

第五次重大修改

HTML

5

中文教程



极客学院出版

## 前言

---

HTML5 是近十年来 Web 开发标准最巨大的飞跃。HTML5 并非仅仅用来表示 Web 内容，它将 Web 带入一个成熟的应用平台，在 HTML5 平台上，视频、音频、图象、动画，以及同电脑的交互都被标准化。

本教程将教会大家HTML5中的新特性，以及每种标签的使用

HTML5 中的一些有趣的新特性：

- 用于绘画的 canvas 元素
- 用于媒介回放的 video 和 audio 元素
- 对本地离线存储的更好的支持
- 新的特殊内容元素，比如 article、footer、header、nav、section
- 新的表单控件，比如 calendar、date、time、email、url、search

HTML5 浏览器支持：

最新版本的 Safari、Chrome、Firefox 以及 Opera 支持某些 HTML5 特性。Internet Explorer 9 将支持某些 HTML5 特性。



图片 .1 HTML5

HTML5 即最新的增强版 HTML。

从技术上讲，HTML 并不是一门编程语言，而是一种标记语言。

本教程旨在很好的理解 HTML5。

更新日期	更新内容
2015-04-13	HTML 5 中文教程发布

# 目录

---

前言	1
第 1 章 HTML5 教程	5
HTML5 概述	6
HTML5 语法	8
HTML5 属性	11
HTML5 事件	13
HTML5 表单 2.0	16
HTML5 SVG 教程	19
HTML5 MathML 教程	25
HTML5 Web 存储	28
HTML5 Web SQL 数据库	31
HTML5 服务器推送事件	34
HTML5 WebSockets 教程	36
HTML5 画布	40
HTML5 音频和视频	43
HTML5 地理定位	47
HTML5 微数据	51
HTML5 拖放	54
HTML5 Web Workers	59
第 2 章 HTML5 标签参考	63
HTML5 标签参考	63
HTML5 过时标签和属性	68
HTML5 新标签（元素）	71
第 3 章 有用的 HTML5 参考	73

	HTML5 快速指南 . . . . .	74
	HTML5 颜色名称 . . . . .	78
	HTML5 字体参考 . . . . .	82
	HTML5 URL 编码 . . . . .	84
	HTML5 字符实体参考 . . . . .	91
	HTML5 字符编码 . . . . .	96
第 4 章	HTML5 工具 . . . . .	97
	HTML5 Modernizr . . . . .	98
	HTML5 验证 . . . . .	102
	HTML5 在线编辑器 . . . . .	104
	HTML5 颜色代码生成器 . . . . .	105
第 5 章	有用的 HTML5 资源 . . . . .	106
	有用的 HTML5 资源 . . . . .	106



T



1

# HTML5 教程



## HTML5 概述

---

HTML5 是 HTML 标准的下一个重要版本，用来替代 HTML 4.01, XHTML 1.0 以及 XHTML 1.1。HTML5 也是一种在万维网上构建和呈现内容的标准。

HTML5 是万维网联盟（W3C）和网页超文本技术工作小组（WHATWG）合作的产物。

这一新标准中加入了视频播放和拖放等特性，过去这都依赖于第三方浏览器插件，比如 Adobe Flash, Microsoft Silverlight 以及 Google Gears。

### 浏览器支持

最新版 Apple Safari, Mozilla FireFox 和 Opera 支持大部分 HTML5 特性，IE9 也支持一些 HTML5 的功能。

预装在 iPhones, iPads 和 Android 手机上的手机浏览器都对 HTML5 有良好的支持。

### 新特性

HTML5 引入了许多新元素和属性帮助我们构建现代化的网站。下面是 HTML5 引入的主要特性：

- **新的语义化元素：** 比如 `<header>`, `<footer>` 和 `<section>`。
- **表单 2.0：** 改进了 HTML Web 表单，为 `<input>` 标签引入了一些新的属性。
- **持久的本地存储：** 为了不通过第三方插件实现。
- **WebSocket：** 用于 Web 应用程序的下一代双向通信技术。
- **服务器推送事件：** HTML5 引入了从 Web 服务器到 Web 浏览器的事件，也被称作服务器推送事件（SSE）。
- **Canvas：** 支持用 JavaScript 以编程的方式进行二维绘图。
- **音频和视频：** 在网页中嵌入音频或视频而无需借助第三方插件。
- **地理定位：** 用户可以选择与我们的网页共享他们的地理位置。
- **微数据：** 允许我们创建 HTML5 之外的自定义词汇表，以及使用自定义语义扩展网页。
- **拖放：** 把同一网页上的条目从一个位置拖放到另一个位置。

## 向后兼容

HTML5 被设计为尽可能的对现有浏览器向后兼容。新特性都是建立在现有特性的基础上，并且允许我们为旧浏览器提供备用内容。

建议使用少量的 JavaScript 代码检测单个 HTML5 特性的支持度。



## HTML5 语法

---

HTML5 有“自己的” HTML 语法，它与已经发布在网络上的 HTML 4 以及 XHTML1 文档兼容，但是不兼 HTML 4 中更复杂的 SGML 特性。

HTML5 并没有 XHTML 中需要小写标签名，属性要带引号，属性必须有一个值以及必须闭合所有空元素的语法规则。

但是 HTML5 更具灵活性，支持下列形式：

- 标签名大写。
- 属性的双引号可选。
- 属性值可选。
- 闭合空元素可选。

### DOCTYPE

在老版本的 HTML 中，DOCTYPE 很长，因为 HTML 语言是基于 SGML 的，需要引用一个 DTD。

HTML5 作者可以使用简单的语法来指定如下形式的 DOCTYPE：

```
<!DOCTYPE html>
```

上述语法不区分大小写。

### 字符编码

HTML5 作者可以使用简单的语法指定字符编码，如下所示：

```
<meta charset="UTF-8">
```

上述语法不区分大小写。

### <script> 标签

常见的做法是给 script 元素添加一个值为 “text/javascript” 的 type 属性，如下所示：

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

HTML5 移除了所需的额外信息，我们可以使用如下所示的简单语法：

```
<script src="scriptfile.js"></script>
```

## link 标签

目前为止我们这样编写 <link>：

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

HTML5 移除了所需的额外信息，我们可以使用如下所示的简单语法：

```
<link rel="stylesheet" href="stylefile.css">
```

## HTML5 元素

HTML5 元素使用起始标签和结束标签标记。标签使用尖括号之间的标签名限定。

起始标签和结束标签的区别在于后者标签名前面包含一个斜杠。

下面是一个 HTML5 元素示例：

```
<p>...</p>
```

HTML5 标签名不区分大小写，可以全部大写或者混合使用，虽然最常见的约定是始终使用小写。

大多数元素都包含一些内容，比如 <p>...</p> 包含一个段落。但是，有些元素不能包含任意内容，它们被称作空白元素。比如，br, hr, link 和 meta 等等。

这里有一个完整的 [HTML5 元素列表](#)。

## HTML5 属性

元素可以包含属性（attributes），用来给一个元素设置各种属性（properties）。

有些属性被定义为全局的，可以用在任何元素上，而其他的被定义为元素特有的。所有的属性都有一个名称和一个值，看起来如下面的示例所示。

下面是一个使用 HTML5 属性的例子，演示了如何用名为 class 的属性和值 “example” 标记一个 div 元素：

```
<div class="example">...</div>
```

属性只能在起始标签中指定，绝对不能用在结束标签中。

HTML5 属性不区分大小写，可以全部大写或者混合使用，尽管最常见的约定是始终使用小写。

这里有一个完整的 [HTML5 属性](#) 列表。

## HTML5 文档

为了得到更好的结构，引入了下面的标签：

- **section:** 这个标签表示一个通用的文档或者应用程序节。它可以和 h1-h6 一起使用来表示文档结构。
- **article:** 这个标签表示文档内容的一个独立块，比如博客条目或者报纸上的文章。
- **aside:** 这个标签表示与页面其他部分略微相关的内容块。
- **header:** 这个标签表示一个节的头部。
- **footer:** 这个标签表示一个节的脚注，可以包含作者，版权等信息。
- **nav:** 这个标签表示用于导航文档的节。
- **dialog:** 这个标签可以用于标记会话。
- **figure:** 这个标签可以用于关联标题和某些嵌入内容，比如图表和视频。

一个 HTML5 文档的标记看起来就像下面这样：

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>...</title>
</head>
<body>
  <header>...</header>
  <nav>...</nav>
  <article>
    <section>
      ...
    </section>
  </article>
  <aside>...</aside>
  <footer>...</footer>
</body>
```

便于学习这一概念 - 可以进行[在线练习](#)。

# HTML5 属性

正如 HTML5 语法中所阐述的，元素可以包含属性（attributes）给一个元素设置各种属性（properties）。

有些属性被定义为全局的，可以用在任何元素上，而其他的被定义为元素特有的。所有的属性都有一个名称和一个值，看起来如下面的示例所示。

下面是一个使用 HTML5 属性的例子，演示了如何用名为 class 的属性和值 “example” 标记一个 div 元素：

```
<div class="example">...</div>
```

属性只能在起始标签中指定，绝对不能用在结束标签中。

HTML5 属性不区分大小写，可以全部大写或者混合使用，尽管最常见的约定是始终使用小写。

## 标准属性

下面列出的属性几乎所有的 HTML5 标签都支持。

属性	选项	功能
accesskey	用户自定义	定义访问元素的键盘快捷键。
align	right, left, center	水平对齐标签。
background	URL	在元素后面设置一个背景图像。
bgcolor	数值，十六进制值，RGB值	在元素后面设置一个背景颜色。
class	用户定义。	分类一个元素，便于使用级联样式表。
contenteditable	true, false	定义用户是否可以编辑元素的内容。
contextmenu	Menu id	为元素定义上下文菜单。
data-XXXX	用户定义。	自定义属性。 HTML 文档的作者可以定义自己的属性。 自定义属性必须以 "data-" 开头。
draggable	true, false, auto	定义用户是否可以拖动元素。
height	数字值	定义表格，图像或表格单元的高度。
hidden	hidden	定义元素是否应该可见。
id	用户定义。	命名元素，便于使用级联样式表。
item	元素列表。	用于组合元素。
itemprop	条目列表。	用于组合条目。
spellcheck	true, false	定义元素是否必须有拼写或错误检查。

style	CSS 样式表。	给元素定义内联样式。
subject	用户定义 id。	定义元素关联的条目。
tabindex	Tab number	定于元素的 tab 键顺序。
title	用户定义。	元素的“弹出”标题。
valign	top, middle, bottom	HTML 元素内标签的垂直对齐方式。
width	数字值。	定义表格，图像和表格单元的宽度。

完整的 HTML5 标签列表以及相关的属性请参考 [HTML5 标签](#)。

## 自定义属性

HTML5 还引入了一个新特性，就是可以添加自定义的数据属性。

自定义数据属性以 `data-` 开头，基于我们的需求命名。下面是一个简单的例子：

```
<div class="example" data-subject="physics" data-level="complex">
...
</div>
```

上面的例子中两个叫做 `data-subject` 和 `data-level` 的自定义属性在 HTML5 中是完全有效的。我们还可以使用 JavaScript API 或者在 CSS 中以获取标准属性类似的方式获取它们的值。

## HTML5 事件

---

当用户访问我们的网站时，他们会点击文本，图片，链接，将鼠标悬停在某些东西上面等等。这些都是 JavaScript 调用事件的例子。

我们可以在 JavaScript 或者 vbscript 中编写事件处理程序，然后把这些事件处理程序指定为事件标签属性的值。下面列出了 HTML5 规范定义的各种事件属性。

当任意事件发生在 HTML5 元素上时，下列属性可以用来触发任何作为值提供的 JavaScript 和 vbscript 代码。

这里我们只涵盖元素特定的事件，后面的章节会详细讨论这些元素。

属性	值	描述
offline	script	文档进入离线状态时触发。
onabort	script	事件中断时触发。
onafterprint	script	文档被打印后触发。
onbeforeunload	script	文档载入前触发。
onbeforeprint	script	文档被打印前触发。
onblur	script	窗口失去焦点时触发。
oncanplay	script	媒体停止缓冲，可以开始播放时触发。
oncanplaythrough	script	媒体可以播放到结束时触发，无需停止缓冲。
onchange	script	元素发生变化时触发。
onclick	script	鼠标点击触发。
oncontextmenu	script	上下文菜单被触发时触发。
ondblclick	script	双击鼠标时触发。
ondrag	script	元素被拖动时触发。
ondragend	script	拖拽操作结束时触发。
ondragenter	script	元素被拖拽到有效放置目标时触发。
ondragleave	script	元素离开有效放置目标时触发。
ondragover	script	元素被拖放到有效目标上时触发。
ondragstart	script	拖拽操作开始时触发。
ondrop	script	拖动的元素被放置时触发。
ondurationchange	script	媒体时长改变时触发。
onemptied	script	媒体资源元素突然清空时触发。
onended	script	媒体到达终点时触发。
onerror	script	发生错误时触发。
onfocus	script	窗口获得焦点时触发。
onformchange	script	表单变化时触发。

onforminput	script	表单获得用户输入时触发。
onhaschange	script	文档变化时触发。
oninput	script	元素获得用户输入时触发。
oninvalid	script	元素失效时触发。
onkeydown	script	键盘按下时触发。
onkeypress	script	键盘按下并释放时触发。
onkeyup	script	按键释放时触发。
onload	script	载入文档时触发。
onloadeddata	script	载入媒体数据时触发。
onloadedmetadata	script	媒体元素的媒体数据载入时触发。
onloadstart	script	浏览器开始载入媒体数据时触发。
onmessage	script	消息被触发时触发。
onmousedown	script	鼠标按键被按下时触发。
onmousemove	script	鼠标指针移动时触发。
onmouseout	script	鼠标指针移出元素时触发。
onmouseover	script	鼠标指针移入元素时触发。
onmouseup	script	鼠标按键释放时触发。
onmousewheel	script	鼠标滚轮转动时触发。
onoffline	script	文档进入离线状态时触发。
onoine	script	文档上线时触发。
ononline	script	文档上线时触发。
onpagehide	script	窗口隐藏时触发。
onpageshow	script	窗口变得可见时触发。
onpause	script	媒体数据暂停时触发。
onplay	script	媒体数据开始播放时触发。
onplaying	script	媒体数据播放时触发。
onpopstate	script	窗口历史信息改变时触发。
onprogress	script	浏览器获取媒体数据时触发。
onratechange	script	媒体数据的播放比率改变时触发。
onreadystatechange	script	ready-state 改变时触发。
onredo	script	文档执行 redo 操作时触发。
onresize	script	调整窗口尺寸时触发。
onscroll	script	元素的滚动条滚动时触发。
onseeked	script	媒体元素的 seeking 属性不在为真并结束时触发。
onseeking	script	媒体元素的 seeking 属性为真, seeking 开始时触发。
onselect	script	元素被选中时触发。
onstalled	script	获取媒体数据发生错误时触发。
onstorage	script	载入文档时触发。

onsubmit	script	表单提交时触发。
onsuspend	script	浏览器获取媒体数据，但获取整个媒体文件中止时触发。
ontimeupdate	script	媒体播放位置改变时触发。
onundo	script	文档执行 undo 操作时触发。
onunload	script	用户离开文档时触发。
onvolumechange	script	媒体音量发生变化，包括设置为“静音”时触发。
onwaiting	script	媒体停止播放，等待恢复时触发。



# HTML5 表单 2.0

Web 表单 2.0 就是 HTML4 表单特性的一个扩展。HTML5 中的表单元素和属性相比 HTML4 提供了更大程度的语义标记，移除了大量 HTML4 中需要的繁琐脚本和样式。

## HTML4 中的 <input> 元素

HTML4 输入框元素使用 `type` 属性指定数据类型。HTML4 提供了下列类型：

类型	描述
text	自由形式的文本字段，名义上没有换行符。
password	用于敏感信息的自由形式的文本字段，名义上没有换行符。
checkbox	预定义列表中的一组零个或多个值。
radio	一个枚举值。
submit	一个自由形式的启动表单的按钮。
file	带有 MIME 类型的任意文件以及可选的文件名。
image	一个坐标，相对于特定图片的尺寸，额外的语义是它必须是最后选中的值，同时启动表单提交。
hidden	默认不显示给用户的任意字符串。
select	枚举值，类似 radio 类型。
textarea	自由形式的文本字段，名义上没有换行的限制。
button	自由形式的按钮，可以启动按钮相关的任何事件。

下面是一个使用标注标签，单选按钮以及提交按钮的简单示例：

```
...
<form action="http://example.com/cgiscript.pl" method="post">
  <p>
    <label for="firstname">first name: </label>
    <input type="text" id="firstname"><br />
    <label for="lastname">last name: </label>
    <input type="text" id="lastname"><br />
    <label for="email">email: </label>
    <input type="text" id="email"><br>
    <input type="radio" name="sex" value="male"> Male<br>
    <input type="radio" name="sex" value="female"> Female<br>
    <input type="submit" value="send"> <input type="reset">
  </p>
</form>
...
```

## HTML5 中的 <input> 元素

除了上面提到的属性，HTML5 给输入框元素的 `type` 属性引入了几个新值。如下表所列。

注意： 请使用最新版的 **Opera** 浏览器运行下面所有例子。

类型	描述
<a href="#">datetime</a>	按照 ISO 8601 编码，时区设置为 UTC 的日期和时间（包括年，月，日，时，分，秒，分秒）。
<a href="#">datetime-local</a>	按照 ISO 8601 编码的日期和时间（包括年，月，日，时，分，秒，分秒），不带时区信息。
<a href="#">date</a>	按照 ISO 8601 编码的日期（包括年，月，日）。
<a href="#">month</a>	由 ISO 8601 编码的年和月组成的日期。
<a href="#">week</a>	由 ISO 8601 编码的年和星期数组成的日期。
<a href="#">time</a>	按照 ISO 8601 编码时间（包括时，分，秒，和分秒）。
<a href="#">number</a>	只接受数值。step 属性可以指定精度，默认为1。
<a href="#">range</a>	range 类型适用于应该包含某个范围内数值的输入字段。
<a href="#">email</a>	只接受邮箱值。这个类型适用于应该包含一个邮箱地址的输入字段。如果尝试提交一个简单的文本，它会强制要求输入 email@example.com 格式的邮箱地址。
<a href="#">url</a>	只接受 URL 值。这个类型适用于应该包含一个 URL 地址的输入字段。如果尝试提交一个简单的文本，它会强制要求输入 http://www.example.com 或者 http://example.com 格式的 URL 地址。

## <output> 元素

HTML5 还引入了一个新元素 `<output>`，用来表示不同类型的输出结果，比如输出由脚本所写。

还可以用 `for` 属性指定输出元素和文档中影响计算的其他元素之间的关系（比如，作为输入源或者参数）。`for` 属性的值是一个由空格分隔的其他元素的 IDs 列表。

便于学习这一概念 - 请进行[在线练习](#)。

## placeholder 属性

HTML5 引入了一个叫做 `placeholder` 的新属性。这个属性在 `<input>` 和 `<textarea>` 元素上为用户提供了在这个字段可以输入什么的提示。占位符字符不能包含回车符或者换行符。

下面是 `placeholder` 属性的简单语法：

```
<input type="text" name="search" placeholder="search the web"/>
```

这个属性只有最新版的 Mozilla, Safari 以及 Chrome 浏览器支持。

便于学习这一概念 - 请进行[在线练习](#)。

## required 属性

现在，我们不需要使用 JavaScript 处理诸如空文本框永远不能被提交的这类客户端验证了，因为 HTML5 引入了一个叫做 `required` 的新属性，可以按照如下方式使用，它会保证输入框有值：

```
<input type="text" name="search" required/>
```

这个属性只有最新版的 Mozilla, Safari 以及 Chrome 浏览器支持。

便于学习这一概念 - 请进行[在线练习](#)。

## HTML5 SVG 教程

---

SVG 表示可伸缩矢量图形，这是一门用于描述 2D 图形的语言，图形应用使用 XML 编写，然后 XML 由 SVG 阅读器程序呈现。

SVG 主要用于矢量类型的图表，比如饼图，X, Y 坐标系统中的二维图等等。

SVG 在 2003 年 1 月 14 日成为 W3C 推荐标准，你可以在 [SVG 规范](#) 页面中查看最新版本的 SVG 规范。

### 查看 SVG 文件

大多数 Web 浏览器都可以显示 SVG，就像它们可以显示 PNG, GIF 以及 JPG 图形。IE 用户可能需要安装 [Adobe SVG 阅读器](#) 以便能够在浏览器中查看 SVG。

### 在 HTML5 中嵌入 SVG

HTML5 允许我们直接使用 `<svg>...</svg>` 标签嵌入 SVG，下面是简单的语法：

```
<svg xmlns="http://www.w3.org/2000/svg">
...
</svg>
```

Firefox 3.7 还引入了一个配置选项（“about:config”），可以通过下列步骤启用 HTML5：

1. 在 Firefox 地址栏中输入 `about:config`。
2. 在出现警告消息的地方点击 “I’ll be careful, I promise!” 按钮（确保遵守它）。
3. 在页面顶部的过滤器中输入 `html5.enable`。
4. 默认可能被禁用了，因此要点击它切换它的值为 `true`。

现在，Firefox HTML5 解析器应该启用了，然后可以实验下面的例子。

### HTML5 – SVG 圆

下面是一个 SVG 示例的 HTML5 版本，用 `<circle>` 标签绘制一个圆：

```
<!DOCTYPE html>
<head>
<title>SVG</title>
```

```
<meta charset="utf-8" />
</head>
<body>
<h2>HTML5 SVG Circle</h2>
<svg id="svgelem" height="200" xmlns="http://www.w3.org/2000/svg">
  <circle id="redcircle" cx="50" cy="50" r="50" fill="red" />
</svg>
</body>
</html>
```

在启用 HTML5 的最新版 FireFox 中会生成如下结果：

## HTML5 SVG Circle



图片 1.1 HTML5 SVG Circle

便于学习这一概念 - 请使用 FireFox 3.7 或更高的版本进行[在线练习](#)。

## HTML5 - SVG 矩形

下面是一个 SVG 示例的 HTML5 版本，用 <rect> 标签绘制一个矩形：

```
<!DOCTYPE html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h2>HTML5 SVG Rectangle</h2>
<svg id="svgelem" height="200" xmlns="http://www.w3.org/2000/svg">
  <rect id="redrect" width="300" height="100" fill="red" />
</svg>
</body>
</html>
```

在启用 HTML5 的最新版 FireFox 中会生成如下结果：

## HTML5 SVG Rectangle



图片 1.2 HTML5 SVG Rectangle

便于学习这一概念 - 请使用 Firefox 3.7 或更高的版本进行[在线练习](#)。

### HTML5 - SVG 线条

下面是一个 SVG 示例的 HTML5 版本，用 `<line>` 标签绘制一个线条：

```
<!DOCTYPE html>
<html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h2>HTML5 SVG Line</h2>
<svg id="svgElem" height="200" xmlns="http://www.w3.org/2000/svg">
  <line x1="0" y1="0" x2="200" y2="100"
    style="stroke:red;stroke-width:2"/>
</svg>
</body>
</html>
```

你可以使用 `style` 属性给它设置额外的样式信息，比如笔画，填充色，笔画宽度等等。

在启用 HTML5 的最新版 Firefox 中会生成如下结果：

## HTML5 SVG Line



图片 1.3 HTML5 SVG Line

便于学习这一概念 - 请使用 Firefox 3.7 或更高的版本进行[在线练习](#)。

## HTML5 – SVG 椭圆

下面是一个 SVG 示例的 HTML5 版本，用 `<ellipse>` 标签绘制一个椭圆：

```
<!DOCTYPE html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h2>HTML5 SVG Ellipse</h2>
<svg id="svgelem" height="200" xmlns="http://www.w3.org/2000/svg">
  <ellipse cx="100" cy="50" rx="100" ry="50" fill="red" />
</svg>
</body>
</html>
```

在启用 HTML5 的最新版 FireFox 中会生成如下结果：



图片 1.4 HTML5 SVG Ellipse

便于学习这一概念 – 请使用 FireFox 3.7 或更高的版本进行[在线练习](#)。

## HTML5 – SVG 多边形

下面是一个 SVG 示例的 HTML5 版本，用 `<polygon>` 标签绘制一个多边形：

```
<!DOCTYPE html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h2>HTML5 SVG Polygon</h2>
<svg id="svgelem" height="200" xmlns="http://www.w3.org/2000/svg">
  <polygon points="20,10 300,20 170,50" fill="red" />
</svg>
</body>
</html>
```

在启用 HTML5 的最新版 FireFox 中会生成如下结果：

## HTML5 SVG Polygon



图片 1.5 HTML5 SVG Polygon

便于学习这一概念 - 请使用 Firefox 3.7 或更高的版本进行[在线练习](#)。

### HTML5 - SVG 折线

下面是一个 SVG 示例的 HTML5 版本，用 `<polyline>` 标签绘制一个折线图：

```
<!DOCTYPE html>
<html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h2>HTML5 SVG Polyline</h2>
<svg id="svgelem" height="200" xmlns="http://www.w3.org/2000/svg">
  <polyline points="0,0 0,20 20,20 20,40 40,40 40,60" fill="red" />
</svg>
</body>
</html>
```

在启用 HTML5 的最新版 Firefox 中会生成如下结果：

## HTML5 SVG Polyline



图片 1.6 HTML5 SVG Polyline

便于学习这一概念 - 请使用 Firefox 3.7 或更高的版本进行[在线练习](#)。

### HTML5 - SVG 渐变

下面是一个 SVG 示例的 HTML5 版本，用 `<ellipse>` 标签绘制一个椭圆，使用 `<radialGradient>` 标签定义一个 SVG 径向渐变。

我们可以以类似的方式用 `<linearGradient>` 标签创建 SVG 线性渐变。



```

<!DOCTYPE html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h2>HTML5 SVG Gradient Ellipse</h2>
<svg id="svgElem" height="200" xmlns="http://www.w3.org/2000/svg">
  <defs>
    <radialGradient id="gradient" cx="50%" cy="50%" r="50%"
      fx="50%" fy="50%">
      <stop offset="0%" style="stop-color:rgb(200,200,200);
        stop-opacity:0"/>
      <stop offset="100%" style="stop-color:rgb(0,0,255);
        stop-opacity:1"/>
    </radialGradient>
  </defs>
  <ellipse cx="100" cy="50" rx="100" ry="50"
    style="fill:url(#gradient)" />
</svg>
</body>
</html>

```

在启用 HTML5 的最新版 FireFox 中会生成如下结果：

## HTML5 SVG Gradient Ellipse



图片 1.7 HTML5 SVG Gradient Ellipse

便于学习这一概念 - 请使用 FireFox 3.7 或更高的版本进行[在线练习](#)。

## HTML5 MathML 教程

---

HTML5 的 HTML 语法允许我们在文档内使用 `<math>...</math>` 标签应用 MathML 元素。

大多数浏览器都能显示 MathML 标签。如果你的浏览器不支持 MathML，建议你使用最新版的 FireFox。

可以在 [MathML 2.0 规范](#) 页面查看 W3C 的 MathML 规范。

### MathML 示例

下面是一个使用 MathML 的有效 HTML5 文档：

```
html
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>Pythagorean theorem</title>
</head>
<body>
  <math xmlns="http://www.w3.org/1998/Math/MathML">
    <mrow>
      <msup><mi>a</mi><mn>2</mn></msup>
      <mo>+</mo>
      <msup><mi>b</mi><mn>2</mn></msup>
      <mo>=</mo>
      <msup><mi>c</mi><mn>2</mn></msup>
    </mrow>
  </math>
</body>
</html>
```

这会生成如下结果：

$a^2 + b^2 = c^2$

便于学习这一概念 - 请使用 FireFox 3.7 或更高版本进行[在线练习](#)。

## 使用 MathML 字符

想象一下，下面是一个使用字符 `&InvisibleTimes;` 的标记：

```
html
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>MathML Examples</title>
</head>
<body>
  <math xmlns="http://www.w3.org/1998/Math/MathML">
    <mrow>
      <mrow>
        <msup>
          <mi>x</mi>
          <mn>2</mn>
        </msup>
        <mo>+</mo>
        <mrow>
          <mn>4</mn>
          <mo></mo>
          <mi>x</mi>
        </mrow>
        <mo>+</mo>
        <mn>4</mn>
      </mrow>
      <mo>=</mo>
      <mn>0</mn>
    </mrow>
  </math>
</body>
</html>
```

这会生成如下结果。如果你不能看到  $x^2 + 4x + 4 = 0$  这样正确的结果，请使用 Firefox 3.5 或更高的版本。

$x^2 + 4x + 4 = 0$

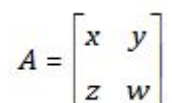
便于学习这一概念 - 请使用 Firefox 3.7 或更高版本进行[在线练习](#)。

## 矩阵表达示例

想象一下下面的例子，它会被用来表示一个简单的 2x2 矩阵：

```
html
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>MathML Examples</title>
</head>
<body>
  <math xmlns="http://www.w3.org/1998/Math/MathML">
    <mrow>
      <mi>A</mi>
      <mo>=</mo>
      <mfenced open="[" close="]">
        <mtable>
          <mtr>
            <mtd><mi>x</mi></mtd>
            <mtd><mi>y</mi></mtd>
          </mtr>
          <mtr>
            <mtd><mi>z</mi></mtd>
            <mtd><mi>w</mi></mtd>
          </mtr>
        </mtable>
      </mfenced>
    </mrow>
  </math>
</body>
</html>
```

这会生成如下结果。如果不能看到正确的结果，请使用 Firefox 3.7 或更高的版本。



$$A = \begin{bmatrix} x & y \\ z & w \end{bmatrix}$$

图片 1.8 2x2 matrix

便于学习这一概念 - 请使用 Firefox 3.7 或更高版本进行[在线练习](#)。

## HTML5 Web 存储

---

HTML5 引入了两种机制，类似于 HTTP 的会话 cookies，用于在客户端存储结构化数据以及克服以下缺点。

- 每个 HTTP 请求中都包含 Cookies，从而导致传输相同的数据减缓我们的 Web 应用程序。
- 每个 HTTP 请求中都包含 Cookies，从而导致发送未加密的数据到互联网上。
- Cookies 只能存储有限的 4KB 数据，不足以存储所需的数据。

这两种存储方式是 `session storage` 和 `local storage`，它们将用于处理不同的情况。

几乎所有最新版的浏览器都支持 HTML5 存储，包括 IE 浏览器。

### 会话存储

会话存储被设计用于用户执行单一事务的场景，但是同时可以在不同的窗口中执行多个事务。

#### 示例

比如，如果用户在同一网站的两个不同的窗口中购买机票。如果该网站使用 cookie 跟踪用户购买的机票，当用户在窗口中点击页面时，从一个窗口到另一个时当前已经购买的机票会“泄漏”，这可能导致用户购买同一航班的两张机票而没有注意到。

HTML5 引入了 `sessionStorage` 属性，这个网站可以用来把数据添加到会话存储中，用户仍然可以在打开的持有该会话的窗口中访问同一站点的任意页面，当关闭窗口时，会话也会丢失。

下面的代码将会设置一个会话变量，然后访问该变量：

```
<!DOCTYPE HTML>
<html>
<body>

<script type="text/javascript">
  if( sessionStorage.hits ){
    sessionStorage.hits = Number(sessionStorage.hits) +1;
  }else{
    sessionStorage.hits = 1;
  }
  document.write("Total Hits : " + sessionStorage.hits );
</script>
<p>Refresh the page to increase number of hits.</p>
<p>Close the window and open it again and check the result.</p>
```

```
</body>
</html>
```

便于学习上面的概念 - 请进行[在线练习](#)。

## 本地存储

本地存储被设计用于跨多个窗口进行存储，并持续处在当前会话上。尤其是，出于性能的原因 Web 应用程序可能希望在客户端存储百万字节的用户数据，比如用户撰写的整个文档或者是用户的邮箱。

Cookies 并不能很好的处理这种情况，因为每个请求都会传输。

### 示例

HTML5 引入了 *localStorage* 属性，可以用于访问页面的本地存储区域而没有时间限制，无论何时我们使用这个页面的时候本地存储都是可用的。

下面的代码设置了一个本地存储变量，每次访问这个页面时都可以访问该变量，甚至是下次打开窗口时：

```
<!DOCTYPE HTML>
<html>
<body>

  <script type="text/javascript">
    if( localStorage.hits ){
      localStorage.hits = Number(localStorage.hits) +1;
    }else{
      localStorage.hits = 1;
    }
    document.write("Total Hits : " + localStorage.hits );
  </script>
  <p>Refresh the page to increase number of hits.</p>
  <p>Close the window and open it again and check the result.</p>

</body>
</html>
```

便于学习上述概念 - 请进行[在线练习](#)。

## 删除 Web 存储

在本地机器上存储敏感数据可能是危险的，可能会留下安全隐患。

会话存储数据在会话终止之后将由浏览器立即删除。

要清除本地存储设置需要调用 `localStorage.remove('key')`；这个 'key' 就是我们想要移除的值对应的键。如果想要清除所有设置，需要调用 `localStorage.clear()` 方法。

下面的代码会完全清除本地存储：

```
<!DOCTYPE HTML>
<html>
<body>

  <script type="text/javascript">
    localStorage.clear();

    // Reset number of hits.
    if( localStorage.hits ){
      localStorage.hits = Number(localStorage.hits) +1;
    }else{
      localStorage.hits = 1;
    }
    document.write("Total Hits : " + localStorage.hits );
  </script>
  <p>Refreshing the page would not to increase hit counter.</p>
  <p>Close the window and open it again and check the result.</p>

</body>
</html>
```

便于学习上述概念 - 请进行[在线练习](#)。

## HTML5 Web SQL 数据库

---

Web SQL 数据库 API 并不是 HTML5 规范的一部分，但是它是一个独立的规范，引入了一组使用 SQL 操作客户端数据库的 APIs。

假定你是一个优秀的 Web 开发人员，如果是这样的话，毫无疑问你会很清楚 SQL 和 RDBMS 的概念。如果你仍然需要一个 SQL 的议题，可以学习我们的 [SQL 教程](#)。

我们可以在最新版的 Safari, Chrome 和 Opera 中使用 Web SQL 数据库。

### 核心方法

下面是规范中定义的三个核心方法。也会涵盖在本教程中：

1. **openDatabase**: 这个方法使用现有的数据库或者新建的数据库创建一个数据库对象。
2. **transaction**: 这个方法让我们能够控制一个事务，以及基于这种情况执行提交或者回滚。
3. **executeSql**: 这个方法用于执行实际的 SQL 查询。

### 开启数据库

如果数据库已经存在，*openDatabase* 方法负责开启数据库，如果不存在，这个方法会创建它。

使用下面的代码可以创建并开启一个数据库：

```
var db = openDatabase('mydb', '1.0', 'Test DB', 2 * 1024 * 1024);
```

上面的方法接受下列五个参数：

1. 数据库名称
2. 版本号
3. 描述文本
4. 数据库大小
5. 创建回调

最后也是第五个参数，创建回调会在创建数据库后被调用。然而，即使没有这个特性(功能)，运行时仍然会创建数据库以及正确的版本。



## 执行查询

执行查询需要使用 `database.transaction()` 函数。这个函数需要一个参数，它是一个负责实际执行查询的函数，如下所示：

```
var db = openDatabase('mydb', '1.0', 'Test DB', 2 * 1024 * 1024);
db.transaction(function (tx) {
    tx.executeSql('CREATE TABLE IF NOT EXISTS LOGS (id unique, log)');
});
```

上面的查询语句会在 'mydb' 数据库中创建一个叫做的 LOGS 的表。

## 插入操作

为了在表中创建条目，我们在上面的例子中加入简单的 SQL 查询，如下所示：

```
var db = openDatabase('mydb', '1.0', 'Test DB', 2 * 1024 * 1024);
db.transaction(function (tx) {
    tx.executeSql('CREATE TABLE IF NOT EXISTS LOGS (id unique, log)');
    tx.executeSql('INSERT INTO LOGS (id, log) VALUES (1, "foobar")');
    tx.executeSql('INSERT INTO LOGS (id, log) VALUES (2, "logmsg")');
});
```

创建条目时还可以传递如下所示的动态值：

```
var db = openDatabase('mydb', '1.0', 'Test DB', 2 * 1024 * 1024);
db.transaction(function (tx) {
    tx.executeSql('CREATE TABLE IF NOT EXISTS LOGS (id unique, log)');
    tx.executeSql('INSERT INTO LOGS
                    (id,log) VALUES (?, ?)', [e_id, e_log];
});
```

这里的 `e_id` 和 `e_log` 是外部变量，`executeSql` 会映射数组参数中的每个条目给 "?"。

## 读取操作

要读取已经存在的记录，我们可以使用回调来捕获结果，如下所示：

```
var db = openDatabase('mydb', '1.0', 'Test DB', 2 * 1024 * 1024);
db.transaction(function (tx) {
    tx.executeSql('CREATE TABLE IF NOT EXISTS LOGS (id unique, log)');
    tx.executeSql('INSERT INTO LOGS (id, log) VALUES (1, "foobar")');
    tx.executeSql('INSERT INTO LOGS (id, log) VALUES (2, "logmsg")');
});
db.transaction(function (tx) {
    tx.executeSql('SELECT * FROM LOGS', [], function (tx, results) {
        var len = results.rows.length, i;
        msg = "<p>Found rows: " + len + "</p>";
    });
});
```

```

document.querySelector('#status').innerHTML += msg;
for (i = 0; i < len; i++){
    alert(results.rows.item(i).log );
}
}, null);
});

```

## 最终示例

最后，然我们把这个例子放到如下所示的完整 HTML5 文档中，然后尝试在 Safari 浏览器中运行它：

```

<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
var db = openDatabase('mydb', '1.0', 'Test DB', 2 * 1024 * 1024);
var msg;
db.transaction(function (tx) {
    tx.executeSql('CREATE TABLE IF NOT EXISTS LOGS (id unique, log)');
    tx.executeSql('INSERT INTO LOGS (id, log) VALUES (1, "foobar")');
    tx.executeSql('INSERT INTO LOGS (id, log) VALUES (2, "logmsg")');
    msg = '<p>Log message created and row inserted.</p>';
    document.querySelector('#status').innerHTML = msg;
});

db.transaction(function (tx) {
    tx.executeSql('SELECT * FROM LOGS', [], function (tx, results) {
        var len = results.rows.length, i;
        msg = "<p>Found rows: " + len + "</p>";
        document.querySelector('#status').innerHTML += msg;
        for (i = 0; i < len; i++){
            msg = "<p><b>" + results.rows.item(i).log + "</b></p>";
            document.querySelector('#status').innerHTML += msg;
        }
    }, null);
});
</script>
</head>
<body>
<div id="status" name="status">Status Message</div>
</body>
</html>

```

在最新版的 Safari 或者 Opera 浏览器中这会生成如下所示结果：

```

Log message created and row inserted.

Found rows: 2

foobar

logmsg

```

便于学习这一概念 - 请使用最新版的 Safari 或者 Opera 进行[在线练习](#)。

## HTML5 服务器推送事件

---

传统的 Web 应用程序生成发送到 Web 服务器端的事件。比如，点击一个链接会从服务器请求一个新页面。

这种从 Web 浏览器到 Web 服务器的时间类型可以称作客户端事件。

随着 HTML5 的出现，WHATWG Web Applications 1.0 引入了一个从 Web 服务器到 Web 浏览器的事件流，被称作服务器推送事件（SSE）。使用 SSE 可以不停的将 DOM 事件推送到用户的浏览器中。

这个事件流方法会打开一个到服务器的持久连接，新消息可用时发送数据到客户端，从而不再需要持续的轮询。

### SSE Web 应用程序

要在 Web 应用程序中使用服务器推送事件，我们需要给文档添加一个 `<eventsource>` 元素。

`<eventsource>` 元素的 `src` 属性应该指向一个 URL，这个 URL 应该提供一个 HTTP 持久连接用于发送包含事件的数据流。

这个 URL 将会指向一个持续发送事件数据的 PHP，PERL 或者任意 Python 脚本。下面是一个简单的期望获得服务器时间的 Web 应用程序示例。

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
/* Define event handling logic here */
</script>
</head>
<body>
<div id="sse">
  <eventsource src="/cgi-bin/ticker.cgi" />
</div>
<div id="ticker">
  <TIME>
</div>
</body>
</html>
```

### SSE 服务器端脚本

服务器端脚本应该发送 `Content-type` 头指定类型为 `text/event-stream`，如下所示：

```
print "Content-Type: text/event-stream\n\n";
```

设置 Content-type 之后，服务器端脚本将发送一个后面紧跟事件名称的 **Event:** 标签。下面的示例将会发送一个以换行符结束的 **Server-Time** 作为事件名称。

```
print "Event: server-time\n";
```

最后一步是使用 **Data:** 标签发送事件数据，紧随其后的是以换行符结束的整数字符串值，如下所示：

```
$time = localtime();
print "Data: $time\n";
```

下面是用 perl 编写的完整 `ticker.cgi`：

```
#!/usr/bin/perl

print "Content-Type: text/event-stream\n\n";
while(true){
    print "Event: server-time\n";
    $time = localtime();
    print "Data: $time\n";
    sleep(5);
}
```

## 处理服务器推送事件

让我们修改一下我们的 Web 应用程序来处理服务器推送时间。下面是最终示例：

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
    document.getElementsByTagName("eventsources")[0].
        addEventListener("server-time", eventHandler, false);
    function eventHandler(event)
    {
        // Alert time sent by the server
        document.querySelector('#ticker').innerHTML = event.data;
    }
</script>
</head>
<body>
<div id="sse">
    <eventsources src="/cgi-bin/ticker.cgi" />
</div>
<div id="ticker" name="ticker">
    [TIME]
</div>
</body>
</html>
```

在测试服务器推送事件之前，建议你确保你的 Web 浏览器支持这一概念。

# HTML5 WebSockets 教程

Web Sockets 是用于 Web 应用程序的新一代双向通信技术，运行在单一套接字之上，它通过 JavaScript 接口暴露给 HTML5 兼容的浏览器中。

一旦取得 Web 服务器上的 Web Socket 连接之后，就可以通过调用 `send()` 方法从浏览器发送数据到服务器上，通过 `onmessage` 事件处理程序从服务器接收数据到浏览器中。

下面是创建一个新的 WebSocket 对象的 API。

```
var Socket = new WebSocket(url, [protocol] );
```

第一个参数 `url` 用于指定要连接的 URL。第二个属性 - 端口是可选的，如果提供，就会指定一个服务器必须支持连接成功的子协议。

## WebSocket 属性

下面是 WebSocket 对象的属性。假定我们已经创建了上述的 Socket 对象：

属性	描述
Socket.readyState	只读属性 <b>readyState</b> 表示连接的状态。有以下取值：  1. 0 表示连接尚未建立。  2. 1 表示连接已建立，可以进行通信。  3. 2 表示连接正在进行关闭握手。  4. 3 表示连接已经关闭或者连接不能打开。
Socket.bufferedAmount	只读属性 <b>bufferedAmount</b> 表示已经使用 <code>send()</code> 方法排队的 UTF-8 文本字节数。

## WebSocket 事件

下面是 WebSocket 对象相关的事件。假定我们已经创建了上述的 Socket 对象：

事件	事件处理程序	描述
open	Socket.onopen	建立 socket 连接时触发这个事件。
message	Socket.onmessage	客户端从服务器接收数据时触发。
error	Socket.onerror	连接发生错误时触发。
close	Socket.onclose	连接被关闭时触发。

## WebSocket 方法

下面是 WebSocket 对象相关的方法。假定我们已经创建了上述的 Socket 对象：

方法	描述
Socket.send()	send(data) 方法使用连接传输数据。
Socket.close()	close() 方法用于终止任何现有连接。

## WebSocket 示例

一个 WebSocket 就是客户端和服务端之间的标准双向 TCP 套接字。套接字以 HTTP 连接开始，在 HTTP 握手之后“升级”为 TCP 套接字。握手之后，任意一端都可以发送数据。

### 客户端 HTML 和 JavaScript 代码

编写这篇教程时，只有少数几个浏览器支持 WebSocket() 接口。你可以使用最新版的 Chrome, Mozilla, Opera 和 Safari 尝试下面这个例子。

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
function WebSocketTest()
{
  if ("WebSocket" in window)
  {
    alert("WebSocket is supported by your Browser!");
    // Let us open a web socket
    var ws = new WebSocket("ws://localhost:9998/echo");
    ws.onopen = function()
    {
      // Web Socket is connected, send data using send()
      ws.send("Message to send");
      alert("Message is sent...");
    };
  }
}
```

```

ws.onmessage = function (evt)
{
    var received_msg = evt.data;
    alert("Message is received...");
};
ws.onclose = function()
{
    // websocket is closed.
    alert("Connection is closed...");
};
}
else
{
    // The browser doesn't support WebSocket
    alert("WebSocket NOT supported by your Browser!");
}
}
</script>
</head>
<body>
<div id="sse">
    <a href="javascript:WebSocketTest()">Run WebSocket</a>
</div>
</body>
</html>

```

## 安装 pywebsocket

在测试上面的客户端程序之前，需要一个支持 WebSocket 的服务器。可以从 [pywebsocket](#) 下载 `mod_pywebsocket-x.x.x.tar.gz`，它只在为 Apache HTTP 服务器提供 WebSocket 扩展，然后按照如下步骤安装。

1. 解压缩和解压下载的文件
2. 进入 `pywebsocket-x.x.x/src/` 目录。
3. 执行 `$python setup.py build`
4. 执行 `$sudo python setup.py install`
5. 然后通过 `$pydoc mod_pywebsocket` 读取文档

这将会把他安装到我们的 python 环境中。

## 启动服务器

进入 `pywebsocket-x.x.x/src/mod_pywebsocket` 文件夹并运行如下命令：

```
$sudo python standalone.py -p 9998 -w ../example/
```

这会启动监听 9998 端口的服务器，然后使用通过 `-w` 选项指定的处理程序目录，也就是 `echo_wsh.py` 所在目录。

现在使用 Chrome 浏览器打开起初创建的 html 文件。如果浏览器支持 `WebSocket()`，那么会得到一个指示浏览器支持 `WebSocket` 的消息框，当我们点击 “Run `WebSocket`” 时会得到服务器脚本发出的 `Goodbye` 信息。



## HTML5 画布

HTML5 `<canvas>` 元素为我们使用 JavaScript 绘制图形提供了一种简单而又强大的方式。它可以用来绘制图表，制作摄影作品或者做一些简单（以及复杂）的动画。

这里有一个简单的 `<canvas>` 元素，除了所有核心的 HTML5 属性，比如 `id`，`name` 和 `class` 等等之外，它只有两个特定的属性 `width` 和 `height`。

```
<canvas id="mycanvas" width="100" height="100"></canvas>
```

使用 `_getElementById()` 方法很容易找到这个 `<canvas>` 元素，如下所示：

```
var canvas = document.getElementById("mycanvas");
```

我们来看一个在 HTML5 文档中使用 `<canvas>` 元素的简单示例。

```
<!DOCTYPE HTML>
<html>
<head>
<style>
#mycanvas{
    border:1px solid red;
}
</style>
</head>
<body>
    <canvas id="mycanvas" width="100" height="100"></canvas>
</body>
</html>
```

便于学习上述概念 - 请使用最新版的 Safari 或者 Opera 进行[在线练习](#)。

### 渲染上下文

`<canvas>` 初始为空，要显示某物，脚本首先需要访问渲染上下文，然后再上面绘图。

`canvas` 元素有一个叫做 `getContext` 的 DOM 方法，用于获得渲染上下文和它的绘图功能。这个函数接受一个参数，`2d` 上下文类型。

下面的代码就是访问需要的上下文以及检测浏览器是否支持 `<canvas>` 元素：

```
var canvas = document.getElementById("mycanvas");
if (canvas.getContext){
    var ctx = canvas.getContext('2d');
    // drawing code here
} else {
```

```
// canvas-unsupported code here
}
```

浏览器支持

最新版的 FireFox, Safari, Chrome 和 Opera 都支持 HTML5 Canvas, 但是 IE8 不支持原生 Canvas。

我们可以使用 [ExplorerCanvas](#) 让 IE 浏览器支持 Canvas。只需按照如下方式引入这个脚本即可：

```
<!--[if IE]><script src="excanvas.js"></script><![endif]-->
```

HTML5 Canvas 示例

本教程涵盖下列 HTML5 <canvas> 元素相关的示例。

示例	描述
<a href="#">绘制矩形</a>	学习如何使用 HTML5 <canvas> 元素绘制矩形。
<a href="#">绘制路径</a>	学习如何在 HTML5 <canvas> 元素中使用路径创建形状。
<a href="#">绘制线条</a>	学习如何使用 HTML5 <canvas> 元素绘制线条。
<a href="#">绘制贝塞尔曲线</a>	学习如何使用 HTML5 <canvas> 元素绘制贝塞尔曲线。
<a href="#">绘制二次曲线</a>	学习如何使用 HTML5 <canvas> 元素绘制二次曲线。
<a href="#">使用图像</a>	学习如何在 HTML5 <canvas> 元素中使用图像。
<a href="#">创建渐变</a>	学习如何使用 HTML5 <canvas> 元素创建渐变。
<a href="#">样式和颜色</a>	学习如何使用 HTML5 <canvas> 元素应用样式和颜色。
<a href="#">文本和字体</a>	学习如何使用不同的字体和尺寸绘制神奇的文字。

<a href="#">图案和投影</a>	学习如何绘制不同的图案和阴影。
<a href="#">画布状态</a>	学习如何在画布上做复杂绘图时保存和恢复画布状态。
<a href="#">画布平移</a>	这个方法用于移动画布和它原点到网格上的不同点。
<a href="#">画布旋转</a>	这个方法用于围绕当前原点旋转画布。
<a href="#">画布缩放</a>	这个方法用于增大或者减小画布网格中的单位。
<a href="#">画布变换</a>	这个方法允许我们直接修改变换矩阵。
<a href="#">画布合成</a>	这个方法用于从画布上屏蔽某些区域或者清除某一部分。
<a href="#">Canvas 动画</a>	学习如何使用 HTML5 画布和 JavaScript 创建基本的动画。

# HTML5 音频和视频

HTML5 特性，包括原生音频和视频支持而无需 Flash。

HTML5 <audio> 和 <video> 标签让我们给站点添加媒体变得简单。我们只需要设置 **src** 属性来识别媒体资源，包含 **controls** 属性让用户可以播放和暂停媒体。

## 嵌入视频

下面是在 Web 页面中嵌入视频文件最简单的形式：

```
<video src="foo.mp4" width="300" height="200" controls>
  Your browser does not support the <video> element.
</video>
```

目前的 HTML5 规范草案还没有指定浏览器应该在 video 标签中支持哪种视频格式。但是最常用的视频格式是：

- 1. **Ogg**: 带有 Theora 视频编码器和 Vorbis 音频编码器的 Ogg 文件。
- 2. **mpeg4**: 带有 H.264 视频编码器和 AAC 音频编码器的 MPEG4 文件。

我们可以使用带有媒体类型和其他属性的 <source> 标签指定媒体文件。video 元素允许使用多个 source 元素，浏览器会使用第一个认可的格式：

```
<!DOCTYPE HTML>
<html>
<body>
  <video width="300" height="200" controls autoplay>
    <source src="/html5/foo.ogg" type="video/ogg" />
    <source src="/html5/foo.mp4" type="video/mp4" />
    Your browser does not support the <video> element.
  </video>
</body>
</html>
```

便于学习上述概念 - 请使用最新版的 Safari 或 Opera 进行[在线练习](#)。

## Video 属性规范

HTML5 video 标签可以使用多个属性控制外观和感觉以及各种控制功能：

属性	描述
autoplay	如果指定这个布尔值属性，只要没有停止加载数据，视频就会立刻开始自动播放。
autobuffer	如果指定这个布尔值属性，即使没有设置自动播放，视频也会自动开始缓冲。

controls	如果指定这个属性，就允许用户控制视频播放，包括音量控制，快进，暂停或者恢复播放。
height	这个属性以 CSS 像素的形式指定视频显示区域的高度。
loop	如果指定这个布尔值属性，表示允许播放结束后自动回放。
preload	指定这个属性，视频会在载入页面时加载并准备就绪。如果指定自动播放则忽略。
poster	这是一个图像 URL，显示到用户播放或快进。
src	要嵌入的视频 URL。可选，可以在 video 块中使用 <source> 元素替代来指定要嵌入的视频。
width	这个属性以 CSS 像素的形式指定视频显示区域的宽度。

## 嵌入音频

HTML5 支持的 <audio> 标签用于在如下所示的 HTML 或 XHTML 文档中嵌入语音内容。

```
<audio src="foo.wav" controls autoplay>
  Your browser does not support the <audio> element.
</audio>
```

当前的 HTML 草案规范还没有指定浏览器应该在 audio 标签中支持哪种音频格式。但是最常用的音频格式是 ogg, mp3 和 wav。

我们可以使用带媒体类型以及其他属性的 <source> 标签指定媒体。Audio 元素允许使用多个 source 元素，并且浏览器会使用第一个认可的格式：

```
<!DOCTYPE HTML>
<html>
<body>
  <audio controls autoplay>
    <source src="/html5/audio.ogg" type="audio/ogg" />
    <source src="/html5/audio.wav" type="audio/wav" />
    Your browser does not support the <audio> element.
  </audio>
</body>
</html>
```

便于学习上述概念 - 请使用最新版的 Safari 或 Opera 进行[在线练习](#)。

### Audio 属性规范

HTML5 audio 标签可以使用多个属性来控制外观，感受以及各种控制功能：

属性	描述
autoplay	如果指定这个布尔值属性，只要没停止加载数据，音频就会立刻自动开始播放。
autobuffer	如果指定这个布尔值属性，即使没有设置自动播放，音频也会自动开始缓冲。
controls	如果指定这个属性，表示允许用户控制音频播放，包括音量控制，快进以及暂停/恢复播放。
loop	如果指定这个布尔值属性，表示允许音频播放结束后自动回放。

preload	这个属性指定加载页面时加载音频并准备就绪。如果指定自动播放则忽略。
src	要嵌入的音频 URL。可选，可以在音频块里面使用 <source> 元素指定要嵌入的音频来替代。

## 处理媒体事件

HTML5 audio 和 video 标签可以用多个属性利用 JavaScript 控制各种控制功能：

事件	描述
abort	播放中止时生成这个事件。
canplay	足够的的数据可用并且媒体可以播放时生成这个事件。
ended	播放完成时生成这个事件。
error	发生错误时生成这个事件。
loadeddata	媒体第一帧载入完成时生成这个事件。
loadstart	开始加载媒体时生成这个事件。
pause	播放暂停时生成这个事件。
play	播放开始或者恢复时生成这个事件。
progress	定期通知媒体下载进度时生成这个事件。
ratechange	播放速度改变时生成这个事件。
seeked	快进操作完成时生成这个事件。
seeking	快进操作开始时生成这个事件。
suspend	媒体加载被暂停时生成这个事件。
volumechange	音频音量变化时生成这个事件。
waiting	请求操作（比如播放）被延迟，等待另一个操作完成（比如快进）时生成这个事件。

下面是一个允许播放给定视频的示例：

```
<!DOCTYPE HTML>
<head>
<script type="text/javascript">
function PlayVideo() {
    var v = document.getElementsByTagName("video")[0];
    v.play();
}
</script>
</head>
<html>
<body>
    <form>
        <video width="300" height="200" src="/html5/foo.mp4">
            Your browser does not support the <video> element.
        </video>
        <input type="button" onclick="PlayVideo();" value="Play"/>
    </form>
</body>
</html>
```

便于学习上述概念 - 请使用最新版的 Safari 或 Opera 进行[在线练习](#)。

## 配置服务器媒体类型

大多数服务器默认都没使用正确的 MIME 类型提供 Ogg 或 mp4 媒体，因此我们可能需要添加适当的配置。

```
AddType audio/ogg .oga
AddType audio/wav .wav
AddType video/ogg .ogv .ogg
AddType video/mp4 .mp4
```

## HTML5 地理定位

HTML5 Geolocation API 允许我们对喜欢的网站共享我们的位置。JavaScript 可以捕获到纬度和经度，还可以发送给后端服务器，然后做一些位置感知的事情，比如查找本地企业或者在地图上显示我们的位置。

当前大多数浏览器和移动设备都支持 Geolocation API。地理定位 APIs 是作为全局 navigator 对象的一个新属性工作的。可以按照如下方式创建地理定位对象：

```
var geolocation = navigator.geolocation;
```

地理对象是一个允许组件检索设备地理位置相关信息的服务对象。

### Geolocation 方法

地理定位对象提供了下列方法：

方法	描述
<a href="#">getCurrentPosition()</a>	这个方法用于检索用户的当前地理位置。
<a href="#">watchPosition()</a>	这个方法用于检索设备当前地理位置定期更新信息。
<a href="#">clearWatch()</a>	这个方法用于取消 watchPosition 方法调用。

### 示例

下面是一个使用上述方法的示例代码：

```
function getLocation() {  
    var geolocation = navigator.geolocation;  
    geolocation.getCurrentPosition(showLocation, errorHandler);  
}
```

这里的 showLocation 和 errorHandler 分别是用来下一节会讲述的获取实际位置和处理错误的回调方法。



## Location 属性

地理定位方法 `getCurrentPosition()` 和 `getPositionUsingMethodName()` 指定了检索位置信息的回调方法。这些方法使用一个存储完整位置信息的 `Position` 对象异步调用。

这个 `Position` 对象指定了设备的当前地理位置。这个位置以一组带有方向和速度信息的地理坐标表示。

下面的表格描述了 `Position` 对象的属性。对于可选属性，如果系统没有提供值，则该属性值为 `null`。

属性	类型	描述
<code>coords</code>	<code>objects</code>	表示设备的地理位置。位置以一组带有方向和速度信息的地理坐标表示。
<code>coords.latitude</code>	<code>Number</code>	十进制的纬度估值。值范围为 <code>[-90.00, +90.00]</code> 。
<code>coords.longitude</code>	<code>Number</code>	十进制的经度固执。值范围为 <code>[-180.00, +180.00]</code> 。
<code>coords.altitude</code>	<code>Number</code>	<b>【可选】</b> WGS-84 球面以上的海拔高度固执，以米为单位计算。
<code>coords.accuracy</code>	<code>Number</code>	<b>【可选】</b> 以米为单位的纬度和经度精确估值。
<code>coords.altitudeAccuracy</code>	<code>Number</code>	<b>【可选】</b> 以米为单位的海拔高度精确估值。
<code>coords.heading</code>	<code>Number</code>	<b>【可选】</b> 相对正北方向设备以顺时针方向运动计算的当前方向。
<code>coords.speed</code>	<code>Number</code>	<b>【可选】</b> 以米/每秒为单位的设备当前地面速度。
<code>timestamp</code>	<code>date</code>	检索位置信息和创建 <code>Position</code> 对象的日期时间。

示例

下面是一个使用 Position 对象的示例代码。其中 showLocation 方法是一个回调方法：

```
function showLocation( position ) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;
    ...
}
```

处理错误

地理定位是复杂的。非常需要我们捕获任意错误并优雅的处理它。

地理定位方法 getCurrentPosition() 和 watchPosition() 可以使用一个提供 PositionError 对象的错误处理回调方法。这个对象有下列两属性。

属性	类型	描述
code	Number	错误码。
message	String	错误描述信息。

下面这个表格描述了 PositionError 对象可能返回的错误码。

错误码	常量	描述
0	UNKNOWN_ERROR	由于未知错误，检索设备位置信息失败。
1	PERMISSION_DENIED	由于应用程序没有权限使用位置服务，检索设备位置信息失败。
2	POSITION_UNAVAILAB LE	设备位置信息无法确定。
3	TIMEOUT	不能在给定的最大超时区间内检索位置信息。

示例

下面是一个使用 `PositionError` 对象的示例代码。其中 `errorHandler` 方法是一个回调方法：

```
function errorHandler( err ) {
  if (err.code == 1) {
    // access is denied
  }
  ...
}
```

Position 选项

下面是 `getCurrentPosition()` 方法的实际语法：

```
getCurrentPosition(callback, errorCallback, options)
```

其中第三个参数是指定一组检索设备地理位置选项的 `PositionOptions` 对象。

下列选项可以指定为第三个参数：

属性	类型	描述
<code>enableHighAccuracy</code>	Boolean	是否想要检索最精准的位置估值。默认值为 <code>false</code> 。
<code>timeout</code>	Number	<code>timeout</code> 属性就是 Web 应用程序要等待定位的毫秒数。
<code>maximumAge</code>	Number	用于缓存位置信息的过期时间毫秒数。

示例

下面这个示例代码展示了如何使用上面提到的方法：

```
function getLocation() {
  var geolocation = navigator.geolocation;
  geolocation.getCurrentPosition(showLocation, errorHandler, {maximumAge: 75000});
}
```

# HTML5 微数据

微数据是一种在网页中提供附加语义的标准化方式。

微数据允许我们定义自定义元素以及在网页中嵌入自定义属性。从较高的角度而言，微数组由一组名-值对组成。

这个分组被称作条目，每个名-值对就是一个属性。条目和属性使用普通元素表示。

## 示例

- 用 `itemscope` 属性创建一个条目。
- 给条目添加一个属性，`itemprop` 属性用于条目的后代。

这里有两个条目，其中每个条目都有一个 “name” 属性：

```
<div itemscope>
  <p>My name is <span itemprop="name">Zara</span>.</p>
</div>

<div itemscope>
  <p>My name is <span itemprop="name">Nuha</span>.</p>
</div>
```

属性值通常是字符串，但是它可以是下列数据类型。

## 全局属性

微数据引入了五个全局属性，这些属性适用于在任意元素以及为数据提供上下文机制。

属性	描述
itemscope	用于创建一个条目。itemscope 属性是一个布尔值属性，说明页面上有微数据以及它从哪里开始。
itemtype	这个属性是一个有效的 URL，用于定义条目以及为属性提供上下文。
itemid	这个属性是条目的全局标识符。
itemprop	这个属性为条目定义属性。
itemref	这个属性提供了一个附加元素列表来抓取条目的名-值对。

## 属性数据类型

属性通常有一个正如上面的例子中提到的字符串值，但是它们的值也可以是 URLs。下面的例子中有一个 “image” 属性，它的值是一个 URL：

```
<div itemscope>
  
</div>
```

属性的值也可以是日期，时间或者日期和时间。下面是一个使用 `time` 元素和 `datetime` 属性的实现。

```
<div itemscope>
My birthday is:
<time itemprop="birthday" datetime="1971-05-08">
  Aug 5th 1971
</time>
</div>
```

属性本身也可以是一组名-值对，通过在元素上放置 `itemscope` 属性来声明属性。

## 微数据 API 支持

如果浏览器支持 HTML5 微数据 API，那么全局 `document` 对象上就会有一个 `getItems()` 函数。如果浏览器不支持微数据，`getItems()` 函数就会是 `undefined`。

```
function supports_microdata_api() {
  return !!document.getItems;
}
```

Modernizr 还不支持微数据 API 检测，因此我们需要使用像上面一样列出的函数来检测。

HTML5 微数据标准包含 HTML 标记（主要用于搜索引擎）和一系列 DOM 函数（主要用于浏览器）。

我们可以在网页中引入微数据标记，不理解微数据属性的搜索引擎将会忽略它们。但是，如果需要通过 DOM 访问或者操作微数据，我们还需要检查浏览器是否支持微数据 DOM API。

## 定义微数据词汇表

要定义一个微数据词汇表我们需要一个只想有效网页的命名空间 URL。例如 `http://data-vocabulary.org/Person` 可以用作带下列命名属性的个人微数据词汇表的命名空间。

- `name` - 人名，简单的字符串值。
- `Photo` - 指向人物照片的 URL。

- URL – 属于个人的网站。

下面是一个使用个人微数据相关属性的例子：

```
<section itemscope itemtype="http://data-vocabulary.org/Person">
<h1 itemprop="name">Andy Runie</h1>
<p>

</p>
<a itemprop="url" href="http://www.example.com/blog">My Blog</a>
</section>
```

Google 支持将微数据作为他们 Rich Snippets 程序的一部分。当 Google 的网页爬虫解析我们的页面并发现符合 `http://data-vocabulary.org/Person` 词汇表的微数据属性时，它会解析出这些属性并把他们存储到其他页面数据的旁边。

你可以利用 [Rich Snippets 测试工具](http://www.tutorialspoint.com/html5/microdata.htm) 使用 `http://www.tutorialspoint.com/html5/microdata.htm` 测试上面的例子。

## HTML5 拖放

拖放（DnD）是一个强大的用户界面相关的概念，借助鼠标点击的帮助，它让复制，重新排序以及删除条目变得很容易。这就允许用户在某个元素上点击并按住鼠标按键，把它拖到另外一个位置，然后释放鼠标按键把与元素放到该位置。

要使用传统的 HTML4 实现拖放的功能，开发者要么使用复杂的 JavaScript 变成，要么使用 JavaScript 框架，比如 jQuery 等等。

现在 HTML5 提出了拖放（DnD）API，为浏览器带来了原生拖放支持，让编码变得更容易。

所有的主流浏览器比如 Chrome, FireFox 3.5 以及 Safari 4 等等都支持 HTML5 拖放。

### Drag 和 Drop 事件

下面列出了一些在执行拖放操作各个阶段会触发的事件：

事件	描述
dragstart	用户开始拖动对象时触发。
dragenter	鼠标初次移到目标元素并且正在进行拖动时触发。这个事件的监听器应该指出这个位置是否允许放置元素。如果没有监听器或者监听器不执行任何操作，默认情况下不允许放置。
dragover	拖动时鼠标移到某个元素上的时候触发。大多数时候，监听器触发的操作与 dragenter 事件相同。
dragleave	拖动时鼠标离开某个元素的时候触发。监听器应该移除用于放置反馈的高亮或插入标记。
drag	对象被拖拽时每次鼠标移动都会触发。
drop	拖动操作结束，放置元素时触发。监听器负责检索被拖动的数据以及在放置位置插入它。
dragend	拖动对象时用户释放鼠标按键的时候触发。

注意：只会触发拖动事件；拖动操作期间鼠标事件，比如 `mousemove` 并不会触发。

DataTransfer 对象

所有 drag 和 drop 事件的事件监听器都接收一个 Event 对象作为参数，它有一个叫做 dataTransfer 的只读属性。event.dataTransfer 返回与事件相关的 DataTransfer 对象，如下所示：

```
function EnterHandler(event) {
    DataTransfer dt = event.dataTransfer;
    .....
}
```

DataTransfer 对象持有 drag 和 drop 操作相关的数据。可以检索这些数据以及设置 DataTransfer 对象相关联的各种属性，正如下面所阐述的：

S. N. DataTransfer 属性及其描述	
1	<div>dataTransfer.dropEffect [ = value ]</div> <div><ul style="list-style-type: none"><li>• 返回当前选中的操作类型。</li><li>• 这个属性也可以设置改变当前选中的操作。</li><li>• 可选值为none，copy，link和move。</li></ul></div>
2	<div>dataTransfer.effectAllowed [ = value ]</div> <div><ul style="list-style-type: none"><li>• 返回允许的操作类型。</li><li>• 这个属性也可以设置改变允许的操作。</li><li>• 可选值为none，copy，copyLink，copyMove，link，linkMove，move，all和uninitialized。</li></ul></div>
3	<div>dataTransfer.types</div> <div>返回列出设置给 dragstart 事件格式的 DOMStringList。此外，如果任意文件被拖拽，那么其中一个类型将会是字符串“Files”。</div>
4	<div>dataTransfer.clearData( [ format ] )</div> <div>移除指定格式的数据。如果省略参数则移除所有数据。</div>
5	<div>dataTransfer.setData(format, data)</div> <div>添加给定的数据。</div>



6	<code>data = dataTransfer.getData(format)</code>  返回指定的数据。如果没有该数据则返回空字符串。
7	<code>dataTransfer.files</code>  如果有，返回表示被拖拽文件的 <code>FileList</code> 。
8	<code>dataTransfer.setDragImage(element, x, y)</code>  使用给定的元素更新拖拽反馈信息，替换先前指定的反馈信息。
9	<code>dataTransfer.addElement(element)</code>  把给定的元素添加到用来渲染拖拽反馈的元素列表。

## Drag 和 Drop 过程

下面是实现拖拽操作的步骤：

### 步骤1：创建一个可拖拽对象

下面是要采用的步骤：

- 如果想要拖动某个元素，需要设置元素的 `draggable` 属性为 `true`。
- 给 `dragstart` 设置一个事件监听器存储拖拽数据。
- 事件监听器 `dragstart` 会设置允许的效果（`copy`，`move`，`link`或者是组合形式的）。

下面是一个让对象可拖拽的示例：

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
#boxA, #boxB {

    float:left;padding:10px;margin:10px; -moz-user-select:none;
}
#boxA { background-color: #6633FF; width:75px; height:75px; }

#boxB { background-color: #FF6699; width:150px; height:150px; }

</style>
<script type="text/javascript">
function dragStart(ev) {
    ev.dataTransfer.effectAllowed='move';
```

```

    ev.dataTransfer.setData("Text", ev.target.getAttribute('id'));
    ev.dataTransfer.setDragImage(ev.target, 0, 0);
    return true;
}
</script>
</head>
<body>
<center>
<h2>Drag and drop HTML5 demo</h2>
<div>Try to drag the purple box around.</div>

<div id="boxA" draggable="true"
      ondragstart="return dragStart(event)">
  <p>Drag Me</p>
</div>
<div id="boxB">Dustbin</div>
</center>
</body>
</html>

```

便于学习上述概念 - 请使用最新版的 FireFox, Safari 或 Chrome 进行[在线练习](#)。

## 步骤2: 放置对象

为了接受放置, 放置目标至少要监听三个事件。

- **dragenter** 事件, 用来确定放置目标是否接受放置。如果放置被接受, 那么这个事件必须取消。
- **dragover** 事件, 用来确定给用户显示怎样的反馈信息。如果这个事件被取消, 反馈信息 (通常就是光标) 就会基于 **dropEffect** 属性的值更新。
- 最后是 **drop** 事件, 允许执行真正的放置。

下面是把一个对象放入另一个对象的示例:

```

<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
#boxA, #boxB {

    float:left;padding:10px;margin:10px;-moz-user-select:none;
}
#boxA { background-color: #6633FF; width:75px; height:75px; }

#boxB { background-color: #FF6699; width:150px; height:150px; }

</style>
<script type="text/javascript">
function dragStart(ev) {
    ev.dataTransfer.effectAllowed='move';
    ev.dataTransfer.setData("Text", ev.target.getAttribute('id'));
    ev.dataTransfer.setDragImage(ev.target, 0, 0);
    return true;
}
function dragEnter(ev) {
    event.preventDefault();

```

```

    return true;
}
function dragOver(ev) {
    return false;
}
function dragDrop(ev) {
    var src = ev.dataTransfer.getData("Text");
    ev.target.appendChild(document.getElementById(src));
    ev.stopPropagation();
    return false;
}
</script>
</head>
<body>
<center>
<h2>Drag and drop HTML5 demo</h2>
<div>Try to move the purple box into the pink box.</div>

<div id="boxA" draggable="true"
    ondragstart="return dragStart(event)">
    <p>Drag Me</p>
</div>
<div id="boxB" ondragenter="return dragEnter(event)"
    ondrop="return dragDrop(event)"
    ondragover="return dragOver(event)">Dustbin</div>
</center>
</body>
</html>

```

便于学习上述概念 – 请使用最新版的 Firefox, Safari 或 Chrome 进行[在线练习](#)。

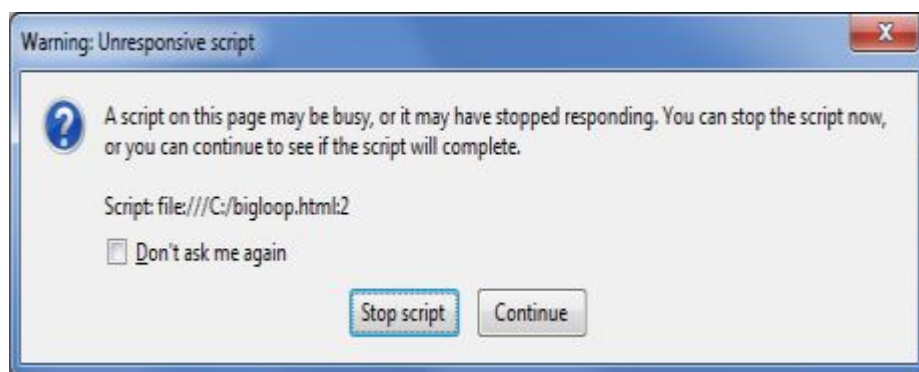
## HTML5 Web Workers

JavaScript 被设计为运行在一个单线程环境中，这意味着多个脚本不能同时运行。考虑这样一种情况，我们需要处理 UI 事件，查询和处理大量的 API 数据以及操作 DOM。

在 CPU 使用率过高的情况下 JavaScript 还会造成浏览器假死。我们来举一个简单的例子，用 JavaScript 运行一个大循环：

```
<!DOCTYPE HTML>
<html>
<head>
<title>Big for loop</title>
<script>
  function bigLoop() {
    for (var i = 0; i <= 10000000000; i += 1) {
      var j = i;
    }
    alert("Completed " + j + "iterations" );
  }
  function sayHello() {
    alert("Hello sir...." );
  }
</script>
</head>
<body>
  <input type="button" onclick="bigLoop();" value="Big Loop" />
  <input type="button" onclick="sayHello();" value="Say Hello" />
</body>
</html>
```

尝试运行在线示例来查看其结果 — 可以使用任何浏览器运行[在线示例](#)。当我们在 Firefox 中点击 *Big Loop* 按钮是它会显示如下所示结果：



图片 1.9 big loop

## 什么是 Web Workers?

上述情况可以使用 Web Workers 处理，它可以处理所有昂贵的计算任务而不阻塞用户界面，它通常运行在单独的线程上。

Web Workers 允许长时间运行脚本，而不阻塞脚本响应点击或者其他用户交互；它还允许执行长期任务而无需页面保持响应。

Web Workers 就是背景脚本，它们是相对重量级的，并不打算被大量使用。比如，它并不适合为一个四百万像素图像的每一个像素启动一个 worker。

当脚本执行在 Web Worker 中时，它不能访问 Web 页面的 window 对象（window.document），这意味着这个 Web Workers 不必直接访问该页面和 DOM API。尽管 Web Workers 不能阻塞浏览器 UI，但它们仍然会消耗 CPU 周期，降低系统的响应速度。

## Web Workers 的工作原理

Web Workers 使用 JavaScript 文件 URL 初始化，这个文件包含要在 worker 中执行的代码。这个代码可以设置事件监听器，与引入它的主页面进行通信。下面是简单的语法：

```
var worker = new Worker('bigLoop.js');
```

如果指定的 JavaScript 文件已经存在，浏览器会启动一个新异步下载的 worker 线程。如果 worker 路径返回 404，那么这个 worker 会静默失败。

如果你的应用程序要支持多个 JavaScript 文件，你可以将它们导入到 `importScripts()` 方法中，这个方法接收逗号分割的文件名作为参数，如下所示：

```
importScripts("helper.js", "anotherHelper.js");
```

一旦 Web Worker 启动，就可以使用 `postMessage()` 方法在 web worker 和父页面之间进行通信。依赖于浏览器或浏览器版本，`postMessage()` 方法可以接受一个字符串或者 JSON 对象作为它的唯一参数。

通过 Web Worker 攒底的消息可以在主页面中使用 `onmessage` 事件访问。现在让我们用 Web Worker 编写上面的 bigLoop 示例。下面是要启动 web worker 执行循环和返回变量 `j` 最终值的主页面（hello.htm）。

```
<!DOCTYPE HTML>
<html>
<head>
<title>Big for loop</title>
<script>
    var worker = new Worker('bigLoop.js');
```

```

    worker.onmessage = function (event) {
        alert("Completed " + event.data + "iterations" );
    };

    function sayHello(){
        alert("Hello sir...." );
    }
</script>
</head>
<body>
    <input type="button" onclick="sayHello();" value="Say Hello"/>
</body>
</html>

```

下面是 bigLoop.js 文件的内容。这里使用 `postMessage()` API 来传递通信信息给主页面。

```

for (var i = 0; i <= 10000000000; i += 1){
    var j = i;
}
postMessage(j);

```

接下来我们把 hello.htm 和 bigLoop.js 文件放入同一个目录中，然后使用最新版的 Safari 或 FireFox 访问 hello.htm。

也可以在线查看上述示例的结果 - 可以使用最新版的 Safari 或 FireFox [在线尝试](#)。

## 停止 Web Workers

Web Workers 不能自行停止，但是启动它们的页面可以通过调用 `terminate()` 方法停止它们。

```
worker.terminate();
```

被终止的 Web Worker 将不再响应消息或者执行任何附加的计算。我们并不能重启 worker；但是，可以使用同一 URL 创建一个新的 worker。

## 错误处理

下面的代码展示了一个在 Web Worker JavaScript 文件中错误处理函数输出错误日志到控制台的例子。这个带有错误处理代码的例子如下所示：

```

<!DOCTYPE HTML>
<html>
<head>
<title>Big for loop</title>
<script>
    var worker = new Worker('bigLoop.js');
    worker.onmessage = function (event) {
        alert("Completed " + event.data + "iterations" );
    };

```

```

    worker.onerror = function (event) {
        console.log(event.message, event);
    };

    function sayHello() {
        alert("Hello sir...." );
    }
</script>
</head>
<body>
    <input type="button" onclick="sayHello();" value="Say Hello"/>
</body>
</html>

```

## 浏览器支持检测

下面的代码可用于浏览器中检测是否支持 Web Worker 特性：

```

<!DOCTYPE HTML>
<html>
<head>
<title>Big for loop</title>
<script src="/js/modernizr-1.5.min.js"></script>
<script>
    if (Modernizr.webworkers) {
        alert("Congratulation!! you have web workers support." );
    }else{
        alert("Sorry!! you do not have web workers support." );
    }
</script>
</head>
<body>
    <p>Checking for Browser Support for web workers</p>
</body>
</html>

```

可以尝试在线示例查看结果 - 请使用不同的浏览器运行[在线示例](#)。



2

## HTML5 标签参考





## HTML5 标签参考

---

下面提供了一个 HTML5 标准标签的完整列表。所有的标签按照字母顺序排列，新引入或者 HTML5 中过时的标签都带有一个指示。

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>	<b>不赞成使用：</b> 定义首字母缩写。
<address>	定义一个 address 元素。
<applet>	<b>不赞成使用：</b> 定义一个 applet。
<area>	在图像映射内定义区域。
<article>	<b>新标签：</b> 定义一个独立的文档内容块，比如博客条目或报纸上的文章。
<aside>	<b>新标签：</b> 定义一个与页面其他部分相关的内容块。
<audio>	<b>新标签：</b> 指定一个音频文件。
<base>	为页面中所有链接指定一个基准 URL。
<basefont>	<b>不赞成使用：</b> 指定一个基准字体。
<bdo>	定义文本的显示方向。
<bgsound>	定义背景音乐。
<blink>	定义闪烁文本。
<blockquote>	定义长引用。
<body>	定义一个主体元素。
 	插入一个换行符。
<button>	定义一个按钮。
<canvas>	<b>新标签：</b> 用于在运行时渲染动态的位图图形，比如图表或游戏。
<caption>	定义表格标题。
<center>	<b>不赞成使用：</b> 定义居中文本。
<col>	定义表格的列属性。
<colgroup>	定义表格列组。
<command>	<b>新标签：</b> 定义一个用户可以调用的命令。
<comment>	在文档中植入一个注释。
<datalist>	<b>新标签：</b> 连同新的用于输入的列表属性可以制作下拉列表框。
<dd>	指定一个定义列表的描述。
<del>	定义一个被删除的文本。

标签	描述
<details>	<b>新标签：</b> 定义用户可以按需获取的附加信息或控件。
<dir>	<b>不赞成使用：</b> 定义一个目录列表。
<div>	在文档中定义一个节。
<dl>	定义一个定义列表。
<dt>	定义一个定义列表项。
<embed>	<b>新标签：</b> 定义一个外部交互内容或者插件。
<fieldset>	定义一个表单控件组。
<figure>	<b>新标签：</b> 指定一块自包含的内容流，通常被主文档流用作独立的单位引用。
<b>	定义加粗文本。
<big>	<b>不赞成使用：</b> 定义大号文本。
<i>	定义斜体文本。
<small>	定义小号文本。
<tt>	<b>不赞成使用：</b> 定义打字机文本。
<font>	<b>不赞成使用：</b> 定义文本的字体，尺寸和颜色。
<footer>	给 section 定义页脚，可以包含作者信息，版权信息等等。
<form>	定义一个表单。
<frame>	<b>不赞成使用：</b> 定义子窗口（或框架）。
<frameset>	<b>不赞成使用：</b> 定义框架集。
<head>	定义关于文档的信息。
<header>	<b>新标签：</b> 定义一组介绍或者导航信息。
<hgroup>	<b>新标签：</b> 定义节的页眉。
<h1> to <h6>	定义 1 到 6 号标题。
<hr>	定义水平分割线。
<html>	定义一个 html 文档。
<isindex>	<b>不赞成使用：</b> 定义一个单行输入字段。
<iframe>	定义一个内联子窗口（框架）。
<ilayer>	定义内联层。
<img>	定义图像。
<input>	定义输入字段。
<ins>	定义被插入的文本。
<keygen>	<b>新标签：</b> 定义生成密钥的控件。
<keygen>	在表单中生成密钥信息。
<label>	为表单控件定义标注。
<layer>	定义一个层。
<legend>	在 fieldset 中定义标题。
<li>	定义列表项。
<link>	定义资源引用。

标签	描述
<map>	定义图像映射。
<mark>	<b>新标签：</b> 在文档中标记或高亮文本用于引用参考，表示它在另一个上下文中的相关性。
<marquee>	创建一个滚动文本字幕。
<menu>	<b>不赞成使用：</b> 定义菜单列表。
<meta>	定义元信息。
<meter>	<b>新标签</b> 定义度量，比如磁盘使用率。
<multicol>	定义多列文本流。
<nav>	<b>新标签：</b> 定义用于文档导航的节。
<nobr>	不允许在闭合文本中折行。
<noembed>	定义不支持 <embed> 标签的浏览器中呈现的内容。
<noframes>	<b>不赞成使用：</b> 定义不支持框架的替代内容。
<noscript>	定义不支持客户端脚本的替代内容。
<object>	定义内嵌对象。
<ol>	定义有序列表。
<optgroup>	定义选项组合。
<option>	定义下拉列表选项。
<output>	<b>新标签：</b> 定义输出类型，比如根据脚本计算。
<p>	定义段落。
<param>	定义对象的参数。
<cite>	定义引用。
<code>	定义计算机代码文本。
<dfn>	定义定义项目。
<em>	定义强调文本。
<kbd>	定义键盘文本。
<samp>	定义计算机代码样本。
<strong>	定义强调文本。
<var>	定义变量。
<plaintext>	<b>不赞成使用：</b> 渲染文档提示作为预格式文本。
<pre>	定义预格式文本。
<progress>	<b>新标签：</b> 定义任务进度，比如下载或者执行了一系列昂贵的操作时。
<q>	定义短引用。
<ruby>	<b>新标签：</b> 和 <rt> 以及 <rp> 一起使用，可用于标记 ruby 注释。
<script>	定义脚本。
<section>	<b>新标签：</b> 表示通用的文档或应用的节。
<select>	定义选择列表。
<spacer>	定义空白。
<span>	定义文档中的节。

标签	描述
<s>	不赞成使用： 定义加删除线的文本。
<strike>	不赞成使用： 定义加删除线文本。
<style>	定义样式定义。
<sub>	定义下标文本。
<sup>	定义上标文本。
<table>	定义表格。
<tbody>	定义表格主体。
<td>	定义表格单元。
<textarea>	定义文本输入区域。
<tfoot>	定义表格脚注。
<th>	定义表格表头单元。
<thead>	定义表格表头。
<time>	新标签： 定义日期/时间。
<title>	定义文档的标题。
<tr>	定义表格的行。
<u>	不赞成使用： 定义下划线文本。
<ul>	定义无序列表。
<video>	新标签： 定义视频文件。
<wbr>	新标签： 定义换行位置。
<wbr>	在 <nobr> 中指出潜在的单词换行点。
<xmp>	不赞成使用： 定义预格式文本。

## HTML5 过时标签和属性

### 过时标签

下列元素在 HTML5 中不再可用，其功能可以通过 CSS 更好的处理：

标签（元素）	描述
<acronym>	定义首字母缩写。
<applet>	定义嵌入的 applet。
<basefont>	定义页面基准字体。
<big>	定义大号文本。
<center>	定义居中文本。
<dir>	定义目录列表。
<font>	定义文本字体、尺寸和颜色。
<frame>	定义框架。
<frameset>	定义框架集。
<isindex>	定义单行输入字段。
<noframes>	定义不支持框架的替代内容。
<s>	定义加删除线的文本。
<strike>	定义加删除线的文本。
<tt>	定义打字机文本。
<u>	定义下划线文本。

### 过时属性

HTML5 中不再有 HTML4 中的外观属性，因为它们的功能可以通过 CSS 更好的处理。有些来自 HTML4 的属性不在允许出现在 HTML5 中，甚至它们已经被完全删除了。

下面的表格中是已经被移除的属性以及它们所影响的标签（元素）。其中一些属性已经被永久删除。

被移除的属性	所属元素
rev	link, a
charset	link and a
shape	a
coords	a
longdesc	img and iframe.

target	link
nohref	area
profile	head
version	html
name	img
scheme	meta
archive	object
classid	object
codebase	object
codetype	object
declare	object
standby	object
valuetype	param
type	param
axis	td and t
abbr	td and t
scope	td
align	caption, iframe, img, input, object, legend, table, hr, div, h1, h2, h3, h4, h5, h6, p, col, colgroup, tbody, td, tfoot, th, thead and tr.
alink	body
link	body
vlink	body
text	body
background	body
bgcolor	table, tr, td, th and body.
border	table and object.
cellpadding	table
cellspacing	table
char	col, colgroup, tbody, td, tfoot, th, thead and tr.
charoff	col, colgroup, tbody, td, tfoot, th, thead and tr.
clear	br
compact	dl, menu, ol and ul.
frame	table
compact	dl, menu, ol and ul.
frame	table
frameborder	iframe
hspace	img and object.

vspace	img and object.
marginheight	iframe
marginwidth	iframe
noshade	hr
nowrap	td and th
rules	table
scrolling	iframe
size	hr
type	li, ol and ul.
valign	col, colgroup, tbody, td, tfoot, th, thead and tr
width	hr, table, td, th, col, colgroup and pre.

## HTML5 新标签（元素）

下面是标签（元素）已被 HTML5 引入：

标签（元素）	描述
<article>	表示文档内容的一个独立片段， 比如博客条目或报纸上的文章。
<aside >	表示与页面其他部分略微相关的内容片段。
<audio>	定义一个音频文件。
<canvas>	用于在运行时渲染动态位图图形，比如图表和游戏。
<command>	表示用户可调用的命令。
<datalist>	与输入框的 list 属性一起使用可用来创建组合框。
<details>	表示用户可以按需获取的额外信息或控件。
<embed>	定义外部交互内容或插件。
<figure>	表示一块自包含的内容流，通常被作为主文档流的独立单元引用。
<footer>	表示一个节的脚注，可以包含作者，版权信息等内容。
<header>	表示一组介绍性或者导航性的组合。
<hgroup>	表示一个节的头部。
<keygen>	表示生成密钥对的控件。
<mark>	在文档中标记或高亮文本用于引用参考，表示它在另一个上下文中的相关性。
<meter>	表示一个度量，如磁盘使用率。
<nav>	表示用于导航文档的节。
<output>	表示某种输出类型，比如通过脚本计算。
<progress>	表示一个任务的进度，比如下载或者执行一系列的昂贵操作时。
<ruby>	和 <rt> 以及 <rp> 一起使用，可用于标记 ruby 注释。
<section>	表示一个通用文档或者应用程序部分。
<time>	表示日期或时间。
<video>	定义一个视频文件。
<wbr>	表示一个换行机会。

### <input> 标签新增的类型

input 元素的 type 属性现在拥有下列新值：

类型	描述
color	拾色器，可以通过色盘或者调色板来表现。
date	日历日期选择器。



<code>datetime-local</code>	显示日期和时间，没有设置或者指定时区。
<code>datetime</code>	显示完整的日期和时间，包括时区。
<code>email</code>	输入类型应该是邮件。
<code>month</code>	给定年份的月份选择器。
<code>number</code>	只能包含一个数字值的字段。
<code>range</code>	区间数值选择器，通常显示为一个滑块。
<code>search</code>	为搜索引擎提供关键字。比如，浏览器顶部的搜索栏。
<code>tel</code>	输入类型应该是电话号码。
<code>time</code>	时间指示器或选择器，不带时区信息。
<code>url</code>	输入类型应该是 URL 类型。
<code>week</code>	给定年份的周选择器。



3

有用的 HTML5 参考



## HTML5 快速指南

---

HTML5 是 HTML 标准的下一个重要版本，用于接替 HTML 4.01，XHTML 1.0 和 XHTML 1.1。HTML5 也是一种在万维网上构建和呈现内容的标准。

最新版的 Apple Safari, Google Chrome, Mozilla FireFox 和 Opera 支持大部分 HTML 5 特性，IE 9.0 也支持一些 HTML5 功能。

### 新特性

HTML5 引入了许多新元素和属性帮助我们构建现代化的网站。下面是 HTML5 引入的主要特性：

- **新的语义化元素：** 比如 `<header>`，`<footer>` 和 `<section>`。
- **表单 2.0：** 改进了 HTML Web 表单，给 `<input>` 标签引入了一些新的属性。
- **持久化本地存储：** 为了不使用第三方插件实现、
- **WebSocket：** 用于 Web 应用程序的下一代双向通信技术。
- **服务器推送事件：** HTML5 引入了从 Web 服务器到 Web 浏览器的事件，也被称作服务器推送事件（SSE）。
- **Canvas：** 支持用 JavaScript 以编程的方式进行二维绘图。
- **音频和视频：** 在网页中嵌入音频或视频而无需借助第三方插件。
- **地理定位：** 用户可以选择与我们的网页共享他们的地理位置。
- **Microdata：** 允许我们创建 HTML5 之外的自定义词汇表，以及使用自定义语义扩展网页。
- **拖放：** 把同一网页上的条目从一个位置拖放到另一个位置。

### HTML5 语法

HTML5 有“自己的” HTML 语法，它与已经发布在网络上的 HTML 4 以及 XHTML1 文档兼容，但是不兼容 HTML 4 中更复杂的 SGML 特性。

HTML5 并没有 XHTML 中需要小写标签名，属性要带引号，属性必须有一个值以及必须闭合所有空元素的语法规则。

但是 HTML5 更具灵活性，支持下列形式：

- 标签名大写。
- 属性的双引号可选。
- 属性值可选。
- 闭合空元素可选。

## DOCTYPE

在老版本的 HTML 中，DOCTYPE 很长，因为 HTML 语言是基于 SGML 的，需要引用一个 DTD。

HTML5 作者可以使用简单的语法来指定如下形式的 DOCTYPE：

```
<!DOCTYPE html>
```

上述语法不区分大小写。

## 字符编码

HTML5 作者可以使用如下所示的简单语法指定字符编码：

```
<meta charset="UTF-8">
```

上述语法不区分大小写。

## HTML5 元素

HTML5 元素使用起始标签和结束标签标记。标签使用尖括号之间的标签名限定。

起始标签和结束标签的区别在于后者标签名前面包含一个斜杠。

下面是一个 HTML5 元素示例：

```
<p>...</p>
```

HTML5 标签名不区分大小写，可以全部大写或者混合使用，虽然最常见的约定是始终使用小写。

大多数元素都包含一些内容，比如 <p>...</p> 包含一个段落。但是，有些元素不能包含任意内容，它们被称作空白元素。比如，br, hr, link 和 meta 等等。

这里有一个完整的 [HTML5 元素列表](#)。

## HTML5 属性

元素可以包含属性（attributes），用来给一个元素设置各种属性（properties）。

有些属性被定义为全局的，可以用在任何元素上，而其他的被定义为元素特有的。所有的属性都有一个名称和一个值，看起来如下面的示例所示。

下面是一个使用 HTML5 属性的例子，演示了如何用名为 class 的属性和值 “example” 标记一个 div 元素：

```
<div class="example">...</div>
```

属性只能在起始标签中指定，绝对不能用在结束标签中。

HTML5 属性不区分大小写，可以全部大写或者混合使用，尽管最常见的约定是始终使用小写。

这里有一个完整的 [HTML5 属性列表](#)。

## HTML5 事件

当用户访问我们的网站时，他们会点击文本，图片，链接，将鼠标悬停在某些东西上面等等。这些都是 JavaScript 调用事件的例子。

我们可以在 JavaScript 或者 vbscript 中编写事件处理程序，然后把这些事件处理程序指定为事件标签属性的值。下面列出了 HTML5 规范定义的各种事件属性。

当任意事件发生在 HTML5 元素上时，下列属性可以用来触发任何作为值提供的 JavaScript 和 vbscript 代码。

这里有一个完整的 [HTML5 事件列表](#)。

## HTML5 文档

为了得到更好的结构，引入了下面的标签：

- **section:** 这个标签表示一个通用的文档或者应用程序节。它可以和 h1-h6 一起使用来表示文档结构。
- **article:** 这个标签表示文档内容的一个独立块，比如博客条目或者报纸上的文章。
- **aside:** 这个标签表示与页面其他部分略微相关的内容块。

- **header:** 这个标签表示一个节的头部。
- **footer:** 这个标签表示一个节的脚注，可以包含作者，版权等信息。
- **nav:** 这个标签表示用于导航文档的节。
- **dialog:** 这个标签可以用于标记会话。
- **figure:** 这个标签可以用于关联标题和某些嵌入内容，比如图表和视频。

一个 HTML5 文档的标记看起来就像下面这样：

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>...</title>
</head>
<body>
  <header>...</header>
  <nav>...</nav>
  <article>
    <section>
      ...
    </section>
  </article>
  <aside>...</aside>
  <footer>...</footer>
</body>
```

便于学习这一概念 – 可以进行[在线练习](#)。

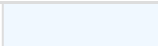
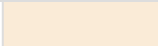










我知道你不会满足于这个快速指南，因此我建议你探索完整的 HTML5 教程，因为它在 HTML4 版本的基础上有巨大的变化，这并不能在简短的随笔中覆盖。








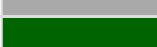












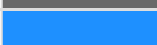

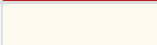

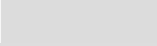





## HTML5 颜色名称

下面的表格演示了 HTML 3.2 引入的 16 种颜色名称，它是用来支持 8位显卡提供的 16 种颜色：

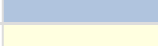
颜色名称	十六进制值	颜色	演示
aqua	#00ffff		<a href="#">Demo</a>
black	#000000		<a href="#">Demo</a>
blue	#0000ff		<a href="#">Demo</a>
fuchsia	#ff00ff		<a href="#">Demo</a>
green	#008000		<a href="#">Demo</a>
gray	#808080		<a href="#">Demo</a>
lime	#00ff00		<a href="#">Demo</a>
maroon	#800000		<a href="#">Demo</a>
navy	#000080		<a href="#">Demo</a>
olive	#808000		<a href="#">Demo</a>
purple	#800080		<a href="#">Demo</a>
red	#ff0000		<a href="#">Demo</a>
silver	#c0c0c0		<a href="#">Demo</a>
teal	#008080		<a href="#">Demo</a>
white	#ffffff		<a href="#">Demo</a>
yellow	#ffff00		<a href="#">Demo</a>


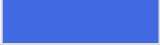











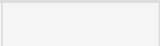



下面是其他颜色，它们并不是 HTML 或者 XHTML 的一部分，但是大多数版本的 IE 和 Netscape 都支持。

颜色名称	十六进制值	颜色	演示
aliceblue	#f0f8ff		<a href="#">Demo</a>
antiquewhite	#faebd7		<a href="#">Demo</a>
aquamarine	#7fffd4		<a href="#">Demo</a>
azure	#f0ffff		<a href="#">Demo</a>
beige	#f5f5dc		<a href="#">Demo</a>
bisque	#ffe4c4		<a href="#">Demo</a>
blanchedalmond	#ffebcd		<a href="#">Demo</a>
blueviolet	#8a2be2		<a href="#">Demo</a>
brown	#a52a2a		<a href="#">Demo</a>
burlywood	#deb887		<a href="#">Demo</a>
cadetblue	#5f9ea0		<a href="#">Demo</a>
chartreuse	#7fff00		<a href="#">Demo</a>

chocolate	#d2691e		<a href="#">Demo</a>
coral	#ff7f50		<a href="#">Demo</a>
cornflowerblue	#6495ed		<a href="#">Demo</a>
cornsilk	#fff8dc		<a href="#">Demo</a>
crimson	#dc143c		<a href="#">Demo</a>
cyan	#00ffff		<a href="#">Demo</a>
darkblue	#00008b		<a href="#">Demo</a>
darkcyan	#008b8b		<a href="#">Demo</a>
darkgoldenrod	#b8860b		<a href="#">Demo</a>
darkgray	#a9a9a9		<a href="#">Demo</a>
darkgreen	#006400		<a href="#">Demo</a>
darkkhaki	#bdb76b		<a href="#">Demo</a>
darkmagenta	#8b008b		<a href="#">Demo</a>
darkolivegreen	#556b2f		<a href="#">Demo</a>
darkorange	#ff8c00		<a href="#">Demo</a>
darkorchid	#9932cc		<a href="#">Demo</a>
darkred	#8b0000		<a href="#">Demo</a>
darksalmon	#e9967a		<a href="#">Demo</a>
darkseagreen	#8fbc8f		<a href="#">Demo</a>
darkslateblue	#483d8b		<a href="#">Demo</a>
darkslategray	#2f4f4f		<a href="#">Demo</a>
darkturquoise	#00ced1		<a href="#">Demo</a>
darkviolet	#9400d3		<a href="#">Demo</a>
deeppink	#ff1493		<a href="#">Demo</a>
deepskyblue	#00bfff		<a href="#">Demo</a>
dimgray	#696969		<a href="#">Demo</a>
dodgerblue	#1e90ff		<a href="#">Demo</a>
firebrick	#b22222		<a href="#">Demo</a>
floralwhite	#fffaf0		<a href="#">Demo</a>
forestgreen	#228b22		<a href="#">Demo</a>
gainsboro	#dcdcdc		<a href="#">Demo</a>
ghostwhite	#f8f8ff		<a href="#">Demo</a>
gold	#ffd700		<a href="#">Demo</a>
goldenrod	#daa520		<a href="#">Demo</a>
gray	#808080		<a href="#">Demo</a>
greenyellow	#adff2f		<a href="#">Demo</a>
honeydew	#f0fff0		<a href="#">Demo</a>
hotpink	#ff69b4		<a href="#">Demo</a>



indianred	#cd5c5c		<a href="#">Demo</a>
indigo	#4b0082		<a href="#">Demo</a>
ivory	#fffff0		<a href="#">Demo</a>
khaki	#f0e68c		<a href="#">Demo</a>
lavender	#e6e6fa		<a href="#">Demo</a>
lavenderblush	#fff0f5		<a href="#">Demo</a>
lawngreen	#7cfc00		<a href="#">Demo</a>
lemonchiffon	#ffffac		<a href="#">Demo</a>
lightblue	#add8e6		<a href="#">Demo</a>
lightcoral	#f08080		<a href="#">Demo</a>
lightcyan	#e0ffff		<a href="#">Demo</a>
lightgoldenrodyellow	#fafad2		<a href="#">Demo</a>
lightgreen	#90ee90		<a href="#">Demo</a>
lightgrey	#d3d3d3		<a href="#">Demo</a>
lightpink	#ffb6c1		<a href="#">Demo</a>
lightsalmon	#ffa07a		<a href="#">Demo</a>
lightseagreen	#20b2aa		<a href="#">Demo</a>
lightskyblue	#87cefa		<a href="#">Demo</a>
lightslategray	#778899		<a href="#">Demo</a>
lightsteelblue	#b0c4de		<a href="#">Demo</a>
lightyellow	#ffffe0		<a href="#">Demo</a>
limegreen	#32cd32		<a href="#">Demo</a>
linen	#faf0e6		<a href="#">Demo</a>
magenta	#ff00ff		<a href="#">Demo</a>
mediumblue	#0000cd		<a href="#">Demo</a>
mediumorchid	#ba55d3		<a href="#">Demo</a>
mediumpurple	#9370db		<a href="#">Demo</a>
midnightblue	#191970		<a href="#">Demo</a>
mistyrose	#ffe4e1		<a href="#">Demo</a>
moccasin	#ffe4b5		<a href="#">Demo</a>
oldlace	#fdf5e6		<a href="#">Demo</a>
orange	#ffa500		<a href="#">Demo</a>
orchid	#da70d6		<a href="#">Demo</a>
peachpuff	#ffdab9		<a href="#">Demo</a>
peru	#cd853f		<a href="#">Demo</a>
pink	#ffc0cb		<a href="#">Demo</a>
plum	#dda0dd		<a href="#">Demo</a>
purple	#800080		<a href="#">Demo</a>

rosybrown	#bc8f8f		<a href="#">Demo</a>
royalblue	#4169e1		<a href="#">Demo</a>
salmon	#fa8072		<a href="#">Demo</a>
sandybrown	#f4a460		<a href="#">Demo</a>
seagreen	#2e8b57		<a href="#">Demo</a>
sienna	#a0522d		<a href="#">Demo</a>
skyblue	#87ceeb		<a href="#">Demo</a>
slateblue	#6a5acd		<a href="#">Demo</a>
steelblue	#4682b4		<a href="#">Demo</a>
tan	#d2b48c		<a href="#">Demo</a>
thistle	#d8bfd8		<a href="#">Demo</a>
tomato	#ff6347		<a href="#">Demo</a>
violet	#ee82ee		<a href="#">Demo</a>
wheat	#f5deb3		<a href="#">Demo</a>
whitesmoke	#f5f5f5		<a href="#">Demo</a>
yellow	#ffff00		<a href="#">Demo</a>
yellowgreen	#9acd32		<a href="#">Demo</a>

# HTML5 字体参考

字体是特定于平台的。如果我们使用不同的系统，那么对于任何 Web 页面都会得到不同的外观和感受。

从 4.0 开始 HTML `<FONT>` 标签就已经被弃用了，现在所有的字体都使用 CSS 设置。这里有一个给网页主题设置字体的简单语法：

```
body { font-family: "new century schoolbook"; }
```

或者

```
<body style="font-family:new century schoolbook;" >
```

方便起见，可以进行[在线练习](#)。

## 微软 Windows 系统和浏览器字体

Font	Font	Font
Andale Mono	Arial	Arial Bold
Arial Italic	Arial Bold Italic	Arial Black
Comic Sans MS	Comic Sans MS Bold	Courier New
Courier New Bold	Courier New Italic	Courier New Bold Italic
Georgia	Georgia Bold	Georgia Italic
Georgia Bold Italic	Impact	Lucida Console
Lucida Sans Unicode	Marlett	Minion Web
Symbol	Times New Roman	Times New Roman Bold
Times New Roman Italic	Times New Roman Bold Italic	Tahoma
Trebuchet MS	Trebuchet MS Bold	Trebuchet MS Italic
Trebuchet MS Bold Italic	Verdana	Verdana Bold
Verdana Italic	Verdana Bold Italic	Webdings

在 <http://www.microsoft.com/typography/fonts> 上可以找到更多关于微软字体的信息。

可以在 [微软字体示例](#) 页面查看示例字体。

## Mac 系统字体

下面列出了 Mac OS 7 以及更高版本支持的字体。

Font	Font	Font
American Typewriter	Andale Mono	Apple Chancery
Arial	Arial Black	Brush Script
Baskerville	Big Caslon	Comic Sans MS
Copperplate	Courier New	Gill Sans
Futura	Herculanum	Impact
Lucida Grande	Marker Felt	Optima
Trebuchet MS	Verdana	Webdings
Palatino	Symbol	Times
Osaka	Papyrus	Times New Roman
Textile	Zapf Dingbats	Zapfino
Techno	Hoefler Text	Skia
Hoefler Text Ornaments	Capitals	Charcoal
Gadget	Sand	

可以在 [Mac 字体示例](#) 页面查看示例字体。

## Unix 系统字体

下面列出了大多数 Unix 系统变体支持的字体。

Font	Font	Font
Charter	Clean	Courier
Fixed	Helvetica	Lucida
Lucida bright	Lucida Typewriter	New Century Schoolbook
Symbol	Terminal	Times
Utopia		

可以在 [Unix 字体示例](#) 页查看示例字体。

## HTML5 URL 编码

URL 编码就是将 URLs 中不宜打印的字符或者具有特殊意义的字符转换为 Web 浏览器和服务端明白且普遍接受的表示法。 这些字符包括：

- **ASCII 控制字符** – 不宜打印的字符通常用于输出控制。字符范围是十六进制的 00-1F（十进制的 0-31）和 7F（十进制的 127）。下面提供了完整的编码表。
- **非 ASCII 控制字符** – 这些字符超出了 128 个 ASCII 字符集的范围。这个范围是 ISO-拉丁字符集的一部分以及包含整个十六进制的 ISO-拉丁字符集 00-FF （十进制的 128-255）的“前半部分”。下面提供了完整的编码表。
- **保留字符** – 诸如美元符号，和号，加号，通用符号，正斜杠，冒号，分号，等号，问号以及 “at” 这类符号。所有这些符号在 URL 内都有不同的意义，因此需要编码。下面提供了完整的编码表。
- **不安全字符** – 包括空格，问号，小于符号，大于符号，磅字符，百分比符号，大括号左边部分，大括号右边部分，管道符，反斜杠，插入符号，波浪线。左方括号，右方括号，沉音符。出于某些原因，这些字符出现在 URLs 中存在被误解的可能性。这些字符也应该始终被编码。下面提供了完整的编码表。

编码表示法需要三个字符替换期望的字符：一个百分号，两个在 ASCII 字符集中表示字符位置的十六进制数字、

### 示例

最常见的特殊字符之一便是空格。我们不能在 URL 中直接输入一个空格。空格在字符集中就是十六进制的 20。因此请求服务器时可以使用 %20 表示空格。

```
http://www.example.com/new%20pricing.html
```

这个 URL 实际上是从 www.example.com 检索一个名为 new pricing.html 的文档。

### ASCII 控制字符编码

包括十六进制的 00-1F（十进制的 0-31）和 7F（十进制的 127）字符码。

十进制格式	十六进制值	字符	URL 编码
0	00		%00
1	01		%01
2	02		%02

3	03		%03
4	04		%04
5	05		%05
6	06		%06
7	07		%07
8	08	退格符	%08
9	09	tab	%09
10	0a	换行符	%0a
11	0b		%0b
12	0c		%0c
13	0d	回车符	%0d
14	0e		%0e
15	0f		%0f
16	10		%10
17	11		%11
18	12		%12
19	13		%13
20	14		%14
21	15		%15
22	16		%16
23	17		%17
24	18		%18
25	19		%19
26	1a		%1a
27	1b		%1b
28	1c		%1c
29	1d		%1d
30	1e		%1e
31	1f		%1f
127	7f		%7f

### 非 ASCII 控制字符编码

包括整个十六进制的 ISO-拉丁字符集 80-FF（十进制的 128-255）编码的“前半部分”。

十进制格式	十六进制值	字符	URL 编码
128	80	€	%80
129	81	?	%81

130	82	,	%82
131	83	<i>f</i>	%83
132	84	„	%84
133	85	...	%85
134	86	†	%86
135	87	‡	%87
136	88	^	%88
137	89	‰	%89
138	8a	Š	%8a
139	8b	‹	%8b
140	8c	Œ	%8c
141	8d	?	%8d
142	8e	?	%8e
143	8f	?	%8f
144	90	?	%90
145	91	‘	%91
146	92	’	%92
147	93	“	%93
148	94	”	%94
149	95	•	%95
150	96	–	%96
151	97	—	%97
152	98	~	%98
153	99	™	%99
154	9a	š	%9a
155	9b	›	%9b
156	9c	œ	%9c
157	9d	?	%9d
158	9e	?	%9e
159	9f	ÿ	%9f
160	a0		%a0
161	a1	ı	%a1
162	a2	¢	%a2
163	a3	£	%a3
164	a4	∕	%a4
165	a5	¥	%a5
166	a6	¡	%a6
167	a7	§	%a7

168	a8	¨	%a8
169	a9	©	%a9
170	aa	ª	%aa
171	ab	«	%ab
172	ac	¬	%ac
173	ad		%ad
174	ae	®	%ae
175	af	—	%af
176	b0	º	%b0
177	b1	±	%b1
178	b2	²	%b2
179	b3	³	%b3
180	b4	´	%b4
181	b5	µ	%b5
182	b6	¶	%b6
183	b7	•	%b7
184	b8	,	%b8
185	b9	¹	%b9
186	ba	º	%ba
187	bb	»	%bb
188	bc	¼	%bc
189	bd	½	%bd
190	be	¾	%be
191	bf	¿	%bf
192	c0	À	%c0
193	c1	Á	%c1
194	c2	Â	%c2
195	c3	Ã	%c3
196	c4	Ä	%c4
197	c5	Å	%c5
198	c6	Æ	%v6
199	c7	Ç	%c7
200	c8	È	%c8
201	c9	É	%c9
202	ca	Ê	%ca
203	cb	Ë	%cb
204	cc	Ì	%cc
205	cd	Í	%cd



206	ce	Î	%ce
207	cf	Ĩ	%cf
208	d0	Ð	%d0
209	d1	Ñ	%d1
210	d2	Ò	%d2
211	d3	Ó	%d3
212	d4	Ô	%d4
213	d5	Õ	%d5
214	d6	Ö	%d6
215	d7	×	%d7
216	d8	Ø	%d8
217	d9	Ù	%d9
218	da	Ú	%da
219	db	Û	%db
220	dc	Ü	%dc
221	dd	Ý	%dd
222	de	Þ	%de
223	df	ß	%df
224	e0	à	%e0
225	e1	á	%e1
226	e2	â	%e2
227	e3	ã	%e3
228	e4	ä	%e4
229	e5	å	%e5
230	e6	æ	%e6
231	e7	ç	%e7
232	e8	è	%e8
233	e9	é	%e9
234	ea	ê	%ea
235	eb	ë	%eb
236	ec	ì	%ec
237	ed	í	%ed
238	ee	î	%ee
239	ef	ï	%ef
240	f0	ð	%f0
241	f1	ñ	%f1
242	f2	ò	%f2
243	f3	ó	%f3

244	f4	ô	%f4
245	f5	õ	%f5
246	f6	ö	%f6
247	f7	÷	%f7
248	f8	ø	%f8
249	f9	ù	%f9
250	fa	ú	%fa
251	fb	û	%fb
252	fc	ü	%fc
253	fd	ý	%fd
254	fe	þ	%fe
255	ff	ÿ	%ff

保留字符编码

下表用于编码保留字符。

十进制格式	十六进制值	字符	URL 编码
36	24	\$	%24
38	26	&	%26
43	2b	+	%2b
44	2c	,	%2c
47	2f	/	%2f
58	3a	:	%3a
59	3b	;	%3b
61	3d	=	%3d
63	3f	?	%3f
64	40	@	%40

不安全字符编码

下表用于编码不安全字符。

十进制格式	十六进制值	字符	URL 编码
32	20	space	%20
34	22	"	%22
60	3c	<	%3c
62	3e	>	%3e

35	23	#	%23
37	25	%	%25
123	7b	{	%7b
125	7d	}	%7d
124	7c		%7c
92	5c	\	%5c
94	5e	^	%5e
126	7e	~	%7e
91	5b	[	%5b
93	5d	]	%5d
96	60	`	%60

## HTML5 字符实体参考

### HTML5 中的字符实体

有些字符在 HTML5 中是保留字。比如，我们不能在文本中使用大于，小于标记或者尖括号，因为浏览器可能会误认为它们是标记。

HTML5 处理程序必须支持下表中列出的 5 个特殊字符。

符号	描述	实体名称	实体编号
"	引号	&quot;	&#34;
'	撇号	&apos;	&#39;
&	和号	&amp;	&#38;
<	小于号	&lt;	&#60;
>	大于号	&gt;	&#62;

要在页面中编写一个元素和属性把代码展示给用户而不是让浏览器处理（比如：<div id="character">）需要这样编写：

```
&lt;div id=&quot;character&quot;&gt;
```

HTML5 处理器还应该支持一长串特殊字符。为了将它们呈现在文档中，我们可以使用数值码或者实体名称。例如，要插入一个版权符号，我们可以使用下列形式之一：

```
&copy; 2007

或者

&#169; 2007
```

ISO 8859-1 字符集是应用最广泛的一种字符集。下面给出了一个完整的 ISO 8859-1 字符集参考。

### ISO 8859-1 符号实体

显示结果	描述	实体名称	实体编号
	空格	&nbsp;	&#160;
¡	倒感叹号	&iexcl;	&#161;
¤	货币	&curren;	&#164;
¢	分	&cent;	&#162;

£	磅	&pound;	&#163;
¥	元	&yen;	&#165;
	断竖线	&brvbar;	&#166;
§	节	&sect;	&#167;
¨	间隔分音符	&uml;	&#168;
©	版权标识	&copy;	&#169;
ª	阴性顺序标识符	&ordf;	&#170;
«	角引号（左）	&laquo;	&#171;
¬	逻辑非	&not;	&#172;
	软连字符	&shy;	&#173;
®	注册商标	&reg;	&#174;
™	商标	&trade;	&#8482;
—	间隔长音符	&macr;	&#175;
°	度	&deg;	&#176;
±	加或减	&plusmn;	&#177;
²	上标 2	&sup2;	&#178;
³	上标 3	&sup3;	&#179;
´	急性间距	&acute;	&#180;
μ	百万分之一	&micro;	&#181;
¶	段落	&para;	&#182;
•	中间点	&middot;	&#183;
¸	间距变音符	&cedil;	&#184;
¹	上标 1	&sup1;	&#185;
º	阳性顺序指示符	&ordm;	&#186;
»	角引号（右）	&raquo;	&#187;
¼	分数 1/4	&frac14;	&#188;
½	分数 1/2	&frac12;	&#189;
¾	分数 3/4	&frac34;	&#190;
¿	倒问号	&iquest;	&#191;
×	乘号	&times;	&#215;
÷	除号	&divide;	&#247;

ISO 8859-1 字符实体

显示结果	描述	实体名称	实体编码
À	带沉音符的大写 a	&Agrave;	&#192;
Á	带重音符的大写 a	&Aacute;	&#193;

Â	带抑扬音符的大写 a	&Acirc;	&#194;
Ã	带波浪符的大写 a	&Atilde;	&#195;
Ä	带变元音符的大写 a	&Auml;	&#196;
Å	带环形符的大写 a	&Aring;	&#197;
Æ	大写 ae	&AElig;	&#198;
Ç	带变音符的大写 c	&Ccedil;	&#199;
È	带沉音符的大写 e	&Egrave;	&#200;
É	带重音符的大写 e	&Eacute;	&#201;
Ê	带抑扬音符的大写 e	&Ecirc;	&#202;
Ë	带变元音符的大写 e	&Euml;	&#203;
Ì	带沉音符的大写 i	&Igrave;	&#204;
Í	带重音符的大写 i	&Iacute;	&#205;
Î	带抑扬音符的大写 i	&Icirc;	&#206;
Ï	带变元音符的大写 i	&Iuml;	&#207;
Ð	冰岛语, 大写 eth	&ETH;	&#208;
Ñ	带波浪符的大写 n	&Ntilde;	&#209;
Ò	带沉音符的大写 o	&Ograve;	&#210;
Ó	带重音符的大写 o	&Oacute;	&#211;
Ô	带抑扬音符的大写 o	&Ocirc;	&#212;
Õ	带波浪符的大写 o	&Otilde;	&#213;
Ö	带变元音符的大写 o	&Ouml;	&#214;
Ø	带斜线的大写 o	&Oslash;	&#216;
Ù	带沉音符的大写 u	&Ugrave;	&#217;
Ú	带重音符的大写 u	&Uacute;	&#218;
Û	带抑扬音符的大写 u	&Ucirc;	&#219;
Ü	带变元音符的大写 u	&Uuml;	&#220;
Ý	带重音符的大写 y	&Yacute;	&#221;
Þ	冰岛语, 大写 THORN	&THORN;	&#222;
ß	德语, 小号半升音符 s	&szlig;	&#223;
à	带沉音符的小号 a	&agrave;	&#224;
á	带重音符的小号 a	&aacute;	&#225;
â	带抑扬音符的小号 a	&acirc;	&#226;
ã	带波浪符的小号 a	&atilde;	&#227;
ä	带变元音符的小号 a	&auml;	&#228;
å	带环形的小号 a	&aring;	&#229;
æ	小号 ae	&aelig;	&#230;
ç	带变音符的小号 c	&ccedil;	&#231;
è	带陈音符的小号 e	&egrave;	&#232;

é	带重音符的小号 e	&eacute;	&#233;
ê	带抑扬音符的小号 e	&ecirc;	&#234;
ë	带变元音符的小号 e	&euml;	&#235;
ì	带沉音符的小号 i	&igrave;	&#236;
í	带重音符的小号 i	&iacute;	&#237;
î	带抑扬音符的小号 i	&icirc;	&#238;
ï	带变元音符的小号 i	&iuml;	&#239;
ð	冰岛语，小号 eth	&eth;	&#240;
ñ	带波浪符的小号 n	&ntilde;	&#241;
ò	带沉音符的小号 o	&ograve;	&#242;
ó	带重音符的小号 o	&oacute;	&#243;
ô	带抑扬音符的小号 o	&ocirc;	&#244;
õ	带波浪符的小号 o	&otilde;	&#245;
ö	带变元音符的小号 o	&ouml;	&#246;
ø	带斜线的小号 o	&oslash;	&#248;
ù	带沉音符的小号 u	&ugrave;	&#249;
ú	带重音符的小号 u	&uacute;	&#250;
û	带抑扬音符的小号 u	&ucirc;	&#251;
ü	带变元音符的小号 u	&uuml;	&#252;
ý	带重音符的小号 y	&yacute;	&#253;
þ	冰岛语，小号 thorn	&thorn;	&#254;
ÿ	带变元音符的小号 y	&yuml;	&#255;

HTML5 浏览器支持的其他实体

显示结果	描述	实体名称	实体编码
Œ	大写连字符 OE	&OElig;	&#338;
œ	小号连字符 oe	&oelig;	&#339;
Š	带抑扬符的大写 S	&Scaron;	&#352;
š	带抑扬符的小号 S	&scaron;	&#353;
Ÿ	带变音符的大写 Y	&Yuml;	&#376;
^	抑扬音符修饰符字母	&circ;	&#710;
~	小号波浪符	&tilde;	&#732;
	en 空格	&ensp;	&#8194;
	em 空格	&emsp;	&#8195;
	thin 空格	&thinsp;	&#8201;
??	零宽不连字	&zwnj;	&#8204;

??	零宽连字	&zwj;	&#8205;
	从左到右标记	&lrm;	&#8206;
	从右到左标记	&rlm;	&#8207;
-	en 破折号	&ndash;	&#8211;
—	em 破折号	&mdash;	&#8212;
‘	左单引号	&lsquo;	&#8216;
’	右单引号	&rsquo;	&#8217;
,	低位单引号	&sbquo;	&#8218;
“	左双引号	&ldquo;	&#8220;
”	右双引号	&rdquo;	&#8221;
„	低位双引号	&bdquo;	&#8222;
†	剑号	&dagger;	&#8224;
‡	双剑号	&Dagger;	&#8225;
…	省略号	&hellip;	&#8230;
‰	千分率	&permil;	&#8240;
◁	角形左引号	&lsaquo;	&#8249;
▷	角形右引号	&rsaquo;	&#8250;
€	欧元符	&euro;	&#8364;



## HTML5 字符编码

---

字符编码就是将字节转换为字符的一种方法。要验证或者显示一个 HTML 文档，程序必须选择一个字符编码。HTML5 作者有三种方式设置字符编码：

### HTTP Content-Type 头：

如果你在编写 cgi 程序或者类似的程序，那么可以使用 HTTP *Content-Type* 头设置任意字符编码：

下面是一个简单的例子：

```
print "Content-Type: text/html; charset=utf-8\r\n";
```

### <meta> 元素：

可以使用带有 `charset` 属性的 `<meta>` 元素指定 HTML5 文档前 512 个字节的编码：

下面是简化的例子：

```
<meta charset="UTF-8">
```

尽管这种语法是被允许的，但上述语法需要使用 `<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">` 替换。

### Unicode 字节顺序标记 (BOM)

一个字节顺序标记 (BOM) 由数据流开头的 U+FEFF 字符码组成，它可以用作定义字节顺序和编码形式的签名，主要是未标记的明文文件。

许多 Windows 程序（包括 Windows 记事本）都会在保存为 UTF-8 的任意文档开头添加 0xEF, 0xBB, 0xBF。这就是 Unicode 字节顺序标记 (BOM) 的 UTF-8 编码，通常被称为 UTF-8 BOM，尽管它和字节顺序没有关系。

对于 HTML5 文档，我们可以在文件的开头使用 Unicode 字节顺序标记 (BOM) 字符。这个字符为使用的编码提供了签名。



4

HTML5 工具



## HTML5 Modernizr

---

Modernizr 是一个很小的用来检测下一代 Web 技术原生实现可用性的 JavaScript 库。

HTML5 和 CSS3 都引入了各自的新特性，但是同时也有很多浏览器不支持这些新特性。

Modernizr 提供了一种简单的方式检测任意新特性，从而让我们可以采取相应的操作。比如，浏览器还不支持视频特性，那么我们可以显示一个简单的页面。

我们还可以基于某个特性的可用性来创建 CSS 规则，如果浏览器不支持某个新特性，那么这些规则将会自动应用到网页上。

可以从 [Modernizr 官网](#) 下载最新版的程序。

### 语法

开始使用 Modernizr 之前，需要在 HTML 页面的头部引入这个 JavaScript 库，如下所示：

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

正如上面提到的，我们可以基于特性的可用性来创建 CSS 规则，如果浏览器不支持这个新特性，那么这些规则就会自动应用到网页上。

下面是处理支持和不支持特性的一个简单语法：

```
/* In your CSS: */
.no-audio #music {
    display: none; /* Don't show Audio options */
}
.audio #music button {
    /* Style the Play and Pause buttons nicely */
}

<!-- In your HTML: -->
<div id="music">
    <audio>
        <source src="audio.ogg" />
        <source src="audio.mp3" />
    </audio>
    <button id="play">Play</button>
    <button id="pause">Pause</button>
</div>
```

这里要注意的是，我们需要为想要处理的浏览器还不支持的特性或者属性使用 “no-” 前缀。

下面是通过 JavaScript 检测特定属性的语法：

```
if (Modernizr.audio) {  
    /* properties for browsers that  
    support audio */  
} else {  
    /* properties for browsers that  
    does not support audio */  
}
```

便于学习上述概念 - 请使用不同的浏览器进行[在线练习](#)。

## Modernizr 提供的特性检测

下面是可以通过 Modernizr 进行检测的特性列表：

特性	CSS 属性	JavaScript 检测
@font-face	.fontface	Modernizr.fontface
Canvas	.canvas	Modernizr.canvas
Canvas Text	.canvastext	Modernizr.canvastext
HTML5 Audio	.audio	Modernizr.audio
HTML5 Audio formats	NA	Modernizr.audio[format]
HTML5 Video	.video	Modernizr.video
HTML5 Video Formats	NA	Modernizr.video[format]
rgba()	.rgba	Modernizr.rgba
hsla()	.hsla	Modernizr.hsla
border-image	.borderimage	Modernizr.borderimage
border-radius	.borderradius	Modernizr.borderradius
box-shadow	.boxshadow	Modernizr.boxshadow
Multiple backgrounds	.multiplebgs	Modernizr.multiplebgs
opacity	.opacity	Modernizr.opacity
CSS Animations	.cssanimations	Modernizr.cssanimations
CSS Columns	.csscolumns	Modernizr.csscolumns
CSS Gradients	.cssgradients	Modernizr.cssgradients
CSS Reflections	.cssreflections	Modernizr.cssreflections
CSS 2D Transforms	.csstransforms	Modernizr.csstransforms

CSS 3D Transforms	.csstransforms3d	Modernizr.csstransforms3d
CSS Transitions	.csstransitions	Modernizr.csstransitions
Geolocation API	.geolocation	Modernizr.geolocation
Input Types	NA	Modernizr.inputtypes[type]
Input Attributes	NA	Modernizr.input[attribute]
localStorage	.localStorage	Modernizr.localStorage
sessionStorage	.sessionstorage	Modernizr.sessionstorage
Web Workers	.webworkers	Modernizr.webworkers
applicationCache	.applicationcache	Modernizr.applicationcache
SVG	.svg	Modernizr.svg
SVG Clip Paths	.svgclippaths	Modernizr.svgclippaths
SMIL	.smil	Modernizr.smil
Web SQL Database	.websqldatabase	Modernizr.websqldatabase
IndexedDB	.indexeddb	Modernizr.indexeddb
Web Sockets	.websockets	Modernizr.websockets
Hashchange Event	.hashchange	Modernizr.hashchange
History Management	.historymanagement	Modernizr.historymanagement
Drag and Drop	.draganddrop	Modernizr.draganddrop
Cross-window Messaging	.crosswindowmessaging	Modernizr.crosswindowmessaging
addTest() Plugin API	NA	Modernizr.addTest(str,fn)

## HTML5 验证

---

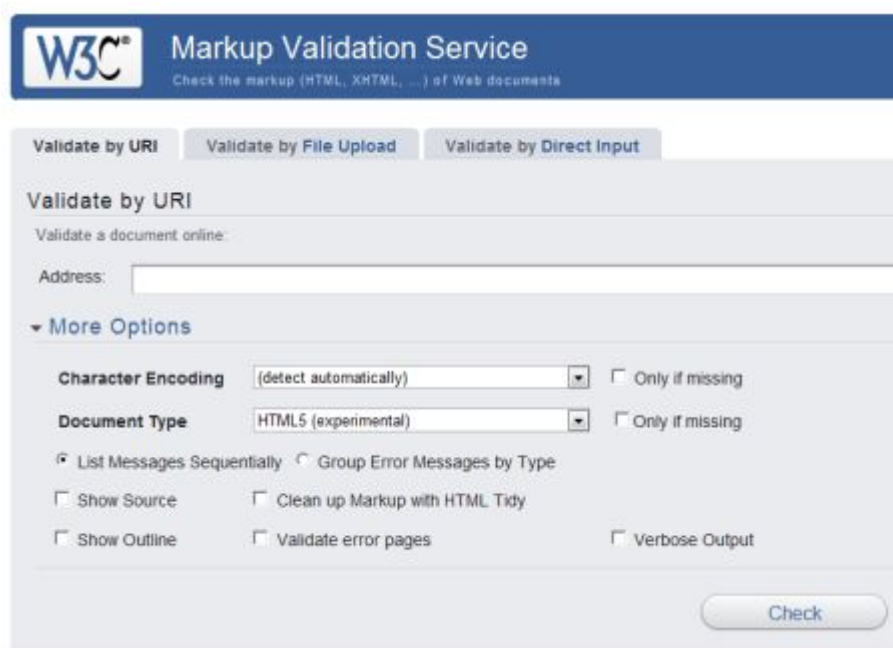
编写这个教程时 HTML5 还处在初始阶段，网络上只有少量的几个验证器可用。

下面两个验证器是非常可靠且可用的。

### (1) W3C Markup Validator

[W3C Markup Validator](#) 可以检查 HTML, XHTML, SMIL, MathML 等格式的 Web 文档标记的有效性。这个验证器是 W3C 标准验证服务 Unicorn 的一部分。

要对 HTML5 使用这个验证器，需要使用 **More Options** 选项以及选择 **Document Type** 为 **HTML5(实验性)**，如下所示：



图片 4.3 w3c-markup-validation

### (2) The Validator.nu (X)HTML5 Validator

下面是另一个当前已知的 HTML5 验证器。Henri 的 [Validator.nu \(X\)HTML5 Validator](#)（高度实验性的）。

**Validator.nu (X)HTML5 Validator (Highly Experimental)**

Validator Input

Address

☐ Show Image Report

☐ Show Source

[About this Service](#) • [More options](#)

图片 4.4 validator.nu-html5-validator

这个验证器有兼容性问题，可以在低版本的 IE 或者 Mozilla 中尝试。



## HTML5 在线编辑器

---

- [HTML5 Online Editor](#)

## HTML5 颜色代码生成器

---

通常我们需要使用不同的颜色装饰我们的 HTML 页面，HTML5 提供了大量的属性可以用来设置背景颜色，字体颜色等等。

要在 HTML5 页面中使用颜色需要有效的颜色代码。下面两个工具可以帮助我们生成有效的 HTML5 颜色代码。

### Java Servlet Color Picker

- 点击喜欢的颜色，拾取等价的颜色代码，然后在 HTML 代码或者 CSS 中使用。
- 这个调色板生成的所有代码都会是十六进制的。
- 这个颜色调色板需要在浏览器中启用 Java。

### jQuery Color Picker

- 点击喜欢的颜色，拾取等价的颜色代码，然后在 HTML 代码或者 CSS 中使用。
- 这个调色板生成的所有代码都会是十六进制的。



5

有用的 HTML5 资源



## 有用的 HTML5 资源

---

如果想要在这个页面上列出你的网站，书籍或者其他资源，请联系 [webmaster@tutorialspoint.com](mailto:webmaster@tutorialspoint.com)

- [HTML5](#) - W3C 规范。
- [WHATWG HTML5 草案](#) - WHATWG 社区的 HTML5 工作草案。
- [Canvas 教程 - MDC](#) - 学习如何在 Web 页面中使用 HTML Canvas 特性。
- [HTML5 介绍](#) - 学习如何在 Web 设计和 Web 应用程序中使用 HTML5。
- [W3C 规范列表](#) - W3C 规范列表大全。在这个页面上可以找到大多数最新的信息。
- [Web 标准清单](#) - Max Design 为标准领域提供了一个很好的资源：Web 标准清单。这个列表旨在帮助人们理解标准的广度以及为开发者提供工具。

### 有用的书籍

#### [书籍列表](#)

# 极客学院

jikexueyuan.com

中国最大的IT职业在线教育平台



更多信息请访问 

<http://wiki.jikexueyuan.com/project/html5/>