Notes Collection of HTML5

HTML

THEQIONG.COM

穷屌丝联盟

HTML5 Notes

穷屌丝联盟

theqiong.com

2014 年 3 月 10 日

Contents

List of Figures

List of Tables

Part I

HTML5

Chapter 1

Introduction

HTML5 is a markup language used for structuring and presenting content for the World Wide Web and a core technology of the Internet. It is the fifth revision of the HTML standard (created in 1990 and standardized as HTML 4 as of 1997) and, as of December 2012, is a candidate recommendation of the World Wide Web Consortium (W3C). Its core aims have been to improve the language with support for the latest multimedia while keeping it easily readable by humans and consistently understood by computers and devices (web browsers, parsers, etc.). HTML5 is intended to subsume not only HTML 4, but also XHTML 1 and DOM Level 2 HTML.

HTML 5 是下一代的 HTML，将成为 HTML、XHTML 以及 HTML DOM 的新标准。HTML5 现在仍处于完善之中，但是大部分现代浏览器已经具备了某些 HTML5 支持。

Following its immediate predecessors HTML 4.01 and XHTML 1.1, HTML5 is a response to the fact that the HTML and XHTML in common use on the World Wide Web are a mixture of features introduced by various specifications, along with those introduced by software products such as web browsers, those established by common practice, and the many syntax errors in existing web documents. It is also an attempt to define a single markup language that can be written in either HTML or XHTML syntax. It includes detailed processing models to encourage more interoperable implementations; it extends, improves and rationalises the markup available for documents, and introduces markup and application programming interfaces (APIs) for complex web applications. For the same reasons, HTML5 is also a potential candidate for cross-platform mobile applications. Many features of HTML5 have been built with the consideration of being able to run on low-powered devices such as smartphones and tablets. In December 2011, research firm Strategy Analytics forecast sales of HTML5 compatible phones will top 1 billion in 2013.

In particular, HTML5 adds many new syntactic features. These include the new <video>,

<audio> and <canvas> elements, as well as the integration of scalable vector graphics (SVG) content (that replaces the uses of generic <object> tags) and MathML for mathematical formulas. These features are designed to make it easy to include and handle multimedia and graphical content on the web without having to resort to proprietary plugins and APIs. Other new elements, such as <section>, <article>, <header> and <nav>, are designed to enrich the semantic content of documents. New attributes have been introduced for the same purpose, while some elements and attributes have been removed. Some elements, such as <a>, <cite> and <menu> have been changed, redefined or standardized. The APIs and Document Object Model (DOM) are no longer afterthoughts, but are fundamental parts of the HTML5 specification. HTML5 also defines in some detail the required processing for invalid documents so that syntax errors will be treated uniformly by all conforming browsers and other user agents.

Chapter 2

History

The Web Hypertext Application Technology Working Group (WHATWG) began work on the new standard in 2004. At that time, HTML 4.01 had not been updated since 2000, and the World Wide Web Consortium (W3C) was focusing future developments on XHTML 2.0. In 2009, the W3C allowed the XHTML 2.0 Working Group's charter to expire and decided not to renew it. W3C and WHATWG are currently working together on the development of HTML5.

While HTML5 is often compared to Flash, the two technologies are very different. Both include features for playing audio and video within web pages, and for using Scalable Vector Graphics. HTML5 on its own cannot be used for animation and interactivity —it must be supplemented with CSS3 or JavaScript. There are many Flash capabilities that have no direct counterpart in HTML5.

Although HTML5 has been well known among web developers for years, it became the topic of mainstream media around April 2010 after Apple Inc's then-CEO Steve Jobs issued a public letter titled "Thoughts on Flash" where he concludes that "[Adobe] Flash is no longer necessary to watch video or consume any kind of web content" and that "new open standards created in the mobile era, such as HTML5, will win". This sparked a debate in web development circles where some suggested that while HTML5 provides enhanced functionality, developers must consider the varying browser support of the different parts of the standard as well as other functionality differences between HTML5 and Flash. In early November 2011, Adobe announced that it will discontinue development of Flash for mobile devices and reorient its efforts in developing tools utilizing HTML5.

2.1 Standardization process

The Mozilla Foundation and Opera Software presented a position paper at a World Wide Web Consortium (W3C) workshop in June 2004, focusing on developing technologies that are backward compatible with existing browsers,[18] including an initial draft specification of Web Forms 2.0. The workshop concluded with a vote, 8 for, 14 against, for continuing work on HTML. Later that month, work based upon that position paper moved to the newly formed Web Hypertext Application Technology Working Group (WHATWG), and a second draft, Web Applications 1.0, was also announced. The two specifications were later merged to form HTML5. The HTML5 specification was adopted as the starting point of the work of the new HTML working group of the W3C in 2007.

- 2008 – First Public Working Draft
  WHATWG published the First Public Working Draft of the specification on 22 January 2008.[22] Parts of HTML5 have been implemented in browsers despite the whole specification not yet having reached final Recommendation status.
- 2011 – Last Call
  On 14 February 2011, the W3C extended the charter of its HTML Working Group with clear milestones for HTML5. In May 2011, the working group advanced HTML5 to ”Last Call”, an invitation to communities inside and outside W3C to confirm the technical soundness of the specification. The W3C is developing a comprehensive test suite to achieve broad interoperability for the full specification by 2014, which is now the target date for Recommendation.[23] In January 2011, the WHATWG renamed its ”HTML5” living standard to ”HTML”. The W3C nevertheless continues its project to release HTML5.
- 2012 – Candidate Recommendation
  In July 2012, WHATWG and W3C decided on a degree of separation. W3C will continue the HTML5 specification work, focusing on a single definitive standard, which is considered as a ”snapshot” by WHATWG. The WHATWG organization will continue its work with HTML5 as a ”Living Standard”. The concept of a living standard is that it is never complete and is always being updated and improved. New features can be added but functionality will not be removed.
  In December 2012, W3C designated HTML5 as a Candidate Recommendation. The criterion for advancement to W3C Recommendation is ”two 100% complete and fully interoperable implementations”.

HTML5 是 W3C[1]与 WHATWG[2]合作的结果。

WHATWG 致力于 web 表单和应用程序，而 W3C 专注于 XHTML 2.0。在 2006 年，双方决定进行合作，来创建一个新版本的 HTML，他们为 HTML5 建立的一些规则如下：

- 新特性应该基于 HTML、CSS、DOM 以及 JavaScript。
- 减少对外部插件的需求（比如 Flash）
- 更优秀的错误处理
- 更多取代脚本的标记
- HTML5 应该独立于设备
- 开发进程应对公众透明

陆续提出了 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

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。

2.2 Plan 2014

In September 2012, the W3C proposed a plan to release a stable HTML5 Recommendation by the end of 2014 and an HTML 5.1 specification Recommendation by the end of 2016.

2.2.1 Core HTML specification

The combined timelines for HTML 5.0, HTML 5.1 and HTML 5.2:

	2012	2013	2014	2015	2016
HTML 5.0	Candidate Rec	Call for Review	Recommendation		
HTML 5.1	1st Working Draft		Last Call	Candidate Rec	Recommendation
HTML 5.2				1st Working Draft	

The W3C proposed a greater reliance on modularity as a key part of the plan to make faster

[1]W3C 指 World Wide Web Consortium，万维网联盟。

[2]WHATWG 指 Web Hypertext Application Technology Working Group。

progress, meaning identifying specific features, either proposed or already existing in the spec, and advancing them as separate specifications. Some technologies that were originally defined in HTML5 itself are now defined in separate specifications:

- HTML Working Group – Microdata, HTML Canvas 2D Context
- Web Apps WG – Web Messaging, Web Workers, Web Storage, WebSocket API, Server-Sent Events
- IETF HyBi WG – WebSocket Protocol
- WebRTC WG – WebRTC
- W3C Web Media Text Tracks CG – WebVTT

Some specifications that were initially developed standalone have been adapted as HTML5 extensions or features by reference: SVG, MathML, WAI-ARIA.

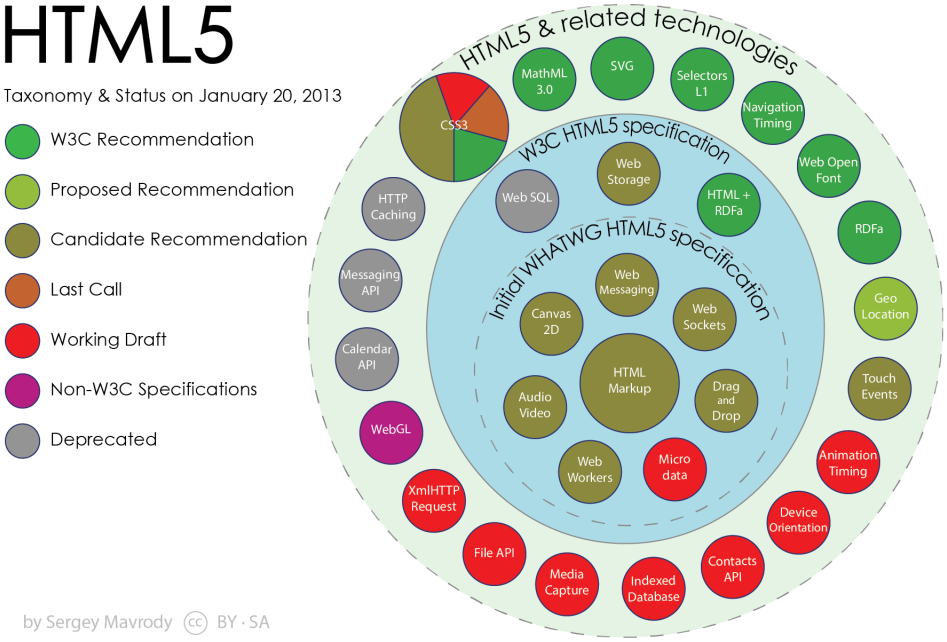

Figure 2.1: HTML5 APIs and related technologies

Chapter 3

Features

3.1 Markup

HTML5 introduces elements and attributes that reflect typical usage on modern websites. Some of them are semantic replacements for common uses of generic block ( ) and inline () elements, for example <nav> (website navigation block), <footer> (usually referring to bottom of web page or to last lines of HTML code), or <audio> and <video> instead of <object>. Some deprecated elements from HTML 4.01 have been dropped, including purely presentational elements such as <font> and , whose effects have long been superseded by the more capable Cascading Style Sheets. There is also a renewed emphasis on the importance of DOM scripting (e.g., JavaScript) in Web behavior.

The HTML5 syntax is no longer based on SGML despite the similarity of its markup. It has, however, been designed to be backward compatible with common parsing of older versions of HTML. It comes with a new introductory line that looks like an SGML document type declaration, <!DOCTYPE html>, which triggers the standards-compliant rendering mode. As of 5 January 2009, HTML5 also includes Web Forms 2.0, a previously separate WHATWG specification.

通过制定如何处理所有 HTML 元素以及如何从错误中恢复的精确规则，HTML 5 改进了互操作性，并减少了开发成本。

HTML 5 中增加了嵌入音频、视频和图形的功能，客户端数据存储，以及交互式文档。

HTML 5 还包含了新的元素，比如：<nav>, <header>, <footer> 以及 <figure> 等等。

3.1.1 HTML 5 Elements

下面的表格列出了所有的 HTML5/HTML 4.01/XHTML 元素，并定义了每个元素可以出现在哪种文档类型声明 (DTD) 中。

Table 3.1: HTML5/HTML 4.01/XHTML 元素与 DTD

Tag	HTML5	HTML 4.01 / XHTML 1.0			XHTML1.1
		Transitional	Strict	Frameset	
<a>	Yes	Yes	Yes	Yes	Yes
<abbr>	Yes	Yes	Yes	Yes	Yes
<acronym>	No	Yes	Yes	Yes	Yes
<address>	Yes	Yes	Yes	Yes	Yes
<applet>	No	Yes	No	Yes	No
<area>	Yes	Yes	Yes	Yes	No
<article>	Yes	No	No	No	No
<aside>	Yes	No	No	No	No
<audio>	Yes	No	No	No	No
<b>	Yes	Yes	Yes	Yes	Yes
<base>	Yes	Yes	Yes	Yes	Yes
<bdi>	Yes	No	No	No	No
<basefont>	No	Yes	No	Yes	No
<bdo>	Yes	Yes	Yes	Yes	No
<big>	No	Yes	Yes	Yes	Yes
<blockquote>	Yes	Yes	Yes	Yes	Yes
<body>	Yes	Yes	Yes	Yes	Yes
 	Yes	Yes	Yes	Yes	Yes
<button>	Yes	Yes	Yes	Yes	Yes
<canvas>	Yes	No	No	No	No
<caption>	Yes	Yes	Yes	Yes	Yes
	No	Yes	No	Yes	No
<cite>	Yes	Yes	Yes	Yes	Yes
<code>	Yes	Yes	Yes	Yes	Yes

...

...

Tag	HTML5	HTML 4.01 / XHTML 1.0			XHTML1.1
		Transitional	Strict	Frameset	
<col>	Yes	Yes	Yes	Yes	No
<colgroup>	Yes	Yes	Yes	Yes	No
<command>	Yes	No	No	No	No
<datalist>	Yes	No	No	No	No
<dd>	Yes	Yes	Yes	Yes	Yes
<del>	Yes	Yes	Yes	Yes	No
<details>	Yes	No	No	No	No
<dfn>	Yes	Yes	Yes	Yes	Yes
<dir>	No	Yes	No	Yes	No
	Yes	Yes	Yes	Yes	Yes
<dl>	Yes	Yes	Yes	Yes	Yes
<dt>	Yes	Yes	Yes	Yes	Yes
	Yes	Yes	Yes	Yes	Yes
<embed>	Yes	No	No	No	No
<fieldset>	Yes	Yes	Yes	Yes	Yes
<figcaption>	Yes	No	No	No	No
<figure>	Yes	No	No	No	No
<font>	No	Yes	No	Yes	No
<footer>	Yes	No	No	No	No
<form>	Yes	Yes	Yes	Yes	Yes
<frame>	No	No	No	Yes	No
<frameset>	No	No	No	Yes	No
<h1> to <h6>	Yes	Yes	Yes	Yes	Yes
<head>	Yes	Yes	Yes	Yes	Yes
<header>	Yes	No	No	No	No
<hgroup>	Yes	No	No	No	No
<hr>	Yes	Yes	Yes	Yes	Yes
<html>	Yes	Yes	Yes	Yes	Yes
<i>	Yes	Yes	Yes	Yes	Yes

...

...

Tag	HTML5	HTML 4.01 / XHTML 1.0			XHTML1.1
		Transitional	Strict	Frameset	
<iframe>	Yes	Yes	No	Yes	No
<img>	Yes	Yes	Yes	Yes	Yes
<input>	Yes	Yes	Yes	Yes	Yes
<ins>	Yes	Yes	Yes	Yes	No
<isindex>	No	Yes	No	Yes	No
<kbd>	Yes	Yes	Yes	Yes	Yes
<keygen>	Yes	No	No	No	No
<label>	Yes	Yes	Yes	Yes	Yes
<legend>	Yes	Yes	Yes	Yes	Yes
	Yes	Yes	Yes	Yes	Yes
<link>	Yes	Yes	Yes	Yes	Yes
<map>	Yes	Yes	Yes	Yes	No
<menu>	Yes	Yes	No	Yes	No
<meta>	Yes	Yes	Yes	Yes	Yes
<meter>	Yes	No	No	No	No
<nav>	Yes	No	No	No	No
<noframes>	No	Yes	No	Yes	No
<noscript>	Yes	Yes	Yes	Yes	Yes
<object>	Yes	Yes	Yes	Yes	Yes
	Yes	Yes	Yes	Yes	Yes
<optgroup>	Yes	Yes	Yes	Yes	Yes
<option>	Yes	Yes	Yes	Yes	Yes
<output>	Yes	No	No	No	No
<p>	Yes	Yes	Yes	Yes	Yes
<param>	Yes	Yes	Yes	Yes	Yes
<pre>	Yes	Yes	Yes	Yes	Yes
<progress>	Yes	No	No	No	No
<q>	Yes	Yes	Yes	Yes	Yes
<rp>	Yes	No	No	No	No

...

...

Tag	HTML5	HTML 4.01 / XHTML 1.0			XHTML1.1
		Transitional	Strict	Frameset	
<rt>	Yes	No	No	No	No
<ruby>	Yes	No	No	No	No
<s>	Yes	Yes	No	Yes	No
<samp>	Yes	Yes	Yes	Yes	Yes
<script>	Yes	Yes	Yes	Yes	Yes
<section>	Yes	No	No	No	No
<select>	Yes	Yes	Yes	Yes	Yes
<small>	Yes	Yes	Yes	Yes	Yes
<source>	Yes	No	No	No	No
	Yes	Yes	Yes	Yes	Yes
<strike>	No	Yes	No	Yes	No
	Yes	Yes	Yes	Yes	Yes
<style>	Yes	Yes	Yes	Yes	Yes
<sub>	Yes	Yes	Yes	Yes	Yes
<summary>	Yes	No	No	No	No
<sup>	Yes	Yes	Yes	Yes	Yes
<table>	Yes	Yes	Yes	Yes	Yes
<tbody>	Yes	Yes	Yes	Yes	No
<td>	Yes	Yes	Yes	Yes	Yes
<textarea>	Yes	Yes	Yes	Yes	Yes
<tfoot>	Yes	Yes	Yes	Yes	No
<th>	Yes	Yes	Yes	Yes	Yes
<thead>	Yes	Yes	Yes	Yes	No
<time>	Yes	No	No	No	No
<title>	Yes	Yes	Yes	Yes	Yes
<tr>	Yes	Yes	Yes	Yes	Yes
<track>	Yes	No	No	No	No
<tt>	No	Yes	Yes	Yes	Yes
<u>	No	Yes	No	Yes	No

...

...

Tag	HTML5	HTML 4.01 / XHTML 1.0			XHTML1.1
		Transitional	Strict	Frameset	
	Yes	Yes	Yes	Yes	Yes
<var>	Yes	Yes	Yes	Yes	Yes
<video>	Yes	No	No	No	No
<wbr>	Yes	No	No	No	No

3.1.2 HTML 5 Attributes

所有 HTML 5 标签均支持下面列出的属性[1]，仅有少数例外。

Table 3.2: HTML 5 Attributes

属性	值	描述
accesskey	character	规定访问元素的键盘快捷键
class	classname	规定元素的类名（用于规定样式表中的类)。
contenteditable	true false	规定是否允许用户编辑内容。
contextmenu	menu_id	规定元素的上下文菜单。
data-yourvalue	value	创作者定义的属性。 HTML 文档的创作者可以定义他们自己的属性。 必须以 “data-” 开头。
dir	ltr rtl	规定元素中内容的文本方向。
draggable	true false auto	规定是否允许用户拖动元素。
hidden	hidden	规定该元素是无关的。被隐藏的元素不会显示。
id	id	规定元素的唯一 ID。
item	empty url	用于组合元素。

[1]注释：HTML 4.01 不再支持 accesskey 属性：

属性	值	描述
itemprop	url group value	用于组合项目。
lang	language_code	规定元素中内容的语言代码。
spellcheck	true false	规定是否必须对元素进行拼写或语法检查。
style	style_definition	规定元素的行内样式。
subject	id	规定元素对应的项目。
tabindex	number	规定元素的 tab 键控制次序。
title	text	规定有关元素的额外信息。

3.1.3 HTML 5 Events

HTML 4 增加了通过事件触发浏览器中行为的能力，比如当用户点击某个元素时启动一段 JavaScript。

下面的表格列出了可插入 HTML 5 元素中以定义事件行为的标准事件属性。

- Window 事件属性 - Window Event Attributes
- 表单事件 - Form Events
- 键盘事件 - Keybord Events
- 鼠标事件 - Mouse Events
- 媒介事件 - Media Events

HTML 5 Window Event Attributes

window 对象触发的事件，适用于 <body> 标签。

Table 3.3: HTML 5 Window Event Attributes

属性	值	描述
onafterprint	script	在打印文档之后运行脚本
onbeforeprint	script	在文档打印之前运行脚本
onbeforeonload	script	在文档加载之前运行脚本
onblur	script	当窗口失去焦点时运行脚本

属性	值	描述
onerror	script	当错误发生时运行脚本
onfocus	script	当窗口获得焦点时运行脚本
onhaschange	script	当文档改变时运行脚本
onload	script	当文档加载时运行脚本
onmessage	script	当触发消息时运行脚本
onoffline	script	当文档离线时运行脚本
ononline	script	当文档上线时运行脚本
onpagehide	script	当窗口隐藏时运行脚本
onpageshow	script	当窗口可见时运行脚本
onpopstate	script	当窗口历史记录改变时运行脚本
onredo	script	当文档执行再执行操作（redo）时运行脚本
onresize	script	当调整窗口大小时运行脚本
onstorage	script	当文档加载加载时运行脚本
onundo	script	当 Web Storage 区域更新时（存储空间中的数据发生变化时）
onunload	script	当用户离开文档时运行脚本

HTML 5 Form Events

由 HTML 表单内部的动作触发的事件，适用于所有 HTML 5 元素，不过最常用于表单元素中。

Table 3.4: HTML 5 Form Events

属性	值	描述
onblur	script	当元素失去焦点时运行脚本
onchange	script	当元素改变时运行脚本
oncontextmenu	script	当触发上下文菜单时运行脚本
onfocus	script	当元素获得焦点时运行脚本
onformchange	script	当表单改变时运行脚本
onforminput	script	当表单获得用户输入时运行脚本
oninput	script	当元素获得用户输入时运行脚本

属性	值	描述
oninvalid	script	当元素无效时运行脚本
onreset	script	当表单重置时运行脚本。HTML 5 不支持。
onselect	script	当选取元素时运行脚本
onsubmit	script	当提交表单时运行脚本

HTML 5 Keyboard Events

由键盘触发的事件，适用于所有 HTML 5 元素。

Table 3.5: HTML 5 Keyboard Events

属性	值	描述
onkeydown	script	当按下按键时运行脚本
onkeypress	script	当按下并松开按键时运行脚本
onkeyup	script	当松开按键时运行脚本

HTML 5 Mouse Events

由鼠标或相似的用户动作触发的事件，适用于所有 HTML 5 元素。

Table 3.6: HTML 5 Mouse Events

属性	值	描述
onclick	script	当单击鼠标时运行脚本
ondblclick	script	当双击鼠标时运行脚本
ondrag	script	当拖动元素时运行脚本
ondragend	script	当拖动操作结束时运行脚本
ondragenter	script	当元素被拖动至有效的拖放目标时运行脚本
ondragleave	script	当元素离开有效拖放目标时运行脚本
ondragover	script	当元素被拖动至有效拖放目标上方时运行脚本
ondragstart	script	当拖动操作开始时运行脚本
ondrop	script	当被拖动元素正在被拖放时运行脚本
onmousedown	script	当按下鼠标按钮时运行脚本

属性	值	描述
onmousemove	script	当鼠标指针移动时运行脚本
onmouseout	script	当鼠标指针移出元素时运行脚本
onmouseover	script	当鼠标指针移至元素之上时运行脚本
onmouseup	script	当松开鼠标按钮时运行脚本
onmousewheel	script	当转动鼠标滚轮时运行脚本
onscroll	script	当滚动元素滚动元素的滚动条时运行脚本

HTML 5 Media Events

由视频、图像以及音频等媒介触发的事件，适用于所有 HTML 5 元素，不过在媒介元素（诸如 audio、embed、img、object 以及 video）中最常用。

Table 3.7: HTML 5 Media Events

属性	值	描述
onabort	script	当发生中止事件时运行脚本
oncanplay	script	当媒介能够开始播放但可能因缓冲而需要停止时运行脚本
oncanplaythrough	script	当媒介能够无需因缓冲而停止即可播放至结尾时运行脚本
ondurationchange	script	当媒介长度改变时运行脚本
onemptied	script	当媒介资源元素突然为空时（网络错误、加载错误等）运行脚本
onended	script	当媒介已抵达结尾时运行脚本
onerror	script	当在元素加载期间发生错误时运行脚本
onloadeddata	script	当加载媒介数据时运行脚本
onloadedmetadata	script	当媒介元素的持续时间以及其他媒介数据已加载时运行脚本
onloadstart	script	当浏览器开始加载媒介数据时运行脚本
onpause	script	当媒介数据暂停时运行脚本
onplay	script	当媒介数据将要开始播放时运行脚本
onplaying	script	当媒介数据已开始播放时运行脚本

属性	值	描述
onprogress	script	当浏览器正在取媒介数据时运行脚本
onratechange	script	当媒介数据的播放速率改变时运行脚本
onreadystatechange	script	当就绪状态（ready-state）改变时运行脚本
onseeked	script	当媒介元素的定位属性（seeking attribute）不再为真且定位已结束时运行脚本
onseeking	script	当媒介元素的定位属性为真且定位已开始时运行脚本
onstalled	script	当取回媒介数据过程中（延迟）存在错误时运行脚本
onsuspend	script	当浏览器已在取媒介数据但在取回整个媒介文件之前停止时运行脚本
ontimeupdate	script	当媒介改变其播放位置时运行脚本
onvolumechange	script	当媒介改变音量亦或当音量被设置为静音时运行脚本
onwaiting	script	当媒介已停止播放但打算继续播放时运行脚本

3.2 HTML 5 Input

HTML5 拥有多个新的表单输入类型，这些新特性提供了更好的输入控制和验证。

- email
- url
- number
- range
- Date pickers (date, month, week, time, datetime, datetime-local)
- search
- color

Table 3.8: HTML 5 Input 类型

Input type	IE	Firefox	Opera	Chrome	Safari
email	No	4.0	9.0	10.0	No

Input type	IE	Firefox	Opera	Chrome	Safari
url	No	4.0	9.0	10.0	No
number	No	No	9.0	7.0	No
range	No	No	9.0	4.0	4.0
Date pickers	No	No	9.0	10.0	No
search	No	4.0	11.0	10.0	No
color	No	No	11.0	No	No

Opera 对新的输入类型的支持最好。不过即使在其他浏览器中不被支持，仍然可以显示为常规的文本域。

3.2.1 email

email 类型用于应该包含 e-mail 地址的输入域。在提交表单时，会自动验证 email 域的值。

```
1 E-mail: <input type="email" name="user_email" />
```

iPhone 中的 Safari 浏览器支持 email 输入类型，并通过改变触摸屏键盘来配合它（添加 @ 和.com 选项）。

3.2.2 url

url 类型用于应该包含 URL 地址的输入域。在提交表单时，会自动验证 url 域的值。

```
1 Homepage: <input type="url" name="user_url" />
```

iPhone 中的 Safari 浏览器支持 url 输入类型，并通过改变触摸屏键盘来配合它（添加.com 选项）。

3.2.3 number

number 类型用于应该包含数值的输入域。另外，还可以设定对所接受的数字的限定：

```
1 Points: <input type="number" name="points" min="1" max="10" />
```

使用下面的属性来规定对数字类型的限定：

Table 3.9: HTML 5 数字类型的限定

属性	值	描述
max	number	规定允许的最大值
min	number	规定允许的最小值
step	number	规定合法的数字间隔（如果 step="3"，则合法的数是 -3,0,3,6 等）
value	number	规定默认值

iPhone 中的 Safari 浏览器支持 number 输入类型，并通过改变触摸屏键盘来配合它（显示数字）。

3.2.4 range

range 类型用于应该包含一定范围内数字值的输入域。range 类型显示为滑动条。另外，也能够设定对所接受的数字的限定：

```
1 <input type="range" name="points" min="1" max="10" />
```

使用下面的属性来规定对数字类型的限定：

Table 3.10: HTML 5 数字类型的限定

属性	值	描述
max	number	规定允许的最大值
min	number	规定允许的最小值
step	number	规定合法的数字间隔（如果 step="3"，则合法的数是 -3,0,3,6 等）
value	number	规定默认值

3.2.5 Date Pickers

HTML5 拥有多个可供选取日期和时间的新输入类型：

- date - 选取日、月、年
- month - 选取月、年
- week - 选取周和年
- time - 选取时间（小时和分钟）

- datetime - 选取时间、日、月、年（UTC 时间）
- datetime-local - 选取时间、日、月、年（本地时间）

下面的例子允许用户从日历中选取一个日期：

```
Date: <input type="date" name="user_date" />
```

3.2.6 search

search 类型用于搜索域，比如站点搜索或 Google 搜索。search 域显示为常规的文本域。

3.3 HTML 5 Form

HTML 5 新的表单元素包括：

- datalist
- keygen
- output

HTML5 拥有若干涉及表单的元素和属性。

Table 3.11: HTML 5 Form 类型

Input type	IE	Firefox	Opera	Chrome	Safari
datalist	No	No	9.5	No	No
keygen	No	No	10.5	3.0	No
output	No	No	9.5	No	No

3.3.1 datalist

datalist 元素规定输入域的选项列表。

列表是通过 datalist 内的 option 元素[2]创建的。

如需把 datalist 绑定到输入域，需要用输入域的 list 属性引用 datalist 的 id：

```
Webpage: <input type="url" list="url_list" name="link" />
<datalist id="url_list">
<option label="W3School" value="http://www.W3School.com.cn" />
<option label="Google" value="http://www.google.com" />
```

[2]提示：option 元素永远都要设置 value 属性。

```
<option label="Microsoft" value="http://www.microsoft.com" />
</datalist>
```

3.3.2 keygen

keygen 元素的作用是提供一种验证用户的可靠方法[3]。

keygen 元素是密钥对生成器（key-pair generator）。当提交表单时，会生成两个键，一个是私钥，一个公钥。

私钥（private key）存储于客户端，公钥（public key）则被发送到服务器。公钥可用于之后验证用户的客户端证书（client certificate）。

```
<form action="demo_form.asp" method="get">
Username: <input type="text" name="usr_name" />
Encryption: <keygen name="security" />
<input type="submit" />
</form>
```

3.3.3 output

output 元素用于不同类型的输出，比如计算或脚本输出：

```
<output id="result" onforminput="resCalc()"></output>
```

Table 3.12: HTML 5 表单属性

Input type	IE	Firefox	Opera	Chrome	Safari
autocomplete	8.0	3.5	9.5	3.0	4.0
autofocus	No	No	10.0	3.0	4.0
form	No	No	9.5	No	No
form overrides	No	No	10.5	No	No
height and width	8.0	3.5	9.5	3.0	4.0
list	No	No	9.5	No	No
min, max and step	No	No	9.5	3.0	No
multiple	No	3.5	No	3.0	4.0
novalidate	No	No	No	No	No

[3]目前，浏览器对此元素的糟糕的支持度不足以使其成为一种有用的安全标准。

Input type	IE	Firefox	Opera	Chrome	Safari
pattern	No	No	9.5	3.0	No
placeholder	No	No	No	3.0	3.0
required	No	No	9.5	3.0	No

3.3.4 autocomplete

autocomplete[4] 属性规定 form 或 input 域应该拥有自动完成功能。

当用户在自动完成域中开始输入时，浏览器应该在该域中显示填写的选项：

```
<form action="demo_form.asp" method="get" autocomplete="on">
First name: <input type="text" name="fname" /><br />
Last name: <input type="text" name="lname" /><br />
E-mail: <input type="email" name="email" autocomplete="off" /><br />
<input type="submit" />
</form>
```

注释：在某些浏览器中，可能需要启用自动完成功能，以使该属性生效。

3.3.5 autofocus

autofocus[5] 属性规定在页面加载时，域自动地获得焦点。

```
User name: <input type="text" name="user_name" autofocus="autofocus" />
```

3.3.6 form

form[6] 属性规定输入域所属的一个或多个表单。

form 属性必须引用所属表单的 id：

```
<form action="demo_form.asp" method="get" id="user_form">
First name:<input type="text" name="fname" />
<input type="submit" />
</form>
Last name: <input type="text" name="lname" form="user_form" />
```

注释：如需引用一个以上的表单，需要使用空格分隔的列表。

[4]注释：autocomplete 适用于 <form> 标签，以及以下类型的 <input> 标签：text, search, url, telephone, email, password, datepickers, range 以及 color。

[5]注释：autofocus 属性适用于所有 <input> 标签的类型。

[6]注释：form 属性适用于所有 <input> 标签的类型。

3.3.7 override

表单重写属性（form override attributes）[7]允许开发者重写 form 元素的某些属性设定。表单重写属性有：

- formaction - 重写表单的 action 属性
- formenctype - 重写表单的 enctype 属性
- formmethod - 重写表单的 method 属性
- formnovalidate - 重写表单的 novalidate 属性
- formtarget - 重写表单的 target 属性

```
<form action="demo_form.asp" method="get" id="user_form">
E-mail: <input type="email" name="userid" /><br />
<input type="submit" value="Submit" />
<br />
<input type="submit" formaction="demo_admin.asp" value="Submit as admin" />
<br />
<input type="submit" formnovalidate="true" value="Submit without validation" />
<br />
</form>
```

注释：这些属性对于创建不同的提交按钮很有帮助。

3.3.8 height & width

height[8] 和 width[9] 属性规定用于 image 类型的 input 标签的图像高度和宽度。

```
<input type="image" src="img_submit.gif" width="99" height="99" />
```

3.3.9 list

list[10] 属性规定输入域的 datalist，datalist 是输入域的选项列表。

```
Webpage: <input type="url" list="url_list" name="link" />
<datalist id="url_list">
<option label="W3Schools" value="http://www.w3school.com.cn" />
<option label="Google" value="http://www.google.com" />
<option label="Microsoft" value="http://www.microsoft.com" />
```

[7]注释：表单重写属性适用于以下类型的 <input> 标签：submit 和 image。
[8]注释：height 和 width 属性只适用于 image 类型的 <input> 标签。
[9]注释：height 和 width 属性只适用于 image 类型的 <input> 标签。
[10]注释：list 属性适用于以下类型的 <input> 标签：text, search, url, telephone, email, date pickers, number, range 以及 color。

```
</datalist>
```

3.3.10 min, max, step

min[11]、max[12] 和 step[13] 属性用于为包含数字或日期的 input 类型规定限定（约束），其中：

- max 属性规定输入域所允许的最大值。
- min 属性规定输入域所允许的最小值。
- step 属性为输入域规定合法的数字间隔（如果 step="3"，则合法的数是 -3,0,3,6 等）。

下面的例子显示一个数字域，该域接受介于 0 到 10 之间的值，且步进为 3（即合法的值为 0、3、6 和 9）：

```
Points: <input type="number" name="points" min="0" max="10" step="3" />
```

3.3.11 multiple

multiple[14] 属性规定输入域中可选择多个值。

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

3.3.12 novalidate

novalidate[15] 属性规定在提交表单时不应该验证 form 或 input 域。

```
<form action="demo_form.asp" method="get" novalidate="true">
E-mail: <input type="email" name="user_email" />
<input type="submit" />
</form>
```

3.3.13 pattern

pattern[16] 属性规定用于验证 input 域的模式（pattern），模式（pattern）是正则表达式。

[11]注释：min、max 和 step 属性适用于以下类型的 <input> 标签：date pickers、number 以及 range。
[12]注释：min、max 和 step 属性适用于以下类型的 <input> 标签：date pickers、number 以及 range。
[13]注释：min、max 和 step 属性适用于以下类型的 <input> 标签：date pickers、number 以及 range。
[14]注释：multiple 属性适用于以下类型的 <input> 标签：email 和 file。
[15]注释：novalidate 属性适用于 <form> 以及以下类型的 <input> 标签：text, search, url, telephone, email, password, date pickers, range 以及 color.
[16]注释：pattern 属性适用于以下类型的 <input> 标签：text, search, url, telephone, email 以及 password。

下面的例子显示了一个只能包含三个字母的文本域（不含数字及特殊字符）：

```
Country code: <input type="text" name="country_code"
pattern="[A-z]{3}" title="Three letter country code" />
```

3.3.14 placeholder

placeholder[17] 属性提供一种提示（hint），描述输入域所期待的值。
提示（hint）会在输入域为空时显示出现，会在输入域获得焦点时消失：

```
<input type="search" name="user_search" placeholder="Search" />
```

3.3.15 required

required[18] 属性规定必须在提交之前填写输入域（不能为空）。

```
Name: <input type="text" name="usr_name" required="required" />
```

3.4 APIs

In addition to specifying markup, HTML5 specifies scripting application programming interfaces (APIs) that can be used with JavaScript. Existing document object model (DOM) interfaces are extended and de facto features documented. There are also new APIs, such as:

- The canvas element for immediate mode 2D drawing.
- Timed media playback
- Offline Web Applications
- Document editing
- Drag-and-drop
- Cross-document messaging
- Browser history management
- MIME type and protocol handler registration
- Microdata
- Web Storage, a key-value pair storage framework that provides behaviour similar to cookies but with larger storage capacity and improved API

[17]注释：placeholder 属性适用于以下类型的 <input> 标签：text, search, url, telephone, email 以及 password。
[18]注释：required 属性适用于以下类型的 <input> 标签：text, search, url, telephone, email, password, date pickers, number, checkbox, radio 以及 file。

Not all of the above technologies are included in the W3C HTML5 specification, though they are in the WHATWG HTML specification. Some related technologies, which are not part of either the W3C HTML5 or the WHATWG HTML specification, are as follows. The W3C publishes specifications for these separately:

- Geolocation
- Web SQL Database, a local SQL Database (no longer maintained).
- The Indexed Database API, an indexed hierarchical key-value store (formerly WebSimpleDB).
- HTML5 File API, handles file uploads and file manipulation.
- Directories and System, an API intended to satisfy client-side-storage use cases not well served by databases.
- File Writer, an API for writing to files from web applications.
- Web Audio API, a high-level JavaScript API for processing and synthesizing audio in web applications.

HTML5 alone cannot provide animation within web pages. Either JavaScript or CSS3 is necessary for animating HTML elements. Animation is also possible using JavaScript and HTML 4, and within SVG elements through SMIL, although browser support of the latter remains uneven as of 2011.

3.5 2D drawing

HTML5 的 canvas 元素使用 JavaScript 在网页上绘制图像，canvas 拥有多种绘制路径、矩形、圆形、字符以及添加图像的方法。

画布是一个矩形区域，开发者可以控制其每一像素。不过，<canvas> 元素本身并没有绘制能力（它仅仅是图形的容器），开发者必须使用脚本来完成实际的绘图任务[19]。

向 HTML5 页面添加 canvas 元素时，要规定元素的 id、宽度和高度：

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

3.5.1 rectangle

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：

*/,

[19]Internet Explorer 9、Firefox、Opera、Chrome 以及 Safari 支持 <canvas> 及其属性和方法。Internet Explorer 8 以及更早的版本不支持 <canvas> 元素。

```
<script type="text/javascript">
  var c=document.getElementById("myCanvas");
  var cxt=c.getContext("2d");
  cxt.fillStyle="#FF0000";
  cxt.fillRect(0,0,150,75);
</script>
```

getContext() 方法可返回一个对象，该对象提供了用于在画布上绘图的方法和属性。

JavaScript 使用 id 来寻找 canvas 元素：

*/,

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

然后，创建 context 对象：

*/,

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

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

下面的两行代码绘制一个红色的矩形：

*/,

```
cxt.fillStyle="#FF0000";
cxt.fillRect(0,0,150,75);
```

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

这里的 fillRect 方法拥有参数 (0,0,150,75)，意义是在画布上绘制 150×75 的矩形，从左上角开始 (0,0)。

如下图所示，画布的 X 和 Y 坐标用于在画布上对绘画进行定位。

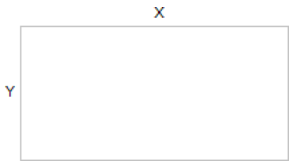

Figure 3.1: 画布的 X 和 Y 坐标

3.5.2 line

下面的 JavaScript 代码通过指定从何处开始，在何处结束，来绘制一条线：

*/,

```
<script type="text/javascript">
  var c=document.getElementById("drawline");
  var cxt=c.getContext("2d");
  cxt.moveTo(10,10);
  cxt.lineTo(150,50);
  cxt.lineTo(10,50);
  cxt.stroke();
</script>
```

相应的 canvas 元素：

```
<canvas id="drawline" width="200" height="100"
style="border:1px solid #c3c3c3;">
Your browser does not support the canvas element.
</canvas>
```

3.5.3 circle

通过规定尺寸、颜色和位置，来绘制一个圆，JavaScript 代码如下：

*/,

```
<script type="text/javascript">
  var c=document.getElementById("drawcircle");
  var cxt=c.getContext("2d");
  cxt.fillStyle="#FF0000";
  cxt.beginPath();
  cxt.arc(70,18,15,0,Math.PI*2,true);
  cxt.closePath();
  cxt.fill();
</script>
```

相应的 canvas 元素：

```
<canvas id="drawcircle" width="200" height="100"
style="border:1px solid #c3c3c3;">
Your browser does not support the canvas element.
</canvas>
```

3.5.4 gradient

使用指定的颜色来绘制渐变背景，JavaScript 代码如下：

*/,

```
<script type="text/javascript">
  var c=document.getElementById("myCanvas");
  var cxt=c.getContext("2d");
  var grd=cxt.createLinearGradient(0,0,175,50);
  grd.addColorStop(0,"#FF0000");
  grd.addColorStop(1,"#00FF00");
  cxt.fillStyle=grd;
  cxt.fillRect(0,0,175,50);
</script>
```

相应的 canvas 元素：

```
<canvas id="myCanvas" width="200" height="100"
style="border:1px solid #c3c3c3;">
Your browser does not support the canvas element.
</canvas>
```

3.5.5 image

通过 canvas 预算，可以把一幅图像放置到画布上，JavaScript 代码如下：

*/,

```
<script type="text/javascript">
  var c=document.getElementById("myCanvas");
  var cxt=c.getContext("2d");
  var img=new Image()
  img.src="flower.png"
  cxt.drawImage(img,0,0);
</script>
```

相应的 canvas 元素：

```
<canvas id="myCanvas" width="200" height="100"
style="border:1px solid #c3c3c3;">
Your browser does not support the canvas element.
</canvas>
```

Table 3.13: HTML5 Canvas 属性

属性	描述
颜色、样式和阴影	

属性	描述
fillStyle	设置或返回用于填充绘画的颜色、渐变或模式
strokeStyle	设置或返回用于笔触的颜色、渐变或模式
shadowColor	设置或返回用于阴影的颜色
shadowBlur	设置或返回用于阴影的模糊级别
shadowOffsetX	设置或返回阴影距形状的水平距离
shadowOffsetY	设置或返回阴影距形状的垂直距离
线条样式	
lineCap	设置或返回线条的结束端点样式
lineJoin	设置或返回两条线相交时，所创建的拐角类型
lineWidth	设置或返回当前的线条宽度
miterLimit	设置或返回最大斜接长度
文本	
font	设置或返回文本内容的当前字体属性
textAlign	设置或返回文本内容的当前对齐方式
textBaseline	设置或返回在绘制文本时使用的当前文本基线
像素操作	
width	返回 ImageData 对象的宽度
height	返回 ImageData 对象的高度
data	返回一个对象，其包含指定的 ImageData 对象的图像数据
合成	
globalAlpha	设置或返回绘图的当前 alpha 或透明值
globalCompositeOperation	设置或返回新图像如何绘制到已有的图像上

Table 3.14: HTML5 Canvas 方法

方法	描述
颜色、样式和阴影	
createLinearGradient()	创建线性渐变（用在画布内容上）
createPattern()	在指定的方向上重复指定的元素
createRadialGradient()	创建放射状/环形的渐变（用在画布内容上）

方法	描述
addColorStop()	规定渐变对象中的颜色和停止位置
矩形	
rect()	创建矩形
fillRect()	绘制“被填充”的矩形
strokeRect()	绘制矩形（无填充）
clearRect()	在给定的矩形内清除指定的像素
路径	
fill()	填充当前绘图（路径）
stroke()	绘制已定义的路径
beginPath()	起始一条路径，或重置当前路径
moveTo()	把路径移动到画布中的指定点，不创建线条
closePath()	创建从当前点回到起始点的路径
lineTo()	添加一个新点，然后在画布中创建从该点到最后指定点的线条
clip()	从原始画布剪切任意形状和尺寸的区域
quadraticCurveTo()	创建二次贝塞尔曲线
bezierCurveTo()	创建三次方贝塞尔曲线
arc()	创建弧/曲线（用于创建圆形或部分圆）
arcTo()	创建两切线之间的弧/曲线
isPointInPath()	如果指定的点位于当前路径中，则返回 true，否则返回 false
转换	
scale()	缩放当前绘图至更大或更小
rotate()	旋转当前绘图
translate()	重新映射画布上的 (0,0) 位置
transform()	替换绘图的当前转换矩阵
setTransform()	将当前转换重置为单位矩阵。然后运行 transform()
文本	
fillText()	在画布上绘制“被填充的”文本
strokeText()	在画布上绘制文本（无填充）

方法	描述
measureText()	返回包含指定文本宽度的对象
图像绘制	
drawImage()	向画布上绘制图像、画布或视频
像素操作	
createImageData()	创建新的、空白的 ImageData 对象
getImageData()	返回 ImageData 对象，该对象为画布上指定的矩形复制像素数据
putImageData()	把图像数据（从指定的 ImageData 对象）放回画布上
其他	
save()	保存当前环境的状态
restore()	返回之前保存过的路径状态和属性
createEvent()	
getContext()	
toDataURL()	

3.6 Canvas & SVG

HTML5 同时支持内联 SVG 和画布，SVG 元素可以直接嵌入 HTML 页面中：

```
<!DOCTYPE html>
<html>
<body>
<svg xmlns="http://www.w3.org/2000/svg" version="1.1" height="190">
  <polygon points="100,10 40,180 190,60 10,60 160,180"
  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />
</svg>
</body>
</html>
```

SVG 相比其他图像格式（比如 JPEG 和 GIF）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

Internet Explorer 9、Firefox、Opera、Chrome 以及 Safari 支持内联 SVG。

尽管 Canvas 和 SVG 都允许开发者在浏览器中创建图形，但是它们在根本上是不同的。

SVG 是一种使用 XML 描述 2D 图形的语言。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，因此基于 XML 的 SVG DOM 中的每个元素都是可用的，开发者可以为某个元素附加 JavaScript 事件处理器。

Canvas 通过 JavaScript 来绘制 2D 图形，而且 Canvas 是逐像素进行渲染的，因此在 <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。

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

3.7 HTML 5 Media

HTML5 提供了播放音频/视频的标准。

直到现在，仍然不存在一项旨在网页上播放音频/视频的标准。大多数音频/视频是通过插件（比如 Flash、Quicktime 等）来播放的。然而，并非所有浏览器都拥有同样的插件。

HTML5 规定了通过 audio/video 元素来包含音频/视频的标准方法。audio 元素能够播放声音文件或者音频流，video 元素用来播放视频文件或者视频流。

3.7.1 <audio></audio>

当前，audio 元素支持三种音频格式：

- Ogg Vorbis
- MP3
- Wav

要在 HTML5 中播放音频，开发者所有需要的是：

```
<audio src="song.ogg" controls="controls"></audio>
```

其中，control 属性供添加播放、暂停和音量控件。<audio> 与 </audio> 之间插入的内容是供不支持 audio 元素的浏览器显示的：

```
<audio src="song.ogg" controls="controls">
Your browser does not support the audio tag.
</audio>
```

上面的例子使用一个 Ogg 文件，适用于 Firefox、Opera 以及 Chrome 浏览器。

要确保适用于 Safari 浏览器，音频文件必须是 MP3 或 Wav 类型。

audio 元素允许多个 source 元素。source 元素可以链接不同的音频文件。浏览器将使用第一个可识别的格式：

```
<audio controls="controls">
  <source src="song.ogg" type="audio/ogg">
  <source src="song.mp3" type="audio/mpeg">
Your browser does not support the audio tag.
</audio>
```

Internet Explorer 8 不支持 audio 元素。在 IE 9 中，将提供对 audio 元素的支持。

3.7.2 <video></video>

当前，video 元素支持三种视频格式：

- Ogg
- MPEG 4
- WebM

其中：

* Ogg = 带有 Theora 视频编码和 Vorbis 音频编码的 Ogg 文件
* MPEG4 = 带有 H.264 视频编码和 AAC 音频编码的 MPEG 4 文件
* WebM = 带有 VP8 视频编码和 Vorbis 音频编码的 WebM 文件

要在 HTML5 中显示视频，最简单的代码是：

```
<video src="movie.ogg" controls="controls"></video>
```

其中，control 属性供添加播放、暂停和音量控件，而且包含宽度和高度属性也是不错的主意。

<video> 与 </video> 之间插入的内容是供不支持 video 元素的浏览器显示的：

```
<video src="movie.ogg" width="320" height="240" controls="controls">
Your browser does not support the video tag.
</video>
```

上面的例子使用一个 Ogg 文件，适用于 Firefox、Opera 以及 Chrome 浏览器。若要适用于 Safari 浏览器，视频文件必须是 MPEG4 类型。

video 元素允许多个 source 元素。source 元素可以链接不同的视频文件，浏览器将使用第一个可识别的格式：

```
<video width="320" height="240" controls="controls">
  <source src="movie.ogg" type="video/ogg">
  <source src="movie.mp4" type="video/mp4">
Your browser does not support the video tag.
</video>
```

Internet Explorer 8 不支持 video 元素。在 IE 9 中，将提供对使用 MPEG4 的 video 元素的支持。

HTML5 <video> 元素同样拥有方法、属性和事件。其中，方法用于播放、暂停以及加载等，属性（比如时长、音量等）可以被读取或设置，而 DOM 事件能够通知用户，比方说，<video> 元素开始播放、已暂停，已停止，等等。

下例中简单的方法演示了如何使用 <video> 元素，读取并设置属性，以及如何调用方法。

```
<div style="text-align:center;">
  <button onclick="playPause()">播放/暂停</button>
  <button onclick="makeBig()">大</button>
  <button onclick="makeNormal()">中</button>
  <button onclick="makeSmall()">小</button>
  <br />
  <video id="video1" width="420" style="margin-top:15px;">
    <source src="/example/html5/mov_bbb.mp4" type="video/mp4" />
    <source src="/example/html5/mov_bbb.ogg" type="video/ogg" />
    Your browser does not support HTML5 video.
  </video>
</div>

```

```
<script type="text/javascript">
  var myVideo=document.getElementById("video1");
  function playPause()
  {
  if (myVideo.paused)
    myVideo.play();
  else
   myVideo.pause();
  }

  function makeBig()
  {
    myVideo.width=560;
  }

  function makeSmall()
  {
    myVideo.width=320;
  }

  function makeNormal()
  {
    myVideo.width=420;
  }
</script>
```

上面的例子调用了两个方法：play() 和 pause()，它同时使用了两个属性：paused 和 width。

HTML5 DOM 为 <audio> 和 <video> 元素提供了方法、属性和事件。这些方法、属性和事件允许开发者使用 JavaScript 来操作 <audio> 和 <video> 元素。

Table 3.15: HTML5 Audio/Video 方法

方法	描述
addTextTrack()	向音频/视频添加新的文本轨道
canPlayType()	检测浏览器是否能播放指定的音频/视频类型
load()	重新加载音频/视频元素
play()	开始播放音频/视频

方法	描述
pause()	暂停当前播放的音频/视频

Table 3.16: HTML5 Audio/Video 属性

属性	描述
audioTracks	返回表示可用音轨的 AudioTrackList 对象
autoplay	设置或返回是否在加载完成后随即播放音频/视频
buffered	返回表示音频/视频已缓冲部分的 TimeRanges 对象
controller	返回表示音频/视频当前媒体控制器的 MediaController 对象
controls	设置或返回音频/视频是否显示控件（比如播放/暂停等）
crossOrigin	设置或返回音频/视频的 CORS 设置
currentSrc	返回当前音频/视频的 URL
currentTime	设置或返回音频/视频中的当前播放位置（以秒计）
defaultMuted	设置或返回音频/视频默认是否静音
defaultPlaybackRate	设置或返回音频/视频的默认播放速度
duration	返回当前音频/视频的长度（以秒计）
ended	返回音频/视频的播放是否已结束
error	返回表示音频/视频错误状态的 MediaError 对象
loop	设置或返回音频/视频是否应在结束时重新播放
mediaGroup	设置或返回音频/视频所属的组合（用于连接多个音频/视频元素）
muted	设置或返回音频/视频是否静音
networkState	返回音频/视频的当前网络状态
paused	设置或返回音频/视频是否暂停
playbackRate	设置或返回音频/视频播放的速度
played	返回表示音频/视频已播放部分的 TimeRanges 对象
preload	设置或返回音频/视频是否应该在页面加载后进行加载 如果使用 “autoplay”，则忽略该属性。
readyState	返回音频/视频当前的就绪状态
seekable	返回表示音频/视频可寻址部分的 TimeRanges 对象

属性	描述
seeking	返回用户是否正在音频/视频中进行查找
src	设置或返回音频/视频元素的当前来源
startDate	返回表示当前时间偏移的 Date 对象
textTracks	返回表示可用文本轨道的 TextTrackList 对象
videoTracks	返回表示可用视频轨道的 VideoTrackList 对象
volume	设置或返回音频/视频的音量

Table 3.17: HTML5 Audio/Video 事件

事件	描述
abort	当音频/视频的加载已放弃时
canplay	当浏览器可以播放音频/视频时
canplaythrough	当浏览器可在不因缓冲而停顿的情况下进行播放时
durationchange	当音频/视频的时长已更改时
emptied	当目前的播放列表为空时
ended	当目前的播放列表已结束时
error	当在音频/视频加载期间发生错误时
loadeddata	当浏览器已加载音频/视频的当前帧时
loadedmetadata	当浏览器已加载音频/视频的元数据时
loadstart	当浏览器开始查找音频/视频时
pause	当音频/视频已暂停时
play	当音频/视频已开始或不再暂停时
playing	当音频/视频在已因缓冲而暂停或停止后已就绪时
progress	当浏览器正在下载音频/视频时
ratechange	当音频/视频的播放速度已更改时
seeked	当用户已移动/跳跃到音频/视频中的新位置时
seeking	当用户开始移动/跳跃到音频/视频中的新位置时
stalled	当浏览器尝试获取媒体数据，但数据不可用时
suspend	当浏览器刻意不获取媒体数据时
timeupdate	当目前的播放位置已更改时

事件	描述
volumechange	当音量已更改时
waiting	当视频由于需要缓冲下一帧而停止

注释：在所有属性中，只有 videoWidth 和 videoHeight 属性是立即可用的。在视频的元数据已加载后，其他属性才可用。

3.8 Cache Manifest

HTML5 引入了应用程序缓存（Application Cache）[20]，这意味着 web 应用可进行缓存，并可在没有因特网连接时进行访问。使用 HTML5，通过创建 cache manifest 文件，从而可以轻松地创建 web 应用的离线版本。

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。

要启用应用程序缓存，需要在文档的 <html> 标签中包含 manifest 属性，下面的示例展示了带有 cache manifest 的 HTML 文档（供离线浏览）：

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">
<body>
The content of the document......
</body>
</html>
```

每个指定了 manifest 的页面在用户对其访问时都会被缓存。如果未指定 manifest 属性，则页面不会被缓存（除非在 manifest 文件[21]中直接指定了该页面）。

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 文件可分为三个部分：

- 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）

[20]所有主流浏览器均支持应用程序缓存，除了 Internet Explorer。

[21]manifest 文件的建议的文件扩展名是：".appcache"。

manifest 文件需要配置正确的 MIME-type，即”text/cache-manifest”，而且必须在 web 服务器上进行配置。

一个完整的 Manifest 文件[22]如下：

```
CACHE MANIFEST
# 2012-02-21 v1.0.0
/theme.css
/logo.gif
/main.js

NETWORK:
login.asp

FALLBACK:
/html5/ /404.html
```

其中：

1. CACHE MANIFEST

   第一行，CACHE MANIFEST，是必需的：

   ```
   CACHE MANIFEST
   /theme.css
   /logo.gif
   /main.js
   ```

   上面的 manifest 文件列出了三个资源：一个 CSS 文件，一个 GIF 图像，以及一个 JavaScript 文件。当 manifest 文件加载后，浏览器会从网站的根目录下载这三个文件。然后，无论用户何时与因特网断开连接，这些资源依然是可用的。

2. NETWORK

   下面的 NETWORK 小节规定文件”login.asp” 永远不会被缓存，且离线时是不可用的：

   ```
   NETWORK:
   login.asp
   ```

   可以使用星号来指示所有其他其他资源/文件都需要因特网连接：

   ```
   NETWORK:
   *
   ```

[22]注释：以”#” 开头的是注释行，但也可满足其他用途。应用的缓存会在其 manifest 文件更改时被更新。如果您编辑了一幅图片，或者修改了一个 JavaScript 函数，这些改变都不会被重新缓存。更新注释行中的日期和版本号是一种使浏览器重新缓存文件的办法。

3. FALLBACK

   下面的 FALLBACK 小节规定如果无法建立因特网连接，则用 “offline.html”[23] 替代 /html5/ 目录中的所有文件：

```
1 FALLBACK:
2 /html5/ /404.html
```

一旦应用被缓存，则浏览器会继续展示已缓存的版本，即使后来修改了服务器上的文件，浏览器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[24]
- 由程序来更新应用缓存

另外，浏览器对缓存数据的容量限制可能不太一样（某些浏览器设置的限制是每个站点 5MB）。

3.9 Drag and drop

拖放[25]（Drag 和 drop）是一种常见的特性，即抓取对象以后拖到另一个位置，现在拖放是 HTML5 标准的组成部分，任何元素都能够拖放。

下面的例子是一个简单的拖放实例：

```
1  <!DOCTYPE HTML>
2  <html>
3  <head>
4  <script type="text/javascript">
5  function allowDrop(ev)
6  {
7  ev.preventDefault();
8  }
9
10 function drag(ev)
11 {
12 ev.dataTransfer.setData("Text",ev.target.id);
13 }
14
```

[23]第一个 URI 是资源，第二个是替补。

[24]一旦文件被缓存，则浏览器会继续展示已缓存的版本，即使修改了服务器上的文件。为了确保浏览器更新缓存，需要更新 manifest 文件。

[25]Internet Explorer 9、Firefox、Opera 12、Chrome 以及 Safari 5 支持拖放，但是在 Safari 5.1.2 中不支持拖放。

```
function drop(ev)
{
ev.preventDefault();
var data=ev.dataTransfer.getData("Text");
ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<div id="div1" ondrop="drop(event)"
ondragover="allowDrop(event)"></div>
<img id="drag1" src="img_logo.gif" draggable="true"
ondragstart="drag(event)" width="336" height="69" />

</body>
</html>
```

1. draggable

   首先，为了使元素可拖动，把 draggable 属性设置为 true：

   ```
   <img draggable="true" />
   ```

2. ondragstart 和 setData()

   然后，ondragstart 和 setData() 规定当元素被拖动时，会发生什么。在上面的例子中，ondragstart 属性调用了一个函数，drag(event)，它规定了被拖动的数据。

   dataTransfer.setData() 方法设置被拖数据的数据类型和值：

   */,

   ```
   function drag(ev)
   {
     ev.dataTransfer.setData("Text",ev.target.id);
   }
   ```

   在这个例子中，数据类型是”Text”，值是可拖动元素的 id (”drag1”)。

   默认地，无法将数据/元素放置到其他元素中。如果需要设置允许放置，我们必须阻止对元素的默认处理方式，使用 ondragover 事件规定在何处放置被拖动的数据。

3. ondragover

   下面通过调用 ondragover 事件的 event.preventDefault() 方法来设置允许放置：

   */,

   ```
   event.preventDefault()
   ```

4. ondrop

   当放置被拖数据时，会发生 ondrop 事件。在上面的例子中，ondrop 属性调用了一个函数，drop(event)：

   */,

```
function drop(ev)
{
  ev.preventDefault();
  var data=ev.dataTransfer.getData("Text");
  ev.target.appendChild(document.getElementById(data));
}
```

最后，代码解释：

- 调用 preventDefault() 来避免浏览器对数据的默认处理（drop 事件的默认行为是以链接形式打开）
- 通过 dataTransfer.getData(”Text”) 方法获得被拖的数据。该方法将返回在 setData() 方法中设置为相同类型的任何数据。
- 被拖数据是被拖元素的 id (”drag1”)
- 把被拖元素追加到放置元素（目标元素）中

3.10 Web storage

HTML5 提供了两种在客户端存储数据的新方法：

- localStorage - 没有时间限制的数据存储
- sessionStorage - 针对一个 session 的数据存储

之前，这些都是由 cookie 完成的。但是 cookie 不适合大量数据的存储，因为它们由每个对服务器的请求来传递，这使得 cookie 速度很慢而且效率也不高。

在 HTML5 中，HTML5 使用 JavaScript 来存储和访问数据，而且数据不是由每个服务器请求传递的，而是只有在请求时使用数据，这样就使得在不影响网站性能的情况下存储大量数据成为可能。

对于不同的网站，数据存储于不同的区域，并且一个网站只能访问其自身的数据。

3.10.1 localStorage 方法

localStorage 方法存储的数据没有时间限制。第二天、第二周或下一年之后，数据依然可用。

下面是创建和访问 localStorage 的示例：

*/,

```
<script type="text/javascript">
localStorage.lastname="Smith";
document.write(localStorage.lastname);
</script>
```

下面的例子对用户访问页面的次数进行计数:

*/,

```
<script type="text/javascript">
  if (localStorage.pagecount)
  {
    localStorage.pagecount=Number(localStorage.pagecount) +1;
  }
  else
  {
    localStorage.pagecount=1;
  }
document.write("Visits "+ localStorage.pagecount + " time(s).");
</script>
```

3.10.2 sessionStorage 方法

sessionStorage 方法针对一个 session 进行数据存储。当用户关闭浏览器窗口后，数据会被删除。

下面创建并访问一个 sessionStorage 的示例:

*/,

```
<script type="text/javascript">
sessionStorage.lastname="Smith";
document.write(sessionStorage.lastname);
</script>
```

下面的例子对用户在当前 session 中访问页面的次数进行计数:

*/,

```
<script type="text/javascript">
  if (sessionStorage.pagecount)
  {
  sessionStorage.pagecount=Number(sessionStorage.pagecount) +1;
  }
  else
```

```
  {
  sessionStorage.pagecount=1;
  }
document.write("Visits "+sessionStorage.pagecount+" time(s) this session.");
</script>
```

3.11 Geolocation

HTML5 Geolocation（地理定位）API 用于定位用户的地理位置[26]。

Internet Explorer 9、Firefox、Chrome、Safari 以及 Opera 支持地理定位，而且对于拥有 GPS 的设备，比如 iPhone，地理定位更加精确。

在 HTML5 应用中，可以使用 getCurrentPosition() 方法来获得用户的位置，下面是一个简单的地理定位实例，可返回用户位置的经度和纬度。

*/,

```
<script>
var x=document.getElementById("demo");
function getLocation()
  {
  if (navigator.geolocation)
    {
    navigator.geolocation.getCurrentPosition(showPosition);
    }
  else{x.innerHTML="Geolocation is not supported by this browser.";}
  }
function showPosition(position)
  {
  x.innerHTML="Latitude: " + position.coords.latitude +
  "<br />Longitude: " + position.coords.longitude;
  }
</script>
```

这里展示的过程包括：

1. 检测是否支持地理定位
2. 如果支持，则运行 getCurrentPosition() 方法。如果不支持，则向用户显示一段消息。
3. 如果 getCurrentPosition() 运行成功，则向参数 showPosition 中规定的函数返回一个 coordinates 对象

[26]鉴于该特性可能侵犯用户的隐私，除非用户同意，否则用户位置信息是不可用的。

4. showPosition() 函数获得并显示经度和纬度

这个例子是一个非常基础的地理定位脚本，不含错误处理。getCurrentPosition() 方法的第二个参数用于处理错误。它规定当获取用户位置失败时运行的函数：

*/,

```
function showError(error)
  {
  switch(error.code)
    {
    case error.PERMISSION_DENIED:
      x.innerHTML="User denied the request for Geolocation."
      break;
    case error.POSITION_UNAVAILABLE:
      x.innerHTML="Location information is unavailable."
      break;
    case error.TIMEOUT:
      x.innerHTML="The request to get user location timed out."
      break;
    case error.UNKNOWN_ERROR:
      x.innerHTML="An unknown error occurred."
      break;
    }
  }
```

其中可能会使用到的错误处理代码的意义分别是：

1. Permission denied - 用户不允许地理定位
2. Position unavailable - 无法获取当前位置
3. Timeout - 操作超时

如需在地图中显示结果，就需要访问可使用经纬度的地图服务，比如谷歌地图：

*/,

```
function showPosition(position)
{
var latlon=position.coords.latitude+","+position.coords.longitude;

var img_url="http://maps.googleapis.com/maps/api/staticmap?center="
+latlon+"&zoom=14&size=400x300&sensor=false";

document.getElementById("mapholder").innerHTML="<img src='"+img_url+"' />";
}
```

在上例中，使用返回的经纬度数据在谷歌地图中显示位置（使用静态图像）。

另外，地理定位对于给定位置的信息同样很有用处，一些应用场景包括：

- 更新本地信息
- 显示用户周围的兴趣点
- 交互式车载导航系统 (GPS)

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。

Table 3.18: HTML 5 <audio> 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

Geolocation 对象的 watchPosition() 方法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）。

下面的例子展示 watchPosition() 方法，这里需要一台精确的 GPS 设备来测试该示例（比如 iPhone）：

*/,

```
<script>
var x=document.getElementById("demo");
function getLocation()
  {
  if (navigator.geolocation)
    {
    navigator.geolocation.watchPosition(showPosition);
    }
  else{x.innerHTML="Geolocation is not supported by this browser.";}
  }
function showPosition(position)
```

```
12   {
13   x.innerHTML="Latitude: " + position.coords.latitude +
14   "<br />Longitude: " + position.coords.longitude;
15   }
16 </script>
```

最后，使用 clearWatch() 方法停止 watchPosition()，结束。

3.12 Web Workers

当在 HTML 页面中执行脚本时，页面的状态是不可响应的，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。开发者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 在后台运行。

1. 检测 Web Worker 支持
   除了 Internet Explorer 之外，所有主流浏览器均支持 web worker，因此在创建 web worker 之前，需要先检测用户的浏览器是否支持它：
   */,

```
1  if(typeof(Worker)!=="undefined")
2  {
3  // Yes! Web worker support!
4  // Some code.....
5  }
6  else
7  {
8  // Sorry! No Web Worker support..
9  }
```

2. 创建 web worker 文件
   然后，可以在外部 JavaScript 中开始创建我们的 web worker。
   在这里，为了实现在后台计数的功能，创建了计数脚本[27]。该脚本存储于”demo_workers.js”文件中：
   */,

```
1  var i=0;
2
3  function timedCount()
```

[27]注释：web worker 通常不用于如此简单的脚本，而是用于更耗费 CPU 资源的任务。

```
{
i=i+1;
postMessage(i);
setTimeout("timedCount()",500);
}

timedCount();
```

以上代码中重要的部分是 postMessage() 方法 - 它用于向 HTML 页面传回一段消息。

3. 创建 Web Worker 对象

   在具备了 web worker 文件之后，现在就可以从 HTML 页面调用它。下面的代码检测是否存在 worker，如果不存在，- 它会创建一个新的 web worker 对象，然后运行”demo_workers.js” 中的代码：

   */,

```
 if(typeof(w)=="undefined")
 {
 w=new Worker("demo_workers.js");
 }
```

   然后我们就可以从 web worker 发生和接收消息了，于是接着向 web worker 添加一个”onmessage” 事件监听器：

   */,

```
w.onmessage=function(event){
 document.getElementById("result").innerHTML=event.data;
};
```

   当 web worker 传递消息时，会执行事件监听器中的代码。event.data 中存有来自 event.data 的数据。

4. 终止 Web Worker

   创建 web worker 对象后，它就会持续监听消息（即使在外部脚本完成之后）直到其被终止为止。

   如需终止 web worker，并释放浏览器/计算机资源，可使用 terminate() 方法：

   */,

```
w.terminate();
```

下面是完整的 Web Worker 示例中 HTML 页面的代码：

*/,

```
<!DOCTYPE html>
```

```
<html>
<body>
<p>Count numbers: <output id="result"></output></p>
<button onclick="startWorker()">Start Worker</button>
<button onclick="stopWorker()">Stop Worker</button>
<br />
<script>
var w;

function startWorker(){
  if(typeof(Worker)!=="undefined"){
    if(typeof(w)=="undefined"){
      w=new Worker("demo_workers.js");
    }
    w.onmessage = function (event){
      document.getElementById("result").innerHTML=event.data;
    };
  }
  else{
    document.getElementById("result").innerHTML="Sorry, your browser
     does not support Web Workers...";
  }
}

function stopWorker(){
  w.terminate();
}
</script>
</body>
</html>
```

由于 web worker 位于外部文件中，它们无法访问下列 JavaScript 对象：

- window 对象
- document 对象
- parent 对象

3.13 Server-sent Event

HTML5 服务器发送事件（server-sent event）允许网页获得来自服务器的更新。

Server-Sent 事件指的是网页自动获取来自服务器的更新（注意是单向消息传递），其

实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。

通过服务器发送事件，更新能够自动到达的例子有：Facebook/Twitter 更新、估价更新、新的博文、赛事结果等。

1. 检测 Server-Sent 事件支持

   所有主流浏览器均支持服务器发送事件，除了 Internet Explorer，因此额外的代码来检测服务器发送事件的浏览器支持情况：

   */,

```
if(typeof(EventSource)!=="undefined") {
// Yes! Server-sent events support!
// Some code.....
}
else {
// Sorry! No server-sent events support..
}
```

2. 接收 Server-Sent 事件通知

   EventSource 对象用于接收服务器发送事件通知：

   */,

```
var source=new EventSource("demo_sse.php");
source.onmessage=function(event) {
    document.getElementById("result").innerHTML+=event.data + "<br />";
};
```

   接收服务器发送事件通知的过程如下

   - 创建一个新的 EventSource 对象，然后规定发送更新的页面的 URL（本例中是”demo_sse.php”）
   - 每接收到一次更新，就会发生 onmessage 事件
   - 当 onmessage 事件发生时，把已接收的数据推入 id 为”result” 的元素中

为了让上面的例子可以运行，还需要能够发送数据更新的服务器（比如 PHP 和 ASP）。

服务器端事件流的语法是非常简单，把”Content-Type” 报头设置为 “text/event-stream” 就可以开始发送事件流了。

下面分别是实现发送事件流的 PHP 代码和 ASP 代码：

PHP 代码 (demo_sse.php)：

```
<?php
 header('Content-Type: text/event-stream');
 header('Cache-Control: no-cache');
```

```
4
5   $time = date('r');
6   echo "data: The server time is: {$time}\n\n";
7   flush();
8 ?>
```

ASP 代码 (VB) (demo_sse.asp):

```
1 <%
2   Response.ContentType="text/event-stream"
3   Response.Expires=-1
4   Response.Write("data: " & now())
5   Response.Flush()
6 %>
```

代码解释：

- 把报头”Content-Type” 设置为”text/event-stream”
- 规定不对页面进行缓存
- 输出发送日期（始终以”data: ” 开头）
- 向网页刷新输出数据

在上面的例子中，我们使用 onmessage 事件来获取消息。不过还可以使用其他事件。

Table 3.19: HTML 5 EventSource 对象

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	当错误发生

3.14 XHTML5

XHTML5 is the XML serialization of HTML5. XML documents must be served with an XML Internet media type (often confused with MIME type) such as application/xhtml+xml or application/xml. XHTML5 requires XML's strict, well-formed syntax. The choice between HTML5 and XHTML5 boils down to the choice of a MIME/content type: the media type one chooses determines what type of document should be used. In XHTML5, the HTML5 doctype html is optional and may simply be omitted. HTML that has been written to conform to both the HTML

and XHTML specifications—and which will therefore produce the same DOM tree whether parsed as HTML or XML—is termed ”polyglot markup”.

3.15 Error handling

HTML5 is designed so that old browsers can safely ignore new HTML5 constructs.[citation needed] In contrast to HTML 4.01, the HTML5 specification gives detailed rules for lexing and parsing, with the intent that different compliant browsers will produce the same result in the case of incorrect syntax. Although HTML5 now defines a consistent behavior for ”tag soup” documents, those documents are not regarded as conforming to the HTML5 standard.

3.16 Popularity

According to a report released on 30 September 2011, 34 of the world's top 100 Web sites were using HTML5 – the adoption led by search engines and social networks. Another report released in July 2013 has shown that 153 of the Fortune 500 U.S. companies already implemented HTML5 on their corporate websites.

3.17 Difference from HTML 4.01 and XHTML 1.x

The following is a cursory list of differences and some specific examples.

- New parsing rules: oriented towards flexible parsing and compatibility; not based on SGML
- Ability to use inline SVG and MathML in text/html
- New elements: article, aside, audio, bdi, canvas, command, data, datalist, details, embed, figcaption, figure, footer, header, keygen, mark, meter, nav, output, progress, rp, rt, ruby, section, source, summary, time, track, video, wbr
- New types of form controls: dates and times, email, url, search, number, range, tel, color
- New attributes: charset (on meta), async (on script)
- Global attributes (that can be applied for every element): id, tabindex, hidden, data-* (custom data attributes)
- Deprecated elements will be dropped altogether: acronym, applet, basefont, big, center, dir, font, frame, frameset, isindex, noframes, strike, tt

dev.w3.org provides the latest Editors Draft of ”HTML5 differences from HTML 4”, which provides a complete outline of additions, removals and changes between HTML5 and HTML 4.

Chapter 4

Digital Rights Manage

Industrial players including the BBC, Google, Microsoft, and Netflix have been lobbying for the inclusion of Encrypted Media Extensions (EME), a form of Digital Rights Management (DRM), into the HTML5 standard. As of the end of 2012 and the beginning of 2013, 27 organisations including the Free Software Foundation have started a campaign against including Digital Rights Management in the HTML5 standard. However in late September 2013, the W3C HTML working group decided that Encrypted Media Extensions, a form of DRM was ”in scope” and will potentially be included in the HTML 5.1 standard

Chapter 5

Reconstruction use HTML5

Chapter 6

HTML5 in mobile devices