# HTML

**一、HTML 简介**

**1、什么是 HTML？**

HTML 是用来描述网页的一种语言。

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

**2、HTML 标签**

HTML 标记标签通常被称为 HTML 标签 (HTML tag)。

* HTML 标签是由尖括号包围的关键词，比如 <html>
* HTML 标签通常是成对出现的，比如 <b> 和 </b>
* 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
* 开始和结束标签也被称为开放标签和闭合标签

**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

PS：<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

HTML 链接是通过 <a> 标签进行定义的

HTML 图像是通过 <img> 标签进行定义的

1. **HTML 编辑器**

**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

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

# **三、HTML 元素**

HTML 文档是由 HTML 元素定义的。

**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 /> |  |  |

PS：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）。

**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

**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。

**4、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。

# **四、HTML 属性**

属性为 HTML 元素提供附加信息。

**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。

**2、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。

**3、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，那么必须使用单引号。

**PS:适用于大多数 HTML 元素的属性：**

|  |  |  |
| --- | --- | --- |
| **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） |

# **五、HTML 标题**

**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。

注释：浏览器会自动地在标题的前后添加空行；

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。

**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。

**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。

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）。

**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：<!-- 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。

**5、HTML 标签参考手册**

|  |  |
| --- | --- |
| **标签** | **描述** |
| [<html>](https://www.w3school.com.cn/tags/tag_html.asp) | 定义 HTML 文档。 |
| [<body>](https://www.w3school.com.cn/tags/tag_body.asp) | 定义文档的主体。 |
| [<h1> to <h6>](https://www.w3school.com.cn/tags/tag_hn.asp) | 定义 HTML 标题 |
| [<hr>](https://www.w3school.com.cn/tags/tag_hr.asp) | 定义水平线。 |
| [<!-->](https://www.w3school.com.cn/tags/tag_comment.asp) | 定义注释。 |

# **六、HTML 段落**

**1、HTML 段落**

段落是通过 <p> 标签定义的。

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。）

**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

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。

**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：

<p>This is<br />a para<br />graph with line breaks</p>

<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。

<br> 还是 <br />

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。

**4、HTML 输出 - 有用的提示**

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，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 样式**

style 属性用于改变 HTML 元素的样式。

**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。

**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！

|  |  |
| --- | --- |
| **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！

**3、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>

PS：style 属性淘汰了“旧的” bgcolor 属性。

**4、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>

PS：style 属性淘汰了旧的 <font> 标签。

**5、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>The heading above is aligned to the center of this page.</p>

</body>

</html>

PS：style 属性淘汰了旧的 "align" 属性。

# **HTML 文本格式化**

1. **文本格式化标签**

|  |  |
| --- | --- |
| **标签** | **描述** |
| [<b>](https://www.w3school.com.cn/tags/tag_font_style.asp) | 定义粗体文本。 |
| [<big>](https://www.w3school.com.cn/tags/tag_font_style.asp) | 定义大号字。 |
| [<em>](https://www.w3school.com.cn/tags/tag_phrase_elements.asp) | 定义着重文字。 |
| [<i>](https://www.w3school.com.cn/tags/tag_font_style.asp) | 定义斜体字。 |
| [<small>](https://www.w3school.com.cn/tags/tag_font_style.asp) | 定义小号字。 |
| [<strong>](https://www.w3school.com.cn/tags/tag_phrase_elements.asp) | 定义加重语气。 |
| [<sub>](https://www.w3school.com.cn/tags/tag_sup.asp) | 定义下标字。 |
| [<sup>](https://www.w3school.com.cn/tags/tag_sup.asp) | 定义上标字。 |
| [<ins>](https://www.w3school.com.cn/tags/tag_ins.asp) | 定义插入字。 |
| [<del>](https://www.w3school.com.cn/tags/tag_del.asp) | 定义删除字。 |
| [<s>](https://www.w3school.com.cn/tags/tag_strike.asp) | 不赞成使用。使用 <del> 代替。 |
| [<strike>](https://www.w3school.com.cn/tags/tag_strike.asp) | 不赞成使用。使用 <del> 代替。 |
| [<u>](https://www.w3school.com.cn/tags/tag_u.asp) | 不赞成使用。使用样式（style）代替。 |

**2、“计算机输出”标签**

|  |  |
| --- | --- |
| **标签** | **描述** |
| [<code>](https://www.w3school.com.cn/tags/tag_phrase_elements.asp) | 定义计算机代码。 |
| [<kbd>](https://www.w3school.com.cn/tags/tag_phrase_elements.asp) | 定义键盘码。 |
| [<samp>](https://www.w3school.com.cn/tags/tag_phrase_elements.asp) | 定义计算机代码样本。 |
| [<tt>](https://www.w3school.com.cn/tags/tag_font_style.asp) | 定义打字机代码。 |
| [<var>](https://www.w3school.com.cn/tags/tag_phrase_elements.asp) | 定义变量。 |
| [<pre>](https://www.w3school.com.cn/tags/tag_pre.asp) | 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。 |

1. **引用、引用和术语定义**

|  |  |
| --- | --- |
| **标签** | **描述** |
| [<abbr>](https://www.w3school.com.cn/tags/tag_abbr.asp) | 定义缩写。 |
| [<acronym>](https://www.w3school.com.cn/tags/tag_acronym.asp) | 定义首字母缩写。 |
| [<address>](https://www.w3school.com.cn/tags/tag_address.asp) | 定义地址。 |
| [<bdo>](https://www.w3school.com.cn/tags/tag_bdo.asp) | 定义文字方向。 |
| [<blockquote>](https://www.w3school.com.cn/tags/tag_blockquote.asp) | 定义长的引用。 |
| [<q>](https://www.w3school.com.cn/tags/tag_q.asp) | 定义短的引用语。 |
| [<cite>](https://www.w3school.com.cn/tags/tag_phrase_elements.asp) | 定义引用、引证。 |
| [<dfn>](https://www.w3school.com.cn/tags/tag_phrase_elements.asp) | 定义一个定义项目。 |

# **HTML 引用**

**1、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。

**2、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。

3、**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。

**4、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：

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

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

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

否则，<dfn> 文本内容即是项目，并且父元素包含定义。

**5、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。

**6、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。

**7、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。

1. **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。 |

# **HTML 计算机代码**

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。

* **HTML 键盘格式：**

HTML <kbd> 元素定义键盘输入。

* **HTML 样本格式：**

HTML <samp> 元素定义计算机输出。

* **HTML 代码格式：**

HTML <code> 元素定义编程代码（<code> 元素不保留多余的空格和折行），

如需解决该问题，您必须在 <pre> 元素中包围代码。

* **HTML 变量格式化：**

HTML <var> 元素定义数学变量。

**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 |

# **HTML 注释**

**1、HTML 注释标签**

通过如下语法向 HTML 源代码添加注释：<!-- 在此处写注释 -->

* 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。
* 可以利用注释在 HTML 中放置通知和提醒信息
* 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

**2、条件注释**

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。

**3、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。

# **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。

**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：

**（1）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>

**（2）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>

**（3）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>](https://www.w3school.com.cn/tags/tag_style.asp) | 定义样式定义。 |
| [<link>](https://www.w3school.com.cn/tags/tag_link.asp) | 定义资源引用。 |
| [<div>](https://www.w3school.com.cn/tags/tag_div.asp) | 定义文档中的节或区域（块级）。 |
| [<span>](https://www.w3school.com.cn/tags/tag_span.asp) | 定义文档中的行内的小块或区域。 |
| [<font>](https://www.w3school.com.cn/tags/tag_font.asp) | 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>](https://www.w3school.com.cn/tags/tag_basefont.asp) | 定义基准字体。不赞成使用。请使用样式。 |
| [<center>](https://www.w3school.com.cn/tags/tag_center.asp) | 对文本进行水平居中。不赞成使用。请使用样式。 |

# **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。

**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，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。

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。

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：

（1）通过使用 href 属性 - 创建指向另一个文档的链接

（2）通过使用 name 属性 - 创建文档内的书签

**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。

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。

**3、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：

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

**4、TML 链接 - name 属性**

name 属性规定锚（anchor）的名称。

* 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，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>

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，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。

**5、基本的注意事项**

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。

# **HTML 图像**

**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。

**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">

在浏览器无法载入图像时，替换文本属性告诉读者她们失去的信息。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。

**3、基本的注意事项**

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。

# **HTML 表格**

**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。

**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。

**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。

**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。

1. **表格标签**

|  |  |
| --- | --- |
| **表格** | **描述** |
| [<table>](https://www.w3school.com.cn/tags/tag_table.asp) | 定义表格 |
| [<caption>](https://www.w3school.com.cn/tags/tag_caption.asp) | 定义表格标题。 |
| [<th>](https://www.w3school.com.cn/tags/tag_th.asp) | 定义表格的表头。 |
| [<tr>](https://www.w3school.com.cn/tags/tag_tr.asp) | 定义表格的行。 |
| [<td>](https://www.w3school.com.cn/tags/tag_td.asp) | 定义表格单元。 |
| [<thead>](https://www.w3school.com.cn/tags/tag_thead.asp) | 定义表格的页眉。 |
| [<tbody>](https://www.w3school.com.cn/tags/tag_tbody.asp) | 定义表格的主体。 |
| [<tfoot>](https://www.w3school.com.cn/tags/tag_tfoot.asp) | 定义表格的页脚。 |
| [<col>](https://www.w3school.com.cn/tags/tag_col.asp) | 定义用于表格列的属性。 |
| [<colgroup>](https://www.w3school.com.cn/tags/tag_colgroup.asp) | 定义表格列的组。 |

# **HTML 列表**

HTML 支持有序、无序和定义列表

**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>。

PS：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。

**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。

**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。

## **4、列表标签**

|  |  |
| --- | --- |
| **标签** | **描述** |
| [<ol>](https://www.w3school.com.cn/tags/tag_ol.asp) | 定义有序列表。 |
| [<ul>](https://www.w3school.com.cn/tags/tag_ul.asp) | 定义无序列表。 |
| [<li>](https://www.w3school.com.cn/tags/tag_li.asp) | 定义列表项。 |
| [<dl>](https://www.w3school.com.cn/tags/tag_dl.asp) | 定义定义列表。 |
| [<dt>](https://www.w3school.com.cn/tags/tag_dt.asp) | 定义定义项目。 |
| [<dd>](https://www.w3school.com.cn/tags/tag_dd.asp) | 定义定义的描述。 |
| [<dir>](https://www.w3school.com.cn/tags/tag_dir.asp) | 已废弃。使用 <ul> 代替它。 |
| [<menu>](https://www.w3school.com.cn/tags/tag_menu.asp) | 已废弃。使用 <ul> 代替它。 |

# **HTML <div>和<span>**

可以通过 <div> 和 <span> 将 HTML 元素组合起来。

**1、HTML 块元素**

大多数 HTML 元素被定义为块级元素或内联元素。

（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>

**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>

**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。

**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

## **5、HTML 分组标签**

|  |  |
| --- | --- |
| **标签** | **描述** |
| [<div>](https://www.w3school.com.cn/tags/tag_div.asp" \o "HTML <div> 标签) | 定义文档中的分区或节（division/section）。 |
| [<span>](https://www.w3school.com.cn/tags/tag_span.asp" \o "HTML <span> 标签) | 定义 span，用来组合文档中的行内元素。 |

# **HTML 布局**

### **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 |

# **HTML 响应式**

**1、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

**2、使用 Bootstrap**

另一个创建响应式设计的方法，是使用现成的 CSS 框架。

* Bootstrap 是最流行的开发响应式 web 的 HTML, CSS, 和 JS 框架。
* Bootstrap 帮助您开发在任何尺寸都外观出众的站点：显示器、笔记本电脑、平板电脑或手机

# **HTML 框架**

**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

**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。

**3、基本的注意事项**

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。）

# **HTML Iframe**

iframe 用于在网页内显示网页。

**1、添加 iframe 的语法**

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

URL 指向隔离页面的位置。

**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%"）。

**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：

**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

# **HTML背景**

**1、<body> 拥有两个配置背景的标签。**背景可以是颜色或者图像。

* 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。
* 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。

提示：如果打算使用背景图片，需要紧记以下几点：

* 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？

**2、基本的注意事项**

<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。

# **HTML脚本**

JavaScript 使 HTML 页面具有更强的动态和交互性。

**1、HTML script 元素**

* <script> 标签用于定义客户端脚本，比如 JavaScript。
* script 元素既可包含脚本语句，也可通过 src 属性指向外部脚本文件。
* 必需的 type 属性规定脚本的 MIME 类型。
* JavaScript 最常用于图片操作、表单验证以及内容动态更新。

**2、<noscript> 标签**

* <noscript> 标签提供无法使用脚本时的替代内容，比方在浏览器禁用脚本时，或浏览器不支持客户端脚本时。
* noscript 元素可包含普通 HTML 页面的 body 元素中能够找到的所有元素。
* 只有在浏览器不支持脚本或者禁用脚本时，才会显示 noscript 元素中的内容。

**3、如何应付老式的浏览器**

如果浏览器压根没法识别 <script> 标签，那么 <script> 标签所包含的内容将以文本方式显示在页面上。为了避免这种情况发生，你应该将脚本隐藏在注释标签当中。那些老的浏览器（无法识别 <script> 标签的浏览器）将忽略这些注释，所以不会将标签的内容显示到页面上。而那些新的浏览器将读懂这些脚本并执行它们，即使代码被嵌套在注释标签内。

# **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 |

**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

**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。

**3、相对路径**

相对路径指向了相对于当前页面的文件。

**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。

# **HTML 头部元素**

**1、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>。

**2、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

**3、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）。

**4、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。

**5、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。

**6、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。

**7、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。

## **8、HTML 头部元素**

|  |  |
| --- | --- |
| **标签** | **描述** |
| [<head>](https://www.w3school.com.cn/tags/tag_head.asp" \o "HTML <head> 标签) | 定义关于文档的信息。 |
| [<title>](https://www.w3school.com.cn/tags/tag_title.asp" \o "HTML <title> 标签) | 定义文档标题。 |
| [<base>](https://www.w3school.com.cn/tags/tag_base.asp" \o "HTML <base> 标签) | 定义页面上所有链接的默认地址或默认目标。 |
| [<link>](https://www.w3school.com.cn/tags/tag_link.asp" \o "HTML <link> 标签) | 定义文档与外部资源之间的关系。 |
| [<meta>](https://www.w3school.com.cn/tags/tag_meta.asp" \o "HTML <meta> 标签) | 定义关于 HTML 文档的元数据。 |
| [<script>](https://www.w3school.com.cn/tags/tag_script.asp" \o "HTML <script> 标签) | 定义客户端脚本。 |
| [<style>](https://www.w3school.com.cn/tags/tag_style.asp" \o "HTML <style> 标签) | 定义文档的样式信息。 |

# **HTML 头部元素**

HTML 中的预留字符必须被替换为字符实体。

**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）。

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）。

**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。

## **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; |

# **HTML 统一资源点位管理器**

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

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。 |

# **HTML 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 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。

## **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 |

# **HTML 颜色**

颜色由红色、绿色、蓝色混合而成。

**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) |

**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 |
|  | #A52A2A | Brown |

**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）。

# **HTML <!DOCTYPE>**

<!DOCTYPE> 声明帮助浏览器正确地显示网页。

**1、<!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。

**2、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">

# **XHTML 简介**

XHTML 是以 XML 格式编写的 HTML。

**1、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

**2、为什么使用 XHTML？**

因特网上的很多页面包含了“糟糕”的 HTML。

XML 是一种必须正确标记且格式良好的标记语言。

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。

**3、与 HTML 相比最重要的区别：**

（1）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

（2）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

（3）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

**4、<!DOCTYPE ....> 是强制性的**

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

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。

**5、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

# **XHTML 元素**

XHTML 元素是以 XML 格式编写的 HTML 元素。

**1、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

1. 基本的注意事项

* 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

# **HTML 表单**

HTML 表单用于搜集不同类型的用户输入。

**1、<form> 元素**

HTML 表单用于收集用户输入。

<form> 元素定义 HTML 表单

**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。

**（1）<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。

|  |  |
| --- | --- |
| **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） |

**（2）文本输入**

<input type="text"> 定义用于文本输入的单行输入字段.

**（3）单选按钮输入**

<input type="radio"> 定义单选按钮。

**（4）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

**（5）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。

**（6）Method 属性**

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

* 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。

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。

* 何时使用 POST？

如果表单正在更新数据，或者包含敏感信息（例如密码）。

POST 的安全性更加，因为在页面地址栏中被提交的数据是不可见的。

**（7）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。

**（8）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。

**（9）HTML Form 属性**

HTML <form> 元素，已设置所有可能的属性

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）。 |

# **HTML 表单元素**

**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。

**2、<select> 元素**

<select> 元素定义下拉列表

**3、<option> 元素**

列表通常会把首个选项显示为被选选项。

能够通过添加 selected 属性来定义预定义选项。

**4、<textarea> 元素**

<textarea> 元素定义多行输入字段（文本域）

**5、<button> 元素**

<button> 元素定义可点击的按钮

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。

**6、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。

# **HTML 输入类型**

**1、输入类型：text**

<input type="text"> 定义供文本输入的单行输入字段

**2、输入类型：password**

<input type="password"> 定义密码字段

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）。

**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）

**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

**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。

**6、Input Type: button**

<input type="button> 定义按钮。

**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。

**8、输入类型：number**

<input type="number"> 用于应该包含数字值的输入字段。

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。

## **9、输入限制**

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。 |

**10、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。

**11、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。

**12、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。

能够使用如下属性来规定限制：min、max、step、value。

**13、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。

**14、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。

**15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。

**16、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。

**17、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。

**18、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。

**19、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）。

**20、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。

**21、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。

# HTML 输入属性

**1、value 属性**

value 属性规定输入字段的初始值

**2、readonly 属性**

readonly 属性规定输入字段为只读（不能修改）

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

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

**4、size 属性**

size 属性规定输入字段的尺寸（以字符计）

**5、maxlength 属性**

* maxlength 属性规定输入字段允许的最大长度
* 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。

**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

**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。
* 提示：在某些浏览器中，您也许需要手动启用自动完成功能。

**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。

**9、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。

**10、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。

**11、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"。

**12、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"。

**13、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"。

**14、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"。

**15、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。

**16、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">。

**17、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。

**18、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。

**19、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。

**20、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。

**21、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。

**22、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.

**23、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。

# HTML5 简介

**1、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。

**2、HTML5 - 新的属性语法**

HTML5 标准允许 4 中不同的属性语法。

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="John Doe" disabled> |
| Unquoted | <input type="text" value=John Doe> |
| Double-quoted | <input type="text" value="John Doe"> |
| Single-quoted | <input type="text" value='John Doe'> |

**3、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 浏览器支持

**1、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，可以帮助老式浏览器处理”未知的“ HTML 元素。

注释：甚至可以教授石器时代的 IE6 如何处理未知的 HTML 元素。

**2、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。

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。

**3、向 HTML 添加新元素**

可以通过浏览器 trick 向 HTML 添加任何新元素。

**4、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）。

# HTML5 新元素

**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）。 |

## 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。 |

## **3、HTML5 - 新的属性语法**

该例演示 <input> 标签中使用的不同语法：

|  |  |
| --- | --- |
| **标签** | **描述** |
| Empty | <input type="text" value="John Doe" disabled> |
| Unquoted | <input type="text" value=John> |
| Double-quoted | <input type="text" value="John Doe"> |
| Single-quoted | <input type="text" value='John Doe'> |

在 HTML5 中，根据属性所需，可能会使用所有这四种语法。

## **4、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。 |

## 5、**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。 |

# HTML5 语义元素

**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。

**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>

**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。

**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

**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。
* 能够使用该定义来决定如何嵌套元素吗？不，不能！
* 在因特网上， <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。
* <section> 元素包含 <section> 元素，同时 <article> 元素包含 <article> 元素。

**6、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。**

**7、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。**
* **可以在一个文档中使用多个 <footer> 元素。**

**8、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。
* <img> 元素定义图像，<figcaption> 元素定义标题。

**9、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。”

10、HTML5 中的语义元素

下面列出了以字母顺序排列的 HTML5 新语义元素。

这些链接指向完整的 HTML 参考手册。

|  |  |
| --- | --- |
| **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。 |

四十一、HTML(5) 样式指南和代码约定

**1、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，必须创建属于自己的最佳实践、样式指南和代码约定。

**2、使用正确的文档类型**

始终在文档的首行声明文档类型。

**3、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。

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

**4、关闭所有 HTML 元素**

在 HTML5 中，不必关闭所有元素（例如 <p> 元素）。

建议关闭所有 HTML 元素。

**5、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

**6、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。

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。

**7、空格和等号**

等号两边的空格是合法的，但是精简空格更易阅读。

**8、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。

尽量避免代码行超过 80 个字符。

**9、空行和缩进**

* 勿毫无理由地增加空行。
* 为了提高可读性，增加空行来分隔大型或逻辑代码块。
* 为了提高可读性，增加两个空格的缩进。请勿使用 TAB。
* 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

**10、省略 <html> 和 <body>**

* 在 HTML5 标准中，能够省略 <html> 标签和 <body> 标签。
* 但不推荐省略 <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。
* 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。
* **11、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，能够降低 HTML 的复杂性。

**12、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。

长注释，跨越多行，应该通过 <!-- 和 --> 在独立的行中书写。

**13、样式表**

使用简单的语法来链接样式表（type 属性不是必需的）：

短规则可以压缩为一行；

长规则应该分为多行：

* 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

**14、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，这些小错误将破坏网站。
* 为了避免这些问题，始终使用小写文件名（如果可以的话）。

**15、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。

# HTML5 Canvas

canvas 元素用于在网页上绘制图形。

**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，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。

**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。

**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：

* JavaScript 使用 id 来寻找 canvas 元素；
* 然后，创建 context 对象；
* 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；
* fillStyle 方法染色，fillRect 方法规定了形状、位置和尺寸。

# HTML5 内联 SVG

**1、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

**2、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

**3、把 SVG 直接嵌入 HTML 页面**

在 HTML5 中，能够将 SVG 元素直接嵌入 HTML 页面中。

# HTML5 内联 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。

**1、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。

**2、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。

**3、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

# 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。

**1、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。

**2、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。 |
| RealVideo | * .rm * .ram | RealVideo 格式是由 Real Media 针对因特网开发的。该格式允许低带宽条件下（在线视频、网络电视）的视频流。由于是低带宽优先的，质量常会降低。 |
| Flash | * .swf * .flv | Flash (Shockwave) 格式是由 Macromedia 开发的。Shockwave 格式需要额外的组件来播放。但是该组件会预装到 Firefox 或 IE 之类的浏览器上。 |
| Mpeg-4 | .mp4 | Mpeg-4 (with H.264 video compression) 是一种针对因特网的新格式。事实上，YouTube 推荐使用 MP4。YouTube 接收多种格式，然后全部转换为 .flv 或 .mp4 以供分发。越来越多的视频发布者转到 MP4，将其作为 Flash 播放器和 HTML5 的因特网共享格式。 |

1. **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](https://www.w3school.com.cn/i/beatles.mid)。  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。 |

**5、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。

# HTML Object元素

<object> 的作用是支持 HTML 助手（插件）。

**1、HTML 助手（插件）**

* 辅助应用程序（helper application）是可由浏览器启动的程序。辅助应用程序也称为插件。
* 辅助程序可用于播放音频和视频（以及其他）。辅助程序是使用 <object> 标签来加载的。
* 使用辅助程序播放视频和音频的一个优势是，您能够允许用户来控制部分或全部播放设置。
* 大多数辅助应用程序允许对音量设置和播放功能（比如后退、暂停、停止和播放）的手工（或程序的）控制。

# HTML 音频

**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。

**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）。

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。

**3、使用 <object> 元素**

<object tag> 标签也可以定义外部（非 HTML）内容的容器。

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。

**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。

问题：

* <audio> 标签在 HTML 4 中是无效的。页面无法通过 HTML 4 验证。
* 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。

1. **最好的 HTML 解决方法**

问题：

* 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。

**6、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：

* 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。

**7、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。

**8、内联的声音**

* 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。
* 如果打算在 web 应用程序中使用内联声音，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。
* 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。

## **9、HTML 4.01 多媒体标签**

|  |  |
| --- | --- |
| **标签** | **描述** |
| [<applet>](https://www.w3school.com.cn/tags/tag_applet.asp" \o "HTML <applet> 标签) | 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>](https://www.w3school.com.cn/tags/tag_object.asp" \o "HTML <object> 标签) | 定义内嵌对象。 |
| [<param>](https://www.w3school.com.cn/tags/tag_param.asp" \o "HTML <param> 标签) | 定义对象的参数。 |

## **HTML 5 多媒体标签**

|  |  |
| --- | --- |
| **标签** | **描述** |
| [<audio>](https://www.w3school.com.cn/tags/tag_audio.asp" \o "HTML <audio> 标签) | 标签定义声音，比如音乐或其他音频流。 |
| [<embed>](https://www.w3school.com.cn/tags/tag_embed.asp" \o "HTML <embed> 标签) | 标签定义嵌入的内容，比如插件。 |

# HTML5 地理定位

HTML5 Geolocation（地理定位）用于定位用户的位置。

**1、HTML5 - 使用地理定位**

使用 getCurrentPosition() 方法来获得用户的位置。

**2、在地图中显示结果**

如需在地图中显示结果，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。

**3、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 |

**4、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。

# HTML 本地存储

**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。

**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）

**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。

**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。

# 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。

**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：

* 离线浏览 - 用户可在应用离线时使用它们
* 速度 - 已缓存资源加载得更快
* 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

**2、Cache Manifest 基础**

* 如需启用应用程序缓存，在文档的 <html> 标签中包含 manifest 属性。
* 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。

**3、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）

**4、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：

* 用户清空浏览器缓存
* manifest 文件被修改（参阅下面的提示）
* 由程序来更新应用缓存

# HTML Web Workers

Web worker 是运行在后台的 JavaScript，不会影响页面的性能。

**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。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。

**2、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：

* window 对象
* document 对象
* parent 对象

# HTML Server-Sent 事件

Server-Sent 事件允许网页从服务器获得更新。

**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。

**2、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 |