第 1 章

HTML5 基础

2014 年 10 月 28 日，W3C 的 HTML 工作组发布了 HTML5 的正式推荐标准，标志着一个全新的 Web 应用时代的开启。HTML5 是构建开放 Web 平台的核心，增加了支持 Web 应用的许多新特性，以及更符合开发者使用习惯的新元素，更关注定义清晰、一致的标准，以确保 Web 应用和内容在不同浏览器中的互操作性。本章主要介绍 HTML5 的基础知识和相关概念。

视 频 讲 解

1.1 HTML5 概述

从 2010 年开始，HTML5 和 CSS3 就一直是网络世界倍受追捧的技术热点。以 HTML5+CSS3 为主的网络时代，使互联网进入了一个崭新的发展阶段。

1.1.1 HTML 历史

HTML 从诞生至今，经历了近 30 年的发展，其中经历的版本及发布日期和说明如表 1.1 所示。

表 1.1 HTML 语言的发展过程

版　本	发 布 日 期	说　明
超文本标记语言（第一版）	1993 年 6 月	作为互联网工程工作小组（IETF）工作草案发布，非标准
HTML2.0	1995 年 11 月	作为 RFC 1866 发布，在 RFC 2854 于 2000 年 6 月发布之后被宣布已经过时
HTML3.2	1996 年 1 月 14 日	W3C 推荐标准
HTML4.0	1997 年 12 月 18 日	W3C 推荐标准
HTML4.01	1999 年 12 月 24 日	微小改进，W3C 推荐标准
ISO HTML	2000 年 5 月 15 日	基于严格的 HTML4.01 语法，是国际标准化组织和国际电工委员会的标准
XHTML1.0	2000 年 1 月 26 日	W3C 推荐标准，修订后于 2002 年 8 月 1 日重新发布
XHTML1.1	2001 年 5 月 31 日	较 XHTML1.0 有微小改进
XHTML2.0 草案	没有发布	2009 年，W3C 停止了 XHTML2.0 工作组的工作
HTML5 草案	2008 年 1 月	HTML5 规范先是以草案发布，经历了漫长的过程
HTML5	2014 年 10 月 28 日	W3C 推荐标准
HTML5.1	2017 年 10 月 3 日	W3C 发布 HTML5 第 1 个更新版本（http://www.w3.org/TR/html51/）
HTML5.2	2017 年 12 月 14 日	W3C 发布 HTML5 第 2 个更新版本（http://www.w3.org/TR/html52/）
HTML5.3	2018 年 3 月 15 日	W3C 发布 HTML5 第 3 个更新版本（http://www.w3.org/TR/html53/）
HTML Living Standard	2019 年 5 月 28 日	WHATWG 的 HTML Living Standard 正式取代 W3C 标准成为官方标准（https://html.spec.whatwg.org/multipage/）

提示：从上面 HTML 发展列表来看，HTML 没有 1.0 版本，这主要是因为当时有很多不同的版本。有些人认为 Tim Berners-Lee 的版本应该算初版 HTML，其版本中还没有 img 元素，也就是说 HTML 刚开始时仅能够显示文本信息。

Note

1.1.2 HTML5 起源

在 20 世纪末期，W3C 开始琢磨着改良 HTML 语言，当时的版本是 HTML4.01。但是在后来的开发和维护过程中，出现了方向性分歧：是开发 XHTML1，再到 XHTML2，最后终极目标是 XML；还是坚持实用主义原则，快速开发出改良的 HTML5 版本？

2004 年 W3C 成员内部的一次研讨会上，当时 Opera 公司的代表伊恩 • 希克森（Ian Hickson）提出了一个扩展和改进 HTML 的建议。他建议新任务组可以跟 XHTML2 并行，但是在已有 HTML 的基础上开展工作，目标是对 HTML 进行扩展。但是 W3C 投票表示反对，因为他们认为 HTML 已经毫无前景，XHTML2 才是未来的方向。

然后，Opera、Apple 等浏览器厂商，以及部分成员忍受不了 W3C 的工作机制和拖沓的行事节奏，决定脱离 W3C，他们成立了 WHATWG（Web Hypertext Applications Technology Working Group，Web 超文本应用技术工作组），这就为 HTML5 将来的命运埋下了伏笔。

WHATWG 决定完全脱离 W3C，在 HTML 的基础上开展工作，向其中添加一些新东西。这个工作组的成员里有浏览器厂商，因此他们可以保证实现各种新奇、实用的点子。结果，大家不断提出一些好点子，并且逐一整合到新版本浏览器中。

WHATWG 的工作效率很高，不久就初见成效。在此期间，W3C 的 XHTML2 没有什么实质性的进展。在 2006 年，蒂姆 • 伯纳斯-李写了一篇博客反思 HTML 的发展历史：“你们知道吗？我们错了。我们错在企图一夜之间就让 Web 跨入 XML 时代，我们的想法太不切实际了，是的，也许我们应该重新组建 HTML 工作组了。”

W3C 在 2007 年组建了 HTML5 工作组。这个工作组面临的第一个问题是“我们是从头开始做起呢，还是在 2004 年成立的那个叫 WHATWG 的工作组既有成果的基础上开始工作呢？”

答案是显而易见的，他们当然希望从已经取得的成果着手，以此为基础展开工作。工作组投了一次票，同意在 WHATWG 工作成果的基础上继续开展工作。

第二个问题就是如何理顺两个工作组之间的关系。W3C 这个工作组的编辑应该由谁担任？是不是还让 WHATWG 的编辑，也就是现在 Google 的伊恩 • 希克森来兼任？于是他们又投了一次票，赞成让伊恩 • 希克森担任 W3C HTML5 规范的编辑，同时兼任 WHATWG 的编辑，更有助于新工作组开展工作。

这就是他们投票的结果，也就是我们今天看到的局面：一种格式，两个版本。WHATWG 网站上有这个规范，而 W3C 网站上同样也有一份。

如果不了解内情，你很可能会产生这样的疑问：“哪个版本才是真正的规范？”当然，这两个版本内容是一样的。实际上，这两个版本将来还会分道扬镳。现在已经有了分道扬镳的迹象。W3C 最终要制定一个具体的规范，这个规范会成为一个工作草案，定格在某个历史时刻。

而 WHATWG 还在不断地迭代。即使目前的 HTML5 也不能完全涵盖 WHATWG 正在从事的工作。最准确的理解就是 WHATWG 正在开发一项简单的 HTML 或 Web 技术，因为这才是他们工作的核心目标。然而，同时存在两个这样的工作组，这两个工作组同时开发一个基本相同的规范，这无论如何也容易让人产生误解，误解就可能造成麻烦。

其实这两个工作组背后各自有各自的流程，因为它们的理念完全不同。在 WHATWG 内部，可以

Note

说是一种独裁的工作机制。伊恩·希克森是编辑。他会听取各方意见，在所有成员各抒己见，充分陈述自己的观点之后，他批准自己认为正确的意见。而 W3C 则截然相反，可以说是一种民主的工作机制。所有成员都可以发表意见，而且每个人都有投票表决的权利。这个流程的关键在于投票表决。从表面上看，WHATWG 的工作机制让人难以接受，W3C 的工作机制听起来让人很舒服，至少体现了人人平等的精神。但在实践中，WHATWG 的工作机制运行得非常好。这主要归功于伊恩·希克森。他在听取各方意见时，始终可以做到丝毫不带个人感情色彩。

从原理上讲，W3C 的工作机制很公平，而实际上却非常容易在某些流程或环节上卡壳，造成工作停滞不前，一件事情要达成决议往往需要花费很长时间。那到底哪种工作机制最好呢？笔者认为，最好的工作机制是将二者结合起来。而事实也是两个规范制定主体在共同制定一份相同的规范，这倒是非常有利于两种工作机制相互取长补短。

两个工作组之所以能够同心同德，主要原因是 HTML5 的设计思想。因为从一开始就确定了设计 HTML5 所要坚持的原则。结果，我们不仅看到了一个规范，也就是 W3C 站点上公布的那份文档，即 HTML5 语言规范，还在 W3C 站点上看到了另一份文档，也就是 HTML5 设计原理。

1.1.3　HTML5 组织

HTML5 是 W3C 与 WHATWG 合作的结晶。HTML5 开发主要由下面 3 个组织负责。

- ☑ WHATWG：由来自 Apple、Mozilla、Google、Opera 等浏览器厂商的专家组成，成立于 2004 年。WHATWG 负责开发 HTML 和 Web 应用 API。
- ☑ W3C：指 World Wide Web Consortium，万维网联盟，负责发布 HTML5 规范。
- ☑ IETF（因特网工程任务组）：负责 Internet 协议开发。HTML5 定义的 WebSocket API 依赖于新的 WebSocket 协议，IETF 工作组负责开发这个协议。

1.1.4　HTML5 规则

为了避免 HTML5 开发过程中出现的各种分歧和偏差，HTML5 开发工作组在共识基础上建立一套行事规则。

- ☑ 新特性应该基于 HTML、CSS、DOM 以及 JavaScript。
- ☑ 减少对外部插件的依赖，如 Flash。
- ☑ 更优秀的错误处理机制。
- ☑ 更多取代脚本的标记。
- ☑ HTML5 应该独立于设备。
- ☑ 开发进程应即时、透明，倾听技术社区的声音，吸纳社区内优秀的 Web 应用。
- ☑ 允许试错，允许纠偏，从实践中来，服务于实践，快速迭代。

1.1.5　HTML5 特性

下面简单介绍 HTML5 的特征和优势，以便增强读者自学 HTML5 的动力和明确目标。

1. 兼容性

考虑到互联网上 HTML 文档已经存在 20 多年了，因此支持所有现存 HTML 文档是非常重要的。HTML5 不是颠覆性的革新，它的核心理念就是要保持与过去技术的兼容和过渡。一旦浏览器不支持 HTML5 的某项功能，针对该功能的备选行为就会悄悄运行。

Note

2. 实用性

HTML5 新增加的元素都是对现有网页和用户习惯进行跟踪、分析和概括而推出的。例如，Google 分析了上百万的页面，从中分析出了 DIV 标签的通用 ID 名称，并且发现其重复量很大，如很多开发人员使用 来标记页眉区域，为了解决实际问题，HTML5 就直接添加一个<header>标签。也就是说，HTML5 新增的很多元素、属性或者功能都是根据现实互联网中已经存在的各种应用进行技术提炼，而不是在实验室中进行理想化地虚构新功能。

3. 效率

HTML5 规范是基于用户优先的原则编写的，其宗旨是用户即上帝，这意味着在遇到无法解决的冲突时，规范会把用户放到第一位，其次是页面制作者，再次是浏览器解析标准，接着是规范制定者（如 W3C、WHATWG），最后才考虑理论的纯粹性。因此，HTML5 的绝大部分功能是实用的，只是在有些情况下还不够完美。例如，下面的几种代码写法在 HTML5 中都能被识别。

```
id="prohtml5"
id=prohtml5
ID="prohtml5"
```

当然，上面几种写法比较混乱，不够严谨，但是从用户开发角度考虑，用户不在乎代码怎么写，根据个人书写习惯反而提高了代码编写效率。

4. 安全性

为保证足够安全，HTML5 引入了一种新的基于来源的安全模型，该模型不仅易用，而且对各种不同的 API 都通用。这个安全模型可以不需要借助于任何所谓聪明、有创意却不安全的 hack 就能跨域进行安全对话。

5. 分离

在清晰分离表现与内容方面，HTML5 迈出了很大的步伐。HTML5 在所有可能的地方都努力进行了分离，包括 HTML 和 CSS。实际上，HTML5 规范已经不支持老版本 HTML 的大部分表现功能了。

6. 简化

HTML5 要的就是简单、避免不必要的复杂性。HTML5 的口号是：简单至上，尽可能简化。因此，HTML5 做了以下改进。

- ☑ 以浏览器原生能力替代复杂的 JavaScript 代码。
- ☑ 简化的 DOCTYPE。
- ☑ 简化的字符集声明。
- ☑ 简单而强大的 HTML5 API。

7. 通用性

通用访问的原则可以分成 3 个概念。

- ☑ 可访问性：出于对残障用户的考虑，HTML5 与 WAI（Web 可访问性倡议）和 ARIA（可访问的富 Internet 应用）做到了紧密结合，WAI-ARIA 中以屏幕阅读器为基础的元素已经被添加到 HTML 中。
- ☑ 媒体中立：如果可能的话，HTML5 的功能在所有不同的设备和平台上应该都能正常运行。
- ☑ 支持所有语种：如新的<ruby>元素支持在东亚页面排版中会用到的 Ruby 注释。

8. 无插件

在传统 Web 应用中，很多功能只能通过插件或者复杂的 hack 来实现，但在 HTML5 中提供了对这些功能的原生支持。插件的方式存在很多问题：

Note

☑ 插件安装可能失败。
☑ 插件可以被禁用或屏蔽，如 Flash 插件。
☑ 插件自身会成为被攻击的对象。
☑ 因为插件边界、剪裁和透明度问题，插件不容易与 HTML 文档的其他部分集成。

以 HTML5 中的 canvas 元素为例，有很多非常底层的事情以前是没办法做到的，如在 HTML4 的页面中就难画出对角线，而有了 canvas 就可以很轻易地实现了。基于 HTML5 的各类 API 的优秀设计，可以轻松地对它们进行组合应用。例如，从 video 元素中抓取的帧可以显示在 canvas 里面，用户单击 canvas 即可播放这帧对应的视频文件。

最后，用万维网联盟创始人 Tim Berners-Lee 评论来小结，“今天，我们想做的事情已经不再是通过浏览器观看视频或收听音频，或者在一部手机上运行浏览器。我们希望通过不同的设备，在任何地方，都能够共享照片、网上购物、阅读新闻以及查找信息。虽然大多数用户对 HTML5 和开放 Web 平台（Open Web Platform，OWP）并不熟悉，但是它们正在不断改进用户体验。”

1.1.6　浏览器支持

HTML5 发展的速度非常快，主流浏览器对于 HTML5 各 API 的支持也不尽统一，用户需要访问 https://www.caniuse.com/网站，在首页输入 API 的名称或关键词，了解各浏览器以及各版本对其支持的详细情况，如图 1.1 所示。在默认主题下，绿色表示完全支持，紫色表示部分支持，红色表示不支持。

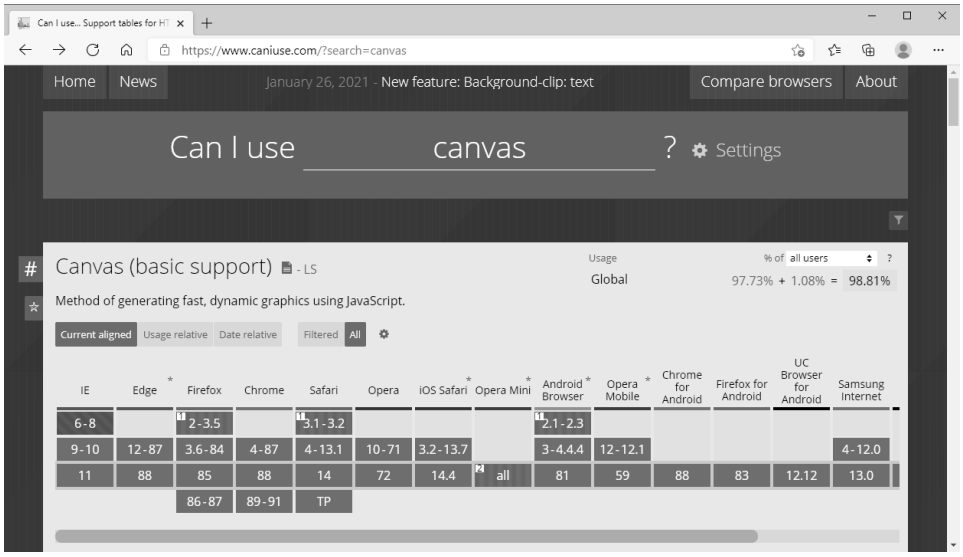

图 1.1　查看各浏览器和各版本对 HTML5 API 的支持情况

如果通过浏览器访问 http://html5test.com/，该网站会直接显示用户当前浏览器和版本对于 HTML5 规范的所有 ALI 支持详情。另外，也可以使用 Modernizr（JavaScript 库）进行特性检测，它提供了非常先进的 HTML5 和 CSS3 检测功能。

1.2　HTML5 设计原则

为了规范 HTML5 开发的兼容性、实用性和互操作性，W3C 发布了 HTML5 设计原则（http://www.w3.org/TR/html-design-principles/），简单说明如下。

1.2.1 避免不必要的复杂性

Note

规范可以写得十分复杂，但浏览器的实现应该非常简单。把复杂的工作留给浏览器后台去处理，用户仅需要输入最简单的字符，甚至不需要输入，才是最佳文档规范。因此，HTML5 首先采用化繁为简的思路进行设计。

【示例 1】在 HTML4.01 中定义文档类型的代码如下。

```
<!DOCTYPE html PUBLIC "-//W3C/DTD HTML4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
```

HTML5 简化如下。

```
<!DOCTYPE html>
```

HTML4.01 和 XHTML 中的 DOCTYPE 过于冗长，但在 HTML5 中只需要简单的<!DOCTYPE html>就可以了。DOCTYPE 是给验证器用的，而非浏览器，浏览器只在做 DOCTYPE 切换时关注这个标签，因此并不需要写得太复杂。

【示例 2】在 HTML4.01 中定义字符编码的代码如下。

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

在 XHTML1.0 中还需要再声明 XML 标签，并在其中指定字符编码。

```
<?xml version="1.0" encoding="UTF-8" ?>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

HTML5 简化如下。

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

关于省略不必要的复杂性，或者说避免不必要的复杂性的例子还有不少。但关键是既能避免不必要的复杂性，还不会妨碍在现有浏览器中使用。

在 HTML5 中，如果使用 link 元素链接到一个样式表，先定义 rel="stylesheet"，然后再定义 type="text/css"，这样就重复了。对浏览器而言，只要设置 rel="stylesheet"就够了，因为它可以识别出要链接的是一个 CSS 样式表，不必要再指定 type 属性。

对 Web 开发而言，大家都使用 JavaScript 脚本语言，也是默认的通用语言，用户可以为 script 元素定义 type="text/javascript"属性，也可以什么都不写，浏览器自然会假设在使用 JavaScript。

1.2.2 支持已有内容

XHTML2.0 最大的问题就是不支持已经存在的内容，这违反了 Postel 法则（即对自己发送的东西要严格，对接收的东西则要宽容）。现实情况中，开发者可以写出各种风格的 HTML，浏览器遇到这些代码时，在内部所构建出的结构应该是一样的，呈现的效果也应该是一样的。

【示例】下面示例展示了编写同样内容的 4 种不同写法，4 种写法唯一的不同点就是语法。

```
<!--写法 1-->
<img src="foo" alt="bar" />
<p class="foo">Hello world</p>
<!--写法 2-->
<img src="foo" alt="bar">
<p class="foo">Hello world
<!--写法 3-->
<IMG SRC="foo" ALT="bar">
<P CLASS="foo">Hello world</P>
<!--写法 4-->
```

```
<img src=foo alt=bar>
<p class=foo>Hello world</p>
```

从浏览器解析的角度分析，这些写法实际上都是一样的。HTML5 必须支持已经存在的约定，适应不同的用户习惯，而不是用户适应浏览器的严格解析标准。

Note

1.2.3　解决实际问题

规范应该去解决现实中实际遇到的问题，而不该考虑那些复杂的理论问题。

【示例】既然有在<a>中嵌套多个段落标签的需要，那就让规范支持它。

如果块内容包含一个标题，一个段落。按 HTML4 规范，必须至少使用两个链接。例如：

```
<h2><a href="#">标题文本</a></h2>
<p><a href="#">段落文本</a></p>
```

在 HTML5 中，只需要把所有内容都包裹在一个链接中就行了。例如：

```
<a href="#">
    <h2>标题文本</h2>
    <p>段落文本</p>
</a>
```

其实这种写法早已存在，当然以前这样写是不合乎规范的。所以说，HTML5 解决现实的问题，其本质还是纠正因循守旧的规范标准，现在把标准改了，允许用户这样写了。

1.2.4　用户怎么使用就怎么设计规范

当一个实践已经被广泛接受时，就应该考虑将它吸纳进来，而不是禁止它或搞一个新的实践出来。例如，HTML5 新增了 nav、section、article、aside 等标签，它们引入了新的文档模型，即文档中的文档。在 section 中，还可以嵌套 h1～h6 的标签，这样就有了无限的标题层级，这也是很早之前 Tim Berners Lee 所设想的。

【示例】下面几行代码相信大家都不会陌生，这些都是频繁被使用过的 ID 名称。

```
<div id="header">...</div>
<div id="navigation">...</div>
<div id="main">...</div>
<div id="aside">...</div>
<div id="footer">...</div>
```

在 HTML5 中，可以用新的元素代替使用。

```
<header>...</header>
<nav>...</nav>
<div id="main">...</div>
<aside>...</aside>
<footer>...</footer>
```

实际上，这并不是 HTML5 工作组发明的，也不是 W3C 开会研究出来的，而是谷歌根据大数据分析用户习惯总结出来的。

1.2.5　优雅地降级

渐进增强的另一面就是优雅地回退。最典型的例子就是使用 type 属性增强表单。

Note

【示例 1】下面代码列出了可以为 type 属性指定新值，如 number、search、range 等。

```
<input type="number" />
<input type="search" />
<input type="range" />
<input type="email" />
<input type="date" />
<input type="url" />
```

最关键的问题在于：浏览器看到这些新 type 值时会如何处理。老版本浏览器是无法理解这些新 type 值的。但是当它们看到自己不理解的 type 值时，会将 type 的值解释为 text。

【示例 2】对于新的 video 元素，它设计得很简单、实用。针对不支持 video 元素的浏览器可以这样写。

```
<video src="movie.mp4">
    <!--回退内容-->
</video>
```

这样 HTML5 视频与 Flash 视频就可以协同起来，用户不用纠结如何选择。

```
<video src="movie.mp4">
    <object data="movie.swf">
        <!--回退内容-->
    </object>
</video>
```

如果愿意的话，还可以使用 source 元素，而非 src 属性来指定不同的视频格式。

```
<video>
    <source src="movie.mp4">
    <source src="movie.ogv">
    <object data="movie.swf">
        <a href="movie.mp4">download</a>
    </object>
</video>
```

上面代码包含了 4 个不同的层次。

☑ 如果浏览器支持 video 元素，也支持 H264，那么用第一个视频。

☑ 如果浏览器支持 video 元素，支持 Ogg，那么用第二个视频。

☑ 如果浏览器不支持 video 元素，那么就要试试 Flash 视频。

☑ 如果浏览器不支持 video 元素，也不支持 Flash 视频，还可以给出下载链接。

总之，无论是 HTML5，还是 Flash，一个也不能少。如果只使用 video 元素提供视频，难免会遇到问题。而如果只提供 Flash 影片，性质是一样的，所以还是应该两者兼顾。

1.2.6 支持的优先级

用户与开发者的重要性要远远高于规范和理论。在考虑优先级时，应该按照下面顺序设计。

```
用户 > 编写 HTML 的开发者 > 浏览器厂商 > 规范制定者 > 理论
```

这个设计原则本质上是一种解决冲突的机制。例如，当面临一个要解决的问题时，如果 W3C 给出了一种解决方案，而 WHATWG 给出了另一种解决方案。一旦遇到冲突，最终用户优先，其次是开发者，再是实现者，然后规范制定者，最后才是理论上的完美。

根据最终用户优先的原理，开发人员在链条中的位置高于实现者，假如我们发现了规范中的某些地方有问题，就不支持实现这个特性，那么就等于把相应的特性给否定了，规范里就得删除，因为用户有更高的权重。本质上用户拥有了更大的发言权，开发人员也拥有更多的主动性。

1.3　HTML5 语法特性

Note

HTML5 以 HTML4 为基础，对 HTML4 进行了全面升级改造。与 HTML4 相比，HTML5 在语法上有很大的变化，具体说明如下。

1.3.1　文档和标记

1. 内容类型

HTML5 的文件扩展名和内容类型保持不变。例如，扩展名仍然为“.html”或“.htm”，内容类型（ContentType）仍然为“text/html”。

2. 文档类型

在 HTML4 中，文档类型的声明方法如下。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

在 HTML5 中，文档类型的声明方法如下。

```
<!DOCTYPE html>
```

当使用工具时，也可以在 DOCTYPE 声明中加入 SYSTEM 识别符，声明方法如下。

```
<!DOCTYPE HTML SYSTEM "about:legacy-compat">
```

在 HTML5 中，DOCTYPE 声明方式是不区分大小写的，引号也不区分是单引号还是双引号。

注意：使用 HTML5 的 DOCTYPE 会触发浏览器以标准模式显示页面。众所周知，网页都有多种显示模式，如怪异模式（Quirks）、标准模式（Standards）。浏览器根据 DOCTYPE 来识别该使用哪种解析模式。

3. 字符编码

在 HTML4 中，使用 meta 元素定义文档的字符编码，如下所示。

```
<meta http-equiv="Content-Type" content="text/html;charset=UTF-8">
```

在 HTML5 中，继续沿用 meta 元素定义文档的字符编码，但是简化了 charset 属性的写法，如下所示。

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

对于 HTML5 来说，上述两种方法都有效，用户可以继续使用前一种方式，即通过 content 元素的属性来指定。但是不能同时混用两种方式。

注意：在传统网页中，可能会存在下面的标记。但 HTML5 中这种字符编码方式是错误的。

```
<meta charset="UTF-8" http-equiv="Content-Type" content="text/html;charset=UTF-8">
```

从 HTML5 开始，对于文件的字符编码推荐使用 UTF-8。

1.3.2　宽松的约定

HTML5 语法是为了保证与之前的 HTML4 语法达到最大限度的兼容而设计的。