第 1 章　JavaScript 基础

JavaScript 是一种基于对象的脚本编程语言，从本章中你可以了解到 JavaScript 是如何以及为何出现的，从它的开始到如今涵盖各种特性的实现，还介绍了 JavaScript 的具体应用及 JavaScript 脚本语言的开发环境。

- JavaScript 和客户端脚本编程的起源
- 在 Web 页面中使用 JavaScript 的方法
- 编写和调试 JavaScript 的几种常用工具

1.1　JavaScript 的历史与现状

1.1.1　JavaScript 的发展

当 Internet 普及越来越广时，对于开发客户端脚本的需求也逐渐增大。此时，大部分 Internet 用户仅仅通过 28.8kbt/s 的 Modem 来连接到网络，尽管这时网页已经不断地变得更大和更复杂。加剧用户痛苦的是，仅仅为了简单的表单有效性验证，就要与服务器进行多次的往返交互。设想一下，用户填写完一个表单，单击提交按钮，等待 30 秒后，看到的却是一条告诉你忘记填写一个必要的字段的信息。那时正处于技术革新最前沿的 Netscape，开始认真考虑开发一种客户端语言来处理简单的问题。

当时为 Netscape 工作的 Brendan Erich，开始着手为即将在 1995 年发行的 Netscape Navigator 2.0 开发一个称之为 LiveScript 的脚本语言，起初的目的是同时在浏览器和服务器使用它。Netscape 与 Sun 公司联手及时完成了 LiveScript 的实现。就在 Netscape Navigator 2.0 即将正式发布前，Netscape 将其更名为 JavaScript，目的是为了利用 Java 这个 Internet 时髦词汇。Netscape 的这一决定也实现了当初的意图，JavaScript 从此变成了因特网的必备组件。

因为 JavaScript 1.0 如此成功，Netscape 在 Navigator 3.0 中发布了 JavaScript 1.1 版本。恰在那个时候，微软决定进军浏览器市场，发布了 IE 3.0b 并搭载了一个 JavaScript 的克隆版，叫做 JScript（这样命名是为了避免与 Netscape 潜在的许可纠纷）。微软步入浏览器领域的这重要一步当然推动了 JavaScript 的进一步发展。在微软进入后，有 3 种不同的 JavaScript 版本同时存在：Netscape Navigator 3.0 中的 JavaScript、IE 中的 JScript 以及 CEnvi 中的 ScriptEase。与其他编程语言不同的是，JavaScript 并没有一个标准来统一其语法或特性，而这 3 种不同的

版本恰恰突显了这个问题，随着用户担心的增加，这个语言的标准化显然已经势在必行。

1997 年，JavaScript 1.1 作为一个草案提交给 ECMA（欧洲计算机制造商协会）。第 39 技术委员会（TC39）被委派来“标准化一个通用、跨平台、中立于厂商的脚本语言的语法和语义”。由来自 Netscape、Sun、微软、Borland 和其他一些对脚本编程感兴趣的公司的程序员组成的 TC39 锤炼出了 ECMA-262，该标准定义了叫作 ECMAScript 的全新脚本语言。在 1998 年，该标准成为了国际标准（ISO/IEC 16262）。这个标准继续处于发展之中。在 2005 年 12 月，ECMA 发布 ECMA-357 标准（ISO/IEC 22537），主要增加了对扩展标记语言 XML 的有效支持。从此，Web 浏览器就开始努力（虽然有着不同程度的成功和失败）将 ECMAScript 作为 JavaScript 实现的基础。

1.1.2 JavaScript 的现状

2015 年 6 月 ECMAScript 6 已经正式发布，作为 ECMAScript 5.1 之后的一次重要改进，其目标是使参照此标准实现的 JavaScript 语言可以更适合用来开发大型的复杂的企业级应用。ECMAScript 6 添加了模块和类等工程化开发语言的必要特性，同时也添加了 Maps、Sets、Promise（异步回调）和 Generators（生成器）等一些实用特性。

虽然 ECMAScript 6 在语言标准上做了大量的更新，但其依旧完全向后兼容以前的版本，也就是说所有的在 ECMAScript 6 语言标准发布之前编写的 JavaScript 老代码都可以在实现了 ECMAScript 6 语言标准的浏览器或其他设备上正常运行。

截至目前，ECMAScript 6 的官方名称是 ECMAScript 2015，其下一个版本将在 2016 年发布，命名为 ECMAScript 2016。ECMA 今后将用频繁发布小规模增量更新的方式公布新的语言标准，所以，新的 JavaScript 语言版本将按照 ECMAScript 加年份的形式命名发布。

1.1.3 JavaScript 的定位

JavaScript 语言是一种脚本语言，ECMAScript 标准定义了其语法规则，JavaScript 语言的学习不仅仅是 JavaScript 语法学习，同时也要掌握 JavaScript 语言宿主的调用。JavaScript 语言宿主是指 JavaScript 语言的运行环境。在 Web 前端开发领域中，浏览器作为 JavaScript 语言宿主提供了许多对象供 JavaScript 语言调用。本书除了介绍 JavaScript 语言之外，也会讲解浏览器提供的对象如何使用。

随着 ECMAScript 标准的完善，各种优秀的 JavaScript 语言编译与运行环境不断涌现。随着 Node.js 等框架的出现，使 JavaScript 语言不仅仅可以被用在 Web 前端开发领域，还可以用在服务器程序的开发中。不过应用 JavaScript 语言进行服务器程序开发并不在本书的介绍范围之内，感兴趣的读者可自行查阅相关的技术资料。

1.1.4 JavaScript 在 Web 前端开发中的作用

HTML 超文本标识语言可用来制作所需的 Web 网页，通过 HTML 符号的描述就可以实现文字、表格、声音、图像、动画等多媒体信息的检索。然而采用单纯的 HTML 技术存在一定的缺陷，那就是它只能提供一种静态的信息资源，缺少动态的效果。这里所说的动态效果分为两种：一种是客户端的动态效果，就是我们看到的 Web 页面是活动的，可以处理各种事件，例如鼠标移动时图片会有翻转效果等；另一种是客户端与服务器端的交互产生的动态效果。实

现客户端的动态效果，JavaScript 无疑是一件适合的工具。JavaScript 的出现，使得信息和用户之间不仅只是一种显示和浏览的关系，而是实现了一种实时的、动态的、交互性的表达能力。从而基于 CGI 的静态 HTML 页面将被可提供动态实时信息并对客户操作进行响应的 Web 页面所取代。JavaScript 脚本正是满足这种需求而产生的语言。

JavaScript 是一种基于对象和事件驱动并具有安全性能的脚本编写语言，它采用小程序段的方式实现编程，像其他脚本语言一样，JavaScript 同样也是一种解释性语言，它提供了一个简易的开发过程。它的基本结构形式与 C、C++、VB、Delphi 十分类似。但它不像这些语言一样，需要先编译，而是在程序运行过程中被逐行地解释。在 HTML 基础上，使用 JavaScript 可以开发交互式 Web 网页，它是通过嵌入或调入在标准的 HTML 语言中实现的。JavaScript 与 HTML 标识结合在一起，实现在一个网页中链接多个对象，与网络客户交互作用，从而可以开发客户端的应用程序，其作用主要体现在以下几个方面。

①动态性。JavaScript 是动态的，它可以直接对用户或客户输入做出响应，无须经过 Web 服务程序。它对用户的反映响应，是采用以事件驱动的方式进行的。所谓事件驱动，就是指在主页中执行了某种操作所产生的动作，就称为“事件”。比如按下鼠标、移动窗口、选择菜单等都可以视为事件。当事件发生后，可能会引起相应的事件响应。

②跨平台。JavaScript 依赖于浏览器本身，与操作环境无关，只要能运行浏览器的计算机，并支持 JavaScript 的浏览器就可以正确执行。

③相对安全性。JavaScript 是客户端脚本，通过浏览器解释执行。它不允许访问本地的硬盘，并且不能将数据存入到服务器上，也不允许对网络文档进行修改和删除，只能通过浏览器实现信息浏览或动态交互，从而有效地防止数据的丢失。

④节省客户端与服务器端的交互时间。随着 WWW 的迅速发展。有许多服务器提供的服务要与客户端进行交互，如确定用户的身份、服务的内容等，这项工作通常由 CGI/PERL 编写相应的接口程序与用户进行交互来完成。很显然，通过网络与用户的交互过程一方面增大了网络的通信量，另一方面影响了服务器的服务性能。服务器为一个用户运行一个 CGI 时，需要一个进程为它服务，它要占用服务器的资源（如 CPU 服务、内存耗费等），如果用户填表出现错误，交互服务占用的时间就会相应增加。被访问的热点主机与用户交互越多，服务器的性能影响就越大。而 JavaScript 是一种基于客户端浏览器的语言，用户在浏览中填表、验证的交互过程只是通过浏览器对调入 HTML 文档中的 JavaScript 源代码进行解释执行来完成的，即使是必须调用 CGI 的部分，浏览器只将用户输入验证后的信息提交给远程的服务器，大大减少了服务器的开销。

1.1.5　Ajax

Ajax 即 Asynchronous JavaScript and XML（异步 JavaScript 和 XML），Ajax 并非缩写词，而是由 Jesse James Gaiiett 创造的名词，是指一种创建交互式网页应用的网页开发技术。Ajax 描述了把 JavaScript 和 Web 服务器组合起来的编程范型，JavaScript 是 Ajax 的核心技术之一，在 Ajax 技术架构中起着不可替代的作用。Ajax 是一种 Web 应用程序开发的手段，它采用客户端脚本与 Web 服务器交换数据，所以不必采用中断交互的完整页面刷新，就可以动态地更新 Web 页面。使用 Ajax 技术可以不必刷新整个页面，只是对页面的局部进行更新，而且还可以节省网络宽带，提高网页加载速度，从而缩短用户等待时间，改善用户的操作体验。

1.2 JavaScript 的运行

1.2.1 JavaScript 代码的装载与解析

当一个 HTML 页面被装载时，它会装载并解析过程中遇到的任何 JavaScript。Script 标签可以出现在文档的 head 中，也可以出现在 body 中。如果有指向外部 JavaScript 文件的链接，它会先装载该链接，再继续解析页面。嵌入第三方的脚本时，如果远程服务器因负担过重而无法及时返回文件，就有可能导致页面的装载时间显著变长。

代码解析是浏览器取得代码并将之转化成可执行代码的过程。这个过程的第一步是检查代码的语法是否正确，如果不正确，过程会立即失败。如果一个包含语法错误的函数被运行，将很可能会得到一条错误消息，显示函数还没定义。当浏览器确认代码合法之后，它会解析 script 块中所有的变量和函数。如果要调用的函数来自其他 script 块或者其他文件，需要确保它在当前 script 元素之前装载。

1.2.2 在 HTML 页面中嵌入 JavaScript

JavaScript 的脚本包括在 HTML 中，成为 HTML 文档的一部分，与 HTML 标识相结合，构成动态的、能够交互的网页。

1. 引入 JavaScript 脚本代码到 HTML 文档中

如果需要把一段 JavaScript 代码插入 HTML 页面，我们需要使用 script 标签（同时使用 type 属性来定义脚本语言）。这样，<script type="text/javascript">和</script>就可以告诉浏览器 JavaScript 代码从何处开始，到何处结束。浏览器载入嵌有 JavaScript 代码的 HTML 文档时，能自动识别 JavaScript 代码的起始标记和结束标记，并将其间的代码按照 JavaScript 语言标准加以解析并运行，然后将运行结果返回 HTML 文档并在浏览器窗口显示。

【例 1-1】将 JavaScript 代码嵌入到 HTML 文档中。

```
<html>
    <head>
        <title>JavaScript Test</title>
    </head>
    <body>
        <center>
            <script language="JavaScript" type="text/javascript">
                document.write("Hello World!");
            </script>
        </center>
    </body>
</html>
```

在例 1-1 的代码中除了 script 标记对之间的内容外，都是最基本的 HTML 代码，可见 script 标记可以将 JavaScript 代码封装并嵌入到 HTML 文档中。script 标记的作用是将 JavaScript 代码封装，并告诉浏览器其间的代码为 JavaScript 代码。这段 JavaScript 代码调用了 document 文档对象的 write()方法将字符串写入到 HTML 文档中。

下面重点介绍 script 标记的几个属性：

（1）language 属性：用于指定封装代码的脚本语言及版本，有的浏览器还支持 PERL、VBScript 等，所有浏览器都支持 JavaScript（当然，非常老的版本除外），同时 language 属性默认值也为 JavaScript。

（2）type 属性：指定 script 标记对之间插入的脚本代码类型。

（3）src 属性：用于将外部的脚本文件内容嵌入到当前文档中，一般在较新版本的浏览器中使用，一般使用 JavaScript 脚本编写的外部脚本文件使用.js 为扩展名，同时在 script 标记中不包含任何内容，如下：

```
<script language="JavaScript" type="text/javascript" src="Hello.js">
</script>
```

下面的例子演示了<script>标记的 src 属性如何引入 JavaScript 脚本代码。

【例 1-2】改写例 1-1 的代码并保存为 1-2.htm：

```
<html>
    <head>
        <title>JavaScript Test</title>
    </head>
    <body>
        <center>
            <script language="JavaScript" type="text/javascript" src="1-2.js">
            </script>
        </center>
    </body>
</html>
```

同时再编辑如下代码并将其保存为 1-2.js：

```
document.write("Hello World!");
```

将 1-2.htm 和 1-2.js 文件放置于同一目录，在浏览器中打开 1-2.htm，结果与例 1-1 显示相同。

可见通过外部引入 JavaScript 代码文件的方式，能实现同样的功能，并具有如下优点：

①将 JavaScript 代码同现有页面的逻辑结构及浏览器结果分离。通过外部代码，可以轻易实现多个页面共用实现相同功能的代码文件，以便通过更新一个代码文件内容达到批量更新的目的。

②浏览器可以实现对目标代码文件的高速缓存，避免额外的由于引用同样功能代码文件而导致下载时间的增加。

与 C++语言等使用外部头文件（.h 文件等）相似，引入 JavaScript 代码时使用外部代码文件的方式符合结构化编程思想。

引用外部文件中的 JavaScript 代码也必须更加谨慎。在某些情况下，引用的外部 JavaScript 代码文件由于功能过于复杂或其他原因导致的加载时间过长有可能导致页面事件得不到处理或者得不到正确的处理，程序员必须谨慎使用并确保脚本加载完成后其中的函数才被页面事件调用，否则浏览器报错。

综上所述，使用引入外部 JavaScript 代码文件的方法是效果与风险并存，开发者应权衡优缺点以决定是将 JavaScript 代码嵌入到目标 HTML 文档中还是通过引用外部代码文件的方式来实现相同的功能。

2. 嵌入 JavaScript 代码的位置

JavaScript 代码可放在 HTML 文档中任何位置。一般来说，可以在 head 标记、body 标记之间按需要插入 JavaScript 代码。

（1）head 标记之间放置

放置在 head 标记之间的 JavaScript 代码一般用于提前载入以响应用户的动作，一般不影响 HTML 文档的浏览器显示内容。如下是其基本文档结构：

```
<html>
    <head>
        <title>文档标题</title>
        <script language="javascript" type="text/javascript">
          //脚本语句…
        </script>
    </head>
    <body>
    </body>
</html>
```

（2）body 标记之间放置

如果需要在页面载入时运行 JavaScript 代码生成网页内容，应将脚本代码放置在 body 标记之间，可根据需要编写多个独立的代码段并与 HTML 代码结合在一起。如下是其基本文档结构：

```
<html>
    <head>
        <title>文档标题</title>
    </head>
    <body>
        <script language="javascript" type="text/javascript">
          //脚本语句…
        </script>
          //HTML 语句
        <script language="javascript" type="text/javascript">
          //脚本语句…
        </script>
    </body>
</html>
```

（3）在两个标记对之间混合放置

如果既需要提前载入脚本代码以响应用户的事件，又需要在页面载入时使用脚本生成页面内容，可以综合以上两种方式。如下是其基本文档结构：

```
<html>
    <head>
        <title>文档标题</title>
        <script language="javascript" type="text/javascript">
          //脚本语句…
        </script>
    </head>
```

```
    <body>
        <script language="javascript" type="text/javascript">
          //脚本语句…
        </script>
    </body>
</html>
```

在 HTML 文档中何种位置嵌入 JavaScript 代码应由其实际功能需求来决定。

1.3　JavaScript 的开发环境

由于 JavaScript 代码是由浏览器解释执行的，所以编写运行 JavaScript 代码并不需要特殊的编程环境，只需要普通的文本编辑器和支持 JavaScript 代码的浏览器即可。

JavaScript 语言编程一般分为如下步骤：

①选择 JavaScript 代码编辑器编辑脚本代码。

②嵌入该 JavaScript 代码到 HTML 文档中。

③选择支持 JavaScript 的浏览器浏览该 HTML 文档。

④如果错误则检查并修正源代码，重新浏览，重复此过程直至代码正确为止。

⑤处理在不同浏览器中 JavaScript 代码不兼容的情况。

1.3.1　编写 JavaScript 代码

由于 JavaScript 纯粹由文本构成，因此编写 JavaScript 代码可以用任何文本编辑器，也可以用编写 HTML 和 CSS 文件的任何程序，或者用像 Visual Studio 和 Eclipse 这样强大的集成开发环境。如果只是用作 Web 前端开发的话，还可以使用类似 Sublime Text 这样专注于代码编写的轻量级且功能强大的文本编辑工具作为 JavaScript 的编写工具。

Sublime Text 是一个功能强大的代码编辑器，支持多种编程语言的语法高亮，拥有优秀的代码自动完成功能，还拥有代码片段（Snippet）等功能，支持 Vim 模式，可以使用 Vim 模式下的多数命令，同时也支持宏。Sublime Text 还具有良好的扩展能力和完全开放的用户自定义配置与编辑状态恢复功能，支持强大的多行选择和多行编辑等。Sublime Text 也是一个跨平台的编辑器，同时支持 Windows、Linux、Mac OS X 等操作系统。

根据需要，本书将重点介绍如何将 Sublime Text 配置成易用且高效的 JavaScript 开发环境。

1．Sublime Text 的版本选择与获取

一般来说，如果是首次使用 Sublime Text，可以直接选择目前的最新版本 Sublime Text 3。Sublime Text 所有版本都可以在其官方网站（http://www.sublimetext.com）上直接下载，Sublime Text 虽然是一款收费软件，但是却可以无限期试用，所以，Sublime Text 对于初学者来说是非常友好的。

根据操作系统选择对应的版本下载安装后运行可以看到如图 1-1 所示的界面。本书选用的是 Sublime Text 3 x64 版本，系统环境为 Windows 10。

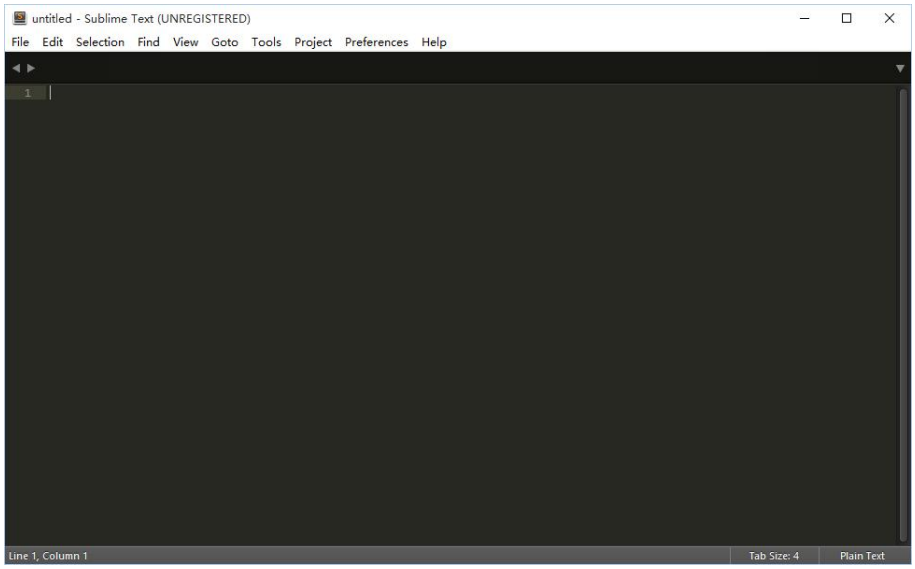

图 1-1　Sublime Text 3 界面

2. JavaScript 开发环境的配置

Sublime Text 3 是一款通用型文本编辑工具，默认安装后并未提供 JavaScript 开发环境所需的代码高亮和智能提示等功能，所以在开始编写 JavaScript 代码前还需安装一些 Sublime Text 的插件。

作为 JavaScript 代码编写工具，笔者在此推荐安装如下 Sublime Text 3 插件。

（1）Package Control

该插件的功能是为 Sublime Text 3 添加一个在线插件管理平台，让用户可以方便地查找与安装插件。Package Control 可以说是 Sublime Text 3 的必装插件。其安装步骤如下：

①运行 Sublime Text 3，使用快捷键 Ctrl+`调出控制台，如图 1-2 所示。

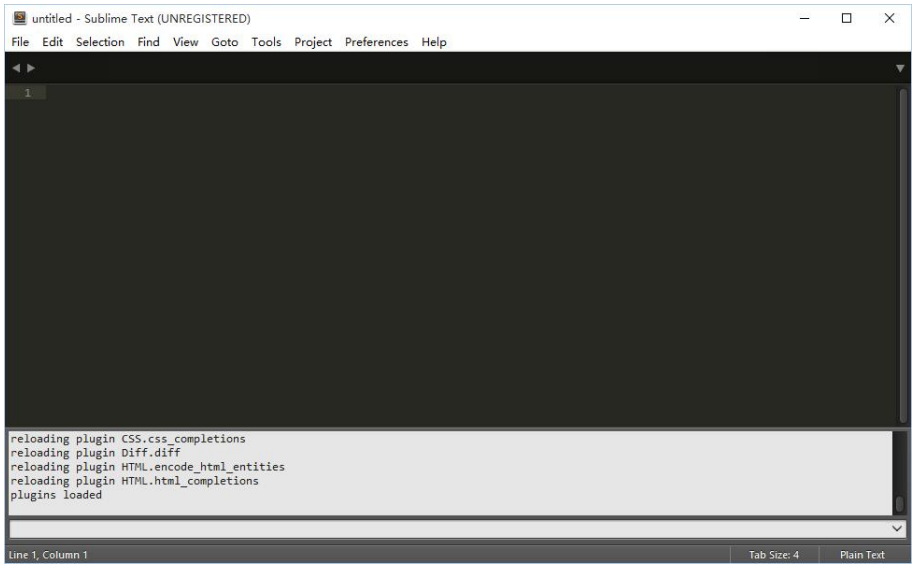

图 1-2　Sublime Text 3 控制台

②保持计算机连上 Internet，复制如图 1-3 所示的安装代码到控制台并运行。运行成功后重启 Sublime Text 3，如果在菜单 Perferences|Package Settings 中看到 Package Control 这一项，则表示安装成功。安装代码在 Package Control 官方网站（https://packagecontrol.io）上有提供，不要试图自行编写。

```
import urllib.request,os,hashlib; h =
'2915d1851351e5ee549c20394736b442' +
'8bc59f460fa1548d1514676163dafc88'; pf = 'Package
Control.sublime-package'; ipp =
sublime.installed_packages_path();
urllib.request.install_opener( urllib.request.build_opener(
urllib.request.ProxyHandler()) ); by = urllib.request.urlopen(
'http://packagecontrol.io/' + pf.replace(' ', '%20')).read();
dh = hashlib.sha256(by).hexdigest(); print('Error validating
download (got %s instead of %s), please try manual install' %
(dh, h)) if dh != h else open(os.path.join( ipp, pf), 'wb'
).write(by)
```

图 1-3　Package Control 安装代码

（2）Emmet

该插件的功能是可以通过 CSS 选择器自动生成 HTML，可以大大提高 HTML 代码的编写效率，对 JavaScript 程序的开发是很有帮助的。例如，通过“div#content>h1+p”这样的 CSS 选择器自动生成如下 HTML 代码：

```
<div id="content">
    <h1></h1>
    <p></p>
</div>
```

Emmet 插件的安装步骤如下：

①运行 Sublime Text 3，使用快捷键 Ctrl+Shift+P 调出 Package Control 命令面板，输入“Install”，选择 Package Control: Install Package 并运行，如图 1-4 所示。

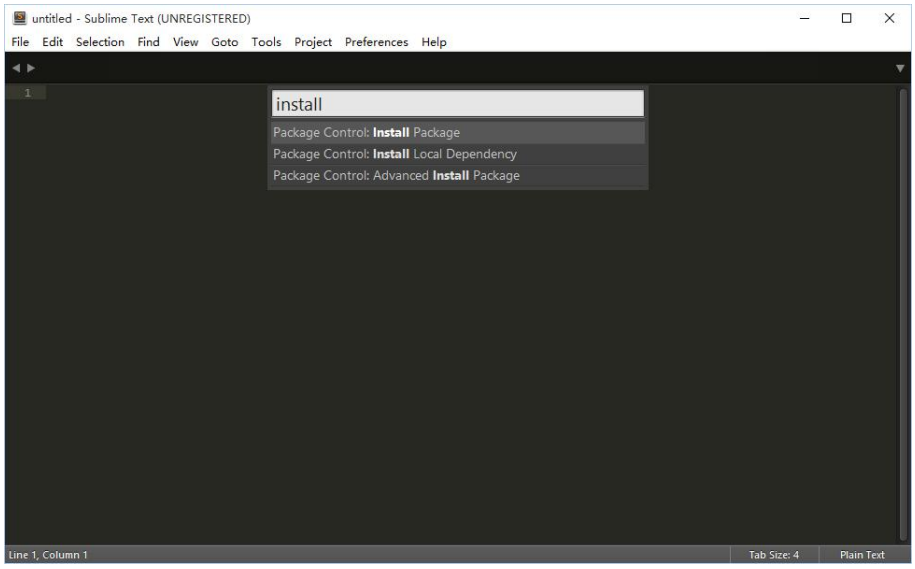

图 1-4　Package Control 命令面板

②在出现的插件选择面板中输入“emmet”，选择如图 1-5 所示的第一项并运行，注意状态栏的提示。在安装好后重启 Sublime Text 3 即可。

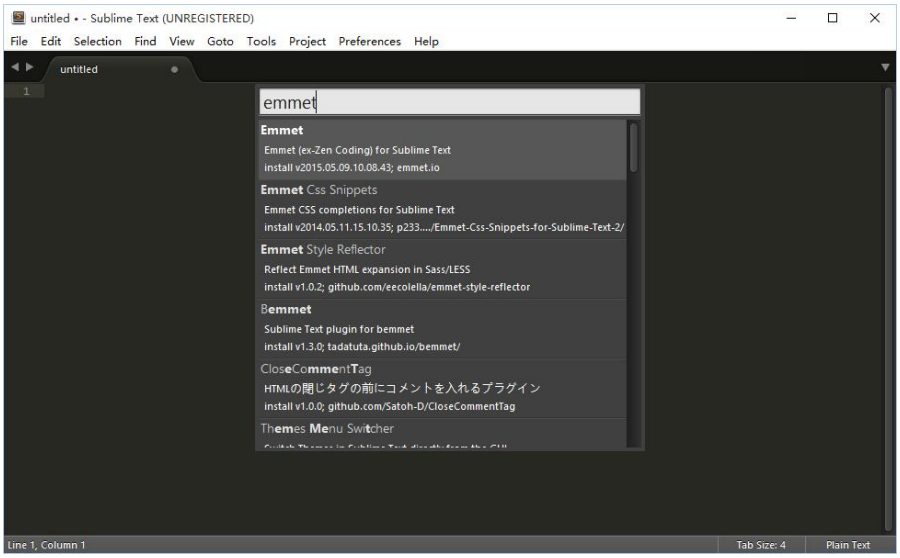

图 1-5　安装 Emmet

③安装好 Emmet 插件后可以在 Sublime Text 3 状态栏的最右边单击鼠标左键，在弹出的快捷菜单中选择 HTML 进入 HTML 编辑模式，如图 1-6 所示。

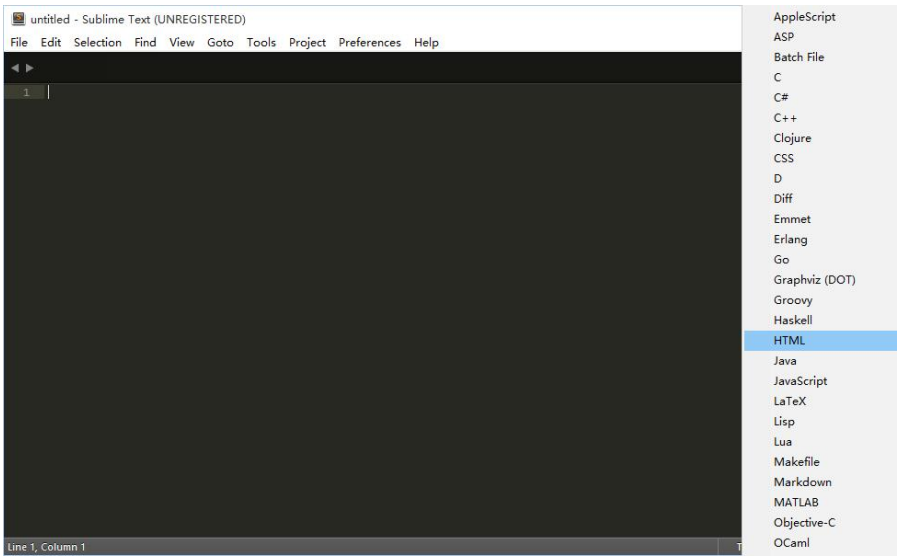

图 1-6　进入 HTML 编辑模式

④在 HTML 编辑模式下输入“div#content>h1+p”并按 Tab 键，如果自动生成了如图 1-7 所示的代码，则说明 Emmet 插件安装成功。由于 Emmet 插件的运行依赖于 PyV8 组件，在该组件没有正确自动安装的情况下 Emmet 插件也不能正常运行，所以在这种情况下还需手动安装 PyV8 组件，具体步骤可以参见其在 Github 上的页面（https://github.com/emmetio/pyv8-binaries）。

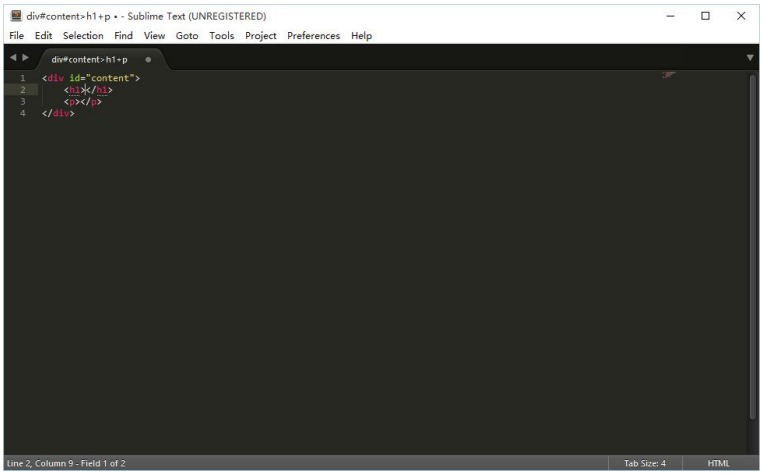

图 1-7　Emmet 插件辅助 HTML 编写

（3）JsFormat

该插件的功能为自动调整 JavaScript 代码的格式，不仅仅可以辅助调整我们自己编写的代码的格式，在阅读其他格式混乱的代码时，JsFormat 插件也特别有用。例如对于这么一段书写混乱的 JavaScript 代码：“var a="Hello World!";function b(){alert(a);}b();”，用 JsFormat 插件格式化后将变成如下样式：

```
var a = "Hello World!";
function b() {
    alert(a);
}
b();
```

而这样一段代码将容易理解得多。

JsFormat 插件安装步骤如下：

①运行 Sublime Text 3，使用快捷键 Ctrl+Shift+P 调出 Package Control 命令面板，输入“Install”，选择 Package Control: Install Package 并运行，如图 1-8 所示。

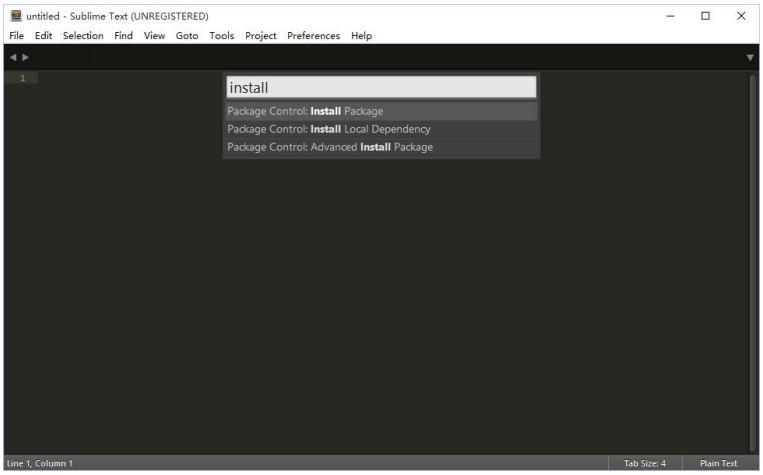

图 1-8　Package Control 命令面板

②在出现的插件选择面板中输入“JsFormat”，选择如图 1-9 所示的第一项并运行，注意状态栏的提示。在安装好后重启 Sublime Text 3 即可。

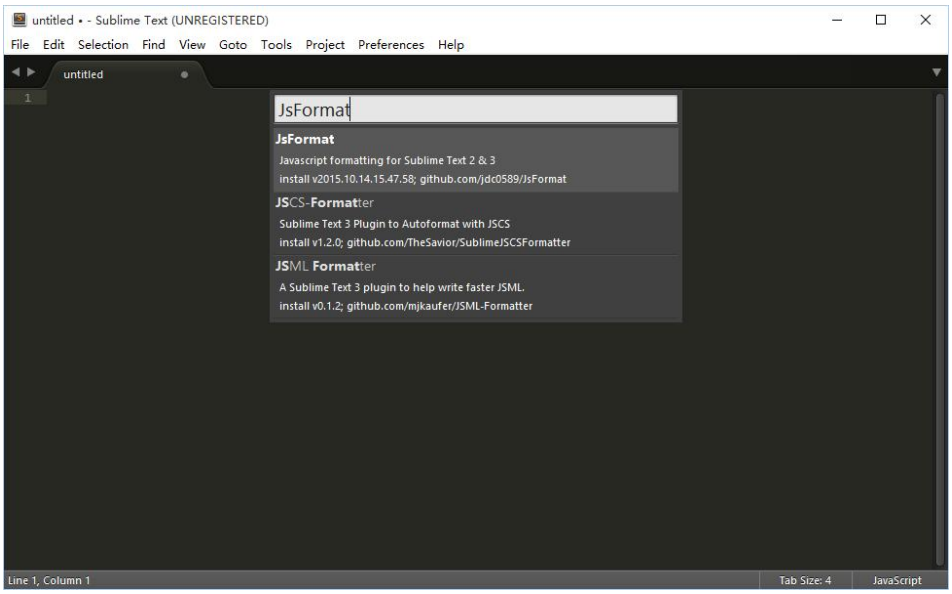

图 1-9　安装 JsFormat 插件

③安装好 JsFormat 插件后可以在 Sublime Text 3 状态栏的最右边单击鼠标左键，在弹出的快捷菜单中选择“JavaScript”进入 JavaScript 编辑模式，如图 1-10 所示。

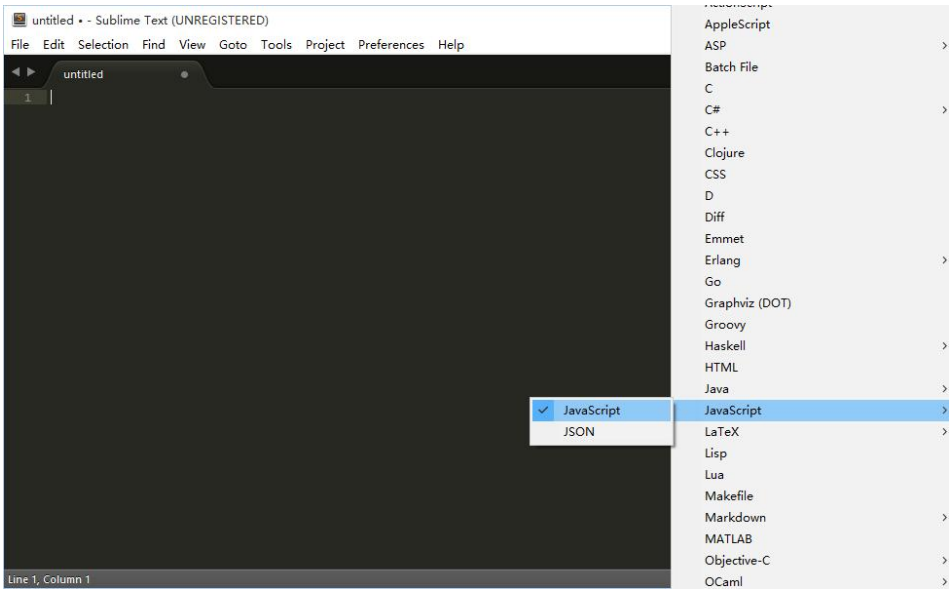

图 1-10　进入 JavaScript 编辑模式

④在 JavaScript 编辑模式下输入“function b(){alert(a);}”并按快捷键 Ctrl+Alt+F，如果自动格式化成如图 1-11 所示的代码，则说明 JsFormat 插件安装成功。

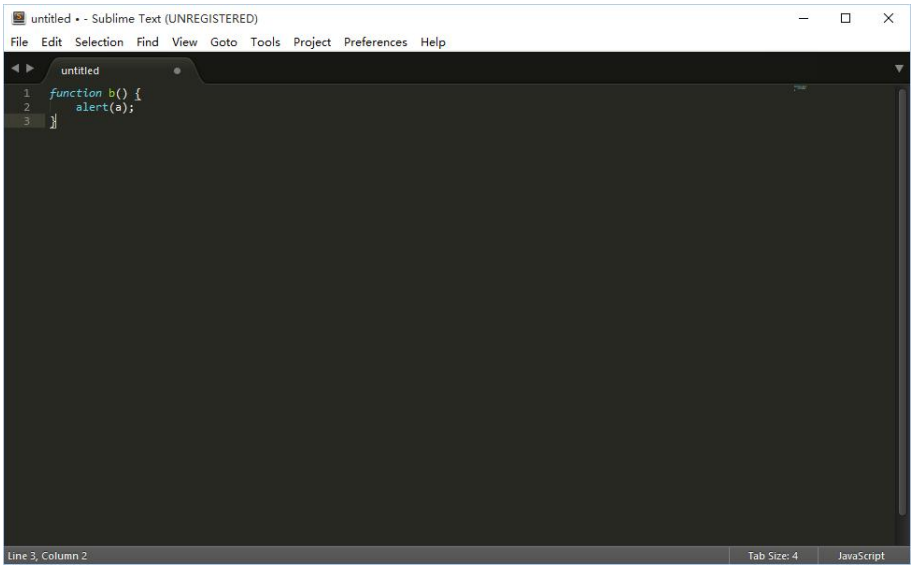

图 1-11　自动格式化的 JavaScript 代码

（4）SublimeCodeIntel

该插件的功能是在编辑器中提供 JavaScript 代码智能提示，对于 JavaScript 语言初学者来说，该插件可以说是必装插件。

SublimeCodeIntel 插件安装步骤如下：

①运行 Sublime Text 3，使用快捷键 Ctrl+Shift+P 调出 Package Control 命令面板，输入“Install”，选择 Package Control: Install Package 并运行，如图 1-12 所示。

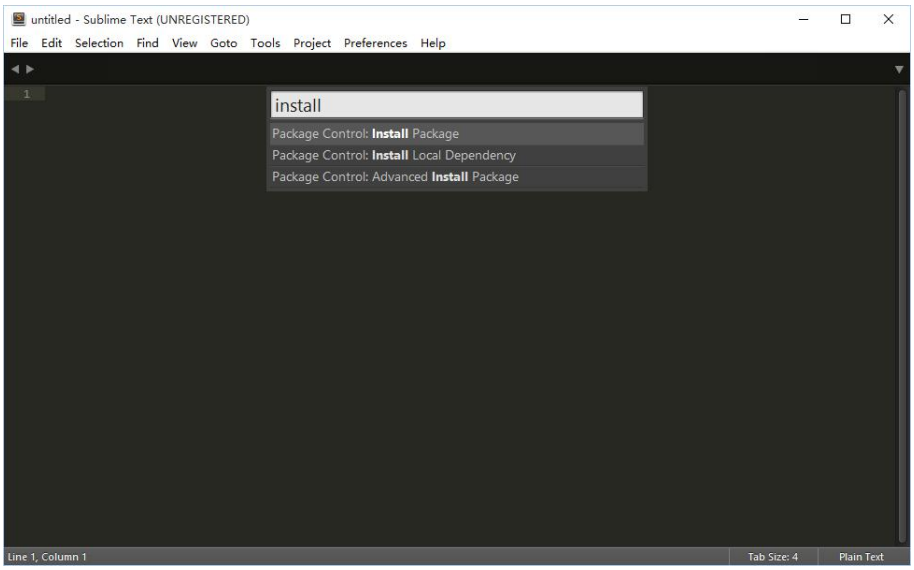

图 1-12　Package Control 命令面板

②在出现的插件选择面板中输入“SublimeCodeIntel”，选择如图 1-13 所示的第一项并运行，注意状态栏的提示。在安装好后重启 Sublime Text 3 即可。

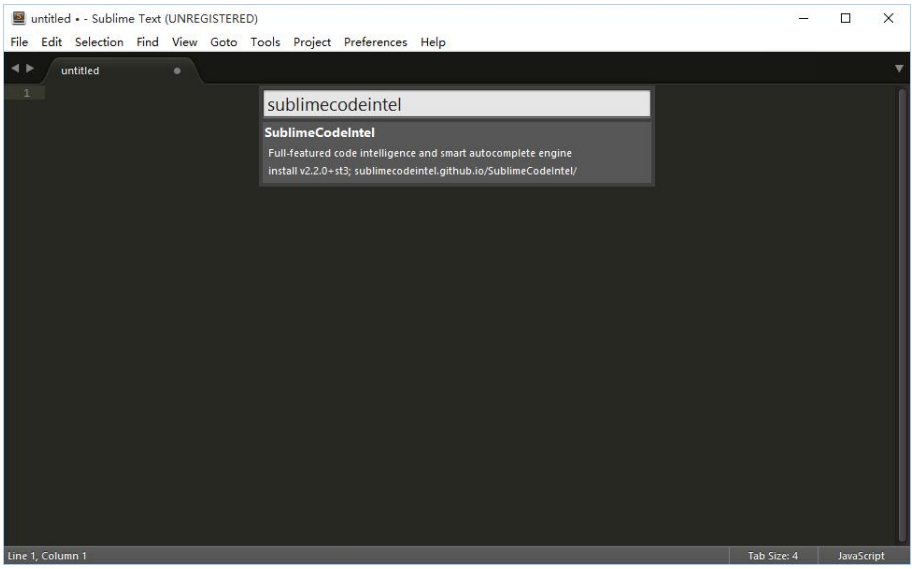

图 1-13　安装 SublimeCodeIntel 插件

③SublimeCodeIntel 插件安装所需的时间较长，安装好后会自动打开其说明文档，如图 1-14 所示。

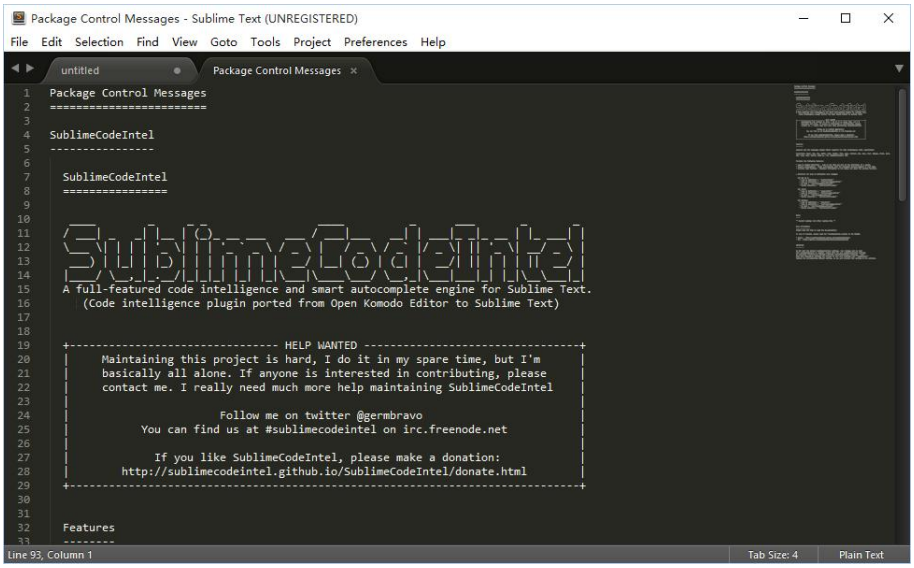

图 1-14　SublimeCodeIntel 插件安装成功

④SublimeCodeIntel 插件安装完成后，在 JavaScript 编辑模式下编写 JavaScript 代码时会自动弹出智能提示，如图 1-15 所示。

至此，JavaScript 代码的编辑环境已配置完成，虽说整个过程对初学者来说有些繁琐，不过一旦熟练使用这款编辑器后将能大大提升 JavaScript 代码的编写效率，从这点上来看花点时间完成编辑环境的配置是值得的，同时笔者也将尽可能在本书的配套资料中提供配置好的编辑器。

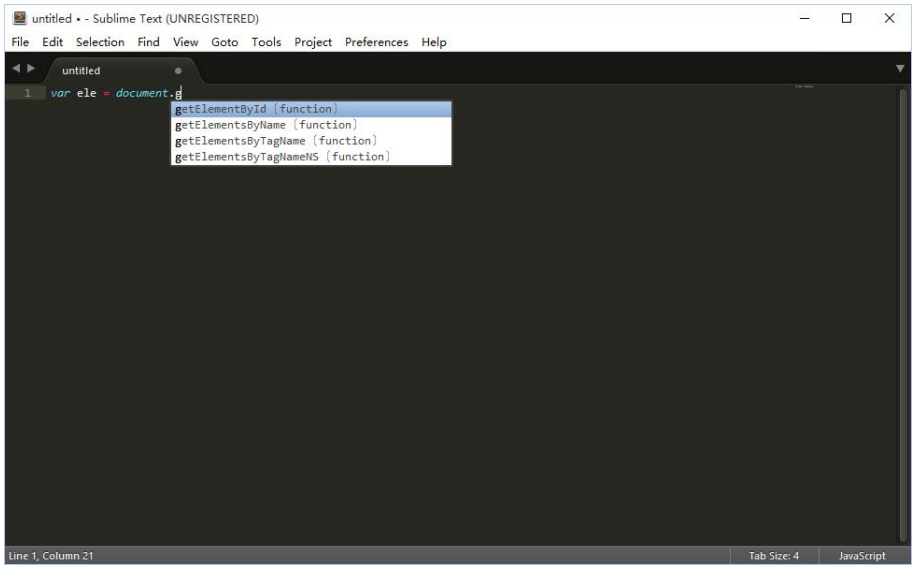

图 1-15　JavaScript 代码智能提示

1.3.2　运行与调试 JavaScript 代码

运行和调试 JavaScript 代码的主要工具是 Web 浏览器，主流的 Web 浏览器还会包含一些 JavaScript 调试程序。对于 JavaScript 代码来说，Mozilla Firefox 是最适合运行与调试用的浏览器之一。Mozilla Firefox Web 浏览器的插件 Firebug 是调试 JavaScript 代码必不可少的，尤其是 Ajax 应用的开发。这个插件程序允许检查一个 Web 页面的所有元素、查看 Ajax 调用的结果以及查看 CSS，所有这些都是实时的，这使得调试更加容易。

用 Firefox 浏览器在任意搜索引擎输入“Firebug”进入其官方网站后根据提示即可安装 Firebug 插件，本书使用的 Firefox 版本是 41.0.2。在 Firefox 中安装好 Firebug 后，我们可以在 Firefox 工具栏中找到 Firebug 插件的图标，如图 1-16 所示。

图 1-16　Firebug 插件

单击图标打开 Firebug，如图 1-17 所示。Firebug 界面上的 HTML、CSS、脚本和 DOM 标签页可以观察到每个元素的状态，在 HTML 标签页中选择一个元素，右侧面板就会显示选中元素上应用的样式信息。还可以进行编辑修改，修改结果会在 Firefox 窗口中实时显示出来，但是在这里进行的修改都是暂时的，刷新页面或关闭窗口之后就会丢失，绝对不会修改原始的文件。

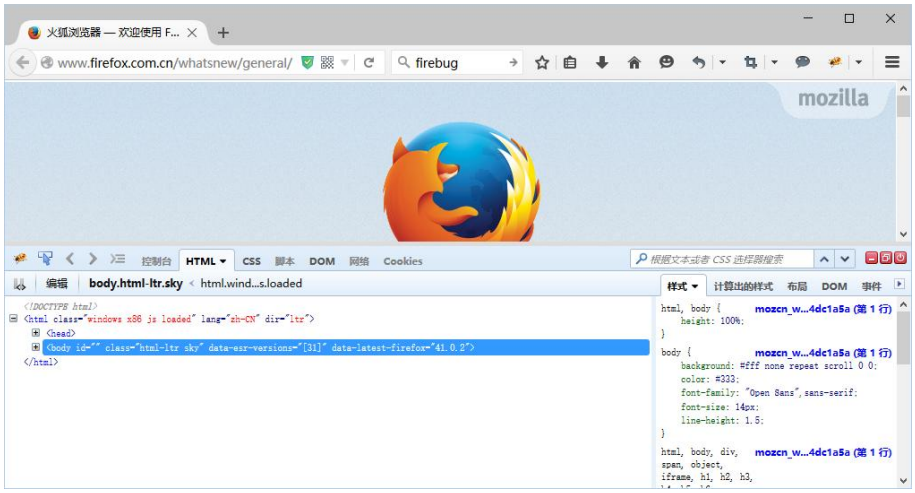

图 1-17　Firebug 中的 HTML 标签页

在 Firebug 窗口的菜单栏部分单击“控制台”标签页，如果控制台被禁用，可单击“控制台”标签页下的“启用”菜单打开并使用控制台，如图 1-18 所示。

图 1-18　Firebug 控制台

JavaScript 代码运行中的错误信息、Ajax 调用、性能分析结果、命令行执行结果都会显示在控制台的界面上。Firebug 提供了很多手段可以让我们将 JavaScript 代码运行中的信息输出到 Firebug 控制台。其中最常用的函数是 console.log()，它会将信息写入到控制台，并且不会干扰当前页面。如果将例 1-1 的结果改由控制台输出可以采用如下方式。

【例 1-3】使用 Firebug 控制台进行信息输出。

```
<html>
    <head>
        <title>使用 Firebug 控制台进行信息输出</title>
        <script type="text/javascript">
            console.log("Hello World!");
        </script>
    </head>
    <body>
    </body>
</html>
```

将该 HTML 文档在 Firefox 中打开，可以在 Firebug 的控制台界面看到如图 1-19 所示的输出结果。

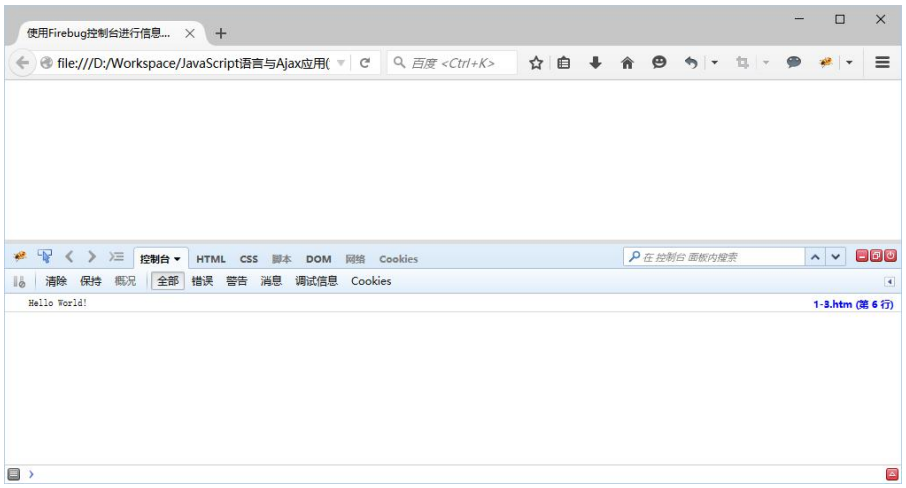

图 1-19　使用 Firebug 控制台进行信息输出

1.3.3　HTTP 调试

在 Web 中进行的所有操作都是运行在 HTTP 传输协议上面的，这是来回传递的信息包都使用的协议。如果能看到实际发送和接收的信息，不但对于 Ajax 调用，对于任何服务器/客户端交互都是很有帮助的。虽然有时候可以在后端记录这些信息，但是记录下来的信息不一定能真实反映出在前端发生的事情。目前有如下几种软件可以实现 HTTP 调试功能。

1．Firebug

通过使用 Firebug 的调试器跟踪 Ajax 调用，可以观察到请求首部和响应首部。这可以很方便地确认收到的数据是否正确。通过对调用的检查还能看到向服务器发送了什么数据，以及从服务器收到了什么数据。

2．Live HTTP Headers

如果要进行更细粒度的 HTTP 请求分析，可以使用 Live HTTP Headers。这个 Firefox 扩展能显示所有 HTTP 请求/响应信息，不但便于进行 Ajax 调用，而且便于监视页面请求（包括表单数据）、重定向甚至 Flash 中发出的服务器调用。它还能重发指定的请求，甚至在重发之前，

还允许修改请求首部，大大方便了对各种情形的测试。

Firebug 在响应信息的分析方面更出色，因此将它和 Live HTTP Headers 结合使用能得到更全面的分析。

3．ieHTTPHeaders

IE 也有类似的插件，名为 ieHTTPHeaders，可以用它分析往来的通信。

4．HttpFox

HttpFox 是一个 Firefox 的插件，功能简单使用方便，就是简单地捕获和查看网络链接。

5．Fiddler

Fiddler 是一款强大的 HTTP 调试软件，安装后同时支持 IE 和 Firefox，并且两个浏览器打开的调试界面是同一个，对于同时要在 IE 和 Firefox 下进行 HTTP 调试的开发人员来说是一个不错的选择。

6．Tamper Date

Tamper Date 是一个 Firefox 的插件，功能非常强大，除了可以捕获 HTTP 请求和应答数据之外，最大的优点是可以自定义 HTTP 请求。

7．HttpWatch

HttpWatch 是一款商业版的 HTTP 分析工具，同时支持 IE 和 Firefox，功能强大。

8．HTTP Analyzer

HTTP Analyzer 也是一款商业版的 Http 分析工具，目前只支持 IE。

本章小结

JavaScript 最初是由 Netscape 公司开发的一种基于对象和事件驱动并具有安全性能的脚本语言，或者称为描述语言，只能用在 Internet 的客户端上。目前的 JavaScript 语言可以运行在多种平台之上，其标准由 ECMA 组织维护。在 HTML 页面中使用 JavaScript 可以开发交互式 Web 页面。JavaScript 使得 Web 网页和用户之间实现了一种实时性的、动态的、交互性的关系，使网页包含更多活跃的元素和更加精彩的内容。

本章主要介绍了 JavaScript 语言的发展历史、使用特点、功能和未来，还介绍了如何在 HTML 页面中嵌入 JavaScript 代码、编辑 JavaScript 代码的环境以及调试 JavaScript 代码的常用工具。

习　　题

1-1　什么是 JavaScript？

1-2　如何在 HTML 页面中嵌入 JavaScript？

1-3　编写一段 JavaScript 代码，在 Firebug 的控制台中输出信息。