第 1 章

JavaScript 简介

本章内容

- JavaScript 历史回顾
- JavaScript 是什么
- JavaScript 与 ECMAScript 的关系
- JavaScript 的不同版本

JavaScript 诞生于 1995 年。当时，它的主要目的是处理以前由服务器端语言（如 Perl）负责的一些输入验证操作。在 JavaScript 问世之前，必须把表单数据发送到服务器端才能确定用户是否没有填写某个必填域，是否输入了无效的值。Netscape Navigator 希望通过 JavaScript 来解决这个问题。在人们普遍使用电话拨号上网的年代，能够在客户端完成一些基本的验证任务绝对是令人兴奋的。毕竟，拨号上网的速度之慢，导致了与服务器的每一次数据交换事实上都成了对人们耐心的一次考验。

自此以后，JavaScript 逐渐成为市面上常见浏览器必备的一项特色功能。如今，JavaScript 的用途早已不再局限于简单的数据验证，而是具备了与浏览器窗口及其内容等几乎所有方面交互的能力。今天的 JavaScript 已经成为一门功能全面的编程语言，能够处理复杂的计算和交互，拥有了闭包、匿名（lamda，拉姆达）函数，甚至元编程等特性。作为 Web 的一个重要组成部分，JavaScript 的重要性是不言而喻的，就连手机浏览器，甚至那些专为残障人士设计的浏览器等非常规浏览器都支持它。当然，微软的例子更为典型。虽然有自己的客户端脚本语言 VBScript，但微软仍然在 Internet Explorer 的早期版本中加入了自己的 JavaScript 实现[①]。

JavaScript 从一个简单的输入验证器发展成为一门强大的编程语言，完全出乎人们的意料。应该说，它既是一门非常简单的语言，又是一门非常复杂的语言。说它简单，是因为学会使用它只需片刻功夫；而说它复杂，是因为要真正掌握它则需要数年时间。要想全面理解和掌握 JavaScript，关键在于弄清楚它的本质、历史和局限性。

1.1 JavaScript 简史

在 Web 日益流行的同时，人们对客户端脚本语言的需求也越来越强烈。那个时候，绝大多数因特网用户都使用速度仅为 28.8kbit/s 的“猫”（调制解调器）上网，但网页的大小和复杂性却不断增加。为完成简单的表单验证而频繁地与服务器交换数据只会加重用户的负担。想象一下：用户填写完一个表单，单击“提交”按钮，然后等待 30 秒钟，最终服务器返回消息说有一个必填字段没有

① 对 IE 而言，当我们提到 JavaScript 时，实际上就是指 IE 对 JavaScript（ECMAScript）的实现——JScript。最早的 JScript 基于 Netscape JavaScript 1.0 开发，于 1996 年 8 月随同 Internet Explorer 3.0 发布。

填好……当时走在技术革新最前沿的 Netscape 公司，决定着手开发一种客户端语言，用来处理这种简单的验证。

当时就职于 Netscape 公司的布兰登·艾奇（Brendan Eich），开始着手为计划于 1995 年 2 月发布的 Netscape Navigator 2 开发一种名为 LiveScript 的脚本语言——该语言将同时在浏览器和服务器中使用（它在服务器上的名字叫 LiveWire）。为了赶在发布日期前完成 LiveScript 的开发，Netscape 与 Sun 公司建立了一个开发联盟。在 Netscape Navigator 2 正式发布前夕，Netscape 为了搭上媒体热炒 Java 的顺风车，临时把 LiveScript 改名为 JavaScript。

由于 JavaScript 1.0 获得了巨大成功，Netscape 随即在 Netscape Navigator 3 中又发布了 JavaScript 1.1。Web 虽然羽翼未丰，但用户关注度却屡创新高。在这样的背景下，Netscape 把自己定位为市场领袖型公司。与此同时，微软决定向与 Navigator 竞争的自家产品 Internet Explorer 浏览器投入更多资源。Netscape Navigator 3 发布后不久，微软就在其 Internet Explorer 3 中加入了名为 JScript 的 JavaScript 实现（命名为 JScript 是为了避开与 Netscape 有关的授权问题）。以现在的眼光来看，微软 1996 年 8 月为进入 Web 浏览器领域而实施的这个重大举措，是导致 Netscape 日后蒙羞的一个标志性事件。然而，这个重大举措同时也标志着 JavaScript 作为一门语言，其开发向前迈进了一大步。

微软推出其 JavaScript 实现意味着有了 3 个不同的 JavaScript 版本：Netscape Navigator 中的 JavaScript、Internet Explorer 中的 Jscript 和 ScriptEase 中的 CEnvi。与 C 及其他编程语言不同，当时还没有标准规定 JavaScript 的语法和特性，3 个不同版本并存的局面已经完全暴露了这个问题。随着业界担心的日益加剧，JavaScript 的标准化问题被提上了议事日程。

1997 年，以 JavaScript 1.1 为蓝本的建议被提交给了欧洲计算机制造商协会（ECMA，European Computer Manufacturers Association）。该协会指定 39 号技术委员会（TC39，Technical Committee #39）负责“标准化一种通用、跨平台、供应商中立的脚本语言的语法和语义”（http://www.ecma international.org/memento/TC39.htm）。TC39 由来自 Netscape、Sun、微软、Borland 及其他关注脚本语言发展的公司的程序员组成，他们经过数月的努力完成了 ECMA-262——定义一种名为 ECMAScript（发音为“ek-ma-script”）的新脚本语言的标准。

第二年，ISO/IEC（International Organization for Standardization and International Electrotechnical Commission，国标标准化组织和国际电工委员会）也采用了 ECMAScript 作为标准（即 ISO/IEC-16262）。自此以后，浏览器开发商就开始致力于将 ECMAScript 作为各自 JavaScript 实现的基础，也在不同程度上取得了成功。

1.2 JavaScript 实现

虽然 JavaScript 和 ECMAScript 通常都被人们用来表达相同的含义，但 JavaScript 的含义却比 ECMA-262 中规定的要多得多。没错，一个完整的 JavaScript 实现应该由下列三个不同的部分组成（见图 1-1）。

- 核心（ECMAScript）
- 文档对象模型（DOM）
- 浏览器对象模型（BOM）

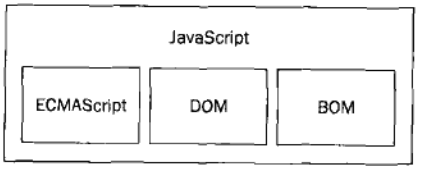

图 1-1

1.2.1 ECMAScript

由 ECMA-262 定义的 ECMAScript 与 Web 浏览器没有依赖关系。实际上，这门语言本身并不包含输入和输出定义。ECMA-262 定义的只是这门语言的基础，而在此基础之上可以构建更完善的脚本语言。我们常见的 Web 浏览器只是 ECMAScript 实现可能的宿主环境之一。宿主环境不仅提供基本的 ECMAScript 实现，同时也会提供该语言的扩展，以便语言与环境之间对接交互。而这些扩展——如 DOM，则利用 ECMAScript 的核心类型和语法提供更多更具体的功能，以便实现针对环境的操作。其他宿主环境包括 Node（一种服务端 JavaScript 平台）和 Adobe Flash。

既然 ECMA-262 标准没有参照 Web 浏览器，那它都规定了些什么内容呢？大致说来，它规定了这门语言的下列组成部分：

- 语法
- 类型
- 语句
- 关键字
- 保留字
- 操作符
- 对象

ECMAScript 就是对实现该标准规定的各个方面内容的语言的描述。JavaScript 实现了 ECMAScript，Adobe ActionScript 同样也实现了 ECMAScript。

1. ECMAScript 的版本

ECMAScript 的不同版本又称为版次，以第 *x* 版表示（意即描述特定实现的 ECMA-262 规范的第 *x* 个版本）。ECMA-262 的最近一版是第 5 版，发布于 2009 年。而 ECMA-262 的第 1 版本质上与 Netscape 的 JavaScript 1.1 相同——只不过删除了所有针对浏览器的代码并作了一些较小的改动；ECMA-262 要求支持 Unicode 标准（从而支持多语言开发），而且对象也变成了平台无关的（Netscape JavaScript 1.1 的对象在不同平台中的实现不一样，例如 `Date` 对象）。这也是 JavaScript 1.1 和 1.2 与 ECMA-262 第 1 版不一致的主要原因。

ECMA-262 第 2 版主要是编辑加工的结果。这一版中内容的更新是为了与 ISO/IEC-16262 保持严格一致，没有作任何新增、修改或删节处理。因此，一般不使用第 2 版来衡量 ECMAScript 实现的兼容性。

ECMA-262 第 3 版才是对该标准第一次真正的修改。修改的内容涉及字符串处理、错误定义和数值输出。这一版还新增了对正则表达式、新控制语句、`try-catch` 异常处理的支持，并围绕标准的国际化做出了一些小的修改。从各方面综合来看，第 3 版标志着 ECMAScript 成为了一门真正的编程语言。

ECMA-262 第 4 版对这门语言进行了一次全面的检核修订。由于 JavaScript 在 Web 上日益流行，开发人员纷纷建议修订 ECMAScript，以使其能够满足不断增长的 Web 开发需求。作为回应，ECMA TC39 重新召集相关人员共同谋划这门语言的未来。结果，出台后的标准几乎在第 3 版基础上完全定义了一门新语言。第 4 版不仅包含了强类型变量、新语句和新数据结构、真正的类和经典继承，还定义了与数据交互的新方式。

与此同时，TC39 下属的一个小组也提出了一个名为 ECMAScript 3.1 的替代性建议，该建议只对这门语言进行了较少的改进。这个小组认为第 4 版给这门语言带来的跨越太大了。因此，该小组建议对这

门语言进行小幅修订，能够在现有 JavaScript 引擎基础上实现。最终，ES3.1 附属委员会获得的支持超过了 TC39，ECMAS-262 第 4 版在正式发布前被放弃。

ECMAScript 3.1 成为 ECMA-262 第 5 版，并于 2009 年 12 月 3 日正式发布。第 5 版力求澄清第 3 版中已知的歧义并增添了新的功能。新功能包括原生 JSON 对象（用于解析和序列化 JSON 数据）、继承的方法和高级属性定义，另外还包含一种严格模式，对 ECMAScript 引擎解释和执行代码进行了补充说明。

2. 什么是 ECMAScript 兼容

ECMA-262 给出了 ECMAScript 兼容的定义。要想成为 ECMAScript 的实现，则该实现必须做到：

- 支持 ECMA-262 描述的所有“类型、值、对象、属性、函数以及程序句法和语义”（ECMA-262 第 1 页）；
- 支持 Unicode 字符标准。

此外，兼容的实现还可以进行下列扩展。

- 添加 ECMA-262 没有描述的“更多类型、值、对象、属性和函数”。ECMA-262 所说的这些新增特性，主要是指该标准中没有规定的新对象和对象的新属性。
- 支持 ECMA-262 没有定义的“程序和正则表达式语法”。（也就是说，可以修改和扩展内置的正则表达式语法。）

上述要求为兼容实现的开发人员基于 ECMAScript 开发一门新语言提供了广阔的空间和极大的灵活性，这也从另一个侧面说明了 ECMAScript 受开发人员欢迎的原因。

3. Web 浏览器对 ECMAScript 的支持

1996 年，Netscape Navigator 3 捆绑发布了 JavaScript 1.1。而相同的 JavaScript 1.1 设计规范随后作为对新标准（ECMA-262）的建议被提交给 Ecma。伴随着 JavaScript 的迅速走红，Netscape 豪情满怀地着手开发 JavaScript 1.2。然而，问题是 Ecma 当时还没有接受 Netscape 的建议。

Netscape Navigator 3 发布后不久，微软也推出了 Internet Explorer 3。微软在 IE 的这一版中捆绑了 JScript 1.0，很多人都认为 JScript 1.0 与 JavaScript 1.1 应该是一样的。但是，由于没有文档依据，加之不适当的特性模仿，JScript 1.0 还是很难与 JavaScript 1.1 相提并论。

1997 年，内置 JavaScript 1.2 的 Netscape Navigator 4 发布；而到这一年年底，ECMA-262 第 1 版也被接受并实现了标准化。结果，虽然 ECMAScript 被认为是基于 JavaScript 1.1 制定的，但 JavaScript 1.2 与 ECMAScript 的第 1 版并不兼容。

JScript 的升级版是 Internet Explorer 4 中内置的 JScript 3.0（随同微软 IIS 3.0 发布的 JScript 2.0 从来也没有移植到浏览器中）。微软通过媒体大肆宣传 JScript 3.0 是世界上第一个 ECMA 兼容的脚本语言，但当时的 ECMA-262 尚未定稿。于是，JScript 3.0 与 JavaScript 1.2 都遭遇了相同的尴尬局面——谁都没有按照最终的 ECMAScript 标准来实现。

Netscape 决定更新其 JavaScript 实现，即在 Netscape Navigator 4.06 中发布 JavaScript 1.3，从而做到了与 ECMA-262 的第一个版本完全兼容。在 JavaScript 1.3 中，Netscape 增加了对 Unicode 标准的支持，并在保留 JavaScript 1.2 新增特性的同时实现了所有对象的平台中立化。

在 Netscape 以 Mozilla 项目的名义开放其源代码时，预期 JavaScript 1.4 将随同 Netscape Navigator 5 一道发布。然而，一个激进的决定，彻底重新设计 Netscape 代码，打乱了原有计划。后来，JavaScript 1.4 只发布了针对 Netscape Enterprise Server 的服务器版，而没有内置于 Web 浏览器中。

到了 2008 年，五大主流 Web 浏览器（IE、Firefox、Safari、Chrome 和 Opera）全部做到了与 ECMA-262

兼容。IE8 是第一个着手实现 ECMA-262 第 5 版的浏览器，并在 IE9 中提供了完整的支持。Firefox 4 也紧随其后做到兼容。下表列出了 ECMAScript 受主流 Web 浏览器支持的情况。

浏 览 器	ECMAScript兼容性	浏 览 器	ECMAScript兼容性
Netscape Navigator 2	—	Opera 6～7.1	第2版
Netscape Navigator 3	—	Opera 7.2+	第3版
Netscape Navigator 4～4.05	—	Safari 1～2.0.*x*	第3版*
Netscape Navigator 4.06～4.79	第1版	Safari 3.*x*	第3版
Netscape 6+（Mozilla 0.6.0+）	第3版	Safari 4.*x*～5.*x*	第5版*
IE3	—	Chrome 1+	第3版
IE4	—	Firefox 1～2	第3版
IE5	第1版	Firefox 3.0.*x*	第3版
IE5.5～IE7	第3版	Firefox 3.5～3.6	第5版*
IE8	第5版*	Firefox 4.0+	第5版
IE9+	第5版		

* 不完全兼容的实现

1.2.2 文档对象模型（DOM）

文档对象模型（DOM，Document Object Model）是针对 XML 但经过扩展用于 HTML 的应用程序编程接口（API，Application Programming Interface）。DOM 把整个页面映射为一个多层节点结构。HTML 或 XML 页面中的每个组成部分都是某种类型的节点，这些节点又包含着不同类型的数据。看下面这个 HTML 页面：

```
<html>
    <head>
        <title>Sample Page</title>
    </head>
    <body>
        <p>Hello World!</p>
    </body>
</html>
```

在 DOM 中，这个页面可以通过见图 1-2 所示的分层节点图表示。

通过 DOM 创建的这个表示文档的树形图，开发人员获得了控制页面内容和结构的主动权。借助 DOM 提供的 API，开发人员可以轻松自如地删除、添加、替换或修改任何节点。

1. 为什么要使用 DOM

在 Internet Explorer 4 和 Netscape Navigator 4 分别支持的不同形式的 DHTML（Dynamic HTML）基础上，开发人员首次无需重新加载网页，就可以修改其外观和内容了。然而，DHTML 在给 Web 技术发展带来巨大进步的同时，也带来了巨大的问题。由于 Netscape 和微软在开发 DHTML 方面各持己见，过去那个只编写一个 HTML 页面就能够在任何浏览器中运行的时代结束了。

对开发人员而言，如果想继续保持 Web 跨平台的天性，就必须额外多做一些工作。而人们真正担心的是，如果不对 Netscape 和微软加以控制，Web 开发领域就会出现技术上两强割据，浏览器互不兼

容的局面。此时，负责制定 Web 通信标准的 W3C（World Wide Web Consortium，万维网联盟）开始着手规划 DOM。

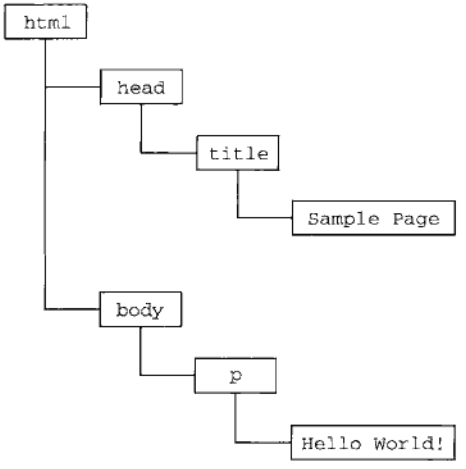

图　1-2

2. DOM 级别

DOM1 级（DOM Level 1）于 1998 年 10 月成为 W3C 的推荐标准。DOM1 级由两个模块组成：DOM 核心（DOM Core）和 DOM HTML。其中，DOM 核心规定的是如何映射基于 XML 的文档结构，以便简化对文档中任意部分的访问和操作。DOM HTML 模块则在 DOM 核心的基础上加以扩展，添加了针对 HTML 的对象和方法。

请读者注意，DOM 并不只是针对 JavaScript 的，很多别的语言也都实现了 DOM。不过，在 Web 浏览器中，基于 ECMAScript 实现的 DOM 的确已经成为 JavaScript 这门语言的一个重要组成部分。

如果说 DOM1 级的目标主要是映射文档的结构，那么 DOM2 级的目标就要宽泛多了。DOM2 级在原来 DOM 的基础上又扩充了（DHTML 一直都支持的）鼠标和用户界面事件、范围、遍历（迭代 DOM 文档的方法）等细分模块，而且通过对象接口增加了对 CSS（Cascading Style Sheets，层叠样式表）的支持。DOM1 级中的 DOM 核心模块也经过扩展开始支持 XML 命名空间。

DOM2 级引入了下列新模块，也给出了众多新类型和新接口的定义。

- DOM 视图（DOM Views）：定义了跟踪不同文档（例如，应用 CSS 之前和之后的文档）视图的接口；
- DOM 事件（DOM Events）：定义了事件和事件处理的接口；
- DOM 样式（DOM Style）：定义了基于 CSS 为元素应用样式的接口；
- DOM 遍历和范围（DOM Traversal and Range）：定义了遍历和操作文档树的接口。

DOM3 级则进一步扩展了 DOM，引入了以统一方式加载和保存文档的方法——在 DOM 加载和保存（DOM Load and Save）模块中定义；新增了验证文档的方法——在 DOM 验证（DOM Validation）模

块中定义。DOM3 级也对 DOM 核心进行了扩展，开始支持 XML 1.0 规范，涉及 XML Infoset、XPath 和 XML Base。

在阅读 DOM 标准的时候，读者可能会看到 DOM0 级（DOM Level 0）的字眼。实际上，DOM0 级标准是不存在的；所谓 DOM0 级只是 DOM 历史坐标中的一个参照点而已。具体说来，DOM0 级指的是 Internet Explorer 4.0 和 Netscape Navigator 4.0 最初支持的 DHTML。

3. 其他 DOM 标准

除了 DOM 核心和 DOM HTML 接口之外，另外几种语言还发布了只针对自己的 DOM 标准。下面列出的语言都是基于 XML 的，每种语言的 DOM 标准都添加了与特定语言相关的新方法和新接口：

- SVG（Scalable Vector Graphic，可伸缩矢量图）1.0；
- MathML（Mathematical Markup Language，数学标记语言）1.0；
- SMIL（Synchronized Multimedia Integration Language，同步多媒体集成语言）。

还有一些语言也开发了自己的 DOM 实现，例如 Mozilla 的 XUL（XML User Interface Language，XML 用户界面语言）。但是，只有上面列出的几种语言是 W3C 的推荐标准。

4. Web 浏览器对 DOM 的支持

在 DOM 标准出现了一段时间之后，Web 浏览器才开始实现它。微软在 IE5 中首次尝试实现 DOM，但直到 IE5.5 才算是真正支持 DOM1 级。在随后的 IE6 和 IE7 中，微软都没有引入新的 DOM 功能，而到了 IE8 才对以前 DOM 实现中的 bug 进行了修复。

Netscape 直到 Netscape 6（Mozilla 0.6.0）才开始支持 DOM。在 Netscape 7 之后，Mozilla 把开发重心转向了 Firefox 浏览器。Firefox 3 完全支持 DOM1 级，几乎完全支持 DOM2 级，甚至还支持 DOM3 级的一部分。（Mozilla 开发团队的目标是构建与标准 100%兼容的浏览器，而他们的努力也得到了回报。）

目前，支持 DOM 已经成为浏览器开发商的首要目标，主流浏览器每次发布新版本都会改进对 DOM 的支持。下表列出了主流浏览器对 DOM 标准的支持情况。

浏 览 器	DOM兼容性
Netscape Navigator 1. ~ 4.*x*	—
Netscape 6+（Mozilla 0.6.0+）	1级、2级（几乎全部）、3级（部分）
IE2～IE4.*x*	—
IE5	1级（最小限度）
IE5.5～IE8	1级（几乎全部）
IE9+	1级、2级、3级
Opera 1～6	—
Opera 7～8.*x*	1级（几乎全部）、2级（部分）
Opera 9～9.9	1级、2级（几乎全部）、3级（部分）
Opera 10+	1级、2级、3级（部分）
Safari 1.0.*x*	1级
Safari 2+	1级、2级（部分）
Chrome 1+	1级、2级（部分）
Firefox 1+	1级、2级（几乎全部）、3级（部分）

1.2.3 浏览器对象模型（BOM）

Internet Explorer 3 和 Netscape Navigator 3 有一个共同的特色，那就是支持可以访问和操作浏览器窗口的浏览器对象模型（BOM，Browser Object Model）。开发人员使用 BOM 可以控制浏览器显示的页面以外的部分。而 BOM 真正与众不同的地方（也是经常会导致问题的地方），还是它作为 JavaScript 实现的一部分但却没有相关的标准。这个问题在 HTML5 中得到了解决，HTML5 致力于把很多 BOM 功能写入正式规范。HTML5 发布后，很多关于 BOM 的困惑烟消云散。

从根本上讲，BOM 只处理浏览器窗口和框架；但人们习惯上也把所有针对浏览器的 JavaScript 扩展算作 BOM 的一部分。下面就是一些这样的扩展：

- 弹出新浏览器窗口的功能；
- 移动、缩放和关闭浏览器窗口的功能；
- 提供浏览器详细信息的 `navigator` 对象；
- 提供浏览器所加载页面的详细信息的 `location` 对象；
- 提供用户显示器分辨率详细信息的 `screen` 对象；
- 对 cookies 的支持；
- 像 `XMLHttpRequest` 和 IE 的 `ActiveXObject` 这样的自定义对象。

由于没有 BOM 标准可以遵循，因此每个浏览器都有自己的实现。虽然也存在一些事实标准，例如要有 `window` 对象和 `navigator` 对象等，但每个浏览器都会为这两个对象乃至其他对象定义自己的属性和方法。现在有了 HTML5，BOM 实现的细节有望朝着兼容性越来越高的方向发展。第 8 章将深入讨论 BOM。

1.3 JavaScript 版本

作为 Netscape “继承人”的 Mozilla 公司，是目前唯一还在沿用最初的 JavaScript 版本编号序列的浏览器开发商。在 Netscape 将源代码提交给开源的 Mozilla 项目的时候，JavaScript 在浏览器中的最后一个版本号是 1.3。（如前所述，1.4 版是只针对服务器的实现。）后来，随着 Mozilla 基金会继续开发 JavaScript，添加新的特性、关键字和语法，JavaScript 的版本号继续递增。下表列出了 Netscape/Mozilla 浏览器中 JavaScript 版本号的递增过程：

浏 览 器	JavaScript版本	浏 览 器	JavaScript版本
Netscape Navigator 2	1.0	Firefox 1.5	1.6
Netscape Navigator 3	1.1	Firefox 2	1.7
Netscape Navigator 4	1.2	Firefox 3	1.8
Netscape Navigator 4.06	1.3	Firefox 3.5	1.8.1
Netscape 6+（Mozilla 0.6.0+）	1.5	Firefox 3.6	1.8.2
Firefox 1	1.5		

实际上，上表中的编号方案源自 Firefox 4 将内置 JavaScript 2.0 这一共识。因此，2.0 版之前每个递增的版本号，表示的是相应实现与 JavaScript 2.0 开发目标还有多大的距离。虽然原计划是这样，但 JavaScript 的这种发展速度让这个计划成为不再可行。目前，JavaScript 2.0 还没有目标实现。

请注意，只有 Netscape/Mozilla 浏览器才遵循这种编号模式。例如，IE 的 JScript 就采用了另一种版本命名方案。换句话说，JScript 的版本号与上表中 JavaScript 的版本号之间不存在任何对应关系。而且，大多数浏览器在提及对 JavaScript 的支持情况时，一般都以 ECMAScript 兼容性和对 DOM 的支持情况为准。

1.4 小结

JavaScript 是一种专为与网页交互而设计的脚本语言，由下列三个不同的部分组成：

- ECMAScript，由 ECMA-262 定义，提供核心语言功能；
- 文档对象模型（DOM），提供访问和操作网页内容的方法和接口；
- 浏览器对象模型（BOM），提供与浏览器交互的方法和接口。

JavaScript 的这三个组成部分，在当前五个主要浏览器（IE、Firefox、Chrome、Safari 和 Opera）中都得到了不同程度的支持。其中，所有浏览器对 ECMAScript 第 3 版的支持大体上都还不错，而对 ECMAScript 5 的支持程度越来越高，但对 DOM 的支持则彼此相差比较多。对 HTML5 已经正式纳入标准的 BOM 来说，尽管各浏览器都实现了某些众所周知的共同特性，但其他特性还是会因浏览器而异。