WebKit 研究报告

侯 炯

二零零八年十二月二十六

目　录

一. Webkit 介绍

WebKit 的前身是 KDE 小组的 KHTML。Apple 将 KHTML 发扬光大，推出了装备 KHTML 改进型的 WebKit 引擎的浏览器 Safari，获得了非常好的反响。

WebKit 内核在手机上的应用十分广泛，例如 Google 的手机 Gphone、 Apple 的 iPhone, Nokia’s Series 60 browser 等所使用的 Browser 内核引擎，都是基于 WebKit。

现在浏览器的内核引擎，基本上是三分天下:

Trident: IE 以 Trident 作为内核引擎。

Gecko: Firefox 是基于 Gecko 开发。

WebKit: Safari, Google Chrome 基于 Webkit 开发。

WebKit 支持功能:

HTML4.0/5.0

CSS1&2

Dom1&2

ECMA262

JS1-6

HTTP/FILE

GIF/JPEG/PNG

XML

SSL3

JVM

FTP

SVG 可缩放矢量图形（Scalable Vector Graphics）

RSS2.0(RDF Site Summary)

浏览器的选型:

1。Geocko 功能强，但太庞大

2。Opera 功能强，但要钱

3。Ipanel 功能一般，也要钱

4。Ants 功能一般，还是要钱

5。WebKit 功能强，不要钱，呵呵就这个（有 nokia,apple,google 为例），它属于 LGPL and BSD licenses.

二. Webkit 编译详解

1.依赖库及介绍

libicu-dev

ICU 是一个成熟，广泛使用的一套为 C / C + +和 Java 库提供 Unicode 的全球化支持软件。ICU 广泛的应用在便携式设备上，并给出相同的结果在所有平台之间的 C / C + +和 Java 软件。

libxslt-dev

XSLT 的英文标准名称为 eXtensible Stylesheet Language Transformation 。根据 W3C 的规范说明书(http://www.w3.org/TR/xslt)，最早设计 XSLT 的用意是帮助 XML 文档(document)转换为其它文档。但是随着发展，XSLT 已不仅仅用于将 XML 转换为 HTML 或其它文本格式，更全面的定义应该是： XSLT 是一种用来转换 XML 文档结构的语言。

libcurl-dev

cURL 是一个利用 URL 语法的文件传输工具,是基于 libcurl 的前端命令行工具。它支持很多协议： FTP, FTPS, HTTP, HTTPS, GOPHER, TELNET, DICT, FILE 以及 LDAP。它同样支持 HTTPS 认证, HTTP POST 方法, HTTP PUT 方法, FTP 上传, kerberos 认证, HTTP 上传, 代理服务器, cookies, 用户名/密码认证, 下载文件断点续传, 上载文件断点续传, http 代理服务器管道（ proxy tunneling），甚至它还支持 IPv6, socks5 代理服务器,通过 http 代理服务器上传文件到 FTP 服务器等等，功能十分强大。

libsqlite3-dev

SQLite 是实现了 SQL 92 标准的一个大子集的嵌入式数据库.其以在一个库中组合了数据库引擎和接口,能将所有数据存储于单个文件中而著名.功能一定程度上居于 MySQL 和 PostgreSQL 之间.尽管如此,在性能上面,SQLite 常常快 2-3 倍 (甚至更多).这利益于其高度调整了的内部架构,因为它除去了服务器端到客户端和客户端到服务器端的通信.

libjpeg62-dev

libjpeg 软件包包含 jpeg 库. 这些库使图形文件在联合图象专家组的标准上压缩. 它是一种"有损耗"的压缩算法.

libpng12-dev

libpng 软件包包含 libpng 库.这些库被其他程序用于读写 png 文件

gperf

'gperf'是一个用 C++编写的完美的 hash 函数生成器.它通过一个完美的 hash 函数 F 转换一个含有 N 元素的用户特定关键字集合到集合 W. F 唯一映射关键字到 W 的 0..K 范围,其中 K>=N 如果 K=N 那么 F 就是最小化的完美 hash 函数.'gperf'生成一个 0..K 元素的静态查找表和一对 C 函数.这些函数决定一个给定的字符串 S 是否在集合 W 中,通过只多一次的查找.

'gperf'普遍用于为多个商业编译器,研究型编译器,语言处理工具的词法分析器生成一个关键字识别器.这些编译器包括 GNU C, GNU

C++, GNU Pascal, GNU Modula 3, 和 GNU indent.完整的'gperf'C++源代码可以通过匿名 ftp`ics.uci.edu' 和 `ftp.santafe.edu'得到.'gperf'已经随 GNU libg++一起发布好几年了

flex

快速词法分析器发生器

Bison

GNU 项目分析器

GUI

支持多种 GUI,包括 gtk, qt, mac, win 等.linux, windows, Mac 都能运行。

2.X11+Gtk+WebKit 交叉编译详解

编译列表:

Name	Version
libpng	1.2.10
pkg-config	0.23
gperf	3.0.3
Tiff	3.8.2
libjpeg	6b
freetype	2.1.10
libxml2	2.6.30
fontconfig	2.4.2

XFree86	4.7.0
glib	2.18.0
atk	1.20.0
cairo	1.2.0
pango	1.20.0
gtk+	2.14.0
ICU	4c-3_6
xslt	1.1.22
curl	7.19.2
sqlite	3.5.6
WebKit	r29711

The cross source:

```
export ARCH=arm
export CC= iwmmxt_le-gcc
export CXX=iwmmxt_le-gcc
export BUILD_ROOT=/uplatform
export PATH=/opt/montavista/cee/devkit/arm/iwmmxt_le/bin/:/opt/montavista/cee/host/bin/:$BUILD_ROOT/usr/bin:$PATH
export PKG_CONFIG_PATH=$BUILD_ROOT/usr/lib/pkgconfig:$BUILD_ROOT/usr/X11R6/lib/pkgconfig
```

说明：CC 是 c 的编译器选项，CXX 是 C++的编译器选项。BUILD_ROOT 是要 build 的路径。PKG_CONFIG_PATH，现在大多数软件，都是通过 pkg-config 来检查依赖关系的，所以要把两者都加到 PKG_CONFIG_PATH 环境变量中

Libpng

```
etho ac_cv_fnuc_malloc_0_nonull=yes>> $ARCH-linux.cache
./configure --prefix=/$BUILD_ROOT/usr --host=$ARCH-linux
make &&make install
```

pkg-config

```
./configure
make &&make install
```

应为需要较高级的版本的 pkg-config 才能编译 glib 和 gtk，如果编译机上的版本够高就不需要在升级 pkg-config 了

Gperf

```
./configure
make &&make install
```

企业版的/usr/bin 下有，可以直接使用，如果没有需要编译一个

Tiff

```
./configure --prefix=/$BUILD_ROOT/usr --host=$ARCH-linux
make &&make install
```

字体读取文件库

Libjpeg

```
./configure --prefix=/$BUILD_ROOT/usr --host=$ARCH-linux
make &&make install
```

freetype

```
./configure --prefix=/$BUILD_ROOT/usr --host=$ARCH-linux
make &&make install
```

libxml2

```
./configure --prefix=/$BUILD_ROOT/usr --host=$ARCH-linux
make &&make install
```

fontconfig

```
./configure --prefix=/$BUILD_ROOT/usr --host=$ARCH-linux
make &&make install
```

XFree86

```
./configure --prefix=/$BUILD_ROOT/usr --host=$ARCH-linux
```

由于 x11 的需求不同，在配置选项的时候要加特定的参数。所以需要知己 configure h 看所需要的选项

Glib

```
ac_cv_type_long_long=yes
glib_cv_long_long_format=ll
glib_cv_stack_grows=no
glib_cv_uscore=no
ac_cv_func_posix_getpwuid_r=yes
./configure --host=arm-linux --prefix=$BUILD_ROOT/usr
make &&make install
```

以上选项加入到 configure 里再 configure,应为我们 arm 上裁剪的是 2.4 内核，所以有些是不支持的，需要屏蔽

Atk

```
export CFLAGS="`pkg-config --cflags glib-2.0` -I$BUILD_ROOT/usr/include"
export LDFLAGS="-L$BUILD_ROOT/usr/lib,-rpath=$BUILD_ROOT/usr/lib"
./configure --host=arm-linux --prefix=$BUILD_ROOT/usr
make &&make install
```

因为 atk 是依赖 glib 的，所以需要指定 glib

Cairo

```
Export CFLAGS="-I$BUILD_ROOT/usr/include -I$BUILD_ROOT/usr/X11R6/include"
Export LDFLAGS="-L$BUILD_ROOT/usr/lib -L$BUILD_ROOT/usr/X11R6/lib -lXft
-lfreetype -lfontconfig -lXrender -lexpat -lXext -lX11"
./configure --host=arm-linux --disable-nls --with-x --prefix=$BUILD_ROOT/usr
--x-includes=$BUILD_ROOT/usr/X11R6/include
--x-libraries=$BUILD_ROOT/usr/X11R6/lib
make &&make install
```

由于我们编译的是 1.2.0 所有不需要编译 pixman

Pango

```
export
FREETYPE_CONFIG=$BUILD_ROOT/usr/bin/freetype-config
Export
LDFLAGS="-L$BUILD_ROOT/usr/lib -Wl,-rpath=$BUILD_ROOT/usr/lib"
export
CFLAGS="`pkg-config --cflags glib-2.0 cairo` -I$BUILD_ROOT/usr/include/freetype2"
```

```
./configure --host=arm-linux --disable-nls --with-x --prefix=$BUILD_ROOT/usr
--x-includes=$BUILD_ROOT/usr/X11R6/include
--x-libraries=$BUILD_ROOT/usr/X11R6/lib
make &&make install
```

因为 pango 依赖 GLib, freetype,cairo,x11，所以要指明其连接

gtk+

首先屏蔽 config.h 里面的

HAVE_XFIXES 0

HAVE_CUPS_API_1_2

再 configure

```
CFLAGS="`pkg-config --cflags pango pangoft2 pangocairo` -I$BUILD_ROOT/usr/include
-I$BUILD_ROOT/usr/include/freetype2 -I$BUILD_ROOT/usr/X11R6/include/"
LDFLAGS="`pkg-config --libs pango pangoft2 pangocairo` -L$BUILD_ROOT/usr/lib
-lintl -L$BUILD_ROOT/usr/X11R6/lib -lXft -lfreetype -lfontconfig -lXrender -lexpat
-lXext -lX11" CC=iwmmxt_le-gcc
./configure --disable-cups --disable-nls --enable-xim=no --disable-xim-inst
--with-xinput=no
make &&make install
```

ICU

```
./configure --enable-static --enable-shared --host=arm-linux
--prefix=$BUILD_ROOT/usr/
```

Xslt

```
./configure --host=arm-linux --with-x --prefix=$BUILD_ROOT/usr
--x-includes=$BUILD_ROOT/usr/X11R6/include
--x-libraries=$BUILD_ROOT/usr/X11R6/lib
```

Curl

```
ac_cv_file___dev_urandom_=yes
./configure --host=arm-linux --prefix=$BUILD_ROOT/usr/ -without-ssl
--without-ca-path --without-ca-bundle
```

在 configure 文件里加 ac_cv_file___dev_urandom_=yes，再 configure

Sqlite

```
./configure --host=arm-linux --prefix=$BUILD_ROOT/usr
make &&make install
```

WebKit

首先执行

sh autogen.sh

执行 autogen 脚本，会生成 configure，再执行 configure

```
CFLAGS=-I$BUILD_ROOT/usr/include  LDFLAGS=-L$BUILD_ROOT/usr/lib  ./configure
--prefix=$BUILD_ROOT/usr  --host=$ARCH-linux  CFLAGS=-I$BUILD_ROOT/usr/include
CPPFLAGS=-I$BUILD_ROOT/usr/include icu_config=$BUILD_ROOT/usr/
```

3. 编译出错 Q&A:

Q: 在 configure 中报找不到某个库，或某个库的版本太低。

A: 1 确认是否有该库，版本是否太低。如果没有请编译该库

2 有该库文件，证明没有正确的连接到在 CFLAGS 和 LDFLAGS 两个环境变量中指明连接的头文件和库。

3 只明了路径为什么还不行？有些要通过 pkg-config 读取.pc 文件的形式才能正确。

Q: 在 configure 中什么找不到，或一些莫名奇妙的错误。

A: 直接打开 configure 文件，找到报错的地方，找到其判断语句并将起屏蔽。如 2.4 的内核不支持 stack_grows，在编译 glib 的时候就要加 glib_cv_stack_grows=no，让其屏蔽过去。

Q: 在 make 的时候包语法错误

A: 找到该文件，直接修改其语法错误。应为是开源软件，有语法错误很正常。

Q: 在 make 的时候出现找不到某库文件

A: 直接在 Makefile 的 LDFLAGS 加其库连接选项，如果本来就不要该库就直接删除。

Q: 在 make 的时候找不到某函数。

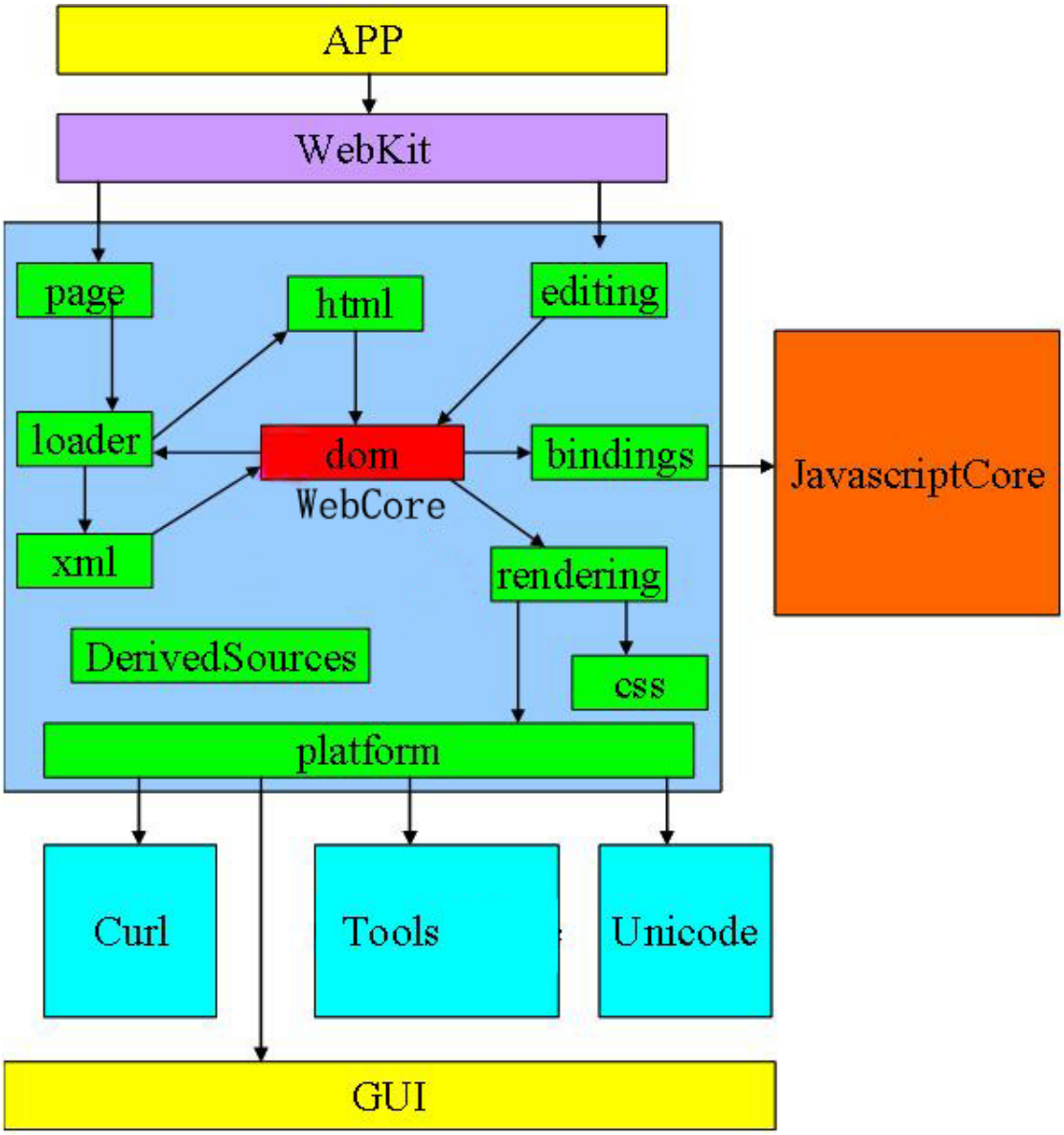
A: 在 config.h 里屏蔽其函数对应的宏。一般来宏来控制某个编译选项，有些函数是通过宏来判断是否执行的。

Q: 其他问题

A: 请 google 一下。哈哈!

三. WebKit 分析

1.体系结构



WebKit 上层组织应用

WebCore

··Page 与外框相关的内容(Frame, Page, History, Focus, Window)

··Loader 加载资源及 Cache

··HTML-DOM HTML 内容及解析

··DOM- DOM CORE 内容

··XML- XML 内容及解析

··Render-排版功能

··CSS-DOM CSS 内容

··Binding-DOM 与 JavascriptCore 绑定的功能

··Editing-所有与编辑相关的功能

JavascriptCore-javascript 引擎

··API-基本 javascript 功能

··Binding 与其它功能绑定的功能,如:DOM,C,JNI

··DerviedSource 自动产生的代码

··ForwordHeads 头文件,无实际意义

··PCRE-Perl-Compatible Regular Expressions

··KJS-Javascript Kernel

··WTF-KDE 的 C++模板库

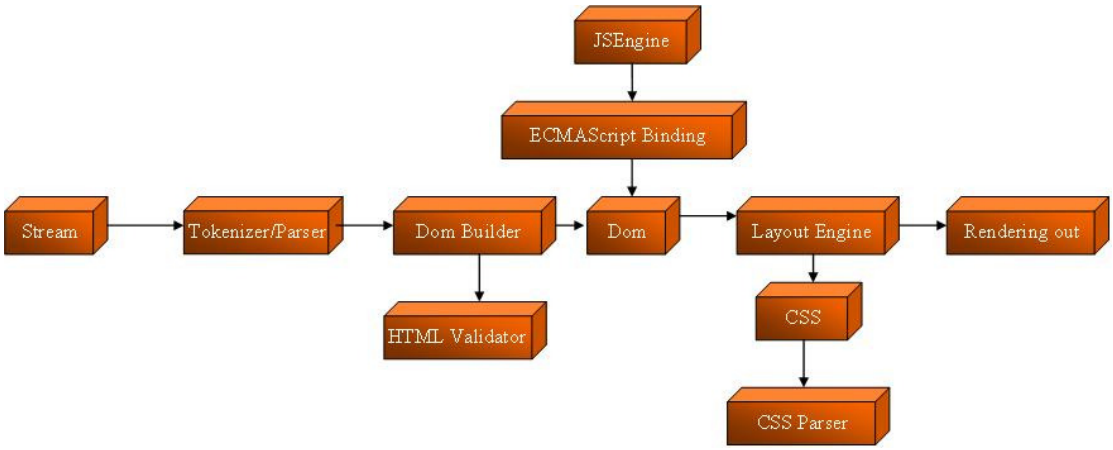
Unicode unicode 库

Tools tools 库

CURL-url 客户端传输库

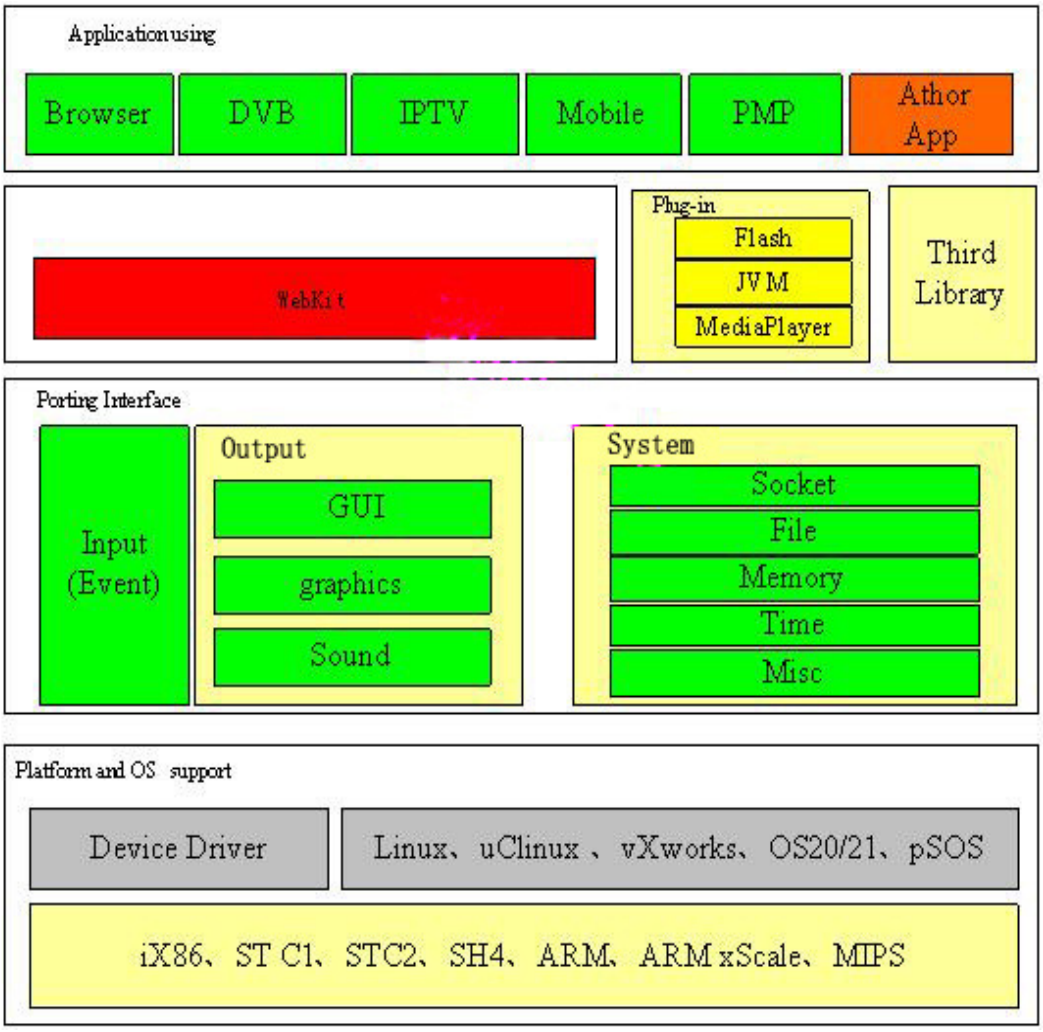
PlatForm- 与平台相关的功能,如图形图像,字体,Unicode, IO,输入法等.

2.解析流程



1. CURL 获得网站的 stream
2. 解析划分字符串
3. 通过 Dom Builder 按合法的 html 规范生成 Dom 树
4. 如果有 javascript，JSEngine 就通过 ECMA-262 标准完善 Dom 树
5. 把 Dom 传给 LayoutEngine,进行布局，如果有 CSS 样式，就通过 CSSParser 解析。
6. 最后 Rendering out 出来

3. 浏览器系统结构



浏览器系统结构图分为四层

第一层为业务应用层，用户可在 webkit 基础上构建各种应用。

第二层为插件及第三方库。

第三层为 webkit 平台 Porting 所需的接口层。

第四层为平台和操作系统支持层，提供 webkit 平台所需的软硬件资源。

四. 各种浏览器比较分析

1. 网页浏览器列表

引擎	网页浏览器
Trident	Internet Explorer，傲游，世界之窗浏览器，Avant，腾讯 TT， Netscape，NetCaptor，Sleipnir，GOSURF，GreenBrowser，KKman
Gecko	Fennec,Firefox,网景(6至9), SeaMonkey,Camino,Flock,Galeon,K-Meleon,Minimo,Mozilla,Sleipnir,Songbird ,XeroBank
KHTML 或 WebKit 框架	Safari,Konqueror,Epiphany,Google Chrome,iCab,OmniWeb,Midori,Shiira
Presto	Opera,任天堂 DS 浏览器
Java	HotJava,Opera Mini,UCWEB
Tasman	Internet Explorer for Mac,MSN for Mac OS X
文字界面	Lynx,Links,w3m
嵌入式系统	Internet Explorer Mobile, Minimo,Opera Mobile ,PSP 浏览器
其它	Amaya ， Dillo ,Mosaic

Trident

又称为 MSHTML，是微软的视窗操作系统（Windows）搭载的网页浏览器—Internet Explorer 的排版引擎的名称，它的第一个版本随着 1997 年 10 月 Internet Explorer 第四版释出，之后不断的加入新的技术并随着新版本的 Internet Explorer 释出。在最新的 Internet Explorer 第七版中，微软将对 Trident 排版引擎做了的重大的变动，除了加入新的技术之外，并增加对网页标准的支持。尽管这些变动已经在相当大的程度上落后了其它的排版引擎，如 Gecko、WebCore、KHTML 及 Presto。

Gecko

是套开放源代码的、以 C++编写的网页排版引擎。目前为 Mozilla 家族网页浏览器以及 Netscape 6 以后版本浏览器所使用。这软件原本是由网景通讯公司开发的，现在则由 Mozilla 基金会维护。这套排版引擎提供了一个丰富的程序界面以供互联网相关的应用程式使用，例如网页浏览器、HTML 编辑器、客户端/服务器等等。虽然最初的主要对象是 Mozilla 的衍生产品，如 Netscape 和 Mozilla Firefox，现在已有很多其他软件现在利用这个排版引擎。Gecko 是跨平台的，能在 Microsoft Windows、Linux 和 Mac OS X 等主要操作系统上运行。

KHTML

KDE 系统自 KDE2 版起，在 KDE 的新程式 Konqueror 的网页浏览器使用了 KHTML 引擎。该引擎以 C++编程语言所写，并以 LGPL 授权，支援大多数网页浏览标准。由于微软的 Internet Explorer 的占有率相

当高，不少以 FrontPage 制作的网页均包含只有 IE 才能读取的非标准语法，为了使 KHTML 引擎可呈现的网页达到最多，部分 IE 专属的语法也一并支援。KHTML 拥有速度快捷的优点，但对错误语法的容忍度则比 Mozilla 产品所使用的 Gecko 引擎小。

Presto

是一个由 Opera Software 开发的浏览器排版引擎，供 Opera 7.0－9.60 版使用。Presto 取代了旧版 Opera 4 至 6 版本使用的 Elektra 排版引擎，包括加入动态功能，例如网页或其部分可随着 DOM 及 Script 语法的事件而重新排版。Presto 在推出后不断有更新版本推出，使不少错误得以修正，以及阅读 Javascript 效能得以最佳化。

Tasman

是微软的 Internet Explorer for Mac 浏览器所使用的排版引擎，也是为尝试支援 W3C 所制定的网页标准而设计的。在 Tasman 推出时，一度是最切合 HTML 及 CSS 等标准的排版引擎。现时微软方面也停止为 Internet Explorer for Mac 提供支援，但新版本的 Tasman 引擎仍被应用在一些微软产品上

2. 浏览参数性能比较

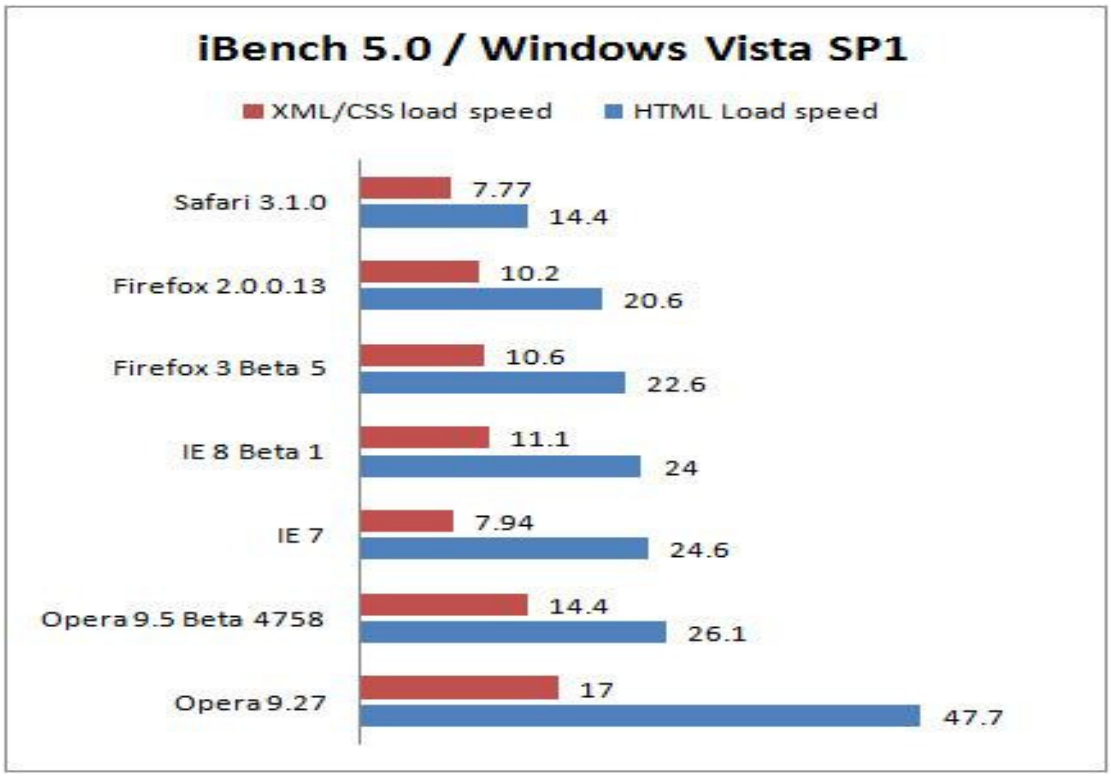
2008 年 ZDnet 用了 7 页的报告比较了世界上最流行的 4 个浏览器: IE, Firefox, Opera, Safari。它使用了 iBench 和 SunSpider 作性能基准测试软件。这里我们摘取其部分有用数据，进行比较。

iBench

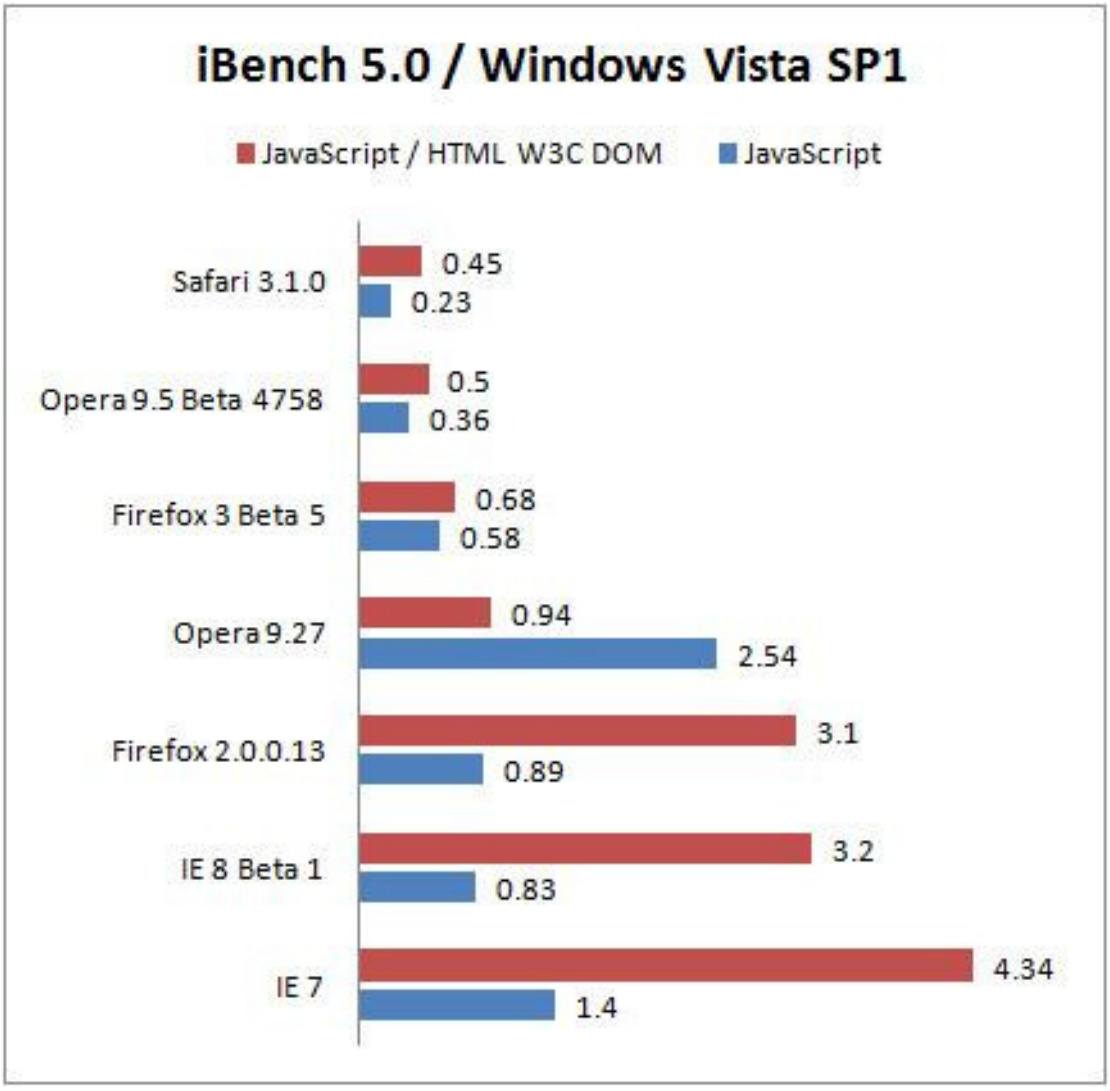
是一款应用广泛、跨平台基准测试程序，它能够检测各种 Web 客户端应用网络新技术时的所表现出来的性能数据。这里的 Web 客户端是指任何用于从网上获取信息的硬件和软件，不论使用 T1 连接互联网 Macintosh 电脑、无线掌上设备、运行 Linux 接收 Web TV 的台式机都在其中。程序将对这些设备进行一系列测试，得出易用特性和网速对应用的影响程度。

SunSpider

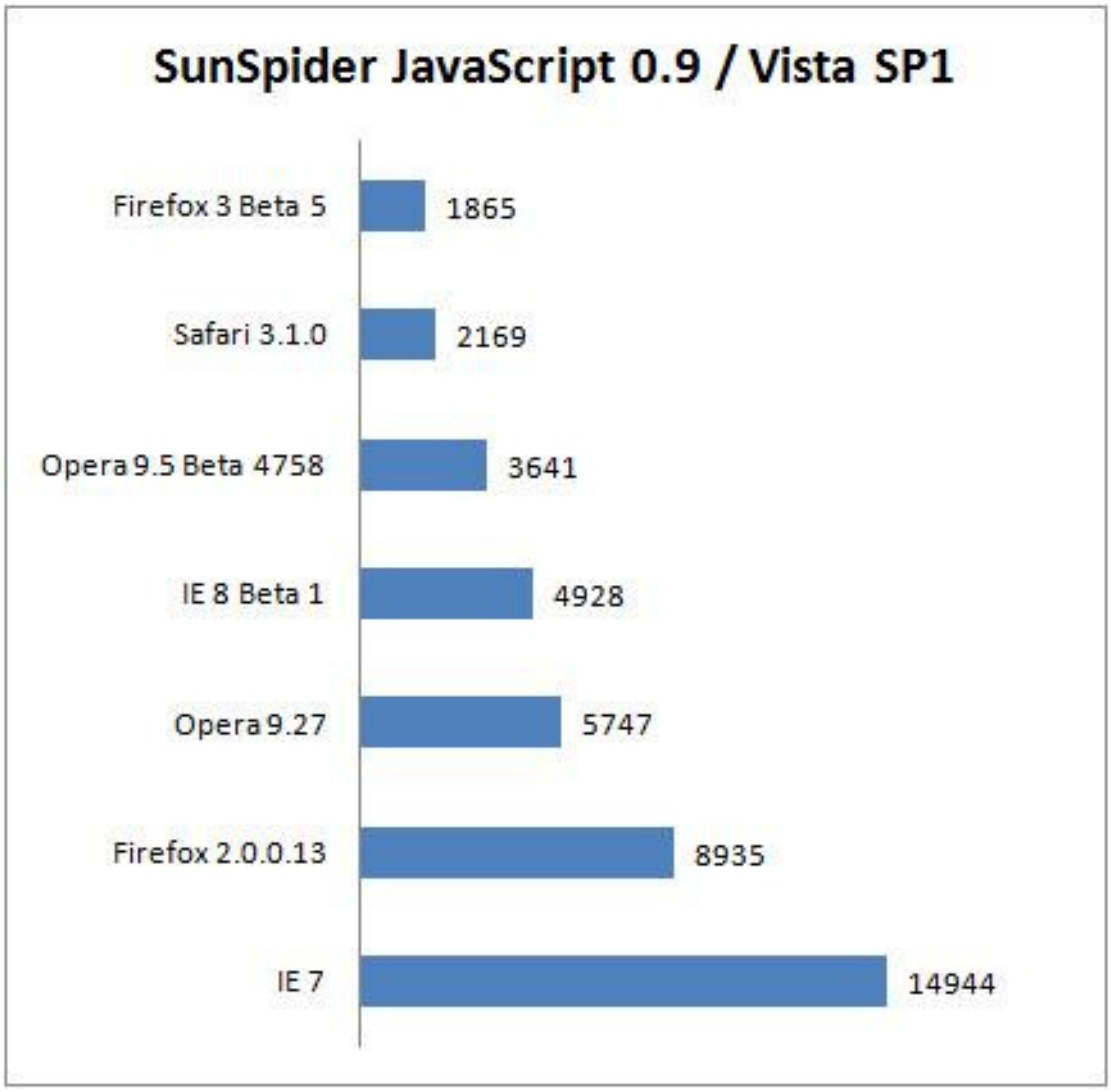
一个 JavaScript 的基准。这个基准测试核心的 JavaScript 语言，并非测试 DOM 或其他浏览器的 API 的 。它的目的是比较不同版本的同一浏览器，以及不同的浏览器之间的比较。是广泛使用的 JavaScript 的基准。



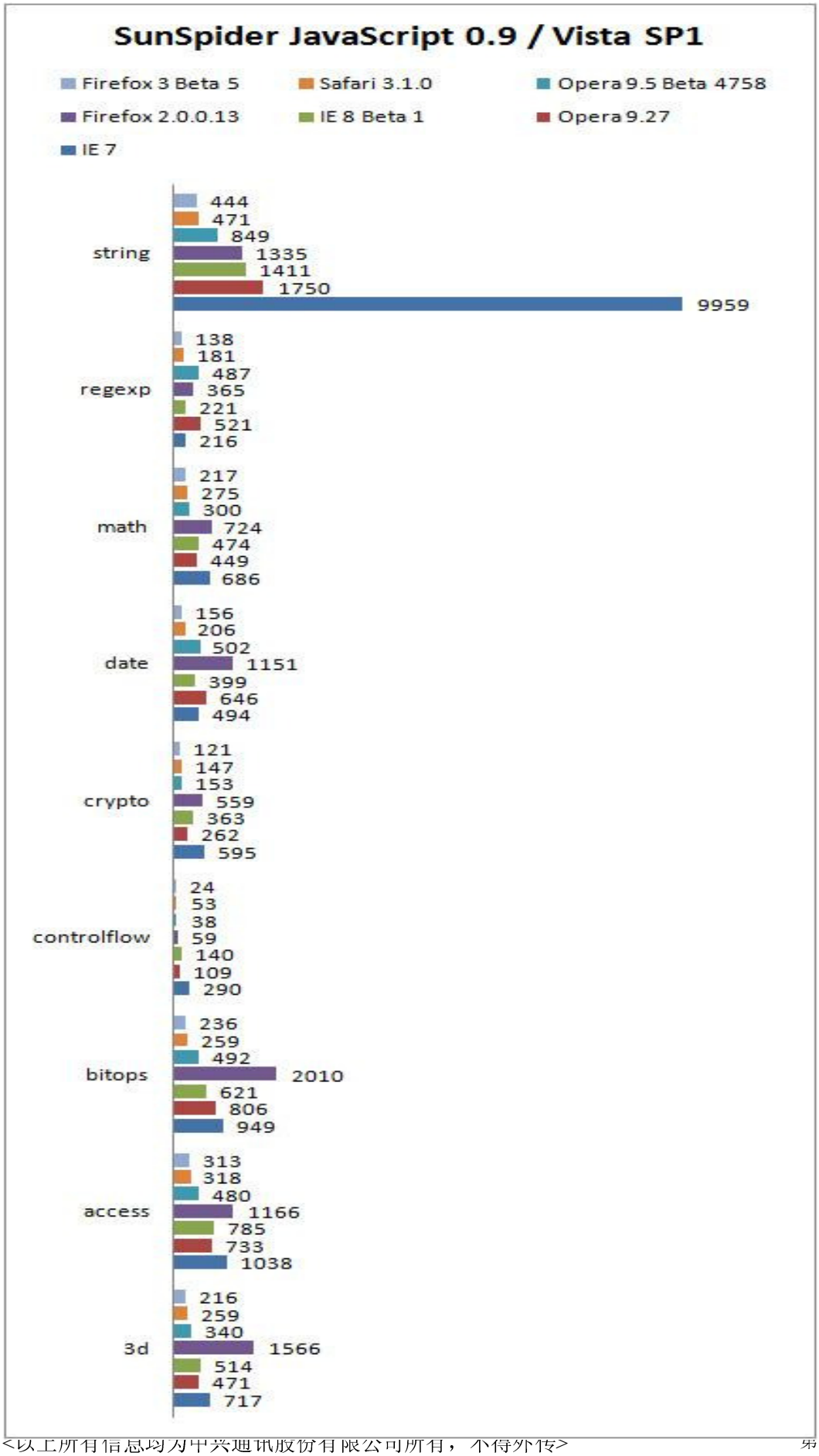
iBench5.0的XML/CSS和HTML的下载结果可以看出Safari和Firefox表现都不错，IE和Opera表现其次。



iBench5.0的JavaScript的测试性能上看,Safari,opera和Firefox都表现不错，IE明显落后。



以上为 SunSpider 测试出的综合指标，越小越好。明显可以看出，Firefox 和 Safari 表现比较好，Opera 其实，IE 比较落后。一下是 SunSpider 测试个参数详细列表:



3. 浏览器使用率分析

全球浏览器使用分布

日期	Internet Explorer	Firefox	Opera	Safari
Q4 2008	70.53%	20.35%	0.73%	6.84%
Q3 2008	72.22%	19.48%	0.71%	6.39%
Q2 2008	73.81%	18.43%	0.71%	6.14%
Q1 2008	75.06%	17.35%	0.67%	5.78%
Q4 2007	77.37%	15.84%	0.62%	5.24%
Q3 2007	78.85%	14.69%	0.51%	4.80%
Q2 2007	78.76%	14.92%	0.46%	4.66%
Q1 2007	79.38%	14.35%	0.50%	4.70%
Q4 2006	80.69%	13.50%	0.56%	4.06%
Q3 2006	82.88%	11.89%	0.62%	3.30%
Q2 2006	84.03%	10.67%	0.57%	3.25%
Q1 2006	85.01%	9.77%	0.53%	3.10%
Q4 2005	85.88%	9.00%	0.54%	2.80%
Q3 2005	86.74%	7.97%	0.55%	2.24%
Q2 2005	87.24%	8.08%	0.52%	1.89%
Q1 2005	89.02%	6.17%	0.49%	1.70%
Q4 2004	91.35%	3.66%	0.51%	1.50%
增长率:	-27.91%	229.82%	43.13%	356%

（由 TheCounter.com Global Statistics 所提供的资料）

4.WebKit 的 SWOT 分析

优势（Strengths）

通过测试结果 WebKit 的性能优越是其最大的优势，再加上其开源项目。性能优越加开源这两大优势，是目前很多嵌入式产品选择他的原因。

劣势（Weaknesses）

因为开源，代码并不是最新的，都是 Apple 和 KDE 的更新产品。代码有滞后性。开发周期较长和人力投入比较大，毕竟代码量大，复杂度高。

机会（Opportunities）

随着通讯技术的发展，以后的手持设备基本都配有上网功能。但目前嵌入式设备上的浏览器还是只有竞争时期，这是一个很好的机遇。如果在做出优秀的适应嵌入式的浏览器，将能成为这一浪潮的弄潮儿。

威胁（ Threats）

大蛋糕大家都想要，腾讯，google 等都开始推出自己的浏览器。嵌入式方面 Opera，Access 等公司都在努力扩张直接的市场，一些大公司也直接晚上 WebKit 推到直接的平台上，Apple,Android,NOKI 等都在自己的手机上推出 WebKit 的浏览器，可见其竞争是非常激烈度。

五. 浏览器的未来

1.微软的梦魇

基于 WebKit 引擎的 Safari for Win 已经推出，我们可以发现 Safari 除了推广 WebKit 引擎之外，实际上有着大得多的野心。

Safari，如果只是为了推广 WebKit 引擎的话，Win 平台上已经有了 Swift 可用，Safari 只能说是加强推广力度罢了，顶多算 是让 Windows 用户尝尝正宗的苹果味。可我们看看 Safari 安装文件将近 30M 的夸张身材，就会意识到 Apple 肯定搞了小动作。打开 Safari，我们很快就能发现它实现了很多非 Win 的 UI 特性：动画卷轴菜单，Mac 风格的次像素字体渲染，网页对象或标签拖动时显示的半透明缩略图等 等。再进入它的安装目录，一切真相大白！

原来 Safari 不仅引入了 WebKit 引擎，更是一举捆绑了 Core Foundation(OS X 的系统级 C 语言 API) ,CFnetwork (OS X 的网络接口 API), Core Graphics(就是 Quartz 2D, 一个矢量构图框架，OS X 图形界面的基石)，当然还不忘对于 Bonjour 更完善的应用支持。

可以这样说,如果在 iTunes 上 Apple 还有所保留的话,在 Safari 上它就表现得相当咄咄逼人了,它几乎把 OS X 一半的重要 Framework 都照搬到了 Windows 上!!! 目的我想应该是再明显不过了吧。你可以想象将来的 Wintel 机 器看起来跑的是 Vista 但实际上 OS X API 一个不缺吗？你可以想象将来程序员只需要使用 Xcode 编程就可以让程序原生运行在 OS X/Linux/Windows 上吗？我们说的不是 Java 虚拟机或者什么 Sandbox，我们说的是性能无损的 Objective-C Binary!

Safari 下一版即将引入本地 SQLite 支持，不难想象 iTunes 也开始采用 WebKit 引擎并将目前是 XML 格式的 Library 转用更强大的 SQLite 存储，再下一步呢？Core Animation 库取代 Direct 3D 来支援 CoverFlow？再下一步呢？ Cocoa Universal for Windows？Maybe？ Why not？这种环环相扣的应用带来了巨大的压迫感。渗透，渗透，再渗透！Apple 将会已浏览器为跳板，取代 Windows 成为真正的平台。

2. 云端技术的发展

云计算（cloud computing），是分布式计算技术的一种，其最基本的概念，是透过网络将庞大的计算处理程序自动分拆成无数个较小的子程序，再交由多部服务器所组成的庞大系统经搜寻、计算分析之后将处理结果回传给用户。透过这项技术，网络服务提供者可以在数秒之内，达成处理数以千万计甚至亿计的信息，达到和“超级计算机”同样强大效能的网络服务。

最简单的云计算技术在网络服务中已经随处可见，例如搜寻引擎、网络信箱等，使用者只要输入简单指令即能得到大量信息。未来如手机、GPS 等行动装置都可以透过云计算技术，发展出更多的应用服务。

云端储存（cloud storage）是一种将数据保存在虚拟服务器上的数据类型，通常意义上，数据存储在第三方媒介，而非特定单一服务器上。

3. 浏览器的未来

浏览器未来可望取代目前的操作平台将成为趋势。随着计算机、手机及连网装置也普及，未来终端运算都会在云端执行。目前计算机用户有 9 成的行为是在网络或靠着浏览器就可以完成，未来可能会再进一步提升到 95％或更高。人们拥有一个强大功能的浏览器，就能满足平时工作生活的需要。在此情况下，浏览器就是未来的操作平台系统。

参考:

1. 基维百科