# 从Windows到Linux的应用移植实现

## 之 浏览器应用技术篇

■ 兰雨晴 洪雪玉 编著

国防工业出版社
National Defense Industry Press

# 从 Windows 到 Linux 的应用移植实现

# 之

## 浏览器应用技术篇

兰雨晴　洪雪玉　编著

国防工业出版社

·北京·

# 内 容 简 介

待移植的应用系统主要采用 C/S、B/S 或二者混合的应用结构。其中，实现 B/S 应用系统的移植，根据近两年的实践经验，Windows 和 Linux 平台上浏览器的应用兼容性问题较为突出，例如，页面解析异常、展示效果不同、脚本执行异常、插件不可用等。这就是本书要关注和解决的主要问题。

在内容编排上，本书包括两大部分内容。第一部分是基本理论，概述了浏览器的发展、浏览器体系结构以及浏览器应用开发技术，重点介绍了浏览器解析原理和渲染模式，为后续章节中基于浏览器的应用移植提供理论基础。第二部分是 IE 到 Firefox 的应用移植实现，重点分析了 IE 和 Firefox 浏览器对 HTML、CSS、JavaScript、插件、扩展等标准和基础应用开发技术的支持程度及差异性，并提供相应的移植解决方案。

本书可作为专业技术人员，诸如软件分析人员、设计人员、开发者、软件工程师和编程人员等的参考用书。

**图书在版编目(CIP)数据**

从 Windows 到 Linux 的应用移植实现之浏览器应用技术篇/兰雨晴，洪雪玉编著. —北京：国防工业出版社，2014.1

ISBN 978-7-118-09134-2

Ⅰ.①从… Ⅱ.①兰…②洪… Ⅲ.①浏览器－计算机技术 Ⅳ.①TP393.092

中国版本图书馆 CIP 数据核字(2013)第 270148 号

※

国防工业出版社出版发行

(北京市海淀区紫竹院南路 23 号 邮政编码 100048)

北京奥鑫印刷厂印刷

新华书店经售

*

**开本** 787×1092 1/16 **印张** 14 **字数** 346 千字

2014 年 1 月第 1 版第 1 次印刷 **印数** 1—7000 册 **定价** 48.00 元

**(本书如有印装错误，我社负责调换)**

国防书店：(010)88540777 发行邮购：(010)88540776

发行传真：(010)88540755 发行业务：(010)88540717

# 前　言

近年来，出于安全方面的考虑，越来越多的用户已经把基于 Windows 等国外商用软件平台的应用系统移植到国产软件平台上，或者正打算这么做，这类需求在国防、政府、金融等关键领域显得尤为突出。随着国产 Linux 操作系统及其配套基础软件的进一步完善，基于国产基础软件构建我国关键领域应用的时机越发成熟。在国家“十一五”和“十二五”规划的推动下，预计未来几年，将有大量的应用系统需要移植到国产软件平台上。

为了顺利实施应用系统移植，首先需要解决 Windows 等国外商用系统与国产 Linux 系统在平台开发技术方面的差异性。中标软件有限公司（以下简称中标软件）作为主要的国产操作系统提供商，为了大力推进国产化进程，促进相关产业链的成熟，成立了专门的技术团队，专注于应用移植技术的研究以及移植方案的设计实现。基于多年的移植工作经验，中标软件于 2013 年 1 月出版了《从 Windows 到 Linux 的应用移植实现之平台技术与接口篇》一书。此书较为全面地阐述了 Windows 和 Linux 在平台开发技术方面的差异性，如何实现应用系统的迁移移植，主要内容包括应用移植的主体工作、移植过程和移植策略，以及两个平台在网络通信、多进程/多线程、图形界面等技术实现上的差异性，并提供了对应的移植方案。

从另外一个角度，原有的应用系统主要采用 C/S、B/S 或二者混合的应用结构。实现 C/S 应用系统的移植，首先需要解决的是 Windows 和 Linux 在平台开发技术方面的差异性。而实现 B/S 应用系统的移植，根据近两年的实践经验，两个平台上浏览器的应用兼容性问题较为突出，例如，页面解析异常、展示效果不同、脚本执行异常、插件不可用等。这就是本书要关注和解决的主要问题。

在内容编排上，本书包括两大部分内容。第一部分是基本理论，概述了浏览器的发展、浏览器体系结构以及浏览器应用开发技术，重点介绍了浏览器解析原理和渲染模式，为后续章节中基于浏览器的应用移植提供理论基础。第二部分是 IE 到 Firefox 的应用移植实现，重点分析了两个浏览器对 HTML、CSS、JavaScript、插件、扩展等标准和基础应用开发技术的支持程度及差异性，并提供相应的移植解决方案。

本书可作为专业技术人员，诸如软件分析人员、设计人员、开发者、软件工程师和编程人员等的参考用书。如果需要从事具体的浏览器应用移植工作，建议重点阅读第 2 章，以及第 4 章至第 8 章的内容，这里给出了具体的差异性问题和解决方法。

本书由兰雨晴、洪雪玉编著。感谢高军、张杨、贾艳彬为本书的编写提供了有力的技术支持，感谢余丹提供的宝贵意见。

由于时间仓促，书中难免存在错误和不足之处，欢迎读者批评指正。

编者

于中标软件有限公司

# 目　录

## 第一部分　基本理论

## 第二部分 IE 到 Firefox 的应用移植实现

# 第一部分 基本理论

## 第1章 浏览器发展概述

在Internet出现之后，浏览器几乎和HTML同步发展。HTML是Internet之上重要的信息载体，而浏览器是HTML的解析表现工具。因此，浏览器是互联网时代重要的基础软件。

在论述浏览器的架构和组件等技术细节之前，有必要概述一下浏览器的发展历史和家族分类。伴随着HTML语言的发明，计算机科学家Tim Berners-Lee在1990年开发了HTML的解析与显示程序，即最早的Internet浏览器。从1994年Netscape公司首先发布商用版的浏览器至今，浏览器走入大众已有近20年时间。世界上的浏览器产品有IE、Firefox、Chrome，以及国产的搜狗、360、遨游、世界之窗等至少几十种。浏览器的功能也从单一的解析显示HTML，到目前的具有渲染CSS、执行JavaScript，以及丰富的扩展和插件功能，甚至Google推出面向Chrome浏览器风格的Google操作系统。

然而，以繁杂的产品探讨浏览器发展历史往往无法找到主线。好在浏览器最核心的部件内核实际上并不多，而且基于内核能够展现出浏览器发展的脉络。因此，本章从最早的Mosaic内核开始，探讨目前四大浏览器内核的产生和发展历程，以及相关产品。

### 1.1 Mosaic和早期浏览器

尽管在1991年至1994年间HTML的发明者和美国伊利诺伊州的伊利诺伊大学厄巴纳的美国国家超级计算应用中心(National Center for Supercomputing Applications，NCSA)开发了众多的浏览器实验版本，然而世界第一个具有深远影响意义的网页浏览器是由NCSA在1993年开发的Mosaic，该浏览器的内核也被习惯地称为Mosaic。

Mosaic起初的Alpha版构建在Unix的X Window系统上，后来逐步开发了对Macintosh和Windows操作系统的支持版本。NCSA的技术团队以Mosaic为基础创建了Mosaic公司，但遗憾的是Mosaic的版权在伊利诺伊大学手中，而且伊利诺伊大学将Mosaic技术卖给了Spy Glass公司。因此，Mosaic公司不得不丢弃Mosaic浏览器相关技术文档，从头开始开发一个新的浏览器，而该浏览器就是第一个获得巨大成功的商业浏览器产品“Netscape Navigator”，Mosaic公司也随后更名为“Netscape Communication Corporation”。

Netscape浏览器并没有把脚步停止在HTML上，Netscape是最早支持用于互联网上且地位不亚于HTML的JavaScript语言的浏览器。JavaScript的前身Cmm语言在1992年由Nombas公司开发，并在Netscape刚刚诞生时研发了几个实验性的Cmm解析软件。受其启发，Netscape公司的Brendan Eich仅用10天时间即发明了JavaScript语言，并在Netscape 2.0中被支持。

在随后的 1996 年秋天，Eich 开发了世界第一款 JavaScript 引擎 SpiderMonkey。

鉴于对互联网的重视，微软公司决定为自己的 Windows 95 操作系统开发一款网页浏览器。微软公司购买了 Spy Glass 公司的 Mosaic 浏览器的技术授权，在此基础上于 1995 年开发了 Internet Explorer(IE)浏览器的第一个版本，并从 Windows 95 R2 开始与操作系统捆绑。

微软 IE 浏览器的前两个版本不温不火，但是从 1996 年发布的 IE 3.0 版本开始，大范围地减少 Spy Glass 技术的使用，同时包括多种新特性，例如首次支持 CSS 技术、引入 ActiveX 控件技术、Java Applet 技术和内嵌网页多媒体技术等。这使得微软的 IE 浏览器比同期的 Netscape 浏览器更为先进，因此大受欢迎，为随后微软和 Netscape 的浏览器大战中微软胜出奠定了重要基础。之后，在微软于 1997 年发布的 IE 第四版中，则完全抛弃了 Spy Glass 的 Mosaic 技术，转为以自己开发的称为 Trident 排版引擎作为浏览器内核，使 IE 浏览器迈入了崭新的阶段。

1998 年，Netscape 在与微软的浏览器大战中失利之后，开放了当时的 Netscape 4.0 的源代码，并且成立了 Mozilla 组织，Mozilla 随后开发出了功能和稳定性比之前 Netscape 各个版本更为出色的 Gecko 内核。之后 Netscape 公司被美国在线(AOL)收购，收购之后的新版本 Netscape 即开始使用 Gecko 内核，由此结束了以 Mosaic 为基础的早期浏览器时代。

早期的两大主流浏览器 IE 和 Netscape 都受了 Mosaic 很大的影响。IE 直接基于 Mosaic 的技术而开发，而 Netscape 虽然没有基于 Mosaic 开发，但是由于 Netscape 的开发者几乎是 Mosaic 开发者的原班人马，因此在技术上一脉相承。由此可见，Mosaic 是早期浏览器发展的基石。

## 1.2 Trident

虽然 Mosaic 浏览器技术对早期的微软 IE 浏览器至关重要，但是从微软 1997 年发布的 IE 4.0 版开始，摒弃了对 Mosaic 技术的依赖，开发并使用了自己称为 Trident 的排版引擎作为新的浏览器内核。

Trident 内核的设计基于微软研发的组件对象模型(COM)技术。Trident 作为一个二进制 COM 服务器组件在浏览器中，由 C++编写的 COM 客户端可以嵌在 IE 浏览器中，并透过 Trident 引擎存取当前显示在浏览器上的网页内容及网页的各种元素的值，从浏览器控件触发的事件亦可被程序获取并进行处理。因此，Trident 内核不仅提供了浏览器解析 HTML 语句并排版显示的功能，而且还提供了 IE 浏览器一定的可编程性和互操作性。

与 Trident 类似的另一个排版引擎 Tasman 也由微软开发，被用作 Macintosh 版的 IE 浏览器的内核。目前微软已经停止了 Macintosh 版 IE 浏览器的开发，但是 Tasman 还在继续，不过对 Tasman 的支持力度明显不如 Trident 引擎。

Trident 从 IE 4.0 一直到目前的 IE 10.0，IE 浏览器的每次更新，Trident 都有相应的提高。但是，在 IE 8.0 之前 Trident 都没有独立发布。直到 IE 8.0 发布，Trident 才独立发布，Trident 首次独立发布的版本是 4.0，该版本比以前的 IE 浏览器都更大范围地兼容了 W3C 标准。对应 IE 9.0 的 Trident 5.0 版本对于微软浏览器开发团队来说是一个具有里程碑意义的产品。Trident 5.0 首次提供了对 HTML 5、SVG 和 CSS 3 的支持，并采用新的 JavaScript 引擎，这无疑是对业界普遍看好的 HTML 5 技术的一次重大推动。IE 浏览器重要历史更新如表 1-1 所列。

表 1-1　IE 浏览器重要历史更新

| 版本 | 发布日期 | 重要改进/事件 |
| --- | --- | --- |
| 1.0 | 1995 年 8 月 | IE 的首发浏览器版本 |
| 2.0 | 1995 年 10 月 | 支持 HTML 表格、框架和其他元件 |
| 3.0 | 1996 年 3 月 | 改进对 HTML 表格、框架、MID 音乐、GIF 动画和其他元件的支持 |
| 4.0 | 1997 年 4 月 | 改进对 CSS 和 Microsoft DOM 的支持 |
| 5.0 | 1999 年 3 月 | 支持 CSS2 属性、框架、XML/XSL |
| 5.5 | 2000 年 7 月 | Windows 9X 系列核心的最终版本 |
| 6.0 | 2001 年 8 月 27 | 更多 CSS 支持和错误修正以遵循 W3C 标准 |
| 7.0 | 2006 年 10 月 18 | 支持 PNG 透明、CSS 错误修正和分布浏览，RSS 技术支持，界面快速索引标签，改进稳定性、安全性和兼容性 |
| 8.0 | 2009 年 3 月 20 | 私密浏览、Web Slices 和自动标签崩溃恢复特性，对 Web 标准的支持度也有极大改进 |
| 9.0 | 2011 年 3 月 15 | 新 JavaScript 引擎，针对文本、图形、视频全面硬件加速，增加 HTML 5 相关技术 |

由于 IE 浏览器的广泛使用，使得很多网站和企业系统都是依据 IE 浏览器作为默认客户端开发并测试的。因此为了最大限度地兼容应用，造成了很多其他浏览器也都基于 Trident 内核开发，比如 Avant Browser、Maxthon、腾讯 TT、MyIE 等。

但是，随着其他内核浏览器的快速发展，以及应用本身越来越注意不同浏览器的兼容性；使得 IE 浏览器的市场份额不断降低，越来越多的多内核浏览器出现，而不只是基于 Trident 内核。

在 IE 9.0 之前，JavaScript 脚本的解析并没有单独的引擎支持。而从 IE 9 开始，微软开发了称为 Chakra 的 Jscript 引擎与 Trident 配合。Jscript 是微软对 ECMAScript 语言的实现，与其他浏览器支持的 JavaScript 略有区别。Chakra 支持在一个独立的 CPU 核心上即时编译脚本，与浏览器并行。该引擎也能够访问电脑的图形处理器(GPU)，特别是对 3D 图形和视频的情况。Chakra 使得 IE 9.0 对 JavaScript 的解析执行效率远高于 IE 之前的版本，但是仍然慢于 Firefox 3.6、Chrome 4.0。IE 比其他浏览器的执行效率慢可能和 IE 支持的 Jscript 语言比一般的 JavaScript 语言功能更为丰富有关。

## 1.3　Gecko

Gecko 网页排版引擎是由 Netscape 公司发起开发的。1997 年，在与微软的浏览器争夺中处于劣势地位的 Netscape 决定开发新的网页排版引擎，希望有更高的执行速度和 W3C 标准兼容度。为此，Netscape 成立了 Mozilla 组织。在 1998 年初，Mozilla 研发出了称为 Raptor 的新排版引擎。但由于商标问题，Raptor 随后更名为 NGLayout，很快 Netscape 最终将其命名为 Gecko。

Netscape 公司随即终止了基于 Mariner 排版引擎的 Netscape 5.0 版本开发，全力围绕 Gecko 内核开发 Netscape 6.0，并在 2000 年发布了该版本的浏览器。在 2003 年 7 月 15 日 Netscape 公司解散的同时，Mozilla 组织升级为 Mozilla 基金会，Gecko 继续由 Mozilla 基金会的雇员和义工维护、发展至今。

Gecko 由 C++语言编写，具有跨平台的特点，支持 Linux、Mac OS X、Solaris、OS/2、AIX、MS Windows 等系统。Gecko 为开源软件，遵从 MPL/GPL/LGPL 协议。最新的 Gecko 版本支持部分 HTML 5.0、部分 CSS 3.0、部分 ECMAScript 5.0 语法标准、部分 DOM 3 标准，

支持 XML 1.0、XHTML 1.0、XSTL 和 Xpath 以及 MathML 和 XForms 等。

与 Gecko 配合使用的重要组件是用来解析执行 JavaScript 的 SpiderMonkey。SpiderMonkey 由 Netscape 公司的 Brendan Eich 在 1996 年用 C++写成，包括四个部分：一个解析器、若干 JIT 即时编译器、一个反编译器和一个垃圾收集器。

SpiderMonkey 共有过三种 JIT 编译器：TraceMonkey、JaegerMonkey 和 IonMonkey。TraceMonkey 是 JavaScript 语言的第一个 JIT 编译器，它在解析执行的过程中记录控制流和数据类型，构建本地代码流程树，以提高执行效率。但在 Firefox 11 后被 JaegerMonkey 替代。

JaegerMonkey 是在 Firefox 4.0 引入的全方法 JIT 编译器，用来取代不能稳定的生成本地代码的 TraceMonkey。与一般的编译器通过优化控制流图表示函数不同，JaegerMonkey 通过操作 SpiderMonkey 的字节码来提高执行速度。

IonMonkey 是 Mozilla 用来取代 JaegerMonkey 的下一代 JIT 编译器。IonMonkey 显得更为传统，它利用静态单指派表(Static Single Assignment Form，SSA)作为中间表示，将 SpiderMonkey 的字节码转换为控制流图。这样，IonMonkey 便拥有了传统的编译器优化能力，如函数内联、线性搜索寄存器分配、死锁消除和循环不变式移动等。

SpiderMonkey 不仅用于 Mozilla 的浏览器套件(Firefox、Thunderbird、SeaMonkey)，还广泛用于其他软件，用于解析执行 JavaScript，如 Adobe 套件、Gnome 3.0、Yahoo！Widget、MongoDB 等。

一种与 SpiderMonkey 类似的 JavaScript 解析器为 Rhino。Rhino 在 1997 年由 Netscape 公司利用 Java 语言开发，原本是用于 Java 版的浏览器中的 JavaScript 解析器，但是目前 Rhino 广泛用于对在 Java 嵌入 JavaScript 脚本的支持，同时，Rhino 也可以将 JavaScript 代码编译为 Java 二进制代码。

基于 Gecko 内核和 SpiderMonkey 引擎最重要的浏览器产品是 Mozilla Firefox。在 Firefox 之前，Mozilla 于 2002 年发布的第一款浏览器产品为 Phoenix 0.1，代号为 Pescadero；2003 年发布的更新版本又将该产品更名为 Firebird 0.6，代号为 Glendale。随后沉寂了 3 年时间没有更新，直到 2006 年，Mozilla 在新的浏览器版本中将名称正式定为 Firefox，当时的版本是 Firefox 2.0。

使用最广泛的产品是于 2007 年后陆续发布的 Firefox 3.x 版本，更创造了 Mozilla Firefox 3.0 发布首日全球下载量突破 800 万的世界吉尼斯纪录。之后又是长达 4 年的沉寂，在 2011 年 Mozilla 发布了 Firefox 4.0，该版本较之前的 3.0 版本有了巨大变化，不仅包括更多的标准兼容，而且更新了 Gecko 版本，添加并修改了大量的扩展和插件编程接口，大大提高了浏览体验并降低了开发难度。在 Firefox 4.0 之后，版本的升级如井喷一样频繁，截止到 2012 年 11 月份，版本已经达到了 Firefox 16。

Mozilla Firefox 的优点主要包括页面加载迅速，JavaScript 执行流畅，硬件加速技术增强图形渲染能力，以及颇为成熟经典的插件和扩展机制。

Mozilla Firefox 的优点很多，与之相对的缺点也不少，尤其是广泛应用的 Firefox 3.x 版本。最重要的缺点是内存管理策略简单，扩展或插件执行时的垃圾收集直到内存占用达到很高的程度的时才执行。此外，利用 C++编写的 Firefox 插件极易出现内存泄漏和各种异常，而且 Firefox 3.x 中不支持插件和浏览器的线程分离。但是，在现阶段的 Firefox 16 中对这些缺点已经进行了不同程度的解决，而且研发出了新一代的扩展开发工具 Add-on SDK 和 Add-on Builder，利用 JavaScript 大大降低了 Firefox 扩展的开发难度。

另两款基于 Gecko 内核的重要开源浏览器是 Galeon 和早期的 Epiphany。Galeon 于 2000

年 6 月由 Macro Pesenti Gritti 发布第一版，即 Galeon 0.6，该款基于 Gecko 内核的浏览器甚至比 Mozilla 基金会 Mozilla 1.0 发布的还早。在 2002 年发布用 GTK 2 编写的 Galeon 1.3 之后，Macro 开始了新的称为 Epiphany 的浏览器的开发。

Galeon 的特点是简单和配置丰富，而 Epiphany 的最大特点是简洁和易用。正如其官方网站上所说的，Epiphany 让你能专注于互联网的内容，而不是为功能繁杂的浏览器所困扰。就影响力而言，Epiphany 更广泛，Epiphany 已经成为 Gnome 3 的指定浏览器，目前的稳定版本是 Epiphany 3.0.4。但是，基于 Gecko 内核的 Epiphany 仅限于低版本，从 2008 年的 Epiphany 2.28 开始，便放弃了 Gecko 内核，开始基于 Webkit 内核。

## 1.4 KHTML 和 WebKit

在浏览器的发展历史中，KHTML 是在 Linux 下的桌面环境 KDE 中不起眼的网页排版引擎。KHTML 出现自 1998 年，Waldo Bastian 重构了 C++ khtmlw 库，增加了 Unicode 支持和 Qt 2 的支持，形成了 KHTML 的早期版本。

在 1999 年，Lars Knoll 启动了 KHTML 工作，重写了以前的代码，增加了对 W3C DOM 标准的支持，同时集成了 Harris Porten 的 JavaScript 引擎 KJS。之后 Knoll 继续为 KHTML 增加对 CSS 标准的支持，并稳定了软件体系结构。最终在 2000 年 KDE 2 发布时，KHTML 终于被包括进来，成为 KDE 环境下浏览器 Konqueror 的内核。

其在执行速度方面不如 Mozilla Gecko，对 SVG 的支持并不完整，语法要求却更为严格。由于 IE 浏览器的广泛使用，KHTML 又不得不做出妥协，兼容很多 IE 浏览器特有的非标准语法。很明显，KHTML 也许只能算作 Linux 下开源网页排版引擎的一个权宜之计，不会有什么作为。但是 Apple 公司和 Google 公司的介入改变了 KHTML 的命运。

### 1.4.1 Apple Safari

2002 年，Apple 公司在 KHTML 的基础上开发了开源的排版引擎 WebCore，作为后来的 WebKit 内核的最重要组件，并将内核用于自己的浏览器 Safari。作为 KHTML 的分支，Apple 公司经常将 WebKit 的更新同步发送给 KHTML，虽然有时一些 Mac OS X 系统特有代码并不能被 KHTML 使用，但是 KHTML 还是借此在效率和标准兼容性上提高了不少。

WebKit 内核已经被包括 Safari 和 Chrome 在内的众多浏览器采用。由于 Apple 和 Google 等公司对 WebKit 的不断优化，以及 Safari 和 Chrome 本身在其他功能上的优秀表现，截至 2012 年 9 月，WebKit 的市场占有率已经达到了 44%，而基于 Trident 内核的 IE 浏览器则滑落到了 22%。

Safari 团队对 WebKit 的开发一直在继续。2005 年，Safari 开发者 Dave Hyatt 宣布全面开源 WebKit，而不只是 WebCore 和 JavaScriptCore；2005 年底又加入了 WebKit 对 SVG 的支持；2007 年，Webkit 实现了 CSS 扩展功能，包括动画和 2D、3D 变换，同时支持 HTML 5 的媒体特性；2009 年完成了新的 JavaScript 解析执行引擎 SquirrelFish Extreme 开发，该引擎将 JavaScript 编译为本地代码，加快其执行效率。最终，Apple 在 2010 年宣布开发 WebKit 2，以提供更高的抽象度和软件复用能力，同时，应用编程接口也不再与之前的 WebKit 兼容。

Apple 公司的 WebKit 包括三个组件：WebCore、JavaScriptCore 和 Drosera。WebCore 是遵循 LGPL 的开源排版渲染引擎，提供了 Cocoa API 和跨平台的 C++ API，WebCore 通过了 ACID2 和 ACID3 的测试；JavaScriptCore 是 WebKit 的 JavaScript 引擎，最早基于 KHTML 的

KJS 引擎和正则表达式库 PCRE，目前已被更先进的 SquirrelFish Extreme 取代；Drosera 是一个 JavaScript 调试工具，目前已经被 Web Inspector 的相关功能取代。

除了具有优秀的内核，Safari 浏览器也是一款非常优秀的产品。从 2003 年到现在经历了多个版本，不仅具备了由多标签管理、地址栏和搜索栏结合的智能地址栏和私密浏览等基本功能，还通过云计算技术与其他的 Apple 设备紧密结合，提供书签同步功能等。Safari 最突出的特点是其网页渲染速度，尤其是 JavaScript 的执行效率。在对 Safari 4 的测试中，分别在 PC 和 Mac OS X 使用 SunSpider 进行测试，结果 Safari 4 的速度是 IE 7 的 42 倍，IE 8 的 6 倍，Firefox 3 的 3.5 倍，Google Chrome 的 1.2 倍。Safari 浏览器基本信息如下表 1-2 所列。

表 1-2　Safari 浏览器基本信息

| 开发公司 | Apple 苹果公司 | 开发公司 | Apple 苹果公司 |
|---|---|---|---|
| 初始版本 | 版本号：Safari 1.0<br>2003 年 1 月 7 日 首发布版本 | 发展状态 | 支持开发中 |
| 最新版本 | 版本号：Safari 5.1<br>2011 年 7 月 20 日最新发布 | 文件大小 | 45MB |
| 操作系统 | Mac OS X、Microsoft Windows、iOS | 支持标准 | HTML 5 和 CSS 3 以及 JavaScript 新标准支持 |
| 排版引擎 | WebKit(基于 KHTML) | 官方网站 | www.apple.com/safari/ |
| 支持语言 | 5 种(Safari 5) | | |

除此之外，Safari 还有诸多优点，如用户界面美观，字体管理卓越；具备 VoiceOver 屏幕阅读器；支持 ARIA；全页面缩放；硬件加速和丰富的 CSS 3 支持。

Safari 的缺点主要是内存占用率高和页面兼容性低。此外，对于非 Apple 操作系统平台，Safari 在各方面的表现明显不如其他主流浏览器，甚至有些功能缺失。

### 1.4.2　Google Chrome

Google Chrome 是一个历史并不长的浏览器，它基于开源 Webkit 排版引擎(也借鉴了 Mozilla 的一些相关技术)，于 2008 年发布了第一版。但是，其发展迅速，每次更新都有巨大的提高。其发布首日占有率达 3%，其后一度跌至 1%。但自 2009 年开始，凭借对 Chrome 的性能和功能的不断提高，其市场占有率也不断提高。在 2010 年，Chrome 的市场占有率超越 Opera 和 Safari；在 2011 年，Chrome 超越了 Firefox；最终在 2012 年，Chrome 超越了 IE 浏览器，以超过 30%的市场占有率排名世界第一。

Chrome 提供包括了目前浏览器比较齐全的功能，包括：

安全功能：提供黑名单的定期下载，每个标签页提供单独的沙盒，以及多进程、任务管理器等。

隐私功能：提供隐身模式、无痕浏览等功能。

性能优化：全新的 JavaScript 引擎 V8，DNS 预取功能等。

人际交互：窗体自动填入、交互式智能搜索等。

自定义扩展支持：Chrome 提供类似于 Firefox 样式的扩展功能，使用户能够丰富浏览器的功能。

与 Chrome 的 WebKit 同样重要的另一个核心组件是其自主研发的称为 V8 的 JavaScript

引擎。V8 将 JavaScript 编译成机器码，而不是编译为中间的二进制字节码或者直接解析执行，由此获得性能的提升。同时，V8 在 JavaScript 对象属性创建和查找，以及垃圾收集机制上也做了不同程度的优化。根据评测，V8 执行 JavaScript 代码的效率要高于 Chakra、SpiderMonkey 和 JavaScriptCore。V8 引擎是遵循 BSD 协议的开源项目，源码可以在 Google Code 上下载。

## 1.5 Presto

虽然在 1995 年 Opera 公司成立之前，Opera 浏览器的开发就开始了，但是一直没有对外发布。1992 年左右，Opera 创始人 Jon Stephenson von Tetzchner 等人在 Televerket(挪威最大的电信公司，现为 Telenor)公司的研究小组工作，他们觉得 Mosaic 能实现的结构太单调，决定开发一个新的浏览器，即 Opera 的原型。

直到 1996 年，Opera 2 版本才发布，并支持 Windows 系统。在之后的版本中分别增加了多系统支持，CSS 支持，鼠标手势功能和 Unicode 支持等。截止到 2012 年的 11 月份，Opera 已经发布了 12 个版本，在这些版本中最具有里程碑意义的是在 2003 年的 Opera 7 中使用独立的 Presto 排版引擎作为内核，以及在 2010 年发布的 Opera 10.50 版本中引入了全新的 JavaScript 引擎 Carakan。

随着 Presto 不断地更新、不断地完善，比较重要的更新包括 Presto 2.0 中支持 XSTL 和 XPath，以及通过 ACID2；Presto 2.2.15 支持 CSS 选择器 API，ACID3 测试达到满分；在 Presto 2.5 中引入 HTML 5，并在 2.6 和 2.7 版本中完善对 HTML 5 的支持，例如 WebM、地理定位、Web Workers 和 WebSockets 等；在较近的版本 Presto 2.10.289 中支持 WebGL 和硬件加速。

Opera 将 JavaScript 引擎作为 Presto 的一个组件，虽然有代号，但是没有独立发布，也没有版本号。在目前使用的 JavaScript 引擎 Carakan 之前，Presto 1.0 和 Presto 2.0 使用的 JavaScript 引擎称为 Linear 2，Presto 2.1 到 Presto 2.4 版本使用的 JavaScript 引擎称为 Futhark，从 Presto 2.5.24 之后则使用 Carakan。

借助 Linear 2 和 Futhark 引擎，Presto 曾经一度被称为世界上最快的排版引擎。但是，随着 Google V8，Mozilla TraceMonkey 和 Apple SquirrelFish 分别引入机器码语言生成机制，使得 Presto 的 JavaScript 引擎显得没有什么优势。为了改变这一不利局面，Opera 在新的 Carakan 引擎中加入了相比之前很多新的特性，比如采用基于寄存器的位元组码机制、机器语言生成、自动对象分类，以及整体的性能优化。

作为主流浏览器之一，Opera 有其自身的众多优良特性：

页面加载速度快：借助于页面缓存机制，Carakan 引擎和单独的 Vega 矢量图形库，使 Opera 具有快速的网页加载速度和图形渲染速度。

可定制：用户可以通过编辑.ini 文件定义浏览器界面，并提供 UserJS 和 UserCSS 让高级用户改变界面样式。

Turbo 功能：主要针对手机上网功能，预先在 Opera 提供的服务器进行资源压缩，以节省带宽，增加下载速度。

当然，除了这些优点，也存在着一些对 Opera 浏览器的质疑：

网页兼容性低：由于 Opera 严格遵守 W3C 标准，不支持其他扩展标准，因此，以 IE 或 Firefox 为假设的应用可能在 Opera 显示出现问题。

易用性低：Opera 使用脚本定制界面，对普通用户来说操作较为复杂。

内存占用率高：Opera 在打开较少标签的情况下内存占用较其他浏览器高。

可以看出，易用性低与高可定制很难调和，高可定制必然会带来更为复杂的软件设置。

# 第 2 章　浏览器体系结构

## 2.1　浏览器的组成

浏览器一般由用户接口(Shell)和浏览器内核两部分组成，内核又包括页面排版引擎(渲染引擎)和脚本执行引擎(脚本引擎)。其中渲染引擎负责页面中标签解析与样式渲染，脚本引擎负责脚本(如 JavaScript、VBScript 等)的语法分析与解释执行，如图 2-1 所示。

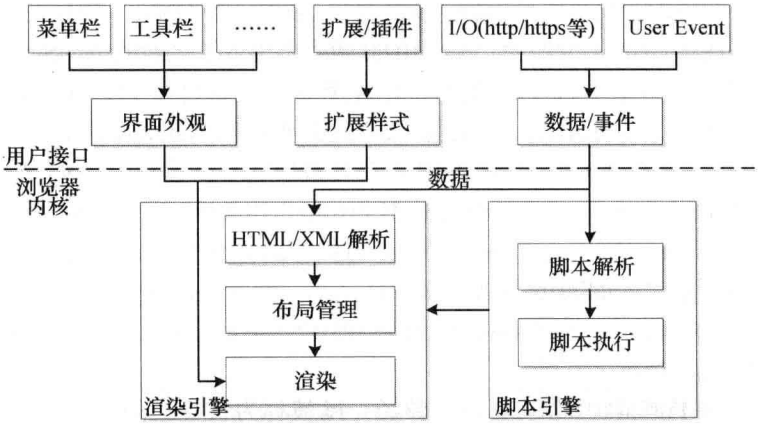


图 2-1　浏览器组成结构示意图

一些浏览器还包含动态组件模块来支持运行时添加第三方的扩展和插件，实现自定义的功能，比如 Firefox 的 XPCOM 组件模型。有时，这种动态组件模型相当复杂，于是单独拿出来作为一种应用编程技术，比如 IE 浏览器的 ActiveX 技术和 Trident 编程需要使用 Windows 的组件对象模型(COM)技术。

浏览器内核可以分成两部分：渲染引擎和脚本引擎。渲染引擎负责取得网页的内容(如 HTML、XML、图像、脚本等)进行信息数据整理，计算网页的显示方式(如页面布局等)，然后会调用系统接口函数绘图输出至显示器或打印机。所有网页浏览器、电子邮件客户端以及其他需要编辑、显示网络内容的应用程序都需要页面渲染引擎[1]。脚本引擎主要指 JS 引擎(JavaScript 引擎)，用来实现脚本语言的解析和执行，以及网页的动态效果。

浏览器发展初期，渲染引擎和脚本引擎并没有明确区分，随着脚本引擎越来越独立，内核就倾向于只指渲染引擎，目前使用互联网标准工程组织(WaSP)的 ACID 标准来测试脚本引擎的兼容性和性能。内核种类很多，常见的有以下四种：Trident，Gecko，Presto，WebKit。渲染引擎和 JS 脚本引擎分别如表 2-1 和表 2-2 所列。