对 <blockquote> 元素进行缩进处理。

<!DOCTYPE html>

<html>

<body>

<p>浏览器通常会对 blockquote 元素进行缩进处理。</p>

<blockquote cite="http://www.worldwildlife.org/who/index.html">

五十年来，WWF 一直致力于保护自然界的未来。

WWF 工作于 100 个国家，并得到美国一百二十万会员及全球近五百万会员的支持。

</blockquote>

</body>

</html>

## 10.4 用于缩略词的 HTML <abbr>

HTML <abbr> 元素定义缩写或首字母缩略语。

对缩写进行标记能够为浏览器、翻译系统以及搜索引擎提供有用的信息。

<!DOCTYPE html>

<html>

<body>

<p><abbr title="World Health Organization">WHO</abbr> 成立于 1948 年。</p>

<p>对缩写进行标记能够为浏览器、翻译系统以及搜索引擎提供有用的信息。</p>

</body>

</html>

## 10.5 用于定义的 HTML <dfn>

HTML <dfn> 元素定义项目或缩写的定义。

<dfn> 的用法，按照 HTML5 标准中的描述，有点复杂：

1. 如果设置了 <dfn> 元素的 title 属性，则定义项目：

<!DOCTYPE html>

<html>

<body>

<p>The <dfn title="World Health Organization">WHO</dfn> was founded in 1948.</p>

<p>对缩写进行标记能够为浏览器、翻译系统以及搜索引擎提供有用的信息。</p>

</body>

</html>

2. 如果 <dfn> 元素包含具有标题的 <abbr> 元素，则 title 定义项目：

<!DOCTYPE html>

<html>

<body>

<p>The

<dfn><abbr title="World Health Organization">WHO</abbr></dfn>

was founded in 1948.

</p>

<p>对缩写进行标记能够为浏览器、翻译系统以及搜索引擎提供有用的信息。</p>

</body>

</html>

3. 否则，<dfn> 文本内容即是项目，并且父元素包含定义。

<!DOCTYPE html>

<html>

<body>

<p>The

<dfn>WHO</dfn> World Health Organization was founded in 1948.

</p>

<p>对缩写进行标记能够为浏览器、翻译系统以及搜索引擎提供有用的信息。</p>

</body>

</html>

注释：如果您希望简而化之，请使用第一条，或使用 <abbr> 代替。

## 10.6 用于联系信息的 HTML <address>

HTML <address> 元素定义文档或文章的联系信息（作者/拥有者）。

此元素通常以斜体显示。大多数浏览器会在此元素前后添加折行。

<!DOCTYPE html>

<html>

<body>

<p>HTML address 元素定义文档或文章的联系信息（作者/拥有者）。</p>

<address>

Written by Bill Gates.<br>

Visit us at:<br>

Example.com<br>

Box 564, Disneyland<br>

USA

</address>

</body>

</html>

## 10.7 用于著作标题的 HTML <cite>

HTML <cite> 元素定义著作的标题。

浏览器通常会以斜体显示 <cite> 元素。

<!DOCTYPE html>

<html>

<body>

<p>HTML cite 元素定义著作的标题。</p>

<p>浏览器通常会以斜体显示 cite 元素。</p>

<img src="img\_the\_scream.jpg" width="220" height="277" alt="The Scream">

<p><cite>The Scream</cite> by Edward Munch. Painted in 1893.</p>

</body>

</html>

## 10.8 用于双向重写的 HTML <bdo>

HTML <bdo> 元素定义双流向覆盖（bi-directional override）。

<bdo> 元素用于覆盖当前文本方向：

<!DOCTYPE html>

<html>

<body>

<p>

如果您的浏览器支持 bdo，则文本将从右向左进行书写 (rtl)：

</p>

<bdo dir="rtl">

This line will be written from right to left

</bdo>

</body>

</html>

## 10.9 HTML 引文、引用和定义元素

|  |  |
| --- | --- |
| 标签 | 描述 |
| <abbr> | 定义缩写或首字母缩略语。 |
| <address> | 定义文档作者或拥有者的联系信息。 |
| <bdo> | 定义文本方向。 |
| <blockquote> | 定义从其他来源引用的节。 |
| <dfn> | 定义项目或缩略词的定义。 |
| <q> | 定义短的行内引用。 |
| <cite> | 定义著作的标题。 |

# 第11章 HTML 注释

注释标签 <!-- 与 --> 用于在 HTML 插入注释。

## 11.1 HTML 注释标签

实例

<!-- 在此处写注释 -->

注释：在开始标签中有一个惊叹号，但是结束标签中没有。

浏览器不会显示注释，但是能够帮助记录您的 HTML 文档。

您可以利用注释在 HTML 中放置通知和提醒信息：

<!DOCTYPE html>

<html>

<body>

<!--这是一段注释。注释不会在浏览器中显示。-->

<p>这是一段普通的段落。</p>

</body>

</html>

注释对于 HTML 纠错也大有帮助，因为您可以一次注释一行 HTML 代码，以搜索错误：

<!DOCTYPE html>

<html>

<body>

<!-- 此时不显示图片

<img border="0" src="/i/tulip\_ballade.jpg" alt="Tulip">

-->

</body>

</html>

## 11.2 条件注释

您也许会在 HTML 中偶尔发现条件注释：

<!--[if IE 8]>

.... some HTML here ....

<![endif]-->

条件注释定义只有 Internet Explorer 执行的 HTML 标签。

软件程序标签

各种 HTML 软件程序也能够生成 HTML 注释。

例如 <!--webbot bot--> 标签会被包围在由 FrontPage 和 Expression Web 创建的 HTML 注释中。

作为一项规则，这些标签的存在，有助于对创建这些标签的软件的支持。

# 第12章 HTML 颜色

颜色由红色、绿色、蓝色混合而成。

## 12.1 颜色值

颜色由一个十六进制符号来定义，这个符号由红色、绿色和蓝色的值组成（RGB）。

每种颜色的最小值是 0（十六进制：#00）。最大值是 255（十六进制：#FF）。

这个表格给出了由三种颜色混合而成的具体效果：

|  |  |  |
| --- | --- | --- |
| Color | Color HEX | Color RGB |
|  | #000000 | rgb(0,0,0) |
|  | #FF0000 | rgb(255,0,0) |
|  | #00FF00 | rgb(0,255,0) |
|  | #0000FF | rgb(0,0,255) |
|  | #FFFF00 | rgb(255,255,0) |
|  | #00FFFF | rgb(0,255,255) |
|  | #FF00FF | rgb(255,0,255) |
|  | #C0C0C0 | rgb(192,192,192) |
|  | #FFFFFF | rgb(255,255,255) |

## 12.2 颜色名

大多数的浏览器都支持颜色名集合。

提示：仅仅有 16 种颜色名被 W3C 的 HTML4.0 标准所支持。它们是：aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, yellow。

如果需要使用其它的颜色，需要使用十六进制的颜色值。

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Color | Color HEX | Color Name |
|  | #F0F8FF | AliceBlue |
|  | #FAEBD7 | AntiqueWhite |
|  | #7FFFD4 | Aquamarine |
|  | #000000 | Black |
|  | #0000FF | Blue |
|  | #8A2BE2 | BlueViolet |

## 12.3 Web安全色

数年以前，当大多数计算机仅支持 256 种颜色的时候，一系列 216 种 Web 安全色作为 Web 标准被建议使用。其中的原因是，微软和 Mac 操作系统使用了 40 种不同的保留的固定系统颜色（双方大约各使用 20 种）。

我们不确定如今这么做的意义有多大，因为越来越多的计算机有能力处理数百万种颜色，不过做选择的还是你自己。

216 跨平台色

最初，216 跨平台 web 安全色被用来确保：当计算机使用 256 色调色板时，所有的计算机能够正确地显示所有的颜色。

## 12.4 HTML 颜色名

提示：仅有 16 种颜色名被 W3C 的 HTML 4.0 标准支持，它们是：aqua、black、blue、fuchsia、gray、green、lime、maroon、navy、olive、purple、red、silver、teal、white、yellow。

如果使用其它颜色的话，就应该使用十六进制的颜色值。

颜色名列表

单击一个颜色名或者 16 进制值，就可以查看与不同文字颜色搭配的背景颜色。

颜色名 十六进制颜色值 颜色

AliceBlue #F0F8FF

AntiqueWhite #FAEBD7

Aqua #00FFFF

Aquamarine #7FFFD4

Azure #F0FFFF

Beige #F5F5DC

Bisque #FFE4C4

Black #000000

BlanchedAlmond #FFEBCD

Blue #0000FF

BlueViolet #8A2BE2

Brown #A52A2A

BurlyWood #DEB887

CadetBlue #5F9EA0

Chartreuse #7FFF00

Chocolate #D2691E

Coral #FF7F50

CornflowerBlue #6495ED

Cornsilk #FFF8DC

Crimson #DC143C

Cyan #00FFFF

DarkBlue #00008B

DarkCyan #008B8B

DarkGoldenRod #B8860B

DarkGray #A9A9A9

DarkGreen #006400

DarkKhaki #BDB76B

DarkMagenta #8B008B

DarkOliveGreen #556B2F

Darkorange #FF8C00

DarkOrchid #9932CC

DarkRed #8B0000

DarkSalmon #E9967A

DarkSeaGreen #8FBC8F

DarkSlateBlue #483D8B

DarkSlateGray #2F4F4F

DarkTurquoise #00CED1

DarkViolet #9400D3

DeepPink #FF1493

DeepSkyBlue #00BFFF

DimGray #696969

DodgerBlue #1E90FF

Feldspar #D19275

FireBrick #B22222

FloralWhite #FFFAF0

ForestGreen #228B22

Fuchsia #FF00FF

Gainsboro #DCDCDC

GhostWhite #F8F8FF

Gold #FFD700

GoldenRod #DAA520

Gray #808080

Green #008000

GreenYellow #ADFF2F

HoneyDew #F0FFF0

HotPink #FF69B4

IndianRed #CD5C5C

Indigo #4B0082

Ivory #FFFFF0

Khaki #F0E68C

Lavender #E6E6FA

LavenderBlush #FFF0F5

LawnGreen #7CFC00

LemonChiffon #FFFACD

LightBlue #ADD8E6

LightCoral #F08080

LightCyan #E0FFFF

LightGoldenRodYellow #FAFAD2

LightGrey #D3D3D3

LightGreen #90EE90

LightPink #FFB6C1

LightSalmon #FFA07A

LightSeaGreen #20B2AA

LightSkyBlue #87CEFA

LightSlateBlue #8470FF

LightSlateGray #778899

LightSteelBlue #B0C4DE

LightYellow #FFFFE0

Lime #00FF00

LimeGreen #32CD32

Linen #FAF0E6

Magenta #FF00FF

Maroon #800000

MediumAquaMarine #66CDAA

MediumBlue #0000CD

MediumOrchid #BA55D3

MediumPurple #9370D8

MediumSeaGreen #3CB371

MediumSlateBlue #7B68EE

MediumSpringGreen #00FA9A

MediumTurquoise #48D1CC

MediumVioletRed #C71585

MidnightBlue #191970

MintCream #F5FFFA

MistyRose #FFE4E1

Moccasin #FFE4B5

NavajoWhite #FFDEAD

Navy #000080

OldLace #FDF5E6

Olive #808000

OliveDrab #6B8E23

Orange #FFA500

OrangeRed #FF4500

Orchid #DA70D6

PaleGoldenRod #EEE8AA

PaleGreen #98FB98

PaleTurquoise #AFEEEE

PaleVioletRed #D87093

PapayaWhip #FFEFD5

PeachPuff #FFDAB9

Peru #CD853F

Pink #FFC0CB

Plum #DDA0DD

PowderBlue #B0E0E6

Purple #800080

Red #FF0000

RosyBrown #BC8F8F

RoyalBlue #4169E1

SaddleBrown #8B4513

Salmon #FA8072

SandyBrown #F4A460

SeaGreen #2E8B57

SeaShell #FFF5EE

Sienna #A0522D

Silver #C0C0C0

SkyBlue #87CEEB

SlateBlue #6A5ACD

SlateGray #708090

Snow #FFFAFA

SpringGreen #00FF7F

SteelBlue #4682B4

Tan #D2B48C

Teal #008080

Thistle #D8BFD8

Tomato #FF6347

Turquoise #40E0D0

Violet #EE82EE

VioletRed #D02090

Wheat #F5DEB3

White #FFFFFF

WhiteSmoke #F5F5F5

Yellow #FFFF00

YellowGreen #9ACD32

# 第13章 HTML CSS

通过使用 HTML4.0，所有的格式化代码均可移出 HTML 文档，然后移入一个独立的样式表。

## 13.1 如何使用样式

当浏览器读到一个样式表，它就会按照这个样式表来对文档进行格式化。有以下三种方式来插入样式表：

外部样式表

当样式需要被应用到很多页面的时候，外部样式表将是理想的选择。使用外部样式表，你就可以通过更改一个文件来改变整个站点的外观。

<head>

<link rel="stylesheet" type="text/css" href="mystyle.css">

</head>

内部样式表

当单个文件需要特别样式时，就可以使用内部样式表。你可以在 head 部分通过 <style> 标签定义内部样式表。

<head>

<style type="text/css">

body {background-color: red}

p {margin-left: 20px}

</style>

</head>

内联样式

当特殊的样式需要应用到个别元素时，就可以使用内联样式。使用内联样式的方法是在相关的标签中使用样式属性。样式属性可以包含任何 CSS 属性。以下实例显示出如何改变段落的颜色和左外边距。

<p style="color: red; margin-left: 20px">

This is a paragraph

</p>

|  |  |
| --- | --- |
|  |  |
| 标签 | 描述 |
| <style> | 定义样式定义。 |
| <link> | 定义资源引用。 |
| <div> | 定义文档中的节或区域（块级）。 |
| <span> | 定义文档中的行内的小块或区域。 |
| <font> | 规定文本的字体、字体尺寸、字体颜色。不赞成使用。请使用样式。 |
| <basefont> | 定义基准字体。不赞成使用。请使用样式。 |
| <center> | 对文本进行水平居中。不赞成使用。请使用样式 |

# 第14章 HTML 链接

HTML 使用超级链接与网络上的另一个文档相连。

几乎可以在所有的网页中找到链接。点击链接可以从一张页面跳转到另一张页面。

## 14.1 HTML 超链接（链接）

超链接可以是一个字，一个词，或者一组词，也可以是一幅图像，您可以点击这些内容来跳转到新的文档或者当前文档中的某个部分。

当您把鼠标指针移动到网页中的某个链接上时，箭头会变为一只小手。

我们通过使用 <a> 标签在 HTML 中创建链接。

有两种使用 <a> 标签的方式：

通过使用 href 属性 - 创建指向另一个文档的链接

通过使用 name 属性 - 创建文档内的书签

## 14.2 HTML 链接语法

链接的 HTML 代码很简单。它类似这样：

<a href="url">Link text</a>

href 属性规定链接的目标。

开始标签和结束标签之间的文字被作为超级链接来显示。

实例

<a href="http://www.w3school.com.cn/">Visit W3School</a>

上面这行代码显示为：Visit W3School

点击这个超链接会把用户带到 W3School 的首页。

提示："链接文本" 不必一定是文本。图片或其他 HTML 元素都可以成为链接。

HTML 链接 - target 属性

使用 Target 属性，你可以定义被链接的文档在何处显示。

下面的这行会在新窗口打开文档：

<html>

<body>

<a href="http://www.w3school.com.cn/" target="\_blank">Visit W3School!</a>

<p>如果把链接的 target 属性设置为 "\_blank"，该链接会在新窗口中打开。</p>

</body>

</html>

## 14.3 HTML 链接 - name 属性

name 属性规定锚（anchor）的名称。

您可以使用 name 属性创建 HTML 页面中的书签。

书签不会以任何特殊方式显示，它对读者是不可见的。

当使用命名锚（named anchors）时，我们可以创建直接跳至该命名锚（比如页面中某个小节）的链接，这样使用者就无需不停地滚动页面来寻找他们需要的信息了。

命名锚的语法：

<a name="label">锚（显示在页面上的文本）</a>

提示：锚的名称可以是任何你喜欢的名字。

提示：您可以使用 id 属性来替代 name 属性，命名锚同样有效。

实例

首先，我们在 HTML 文档中对锚进行命名（创建一个书签）：

<a name="tips">基本的注意事项 - 有用的提示</a>

然后，我们在同一个文档中创建指向该锚的链接：

<a href="#tips">有用的提示</a>

您也可以在其他页面中创建指向该锚的链接：

<a href="http://www.w3school.com.cn/html/html\_links.asp#tips">有用的提示</a>

在上面的代码中，我们将 # 符号和锚名称添加到 URL 的末端，就可以直接链接到 tips 这个命名锚了。

## 14.4 基本的注意事项 - 有用的提示：

注释：请始终将正斜杠添加到子文件夹。假如这样书写链接：href="http://www.w3school.com.cn/html"，就会向服务器产生两次 HTTP 请求。这是因为服务器会添加正斜杠到这个地址，然后创建一个新的请求，就像这样：href="http://www.w3school.com.cn/html/"。

提示：命名锚经常用于在大型文档开始位置上创建目录。可以为每个章节赋予一个命名锚，然后把链接到这些锚的链接放到文档的上部。如果您经常访问百度百科，您会发现其中几乎每个词条都采用这样的导航方式。

提示：假如浏览器找不到已定义的命名锚，那么就会定位到文档的顶端。不会有错误发生。

## 14.5 HTML 链接标签

|  |  |
| --- | --- |
| 标签 | 描述 |
| <a> | 定义锚。 |

# 第15章 HTML 图像

通过使用 HTML，可以在文档中显示图像。

## 15.1 图像标签（<img>）和源属性（Src）

在 HTML 中，图像由 <img> 标签定义。

<img> 是空标签，意思是说，它只包含属性，并且没有闭合标签。

要在页面上显示图像，你需要使用源属性（src）。src 指 "source"。源属性的值是图像的 URL 地址。

定义图像的语法是：

<img src="url" />

URL 指存储图像的位置。如果名为 "boat.gif" 的图像位于 www.w3school.com.cn 的 images 目录中，那么其 URL 为 http://www.w3school.com.cn/images/boat.gif。

浏览器将图像显示在文档中图像标签出现的地方。如果你将图像标签置于两个段落之间，那么浏览器会首先显示第一个段落，然后显示图片，最后显示第二段。

## 15.2 替换文本属性（Alt）

alt 属性用来为图像定义一串预备的可替换的文本。替换文本属性的值是用户定义的。

<img src="boat.gif" alt="Big Boat">

当浏览器无法载入图像时，替换文本属性可告诉读者他们失去的信息。此时，浏览器将显示这个替代性的文本而不是图像。为页面上的图像都加上替换文本属性是个好习惯，这样有助于更好的显示信息，并且对于那些使用纯文本浏览器的人来说是非常有用的。

## 15.3 基本的注意事项 - 有用的提示：

假如某个 HTML 文件包含十个图像，那么为了正确显示这个页面，需要加载 11 个文件。加载图片是需要时间的，所以我们的建议是：慎用图片。

## 15.4 实例

**背景图片：**

<html>

<body background="/i/eg\_background.jpg">

<h3>图像背景</h3>

<p>gif 和 jpg 文件均可用作 HTML 背景。</p>

<p>如果图像小于页面，图像会进行重复。</p>

</body>

</html>

**排列图片：**

<html>

<body>

<h2>未设置对齐方式的图像：</h2>

<p>图像 <img src ="/i/eg\_cute.gif"> 在文本中</p>

<h2>已设置对齐方式的图像：</h2>

<p>图像 <img src="/i/eg\_cute.gif" align="bottom"> 在文本中</p>

<p>图像 <img src ="/i/eg\_cute.gif" align="middle"> 在文本中</p>

<p>图像 <img src ="/i/eg\_cute.gif" align="top"> 在文本中</p>

<p>请注意，bottom 对齐方式是默认的对齐方式。</p>

</body>

</html>

**浮动图片：**

<html>

<body>

<p>

<img src ="/i/eg\_cute.gif" align ="left">

带有图像的一个段落。图像的 align 属性设置为 "left"。图像将浮动到文本的左侧。

</p>

<p>

<img src ="/i/eg\_cute.gif" align ="right">

带有图像的一个段落。图像的 align 属性设置为 "right"。图像将浮动到文本的右侧。

</p>

</body>

</html>

**调整图片大小：**

<html>

<body>

<img src="/i/eg\_mouse.jpg" width="50" height="50">

<br />

<img src="/i/eg\_mouse.jpg" width="100" height="100">

<br />

<img src="/i/eg\_mouse.jpg" width="200" height="200">

<p>通过改变 img 标签的 "height" 和 "width" 属性的值，您可以放大或缩小图像。</p>

</body>

</html>

**为图片显示替换图片：**

<html>

<body>

<p>仅支持文本的浏览器无法显示图像，仅仅能够显示在图像的 "alt" 属性中指定的文本。在这里，"alt" 的文本是“向左转”。</p>

<p>请注意，如果您把鼠标指针移动到图像上，大多数浏览器会显示 "alt" 文本。</p>

<img src="/i/eg\_goleft.gif" alt="向左转" />

<p>如果无法显示图像，将显示 "alt" 属性中的文本：</p>

<img src="/i/eg\_goleft123.gif" alt="向左转" />

</body>

</html>

**制作图片链接：**

<html>

<body>

<p>

您也可以把图像作为链接来使用：

<a href="/example/html/lastpage.html">

<img border="0" src="/i/eg\_buttonnext.gif" />

</a>

</p>

</body>

</html>

**创建图片映射：**

<html>

<body>

<p>请点击图像上的星球，把它们放大。</p>

<img

src="/i/eg\_planets.jpg"

border="0" usemap="#planetmap"

alt="Planets" />

<map name="planetmap" id="planetmap">

<area

shape="circle"

coords="180,139,14"

href ="/example/html/venus.html"

target ="\_blank"

alt="Venus" />

<area

shape="circle"

coords="129,161,10"

href ="/example/html/mercur.html"

target ="\_blank"

alt="Mercury" />

<area

shape="rect"

coords="0,0,110,260"

href ="/example/html/sun.html"

target ="\_blank"

alt="Sun" />

</map>

<p><b>注释：</b>img 元素中的 "usemap" 属性引用 map 元素中的 "id" 或 "name" 属性（根据浏览器），所以我们同时向 map 元素添加了 "id" 和 "name" 属性。</p>

</body>

</html>

**把图像转换为图像映射：**

<!DOCTYPE html>

<html>

<body>

<p>请把鼠标移动到图像上，看一下状态栏的坐标如何变化。</p>

<a href="/example-2019/html/html\_ismap.html">

<img src="/i/eg\_planets.jpg" ismap />

</a>

</body>

</html>

## 15.5 图像标签

|  |  |
| --- | --- |
| 标签 | 描述 |
| <img> | 定义图像。 |
| <map> | 定义图像地图。 |
| <area> | 定义图像地图中的可点击区域。 |

# 第16章 HTML 表格

你可以使用 HTML 创建表格。

## 16.1 表格

表格由 <table> 标签来定义。每个表格均有若干行（由 <tr> 标签定义），每行被分割为若干单元格（由 <td> 标签定义）。字母 td 指表格数据（table data），即数据单元格的内容。数据单元格可以包含文本、图片、列表、段落、表单、水平线、表格等等。

<table border="1">

<tr>

<td>row 1, cell 1</td>

<td>row 1, cell 2</td>

</tr>

<tr>

<td>row 2, cell 1</td>

<td>row 2, cell 2</td>

</tr>

</table>

## 16.2 表格和边框属性

如果不定义边框属性，表格将不显示边框。有时这很有用，但是大多数时候，我们希望显示边框。

使用边框属性来显示一个带有边框的表格：

<table border="1">

<tr>

<td>Row 1, cell 1</td>

<td>Row 1, cell 2</td>

</tr>

</table>

## 16.3 表格的表头

表格的表头使用 <th> 标签进行定义。

大多数浏览器会把表头显示为粗体居中的文本：

<table border="1">

<tr>

<th>Heading</th>

<th>Another Heading</th>

</tr>

<tr>

<td>row 1, cell 1</td>

<td>row 1, cell 2</td>

</tr>

<tr>

<td>row 2, cell 1</td>

<td>row 2, cell 2</td>

</tr>

</table>

## 16.4 表格中的空单元格

在一些浏览器中，没有内容的表格单元显示得不太好。如果某个单元格是空的（没有内容），浏览器可能无法显示出这个单元格的边框。

<table border="1">

<tr>

<td>row 1, cell 1</td>

<td>row 1, cell 2</td>

</tr>

<tr>

<td></td>

<td>row 2, cell 2</td>

</tr>

</table>

注意：这个空的单元格的边框没有被显示出来。为了避免这种情况，在空单元格中添加一个空格占位符，就可以将边框显示出来。

<table border="1">

<tr>

<td>row 1, cell 1</td>

<td>row 1, cell 2</td>

</tr>

<tr>

<td>&nbsp;</td>

<td>row 2, cell 2</td>

</tr>

</table>

## 16.5 更多实例

**没有边框的表格**

<html>

<body>

<h4>这个表格没有边框：</h4>

<table>

<tr>

<td>100</td>

<td>200</td>

<td>300</td>

</tr>

<tr>

<td>400</td>

<td>500</td>

<td>600</td>

</tr>

</table>

<h4>这个表格也没有边框：</h4>

<table border="0">

<tr>

<td>100</td>

<td>200</td>

<td>300</td>

</tr>

<tr>

<td>400</td>

<td>500</td>

<td>600</td>

</tr>

</table>

</body>

</html>

**表格中的表头(Heading)**

<html>

<body>

<h4>表头：</h4>

<table border="1">

<tr>

<th>姓名</th>

<th>电话</th>

<th>电话</th>

</tr>

<tr>

<td>Bill Gates</td>

<td>555 77 854</td>

<td>555 77 855</td>

</tr>

</table>

<h4>垂直的表头：</h4>

<table border="1">

<tr>

<th>姓名</th>

<td>Bill Gates</td>

</tr>

<tr>

<th>电话</th>

<td>555 77 854</td>

</tr>

<tr>

<th>电话</th>

<td>555 77 855</td>

</tr>

</table>

</body>

</html>

**空单元格**

<html>

<body>

<table border="1">

<tr>

<td>Some text</td>

<td>Some text</td>

</tr>

<tr>

<td></td>

<td>Some text</td>

</tr>

</table>

<p>正如您看到的，其中一个单元没有边框。这是因为它是空的。在该单元中插入一个空格后，仍然没有边框。</p>

<p>我们的技巧是在单元中插入一个 no-breaking 空格。</p>

<p>no-breaking 空格是一个字符实体。如果您不清楚什么是字符实体，请阅读关于字符实体的章节。</p>

<p>no-breaking 空格由和号开始 ("&")，然后是字符"nbsp"，并以分号结尾(";")。</p>

</body>

</html>

**带有标题的表格**

<html>

<body>

<h4>这个表格有一个标题，以及粗边框：</h4>

<table border="6">

<caption>我的标题</caption>

<tr>

<td>100</td>

<td>200</td>

<td>300</td>

</tr>

<tr>

<td>400</td>

<td>500</td>

<td>600</td>

</tr>

</table>

</body>

</html>

**跨行或跨列的表格单元格**

<html>

<body>

<h4>横跨两列的单元格：</h4>

<table border="1">

<tr>

<th>姓名</th>

<th colspan="2">电话</th>

</tr>

<tr>

<td>Bill Gates</td>

<td>555 77 854</td>

<td>555 77 855</td>

</tr>

</table>

<h4>横跨两行的单元格：</h4>

<table border="1">

<tr>

<th>姓名</th>

<td>Bill Gates</td>

</tr>

<tr>

<th rowspan="2">电话</th>

<td>555 77 854</td>

</tr>

<tr>

<td>555 77 855</td>

</tr>

</table>

</body>

</html>

**表格内的标签**

<html>

<body>

<table border="1">

<tr>

<td>

<p>这是一个段落。</p>

<p>这是另一个段落。</p>

</td>

<td>这个单元包含一个表格：

<table border="1">

<tr>

<td>A</td>

<td>B</td>

</tr>

<tr>

<td>C</td>

<td>D</td>

</tr>

</table>

</td>

</tr>

<tr>

<td>这个单元包含一个列表：

<ul>

<li>苹果</li>

<li>香蕉</li>

<li>菠萝</li>

</ul>

</td>

<td>HELLO</td>

</tr>

</table>

</body>

</html>

**单元格边距(Cell padding)**

<html>

<body>

<h4>没有 cellpadding：</h4>

<table border="1">

<tr>

<td>First</td>

<td>Row</td>

</tr>

<tr>

<td>Second</td>

<td>Row</td>

</tr>

</table>

<h4>带有 cellpadding：</h4>

<table border="1"

cellpadding="10">

<tr>

<td>First</td>

<td>Row</td>

</tr>

<tr>

<td>Second</td>

<td>Row</td>

</tr>

</table>

</body>

</html>

**单元格间距(Cell spacing)**

<html>

<body>

<h4>没有 cellspacing：</h4>

<table border="1">

<tr>

<td>First</td>

<td>Row</td>

</tr>

<tr>

<td>Second</td>

<td>Row</td>

</tr>

</table>

<h4>带有 cellspacing：</h4>

<table border="1"

cellspacing="10">

<tr>

<td>First</td>

<td>Row</td>

</tr>

<tr>

<td>Second</td>

<td>Row</td>

</tr>

</table>

</body>

</html>

**向表格添加背景颜色或背景图像**

<html>

<body>

<h4>背景颜色：</h4>

<table border="1"

bgcolor="red">

<tr>

<td>First</td>

<td>Row</td>

</tr>

<tr>

<td>Second</td>

<td>Row</td>

</tr>

</table>

<h4>背景图像：</h4>

<table border="1"

background="/i/eg\_bg\_07.gif">

<tr>

<td>First</td>

<td>Row</td>

</tr>

<tr>

<td>Second</td>

<td>Row</td>

</tr>

</table>

</body>

</html>

**向表格单元添加背景颜色或者背景图像**

<html>

<body>

<h4>单元背景：</h4>

<table border="1">

<tr>

<td bgcolor="red">First</td>

<td>Row</td>

</tr>

<tr>

<td

background="/i/eg\_bg\_07.gif">

Second</td>

<td>Row</td>

</tr>

</table>

</body>

</html>

**在表格单元中排列内容**

<html>

<body>

<table width="400" border="1">

<tr>

<th align="left">消费项目....</th>

<th align="right">一月</th>

<th align="right">二月</th>

</tr>

<tr>

<td align="left">衣服</td>

<td align="right">$241.10</td>

<td align="right">$50.20</td>

</tr>

<tr>

<td align="left">化妆品</td>

<td align="right">$30.00</td>

<td align="right">$44.45</td>

</tr>

<tr>

<td align="left">食物</td>

<td align="right">$730.40</td>

<td align="right">$650.00</td>

</tr>

<tr>

<th align="left">总计</th>

<th align="right">$1001.50</th>

<th align="right">$744.65</th>

</tr>

</table>

</body>

</html>

**框架(frame)属性**

<html>

<body>

<p><b>注释：</b>frame 属性无法在 Internet Explorer 中正确地显示。</p>

<p>Table with frame="box":</p>

<table frame="box">

<tr>

<th>Month</th>

<th>Savings</th>

</tr>

<tr>

<td>January</td>

<td>$100</td>

</tr>

</table>

<p>Table with frame="above":</p>

<table frame="above">

<tr>

<th>Month</th>

<th>Savings</th>

</tr>

<tr>

<td>January</td>

<td>$100</td>

</tr>

</table>

<p>Table with frame="below":</p>

<table frame="below">

<tr>

<th>Month</th>

<th>Savings</th>

</tr>

<tr>

<td>January</td>

<td>$100</td>

</tr>

</table>

<p>Table with frame="hsides":</p>

<table frame="hsides">

<tr>

<th>Month</th>

<th>Savings</th>

</tr>

<tr>

<td>January</td>

<td>$100</td>

</tr>

</table>

<p>Table with frame="vsides":</p>

<table frame="vsides">

<tr>

<th>Month</th>

<th>Savings</th>

</tr>

<tr>

<td>January</td>

<td>$100</td>

</tr>

</table>

</body>

</html>

## 16.6 表格标签

|  |  |
| --- | --- |
| 表格 | 描述 |
| <table> | 定义表格 |
| <caption> | 定义表格标题。 |
| <th> | 定义表格的表头。 |
| <tr> | 定义表格的行。 |
| <td> | 定义表格单元。 |
| <thead> | 定义表格的页眉。 |
| <tbody> | 定义表格的主体。 |
| <tfoot> | 定义表格的页脚。 |
| <col> | 定义用于表格列的属性。 |
| <colgroup> | 定义表格列的组。 |

# 第17章 HTML 列表

HTML 支持有序、无序和定义列表

## 17.1 无序列表

无序列表是一个项目的列表，此列项目使用粗体圆点（典型的小黑圆圈）进行标记。

无序列表始于 <ul> 标签。每个列表项始于 <li>。

<ul>

<li>Coffee</li>

<li>Milk</li>

</ul>

浏览器显示如下：

Coffee

Milk

列表项内部可以使用段落、换行符、图片、链接以及其他列表等等。

## 17.2 有序列表

同样，有序列表也是一列项目，列表项目使用数字进行标记。

有序列表始于 <ol> 标签。每个列表项始于 <li> 标签。

<ol>

<li>Coffee</li>

<li>Milk</li>

</ol>

浏览器显示如下：

Coffee

Milk

列表项内部可以使用段落、换行符、图片、链接以及其他列表等等。

## 17.3 定义列表

自定义列表不仅仅是一列项目，而是项目及其注释的组合。

自定义列表以 <dl> 标签开始。每个自定义列表项以 <dt> 开始。每个自定义列表项的定义以 <dd> 开始。

<dl>

<dt>Coffee</dt>

<dd>Black hot drink</dd>

<dt>Milk</dt>

<dd>White cold drink</dd>

</dl>

浏览器显示如下：

Coffee

Black hot drink

Milk

White cold drink

定义列表的列表项内部可以使用段落、换行符、图片、链接以及其他列表等等。

## 17.4 更多实例

**不同类型的无序列表**

<html>

<body>

<h4>Disc 项目符号列表：</h4>

<ul type="disc">

<li>苹果</li>

<li>香蕉</li>

<li>柠檬</li>

<li>桔子</li>

</ul>

<h4>Circle 项目符号列表：</h4>

<ul type="circle">

<li>苹果</li>

<li>香蕉</li>

<li>柠檬</li>

<li>桔子</li>

</ul>

<h4>Square 项目符号列表：</h4>

<ul type="square">

<li>苹果</li>

<li>香蕉</li>

<li>柠檬</li>

<li>桔子</li>

</ul>

</body>

</html>

**不同类型的有序列表**

<html>

<body>

<h4>数字列表：</h4>

<ol>

<li>苹果</li>

<li>香蕉</li>

<li>柠檬</li>

<li>桔子</li>

</ol>

<h4>字母列表：</h4>

<ol type="A">

<li>苹果</li>

<li>香蕉</li>

<li>柠檬</li>

<li>桔子</li>

</ol>

<h4>小写字母列表：</h4>

<ol type="a">

<li>苹果</li>

<li>香蕉</li>

<li>柠檬</li>

<li>桔子</li>

</ol>

<h4>罗马字母列表：</h4>

<ol type="I">

<li>苹果</li>

<li>香蕉</li>

<li>柠檬</li>

<li>桔子</li>

</ol>

<h4>小写罗马字母列表：</h4>

<ol type="i">

<li>苹果</li>

<li>香蕉</li>

<li>柠檬</li>

<li>桔子</li>

</ol>

</body>

</html>

**嵌套列表**

<html>

<body>

<h4>一个嵌套列表：</h4>

<ul>

<li>咖啡</li>

<li>茶

<ul>

<li>红茶</li>

<li>绿茶</li>

</ul>

</li>

<li>牛奶</li>

</ul>

</body>

</html>

**嵌套列表 2**

<html>

<body>

<h4>一个嵌套列表：</h4>

<ul>

<li>咖啡</li>

<li>茶

<ul>

<li>红茶</li>

<li>绿茶

<ul>

<li>中国茶</li>

<li>非洲茶</li>

</ul>

</li>

</ul>

</li>

<li>牛奶</li>

</ul>

</body>

</html>

**定义列表**

<html>

<body>

<h2>一个定义列表：</h2>

<dl>

<dt>计算机</dt>

<dd>用来计算的仪器 ... ...</dd>

<dt>显示器</dt>

<dd>以视觉方式显示信息的装置 ... ...</dd>

</dl>

</body>

</html>

## 17.5 列表标签

|  |  |
| --- | --- |
| 标签 | 描述 |
| <ol> | 定义有序列表。 |
| <ul> | 定义无序列表。 |
| <li> | 定义列表项。 |
| <dl> | 定义定义列表。 |
| <dt> | 定义定义项目。 |
| <dd> | 定义定义的描述。 |
| <dir> | 已废弃。使用 <ul> 代替它。 |
| <menu> | 已废弃。使用 <ul> 代替它。 |

# 第18章 HTML <div> 和 <span>

可以通过 <div> 和 <span> 将 HTML 元素组合起来。

## 18.1 HTML 块元素

大多数 HTML 元素被定义为块级元素或内联元素。

编者注：“块级元素”译为 block level element，“内联元素”译为 inline element。

块级元素在浏览器显示时，通常会以新行来开始（和结束）。

例子：<h1>, <p>, <ul>, <table>

## 18.2 HTML 内联元素

内联元素在显示时通常不会以新行开始。

例子：<b>, <td>, <a>, <img>

## 18.3 HTML <div> 元素

HTML <div> 元素是块级元素，它是可用于组合其他 HTML 元素的容器。

<div> 元素没有特定的含义。除此之外，由于它属于块级元素，浏览器会在其前后显示折行。

如果与 CSS 一同使用，<div> 元素可用于对大的内容块设置样式属性。

<div> 元素的另一个常见的用途是文档布局。它取代了使用表格定义布局的老式方法。使用 <table> 元素进行文档布局不是表格的正确用法。<table> 元素的作用是显示表格化的数据。

## 18.4 HTML <span> 元素

HTML <span> 元素是内联元素，可用作文本的容器。

<span> 元素也没有特定的含义。

当与 CSS 一同使用时，<span> 元素可用于为部分文本设置样式属性。

## 18.5 HTML 分组标签

|  |  |
| --- | --- |
| 标签 | 描述 |
| <div> | 定义文档中的分区或节（division/section）。 |
| <span> | 定义 span，用来组合文档中的行内元素。 |

# 第19章 HTML 类

对 HTML 进行分类（设置类），使我们能够为元素的类定义 CSS 样式。

为相同的类设置相同的样式，或者为不同的类设置不同的样式。

## 19.1 实例

<!DOCTYPE html>

<html>

<head>

<style>

.cities {

background-color:black;

color:white;

margin:20px;

padding:20px;

}

</style>

</head>

<body>

<div class="cities">

<h2>London</h2>

<p>London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.</p>

<p>Standing on the River Thames, London has been a major settlement for two millennia, its history going back to its founding by the Romans, who named it Londinium.</p>

</div>

</body>

</html>

## 19.2 分类块级元素

HTML <div> 元素是块级元素。它能够用作其他 HTML 元素的容器。

设置 <div> 元素的类，使我们能够为相同的 <div> 元素设置相同的类：

**实例：**

<!DOCTYPE html>

<html>

<head>

<style>

.cities {

background-color:black;

color:white;

margin:20px;

padding:20px;

}

</style>

</head>

<body>

<div class="cities">

<h2>London</h2>

<p>London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.</p>

<p>Standing on the River Thames, London has been a major settlement for two millennia, its history going back to its founding by the Romans, who named it Londinium.</p>

</div>

<div class="cities">

<h2>Paris</h2>

<p>Paris is the capital and most populous city of France.</p>

<p>Situated on the Seine River, it is at the heart of the 蝜e-de-France region, also known as the r間ion parisienne.</p>

<p>Within its metropolitan area is one of the largest population centers in Europe, with over 12 million inhabitants.</p>

</div>

<div class="cities">

<h2>Tokyo</h2>

<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area, and the most populous metropolitan area in the world.</p>

<p>It is the seat of the Japanese government and the Imperial Palace, and the home of the Japanese Imperial Family.</p>

<p>The Tokyo prefecture is part of the world's most populous metropolitan area with 38 million people and the world's largest urban economy.</p>

</div>

</body>

</html>

## 19.3 分类行内元素

HTML <span> 元素是行内元素，能够用作文本的容器。

设置 <span> 元素的类，能够为相同的 <span> 元素设置相同的样式。

**实例：**

<!DOCTYPE html>

<html>

<head>

<style>

span.red {

color:red;

}

</style>

</head>

<body>

<h1>我的<span class="red">重要的</span>标题</h1>

</body>

</html>

# 第20章 HTML id 属性

HTML id 属性用于 为HTML 元素指定唯一的 id。

一个 HTML文档中不能存在多个有相同 id 的元素。

## 20.1 使用 id 属性

id 属性指定 HTML 元素的唯一 ID。 id 属性的值在 HTML 文档中必须是唯一的。

id 属性用于指向样式表中的特定样式声明。JavaScript 也可使用它来访问和操作拥有特定 ID 的元素。

id 的语法是：写一个井号 (#)，后跟一个 id 名称。然后，在花括号 {} 中定义 CSS 属性。

下面的例子中我们有一个 <h1> 元素，它指向 id 名称 "myHeader"。这个 <h1> 元素将根据 head 部分中的 #myHeader 样式定义进行样式设置：

<!DOCTYPE html>

<html>

<head>

<style>

#myHeader {

background-color: lightblue;

color: black;

padding: 40px;

text-align: center;

}

</style>

</head>

<body>

<h1>id 属性</h1>

<p>使用 CSS 来设置 id 为 "myHeader" 的元素的样式：</p>

<h1 id="myHeader">My Header</h1>

</body>

</html>

注释：id 名称对大小写敏感！

注释：id 必须包含至少一个字符，且不能包含空白字符（空格、制表符等）。

## 20.2 Class 与 ID 的差异

同一个类名可以由多个 HTML 元素使用，而一个 id 名称只能由页面中的一个 HTML 元素使用：

<!DOCTYPE html>

<html>

<head>

<style>

/\* 设置 id 为 "myHeader" 的元素的样式 \*/

#myHeader {

background-color: lightblue;

color: black;

padding: 40px;

text-align: center;

}

/\* 设置类名为 "city" 的所有元素的样式 \*/

.city {

background-color: tomato;

color: white;

padding: 10px;

}

</style>

</head>

<body>

<h1>Class 与 ID 的差异</h1>

<p>一个类名可以由多个 HTML 元素使用，而一个 id 名称只能由页面中的一个 HTML 元素使用：</p>

<!-- 拥有唯一 id 的元素 -->

<h1 id="myHeader">我的城市</h1>

<!-- 拥有相同类名的多个元素 -->

<h2 class="city">上海</h2>

<p>中国最大的经济城市。</p>

<h2 class="city">深圳</h2>

<p>中国最具创新力的城市。</p>

<h2 class="city">北京</h2>

<p>中国的首都。</p>

</body>

</html>

## 20.3 通过 ID 和链接实现 HTML 书签

HTML 书签用于让读者跳转至网页的特定部分。

如果页面很长，那么书签可能很有用。

要使用书签，您必须首先创建它，然后为它添加链接。

然后，当单击链接时，页面将滚动到带有书签的位置。

实例

首先，用 id 属性创建书签：

<h2 id="C4">第四章</h2>

然后，在同一张页面中，向这个书签添加一个链接（“跳转到第四章”）：

**实例：**

<!DOCTYPE html>

<html>

<body>

<p><a href="#C4">跳转到 Chapter 4</a></p>

<p><a href="#C10">跳转到 Chapter 10</a></p>

<h2>Chapter 1</h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 2</h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 3</h2>

<p>This chapter explains ba bla bla</p>

<h2 id="C4">Chapter 4</h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 5</h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 6</h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 7</h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 8</h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 9</h2>

<p>This chapter explains ba bla bla</p>

<h2 id="C10">Chapter 10</h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 11</h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 12</h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 13</h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 14</h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 15</h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 16</h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 17</h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 18</h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 19</h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 20</h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 21</h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 22</h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 23</h2>

<p>This chapter explains ba bla bla</p>

</body>

</html>

或者，在另一张页面中，添加指向这个书签的链接（“跳转到第四章”）：

<a href="html\_demo.html#C4">Jump to Chapter 4</a>

## 20.4 在 JavaScript 中使用 id 属性

JavaScript 也可以使用 id 属性为特定元素执行某些任务。

JavaScript 可以使用 getElementById() 方法访问拥有特定 id 的元素：

**实例：**

<!DOCTYPE html>

<html>

<body>

<h1>在 JavaScript 中使用 id 属性</h1>

<p>JavaScript 可以使用 getElementById() 方法访问具有指定 ID 的元素：</p>

<h2 id="myHeader">Hello World!</h2>

<button onclick="displayResult()">改变文本</button>

<script>

function displayResult() {

document.getElementById("myHeader").innerHTML = "Have a nice day!";

}

</script>

</body>

</html>

## 20.5本章总结

id 属性用于为 HTML 元素指定唯一的 id

id 属性的值在 HTML 文档中必须是唯一的

CSS 和 JavaScript 可使用 id 属性来选取元素或设置特定元素的样式

id 属性的值区分大小写

id 属性还可用于创建 HTML 书签

JavaScript 可以使用 getElementById() 方法访问拥有特定 id 的元素

# 第21章 HTML Iframe

iframe 用于在网页内显示网页。

## 21.1 添加 iframe 的语法

<iframe src="URL"></iframe>

URL 指向隔离页面的位置。

## 21.2 Iframe - 设置高度和宽度

height 和 width 属性用于规定 iframe 的高度和宽度。

属性值的默认单位是像素，但也可以用百分比来设定（比如 "80%"）。

<!DOCTYPE html>

<html>

<body>

<iframe src="/example/html/demo\_iframe.html" width="200" height="200"></iframe>

<p>某些老式的浏览器不支持内联框架。</p>

<p>如果不支持，则 iframe 是不可见的。</p>

</body>

</html>

## 21.3 Iframe - 删除边框

frameborder 属性规定是否显示 iframe 周围的边框。

设置属性值为 "0" 就可以移除边框：

<!DOCTYPE html>

<html>

<body>

<iframe src="/example/html/demo\_iframe.html" frameborder="0"></iframe>

<p>某些老式的浏览器不支持内联框架。</p>

<p>如果不支持，则 iframe 是不可见的。</p>

</body>

</html>

## 21.4 使用 iframe 作为链接的目标

iframe 可用作链接的目标（target）。

链接的 target 属性必须引用 iframe 的 name 属性：

<!DOCTYPE html>

<html>

<body>

<iframe src="/example/html/demo\_iframe.html" name="iframe\_a"></iframe>

<p><a href="http://www.w3school.com.cn" target="iframe\_a">W3School.com.cn</a></p>

<p><b>注释：</b>由于链接的目标匹配 iframe 的名称，所以链接会在 iframe 中打开。</p>

</body>

</html>

## 21.5 HTML iframe 标签

|  |  |
| --- | --- |
| 标签 | 描述 |
| <iframe> | 定义内联的子窗口（框架） |

# 第22章 HTML JavaScript

JavaScript 使 HTML 页面更具动态性和交互性

## 22.1 HTML <script> 标签

HTML <script> 标签用于定义客户端脚本（JavaScript）。

<script> 元素即可包含脚本语句，也可通过 src 属性指向外部脚本文件。

JavaScript 的常见用途是图像处理、表单验证和内容的动态更改。

如需选取 HTML 元素，JavaScript 最常用 document.getElementById() 方法。

这个 JavaScript 示例向 id="demo" 的 HTML 元素内写入 "Hello JavaScript!"：

<!DOCTYPE html>

<html>

<body>

<h1>使用 JavaScript 更改文本</h1>

<p>本例把 "Hello JavaScript!" 写入 id="demo" 的 HTML 元素内：</p>

<p id="demo"></p>

<script>

document.getElementById("demo").innerHTML = "Hello JavaScript!";

</script>

</body>

</html>

## 22.2 JavaScript 的味道

JavaScript 能够更改内容：

document.getElementById("demo").innerHTML = "Hello JavaScript!";

JavaScript 能够更改样式：

document.getElementById("demo").style.fontSize = "25px";

document.getElementById("demo").style.color = "red";

document.getElementById("demo").style.backgroundColor = "yellow";

JavaScript 能够更改属性：

document.getElementById("image").src = "picture.gif";

## 22.3 HTML <noscript> 标签

HTML <noscript> 标签定义了替代内容，这些内容将显示给在浏览器中禁用了脚本或浏览器不支持脚本的用户：

<script>

document.getElementById("demo").innerHTML = "Hello JavaScript!";

</script>

<noscript>抱歉，您的浏览器不支持 JavaScript！</noscript>

## 22.4 HTML Script 标签

|  |  |
| --- | --- |
| 标签 | 描述 |
| <script> | 定义客户端脚本。 |
| <noscript> | 为不支持客户端脚本的用户定义替代内容。 |

# 第23章 HTML 文件路径

|  |  |
| --- | --- |
| 路径 | 描述 |
| <img src="picture.jpg"> | picture.jpg 位于与当前网页相同的文件夹 |
| <img src="images/picture.jpg"> | picture.jpg 位于当前文件夹的 images 文件夹中 |
| <img src="/images/picture.jpg"> | picture.jpg 当前站点根目录的 images 文件夹中 |
| <img src="../picture.jpg"> | picture.jpg 位于当前文件夹的上一级文件夹中 |

## 23.1 HTML 文件路径

文件路径描述了网站文件夹结构中某个文件的位置。

文件路径会在链接外部文件时被用到：

网页

图像

样式表

JavaScript

## 23.2 绝对文件路径

绝对文件路径是指向一个因特网文件的完整 URL：

实例

<img src="https://www.w3school.com.cn/images/picture.jpg" alt="flower">

<img> 标签以及 src 和 alt 属性在 HTML 图像这一章做了讲解。

## 23.3 相对路径

相对路径指向了相对于当前页面的文件。

在本例中，文件路径指向了位于当前网站根目录中 images 文件夹里的一个文件：

实例

<img src="/images/picture.jpg" alt="flower">

在本例中，文件路径指向了位于当前文件夹中 images 文件夹里的一个文件：

实例

<img src="images/picture.jpg" alt="flower">

在本例中，文件路径指向了位于当前文件夹的上一级文件夹中 images 文件夹里的一个文件：

实例

<img src="../images/picture.jpg" alt="flower">

## 23.4 好习惯

使用相对路径是个好习惯（如果可能）。

如果使用了相对路径，那么您的网页就不会与当前的基准 URL 进行绑定。所有链接在您的电脑上 (localhost) 或未来的公共域中均可正常工作。

# 第24章 HTML 头部元素

## 24.1 实例

**文档的标题**

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />

<meta http-equiv="Content-Language" content="zh-cn" />

<title>标题不会显示在文档区</title>

</head>

<body>

<p>这段文本会显示出来。</p>

</body>

</html>

**所有链接一个目标**

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />

<meta http-equiv="Content-Language" content="zh-cn" />

<base target="\_blank" />

</head>

<body>

<p>

<a href="http://www.w3school.com.cn" target="\_blank">这个连接</a> 将在新窗口中加载，因为 target 属性被设置为 "\_blank"。

</p>

<p>

<a href="http://www.w3school.com.cn">这个连接</a> 也将在新窗口中加载，即使没有 target 属性。

</p>

</body>

</html>

**文档描述**

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />

<meta name="author"

content="w3school.com.cn">

<meta name="revised"

content="David Yang,8/1/07">

<meta name="generator"

content="Dreamweaver 8.0en">

</head>

<body>

<p>本文档的 meta 属性标识了创作者和编辑软件。</p>

</body>

</html>

**文档关键词**

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />

<meta name="description"

content="HTML examples">

<meta name="keywords"

content="HTML, DHTML, CSS, XML, XHTML, JavaScript, VBScript">

</head>

<body>

<p>本文档的 meta 属性描述了该文档和它的关键词。</p>

</body>

</html>

**重定向用户**

<!DOCTYPE HTML>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />

<meta http-equiv="Refresh" content="5;url=http://www.w3school.com.cn" />

</head>

<body>

<p>

对不起。我们已经搬家了。您的 URL 是 <a href="http://www.w3school.com.cn">http://www.w3school.com.cn</a>

</p>

<p>您将在 5 秒内被重定向到新的地址。</p>

<p>如果超过 5 秒后您仍然看到本消息，请点击上面的链接。</p>

</body>

</html>

## 24.2 HTML <head> 元素

<head> 元素是所有头部元素的容器。<head> 内的元素可包含脚本，指示浏览器在何处可以找到样式表，提供元信息，等等。

以下标签都可以添加到 head 部分：<title>、<base>、<link>、<meta>、<script> 以及 <style>。

## 24.3 HTML <title> 元素

<title> 标签定义文档的标题。

title 元素在所有 HTML/XHTML 文档中都是必需的。

title 元素能够：

定义浏览器工具栏中的标题

提供页面被添加到收藏夹时显示的标题

显示在搜索引擎结果中的页面标题

一个简化的 HTML 文档：

<!DOCTYPE html>

<html>

<head>

<title>Title of the document</title>

</head>

<body>

The content of the document......

</body>

</html>

## 24.4 HTML <base> 元素

<base> 标签为页面上的所有链接规定默认地址或默认目标（target）：

<head>

<base href="http://www.w3school.com.cn/images/" />

<base target="\_blank" />

</head>

## 24.5 HTML <link> 元素

<link> 标签定义文档与外部资源之间的关系。

<link> 标签最常用于连接样式表：

<head>

<link rel="stylesheet" type="text/css" href="mystyle.css" />

</head>

## 24.6 HTML <style> 元素

<style> 标签用于为 HTML 文档定义样式信息。

您可以在 style 元素内规定 HTML 元素在浏览器中呈现的样式：

<head>

<style type="text/css">

body {background-color:yellow}

p {color:blue}

</style>

</head>

## 24.7 HTML <meta> 元素

元数据（metadata）是关于数据的信息。

<meta> 标签提供关于 HTML 文档的元数据。元数据不会显示在页面上，但是对于机器是可读的。

典型的情况是，meta 元素被用于规定页面的描述、关键词、文档的作者、最后修改时间以及其他元数据。

<meta> 标签始终位于 head 元素中。

元数据可用于浏览器（如何显示内容或重新加载页面），搜索引擎（关键词），或其他 web 服务。

针对搜索引擎的关键词

一些搜索引擎会利用 meta 元素的 name 和 content 属性来索引您的页面。

下面的 meta 元素定义页面的描述：

<meta name="description" content="Free Web tutorials on HTML, CSS, XML" />

下面的 meta 元素定义页面的关键词：

<meta name="keywords" content="HTML, CSS, XML" />

name 和 content 属性的作用是描述页面的内容。

## 24.8 HTML <script> 元素

<script> 标签用于定义客户端脚本，比如 JavaScript。

我们会在稍后的章节讲解 script 元素。

## 24.9 HTML 头部元素

|  |  |
| --- | --- |
| 标签 | 描述 |
| <head> | 定义关于文档的信息。 |
| <title> | 定义文档标题。 |
| <base> | 定义页面上所有链接的默认地址或默认目标。 |
| <link> | 定义文档与外部资源之间的关系。 |
| <meta> | 定义关于 HTML 文档的元数据。 |
| <script> | 定义客户端脚本。 |
| <style> | 定义文档的样式信息。 |

# 第25章 HTML 布局

## 25.1 使用 <div> 元素的 HTML 布局

注释：<div> 元素常用作布局工具，因为能够轻松地通过 CSS 对其进行定位。

这个例子使用了四个 <div> 元素来创建多列布局：

**实例**

<!DOCTYPE html>

<html>

<head>

<style>

#header {

background-color:black;

color:white;

text-align:center;

padding:5px;

}

#nav {

line-height:30px;

background-color:#eeeeee;

height:300px;

width:100px;

float:left;

padding:5px;

}

#section {

width:350px;

float:left;

padding:10px;

}

#footer {

background-color:black;

color:white;

clear:both;

text-align:center;

padding:5px;

}

</style>

</head>

<body>

<div id="header">

<h1>City Gallery</h1>

</div>

<div id="nav">

London<br>

Paris<br>

Tokyo<br>

</div>

<div id="section">

<h2>London</h2>

<p>

London is the capital city of England. It is the most populous city in the United Kingdom,

with a metropolitan area of over 13 million inhabitants.

</p>

<p>

Standing on the River Thames, London has been a major settlement for two millennia,

its history going back to its founding by the Romans, who named it Londinium.

</p>

</div>

<div id="footer">

Copyright ? W3Schools.com

</div>

</body>

</html>

**CSS：**

<style>

#header {

background-color:black;

color:white;

text-align:center;

padding:5px;

}

#nav {

line-height:30px;

background-color:#eeeeee;

height:300px;

width:100px;

float:left;

padding:5px;

}

#section {

width:350px;

float:left;

padding:10px;

}

#footer {

background-color:black;

color:white;

clear:both;

text-align:center;

padding:5px;

}

</style>

## 25.2 使用 HTML5 的网站布局

HTML5 提供的新语义元素定义了网页的不同部分：

### 25.2.1 HTML5 语义元素

header 定义文档或节的页眉

nav 定义导航链接的容器

section 定义文档中的节

article 定义独立的自包含文章

aside 定义内容之外的内容（比如侧栏）

footer 定义文档或节的页脚

details 定义额外的细节

summary 定义 details 元素的标题

这个例子使用 <header>, <nav>, <section>, 以及 <footer> 来创建多列布局：

<!DOCTYPE html>

<html>

<head>

<style>

header {

background-color:black;

color:white;

text-align:center;

padding:5px;

}

nav {

line-height:30px;

background-color:#eeeeee;

height:300px;

width:100px;

float:left;

padding:5px;

}

section {

width:350px;

float:left;

padding:10px;

}

footer {

background-color:black;

color:white;

clear:both;

text-align:center;

padding:5px;

}

</style>

</head>

<body>

<header>

<h1>City Gallery</h1>

</header>

<nav>

London<br>

Paris<br>

Tokyo<br>

</nav>

<section>

<h1>London</h1>

<p>

London is the capital city of England. It is the most populous city in the United Kingdom,

with a metropolitan area of over 13 million inhabitants.

</p>

<p>

Standing on the River Thames, London has been a major settlement for two millennia,

its history going back to its founding by the Romans, who named it Londinium.

</p>

</section>

<footer>

Copyright W3Schools.com

</footer>

</body>

</html>

CSS：

header {

background-color:black;

color:white;

text-align:center;

padding:5px;

}

nav {

line-height:30px;

background-color:#eeeeee;

height:300px;

width:100px;

float:left;

padding:5px;

}

section {

width:350px;

float:left;

padding:10px;

}

footer {

background-color:black;

color:white;

clear:both;

text-align:center;

padding:5px;

}

## 25.3 使用表格的 HTML 布局

注释：<table> 元素不是作为布局工具而设计的。

<table> 元素的作用是显示表格化的数据。

使用 <table> 元素能够取得布局效果，因为能够通过 CSS 设置表格元素的样式：

**实例**

<!DOCTYPE html>

<html>

<head>

<style>

table.lamp {

width:100%;

border:1px solid #d4d4d4;

}

table.lamp th, td {

padding:10px;

}

table.lamp th {

width:40px;

}

</style>

</head>

<body>

<table class="lamp">

<tr>

<th>

<img src="/images/lamp.jpg" alt="Note" style="height:32px;width:32px">

</th>

<td>

The table element was not designed to be a layout tool.

</td>

</tr>

</table>

</body>

</html>

CSS：

<style>

table.lamp {

width:100%;

border:1px solid #d4d4d4;

}

table.lamp th, td {

padding:10px;

}

table.lamp td {

width:40px;

}

</style>

# 第26章 HTML 响应式 Web 设计

## 26.1 什么是响应式 Web 设计？

RWD 指的是响应式 Web 设计（Responsive Web Design）

RWD 能够以可变尺寸传递网页

RWD 对于平板和移动设备是必需的

## 26.2 创建您自己的响应式设计

<!DOCTYPE html>

<html lang="en-US">

<head>

<style>

.city {

float: left;

margin: 5px;

padding: 15px;

width: 300px;

height: 300px;

border: 1px solid black;

}

</style>

</head>

<body>

<h1>W3School Demo</h1>

<h2>Resize this responsive page!</h2>

<br>

<div class="city">

<h2>London</h2>

<p>London is the capital city of England.</p>

<p>It is the most populous city in the United Kingdom,

with a metropolitan area of over 13 million inhabitants.</p>

</div>

<div class="city">

<h2>Paris</h2>

<p>Paris is the capital and most populous city of France.</p>

</div>

<div class="city">

<h2>Tokyo</h2>

<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,

and the most populous metropolitan area in the world.</p>

</div>

</body>

</html>

# 第27章 HTML 计算机代码元素

## 27.1 计算机代码

var person = {

firstName:"Bill",

lastName:"Gates",

age:50,

eyeColor:"blue"

}

## 27.2 HTML 计算机代码格式

通常，HTML 使用可变的字母尺寸，以及可变的字母间距。

在显示计算机代码示例时，并不需要如此。

<kbd>, <samp>, 以及 <code> 元素全都支持固定的字母尺寸和间距。

## 27.3 HTML 键盘格式

HTML <kbd> 元素定义键盘输入：

<!DOCTYPE html>

<html>

<body style="font-size:16px">

<p>HTML kbd 元素表示键盘输入：</p>

<p><kbd>File | Open...</kbd></p>

</body>

</html>

## 27.4 HTML 样本格式

HTML <samp> 元素定义计算机输出示例：

<!DOCTYPE html>

<html>

<body style="font-size:16px">

<p>HTML samp 元素表示计算机输出示例：</p>

<samp>

demo.example.com login: Apr 12 09:10:17

Linux 2.6.10-grsec+gg3+e+fhs6b+nfs+gr0501+++p3+c4a+gr2b-reslog-v6.189

</samp>

</body>

</html>

## 27.5 HTML 代码格式

HTML <code> 元素定义编程代码示例：

<!DOCTYPE html>

<html>

<body style="font-size:16px">

<p>编程代码示例：</p>

<code>

var person = {firstName:"Bill", lastName:"Gats", age:50}

</code>

</body>

</html>

<code> 元素不保留多余的空格和折行：

<!DOCTYPE html>

<html>

<body style="font-size:16px">

<p>code 元素不保留多余的空格和折行：</p>

<code>

var person = {

firstName:"Bill",

lastName:"Gates",

age:50,

eyeColor:"blue"

}

</code>

</body>

</html>

如需解决该问题，您必须在 <pre> 元素中包围代码：

<!DOCTYPE html>

<html>

<body style="font-size:16px">

<p>code 元素不保留多余的空格和折行：</p>

<p>如需解决该问题，您必须在 pre 元素中包围代码：</p>

<code>

<pre>

var person = {

firstName:"Bill",

lastName:"Gates",

age:50,

eyeColor:"blue"

}

</pre>

</code>

</body>

</html>

## 27.6 HTML 变量格式化

HTML <var> 元素定义数学变量：

<!DOCTYPE html>

<html>

<body style="font-size:16px">

<p>爱因斯坦的公式：</p>

<p><var>E</var> = <var>m</var> <var>c</var><sup>2</sup></p>

</body>

</html>

## 27.7 HTML 计算机代码元素

|  |  |
| --- | --- |
| 标签 | 描述 |
| <code> | 定义计算机代码文本 |
| <kbd> | 定义键盘文本 |
| <samp> | 定义计算机代码示例 |
| <var> | 定义变量 |
| <pre> | 定义预格式化文本 |

# 第28章 HTML5 语义元素

语义学（源自古希腊）可定义为对语言意义的研究。

语义元素是拥有语义的元素。

## 28.1 什么是语义元素？

语义元素清楚地向浏览器和开发者描述其意义。

非语义元素的例子：<div> 和 <span> - 无法提供关于其内容的信息。

语义元素的例子：<form>、<table> 以及 <img> - 清晰地定义其内容。

所有现代浏览器均支持 HTML5 语义元素。

此外，您可以“帮助”老式浏览器处理“未知元素”。

在 HTML5 浏览器支持这一章学习更多知识。

## 28.2 HTML5 中新的语义元素

许多网站包含了指示导航、页眉以及页脚的 HTML 代码，例如这些：<div id="nav"> <div class="header"> <div id="footer">。

HTML5 提供了定义页面不同部分的新语义元素：

<article>

<aside>

<details>

<figcaption>

<figure>

<footer>

<header>

<main>

<mark>

<nav>

<section>

<summary>

<time>

## 28.3 HTML5 <section> 元素

<section> 元素定义文档中的节。

根据 W3C 的 HTML 文献：“节（section）是有主题的内容组，通常具有标题”。

可以将网站首页划分为简介、内容、联系信息等节。

<section>

<h1>WWF</h1>

<p>The World Wide Fund for Nature (WWF) is....</p>

</section>

## 28.4 HTML5 <article> 元素

<article> 元素规定独立的自包含内容。

文档有其自身的意义，并且可以独立于网站其他内容进行阅读。

<article> 元素的应用场景：

论坛

博客

新闻

**实例**

<article>

<h1>What Does WWF Do?</h1>

<p>WWF's mission is to stop the degradation of our planet's natural environment,

and build a future in which humans live in harmony with nature.</p>

</article>

## 28.5 嵌套语义元素

在 HTML5 标准中，<article> 元素定义完整的相关元素自包含块。

<section> 元素被定义为相关元素块。

我们能够使用该定义来决定如何嵌套元素吗？不，我们不能！

在因特网上，您会发现 <section> 元素包含 <article> 元素的 HTML 页面，还有 <article> 元素包含 <sections> 元素的页面。

您还会发现 <section> 元素包含 <section> 元素，同时 <article> 元素包含 <article> 元素。

HTML5 <header> 元素

<header> 元素为文档或节规定页眉。

<header> 元素应该被用作介绍性内容的容器。

一个文档中可以有多个 <header> 元素。

下例为一篇文章定义了页眉：

实例

<article>

<header>

<h1>What Does WWF Do?</h1>

<p>WWF's mission:</p>

</header>

<p>WWF's mission is to stop the degradation of our planet's natural environment,

and build a future in which humans live in harmony with nature.</p>

</article>

## 28.6 HTML5 <footer> 元素

<footer> 元素为文档或节规定页脚。

<footer> 元素应该提供有关其包含元素的信息。

页脚通常包含文档的作者、版权信息、使用条款链接、联系信息等等。

您可以在一个文档中使用多个 <footer> 元素。

实例

<footer>

<p>Posted by: Hege Refsnes</p>

<p>Contact information: <a href="mailto:someone@example.com">

someone@example.com</a>.</p>

</footer>

## 28.7 HTML5 <nav> 元素

<nav> 元素定义导航链接集合。

<nav> 元素旨在定义大型的导航链接块。不过，并非文档中所有链接都应该位于 <nav> 元素中！

实例

<nav>

<a href="/html/">HTML</a> |

<a href="/css/">CSS</a> |

<a href="/js/">JavaScript</a> |

<a href="/jquery/">jQuery</a>

</nav>

## 28.8 HTML5 <aside> 元素

<aside> 元素页面主内容之外的某些内容（比如侧栏）。

aside 内容应该与周围内容相关。

实例

<p>My family and I visited The Epcot center this summer.</p>

<aside>

<h4>Epcot Center</h4>

<p>The Epcot Center is a theme park in Disney World, Florida.</p>

</aside>

## 28.9 HTML5 <figure> 和 <figcaption> 元素

在书籍和报纸中，与图片搭配的标题很常见。

标题（caption）的作用是为图片添加可见的解释。

通过 HTML5，图片和标题能够被组合在 <figure> 元素中：

实例

<figure>

<img src="pic\_mountain.jpg" alt="The Pulpit Rock" width="304" height="228">

<figcaption>Fig1. - The Pulpit Pock, Norway.</figcaption>

</figure>

<img> 元素定义图像，<figcaption> 元素定义标题。

## 28.10 为何使用 HTML5 元素？

如果使用 HTML4 的话，开发者会使用他们喜爱的属性名来设置页面元素的样式：

header, top, bottom, footer, menu, navigation, main, container, content, article, sidebar, topnav, ...

如此，浏览器便无法识别正确的网页内容。

而通过 HTML5 元素，比如：<header> <footer> <nav> <section> <article>，此问题迎刃而解。

根据 W3C，语义网：

“允许跨应用程序、企业和团体对数据进行分享和重用。”

## 28.11 HTML5 中的语义元素

|  |  |
| --- | --- |
| 标签 | 描述 |
| <article> | 定义文章。 |
| <aside> | 定义页面内容以外的内容。 |
| <details> | 定义用户能够查看或隐藏的额外细节。 |
| <figcaption> | 定义 <figure> 元素的标题。 |
| <figure> | 规定自包含内容，比如图示、图表、照片、代码清单等。 |
| <footer> | 定义文档或节的页脚。 |
| <header> | 规定文档或节的页眉。 |
| <main> | 规定文档的主内容。 |
| <mark> | 定义重要的或强调的文本。 |
| <nav> | 定义导航链接。 |
| <section> | 定义文档中的节。 |
| <summary> | 定义 <details> 元素的可见标题。 |
| <time> | 定义日期/时间。 |

# 第29章 HTML(5) 样式指南和代码约定

HTML 代码约定

web 开发者常常不确定在 HTML 中使用的代码样式和语法。

在 2000 年至 2010 年之间，许多 web 开发者从 HTML 转换为 XHTML。

通过 XHTML，开发者不得不编写有效的“格式良好的”代码。

HTML5 在代码验证时会更宽松一点。

通过 HTML5，您必须创建属于自己的最佳实践、样式指南和代码约定。

智能且有未来保证

对样式的合乎逻辑的使用，可以令其他人更容易理解和使用您的 HTML。

在未来，诸如 XML 阅读器之类的程序，也许需要阅读您的 HTML。

使用格式良好的“近似 XHTML 的”语法，能够更智能。

注释：请始终保持您的样式智能、整洁、纯净、格式良好。

请使用正确的文档类型

请始终在文档的首行声明文档类型：

<!DOCTYPE html>

如果您一贯坚持小写标签，那么可以使用：

<!doctype html>

请使用小写元素名

HTML5 允许在元素名中使用混合大小写字母。

我们推荐使用小写元素名：

混合大小写名称并不好

开发者习惯使用小写名（比如在 XHTML 中）

小写更起来更纯净

小写更易书写

不太好：

<SECTION>

<p>This is a paragraph.</p>

</SECTION>

很糟糕：

<Section>

<p>This is a paragraph.</p>

</SECTION>

还不错：

<section>

<p>This is a paragraph.</p>

</section>

关闭所有 HTML 元素

在 HTML5 中，您不必关闭所有元素（例如 <p> 元素）。

我们建议关闭所有 HTML 元素：

看起来不好：

<section>

<p>This is a paragraph.

<p>This is a paragraph.

</section>

看起来不错：

<section>

<p>This is a paragraph.</p>

<p>This is a paragraph.</p>

</section>

关闭空的 HTML 元素

在 HTML5 中，关闭空元素是可选的。

允许这样：

<meta charset="utf-8">

也允许这样：

<meta charset="utf-8" />

斜杠（/）在 XHTML 和 XML 中是必需的。

如果您期望 XML 软件来访问您的页面，保持这个习惯是个好主意。

使用小写属性名

HTML5 允许大小写混合的属性名。

我们建议使用小写属性名：

混合属性名并不好

开发者习惯于使用小写属性名（比如在 XHTML 中）

小写属性名看情况更纯净

小写属性名更易书写

看起来不好：

<div CLASS="menu">

看起来不错：

<div class="menu">

属性值加引号

HTML5 允许不加引号的属性值。

我们推荐属性值加引号：

如果属性值包含值，则必须使用引号

混合样式绝对不好

加引号的值更易阅读

这个属性值无效，因为值中包含空格：

<table class=table striped>

这样是有效的：

<table class="table striped">

必需的属性

请始终对图像使用 alt 属性。当图像无法显示时该属性很重要。

<img src="html5.gif" alt="HTML5" style="width:128px;height:128px">

请始终定义图像尺寸。这样做会减少闪烁，因为浏览器会在图像加载之前为图像预留空间。

<img src="html5.gif" alt="HTML5" style="width:128px;height:128px">

空格和等号

等号两边的空格是合法的：

<link rel = "stylesheet" href = "styles.css">

但是精简空格更易阅读， But space-less is easier to read, and groups entities better together:

<link rel="stylesheet" href="styles.css">

避免长代码行

当使用 HTML 编辑器时，通过左右滚动来阅读 HTML 代码很不方便。

请尽量避免代码行超过 80 个字符。

空行和缩进

请勿毫无理由地增加空行。

为了提高可读性，请增加空行来分隔大型或逻辑代码块。

为了提高可读性，请增加两个空格的缩进。请勿使用 TAB。

请勿使用没有必要的空行和缩进。没有必要在短的和相关项目之间使用空行，也没有必要缩进每个元素：

不必要：

<body>

<h1>Famous Cities</1>

<h2>Tokyo</h2>

<p>

Tokyo is the capital of Japan, the center of the Greater Tokyo Area,

and the most populous metropolitan area in the world.

It is the seat of the Japanese government and the Imperial Palace,

and the home of the Japanese Imperial Family.

</p>

</body>

更好：

<body>

<h1>Famous Cities</1>

<h2>Tokyo</h2>

<p>

Tokyo is the capital of Japan, the center of the Greater Tokyo Area,

and the most populous metropolitan area in the world.

It is the seat of the Japanese government and the Imperial Palace,

and the home of the Japanese Imperial Family.

</p>

</body>

表格示例：

<table>

<tr>

<th>Name</th>

<th>Description</th>

<tr>

<tr>

<td>A</td>

<td>Description of A</td>

<tr>

<tr>

<td>B</td>

<td>Description of B</td>

<tr>

</table>

列表示例：

<ol>

<li>LondonA</li>

<li>Paris</li>

<li>Tokyo</li>

</ol>

省略 <html> 和 <body>？

在 HTML5 标准中，能够省略 <html> 标签和 <body> 标签。

以下代码作为 HTML5 进行验证：

示例

<!DOCTYPE html>

<head>

<title>Page Title</title>

</head>

<h1>This is a heading</h1>

<p>This is a paragraph.</p>

亲自试一试

我们不推荐省略 <html> 和 <body> 标签。

<html> 元素是文本的根元素。它是规定页面语言的理想位置。

<!DOCTYPE html>

<html lang="en-US">

对于可访问应用程序（屏幕阅读器）和搜索引擎，声明语言很重要。

省略 <html> 或 <body> c可令 DOM 和 XML 软件崩溃。

省略 <body> 会在老式浏览器（IE9）中产生错误。

省略 <head>？

在 HTML5 标准中，<head> 标签也能够被省略。

默认地，浏览器会把 <body> 之前的所有元素添加到默认的 <head> 元素。

通过省略 <head> 标签，您能够降低 HTML 的复杂性：

示例

<!DOCTYPE html>

<html>

<title>Page Title</title>

<body>

<h1>This is a heading</h1>

<p>This is a paragraph.</p>

</body>

</html>

亲自试一试

注释：对于 web 开发者，省略标签的做法是陌生的。建立规则需要时间。

元数据

<title> 元素在 HTML5 中是必需的。请尽可能制作有意义的标题。

<title>HTML5 Syntax and Coding Style</title>

为了确保恰当的解释，以及正确的搜索引擎索引，在文档中对语言和字符编码的定义越早越好：

<!DOCTYPE html>

<html lang="en-US">

<head>

<meta charset="UTF-8">

<title>HTML5 Syntax and Coding Style</title>

</head>

HTML 注释

短注释应该在单行中书写，并在 <!-- 之后增加一个空格，在 --> 之前增加一个空格：

<!-- This is a comment -->

长注释，跨越多行，应该通过 <!-- 和 --> 在独立的行中书写：

<!--

This is a long comment example. This is a long comment example. This is a long comment example.

This is a long comment example. This is a long comment example. This is a long comment example.

-->

长注释更易观察，如果它们被缩进两个空格的话。

样式表

请使用简单的语法来链接样式表（type 属性不是必需的）：

<link rel="stylesheet" href="styles.css">

短规则可以压缩为一行，就像这样：

p.into {font-family:"Verdana"; font-size:16em;}

长规则应该分为多行：

body {

background-color: lightgrey;

font-family: "Arial Black", Helvetica, sans-serif;

font-size: 16em;

color: black;

}

开括号与选择器位于同一行

在开括号之前用一个空格

使用两个字符的缩进

在每个属性与其值之间使用冒号加一个空格

在每个逗号或分号之后使用空格

在每个属性值对（包括最后一个）之后使用分号

只在值包含空格时使用引号来包围值

把闭括号放在新的一行，之前不用空格

避免每行超过 80 个字符

注释：在逗号或分号之后添加空格，是所有书写类型的通用规则。

在 HTML 中加载 JavaScript

请使用简单的语法来加载外部脚本（type 属性不是必需的）：

<script src="myscript.js">

通过 JavaScript 访问 HTML 元素

使用“不整洁”的 HTML 样式的后果，是可能会导致 JavaScript 错误。

这两个 JavaScript 语句会产生不同的结果：

var obj = getElementById("Demo")

var obj = getElementById("demo")

亲自试一试

如果可能，请在 HTML 中使用（与 JavaScript）相同的命名约定。

请访问 JavaScript 样式指南。

使用小写文件名

大多数 web 服务器（Apache、Unix）对文件名的大小写敏感：

不能以 london.jpg 访问 London.jpg。

其他 web 服务器（微软，IIS）对大小写不敏感：

能够以 london.jpg 或 London.jpg 访问 London.jpg。

如果使用混合大小写，那么您必须保持高度的一致性。

如果您从对大小写不敏感的服务器转到一台对大小写敏感的服务器上，这些小错误将破坏您的网站。

为了避免这些问题，请始终使用小写文件名（如果可以的话）。

文件扩展名

HTML 文件名应该使用扩展名 .html（而不是 .htm）。

CSS 文件应该使用扩展名 .css。

JavaScript 文件应该使用扩展名 .js。

# 第30章 HTML 字符实体

HTML 中的预留字符必须被替换为字符实体。

## 30.1 HTML 实体

在 HTML 中，某些字符是预留的。

在 HTML 中不能使用小于号（<）和大于号（>），这是因为浏览器会误认为它们是标签。

如果希望正确地显示预留字符，我们必须在 HTML 源代码中使用字符实体（character entities）。

字符实体类似这样：

&entity\_name;

或者

&#entity\_number;

如需显示小于号，我们必须这样写：&lt; 或 &#60;

提示：使用实体名而不是数字的好处是，名称易于记忆。不过坏处是，浏览器也许并不支持所有实体名称（对实体数字的支持却很好）。

## 30.2 不间断空格（non-breaking space）

HTML 中的常用字符实体是不间断空格(&nbsp;)。

浏览器总是会截短 HTML 页面中的空格。如果您在文本中写 10 个空格，在显示该页面之前，浏览器会删除它们中的 9 个。如需在页面中增加空格的数量，您需要使用 &nbsp; 字符实体。

## 30.3 HTML 中有用的字符实体

注释：实体名称对大小写敏感！

|  |  |  |  |
| --- | --- | --- | --- |
| 显示结果 | 描述 | 实体名称 | 实体编号 |
|  | 空格 | &nbsp; | &#160; |
| < | 小于号 | &lt; | &#60; |
| > | 大于号 | &gt; | &#62; |
| & | 和号 | &amp; | &#38; |
| " | 引号 | &quot; | &#34; |
| ' | 撇号 | &apos; (IE不支持) | &#39; |
| ￠ | 分（cent） | &cent; | &#162; |
| £ | 镑（pound） | &pound; | &#163; |
| ¥ | 元（yen） | &yen; | &#165; |
| € | 欧元（euro） | &euro; | &#8364; |
| § | 小节 | &sect; | &#167; |
| © | 版权（copyright） | &copy; | &#169; |
| ® | 注册商标 | &reg; | &#174; |
| ™ | 商标 | &trade; | &#8482; |
| × | 乘号 | &times; | &#215; |
| ÷ | 除号 | &divide; | &#247; |

# 第31章 HTML 符号

键盘上不存在的字符也可以由实体代替

## 31.1 HTML 符号实体

在上一章中已经讲解了 HTML 实体。

普通键盘上不存在众多数学、技术和货币符号。

如需将此类符号添加到 HTML 页面，您可以使用 HTML 实体名称（HTML entity name）。

如果不存在实体名称，则可使用实体编号，十进制或十六进制的引用。

实例

<p>我将显示 &</p>

<p>我将显示 &#8364;</p>

<p>我将显示 &#x20AC;</p>

将显示为：

我将显示 €

我将显示 €

我将显示 €

## 31.2 HTML 支持的一些数学符号

|  |  |  |  |
| --- | --- | --- | --- |
| 字符 | 数字 | 实体 | 描述 |
| ∀ | &#8704; | &forall; | FOR ALL |
| ∂ | &#8706; | &part; | PARTIAL DIFFERENTIAL |
| ∃ | &#8707; | &exist; | THERE EXISTS |
| ∅ | &#8709; | &empty; | EMPTY SETS |
| ∇ | &#8711; | &nabla; | NABLA |
| ∈ | &#8712; | &isin; | ELEMENT OF |
| ∉ | &#8713; | &notin; | NOT AN ELEMENT OF |
| ∋ | &#8715; | &ni; | CONTAINS AS MEMBER |
| ∏ | &#8719; | &prod; | N-ARY PRODUCT |
| ∑ | &#8721; | &sum; | N-ARY SUMMATION |

## 31.3 HTML 支持的一些希腊字母

|  |  |  |  |
| --- | --- | --- | --- |
| 字符 | 数字 | 实体 | 描述 |
| Α | &#913; | &Alpha; | GREEK CAPITAL LETTER ALPHA |
| Β | &#914; | &Beta; | GREEK CAPITAL LETTER BETA |
| Γ | &#915; | &Gamma; | GREEK CAPITAL LETTER GAMMA |
| Δ | &#916; | &Delta; | GREEK CAPITAL LETTER DELTA |
| Ε | &#917; | &Epsilon; | GREEK CAPITAL LETTER EPSILON |
| Ζ | &#918; | &Zeta; | GREEK CAPITAL LETTER ZETA |

31.4 HTML 支持的一些其他实体。

|  |  |  |  |
| --- | --- | --- | --- |
| 字符 | 数字 | 实体 | 描述 |
| © | &#169; | &copy; | COPYRIGHT SIGN |
| ® | &#174; | &reg; | REGISTERED SIGN |
| € | &#8364; | &euro; | EURO SIGN |
| ™ | &#8482; | &trade; | TRADEMARK |
| ← | &#8592; | &larr; | LEFTWARDS ARROW |
| ↑ | &#8593; | &uarr; | UPWARDS ARROW |
| → | &#8594; | &rarr; | RIGHTWARDS ARROW |
| ↓ | &#8595; | &darr; | DOWNWARDS ARROW |
| ♠ | &#9824; | &spades; | BLACK SPADE SUIT |
| ♣ | &#9827; | &clubs; | BLACK CLUB SUIT |
| ♥ | &#9829; | &hearts; | BLACK HEART SUIT |
| ♦ | &#9830; | &diams; | BLACK DIAMOND SUIT |

# 第32章 在 HTML 中使用表情符号

表情符号（Emoji）是来自 UTF-8 字符集的字符：😄 😍 💗

什么是 Emoji？

表情符号（Emoji）类似图像或图标，但它们并不是。

它们是来自 UTF-8 (Unicode) 字符集的字母（字符）。

提示：UTF-8 几乎涵盖世界上所有字符和符号。

HTML charset 属性

为了正确显示 HTML 页面，Web 浏览器必须知道页面中使用的字符集。

这是在 <meta> 标签中规定的：

<meta charset="UTF-8">

提示：如果未规定，UTF-8 则是 HTML 中的默认字符集。

UTF-8 字符

很多 UTF-8 字符无法在键盘上键入，但始终可以使用数字（被称为实体编号）来显示它们：

A 是 65

B 是 66

C 是 67

实例

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

</head>

<body>

<p>我将显示 A B C</p>

<p>我将显示 &#65; &#66; &#67;</p>

</body>

</html>

例子解释

<meta charset="UTF-8"> 元素定义字符集。

字符 A、B、C 由数字 65、66 以及 67 来显示。

为了让浏览器了解您正在显示字符，您必须以 &# 开头并以 ;（分号）结束实体编号。

Emoji 字符

表情符号也是来自 UTF-8 字母的字符：

😄 是 128516

😍 是 128525

💗 是 128151

实例

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

</head>

<body>

<h1>我的第一个 Emoji</h1>

<p>&#128512;</p>

</body>

</html>

由于表情符号是字符，因此可以像 HTML 中的其他任何字符一样复制、显示和调整它们的大小。

实例

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

</head>

<body>

<h1>放大的表情符号</h1>

<p style="font-size:48px">

&#128512; &#128516; &#128525; &#128151;

</p>

</body>

</html>

UTF-8 中的一些 Emoji 符号

|  |  |
| --- | --- |
| Emoji | 值 |
| 🗻 | &#128507; |
| 🗼 | &#128508; |
| 🗽 | &#128509; |
| 🗾 | &#128510; |
| 🗿 | &#128511; |
| 😀 | &#128512; |
| 😁 | &#128513; |
| 😂 | &#128514; |
| 😃 | &#128515; |
| 😄 | &#128516; |
| 😅 | &#128517; |

# 第33章 HTML 编码（字符集）

为了正确显示 HTML 页面，Web 浏览器必须知道要使用哪个字符集。

## 33.1 从 ASCII 到 UTF-8

ASCII 是第一个字符编码标准。ASCII 定义了 128 种可以在互联网上使用的字符：数字（0-9）、英文字母（A-Z）和一些特殊字符，比如：! $ + - ( ) @ < >。

ISO-8859-1 是 HTML 4 的默认字符集。此字符集支持 256 个不同的字符代码。HTML 4 同时支持 UTF-8。

ANSI（Windows-1252）是原始的 Windows 字符集。 ANSI 与 ISO-8859-1 相同，不同之处在于 ANSI 具有 32 个额外的字符。

HTML5 规范鼓励 Web 开发人员使用 UTF-8 字符集，该字符集涵盖了世界上几乎所有的字符和符号！

## 33.2 HTML charset 属性

为了正确显示 HTML 页面，Web 浏览器必须了解页面中使用的字符集。

这在 <meta> 标签中指定：

<meta charset="UTF-8">

## 33.3 字符集之间的差异

下表展示了上述字符集之间的区别：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Numb | ASCII | ANSI | 8859 | UTF-8 | Description |
| 32 |  |  |  |  | space |
| 33 | ! | ! | ! | ! | exclamation mark |
| 34 | " | " | " | " | quotation mark |
| 35 | # | # | # | # | number sign |
| 36 | $ | $ | $ | $ | dollar sign |
| 37 | % | % | % | % | percent sign |
| 38 | & | & | & | & | ampersand |
| 39 | ' | ' | ' | ' | apostrophe |
| 40 | ( | ( | ( | ( | left parenthesis |
| 41 | ) | ) | ) | ) | right parenthesis |
| 42 | \* | \* | \* | \* | asterisk |
| 43 | + | + | + | + | plus sign |
| 44 | , | , | , | , | comma |
| 45 | - | - | - | - | hyphen-minus |
| 46 | . | . | . | . | full stop |
| 47 | / | / | / | / | solidus |
| 48 | 0 | 0 | 0 | 0 | digit zero |
| 49 | 1 | 1 | 1 | 1 | digit one |
| 50 | 2 | 2 | 2 | 2 | digit two |
| 51 | 3 | 3 | 3 | 3 | digit three |
| 52 | 4 | 4 | 4 | 4 | digit four |
| 53 | 5 | 5 | 5 | 5 | digit five |
| 54 | 6 | 6 | 6 | 6 | digit six |
| 55 | 7 | 7 | 7 | 7 | digit seven |
| 56 | 8 | 8 | 8 | 8 | digit eight |
| 57 | 9 | 9 | 9 | 9 | digit nine |
| 58 | : | : | : | : | colon |
| 59 | ; | ; | ; | ; | semicolon |
| 60 | < | < | < | < | less-than sign |
| 61 | = | = | = | = | equals sign |
| 62 | > | > | > | > | greater-than sign |
| 63 | ? | ? | ? | ? | question mark |
| 64 | @ | @ | @ | @ | commercial at |
| 65 | A | A | A | A | Latin capital letter A |
| 66 | B | B | B | B | Latin capital letter B |
| 67 | C | C | C | C | Latin capital letter C |
| 68 | D | D | D | D | Latin capital letter D |
| 69 | E | E | E | E | Latin capital letter E |
| 70 | F | F | F | F | Latin capital letter F |
| 71 | G | G | G | G | Latin capital letter G |
| 72 | H | H | H | H | Latin capital letter H |
| 73 | I | I | I | I | Latin capital letter I |
| 74 | J | J | J | J | Latin capital letter J |
| 75 | K | K | K | K | Latin capital letter K |
| 76 | L | L | L | L | Latin capital letter L |
| 77 | M | M | M | M | Latin capital letter M |
| 78 | N | N | N | N | Latin capital letter N |
| 79 | O | O | O | O | Latin capital letter O |
| 80 | P | P | P | P | Latin capital letter P |
| 81 | Q | Q | Q | Q | Latin capital letter Q |
| 82 | R | R | R | R | Latin capital letter R |
| 83 | S | S | S | S | Latin capital letter S |
| 84 | T | T | T | T | Latin capital letter T |
| 85 | U | U | U | U | Latin capital letter U |
| 86 | V | V | V | V | Latin capital letter V |
| 87 | W | W | W | W | Latin capital letter W |
| 88 | X | X | X | X | Latin capital letter X |
| 89 | Y | Y | Y | Y | Latin capital letter Y |
| 90 | Z | Z | Z | Z | Latin capital letter Z |
| 91 | [ | [ | [ | [ | left square bracket |
| 92 | \ | \ | \ | \ | reverse solidus |
| 93 | ] | ] | ] | ] | right square bracket |
| 94 | ^ | ^ | ^ | ^ | circumflex accent |
| 95 | \_ | \_ | \_ | \_ | low line |
| 96 | ` | ` | ` | ` | grave accent |
| 97 | a | a | a | a | Latin small letter a |
| 98 | b | b | b | b | Latin small letter b |
| 99 | c | c | c | c | Latin small letter c |
| 100 | d | d | d | d | Latin small letter d |
| 101 | e | e | e | e | Latin small letter e |
| 102 | f | f | f | f | Latin small letter f |
| 103 | g | g | g | g | Latin small letter g |
| 104 | h | h | h | h | Latin small letter h |
| 105 | i | i | i | i | Latin small letter i |
| 106 | j | j | j | j | Latin small letter j |
| 107 | k | k | k | k | Latin small letter k |
| 108 | l | l | l | l | Latin small letter l |
| 109 | m | m | m | m | Latin small letter m |
| 110 | n | n | n | n | Latin small letter n |
| 111 | o | o | o | o | Latin small letter o |
| 112 | p | p | p | p | Latin small letter p |
| 113 | q | q | q | q | Latin small letter q |
| 114 | r | r | r | r | Latin small letter r |
| 115 | s | s | s | s | Latin small letter s |
| 116 | t | t | t | t | Latin small letter t |
| 117 | u | u | u | u | Latin small letter u |
| 118 | v | v | v | v | Latin small letter v |
| 119 | w | w | w | w | Latin small letter w |
| 120 | x | x | x | x | Latin small letter x |
| 121 | y | y | y | y | Latin small letter y |
| 122 | z | z | z | z | Latin small letter z |
| 123 | { | { | { | { | left curly bracket |
| 124 | | | | | | | | | vertical line |
| 125 | } | } | } | } | right curly bracket |
| 126 | ~ | ~ | ~ | ~ | tilde |
| 127 | DEL |  |  |  |  |
| 128 |  | € |  |  | euro sign |
| 129 |  |  |  |  | NOT USED |
| 130 |  | ‚ |  |  | single low-9 quotation mark |
| 131 |  | ƒ |  |  | Latin small letter f with hook |
| 132 |  | „ |  |  | double low-9 quotation mark |
| 133 |  | … |  |  | horizontal ellipsis |
| 134 |  | † |  |  | dagger |
| 135 |  | ‡ |  |  | double dagger |
| 136 |  | ˆ |  |  | modifier letter circumflex accent |
| 137 |  | ‰ |  |  | per mille sign |
| 138 |  | Š |  |  | Latin capital letter S with caron |
| 139 |  | ‹ |  |  | single left-pointing angle quotation mark |
| 140 |  | Œ |  |  | Latin capital ligature OE |
| 141 |  |  |  |  | NOT USED |
| 142 |  | Ž |  |  | Latin capital letter Z with caron |
| 143 |  |  |  |  | NOT USED |
| 144 |  |  |  |  | NOT USED |
| 145 |  | ‘ |  |  | left single quotation mark |
| 146 |  | ’ |  |  | right single quotation mark |
| 147 |  | “ |  |  | left double quotation mark |
| 148 |  | ” |  |  | right double quotation mark |
| 149 |  | • |  |  | bullet |
| 150 |  | – |  |  | en dash |
| 151 |  | — |  |  | em dash |
| 152 |  | ˜ |  |  | small tilde |
| 153 |  | ™ |  |  | trade mark sign |
| 154 |  | š |  |  | Latin small letter s with caron |
| 155 |  | › |  |  | single right-pointing angle quotation mark |
| 156 |  | œ |  |  | Latin small ligature oe |
| 157 |  |  |  |  | NOT USED |
| 158 |  | ž |  |  | Latin small letter z with caron |
| 159 |  | Ÿ |  |  | Latin capital letter Y with diaeresis |
| 160 |  |  |  |  | no-break space |
| 161 |  | ¡ | ¡ | ¡ | inverted exclamation mark |
| 162 |  | ¢ | ¢ | ¢ | cent sign |
| 163 |  | £ | £ | £ | pound sign |
| 164 |  | ¤ | ¤ | ¤ | currency sign |
| 165 |  | ¥ | ¥ | ¥ | yen sign |
| 166 |  | ¦ | ¦ | ¦ | broken bar |
| 167 |  | § | § | § | section sign |
| 168 |  | ¨ | ¨ | ¨ | diaeresis |
| 169 |  | © | © | © | copyright sign |
| 170 |  | ª | ª | ª | feminine ordinal indicator |
| 171 |  | « | « | « | left-pointing double angle quotation mark |
| 172 |  | ¬ | ¬ | ¬ | not sign |
| 173 |  | ­ | ­ | ­ | soft hyphen |
| 174 |  | ® | ® | ® | registered sign |
| 175 |  | ¯ | ¯ | ¯ | macron |
| 176 |  | ° | ° | ° | degree sign |
| 177 |  | ± | ± | ± | plus-minus sign |
| 178 |  | ² | ² | ² | superscript two |
| 179 |  | ³ | ³ | ³ | superscript three |
| 180 |  | ´ | ´ | ´ | acute accent |
| 181 |  | µ | µ | µ | micro sign |
| 182 |  | ¶ | ¶ | ¶ | pilcrow sign |
| 183 |  | · | · | · | middle dot |
| 184 |  | ¸ | ¸ | ¸ | cedilla |
| 185 |  | ¹ | ¹ | ¹ | superscript one |
| 186 |  | º | º | º | masculine ordinal indicator |
| 187 |  | » | » | » | right-pointing double angle quotation mark |
| 188 |  | ¼ | ¼ | ¼ | vulgar fraction one quarter |
| 189 |  | ½ | ½ | ½ | vulgar fraction one half |
| 190 |  | ¾ | ¾ | ¾ | vulgar fraction three quarters |
| 191 |  | ¿ | ¿ | ¿ | inverted question mark |
| 192 |  | À | À | À | Latin capital letter A with grave |
| 193 |  | Á | Á | Á | Latin capital letter A with acute |
| 194 |  | Â | Â | Â | Latin capital letter A with circumflex |
| 195 |  | Ã | Ã | Ã | Latin capital letter A with tilde |
| 196 |  | Ä | Ä | Ä | Latin capital letter A with diaeresis |
| 197 |  | Å | Å | Å | Latin capital letter A with ring above |
| 198 |  | Æ | Æ | Æ | Latin capital letter AE |
| 199 |  | Ç | Ç | Ç | Latin capital letter C with cedilla |
| 200 |  | È | È | È | Latin capital letter E with grave |
| 201 |  | É | É | É | Latin capital letter E with acute |
| 202 |  | Ê | Ê | Ê | Latin capital letter E with circumflex |
| 203 |  | Ë | Ë | Ë | Latin capital letter E with diaeresis |
| 204 |  | Ì | Ì | Ì | Latin capital letter I with grave |
| 205 |  | Í | Í | Í | Latin capital letter I with acute |
| 206 |  | Î | Î | Î | Latin capital letter I with circumflex |
| 207 |  | Ï | Ï | Ï | Latin capital letter I with diaeresis |
| 208 |  | Ð | Ð | Ð | Latin capital letter Eth |
| 209 |  | Ñ | Ñ | Ñ | Latin capital letter N with tilde |
| 210 |  | Ò | Ò | Ò | Latin capital letter O with grave |
| 211 |  | Ó | Ó | Ó | Latin capital letter O with acute |
| 212 |  | Ô | Ô | Ô | Latin capital letter O with circumflex |
| 213 |  | Õ | Õ | Õ | Latin capital letter O with tilde |
| 214 |  | Ö | Ö | Ö | Latin capital letter O with diaeresis |
| 215 |  | × | × | × | multiplication sign |
| 216 |  | Ø | Ø | Ø | Latin capital letter O with stroke |
| 217 |  | Ù | Ù | Ù | Latin capital letter U with grave |
| 218 |  | Ú | Ú | Ú | Latin capital letter U with acute |
| 219 |  | Û | Û | Û | Latin capital letter U with circumflex |
| 220 |  | Ü | Ü | Ü | Latin capital letter U with diaeresis |
| 221 |  | Ý | Ý | Ý | Latin capital letter Y with acute |
| 222 |  | Þ | Þ | Þ | Latin capital letter Thorn |
| 223 |  | ß | ß | ß | Latin small letter sharp s |
| 224 |  | à | à | à | Latin small letter a with grave |
| 225 |  | á | á | á | Latin small letter a with acute |
| 226 |  | â | â | â | Latin small letter a with circumflex |
| 227 |  | ã | ã | ã | Latin small letter a with tilde |
| 228 |  | ä | ä | ä | Latin small letter a with diaeresis |
| 229 |  | å | å | å | Latin small letter a with ring above |
| 230 |  | æ | æ | æ | Latin small letter ae |
| 231 |  | ç | ç | ç | Latin small letter c with cedilla |
| 232 |  | è | è | è | Latin small letter e with grave |
| 233 |  | é | é | é | Latin small letter e with acute |
| 234 |  | ê | ê | ê | Latin small letter e with circumflex |
| 235 |  | ë | ë | ë | Latin small letter e with diaeresis |
| 236 |  | ì | ì | ì | Latin small letter i with grave |
| 237 |  | í | í | í | Latin small letter i with acute |
| 238 |  | î | î | î | Latin small letter i with circumflex |
| 239 |  | ï | ï | ï | Latin small letter i with diaeresis |
| 240 |  | ð | ð | ð | Latin small letter eth |
| 241 |  | ñ | ñ | ñ | Latin small letter n with tilde |
| 242 |  | ò | ò | ò | Latin small letter o with grave |
| 243 |  | ó | ó | ó | Latin small letter o with acute |
| 244 |  | ô | ô | ô | Latin small letter o with circumflex |
| 245 |  | õ | õ | õ | Latin small letter o with tilde |
| 246 |  | ö | ö | ö | Latin small letter o with diaeresis |
| 247 |  | ÷ | ÷ | ÷ | division sign |
| 248 |  | ø | ø | ø | Latin small letter o with stroke |
| 249 |  | ù | ù | ù | Latin small letter u with grave |
| 250 |  | ú | ú | ú | Latin small letter u with acute |
| 251 |  | û | û | û | Latin small letter with circumflex |
| 252 |  | ü | ü | ü | Latin small letter u with diaeresis |
| 253 |  | ý | ý | ý | Latin small letter y with acute |
| 254 |  | þ | þ | þ | Latin small letter thorn |
| 255 |  | ÿ | ÿ | ÿ | Latin small letter y with diaeresis |

## 33.4 ASCII 字符集

ASCII 使用 0 到 31（以及 127）之间的值作为控制字符。

ASCII 使用 32 到 126 的值表示字母、数字和符号。

ASCII 不使用 128 到 255 之间的值。

## 33.5 ANSI 字符集 (Windows-1252)

对于 0 到 127 的值，ANSI 与 ASCII 相同。

ANSI 有一组专有的字符，其值从 128 到 159。

对于 160 到 255 的值，ANSI 与 UTF-8 相同。

## 33.6 ISO-8859-1 字符集

对于 0 到 127 的值，8859-1 与 ASCII 相同。

8859-1 不使用 128 到 159 之间的值。

对于从 160 到 255 的值，8859-1 与 UTF-8 相同。

## 33.7 UTF-8 字符集

对于 0 到 127 的值，UTF-8 与 ASCII 相同。

UTF-8 不使用 12 8到 159 之间的值。

对于 160 到 255 之间的值，UTF-8 与 ANSI 和 8859-1 相同。

UTF-8 从值 256 继续，包含超过 10000 个不同字符。

如需深入研究，请阅读我们完整的 HTML 字符集参考手册。

## 33.8 @charset CSS 规则

您可以使用 CSS @charset 规则来指定样式表中使用的字符编码：

实例

将样式表的编码设置为 Unicode UTF-8：

@charset "UTF-8";

第34章 HTML 统一资源定位器

URL 也被称为网址。

URL 可以由单词组成，比如 “w3school.com.cn”，或者是因特网协议（IP）地址：192.168.1.253。大多数人在网上冲浪时，会键入网址的域名，因为名称比数字容易记忆。

URL - Uniform Resource Locator

当您点击 HTML 页面中的某个链接时，对应的 <a> 标签指向万维网上的一个地址。

统一资源定位器（URL）用于定位万维网上的文档（或其他数据）。

网址，比如 http://www.w3school.com.cn/html/index.asp，遵守以下的语法规则：

scheme://host.domain:port/path/filename

解释：

scheme - 定义因特网服务的类型。最常见的类型是 http

host - 定义域主机（http 的默认主机是 www）

domain - 定义因特网域名，比如 w3school.com.cn

:port - 定义主机上的端口号（http 的默认端口号是 80）

path - 定义服务器上的路径（如果省略，则文档必须位于网站的根目录中）。

filename - 定义文档/资源的名称

编者注：URL 的英文全称是 Uniform Resource Locator，中文也译为“统一资源定位符”。

URL Schemes

以下是其中一些最流行的 scheme：

|  |  |  |
| --- | --- | --- |
| Scheme | 访问 | 用于... |
| http | 超文本传输协议 | 以 http:// 开头的普通网页。不加密。 |
| https | 安全超文本传输协议 | 安全网页。加密所有信息交换。 |
| ftp | 文件传输协议 | 用于将文件下载或上传至网站。 |
| file |  | 您计算机上的文件。 |

# 第34章 HTML 框架

通过使用框架，你可以在同一个浏览器窗口中显示不止一个页面。

垂直框架

<html>

<frameset cols="25%,50%,25%">

<frame src="/example/html/frame\_a.html">

<frame src="/example/html/frame\_b.html">

<frame src="/example/html/frame\_c.html">

</frameset>

</html>

水平框架

<html>

<frameset rows="25%,50%,25%">

<frame src="/example/html/frame\_a.html">

<frame src="/example/html/frame\_b.html">

<frame src="/example/html/frame\_c.html">

</frameset>

</html>

## 34.1 框架

通过使用框架，你可以在同一个浏览器窗口中显示不止一个页面。每份HTML文档称为一个框架，并且每个框架都独立于其他的框架。

使用框架的坏处：

开发人员必须同时跟踪更多的HTML文档

很难打印整张页面

## 34.2 框架结构标签（<frameset>）

框架结构标签（<frameset>）定义如何将窗口分割为框架

每个 frameset 定义了一系列行或列

rows/columns 的值规定了每行或每列占据屏幕的面积

编者注：frameset 标签也被某些文章和书籍译为框架集。

## 34.3 框架标签（Frame）

Frame 标签定义了放置在每个框架中的 HTML 文档。

在下面的这个例子中，我们设置了一个两列的框架集。第一列被设置为占据浏览器窗口的 25%。第二列被设置为占据浏览器窗口的 75%。HTML 文档 "frame\_a.htm" 被置于第一个列中，而 HTML 文档 "frame\_b.htm" 被置于第二个列中：

<frameset cols="25%,75%">

<frame src="frame\_a.htm">

<frame src="frame\_b.htm">

</frameset>

## 34.4 基本的注意事项 - 有用的提示：

假如一个框架有可见边框，用户可以拖动边框来改变它的大小。为了避免这种情况发生，可以在 <frame> 标签中加入：noresize="noresize"。

为不支持框架的浏览器添加 <noframes> 标签。

重要提示：不能将 <body></body> 标签与 <frameset></frameset> 标签同时使用！不过，假如你添加包含一段文本的 <noframes> 标签，就必须将这段文字嵌套于 <body></body> 标签内。（在下面的第一个实例中，可以查看它是如何实现的。）

## 34.5 更多实例

**如何使用 <noframes> 标签**

<html>

<frameset cols="25%,50%,25%">

<frame src="/example/html/frame\_a.html">

<frame src="/example/html/frame\_b.html">

<frame src="/example/html/frame\_c.html">

<noframes>

<body>您的浏览器无法处理框架！</body>

</noframes>

</frameset>

</html>

**混合框架结构**

<html>

<frameset rows="50%,50%">

<frame src="/example/html/frame\_a.html">

<frameset cols="25%,75%">

<frame src="/example/html/frame\_b.html">

<frame src="/example/html/frame\_c.html">

</frameset>

</frameset>

</html>

**含有 noresize="noresize" 属性的框架结构**

<html>

<frameset cols="50%,\*,25%">

<frame src="/example/html/frame\_a.html" noresize="noresize" />

<frame src="/example/html/frame\_b.html" />

<frame src="/example/html/frame\_c.html" />

</frameset>

</html>

**导航框架**

<html>

<frameset cols="120,\*">

<frame src="/demo/html/content.html">

<frame src="/example/html/frame\_a.html" name="showframe">

</frameset>

</html>

**内联框架**

<html>

<body>

<iframe src="/i/eg\_landscape.jpg"></iframe>

<p>一些老的浏览器不支持 iframe。</p>

<p>如果得不到支持，iframe 是不可见的。</p>

</body>

</html>

**跳转至框架内的一个指定的节**

<html>

<frameset cols="20%,80%">

<frame src="/example/html/frame\_a.html">

<frame src="/example/html/link.html#C10">

</frameset>

</html>

**使用框架导航跳转至指定的节**

<html>

<frameset cols="180,\*">

<frame src="/demo/html/content.html">

<frame src="/example/html/link.html" name="showframe">

</frameset>

</html>

# 第35章 HTML 背景

好的背景使站点看上去特别棒。

## 35.1 实例

**搭配良好的背景和颜色**

<html>

<body bgcolor="#d0d0d0">

<p>

这是段落。这是段落。这是段落。这是段落。这是段落。这是段落。这是段落。这是段落。这是段落。这是段落。

</p>

<p>

这是另一个段落。这是另一个段落。这是另一个段落。这是另一个段落。这是另一个段落。这是另一个段落。这是另一个段落。

</p>

</body>

</html>

**搭配得不好的背景和颜色**

<html>

<body bgcolor="#ffffff" text="yellow">

<p>

这是段落。这是段落。这是段落。这是段落。这是段落。这是段落。这是段落。这是段落。这是段落。这是段落。

</p>

<p>

这是另一个段落。这是另一个段落。这是另一个段落。这是另一个段落。这是另一个段落。这是另一个段落。这是另一个段落。

</p>

</body>

</html>

## 35.2 背景（Backgrounds）

<body> 拥有两个配置背景的标签。背景可以是颜色或者图像。

### 35.2.1 背景颜色（Bgcolor）

背景颜色属性将背景设置为某种颜色。属性值可以是十六进制数、RGB 值或颜色名。

<body bgcolor="#000000">

<body bgcolor="rgb(0,0,0)">

<body bgcolor="black">

以上的代码均将背景颜色设置为黑色。

### 35.2.2 背景（Background）

背景属性将背景设置为图像。属性值为图像的URL。如果图像尺寸小于浏览器窗口，那么图像将在整个浏览器窗口进行复制。

<body background="clouds.gif">

<body background="http://www.w3school.com.cn/clouds.gif">

URL可以是相对地址，如第一行代码。也可以使绝对地址，如第二行代码。

提示：如果你打算使用背景图片，你需要紧记一下几点：

背景图像是否增加了页面的加载时间。小贴士：图像文件不应超过 10k。

背景图像是否与页面中的其他图象搭配良好。

背景图像是否与页面中的文字颜色搭配良好。

图像在页面中平铺后，看上去还可以吗？

对文字的注意力被背景图像喧宾夺主了吗？

## 35.3 基本的注意事项 - 有用的提示：

<body> 标签中的背景颜色（bgcolor）、背景（background）和文本（text）属性在最新的 HTML 标准（HTML4 和 XHTML）中已被废弃。W3C 在他们的推荐标准中已删除这些属性。

应该使用层叠样式表（CSS）来定义 HTML 元素的布局和显示属性。

**可用性强的背景图像**

<html>

<body background="/i/eg\_bg\_06.gif">

<h3>图像背景</h3>

<p>gif 和 jpg 文件均可用作 HTML 背景。</p>

<p>如果图像小于页面，图像会进行重复。</p>

</body>

</html>

# 第36章 HTML URL 字符编码

URL 编码会将字符转换为可通过因特网传输的格式。

URL - 统一资源定位器

Web 浏览器通过 URL 从 web 服务器请求页面。

URL 是网页的地址，比如 http://www.w3school.com.cn。

URL 编码

URL 只能使用 ASCII 字符集来通过因特网进行发送。

由于 URL 常常会包含 ASCII 集合之外的字符，URL 必须转换为有效的 ASCII 格式。

URL 编码使用 "%" 其后跟随两位的十六进制数来替换非 ASCII 字符。

URL 不能包含空格。URL 编码通常使用 + 来替换空格。

亲自试一试

如果您点击下面的“提交”按钮，浏览器会在发送输入之前对其进行 URL 编码。服务器上的页面会显示出接收到的输入。

Hello Kitty

试着输入一些字符，然后再次点击提交按钮。

URL 编码示例

|  |  |
| --- | --- |
| 字符 | URL 编码 |
| € | %80 |
| £ | %A3 |
| © | %A9 |
| ® | %AE |
| À | %C0 |
| Á | %C1 |
| Â | %C2 |
| Ã | %C3 |
| Ä | %C4 |
| Å | %C5 |

# 第37章 HTML Web Server

如果希望向世界发布您的网站，那么您必须把它存放在 web 服务器上。

托管自己的网站

在自己的服务器上托管网站始终是一个选项。有几点需要考虑：

硬件支出

如果要运行“真正”的网站，您不得不购买强大的服务器硬件。不要指望低价的 PC 能够应付这些工作。您还需要稳定的（一天 24 小时）高速连接。

软件支出

请记住，服务器授权通常比客户端授权更昂贵。同时请注意，服务器授权也许有用户数量限制。

人工费

不要指望低廉的人工费用。您必须安装自己的硬件和软件。您同时要处理漏洞和病毒，以确保您的服务器时刻正常地运行于一个“任何事都可能发生”的环境中。

使用因特网服务提供商（ISP）

从 ISP 租用服务器也很常见。

大多数小公司会把网站存放到由 ISP 提供的服务器上。其优势有以下几点：

连接速度

大多数 ISP 都拥有连接因特网的高速连接。

强大的硬件

ISP 的 web 服务器通常强大到能够由若干网站分享资源。您还要看一下 ISP 是否提供高效的负载平衡，以及必要的备份服务器。

安全性和可靠性

ISP 是网站托管方面的专家。他们应该提供 99% 以上的在线时间，最新的软件补丁，以及最好的病毒防护。

选择 ISP 时的注意事项

24 小时支持

确保 ISP 提供 24 小时支持。不要使自己置于无法解决严重问题的尴尬境地，同时还必须等待第二个工作日。如果您不希望支付长途电话费，那么免费电话服务也是必要的。

每日备份

确保 ISP 会执行每日备份的例行工作，否则您有可能损失有价值的数据。

流量

研究一下 ISP 的流量限制。如果出现由于网站受欢迎而激增的不可预期的访问量，那么您要确保不会因此支付额外费用。

带宽或内容限制

研究一下 ISP 的带宽和内容限制。如果您计划发布图片或播出视频或音频，请确保您有此权限。

E-mail 功能

请确保 ISP 支持您需要的 e-mail 功能。

数据库访问

如果您计划使用网站数据库中的数据，那么请确保您的 ISP 支持您需要的数据库访问。

在您选取一家 ISP 之前，请务必阅读 W3School 的 **Web 主机教程**。（**https://www.w3school.com.cn/hosting/index.asp）**

# 第38章 HTML <!DOCTYPE>

<!DOCTYPE> 声明帮助浏览器正确地显示网页。

<!DOCTYPE> 声明

Web 世界中存在许多不同的文档。只有了解文档的类型，浏览器才能正确地显示文档。

HTML 也有多个不同的版本，只有完全明白页面中使用的确切 HTML 版本，浏览器才能完全正确地显示出 HTML 页面。这就是 <!DOCTYPE> 的用处。

<!DOCTYPE> 不是 HTML 标签。它为浏览器提供一项信息（声明），即 HTML 是用什么版本编写的。

提示：W3School 即将升级为最新的 HTML5 文档类型。

实例

带有 HTML5 DOCTYPE 的 HTML 文档：

<!DOCTYPE html>

<html>

<head>

<title>Title of the document</title>

</head>

<body>

The content of the document......

</body>

</html>

HTML 版本

从 Web 诞生早期至今，已经发展出多个 HTML 版本：

版本 年份

HTML 1991

HTML+ 1993

HTML 2.0 1995

HTML 3.2 1997

HTML 4.01 1999

XHTML 1.0 2000

HTML5 2012

XHTML5 2013

常用的声明

HTML5

<!DOCTYPE html>

HTML 4.01

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

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

XHTML 1.0

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

如需完整的文档类型声明列表，请访问我们的 **DOCTYPE 参考手册https://www.w3school.com.cn/tags/tag\_doctype.asp**

# 第39章 HTML 快速参考

来自 W3School 的 HTML 快速参考。可以打印它，以备日常使用。

HTML 基础文档

<html>

<head>

<title>Document name goes here</title>

</head>

<body>

Visible text goes here

</body>

</html>

文本元素

<p>This is a paragraph</p>

<br> (line break)

<hr> (horizontal rule)

<pre>This text is preformatted</pre>

逻辑样式

<em>This text is emphasized</em>

<strong>This text is strong</strong>

<code>This is some computer code</code>

物理样式

<b>This text is bold</b>

<i>This text is italic</i>

链接、锚、图像元素

<a href="http://www.example.com/">This is a Link</a>

<a href="http://www.example.com/"><img src="URL"

alt="Alternate Text"></a>

<a href="mailto:webmaster@example.com">Send e-mail</a>A named anchor:

<a name="tips">Useful Tips Section</a>

<a href="#tips">Jump to the Useful Tips Section</a>

无序列表

<ul>

<li>First item</li>

<li>Next item</li>

</ul>

有序列表

<ol>

<li>First item</li>

<li>Next item</li>

</ol>

定义列表

<dl>

<dt>First term</dt>

<dd>Definition</dd>

<dt>Next term</dt>

<dd>Definition</dd>

</dl>

表格

<table border="1">

<tr>

<th>someheader</th>

<th>someheader</th>

</tr>

<tr>

<td>sometext</td>

<td>sometext</td>

</tr>

</table>

框架

<frameset cols="25%,75%">

<frame src="page1.htm">

<frame src="page2.htm">

</frameset>

表单

<form action="http://www.example.com/test.asp" method="post/get">

<input type="text" name="lastname"

value="Nixon" size="30" maxlength="50">

<input type="password">

<input type="checkbox" checked="checked">

<input type="radio" checked="checked">

<input type="submit">

<input type="reset">

<input type="hidden">

<select>

<option>Apples

<option selected>Bananas

<option>Cherries

</select>

<textarea name="Comment" rows="60"

cols="20"></textarea>

</form>

实体

&lt; is the same as <

&gt; is the same as >

&#169; is the same as ©

其他元素

<!-- This is a comment -->

<blockquote>

Text quoted from some source.

</blockquote>

<address>

Address 1<br>

Address 2<br>

City<br>

</address>

Source : <http://www.w3school.com.cn/html/html_quick.asp>

# 第40章 XHTML 简介

XHTML 是以 XML 格式编写的 HTML。

什么是 XHTML？

XHTML 指的是可扩展超文本标记语言

XHTML 与 HTML 4.01 几乎是相同的

XHTML 是更严格更纯净的 HTML 版本

XHTML 是以 XML 应用的方式定义的 HTML

XHTML 是 2001 年 1 月发布的 W3C 推荐标准

XHTML 得到所有主流浏览器的支持

为什么使用 XHTML？

因特网上的很多页面包含了“糟糕”的 HTML。

如果在浏览器中查看，下面的 HTML 代码运行起来非常正常（即使它并未遵守 HTML 规则）：

<html>

<head>

<title>This is bad HTML</title>

<body>

<h1>Bad HTML

<p>This is a paragraph

</body>

XML 是一种必须正确标记且格式良好的标记语言。

如果希望学习 XML，请阅读我们的 XML 教程。

今日的科技界存在一些不同的浏览器技术。其中一些在计算机上运行，而另一些可能在移动电话或其他小型设备上运行。小型设备往往缺乏解释“糟糕”的标记语言的资源和能力。

所以 - 通过结合 XML 和 HTML 的长处，开发出了 XHTML。XHTML 是作为 XML 被重新设计的 HTML。

与 HTML 相比最重要的区别：

文档结构

XHTML DOCTYPE 是强制性的

<html> 中的 XML namespace 属性是强制性的

<html>、<head>、<title> 以及 <body> 也是强制性的

元素语法

XHTML 元素必须正确嵌套

XHTML 元素必须始终关闭

XHTML 元素必须小写

XHTML 文档必须有一个根元素

属性语法

XHTML 属性必须使用小写

XHTML 属性值必须用引号包围

XHTML 属性最小化也是禁止的

<!DOCTYPE ....> 是强制性的

XHTML 文档必须进行 XHTML 文档类型声明（XHTML DOCTYPE declaration）。

您可以在 W3School 的标签参考手册中找到完整的 XHTML 文档类型。

<html>、<head>、<title> 以及 <body> 元素也必须存在，并且必须使用 <html> 中的 xmlns 属性为文档规定 xml 命名空间。

下面的例子展示了带有最少的必需标签的 XHTML 文档：

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Title of document</title>

</head>

<body>

......

</body>

</html>

如何从 HTML 转换到 XHTML

向每张页面的第一行添加 XHTML <!DOCTYPE>

向每张页面的 html 元素添加 xmlns 属性

把所有元素名改为小写

关闭所有空元素

把所有属性名改为小写

为所有属性值加引号

用 W3C 验证器检验 XHTML

在下面的文本框中输入您的网址：

http://www.w3school.com.cn/html/index.asp

XHTML 测验

该测验包含 20 道问题，且没有时间限制。

本测验是非正式的，它仅仅是了解您 XHTML 知识掌握程度的一个不错的途径。

每项正确答案可获得 1 分。在测试结束后，会显示您的总分。最高分为 20 分。

# 第41章 XHTML - 元素

XHTML 元素是以 XML 格式编写的 HTML 元素。

XHTML 元素 - 语法规则

XHTML 元素必须正确嵌套

XHTML 元素必须始终关闭

XHTML 元素必须小写

XHTML 文档必须有一个根元素

XHTML 元素必须正确嵌套

在 HTML 中，某些元素可以不正确地彼此嵌套在一起，就像这样：

<b><i>This text is bold and italic</b></i>

在 XHTML 中，所有元素必须正确地彼此嵌套，就像这样：

<b><i>This text is bold and italic</i></b>

XHTML 元素必须始终关闭

这是错误的：

<p>This is a paragraph

<p>This is another paragraph

这是正确的：

<p>This is a paragraph</p>

<p>This is another paragraph</p>

空元素也必须关闭

这是错误的：

A break: <br>

A horizontal rule: <hr>

An image: <img src="happy.gif" alt="Happy face">

这是正确的：

A break: <br />

A horizontal rule: <hr />

An image: <img src="happy.gif" alt="Happy face" />

XHTML 元素必须小写

这是错误的：

<BODY>

<P>This is a paragraph</P>

</BODY>

这是正确的：

<body>

<p>This is a paragraph</p>

</body>

# 第42章 XHTML - 属性

XHTML 属性是以 XML 格式编写的 HTML 属性。

XHTML 属性 - 语法规则

XHTML 属性必须使用小写

XHTML 属性值必须用引号包围

XHTML 属性最小化也是禁止的

XHTML 属性必须使用小写

这是错误的：

<table WIDTH="100%">

这是正确的：

<table width="100%">

XHTML 属性值必须用引号包围

这是错误的：

<table width=100%>

这是正确的：

<table width="100%">

禁止属性简写

这是错误的：

<input checked>

<input readonly>

<input disabled>

<option selected>

这是正确的：

<input checked="checked" />

<input readonly="readonly" />

<input disabled="disabled" />

<option selected="selected" />

# 第43章 HTML 表单

HTML 表单用于搜集不同类型的用户输入。

## 43.1 <form> 元素

HTML 表单用于收集用户输入。

<form> 元素定义 HTML 表单：

实例

<form>

.

form elements

.

</form>

## 43.2 HTML 表单包含表单元素。

表单元素指的是不同类型的 input 元素、复选框、单选按钮、提交按钮等等。

## 43.3 <input> 元素

<input> 元素是最重要的表单元素。

<input> 元素有很多形态，根据不同的 type 属性。

这是本章中使用的类型：

类型 描述

text 定义常规文本输入。

radio 定义单选按钮输入（选择多个选择之一）

submit 定义提交按钮（提交表单）

注释：您稍后将在本教程学到更多有关输入类型的知识。

**文本输入**

<input type="text"> 定义用于文本输入的单行输入字段：

实例

<form>

First name:<br>

<input type="text" name="firstname">

<br>

Last name:<br>

<input type="text" name="lastname">

</form>

**单选按钮输入**

<input type="radio"> 定义单选按钮。

单选按钮允许用户在有限数量的选项中选择其中之一：

实例

<form>

<input type="radio" name="sex" value="male" checked>Male

<br>

<input type="radio" name="sex" value="female">Female

</form>

**提交按钮**

<input type="submit"> 定义用于向表单处理程序（form-handler）提交表单的按钮。

表单处理程序通常是包含用来处理输入数据的脚本的服务器页面。

表单处理程序在表单的 action 属性中指定：

实例

<form action="action\_page.php">

First name:<br>

<input type="text" name="firstname" value="Mickey">

<br>

Last name:<br>

<input type="text" name="lastname" value="Mouse">

<br><br>

<input type="submit" value="Submit">

</form>

## 43.4 Action 属性

action 属性定义在提交表单时执行的动作。

向服务器提交表单的通常做法是使用提交按钮。

通常，表单会被提交到 web 服务器上的网页。

在上面的例子中，指定了某个服务器脚本来处理被提交表单：

<form action="action\_page.php">

如果省略 action 属性，则 action 会被设置为当前页面。

## 43.5 Method 属性

method 属性规定在提交表单时所用的 HTTP 方法（GET 或 POST）：

<form action="action\_page.php" method="GET">

或：

<form action="action\_page.php" method="POST">

## 43.6 何时使用 GET？

您能够使用 GET（默认方法）：

如果表单提交是被动的（比如搜索引擎查询），并且没有敏感信息。

当您使用 GET 时，表单数据在页面地址栏中是可见的：

action\_page.php?firstname=Mickey&lastname=Mouse

注释：GET 最适合少量数据的提交。浏览器会设定容量限制。

## 43.7 何时使用 POST？

您应该使用 POST：

如果表单正在更新数据，或者包含敏感信息（例如密码）。

POST 的安全性更好，因为在页面地址栏中被提交的数据是不可见的。

## 43.8 Name 属性

如果要正确地被提交，每个输入字段必须设置一个 name 属性。

本例只会提交 "Last name" 输入字段：

实例

<form action="action\_page.php">

First name:<br>

<input type="text" value="Mickey">

<br>

Last name:<br>

<input type="text" name="lastname" value="Mouse">

<br><br>

<input type="submit" value="Submit">

</form>

## 43.9 用 <fieldset> 组合表单数据

<fieldset> 元素组合表单中的相关数据

<legend> 元素为 <fieldset> 元素定义标题。

<!DOCTYPE html>

<html>

<body>

<form action="/demo/demo\_form.asp">

<fieldset>

<legend>Personal information:</legend>

First name:<br>

<input type="text" name="firstname" value="Mickey">

<br>

Last name:<br>

<input type="text" name="lastname" value="Mouse">

<br><br>

<input type="submit" value="Submit">

</fieldset>

</form>

</body>

</html>

## 43.10 HTML Form 属性

HTML <form> 元素，已设置所有可能的属性，是这样的：

实例

<form action="action\_page.php" method="GET" target="\_blank" accept-charset="UTF-8"

ectype="application/x-www-form-urlencoded" autocomplete="off" novalidate>

.

form elements

.

</form>

下面是 <form> 属性的列表：

|  |  |
| --- | --- |
| 属性 | 描述 |
| accept-charset | 规定在被提交表单中使用的字符集（默认：页面字符集）。 |
| action | 规定向何处提交表单的地址（URL）（提交页面）。 |
| autocomplete | 规定浏览器应该自动完成表单（默认：开启）。 |
| enctype | 规定被提交数据的编码（默认：url-encoded）。 |
| method | 规定在提交表单时所用的 HTTP 方法（默认：GET）。 |
| name | 规定识别表单的名称（对于 DOM 使用：document.forms.name）。 |
| novalidate | 规定浏览器不验证表单。 |
| target | 规定 action 属性中地址的目标（默认：\_self）。 |

# 第44章 HTML 表单属性

本章介绍 HTML <form> 元素的不同属性。

## 44.1 Action 属性

action 属性定义提交表单时要执行的操作。

通常，当用户单击“提交”按钮时，表单数据将发送到服务器上的文件中。

在下面的例子中，表单数据被发送到名为 "action\_page.php" 的文件。该文件包含处理表单数据的服务器端脚本：

实例

提交后，将表单数据发送到 "action\_page.php"：

<form action="/action\_page.php">

<label for="fname">First name:</label><br>

<input type="text" id="fname" name="fname" value="Bill"><br>

<label for="lname">Last name:</label><br>

<input type="text" id="lname" name="lname" value="Gates"><br><br>

<input type="submit" value="Submit">

</form>

提示：如果省略 action 属性，则将 action 设置为当前页面。

## 44.2 Target 属性

target 属性规定提交表单后在何处显示响应。

target 属性可设置以下值之一：

|  |  |
| --- | --- |
| 值 | 描述 |
| \_blank | 响应显示在新窗口或选项卡中。 |
| \_self | 响应显示在当前窗口中。 |
| \_parent | 响应显示在父框架中。 |
| \_top | 响应显示在窗口的整个 body 中。 |
| framename | 响应显示在命名的 iframe 中。 |

默认值为 \_self，这意味着响应将在当前窗口中打开。

实例

此处，提交的结果将在新的浏览器标签中打开：

<form action="/action\_page.php" target="\_blank">

## 44.3 Method 属性

method 属性指定提交表单数据时要使用的 HTTP 方法。

表单数据可以作为 URL 变量（使用 method="get"）或作为 HTTP post 事务（使用 method="post"）发送。

提交表单数据时，默认的 HTTP 方法是 GET。

实例

此例在提交表单数据时使用 GET 方法：

<form action="/action\_page.php" method="get">

此例在提交表单数据时使用 POST 方法：

<form action="/action\_page.php" method="post">

关于 GET 的注意事项：

以名称/值对的形式将表单数据追加到 URL

永远不要使用 GET 发送敏感数据！（提交的表单数据在 URL 中可见！）

URL 的长度受到限制（2048 个字符）

对于用户希望将结果添加为书签的表单提交很有用

GET 适用于非安全数据，例如 Google 中的查询字符串

关于 POST 的注意事项：

将表单数据附加在 HTTP 请求的正文中（不在 URL 中显示提交的表单数据）

POST 没有大小限制，可用于发送大量数据。

带有 POST 的表单提交无法添加书签

提示：如果表单数据包含敏感信息或个人信息，请务必使用 POST！

## 44.4 Autocomplete 属性

autocomplete 属性规定表单是否应打开自动完成功能。

启用自动完成功能后，浏览器会根据用户之前输入的值自动填写值。

实例

启用自动填写的表单：

<form action="/action\_page.php" autocomplete="on">

## 44.5 Novalidate 属性

novalidate 属性是一个布尔属性。

如果已设置，它规定提交时不应验证表单数据。

实例

未设置 novalidate 属性的表单：

<form action="/action\_page.php" novalidate>

## 44.6 所有 <form> 属性的列表

|  |  |
| --- | --- |
| 属性 | 描述 |
| accept-charset | 规定用于表单提交的字符编码。 |
| action | 规定提交表单时将表单数据发送到何处。 |
| autocomplete | 规定表单是否应打开自动完成（填写）功能。 |
| enctype | 规定将表单数据提交到服务器时应如何编码（仅供 method="post"）。 |
| method | 规定发送表单数据时要使用的 HTTP 方法。 |
| name | 规定表单名称。 |
| novalidate | 规定提交时不应验证表单。 |
| rel | 规定链接资源和当前文档之间的关系。 |
| target | 规定提交表单后在何处显示接收到的响应。 |

# 第45章 HTML 表单元素

本章描述所有 HTML 表单元素。

## 45.1 <input> 元素

最重要的表单元素是 <input> 元素。

<input> 元素根据不同的 type 属性，可以变化为多种形态。

注释：下一章讲解所有 HTML 输入类型。

## 45.2 <select> 元素（下拉列表）

<select> 元素定义下拉列表：

实例

<select name="cars">

<option value="volvo">Volvo</option>

<option value="saab">Saab</option>

<option value="fiat">Fiat</option>

<option value="audi">Audi</option>

</select>

## 45.3 <option> 元素定义待选择的选项。

列表通常会把首个选项显示为被选选项。

您能够通过添加 selected 属性来定义预定义选项。

实例

<option value="fiat" selected>Fiat</option>

## 45.4 <textarea> 元素

<textarea> 元素定义多行输入字段（文本域）：

实例

<textarea name="message" rows="10" cols="30">

The cat was playing in the garden.

</textarea>

## 45.5 <button> 元素

<button> 元素定义可点击的按钮：

实例

<button type="button" onclick="alert('Hello World!')">Click Me!</button>

## 45.6 HTML5 表单元素

HTML5 增加了如下表单元素：

<datalist>

<keygen>

<output>

注释：默认地，浏览器不会显示未知元素。新元素不会破坏您的页面。

## 45.7 HTML5 <datalist> 元素

<datalist> 元素为 <input> 元素规定预定义选项列表。

用户会在他们输入数据时看到预定义选项的下拉列表。

<input> 元素的 list 属性必须引用 <datalist> 元素的 id 属性。

实例

通过 <datalist> 设置预定义值的 <input> 元素：

<form action="action\_page.php">

<input list="browsers">

<datalist id="browsers">

<option value="Internet Explorer">

<option value="Firefox">

<option value="Chrome">

<option value="Opera">

<option value="Safari">

</datalist>

</form>

# 第46章 HTML 输入类型

本章描述 <input> 元素的输入类型。

## 46.1 输入类型：text

<input type="text"> 定义供文本输入的单行输入字段：

实例

<form>

First name:<br>

<input type="text" name="firstname">

<br>

Last name:<br>

<input type="text" name="lastname">

</form>

## 46.2 输入类型：password

<input type="password"> 定义密码字段：

实例

<form>

User name:<br>

<input type="text" name="username">

<br>

User password:<br>

<input type="password" name="psw">

</form>

注释：password 字段中的字符会被做掩码处理（显示为星号或实心圆）。

## 46.3 输入类型：submit

<input type="submit"> 定义提交表单数据至表单处理程序的按钮。

表单处理程序（form-handler）通常是包含处理输入数据的脚本的服务器页面。

在表单的 action 属性中规定表单处理程序（form-handler）：

实例

<form action="action\_page.php">

First name:<br>

<input type="text" name="firstname" value="Mickey">

<br>

Last name:<br>

<input type="text" name="lastname" value="Mouse">

<br><br>

<input type="submit" value="Submit">

</form>

如果您省略了提交按钮的 value 属性，那么该按钮将获得默认文本：

实例

<form action="action\_page.php">

First name:<br>

<input type="text" name="firstname" value="Mickey">

<br>

Last name:<br>

<input type="text" name="lastname" value="Mouse">

<br><br>

<input type="submit">

</form>

## 46.4 Input Type: radio

<input type="radio"> 定义单选按钮。

Radio buttons let a user select ONLY ONE of a limited number of choices:

实例

<form>

<input type="radio" name="sex" value="male" checked>Male

<br>

<input type="radio" name="sex" value="female">Female

</form>

## 46.5 Input Type: checkbox

<input type="checkbox"> 定义复选框。

复选框允许用户在有限数量的选项中选择零个或多个选项。

实例

<form>

<input type="checkbox" name="vehicle" value="Bike">I have a bike

<br>

<input type="checkbox" name="vehicle" value="Car">I have a car

</form>

## 46.6 Input Type: button

<input type="button> 定义按钮。

实例

<input type="button" onclick="alert('Hello World!')" value="Click Me!">

## 46.7 HTML5 输入类型

HTML5 增加了多个新的输入类型：

color

date

datetime

datetime-local

email

month

number

range

search

tel

time

url

week

注释：老式 web 浏览器不支持的输入类型，会被视为输入类型 text。

### 46.7.1 输入类型：number

<input type="number"> 用于应该包含数字值的输入字段。

您能够对数字做出限制。

根据浏览器支持，限制可应用到输入字段。

实例

<form>

Quantity (between 1 and 5):

<input type="number" name="quantity" min="1" max="5">

</form>

### 46.7.2 输入类型：date

<input type="date"> 用于应该包含日期的输入字段。

根据浏览器支持，日期选择器会出现输入字段中。

实例

<form>

Birthday:

<input type="date" name="bday">

</form>

您可以向输入添加限制：

实例

<form>

Enter a date before 1980-01-01:

<input type="date" name="bday" max="1979-12-31"><br>

Enter a date after 2000-01-01:

<input type="date" name="bday" min="2000-01-02"><br>

</form>

### 46.7.3 输入类型：color

<input type="color"> 用于应该包含颜色的输入字段。

根据浏览器支持，颜色选择器会出现输入字段中。

实例

<form>

Select your favorite color:

<input type="color" name="favcolor">

</form>

### 46.7.4 输入类型：range

<input type="range"> 用于应该包含一定范围内的值的输入字段。

根据浏览器支持，输入字段能够显示为滑块控件。

实例

<form>

<input type="range" name="points" min="0" max="10">

</form>

您能够使用如下属性来规定限制：min、max、step、value

### 46.7.5 输入类型：month

<input type="month"> 允许用户选择月份和年份。

根据浏览器支持，日期选择器会出现输入字段中。

实例

<form>

Birthday (month and year):

<input type="month" name="bdaymonth">

</form>

### 46.7.6 输入类型：week

<input type="week"> 允许用户选择周和年。

根据浏览器支持，日期选择器会出现输入字段中。

实例

<form>

Select a week:

<input type="week" name="week\_year">

</form>

### 46.7.7 输入类型：time

<input type="time"> 允许用户选择时间（无时区）。

根据浏览器支持，时间选择器会出现输入字段中。

实例

<form>

Select a time:

<input type="time" name="usr\_time">

</form>

### 46.7.8 输入类型：datetime

<input type="datetime"> 允许用户选择日期和时间（有时区）。

根据浏览器支持，日期选择器会出现输入字段中。

实例

<form>

Birthday (date and time):

<input type="datetime" name="bdaytime">

</form>

### 46.7.9 输入类型：datetime-local

<input type="datetime-local"> 允许用户选择日期和时间（无时区）。

根据浏览器支持，日期选择器会出现输入字段中。

实例

<form>

Birthday (date and time):

<input type="datetime-local" name="bdaytime">

</form>

### 46.7.10 输入类型：email

<input type="email"> 用于应该包含电子邮件地址的输入字段。

根据浏览器支持，能够在被提交时自动对电子邮件地址进行验证。

某些智能手机会识别 email 类型，并在键盘增加 ".com" 以匹配电子邮件输入。

实例

<form>

E-mail:

<input type="email" name="email">

</form>

### 46.7.11 输入类型：search

<input type="search"> 用于搜索字段（搜索字段的表现类似常规文本字段）。

实例

<form>

Search Google:

<input type="search" name="googlesearch">

</form>

### 46.7.12 输入类型：tel

<input type="tel"> 用于应该包含电话号码的输入字段。

目前只有 Safari 8 支持 tel 类型。

实例

<form>

Telephone:

<input type="tel" name="usrtel">

</form>

### 46.7.13 输入类型：url

<input type="url"> 用于应该包含 URL 地址的输入字段。

根据浏览器支持，在提交时能够自动验证 url 字段。

某些智能手机识别 url 类型，并向键盘添加 ".com" 以匹配 url 输入。

实例

<form>

Add your homepage:

<input type="url" name="homepage">

</form>

## 46.8 输入限制

这里列出了一些常用的输入限制（其中一些是 HTML5 中新增的）：

|  |  |
| --- | --- |
| 属性 | 描述 |
| disabled | 规定输入字段应该被禁用。 |
| max | 规定输入字段的最大值。 |
| maxlength | 规定输入字段的最大字符数。 |
| min | 规定输入字段的最小值。 |
| pattern | 规定通过其检查输入值的正则表达式。 |
| readonly | 规定输入字段为只读（无法修改）。 |
| required | 规定输入字段是必需的（必需填写）。 |
| size | 规定输入字段的宽度（以字符计）。 |
| step | 规定输入字段的合法数字间隔。 |
| value | 规定输入字段的默认值。 |

您将在下一章学到更多有关输入限制的知识。

实例

<form>

Quantity:

<input type="number" name="points" min="0" max="100" step="10" value="30">

</form>

# 第47章 HTML Input 属性

## 47.1 value 属性

value 属性规定输入字段的初始值：

实例

<form action="">

First name:<br>

<input type="text" name="firstname" value="Bill">

<br>

Last name:<br>

<input type="text" name="lastname">

</form>

## 47.2 readonly 属性

readonly 属性规定输入字段为只读（不能修改）：

实例

<form action="">

First name:<br>

<input type="text" name="firstname" value="Bill" readonly>

<br>

Last name:<br>

<input type="text" name="lastname">

</form>

readonly 属性不需要值。它等同于 readonly="readonly"。

## 47.3 disabled 属性

disabled 属性规定输入字段是禁用的。

被禁用的元素是不可用和不可点击的。

被禁用的元素不会被提交。

实例

<form action="">

First name:<br>

<input type="text" name="firstname" value="Bill" disabled>

<br>

Last name:<br>

<input type="text" name="lastname">

</form>

disabled 属性不需要值。它等同于 disabled="disabled"。

## 47.4 size 属性

size 属性规定输入字段的尺寸（以字符计）：

实例

<form action="">

First name:<br>

<input type="text" name="firstname" value="Bill" size="40">

<br>

Last name:<br>

<input type="text" name="lastname">

</form>

## 47.5 maxlength 属性

maxlength 属性规定输入字段允许的最大长度：

实例

<form action="">

First name:<br>

<input type="text" name="firstname" maxlength="10">

<br>

Last name:<br>

<input type="text" name="lastname">

</form>

如设置 maxlength 属性，则输入控件不会接受超过所允许数的字符。

该属性不会提供任何反馈。如果需要提醒用户，则必须编写 JavaScript 代码。

注释：输入限制并非万无一失。JavaScript 提供了很多方法来增加非法输入。如需安全地限制输入，则接受者（服务器）必须同时对限制进行检查。

## 47.6 HTML5 属性

HTML5 为 <input> 增加了如下属性：

autocomplete

autofocus

form

formaction

formenctype

formmethod

formnovalidate

formtarget

height 和 width

list

min 和 max

multiple

pattern (regexp)

placeholder

required

step

并为 <form> 增加如需属性：

autocomplete

Novalidate

## 47.7 autocomplete 属性

autocomplete 属性规定表单或输入字段是否应该自动完成。

当自动完成开启，浏览器会基于用户之前的输入值自动填写值。

提示：您可以把表单的 autocomplete 设置为 on，同时把特定的输入字段设置为 off，反之亦然。

autocomplete 属性适用于 <form> 以及如下 <input> 类型：text、search、url、tel、email、password、datepickers、range 以及 color。

实例

自动完成开启的 HTML 表单（某个输入字段为 off）：

<form action="action\_page.php" autocomplete="on">

First name:<input type="text" name="fname"><br>

Last name: <input type="text" name="lname"><br>

E-mail: <input type="email" name="email" autocomplete="off"><br>

<input type="submit">

</form>

提示：在某些浏览器中，您也许需要手动启用自动完成功能。

## 47.8 novalidate 属性

novalidate 属性属于 <form> 属性。

如果设置，则 novalidate 规定在提交表单时不对表单数据进行验证。

实例

指示表单在被提交时不进行验证：

<form action="action\_page.php" novalidate>

E-mail: <input type="email" name="user\_email">

<input type="submit">

</form>

## 47.8 autofocus 属性

autofocus 属性是布尔属性。

如果设置，则规定当页面加载时 <input> 元素应该自动获得焦点。

实例

使 "First name" 输入字段在页面加载时自动获得焦点：

First name:<input type="text" name="fname" autofocus>

## 47.9 form 属性

form 属性规定 <input> 元素所属的一个或多个表单。

提示：如需引用一个以上的表单，请使用空格分隔的表单 id 列表。

实例

输入字段位于 HTML 表单之外（但仍属表单）：

<form action="action\_page.php" id="form1">

First name: <input type="text" name="fname"><br>

<input type="submit" value="Submit">

</form>

Last name: <input type="text" name="lname" **form**="form1">

## 47.10 formaction 属性

formaction 属性规定当提交表单时处理该输入控件的文件的 URL。

formaction 属性覆盖 <form> 元素的 action 属性。

formaction 属性适用于 type="submit" 以及 type="image"。

实例

拥有两个两个提交按钮并对于不同动作的 HTML 表单：

<form action="action\_page.php">

First name: <input type="text" name="fname"><br>

Last name: <input type="text" name="lname"><br>

<input type="submit" value="Submit"><br>

<input type="submit" formaction="demo\_admin.asp"

value="Submit as admin">

</form>

## 47.11 formenctype 属性

formenctype 属性规定当把表单数据（form-data）提交至服务器时如何对其进行编码（仅针对 method="post" 的表单）。

formenctype 属性覆盖 <form> 元素的 enctype 属性。

formenctype 属性适用于 type="submit" 以及 type="image"。

实例

发送默认编码以及编码为 "multipart/form-data"（第二个提交按钮）的表单数据（form-data）：

<form action="demo\_post\_enctype.asp" method="post">

First name: <input type="text" name="fname"><br>

<input type="submit" value="Submit">

<input type="submit" formenctype="multipart/form-data"

value="Submit as Multipart/form-data">

</form>

## 47.12 formmethod 属性

formmethod 属性定义用以向 action URL 发送表单数据（form-data）的 HTTP 方法。

formmethod 属性覆盖 <form> 元素的 method 属性。

formmethod 属性适用于 type="submit" 以及 type="image"。

实例

第二个提交按钮覆盖表单的 HTTP 方法：

<form action="action\_page.php" method="get">

First name: <input type="text" name="fname"><br>

Last name: <input type="text" name="lname"><br>

<input type="submit" value="Submit">

<input type="submit" formmethod="post" formaction="demo\_post.asp"

value="Submit using POST">

</form>

## 47.13 formnovalidate 属性

novalidate 属性是布尔属性。

如果设置，则规定在提交表单时不对 <input> 元素进行验证。

formnovalidate 属性覆盖 <form> 元素的 novalidate 属性。

formnovalidate 属性可用于 type="submit"。

实例

拥有两个提交按钮的表单（验证和不验证）：

<form action="action\_page.php">

E-mail: <input type="email" name="userid"><br>

<input type="submit" value="Submit"><br>

<input type="submit" formnovalidate value="Submit without validation">

</form>

## 47.14 formtarget 属性

formtarget 属性规定的名称或关键词指示提交表单后在何处显示接收到的响应。

formtarget 属性会覆盖 <form> 元素的 target 属性。

formtarget 属性可与 type="submit" 和 type="image" 使用。

实例

这个表单有两个提交按钮，对应不同的目标窗口：

<form action="action\_page.php">

First name: <input type="text" name="fname"><br>

Last name: <input type="text" name="lname"><br>

<input type="submit" value="Submit as normal">

<input type="submit" formtarget="\_blank"

value="Submit to a new window">

</form>

## 47.15 height 和 width 属性

height 和 width 属性规定 <input> 元素的高度和宽度。

height 和 width 属性仅用于 <input type="image">。

注释：请始终规定图像的尺寸。如果浏览器不清楚图像尺寸，则页面会在图像加载时闪烁。

实例

把图像定义为提交按钮，并设置 height 和 width 属性：

<input type="image" src="img\_submit.gif" alt="Submit" width="48" height="48">

## 47.16 list 属性

list 属性引用的 <datalist> 元素中包含了 <input> 元素的预定义选项。

实例

使用 <datalist> 设置预定义值的 <input> 元素：

<input list="browsers">

<datalist id="browsers">

<option value="Internet Explorer">

<option value="Firefox">

<option value="Chrome">

<option value="Opera">

<option value="Safari">

</datalist>

## 47.17 min 和 max 属性

min 和 max 属性规定 <input> 元素的最小值和最大值。

min 和 max 属性适用于如需输入类型：number、range、date、datetime、datetime-local、month、time 以及 week。

实例

具有最小和最大值的 <input> 元素：

Enter a date before 1980-01-01:

<input type="date" name="bday" max="1979-12-31">

Enter a date after 2000-01-01:

<input type="date" name="bday" min="2000-01-02">

Quantity (between 1 and 5):

<input type="number" name="quantity" min="1" max="5">

## 47.18 multiple 属性

multiple 属性是布尔属性。

如果设置，则规定允许用户在 <input> 元素中输入一个以上的值。

multiple 属性适用于以下输入类型：email 和 file。

实例

接受多个值的文件上传字段：

Select images: <input type="file" name="img" multiple>

## 47.19 pattern 属性

pattern 属性规定用于检查 <input> 元素值的正则表达式。

pattern 属性适用于以下输入类型：text、search、url、tel、email、and password。

提示：请使用全局的 title 属性对模式进行描述以帮助用户。

提示：请在我们的 JavaScript 教程中学习更多有关正则表达式的知识。

实例

只能包含三个字母的输入字段（无数字或特殊字符）：

Country code:

<input type="text" name="country\_code" pattern="[A-Za-z]{3}" title="Three letter country code">

## 47.20 placeholder 属性

placeholder 属性规定用以描述输入字段预期值的提示（样本值或有关格式的简短描述）。

该提示会在用户输入值之前显示在输入字段中。

placeholder 属性适用于以下输入类型：text、search、url、tel、email 以及 password。

实例

拥有占位符文本的输入字段：

<input type="text" name="fname" placeholder="First name">

## 47.21 required 属性

required 属性是布尔属性。

如果设置，则规定在提交表单之前必须填写输入字段。

required 属性适用于以下输入类型：text、search、url、tel、email、password、date pickers、number、checkbox、radio、and file.

实例

必填的输入字段：

Username: <input type="text" name="usrname" required>

## 47.22 step 属性

step 属性规定 <input> 元素的合法数字间隔。

示例：如果 step="3"，则合法数字应该是 -3、0、3、6、等等。

提示：step 属性可与 max 以及 min 属性一同使用，来创建合法值的范围。

step 属性适用于以下输入类型：number、range、date、datetime、datetime-local、month、time 以及 week。

示例

拥有具体的合法数字间隔的输入字段：

<input type="number" name="points" step="3">

# 第48章 HTML Input form\* 属性

本章介绍 HTML <input> 元素的不同 form\* 属性。

## 48.1 form 属性

input 的 form 属性规定 <input> 元素所属的表单。

此属性的值必须等于它所属的 <form> 元素的 id 属性。

实例

位于 HTML 表单（但仍是表单的一部分）之外的输入字段：

<form action="/action\_page.php" id="form1">

<label for="fname">姓氏：</label>

<input type="text" id="fname" name="fname"><br><br>

<input type="submit" value="提交">

</form>

<label for="lname">名字：</label>

<input type="text" id="lname" name="lname" form="form1">

48.2 formaction 属性

input 的 formaction 属性规定当提交表单时，对输入（数据）进行处理的文件的 URL。

注释：该属性会覆盖 <form> 元素的 action 属性。

formaction 属性适用于以下输入类型：submit 和 image。

实例

带有两个提交按钮的 HTML 表单，它们具有不同的操作（action）：

<form action="/action\_page.php">

<label for="fname">姓氏：</label>

<input type="text" id="fname" name="fname"><br><br>

<label for="lname">名字：</label>

<input type="text" id="lname" name="lname"><br><br>

<input type="submit" value="提交">

<input type="submit" formaction="/action\_page2.php" value="以管理员提交">

</form>

## 48.2 formenctype 属性

input 的 formenctype 属性指定提交时应如何编码表单数据（仅适用于 method="post" 的表单）。

注释：此属性将覆盖 <form> 元素的 enctype 属性。

formenctype 属性适用于以下输入类型：submit 和 image。

实例

有两个提交按钮的表单。第一个发送使用默认编码的表单数据，第二个发送编码为 "multipart/form-data" 的表单数据：

<form action="/action\_page\_binary.asp" method="post">

<label for="fname">First name:</label>

<input type="text" id="fname" name="fname"><br><br>

<input type="submit" value="提交">

<input type="submit" formenctype="multipart/form-data"

value="以 Multipart/form-data 编码提交">

</form>

## 48.3 formmethod 属性

input 的 formmethod 属性定义了将表单数据发送到 action URL 的 HTTP 方法。

注释：此属性将覆盖 <form> 元素的 method 属性。

formmethod 属性适用于以下输入类型：submit 和 image。

表单数据可以作为 URL 变量（method="get"）或作为 HTTP post 事务（method="post"）发送。

关于 GET 的注意事项：

以名称/值对的形式将表单数据追加到 URL

永远不要使用 GET 发送敏感数据！（提交的表单数据在 URL 中可见！）

URL 的长度受到限制（2048 个字符）

对于用户希望将结果添加为书签的表单提交很有用

GET 适用于非安全数据，例如 Google 中的查询字符串

关于 POST 的注意事项：

将表单数据附加在 HTTP 请求的正文中（不在 URL 中显示提交的表单数据）

POST 没有大小限制，可用于发送大量数据。

带有 POST 的表单提交无法添加书签

提示：如果表单数据包含敏感信息或个人信息，请务必使用 POST！

实例

有两个提交按钮的表单。第一个使用 method="get" 发送表单数据。第二个使用 method="post" 发送表单数据：

<form action="/action\_page.php" method="get">

<label for="fname">姓氏：</label>

<input type="text" id="fname" name="fname"><br><br>

<label for="lname">名字：</label>

<input type="text" id="lname" name="lname"><br><br>

<input type="submit" value="使用 GET 提交">

<input type="submit" formmethod="post" value="使用 POST 提交">

</form>

## 48.4 formtarget 属性

input 的 formtarget 属性指定一个名称或关键字，该名称或关键字规定在提交表单后在何处显示收到的响应。

注释：此属性将覆盖 <form> 元素的 target 属性。

formtarget 属性适用于以下输入类型：submit 和 image。

实例

有两个提交按钮且有不同目标窗口的表单：

<form action="/action\_page.php">

<label for="fname">姓氏：</label>

<input type="text" id="fname" name="fname"><br><br>

<label for="lname">名字：</label>

<input type="text" id="lname" name="lname"><br><br>

<input type="submit" value="提交">

<input type="submit" formtarget="\_blank" value="提交到新窗口/标签页">

</form>

## 48.5 formnovalidate 属性

input 的 formnovalidate 属性规定提交时不应验证 <input> 元素。

注释：此属性将覆盖 <form> 元素的 novalidate 属性。

formnovalidate 属性适用于以下输入类型：submit。

实例

有两个提交按钮的表单（进行和不进行验证）：

<form action="/action\_page.php">

<label for="email">Enter your email:</label>

<input type="email" id="email" name="email"><br><br>

<input type="submit" value="提交">

<input type="submit" formnovalidate="formnovalidate"

value="不进行验证的提交">

</form>

## 48.6 novalidate 属性

novalidate 属性是 <form> 属性。

如果已设置，novalidate 属性规定在提交时不应验证所有表单数据。

实例

规定在提交时不验证任何表单数据：

<form action="/action\_page.php" novalidate>

<label for="email">Enter your email:</label>

<input type="email" id="email" name="email"><br><br>

<input type="submit" value="提交">

</form>

## 48.7 HTML Form 和 Input 元素

标签 描述

<form> 为用户输入定义 HTML 表单。

<input> 定义输入控件。

# 第49 章 HTML5 Canvas

canvas 元素用于在网页上绘制图形。

## 49.1 什么是 Canvas？

HTML5 的 canvas 元素使用 JavaScript 在网页上绘制图像。

画布是一个矩形区域，您可以控制其每一像素。

canvas 拥有多种绘制路径、矩形、圆形、字符以及添加图像的方法。

## 49.2 创建 Canvas 元素

向 HTML5 页面添加 canvas 元素。

规定元素的 id、宽度和高度：

<canvas id="myCanvas" width="200" height="100"></canvas>

## 49.3 通过 JavaScript 来绘制

canvas 元素本身是没有绘图能力的。所有的绘制工作必须在 JavaScript 内部完成：

<script type="text/javascript">

var c=document.getElementById("myCanvas");

var cxt=c.getContext("2d");

cxt.fillStyle="#FF0000";

cxt.fillRect(0,0,150,75);

</script>

JavaScript 使用 id 来寻找 canvas 元素：

var c=document.getElementById("myCanvas");

然后，创建 context 对象：

var cxt=c.getContext("2d");

getContext("2d") 对象是内建的 HTML5 对象，拥有多种绘制路径、矩形、圆形、字符以及添加图像的方法。

下面的两行代码绘制一个红色的矩形：

cxt.fillStyle="#FF0000";

cxt.fillRect(0,0,150,75);

fillStyle 方法将其染成红色，fillRect 方法规定了形状、位置和尺寸。

## 49.4 理解坐标

上面的 fillRect 方法拥有参数 (0,0,150,75)。

意思是：在画布上绘制 150x75 的矩形，从左上角开始 (0,0)。

如下图所示，画布的 X 和 Y 坐标用于在画布上对绘画进行定位。

<!DOCTYPE HTML>

<html>

<head>

<style type="text/css">

body

{

font-size:70%;

font-family:verdana,helvetica,arial,sans-serif;

}

</style>

<script type="text/javascript">

function cnvs\_getCoordinates(e)

{

x=e.clientX;

y=e.clientY;

document.getElementById("xycoordinates").innerHTML="Coordinates: (" + x + "," + y + ")";

}

function cnvs\_clearCoordinates()

{

document.getElementById("xycoordinates").innerHTML="";

}

</script>

</head>

<body style="margin:0px;">

<p>把鼠标悬停在下面的矩形上可以看到坐标：</p>

<div id="coordiv" style="float:left;width:199px;height:99px;border:1px solid #c3c3c3" onmousemove="cnvs\_getCoordinates(event)" onmouseout="cnvs\_clearCoordinates()"></div>

<br />

<br />

<br />

<div id="xycoordinates"></div>

</body>

</html>

## 49.5 更多 Canvas 实例

下面的在 canvas 元素上进行绘画的更多实例：

**实例 - 线条**

通过指定从何处开始，在何处结束，来绘制一条线：

<!DOCTYPE HTML>

<html>

<body>

<canvas id="myCanvas" width="200" height="100" style="border:1px solid #c3c3c3;">

Your browser does not support the canvas element.

</canvas>

<script type="text/javascript">

var c=document.getElementById("myCanvas");

var cxt=c.getContext("2d");

cxt.moveTo(10,10);

cxt.lineTo(150,50);

cxt.lineTo(10,50);

cxt.stroke();

</script>

</body>

</html>

**实例 - 圆形**

通过规定尺寸、颜色和位置，来绘制一个圆：

<!DOCTYPE HTML>

<html>

<body>

<canvas id="myCanvas" width="200" height="100" style="border:1px solid #c3c3c3;">

Your browser does not support the canvas element.

</canvas>

<script type="text/javascript">

var c=document.getElementById("myCanvas");

var cxt=c.getContext("2d");

cxt.fillStyle="#FF0000";

cxt.beginPath();

cxt.arc(70,18,15,0,Math.PI\*2,true);

cxt.closePath();

cxt.fill();

</script>

</body>

</html>

**实例 - 渐变**

使用您指定的颜色来绘制渐变背景：

<!DOCTYPE HTML>

<html>

<body>

<canvas id="myCanvas" width="200" height="100" style="border:1px solid #c3c3c3;">

Your browser does not support the canvas element.

</canvas>

<script type="text/javascript">

var c=document.getElementById("myCanvas");

var cxt=c.getContext("2d");

var grd=cxt.createLinearGradient(0,0,175,50);

grd.addColorStop(0,"#FF0000");

grd.addColorStop(1,"#00FF00");

cxt.fillStyle=grd;

cxt.fillRect(0,0,175,50);

</script>

</body>

</html>

**实例 - 图像**

把一幅图像放置到画布上：

<!DOCTYPE html>

<html>

<body>

<p>瑕佺敤鐨勫浘鍍忥細</p>

<img id="scream" width="224" height="162"

src="/i/eg\_flower.png" alt="The Scream">

<p>Canvas:</p>

<canvas id="myCanvas" width="244" height="182"

style="border:1px solid #d3d3d3;">

Your browser does not support the HTML5 canvas tag.

</canvas>

<script>

window.onload = function() {

var canvas = document.getElementById("myCanvas");

var ctx = canvas.getContext("2d");

var img = document.getElementById("scream");

ctx.drawImage(img, 10, 10);

};

</script>

</body>

</html>

# 第50章 HTML5 内联 SVG

HTML5 支持内联 SVG。

什么是 SVG？

SVG 指可伸缩矢量图形 (Scalable Vector Graphics)

SVG 用于定义用于网络的基于矢量的图形

SVG 使用 XML 格式定义图形

SVG 图像在放大或改变尺寸的情况下其图形质量不会有损失

SVG 是万维网联盟的标准

SVG 的优势

与其他图像格式相比（比如 JPEG 和 GIF），使用 SVG 的优势在于：

SVG 图像可通过文本编辑器来创建和修改

SVG 图像可被搜索、索引、脚本化或压缩

SVG 是可伸缩的

SVG 图像可在任何的分辨率下被高质量地打印

SVG 可在图像质量不下降的情况下被放大

浏览器支持

Internet Explorer 9、Firefox、Opera、Chrome 以及 Safari 支持内联 SVG。

把 SVG 直接嵌入 HTML 页面

在 HTML5 中，您能够将 SVG 元素直接嵌入 HTML 页面中：

实例

<!DOCTYPE html>

<html>

<body>

<svg xmlns="http://www.w3.org/2000/svg" version="1.1" height="190">

<polygon points="100,10 40,180 190,60 10,60 160,180"

style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />

</svg>

</body>

</html>

# 第51章 HTML 5 Canvas vs. SVG

Canvas 和 SVG 都允许您在浏览器中创建图形，但是它们在根本上是不同的。

SVG

SVG 是一种使用 XML 描述 2D 图形的语言。

SVG 基于 XML，这意味着 SVG DOM 中的每个元素都是可用的。您可以为某个元素附加 JavaScript 事件处理器。

在 SVG 中，每个被绘制的图形均被视为对象。如果 SVG 对象的属性发生变化，那么浏览器能够自动重现图形。

Canvas

Canvas 通过 JavaScript 来绘制 2D 图形。

Canvas 是逐像素进行渲染的。

在 canvas 中，一旦图形被绘制完成，它就不会继续得到浏览器的关注。如果其位置发生变化，那么整个场景也需要重新绘制，包括任何或许已被图形覆盖的对象。

Canvas 与 SVG 的比较

下表列出了 canvas 与 SVG 之间的一些不同之处。

Canvas

依赖分辨率

不支持事件处理器

弱的文本渲染能力

能够以 .png 或 .jpg 格式保存结果图像

最适合图像密集型的游戏，其中的许多对象会被频繁重绘

SVG

不依赖分辨率

支持事件处理器

最适合带有大型渲染区域的应用程序（比如谷歌地图）

复杂度高会减慢渲染速度（任何过度使用 DOM 的应用都不快）

不适合游戏应用

# 第52章 HTML 多媒体

Web 上的多媒体指的是音效、音乐、视频和动画。

现代网络浏览器已支持很多多媒体格式。

什么是多媒体？

多媒体来自多种不同的格式。它可以是您听到或看到的任何内容，文字、图片、音乐、音效、录音、电影、动画等等。

在因特网上，您会经常发现嵌入网页中的多媒体元素，现代浏览器已支持多种多媒体格式。

在本教程中，您将了解到不同的多媒体格式，以及如何在您的网页中使用它们。

浏览器支持

第一款因特网浏览器只支持文本，而且即使是对文本的支持也仅限于单一字体和单一颜色。随后诞生了支持颜色、字体和文本样式的浏览器，图片支持也被加入。

不同的浏览器以不同的方式处理对音效、动画和视频的支持。某些元素能够以内联的方式处理，而某些则需要额外的插件。

您将在下面的章节学习更多有关插件的知识。

多媒体格式

多媒体元素（比如视频和音频）存储于媒体文件中。

确定媒体类型的最常用的方法是查看文件扩展名。当浏览器得到文件扩展名 .htm 或 .html 时，它会假定该文件是 HTML 页面。.xml 扩展名指示 XML 文件，而 .css 扩展名指示样式表。图片格式则通过 .gif 或 .jpg 来识别。

多媒体元素元素也拥有带有不同扩展名的文件格式，比如 .swf、.wmv、.mp3 以及 .mp4。

视频格式

MP4 格式是一种新的即将普及的因特网视频格式。HTML5 、Flash 播放器以及优酷等视频网站均支持它。

|  |  |  |
| --- | --- | --- |
| 格式 | 文件 | 描述 |
| AVI | .avi | AVI (Audio Video Interleave) 格式是由微软开发的。所有运行 Windows 的计算机都支持 AVI 格式。它是因特网上很常见的格式，但非 Windows 计算机并不总是能够播放。 |
| WMV | .wmv | Windows Media 格式是由微软开发的。Windows Media 在因特网上很常见，但是如果未安装额外的（免费）组件，就无法播放 Windows Media 电影。一些后期的 Windows Media 电影在所有非 Windows 计算机上都无法播放，因为没有合适的播放器。 |
| MPEG |  |  |
| .mpg |  |  |
| .mpeg |  |  |
| MPEG (Moving Pictures Expert Group) 格式是因特网上最流行的格式。它是跨平台的，得到了所有最流行的浏览器的支持。 |  |  |
| QuickTime | .mov | QuickTime 格式是由苹果公司开发的。QuickTime 是因特网上常见的格式，但是 QuickTime 电影不能在没有安装额外的（免费）组件的 Windows 计算机上播放。 |

声音格式

|  |  |  |
| --- | --- | --- |
| 格式 | 文件 | 描述 |
| MIDI | .mid  .midi | MIDI (Musical Instrument Digital Interface) 是一种针对电子音乐设备（比如合成器和声卡）的格式。MIDI 文件不含有声音，但包含可被电子产品（比如声卡）播放的数字音乐指令。  点击这里播放 The Beatles。  因为 MIDI 格式仅包含指令，所以 MIDI 文件极其小巧。上面的例子只有 23k 的大小，但却能播放将近 5 分钟。MIDI 得到了广泛的平台上的大量软件的支持。大多数流行的网络浏览器都支持 MIDI。 |
| RealAudio | .rm  .ram | RealAudio 格式是由 Real Media 针对因特网开发的。该格式也支持视频。该格式允许低带宽条件下的音频流（在线音乐、网络音乐）。由于是低带宽优先的，质量常会降低。 |
| Wave | .wav | Wave (waveform) 格式是由 IBM 和微软开发的。所有运行 Windows 的计算机和所有网络浏览器（除了 Google Chrome）都支持它。 |
| WMA | .wma | WMA 格式 (Windows Media Audio)，质量优于 MP3，兼容大多数播放器，除了 iPod。WMA 文件可作为连续的数据流来传输，这使它对于网络电台或在线音乐很实用。 |
| MP3 | .mp3  .mpga | MP3 文件实际上是 MPEG 文件的声音部分。MPEG 格式最初是由运动图像专家组开发的。MP3 是其中最受欢迎的针对音乐的声音格式。期待未来的软件系统都支持它。 |

使用哪种格式？

WAVE 是因特网上最受欢迎的无压缩声音格式，所有流行的浏览器都支持它。如果您需要未经压缩的声音（音乐或演讲），那么您应该使用 WAVE 格式。

MP3 是最新的压缩录制音乐格式。MP3 这个术语已经成为数字音乐的代名词。如果您的网址从事录制音乐，那么 MP3 是一个选项。

# 第53章 HTML 插件

插件（Plug-in）是扩展浏览器标准功能的计算机程序。

## 53.1 插件

插件被设计用于许多不同的目的：

运行 Java 小程序

运行 ActiveX 控件

显示 Flash 电影

显示地图

扫描病毒

验证银行账号

警告！

大多数浏览器不再支持 Java Applet 和插件。

所有浏览器均不再支持 ActiveX 控件。

在现代浏览器中，对 Shockwave Flash 的支持也已关闭。

## 53.2 <object> 元素

所有浏览器均支持 <object> 元素。

<object> 元素定义 HTML 文档中的嵌入式对象。

它旨在将插件（例如 Java applet、PDF 阅读器和 Flash 播放器）嵌入网页中，但也可以用于将 HTML 包含在 HTML 中：

实例

<object width="100%" height="500px" data="snippet.html"></object>

或者您喜爱的图像：

实例

<object data="audi.jpeg"></object>

## 53.3 <embed> 元素

所有主要浏览器均支持 <embed> 元素。

<embed> 元素也可定义了 HTML 文档中的嵌入式对象。

Web 浏览器长期以来一直支持 <embed> 元素。但是，它不属于 HTML5 之前的 HTML 规范的一部分。

实例

<embed src="audi.jpeg">

请注意，<embed> 元素没有结束标记。它无法包含替代文本。

<embed> 元素也可用于在 HTML 中包含 HTML：

实例

<embed width="100%" height="500px" src="snippet.html">

# 第54章 HTML 音频

在 HTML 中播放声音的方法有很多种。

问题，问题，以及解决方法

在 HTML 中播放音频并不容易！

您需要谙熟大量技巧，以确保您的音频文件在所有浏览器中（Internet Explorer, Chrome, Firefox, Safari, Opera）和所有硬件上（PC, Mac , iPad, iPhone）都能够播放。

在本章，W3School 为您总结了问题和解决方法。

## 54.1 使用插件

浏览器插件是一种扩展浏览器标准功能的小型计算机程序。

插件有很多用途：播放音乐、显示地图、验证银行账号，控制输入等等。

可使用 <object> 或 <embed> 标签来将插件添加到 HTML 页面。

这些标签定义资源（通常非 HTML 资源）的容器，根据类型，它们即会由浏览器显示，也会由外部插件显示。

## 54.2 使用 <embed> 元素

<embed> 标签定义外部（非 HTML）内容的容器。（这是一个 HTML5 标签，在 HTML4 中是非法的，但是所有浏览器中都有效）。

下面的代码片段能够显示嵌入网页中的 MP3 文件：

实例

<embed height="100" width="100" src="song.mp3" />

问题：

<embed> 标签在 HTML 4 中是无效的。页面无法通过 HTML 4 验证。

不同的浏览器对音频格式的支持也不同。

如果浏览器不支持该文件格式，没有插件的话就无法播放该音频。

如果用户的计算机未安装插件，无法播放音频。

如果把该文件转换为其他格式，仍然无法在所有浏览器中播放。

注释：使用 <!DOCTYPE html> (HTML5) 解决验证问题。

## 54.3 使用 <object> 元素

<object tag> 标签也可以定义外部（非 HTML）内容的容器。

下面的代码片段能够显示嵌入网页中的 MP3 文件：

实例

<object height="100" width="100" data="song.mp3"></object>

问题：

不同的浏览器对音频格式的支持也不同。

如果浏览器不支持该文件格式，没有插件的话就无法播放该音频。

如果用户的计算机未安装插件，无法播放音频。

如果把该文件转换为其他格式，仍然无法在所有浏览器中播放。

## 54.4 使用 HTML5 <audio> 元素

<audio> 元素是一个 HTML5 元素，在 HTML 4 中是非法的，但在所有浏览器中都有效。

实例

<audio controls="controls">

<source src="song.mp3" type="audio/mp3" />

<source src="song.ogg" type="audio/ogg" />

Your browser does not support this audio format.

</audio>

上面的例子使用了一个 mp3 文件，这样它在 Internet Explorer、Chrome 以及 Safari 中是有效的。

为了使这段音频在 Firefox 和 Opera 中同样有效，添加了一个 ogg 类型的文件。如果失败，会显示错误消息。

问题：

<audio> 标签在 HTML 4 中是无效的。您的页面无法通过 HTML 4 验证。

您必须把音频文件转换为不同的格式。

<audio> 元素在老式浏览器中不起作用。

注释：使用 <!DOCTYPE html> (HTML5) 解决验证问题。

最好的 HTML 解决方法

实例

<audio controls="controls" height="100" width="100">

<source src="song.mp3" type="audio/mp3" />

<source src="song.ogg" type="audio/ogg" />

<embed height="100" width="100" src="song.mp3" />

</audio>

上面的例子使用了两个不同的音频格式。HTML5 <audio> 元素会尝试以 mp3 或 ogg 来播放音频。如果失败，代码将回退尝试 <embed> 元素。

问题：

您必须把音频转换为不同的格式。

<audio> 元素无法通过 HTML 4 和 XHTML 验证。

<embed> 元素无法通过 HTML 4 和 XHTML 验证。

<embed> 元素无法回退来显示错误消息。

注释：使用 <!DOCTYPE html> (HTML5) 解决验证问题。

向网站添加音频的最简单方法

向网页添加音频的最简单的方法是什么？

雅虎的媒体播放器绝对算其中之一。

使用雅虎媒体播放器是一个不同的途径。您只需简单地让雅虎来完成歌曲播放的工作就好了。

它能播放 mp3 以及一系列其他格式。通过一行简单的代码，您就可以把它添加到网页中，轻松地将 HTML 页面转变为专业的播放列表。

## 54.5 雅虎媒体播放器

实例

<a href="song.mp3">Play Sound</a>

<script type="text/javascript" src="http://mediaplayer.yahoo.com/js">

</script>

使用雅虎播放器是免费的。如需使用它，您需要把这段 JavaScript 插入网页底部：

<script type="text/javascript" src="http://mediaplayer.yahoo.com/js"></script>

然后只需简单地把 MP3 文件链接到您的 HTML 中，JavaScript 会自动地为每首歌创建播放按钮：

<a href="song1.mp3">Play Song 1</a>

<a href="song2.mp3">Play Song 2</a>

...

...

...

雅虎媒体播放器为您的用户提供的是一个小型的播放按钮，而不是完整的播放器。不过，当您点击该按钮，会弹出完整的播放器。

请注意，这个播放器始终停靠在窗框底部。只需点击它，就可将其滑出。

## 54.6 使用超链接

如果网页包含指向媒体文件的超链接，大多数浏览器会使用“辅助应用程序”来播放文件。

以下代码片段显示指向 mp3 文件的链接。如果用户点击该链接，浏览器会启动“辅助应用程序”来播放该文件：

实例

<a href="song.mp3">Play the sound</a>

## 54.7 内联的声音

当您在网页中包含声音，或者作为网页的组成部分时，它被称为内联声音。

如果您打算在 web 应用程序中使用内联声音，您需要意识到很多人都觉得内联声音令人恼火。同时请注意，用户可能已经关闭了浏览器中的内联声音选项。

我们最好的建议是只在用户希望听到内联声音的地方包含它们。一个正面的例子是，在用户需要听到录音并点击某个链接时，会打开页面然后播放录音。

## 54.8 HTML 4.01 多媒体标签

|  |  |
| --- | --- |
| 标签 | 描述 |
| <applet> | 不赞成。定义内嵌 applet。 |
| <embed> | HTML4 中不赞成，HTML5 中允许。定义内嵌对象。 |
| <object> | 定义内嵌对象。 |
| <param> | 定义对象的参数。 |

## 54.9 HTML 5 多媒体标签

|  |  |
| --- | --- |
| 标签 | 描述 |
| <audio> | 标签定义声音，比如音乐或其他音频流。 |
| <embed> | 标签定义嵌入的内容，比如插件。 |

# 第55章 HTML 视频

在 HTML 中播放视频的方法有很多种。

<video width="320" height="240" controls="controls">

<source src="movie.mp4" type="video/mp4" />

<source src="movie.ogg" type="video/ogg" />

<source src="movie.webm" type="video/webm" />

<object data="movie.mp4" width="320" height="240">

<embed src="movie.swf" width="320" height="240" />

</object>

</video>

问题，问题，以及解决方法

在 HTML 中播放视频并不容易！

您需要谙熟大量技巧，以确保您的视频文件在所有浏览器中（Internet Explorer, Chrome, Firefox, Safari, Opera）和所有硬件上（PC, Mac , iPad, iPhone）都能够播放。

在本章，W3School 为您总结了问题和解决方法。

## 55.1 使用 <embed> 标签

<embed> 标签的作用是在 HTML 页面中嵌入多媒体元素。

下面的 HTML 代码显示嵌入网页的 Flash 视频：

实例

<embed src="movie.swf" height="200" width="200"/>

问题

HTML4 无法识别 <embed> 标签。您的页面无法通过验证。

如果浏览器不支持 Flash，那么视频将无法播放

iPad 和 iPhone 不能显示 Flash 视频。

如果您将视频转换为其他格式，那么它仍然不能在所有浏览器中播放。

## 55.2 使用 <object> 标签

<object> 标签的作用是在 HTML 页面中嵌入多媒体元素。

下面的 HTML 片段显示嵌入网页的一段 Flash 视频：

实例

<object data="movie.swf" height="200" width="200"/>

问题

如果浏览器不支持 Flash，将无法播放视频。

iPad 和 iPhone 不能显示 Flash 视频。

如果您将视频转换为其他格式，那么它仍然不能在所有浏览器中播放。

## 55.3 使用 <video> 标签

<video> 是 HTML 5 中的新标签。

<video> 标签的作用是在 HTML 页面中嵌入视频元素。

以下 HTML 片段会显示一段嵌入网页的 ogg、mp4 或 webm 格式的视频：

实例

<video width="320" height="240" controls="controls">

<source src="movie.mp4" type="video/mp4" />

<source src="movie.ogg" type="video/ogg" />

<source src="movie.webm" type="video/webm" />

Your browser does not support the video tag.

</video>

问题

您必须把视频转换为很多不同的格式。

<video> 元素在老式浏览器中无效。

<video> 元素无法通过 HTML 4 和 XHTML 验证。

## 55.4 最好的 HTML 解决方法

HTML 5 + <object> + <embed>

<video width="320" height="240" controls="controls">

<source src="movie.mp4" type="video/mp4" />

<source src="movie.ogg" type="video/ogg" />

<source src="movie.webm" type="video/webm" />

<object data="movie.mp4" width="320" height="240">

<embed src="movie.swf" width="320" height="240" />

</object>

</video>

上例中使用了 4 中不同的视频格式。HTML 5 <video> 元素会尝试播放以 mp4、ogg 或 webm 格式中的一种来播放视频。如果均失败，则回退到 <embed> 元素。

问题

您必须把视频转换为很多不同的格式

<video> 元素无法通过 HTML 4 和 XHTML 验证。

<embed> 元素无法通过 HTML 4 和 XHTML 验证。

注释：使用 <!DOCTYPE html> (HTML5) 解决验证问题。

## 55.5 优酷解决方案

在 HTML 中显示视频的最简单的方法是使用优酷等视频网站。

如果您希望在网页中播放视频，那么您可以把视频上传到优酷等视频网站，然后在您的网页中插入 HTML 代码即可播放视频：

<embed src="http://player.youku.com/player.php/sid/XMzI2NTc4NTMy/v.swf"

width="480" height="400"

type="application/x-shockwave-flash">

</embed>

## 55.6 使用超链接

如果网页包含指向媒体文件的超链接，大多数浏览器会使用“辅助应用程序”来播放文件。

以下代码片段显示指向 AVI 文件的链接。如果用户点击该链接，浏览器会启动“辅助应用程序”，比如 Windows Media Player 来播放这个 AVI 文件：

实例

<a href="movie.swf">Play a video file</a>

## 55.7 关于内联视频的一段注释

当视频被包含在网页中时，它被称为内联视频。

如果您打算在 web 应用程序中使用内联视频，您需要意识到很多人都觉得内联视频令人恼火。

同时请注意，用户可能已经关闭了浏览器中的内联视频选项。

我们最好的建议是只在用户希望看到内联视频的地方包含它们。一个正面的例子是，在用户需要看到视频并点击某个链接时，会打开页面然后播放视频。

## 55.8 HTML 4.01 多媒体标签

|  |  |
| --- | --- |
| 标签 | 描述 |
| <applet> | 不赞成。定义内嵌 applet。 |
| <embed> | 不赞成。定义内嵌对象。（HTML5 中允许） |
| <object> | 定义内嵌对象。 |
| <param> | 定义对象的参数。 |

## 55.9 HTML 5 多媒体标签

|  |  |
| --- | --- |
| 标签 | 描述 |
| <video> | 标签定义声音，比如音乐或其他音频流。 |
| <embed> | 标签定义嵌入的内容，比如插件。 |

# 第56 章 HTML YouTube 视频

在 HTML 中包含视频的最简单的方法是，使用 YouTube。

纠结视频格式？

将视频转换为不同的格式可能既困难又耗时。

一个更简单的解决方案是让 YouTube 在您的网页中播放视频。

YouTube Video Id

保存（或播放）视频时，YouTube 会显示一个 id（例如 ih1l6wb7LhU）。

您可以使用这个 id，并在 HTML 代码中引用您的视频。

在 HTML 中保费 YouTube 视频

如需在网页上播放视频，请执行以下操作：

将视频上传到 YouTube

记下视频 id

在您的网页中定义 <iframe> 元素

让 src 属性指向视频的 URL

使用 width 和 height 属性来规定播放器的尺寸

向 URL 添加其他参数（参阅下文）

实例

<iframe width="420" height="315"

src="https://www.youtube.com/embed/ih1l6wb7LhU">

</iframe>

亲自试一试

YouTube Autoplay + Mute

您可以通过在 YouTube URL 上添加 autoplay=1 来让视频在用户访问页面时自动开始播放。但是，自动开始播放视频会让您的访问者感到烦恼！

注意：在大多数情况下，Chromium 浏览器都不允许自动播放。但始终允许静音自动播放。

在 autoplay=1 之后添加 mute=1，可让您的视频自动开始播放（但已静音）。

YouTube - Autoplay + Muted

<iframe width="420" height="315"

src="https://www.youtube.com/embed/ih1l6wb7LhU?autoplay=1&mute=1">

</iframe>

亲自试一试

YouTube Playlist

以逗号分隔的要播放的视频列表（原始 URL 除外）。

YouTube Loop

添加 loop=1 会让您的视频永远循环。

值 0（默认）：视频将播放一次。

值 1：视频将循环（永远）。

YouTube - Loop

<iframe width="420" height="315"

src="https://www.youtube.com/embed/ih1l6wb7LhU?playlist=ih1l6wb7LhU&loop=1">

</iframe>

亲自试一试

YouTube Controls

添加 controls=0 会使视频播放器不显示控件。

值 0：播放器控件不显示。

值 1（默认）：播放器控件显示。

YouTube - Controls

<iframe width="420" height="315"

src="https://www.youtube.com/embed/ih1l6wb7LhU?controls=0">

</iframe>

# 第57章 HTML5 简介

每一章中的 HTML5 示例

实例

<!DOCTYPE html>

<html>

<body>

<video width="420" controls>

<source src="mov\_bbb.mp4" type="video/mp4">

<source src="mov\_bbb.ogg" type="video/ogg">

Your browser does not support the video tag.

</video>

</body>

</html>

什么是 HTML5？

HTML5 是最新的 HTML 标准。

HTML5 是专门为承载丰富的 web 内容而设计的，并且无需额外插件。

HTML5 拥有新的语义、图形以及多媒体元素。

HTML5 提供的新元素和新的 API 简化了 web 应用程序的搭建。

HTML5 是跨平台的，被设计为在不同类型的硬件（PC、平板、手机、电视机等等）之上运行。

注释：在下面的章节中，您将学到如何“帮助”老版本的浏览器处理 HTML5。

HTML5 中的新内容？

HTML5 的新的文档类型（DOCTYPE）声明非常简单：

<!DOCTYPE html>

The new character encoding (charset) declaration is also very simple:

<meta charset="UTF-8">

HTML5 实例：

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Title of the document</title>

</head>

<body>

Content of the document......

</body>

</html>

注释：HTML5 中默认的字符编码是 UTF-8。

HTML5 - 新的属性语法

HTML5 标准允许 4 种不同的属性语法。

本例演示在 <input> 标签中使用的不同语法：

类型 示例

Empty <input type="text" value="Bill Gates" disabled>

Unquoted <input type="text" value=Bill Gates>

Double-quoted <input type="text" value="Bill Gates">

Single-quoted <input type="text" value='Bill Gates'>

在 HTML5 标准中，根据对属性的需求，可能会用到所有 4 种语法。

HTML5 - 新特性

HTML5 的一些最有趣的新特性：

新的语义元素，比如 <header>, <footer>, <article>, and <section>。

新的表单控件，比如数字、日期、时间、日历和滑块。

强大的图像支持（借由 <canvas> 和 <svg>）

强大的多媒体支持（借由 <video> 和 <audio>）

强大的新 API，比如用本地存储取代 cookie。

HTML5 - 被删元素

以下 HTML 4.01 元素已从 HTML5 中删除：

<acronym>

<applet>

<basefont>

<big>

<center>

<dir>

<font>

<frame>

<frameset>

<noframes>

<strike>

<tt>

# 第58章 HTML5 浏览器支持

您可以帮助老版本浏览器处理 HTML5。

HTML5 浏览器支持

所有现代浏览器都支持 HTML5。

此外，所有浏览器，不论新旧，都会自动把未识别元素当做行内元素来处理。

正因如此，您可以帮助老式浏览器处理”未知的“ HTML 元素。

注释：您甚至可以教授石器时代的 IE6 如何处理未知的 HTML 元素。

把 HTML5 元素定义为块级元素

HTML5 定义了八个新的语义 HTML 元素。所有都是块级元素。

您可以把 CSS display 属性设置为 block，以确保老式浏览器中正确的行为：

实例

header, section, footer, aside, nav, main, article, figure {

display: block;

}

向 HTML 添加新元素

您可以通过浏览器 trick 向 HTML 添加任何新元素：

本例向 HTML 添加了一个名为 <myHero> 的新元素，并为其定义 display 样式：

实例

<!DOCTYPE html>

<html>

<head>

<title>Creating an HTML Element</title>

<script>document.createElement("myHero")</script>

<style>

myHero {

display: block;

background-color: #ddd;

padding: 50px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

<myHero>My First Hero</myHero>

</body>

</html>

已添加的 JavaScript 语句 document.createElement("myHero")，仅适用于 IE。

Internet Explorer 的问题

上述方案可用于所有新的 HTML5 元素，但是：

注意：Internet Explorer 8 以及更早的版本，不允许对未知元素添加样式。

幸运的是，Sjoerd Visscher 创造了 "HTML5 Enabling JavaScript", "the shiv"：

<!--[if lt IE 9]>

<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>

<![endif]-->

以上代码是一段注释，但是 IE9 的早期版本会读取它（并理解它）。

完整的 Shiv 解决方案

实例

<!DOCTYPE html>

<html>

<head>

<title>Styling HTML5</title>

<!--[if lt IE 9]>

<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>

<![endif]-->

</head>

<body>

<h1>My First Article</h1>

<article>

London is the capital city of England.

It is the most populous city in the United Kingdom,

with a metropolitan area of over 13 million inhabitants.

</article>

</body>

</html>

引用 shiv 代码的链接必须位于 <head> 元素中，因为 Internet Explorer 需要在读取之前认识所有新元素。

HTML5 Skeleton

实例

<!DOCTYPE html>

<html lang="en">

<head>

<title>HTML5 Skeleton</title>

<meta charset="utf-8">

<!--[if lt IE 9]>

<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js">

</script>

<![endif]-->

<style>

body {font-family: Verdana, sans-serif; font-size:0.8em;}

header,nav, section,article,footer

{border:1px solid grey; margin:5px; padding:8px;}

nav ul {margin:0; padding:0;}

nav ul li {display:inline; margin:5px;}

</style>

</head>

<body>

<header>

<h1>HTML5 SKeleton</h1>

</header>

<nav>

<ul>

<li><a href="html5\_semantic\_elements.asp">HTML5 Semantic</a></li>

<li><a href="html5\_geolocation.asp">HTML5 Geolocation</a></li>

<li><a href="html5\_canvas.asp">HTML5 Graphics</a></li>

</ul>

</nav>

<section>

<h1>Famous Cities</h1>

<article>

<h2>London</h2>

<p>London is the capital city of England. It is the most populous city in the United Kingdom,

with a metropolitan area of over 13 million inhabitants.</p>

</article>

<article>

<h2>Paris</h2>

<p>Paris is the capital and most populous city of France.</p>

</article>

<article>

<h2>Tokyo</h2>

<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,

and the most populous metropolitan area in the world.</p>

</article>

</section>

<footer>

<p>© 2014 W3Schools. All rights reserved.</p>

</footer>

</body>

</html>

# 第59章 HTML5 新元素

HTML5 中的新元素

下面列出的 HTML5 的新元素，以及对它们的描述。

## 59.1 新的语义/结构元素

HTML5 提供的新元素可以构建更好的文档结构：

|  |  |
| --- | --- |
| 标签 | 描述 |
| <article> | 定义文档内的文章。 |
| <aside> | 定义页面内容之外的内容。 |
| <bdi> | 定义与其他文本不同的文本方向。 |
| <details> | 定义用户可查看或隐藏的额外细节。 |
| <dialog> | 定义对话框或窗口。 |
| <figcaption> | 定义 <figure> 元素的标题。 |
| <figure> | 定义自包含内容，比如图示、图表、照片、代码清单等等。 |
| <footer> | 定义文档或节的页脚。 |
| <header> | 定义文档或节的页眉。 |
| <main> | 定义文档的主内容。 |
| <mark> | 定义重要或强调的内容。 |
| <menuitem> | 定义用户能够从弹出菜单调用的命令/菜单项目。 |
| <meter> | 定义已知范围（尺度）内的标量测量。 |
| <nav> | 定义文档内的导航链接。 |
| <progress> | 定义任务进度。 |
| <rp> | 定义在不支持 ruby 注释的浏览器中显示什么。 |
| <rt> | 定义关于字符的解释/发音（用于东亚字体）。 |
| <ruby> | 定义 ruby 注释（用于东亚字体）。 |
| <section> | 定义文档中的节。 |
| <summary> | 定义 <details> 元素的可见标题。 |
| <time> | 定义日期/时间。 |
| <wbr> | 定义可能的折行（line-break）。 |

## 59.2 新的表单元素

|  |  |
| --- | --- |
| 标签 | 描述 |
| <datalist> | 定义输入控件的预定义选项。 |
| <keygen> | 定义键对生成器字段（用于表单）。 |
| <output> | 定义计算结果。 |

## 59.3 新的输入类型

|  |  |
| --- | --- |
| 新的输入类型 | 新的输入属性 |
| color  date  datetime  datetime-local  email  month  number  range  search  tel  time  url  week | autocomplete  autofocus  form  formaction  formenctype  formmethod  formnovalidate  formtarget  height 和 width  list  min 和 max  multiple  pattern (regexp)  placeholder  required  step |

## 59.4 HTML5 - 新的属性语法

HTML5 允许四种不同的属性语法。

该例演示 <input> 标签中使用的不同语法：

|  |  |
| --- | --- |
| 标签 | 描述 |
| Empty | <input type="text" value="Bill Gates" disabled> |
| Unquoted | <input type="text" value=Bill> |
| Double-quoted | <input type="text" value="Bill Gates"> |
| Single-quoted | <input type="text" value='Bill Gates'> |

在 HTML5 中，根据属性所需，可能会使用所有这四种语法。

## 59.5 HTML5 图像

|  |  |
| --- | --- |
| 标签 | 描述 |
| <canvas> | 定义使用 JavaScript 的图像绘制。 |
| <svg> | 定义使用 SVG 的图像绘制。 |
|  |  |

## 59.6 新的媒介元素

|  |  |
| --- | --- |
| 标签 | 描述 |
| <audio> | 定义声音或音乐内容。 |
| <embed> | 定义外部应用程序的容器（比如插件）。 |
| <source> | 定义 <video> 和 <audio> 的来源。 |
| <track> | 定义 <video> 和 <audio> 的轨道。 |
| <video> | 定义视频或影片内容。 |

# 第60章 HTML5 迁移

从 HTML4 迁移至 HTML5

本章讲解如何从一张典型的 HTML4 页面迁移至典型的 HTML5。

本章演示如何把一张已有的 HTML4 页面转换为 HTML5 页面，在不破坏如何原始内容和结构的情况下。

注释：您可以使用相同的技巧从 HTML4 以及 XHTML 迁移至 HTML5。

## 60.1 典型的 HTML4 页面

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

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

<html lang="en">

<head>

<meta http-equiv="Content-Type" content="text/html;charset=utf-8">

<title>HTML4</title>

<style>

body {font-family:Verdana,sans-serif;font-size:0.8em;}

div#header,div#footer,div#content,div#post

{border:1px solid grey;margin:5px;margin-bottom:15px;padding:8px;background-color:white;}

div#header,div#footer {color:white;background-color:#444;margin-bottom:5px;}

div#content {background-color:#ddd;}

div#menu ul {margin:0;padding:0;}

div#menu ul li {display:inline; margin:5px;}

</style>

</head>

<body>

<div id="header">

<h1>Monday Times</h1>

</div>

<div id="menu">

<ul>

<li>News</li>

<li>Sports</li>

<li>Weather</li>

</ul>

</div>

<div id="content">

<h2>News Section</h2>

<div id="post">

<h2>News Article</h2>

<p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum turum ipsum

lurum hurum turum.</p>

<p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum turum ipsum

lurum hurum turum.</p>

</div>

<div id="post">

<h2>News Article</h2>

<p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum turum ipsum

lurum hurum turum.</p>

<p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum turum ipsum

lurum hurum turum.</p>

</div>

</div>

<div id="footer">

<p>&copy; 2014 Monday Times. All rights reserved.</p>

</div>

</body>

</html>

## 60.2 更改为 HTML5 Doctype

修改文档类型，从 HTML4 doctype：

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

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

修改为 HTML5 doctype：

<!DOCTYPE html>

## 60.3 更改为 HTML5 编码

修改编码信息，从 HTML4：

<meta http-equiv="Content-Type" content="text/html;charset=utf-8">

改为 HTML5：

<meta charset="utf-8">

## 60.4 添加 shiv

所有现代浏览器都支持 HTML5 语义元素。

此外，您可以“教授”老式浏览器如何处理“未知元素”。

为 Internet Explorer 支持而添加的 shiv：

<!--[if lt IE 9]>

<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>

<![endif]-->

注释：请在 HTML5 浏览器支持中阅读更多有关 shiv 的知识。

## 60.5 为 HTML5 语义元素添加 CSS

请看已有的 CSS 样式：

div#header,div#footer,div#content,div#post {

border:1px solid grey;margin:5px;margin-bottom:15px;padding:8px;background-color:white;

}

div#header,div#footer {

color:white;background-color:#444;margin-bottom:5px;

}

div#content {

background-color:#ddd;

}

div#menu ul {

margin:0;padding:0;

}

div#menu ul li {

display:inline; margin:5px;

}

Duplicate with equal CSS styles for HTML5 semantic elements:

header,footer,section,article {

border:1px solid grey;margin:5px;margin-bottom:15px;padding:8px;background-color:white;

}

header,footer {

color:white;background-color:#444;margin-bottom:5px;

}

section {

background-color:#ddd;

}

nav ul {

margin:0;padding:0;

}

nav ul li {

display:inline; margin:5px;

}

## 60.6 更改为 HTML5 <header> 和 <footer>

把 id="header" 和 id="footer" 的 <div> 元素：

<div id="header">

<h1>Monday Times</h1>

</div>

.

.

.

<div id="footer">

<p>&copy; 2014 W3Schools. All rights reserved.</p>

</div>

修改为 HTML5 语义元素 <header> 和 <footer>：

<header>

<h1>Monday Times</h1>

</header>

.

.

.

<footer>

<p>© 2014 W3Schools. All rights reserved.</p>

</footer>

## 60.7 更改为 HTML5 <nav>

把 id="menu" 的 <div> 元素：

<div id="menu">

<ul>

<li>News</li>

<li>Sports</a></li>

<li>Weather</li>

</ul>

</div>

修改为 HTML5 语义元素 <nav>：

<nav>

<ul>

<li>News</li>

<li>Sports</a></li>

<li>Weather</li>

</ul>

</nav>

## 60.8 更改为 HTML5 <section>

把 id="content" 的 the <div> 元素：

<div id="content">

.

.

.

</div>

修改为 HTML5 语义元素 <section>：

<section>

.

.

.

</section>

## 60.9 更改为 HTML5 <article>

把 class="post" 的所有 <div> 元素：

<div class="post">

<h2>News Article</h2>

<p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum turum ipsum

lurum hurum turum.</p>

</div>

修改为 HTML5 语义元素 <article>：

<article>

<h2>News Article</h2>

<p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum turum ipsum

lurum hurum turum.</p>

</article>

删除这些“不再需要的”样式：

div#header,div#footer,div#content,div#post {

border:1px solid grey;margin:5px;margin-bottom:15px;padding:8px;background-color:white;

}

div#header,div#footer {

color:white;background-color:#444;margin-bottom:5px;

}

div#content {

background-color:#ddd;

}

div#menu ul {

margin:0;padding:0;

}

div#menu ul li {

display:inline; margin:5px;

}

## 60.10 典型的 HTML5 页面

最后您可以删除 <head> 标签。HTML5 中不再需要它们：

实例

<!DOCTYPE html>

<html lang="en">

<title>HTML5</title>

<meta charset="utf-8">

<!--[if lt IE 9]>

<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js">

</script>

<![endif]-->

<style>

body {

font-family:Verdana,sans-serif;font-size:0.8em;

}

header,footer,section,article {

border:1px solid grey;

margin:5px;margin-bottom:15px;padding:8px;

background-color:white;

}

header,footer {

color:white;background-color:#444;margin-bottom:5px;

}

section {

background-color:#ddd;

}

nav ul {

margin:0;padding:0;

}

nav ul li {

display:inline; margin:5px;

}

</style>

<body>

<header>

<h1>Monday Times</h1>

</header>

<nav>

<ul>

<li>News</li>

<li>Sports</li>

<li>Weather</li>

</ul>

</nav>

<section>

<h2>News Section</h2>

<div id="post">

<h2>News Article</h2>

<p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum turum ipsum

lurum hurum turum.</p>

<p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum turum ipsum

lurum hurum turum.</p>

<p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum turum ipsum

lurum hurum turum.</p>

</div>

<div id="post">

<h2>News Article</h2>

<p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum turum ipsum

lurum hurum turum.</p>

<p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum turum ipsum

lurum hurum turum.</p>

<p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum turum ipsum

lurum hurum turum.</p>

</div>

</section>

<footer>

<p>&copy; 2014 Monday Times. All rights reserved.</p>

</footer>

</body>

</html>

## 60.11 <article> <section> 与 <div> 之间的差异

在 HTML5 标准中，<article> <section> 与 <div> 之间的差异很小，令人困惑。

在 HTML5 标准中，<section> 元素被定位为相关元素的块。

<article> 元素被定义为相关元素的完整的自包含块。

<div> 元素被定义为子元素的块。

如何理解呢？

在上面的例子中，我们曾使用 <section> 作为相关 <articles> 的容器。

但是，我们也能够把 <article> 用作文章的容器。

下面是一些不同的例子：

<article> 中的 <article>：

<article>

<h2>Famous Cities</h2>

<article>

<h2>London</h2>

<p>London is the capital city of England. It is the most populous city in the United Kingdom,

with a metropolitan area of over 13 million inhabitants.</p>

</article>

<article>

<h2>Paris</h2>

<p>Paris is the capital and most populous city of France.</p>

</article>

<article>

<h2>Tokyo</h2>

<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,

and the most populous metropolitan area in the world.</p>

</article>

</article>

<article> 中的 <div>：

<article>

<h2>Famous Cities</h2>

<div class="city">

<h2>London</h2>

<p>London is the capital city of England. It is the most populous city in the United Kingdom,

with a metropolitan area of over 13 million inhabitants.</p>

</div>

<div class="city">

<h2>Paris</h2>

<p>Paris is the capital and most populous city of France.</p>

</div>

<div class="city">

<h2>Tokyo</h2>

<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,

and the most populous metropolitan area in the world.</p>

</div>

</article>

<article> 中的 <section> 中的 <div>：

<article>

<section>

<h2>Famous Cities</h2>

<div class="city">

<h2>London</h2>

<p>London is the capital city of England. It is the most populous city in the United Kingdom,

with a metropolitan area of over 13 million inhabitants.</p>

</div>

<div class="city">

<h2>Paris</h2>

<p>Paris is the capital and most populous city of France.</p>

</div>

<div class="city">

<h2>Tokyo</h2>

<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,

and the most populous metropolitan area in the world.</p>

</div>

</section>

<section>

<h2>Famous Countries</h2>

<div class="country">

<h2>England</h2>

<p>London is the capital city of England. It is the most populous city in the United Kingdom,

with a metropolitan area of over 13 million inhabitants.</p>

</div>

<div class="country">

<h2>France</h2>

<p>Paris is the capital and most populous city of France.</p>

</div>

<div class="country">

<h2>Japan</h2>

<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,

and the most populous metropolitan area in the world.</p>

</div>

</section>

</article>

# 第61章 HTML5 地理定位

HTML5 Geolocation（地理定位）用于定位用户的位置。

<!DOCTYPE html>

<html>

<body>

<p id="demo">点击这个按钮，获得您的位置：</p>

<button onclick="getLocation()">试一下</button>

<div id="mapholder"></div>

<script src="http://maps.google.com/maps/api/js?sensor=false"></script>

<script>

var x=document.getElementById("demo");

function getLocation()

{

if (navigator.geolocation)

{

navigator.geolocation.getCurrentPosition(showPosition,showError);

}

else{x.innerHTML="Geolocation is not supported by this browser.";}

}

function showPosition(position)

{

lat=position.coords.latitude;

lon=position.coords.longitude;

latlon=new google.maps.LatLng(lat, lon)

mapholder=document.getElementById('mapholder')

mapholder.style.height='250px';

mapholder.style.width='500px';

var myOptions={

center:latlon,zoom:14,

mapTypeId:google.maps.MapTypeId.ROADMAP,

mapTypeControl:false,

navigationControlOptions:{style:google.maps.NavigationControlStyle.SMALL}

};

var map=new google.maps.Map(document.getElementById("mapholder"),myOptions);

var marker=new google.maps.Marker({position:latlon,map:map,title:"You are here!"});

}

function showError(error)

{

switch(error.code)

{

case error.PERMISSION\_DENIED:

x.innerHTML="User denied the request for Geolocation."

break;

case error.POSITION\_UNAVAILABLE:

x.innerHTML="Location information is unavailable."

break;

case error.TIMEOUT:

x.innerHTML="The request to get user location timed out."

break;

case error.UNKNOWN\_ERROR:

x.innerHTML="An unknown error occurred."

break;

}

}

</script>

</body>

</html>

## 61.1 定位用户的位置

HTML5 Geolocation API 用于获得用户的地理位置。

鉴于该特性可能侵犯用户的隐私，除非用户同意，否则用户位置信息是不可用的。

## 61.2 浏览器支持

Internet Explorer 9、Firefox、Chrome、Safari 以及 Opera 支持地理定位。

注释：对于拥有 GPS 的设备，比如 iPhone，地理定位更加精确。

## 61.3 HTML5 - 使用地理定位

请使用 getCurrentPosition() 方法来获得用户的位置。

下例是一个简单的地理定位实例，可返回用户位置的经度和纬度。

实例

<script>

var x=document.getElementById("demo");

function getLocation()

{

if (navigator.geolocation)

{

navigator.geolocation.getCurrentPosition(showPosition);

}

else{x.innerHTML="Geolocation is not supported by this browser.";}

}

function showPosition(position)

{

x.innerHTML="Latitude: " + position.coords.latitude +

"<br />Longitude: " + position.coords.longitude;

}

</script>

## 61.4 例子解释：

检测是否支持地理定位

如果支持，则运行 getCurrentPosition() 方法。如果不支持，则向用户显示一段消息。

如果getCurrentPosition()运行成功，则向参数showPosition中规定的函数返回一个coordinates对象

showPosition() 函数获得并显示经度和纬度

上面的例子是一个非常基础的地理定位脚本，不含错误处理。

处理错误和拒绝

getCurrentPosition() 方法的第二个参数用于处理错误。它规定当获取用户位置失败时运行的函数：

实例

function showError(error)

{

switch(error.code)

{

case error.PERMISSION\_DENIED:

x.innerHTML="User denied the request for Geolocation."

break;

case error.POSITION\_UNAVAILABLE:

x.innerHTML="Location information is unavailable."

break;

case error.TIMEOUT:

x.innerHTML="The request to get user location timed out."

break;

case error.UNKNOWN\_ERROR:

x.innerHTML="An unknown error occurred."

break;

}

}

错误代码：

Permission denied - 用户不允许地理定位

Position unavailable - 无法获取当前位置

Timeout - 操作超时

## 61.5 在地图中显示结果

如需在地图中显示结果，您需要访问可使用经纬度的地图服务，比如谷歌地图或百度地图：

实例

function showPosition(position)

{

var latlon=position.coords.latitude+","+position.coords.longitude;

var img\_url="http://maps.googleapis.com/maps/api/staticmap?center="

+latlon+"&zoom=14&size=400x300&sensor=false";

document.getElementById("mapholder").innerHTML="<img src='"+img\_url+"' />";

}

在上例中，我们使用返回的经纬度数据在谷歌地图中显示位置（使用静态图像）。

上面的链接向您演示如何使用脚本来显示带有标记、缩放和拖曳选项的交互式地图。

## 61.6 给定位置的信息

本页演示的是如何在地图上显示用户的位置。不过，地理定位对于给定位置的信息同样很有用处。

案例：

更新本地信息

显示用户周围的兴趣点

交互式车载导航系统 (GPS)

## 61.7 getCurrentPosition() 方法 - 返回数据

若成功，则 getCurrentPosition() 方法返回对象。始终会返回 latitude、longitude 以及 accuracy 属性。如果可用，则会返回其他下面的属性。

|  |  |
| --- | --- |
| 属性 | 描述 |
| coords.latitude | 十进制数的纬度 |
| coords.longitude | 十进制数的经度 |
| coords.accuracy | 位置精度 |
| coords.altitude | 海拔，海平面以上以米计 |
| coords.altitudeAccuracy | 位置的海拔精度 |
| coords.heading | 方向，从正北开始以度计 |
| coords.speed | 速度，以米/每秒计 |
| timestamp | 响应的日期/时间 |

## 61.8 Geolocation 对象 - 其他有趣的方法

watchPosition() - 返回用户的当前位置，并继续返回用户移动时的更新位置（就像汽车上的 GPS）。

clearWatch() - 停止 watchPosition() 方法

下面的例子展示 watchPosition() 方法。您需要一台精确的 GPS 设备来测试该例（比如 iPhone）：

实例

<script>

var x=document.getElementById("demo");

function getLocation()

{

if (navigator.geolocation)

{

navigator.geolocation.watchPosition(showPosition);

}

else{x.innerHTML="Geolocation is not supported by this browser.";}

}

function showPosition(position)

{

x.innerHTML="Latitude: " + position.coords.latitude +

"<br />Longitude: " + position.coords.longitude;

}

</script>

# 第62章 HTML5 拖放

## 62.1 拖放

拖放（Drag 和 Drop）是很常见的特性。它指的是您抓取某物并拖入不同的位置。

拖放是 HTML5 标准的组成部分：任何元素都是可拖放的。

## 62.2 HTML 拖放实例

下列是关于拖放的简单例子：

实例

<!DOCTYPE HTML>

<html>

<head>

<script>

function allowDrop(ev) {

ev.preventDefault();

}

function drag(ev) {

ev.dataTransfer.setData("text", ev.target.id);

}

function drop(ev) {

ev.preventDefault();

var data = ev.dataTransfer.getData("text");

ev.target.appendChild(document.getElementById(data));

}

</script>

</head>

<body>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>

<img id="drag1" src="img\_logo.gif" draggable="true" ondragstart="drag(event)" width="336" height="69">

</body>

</html>

它也许看上去有点复杂，不过让我们研究一下拖放事件的所有不同部分。

## 62.3 把元素设置为可拖放

首先：为了把一个元素设置为可拖放，请把 draggable 属性设置为 true：

<img draggable="true">

## 62.4 拖放的内容 - ondragstart 和 setData()

然后，规定当元素被拖动时发生的事情。

在上面的例子中，ondragstart 属性调用了一个 drag(event) 函数，规定拖动什么数据。

dataTransfer.setData() 方法设置被拖动数据的数据类型和值：

function drag(ev) {

ev.dataTransfer.setData("text", ev.target.id);

}

在本例中，数据类型是 "text"，而值是这个可拖动元素的 id ("drag1")。

## 62.5 拖到何处 - ondragover

ondragover 事件规定被拖动的数据能够被放置到何处。

默认地，数据/元素无法被放置到其他元素中。为了实现拖放，我们必须阻止元素的这种默认的处理方式。

这个任务由 ondragover 事件的 event.preventDefault() 方法完成：

event.preventDefault()

## 62.6 进行放置 - ondrop

当放开被拖数据时，会发生 drop 事件。

在上面的例子中，ondrop 属性调用了一个函数，drop(event)：

function drop(ev) {

ev.preventDefault();

var data = ev.dataTransfer.getData("text");

ev.target.appendChild(document.getElementById(data));

}

代码解释：

调用 preventDefault() 来阻止数据的浏览器默认处理方式（drop 事件的默认行为是以链接形式打开）

通过 dataTransfer.getData() 方法获得被拖的数据。该方法将返回在 setData() 方法中设置为相同类型的任何数据

被拖数据是被拖元素的 id ("drag1")

把被拖元素追加到放置元素中

## 62.7 更多实例

来回拖放图片

如何在两个 <div> 元素之间来回拖放图像：

<!DOCTYPE HTML>

<html>

<head>

<style type="text/css">

#div1, #div2

{float:left; width:198px; height:66px; margin:10px;padding:10px;border:1px solid #aaaaaa;}

</style>

<script type="text/javascript">

function allowDrop(ev)

{

ev.preventDefault();

}

function drag(ev)

{

ev.dataTransfer.setData("Text",ev.target.id);

}

function drop(ev)

{

ev.preventDefault();

var data=ev.dataTransfer.getData("Text");

ev.target.appendChild(document.getElementById(data));

}

</script>

</head>

<body>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)">

<img src="/i/eg\_dragdrop\_w3school.gif" draggable="true" ondragstart="drag(event)" id="drag1" />

</div>

<div id="div2" ondrop="drop(event)" ondragover="allowDrop(event)"></div>

</body>

</html>

# 第63章 HTML 本地存储

HTML 本地存储：优于 cookies。

## 63. 1 什么是 HTML 本地存储？

通过本地存储（Local Storage），web 应用程序能够在用户浏览器中对数据进行本地的存储。

在 HTML5 之前，应用程序数据只能存储在 cookie 中，包括每个服务器请求。本地存储则更安全，并且可在不影响网站性能的前提下将大量数据存储于本地。

与 cookie 不同，存储限制要大得多（至少5MB），并且信息不会被传输到服务器。

本地存储经由起源地（origin）（经由域和协议）。所有页面，从起源地，能够存储和访问相同的数据。

## 63.2 HTML 本地存储对象

HTML 本地存储提供了两个在客户端存储数据的对象：

window.localStorage - 存储没有截止日期的数据

window.sessionStorage - 针对一个 session 来存储数据（当关闭浏览器标签页时数据会丢失）

在使用本地存储时，请检测 localStorage 和 sessionStorage 的浏览器支持：

if (typeof(Storage) !== "undefined") {

// 针对 localStorage/sessionStorage 的代码

} else {

// 抱歉！不支持 Web Storage ..

}

## 63.3 localStorage 对象

localStorage 对象存储的是没有截止日期的数据。当浏览器被关闭时数据不会被删除，在下一天、周或年中，都是可用的。

实例

// 存储

localStorage.setItem("lastname", "Gates");

// 取回

document.getElementById("result").innerHTML = localStorage.getItem("lastname");

实例解释：

创建 localStorage 名称/值对，其中：name="lastname"，value="Gates"

取回 "lastname" 的值，并把它插到 id="result" 的元素中

上例也可这样写：

// 存储

localStorage.lastname = "Gates";

// 取回

document.getElementById("result").innerHTML = localStorage.lastname;

删除 "lastname" localStorage 项目的语法如下：

localStorage.removeItem("lastname");

注释：名称/值对始终存储为字符串。如果需要请记得把它们转换为其他格式！

下面的例子对用户点击按钮的次数进行计数。在代码中，值字符串被转换为数值，依次对计数进行递增：

实例

if (localStorage.clickcount) {

localStorage.clickcount = Number(localStorage.clickcount) + 1;

} else {

localStorage.clickcount = 1;

}

document.getElementById("result").innerHTML = "您已经点击这个按钮 " +

localStorage.clickcount + " 次。";

## 63.4 sessionStorage 对象

sessionStorage 对象等同 localStorage 对象，不同之处在于只对一个 session 存储数据。如果用户关闭具体的浏览器标签页，数据也会被删除。

下例在当前 session 中对用户点击按钮进行计数：

实例

if (sessionStorage.clickcount) {

sessionStorage.clickcount = Number(sessionStorage.clickcount) + 1;

} else {

sessionStorage.clickcount = 1;

}

document.getElementById("result").innerHTML = "在本 session 中，您已经点击这个按钮 " +

sessionStorage.clickcount + " 次。";

# 第64章 HTML5 应用程序缓存

使用应用程序缓存，通过创建 cache manifest 文件，可轻松创建 web 应用的离线版本。

## 64.1 什么是应用程序缓存？

HTML5 引入了应用程序缓存（Application Cache），这意味着可对 web 应用进行缓存，并可在没有因特网连接时进行访问。

应用程序缓存为应用带来三个优势：

1.离线浏览 - 用户可在应用离线时使用它们

2.速度 - 已缓存资源加载得更快

3.减少服务器负载 - 浏览器将只从服务器下载更新过或更改过的资源

## 64.2 HTML Cache Manifest 实例

下例展示了带有 cache manifest 的 HTML 文档（供离线浏览）：

实例

<!DOCTYPE HTML>

<html manifest="demo.appcache">

<body>

文档内容 ......

</body>

</html>

## 64.3 Cache Manifest 基础

如需启用应用程序缓存，请在文档的 <html> 标签中包含 manifest 属性：

<!DOCTYPE HTML>

<html manifest="demo.appcache">

...

</html>

每个指定了 manifest 的页面在用户对其访问时都会被缓存。如果未指定 manifest 属性，则页面不会被缓存（除非在 manifest 文件中直接指定了该页面）。

manifest 文件的建议文件扩展名是：".appcache"。

注意：manifest 文件需要设置正确的 MIME-type，即 "text/cache-manifest"。必须在 web 服务器上进行配置。

## 64.4 Manifest 文件

manifest 文件是简单的文本文件，它告知浏览器被缓存的内容（以及不缓存的内容）。

manifest 文件有三个部分：

CACHE MANIFEST - 在此标题下列出的文件将在首次下载后进行缓存

NETWORK - 在此标题下列出的文件需要与服务器的连接，且不会被缓存

FALLBACK - 在此标题下列出的文件规定当页面无法访问时的回退页面（比如 404 页面）

## 64.5 CACHE MANIFEST

第一行，CACHE MANIFEST，是必需的：

CACHE MANIFEST

/theme.css

/logo.gif

/main.js

上面的 manifest 文件列出了三个资源：一个 CSS 文件，一个 GIF 图像，以及一个 JavaScript 文件。当 manifest 文件被加载后，浏览器会从网站的根目录下载这三个文件。然后，无论用户何时与因特网断开连接，这些资源依然可用。

## 64.6 NETWORK

下面的 NETWORK 部分规定文件 "login.php" 永远不会被缓存，且离线时是不可用的：

NETWORK:

login.asp

可以使用星号来指示所有其他其他资源/文件都需要因特网连接：

NETWORK:

\*

FALLBACK

下面的 FALLBACK 部分规定如果无法建立因特网连接，则用 "offline.html" 替代 /html/ 目录中的所有文件：

FALLBACK:

/html/ /offline.html

注释：第一个 URI 是资源，第二个是替补。

## 64.7 更新缓存

一旦应用被缓存，它就会保持缓存直到发生下列情况：

1.用户清空浏览器缓存

2.manifest 文件被修改（参阅下面的提示）

3.由程序来更新应用缓存

## 64.8 实例 - 完整的 Cache Manifest 文件

CACHE MANIFEST

# 2012-02-21 v1.0.0

/theme.css

/logo.gif

/main.js

NETWORK:

login.asp

FALLBACK:

/html/ /offline.html

提示：以 "#" 开头的是注释行，但也可满足其他用途。应用的缓存只会在其 manifest 文件改变时被更新。如果您编辑了一幅图像，或者修改了一个 JavaScript 函数，这些改变都不会被重新缓存。更新注释行中的日期和版本号是一种使浏览器重新缓存文件的办法。

## 64.9 关于应用程序缓存的注意事项

请留心缓存的内容。

一旦文件被缓存，则浏览器会继续展示已缓存的版本，即使您修改了服务器上的文件。为了确保浏览器更新缓存，您需要更新 manifest 文件。

注释：浏览器对缓存数据的容量限制可能不太一样（某些浏览器的限制是每个站点 5MB）。

# 第65章 HTML Web Workers

Web worker 是运行在后台的 JavaScript，不会影响页面的性能。

## 65.1 什么是 Web Worker？

当在 HTML 页面中执行脚本时，页面是不可响应的，直到脚本已完成。

Web worker 是运行在后台的 JavaScript，独立于其他脚本，不会影响页面的性能。您可以继续做任何愿意做的事情：点击、选取内容等等，而此时 web worker 运行在后台。

## 65.2 HTML Web Workers 实例

下面的例子创建了一个简单的 web worker，在后台计数：

计数：

启动 Worker 停止 Worker

<!DOCTYPE html>

<html>

<body>

<p>计数: <output id="result"></output></p>

<button onclick="startWorker()">开始 Worker</button>

<button onclick="stopWorker()">停止 Worker</button>

<br /><br />

<script>

var w;

function startWorker()

{

if(typeof(Worker)!=="undefined")

{

if(typeof(w)=="undefined")

{

w=new Worker("/example/html5/demo\_workers.js");

}

w.onmessage = function (event) {

document.getElementById("result").innerHTML=event.data;

};

}

else

{

document.getElementById("result").innerHTML="Sorry, your browser does not support Web Workers...";

}

}

function stopWorker()

{

w.terminate();

}

</script>

</body>

</html>

## 65.3 检测 Web Worker 支持

在创建 web worker 之前，请检测用户浏览器是否支持它：

if (typeof(Worker) !== "undefined") {

// 是的！支持 Web worker！

// 一些代码.....

} else {

// 抱歉！不支持 Web Worker！

}

## 65.4 创建 Web Worker 文件

现在，让我们在一个外部 JavaScript 文件中创建我们的 web worker。

在此处，我们创建了计数脚本。该脚本存储于 "demo\_workers.js" 文件中：

var i = 0;

function timedCount() {

i = i + 1;

postMessage(i);

setTimeout("timedCount()",500);

}

timedCount();

以上代码中重要的部分是 postMessage() 方法 - 它用于向 HTML 页面传回一段消息。

注释: web worker 通常不用于如此简单的脚本，而是用于更耗费 CPU 资源的任务。

## 65.5 创建 Web Worker 对象

现在我们已经有了 web worker 文件，我们需要从 HTML 页面调用它。

下面的代码行检测是否存在 worker，如果不存在，- 它会创建一个新的 web worker 对象，然后运行 "demo\_workers.js" 中的代码：

if (typeof(w) == "undefined") {

w = new Worker("demo\_workers.js");

}

然后我们就可以从 web worker 发生和接收消息了。

向 web worker 添加一个 "onmessage" 事件监听器：

w.onmessage = function(event){

document.getElementById("result").innerHTML = event.data;

};

当 web worker 传送消息时，会执行事件监听器中的代码。来自 web worker 的数据会存储于 event.data 中。

## 65.6 终止 Web Worker

当创建 web worker 后，它会继续监听消息（即使在外部脚本完成后）直到其被终止为止。

如需终止 web worker，并释放浏览器/计算机资源，请使用 terminate() 方法：

w.terminate();

## 65.7 复用 Web Worker

如果您把 worker 变量设置为 undefined，在其被终止后，可以重复使用该代码：

w = undefined;

## 65.8 完整的 Web Worker 实例代码

我们已经看到了 .js 文件中的 Worker 代码。下面是 HTML 页面的代码：

实例

<!DOCTYPE html>

<html>

<body>

<p>Count numbers: <output id="result"></output></p>

<button onclick="startWorker()">Start Worker</button>

<button onclick="stopWorker()">Stop Worker</button>

<br><br>

<script>

var w;

function startWorker() {

if(typeof(Worker) !== "undefined") {

if(typeof(w) == "undefined") {

w = new Worker("demo\_workers.js");

}

w.onmessage = function(event) {

document.getElementById("result").innerHTML = event.data;

};

} else {

document.getElementById("result").innerHTML = "Sorry! No Web Worker support.";

}

}

function stopWorker() {

w.terminate();

w = undefined;

}

</script>

</body>

</html>

## 65.9 Web Worker 和 DOM

由于 web worker 位于外部文件中，它们无法访问下例 JavaScript 对象：

1.window 对象

2.document 对象

3.parent 对象

# 第66章 HTML Server-Sent 事件

Server-Sent 事件允许网页从服务器获得更新。

## 66.1 Server-Sent 事件 - One Way Messaging

Server-Sent 事件指的是网页自动从服务器获得更新。

以前也可能做到这一点，前提是网页不得不询问是否有可用的更新。通过 Server-Sent 事件，更新能够自动到达。

例如：Facebook/Twitter 更新、股价更新、新的博文、赛事结果，等等。

## 66.2 接收 Server-Sent 事件通知

EventSource 对象用于接收服务器发送事件通知：

实例

<!DOCTYPE html>

<html>

<body>

<h1>获得服务器更新</h1>

<div id="result"></div>

<script>

if(typeof(EventSource)!=="undefined")

{

var source=new EventSource("/example/html5/demo\_sse.php");

source.onmessage=function(event)

{

document.getElementById("result").innerHTML+=event.data + "<br />";

};

}

else

{

document.getElementById("result").innerHTML="抱歉，您的浏览器不支持 server-sent 事件 ...";

}

</script>

</body>

</html>

例子解释：

创建一个新的 EventSource 对象，然后规定发送更新的页面的 URL（本例中是 "demo\_sse.php"）

每当接收到一次更新，就会发生 onmessage 事件

当 onmessage 事件发生时，把已接收的数据推入 id 为 "result" 的元素中

## 66.3 检测 Server-Sent 事件支持

在 TIY 实例中，我们编写了一段额外的代码来检测服务器发送事件的浏览器支持：

if(typeof(EventSource) !== "undefined") {

// 是的！支持服务器发送事件！

// 一些代码.....

} else {

// 抱歉！不支持服务器发送事件！

}

## 66.4 服务器端代码实例

为了使上例运行，您需要能够发送数据更新的服务器（比如 PHP 或 ASP）。

服务器端事件流的语法非常简单。请把 "Content-Type" 报头设置为 "text/event-stream"。现在，您可以开始发送事件流了。

PHP 中的代码 (demo\_sse.php)：

<?php

header('Content-Type: text/event-stream');

header('Cache-Control: no-cache');

$time = date('r');

echo "data: The server time is: {$time}\n\n";

flush();

?>

ASP 中的代码 (VB) (demo\_sse.asp)：

<%

Response.ContentType = "text/event-stream"

Response.Expires = -1

Response.Write("data: The server time is: " & now())

Response.Flush()

%>

代码解释：

1.把报头 "Content-Type" 设置为 "text/event-stream"

2.规定不对页面进行缓存

3.输出要发送的日期（始终以 "data: " 开头）

4.向网页刷新输出数据

## 66.5 EventSource 对象

在上例中，我们使用 onmessage 事件来获取消息。不过还可以使用其他事件：

|  |  |
| --- | --- |
| 事件 | 描述 |
| onopen | 当通往服务器的连接被打开 |
| onmessage | 当接收到消息 |
| onerror | 当发生错误 |

# 第67章 HTML 标签参考手册

按字母顺序排列 HTML 标签

|  |  |
| --- | --- |
| 标签 | 描述 |
| <!--...--> | 定义注释。 |
| <!DOCTYPE> | 定义文档类型。 |
| <a> | 定义锚。 |
| <abbr> | 定义缩写。 |
| <acronym> | 定义只取首字母的缩写。HTML5 中不支持。请使用 <abbr> 代替。 |
| <address> | 定义文档作者或拥有者的联系信息。 |
| <applet> | 定义嵌入的 applet。HTML5 中不支持。请使用 <embed> 和 <object> 代替。 |
| <area> | 定义图像映射内部的区域。 |
| <article> | 定义文章。 |
| <aside> | 定义页面内容之外的内容。 |
| <audio> | 定义声音内容。 |
| <b> | 定义粗体字。 |
| <base> | 定义页面中所有链接的默认地址或默认目标。 |
| <basefont> | 定义页面中文本的默认字体、颜色或尺寸。HTML5 中不支持。请使用 CSS 代替。 |
| <bdi> | 定义文本的文本方向，使其脱离其周围文本的方向设置。 |
| <bdo> | 定义文字方向。 |
| <big> | 定义大号文本。HTML5 中不支持。请使用 CSS 代替。 |
| <blockquote> | 定义长的引用。 |
| <body> | 定义文档的主体。 |
| <br> | 定义简单的折行。 |
| <button> | 定义按钮 (push button)。 |
| <canvas> | 定义图形。 |
| <caption> | 定义表格标题。 |
| <center> | 定义居中文本。HTML5 中不支持。请使用 CSS 代替。 |
| <cite> | 定义引用(citation)。 |
| <code> | 定义计算机代码文本。 |
| <col> | 定义表格中一个或多个列的属性值。 |
| <colgroup> | 定义表格中供格式化的列组。 |
| <command> | 定义命令按钮。 |
| <data> | 添加给定内容的机器可读翻译。 |
| <datalist> | 定义下拉列表。 |
| <dd> | 定义定义列表中项目的描述。 |
| <del> | 定义被删除文本。 |
| <details> | 定义元素的细节。 |
| <dir> | 定义目录列表。HTML5 中不支持。请使用 CSS 代替。 |
| <div> | 定义文档中的节。 |
| <dfn> | 定义定义项目。 |
| <dialog> | 定义对话框或窗口。 |
| <dl> | 定义定义列表。 |
| <dt> | 定义定义列表中的项目。 |
| <em> | 定义强调文本。 |
| <embed> | 定义外部交互内容或插件。 |
| <fieldset> | 定义围绕表单中元素的边框。 |
| <figcaption> | 定义 figure 元素的标题。 |
| <figure> | 定义媒介内容的分组，以及它们的标题。 |
| <font> | 定义文字的字体、尺寸和颜色。HTML5 中不支持。请使用 CSS 代替。 |
| <footer> | 定义 section 或 page 的页脚。 |
| <form> | 定义供用户输入的 HTML 表单。 |
| <frame> | 定义框架集的窗口或框架。HTML5 中不支持。 |
| <frameset> | 定义框架集。HTML5 中不支持。 |
| <h1> to <h6> | 定义 HTML 标题。 |
| <head> | 定义关于文档的信息。 |
| <header> | 定义 section 或 page 的页眉。 |
| <hr> | 定义水平线。 |
| <html> | 定义 HTML 文档。 |
| <i> | 定义斜体字。 |
| <iframe> | 定义内联框架。 |
| <img> | 定义图像。 |
| <input> | 定义输入控件。 |
| <ins> | 定义被插入文本。 |
| <isindex> | 定义与文档相关的可搜索索引。不赞成使用。 |
| <kbd> | 定义键盘文本。 |
| <keygen> | 定义生成密钥。 |
| <label> | 定义 input 元素的标注。 |
| <legend> | 定义 fieldset 元素的标题。 |
| <li> | 定义列表的项目。 |
| <link> | 定义文档与外部资源的关系。 |
| <main> | 定义文档的主要内容。 |
| <map> | 定义图像映射。 |
| <mark> | 定义有记号的文本。 |
| <menu> | 定义命令的列表或菜单。 |
| <menuitem> | 定义用户可以从弹出菜单调用的命令/菜单项目。 |
| <meta> | 定义关于 HTML 文档的元信息。 |
| <meter> | 定义预定义范围内的度量。 |
| <nav> | 定义导航链接。 |
| <noframes> | 定义针对不支持框架的用户的替代内容。HTML5 中不支持。 |
| <noscript> | 定义针对不支持客户端脚本的用户的替代内容。 |
| <object> | 定义内嵌对象。 |
| <ol> | 定义有序列表。 |
| <optgroup> | 定义选择列表中相关选项的组合。 |
| <option> | 定义选择列表中的选项。 |
| <output> | 定义输出的一些类型。 |
| <p> | 定义段落。 |
| <param> | 定义对象的参数。 |
| <pre> | 定义预格式文本。 |
| <progress> | 定义任何类型的任务的进度。 |
| <q> | 定义短的引用。 |
| <rp> | 定义若浏览器不支持 ruby 元素显示的内容。 |
| <rt> | 定义 ruby 注释的解释。 |
| <ruby> | 定义 ruby 注释。 |
| <s> | 定义加删除线的文本。 |
| <samp> | 定义计算机代码样本。 |
| <script> | 定义客户端脚本。 |
| <section> | 定义 section。 |
| <select> | 定义选择列表（下拉列表）。 |
| <small> | 定义小号文本。 |
| <source> | 定义媒介源。 |
| <span> | 定义文档中的节。 |
| <strike> | 定义加删除线文本。HTML5 中不支持。请使用 <del> 或 <s> 代替。 |
| <strong> | 定义强调文本。 |
| <style> | 定义文档的样式信息。 |
| <sub> | 定义下标文本。 |
| <summary> | 为 <details> 元素定义可见的标题。 |
| <sup> | 定义上标文本。 |
| <svg> | 定义 SVG 图形的容器。 |
| <table> | 定义表格。 |
| <tbody> | 定义表格中的主体内容。 |
| <td> | 定义表格中的单元。 |
| <template> | 定义用作容纳页面加载时隐藏内容的容器。 |
| <textarea> | 定义多行的文本输入控件。 |
| <tfoot> | 定义表格中的表注内容（脚注）。 |
| <th> | 定义表格中的表头单元格。 |
| <thead> | 定义表格中的表头内容。 |
| <time> | 定义日期/时间。 |
| <title> | 定义文档的标题。 |
| <tr> | 定义表格中的行。 |
| <track> | 定义用在媒体播放器中的文本轨道。 |
| <tt> | 定义打字机文本。HTML5 中不支持。请使用 CSS 代替。 |
| <u> | 定义下划线文本。 |
| <ul> | 定义无序列表。 |
| <var> | 定义文本的变量部分。 |
| <video> | 定义视频。 |
| <wbr> | 定义可能的换行符。 |
| <xmp> | 定义预格式文本。HTML5 中不支持。 |

# 第68章 HTML 全局属性

## 68.1 HTML 全局属性

全局属性是可与所有 HTML 元素一起使用的属性。

## 68.2 HTML 全局属性

= HTML5 中添加的属性。

|  |  |
| --- | --- |
| 属性 | 描述 |
| accesskey | 规定激活元素的快捷键。 |
| class | 规定元素的一个或多个类名（引用样式表中的类）。 |
| contenteditable | 规定元素内容是否可编辑。 |
| contextmenu | 规定元素的上下文菜单。上下文菜单在用户点击元素时显示。 |
| data-\* | 用于存储页面或应用程序的私有定制数据。 |
| dir | 规定元素中内容的文本方向。 |
| draggable | 规定元素是否可拖动。 |
| dropzone | 规定在拖动被拖动数据时是否进行复制、移动或链接。 |
| hidden | 规定元素仍未或不再相关。 |
| id | 规定元素的唯一 id。 |
| lang | 规定元素内容的语言。 |
| spellcheck | 规定是否对元素进行拼写和语法检查。 |
| style | 规定元素的行内 CSS 样式。 |
| tabindex | 规定元素的 tab 键次序。 |
| title | 规定有关元素的额外信息。 |
| translate | 规定是否应该翻译元素内容。 |

# 第69章 HTML 事件参考手册

## 69.1 全局事件属性

HTML 有能力让事件触发浏览器中的动作，例如当用户单击元素时启动 JavaScript。

如需了解有关编程事件的更多信息，请访问我们的 JavaScript 教程。

以下是可添加到 HTML 元素以定义事件操作的全局事件属性。

## 69.2 Window 事件属性

针对 window 对象触发的事件（应用到 <body> 标签）：

|  |  |  |
| --- | --- | --- |
| 属性 | 值 | 描述 |
| onafterprint | script | 文档打印之后运行的脚本。 |
| onbeforeprint | script | 文档打印之前运行的脚本。 |
| onbeforeunload | script | 文档卸载之前运行的脚本。 |
| onerror | script | 在错误发生时运行的脚本。 |
| onhaschange | script | 当文档已改变时运行的脚本。 |
| onload | script | 页面结束加载之后触发。 |
| onmessage | script | 在消息被触发时运行的脚本。 |
| onoffline | script | 当文档离线时运行的脚本。 |
| ononline | script | 当文档上线时运行的脚本。 |
| onpagehide | script | 当窗口隐藏时运行的脚本。 |
| onpageshow | script | 当窗口成为可见时运行的脚本。 |
| onpopstate | script | 当窗口历史记录改变时运行的脚本。 |
| onredo | script | 当文档执行撤销（redo）时运行的脚本。 |
| onresize | script | 当浏览器窗口被调整大小时触发。 |
| onstorage | script | 在 Web Storage 区域更新后运行的脚本。 |
| onundo | script | 在文档执行 undo 时运行的脚本。 |
| onunload | script | 一旦页面已下载时触发（或者浏览器窗口已被关闭）。 |

## 69.3 Form 事件

由 HTML 表单内的动作触发的事件（应用到几乎所有 HTML 元素，但最常用在 form 元素中）：

|  |  |  |
| --- | --- | --- |
| 属性 | 值 | 描述 |
| onblur | script | 元素失去焦点时运行的脚本。 |
| onchange | script | 在元素值被改变时运行的脚本。 |
| oncontextmenu | script | 当上下文菜单被触发时运行的脚本。 |
| onfocus | script | 当元素获得焦点时运行的脚本。 |
| onformchange | script | 在表单改变时运行的脚本。 |
| onforminput | script | 当表单获得用户输入时运行的脚本。 |
| oninput | script | 当元素获得用户输入时运行的脚本。 |
| oninvalid | script | 当元素无效时运行的脚本。 |
| onreset | script | 当表单中的重置按钮被点击时触发。HTML5 中不支持。 |
| onselect | script | 在元素中文本被选中后触发。 |
| onsubmit | script | 在提交表单时触发。 |

## 69.4 Keyboard 事件

|  |  |  |
| --- | --- | --- |
| 属性 | 值 | 描述 |
| onkeydown | script | 在用户按下按键时触发。 |
| onkeypress | script | 在用户敲击按钮时触发。 |
| onkeyup | script | 当用户释放按键时触发。 |

## 69.5 Mouse 事件

由鼠标或类似用户动作触发的事件：

|  |  |  |
| --- | --- | --- |
| 属性 | 值 | 描述 |
| onclick | script | 元素上发生鼠标点击时触发。 |
| ondblclick | script | 元素上发生鼠标双击时触发。 |
| ondrag | script | 元素被拖动时运行的脚本。 |
| ondragend | script | 在拖动操作末端运行的脚本。 |
| ondragenter | script | 当元素元素已被拖动到有效拖放区域时运行的脚本。 |
| ondragleave | script | 当元素离开有效拖放目标时运行的脚本。 |
| ondragover | script | 当元素在有效拖放目标上正在被拖动时运行的脚本。 |
| ondragstart | script | 在拖动操作开端运行的脚本。 |
| ondrop | script | 当被拖元素正在被拖放时运行的脚本。 |
| onmousedown | script | 当元素上按下鼠标按钮时触发。 |
| onmousemove | script | 当鼠标指针移动到元素上时触发。 |
| onmouseout | script | 当鼠标指针移出元素时触发。 |
| onmouseover | script | 当鼠标指针移动到元素上时触发。 |
| onmouseup | script | 当在元素上释放鼠标按钮时触发。 |
| onmousewheel | script | 当鼠标滚轮正在被滚动时运行的脚本。 |
| onscroll | script | 当元素滚动条被滚动时运行的脚本。 |

## 69.6 Media 事件

由媒介（比如视频、图像和音频）触发的事件（适用于所有 HTML 元素，但常见于媒介元素中，比如 <audio>、<embed>、<img>、<object> 以及 <video>）:

|  |  |  |
| --- | --- | --- |
| 属性 | 值 | 描述 |
| onabort | script | 在退出时运行的脚本。 |
| oncanplay | script | 当文件就绪可以开始播放时运行的脚本（缓冲已足够开始时）。 |
| oncanplaythrough | script | 当媒介能够无需因缓冲而停止即可播放至结尾时运行的脚本。 |
| ondurationchange | script | 当媒介长度改变时运行的脚本。 |
| onemptied | script | 当发生故障并且文件突然不可用时运行的脚本（比如连接意外断开时）。 |
| onended | script | 当媒介已到达结尾时运行的脚本（可发送类似“感谢观看”之类的消息）。 |
| onerror | script | 当在文件加载期间发生错误时运行的脚本。 |
| onloadeddata | script | 当媒介数据已加载时运行的脚本。 |
| onloadedmetadata | script | 当元数据（比如分辨率和时长）被加载时运行的脚本。 |
| onloadstart | script | 在文件开始加载且未实际加载任何数据前运行的脚本。 |
| onpause | script | 当媒介被用户或程序暂停时运行的脚本。 |
| onplay | script | 当媒介已就绪可以开始播放时运行的脚本。 |
| onplaying | script | 当媒介已开始播放时运行的脚本。 |
| onprogress | script | 当浏览器正在获取媒介数据时运行的脚本。 |
| onratechange | script | 每当回放速率改变时运行的脚本（比如当用户切换到慢动作或快进模式）。 |
| onreadystatechange | script | 每当就绪状态改变时运行的脚本（就绪状态监测媒介数据的状态）。 |
| onseeked | script | 当 seeking 属性设置为 false（指示定位已结束）时运行的脚本。 |
| onseeking | script | 当 seeking 属性设置为 true（指示定位是活动的）时运行的脚本。 |
| onstalled | script | 在浏览器不论何种原因未能取回媒介数据时运行的脚本。 |
| onsuspend | script | 在媒介数据完全加载之前不论何种原因终止取回媒介数据时运行的脚本。 |
| ontimeupdate | script | 当播放位置改变时（比如当用户快进到媒介中一个不同的位置时）运行的脚本。 |
| onvolumechange | script | 每当音量改变时（包括将音量设置为静音）时运行的脚本。 |
| onwaiting | script | 当媒介已停止播放但打算继续播放时（比如当媒介暂停已缓冲更多数据）运行脚本 |

# 第70章 HTML Canvas 参考手册

## 70.1 HTML Canvas 参考手册

HTML <canvas> 标签用于通过脚本（通常是 JavaScript）动态绘制图形。

如需了解有关 <canvas> 的更多信息，请阅读我们的 HTML Canvas 教程。

## 70.2 颜色、样式和阴影

|  |  |
| --- | --- |
| 属性 | 描述 |
| fillStyle | 设置或返回用于填充绘画的颜色、渐变或模式 |
| strokeStyle | 设置或返回用于笔触的颜色、渐变或模式 |
| shadowColor | 设置或返回用于阴影的颜色 |
| shadowBlur | 设置或返回用于阴影的模糊级别 |
| shadowOffsetX | 设置或返回阴影距形状的水平距离 |
| shadowOffsetY | 设置或返回阴影距形状的垂直距离 |
| 方法 | 描述 |
| createLinearGradient() | 创建线性渐变（用在画布内容上） |
| createPattern() | 在指定的方向上重复指定的元素 |
| createRadialGradient() | 创建放射状/环形的渐变（用在画布内容上） |
| addColorStop() | 规定渐变对象中的颜色和停止位置 |

## 70.3 线条样式

|  |  |
| --- | --- |
| 属性 | 描述 |
| lineCap | 设置或返回线条的结束端点样式 |
| lineJoin | 设置或返回两条线相交时，所创建的拐角类型 |
| lineWidth | 设置或返回当前的线条宽度 |
| miterLimit | 设置或返回最大斜接长度 |

## 70.4 矩形

|  |  |
| --- | --- |
| 方法 | 描述 |
| rect() | 创建矩形 |
| fillRect() | 绘制“被填充”的矩形 |
| strokeRect() | 绘制矩形（无填充） |
| clearRect() | 在给定的矩形内清除指定的像素 |
|  |  |

## 70.5 路径

|  |  |
| --- | --- |
| 方法 | 描述 |
| fill() | 填充当前绘图（路径） |
| stroke() | 绘制已定义的路径 |
| beginPath() | 起始一条路径，或重置当前路径 |
| moveTo() | 把路径移动到画布中的指定点，不创建线条 |
| closePath() | 创建从当前点回到起始点的路径 |
| lineTo() | 添加一个新点，然后在画布中创建从该点到最后指定点的线条 |
| clip() | 从原始画布剪切任意形状和尺寸的区域 |
| quadraticCurveTo() | 创建二次贝塞尔曲线 |
| bezierCurveTo() | 创建三次方贝塞尔曲线 |
| arc() | 创建弧/曲线（用于创建圆形或部分圆） |
| arcTo() | 创建两切线之间的弧/曲线 |
| isPointInPath() | 如果指定的点位于当前路径中，则返回 true，否则返回 false |

## 70.6 转换

|  |  |
| --- | --- |
| 方法 | 描述 |
| scale() | 缩放当前绘图至更大或更小 |
| rotate() | 旋转当前绘图 |
| translate() | 重新映射画布上的 (0,0) 位置 |
| transform() | 替换绘图的当前转换矩阵 |
| setTransform() | 将当前转换重置为单位矩阵。然后运行 transform() |

## 70.7 文本

|  |  |
| --- | --- |
| 属性 | 描述 |
| font | 设置或返回文本内容的当前字体属性 |
| textAlign | 设置或返回文本内容的当前对齐方式 |
| textBaseline | 设置或返回在绘制文本时使用的当前文本基线 |
| 方法 | 描述 |
| fillText() | 在画布上绘制“被填充的”文本 |
| strokeText() | 在画布上绘制文本（无填充） |
| measureText() | 返回包含指定文本宽度的对象 |

## 70.8 图像绘制

|  |  |
| --- | --- |
| 方法 | 描述 |
| drawImage() | 向画布上绘制图像、画布或视频 |

## 70.9 像素操作

|  |  |
| --- | --- |
| 属性 | 描述 |
| width | 返回 ImageData 对象的宽度 |
| height | 返回 ImageData 对象的高度 |
| data | 返回一个对象，其包含指定的 ImageData 对象的图像数据 |
| 方法 | 描述 |
| createImageData() | 创建新的、空白的 ImageData 对象 |
| getImageData() | 返回 ImageData 对象，该对象为画布上指定的矩形复制像素数据 |
| putImageData() | 把图像数据（从指定的 ImageData 对象）放回画布上 |

## 70.10 合成

|  |  |
| --- | --- |
|  |  |
| 属性 | 描述 |
| globalAlpha | 设置或返回绘图的当前 alpha 或透明值 |

## 70.11 其他

|  |  |
| --- | --- |
| 方法 | 描述 |
| save() | 保存当前环境的状态 |
| restore() | 返回之前保存过的路径状态和属性 |
| createEvent() |  |
| getContext() |  |
| toDataURL() |  |

# 第71章 HTML 音频/视频参考手册