HZ BOOKS
华章IT

经典畅销书系“深入理解Android”系列专题卷第3部，前百度移动浏览器业务部资深工程师撰写

依托Android源代码，根据WebKit处理数据的流程和它的功能模块，对WebKit的工作机制和实现原理进行了深度的剖析

移动开发

孟德国 王耀龙 周金利 黎欢◎著

Understanding Android Internals: WebKit

深入理解Android

WebKit卷

机械工业出版社
China Machine Press

移动开发

Understanding Android Internals: WebKit

深入理解Android

WebKit卷

孟德国 王耀龙 周金利 黎欢◎著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

深入理解 Android：WebKit 卷 / 孟德国等著 . —北京：机械工业出版社，2016.3
（移动开发）

ISBN 978-7-111-52921-7

I. 深… II. 孟… III. ① 移动终端 - 应用程序 - 程序设计 ② 网页制作工具 - 程序设计
IV. ① TN929.53 ② TP393.092

中国版本图书馆 CIP 数据核字（2016）第 028781 号

深入理解 Android：WebKit 卷

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：姜 影 高婧雅 责任校对：董纪丽
印 刷：北京市荣盛彩色印刷有限公司 版 次：2016 年 3 月第 1 版第 1 次印刷
开 本：186mm × 240mm 1/16 印 张：15.5
书 号：ISBN 978-7-111-52921-7 定 价：59.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzit@hzbook.com

Forword 序

2011 年 12 月我就开始关注 Android 领域中一个技术难度极富挑战性的部分——WebKit。这个名字听起来如雷贯耳的东西就是当前绝大部分浏览器的核心引擎。学习 WebKit 的过程注定是艰难困苦的，因为在这个领域一直没有系统成型的书籍，仅有的资料也是零碎片段。当年为了学习它，我甚至在 Windows 上编译了 WebKit，并做了一个简单的浏览器程序，然后试图利用 Visual Studio 调试这个浏览器来一窥 WebKit 的精髓。

学习 WebKit 的窘境一直延续，直到我们结识了当时还在百度从事浏览器开发的孟德国、王耀龙、周金利、黎欢四位兄弟。这是四个敢于挑战，重视承诺的年轻人。他们历时 4 年，历经跳槽，换岗，四人各奔东西，但仍然坚守当年的承诺，精诚合作，完成了深入理解 Android 系列中最有挑战性的 WebKit 一书。这不仅是四位作者的胜利，也是我和福川的胜利。因为这或许是深入理解 Android 系列丛书中唯一一本由多个作者共同撰写的书籍了。当时做出这样的决策是需要勇气的。因为多个作者编写同一本书，必然会经历前面所说的主创人员分合的问题，而如何保持每个作者高度一致的战斗力，并统一、平顺地组合多个作者的编撰文稿都是非常困难的事情。值得欣慰的是，今天这本 WebKit 书籍的出版，证明了我们当初的决定和作者团队的智慧，以及他们的努力都是经得起考验的。

如果说深入理解 Android 系列其他书籍是手术刀式分析 Android 源码的话，本书则以提纲挈领式的讲解帮助开发者迅速掌握 WebKit 大框架，大脉络。我曾经就此区别专门和作者团队讨论过，手术刀式的源码分析固然能满足好奇心，但是在大部分开发者对 WebKit 的宏观架构毫无了解的情况下，贸然“深入”源码分析，将极大浪费的读的者时间，而且往往事倍功半。所以，我相信在本书的内容编撰上，作者团队是认真负责的。

再次感谢这四个年轻人。因为从今天开始，各位读者就可以和我一样，通过这本书来领略 WebKit 的美了。

邓凡平

前　言 Preface

为什么要写这本书

在 PC 互联网时代，用户开启电脑后手动打开的第一个应用程序，如果不是 QQ，那往往就是浏览器。在移动互联网无比繁荣的今天，移动浏览器虽然没有像 PC 浏览器那样占据资讯第一入口的地位，但浏览器引擎一个华丽的转身，找到了自己新的、更广阔的发展空间——嵌入到各个超级 App 中，比如微信、百度搜索框等，无缝展示 Web 资源，由此可见，浏览器引擎依旧非常重要。

浏览器的重要性毋庸讳言，在这便捷的工具中，用户只需键入一个文本的 URL 或者点击一个链接，瞬间绚丽的新页面就展示在面前。浏览器具备什么样的魔法使这一切悄然发生呢？相信普通用户和众多的前端开发者都会有这个疑问。阅读开源的浏览器引擎代码（比如 WebKit），可以帮我们解开这些疑惑，这正是本书的内容。

WebKit 引擎内容庞大复杂，是一个完整的网页内容解析工具，集成 WebKit 的具体平台只需对接网络库及图形库，便可使用 WebKit 的强大解析功能。利用具体平台提供的图形库及显示系统便可实现网页的渲染和显示。WebKit 引擎由众多的子模块组成，模块功能高度内聚，彼此协同工作处理数据流。代码考究，格式优美，内容精彩，是开源代码中的上品，极具研究和学习价值。

由于 WebKit 的庞大，初学者往往无从下手，对比 Linux Kernel，虽然 Linux Kernel 的源码复杂度高于 WebKit，但相关领域已经有大量的优秀书籍，加之操作系统、计算机体系结构及接口技术是计算机相关专业学生的必修课，这都使得 Linux Kernel 的学习曲线不再陡峭。而在浏览器引擎方面，至今学习资源仍然非常少，WebKit 官网、Chromium 官网、W3C 官网、html5rock 等网站虽然有一些原理性和框架性的描述，但不够具体，要想真正了解浏览器引擎还是要跟随笔者一起阅读代码。

读者对象

本书主要介绍 Android 4.2 平台 WebKit 的原理与实现，今天看来虽然 Android 版本略显陈旧，但 WebKit 的架构与原理是没有变化的，所以本书也可帮助读者理解其他平台或其他版本 WebKit 引擎浏览器的原理。

本书的主要目标读者有如下几类：

- WebKit 内核工程师；
- 手机浏览器及浏览器类应用开发者；
- 前端开发工程师；
- 对浏览器感兴趣的大专院校在校学生。

为方便读者查阅代码，笔者已经将去掉 .repo 和 .git 目录的 Android 4.2 版本的全部源代码，上传至百度网盘，链接为：http://pan.baidu.com/s/1hqJEyP2。

如何阅读本书

本书的主体部分按浏览器处理数据的流程来组织，所以建议读者从头至尾阅读，以便于完整地了解 WebKit。当然由于 WebKit 的各个模块依据功能进行了清晰的划分，读者也可根据兴趣阅读各相关章节。

本书各章内容安排如下：

第 1 章作为全书的开篇，介绍了 Android 全源码开发环境的搭建过程，读者可在全源码开发环境的基础上研究 WebKit 代码。

第 2 章介绍浏览器工作原理及 WebKit 概览，对比了当前主流浏览器引擎，讲解了 WebKit 的优缺点、历史和现状，以及设计架构。

第 3 章介绍 WTF 库，包括智能指针、Assert、内存管理与容器、线程封装、WebKit 运行时线程结构。

第 4 章介绍 Loader 与网络库，包括 Loader 的设计与实现架构、资源加载流程、MainResourceLoader 和 SubResourceLoader、chrome-net 网络库，以及 Web Cache。

第 5 章介绍网页解析，包括 HTML 语法解析、网页处理一般过程、CSS 样式处理、JS 脚本执行等内容。

第 6 章介绍排版布局，包括 CSS 框架模型、布局计算，以及 Render 相关的核心类。

第 7 章介绍渲染与硬件加速，包括软件绘制流程、软件合成、硬件加速合成等。

第 8 章介绍 Android WebKit 框架，包括 Android Framework 介绍、Android WebKit Java 层核

心类与主要接口、Android WebKit 框架实现的源码解析，并基于 Android WebKit 的浏览器做了范例实现。

第 9 章介绍 JavaScript 扩展接口，包括 V8 原理及接口、WebKit JavaScript 接口，并做了 HTMLElement binding 实例分析和 HTML 5 扩展分析。

第 10 章介绍 WebKit 的插件系统，包括 NPAPI 插件接口详解、WebKit 的插件实现、Android 平台插件开发等。

第 11 章介绍 Remote Inspector，包括 Remote Inspector 实现架构、Remote Inspector 协议、Inspector 代码分析、BackEnd 代码分析、FrontEnd 代码分析等。

勘误和支持

由于时间仓促加之笔者水平及视野有限，错误和疏漏是难免的，欢迎读者批评指正。如果有任何宝贵意见，欢迎通过如下邮箱联系笔者：webkitbook@163.com。

致谢

感谢机械工业出版社华章公司的策划编辑杨福川和丛书主编邓凡平先生，是你们的鼓励和帮助引导我们顺利完成全部书稿。

感谢姜影编辑，感谢你对书籍初稿的订正和修改建议。

另外，还要感谢曾经的同事马兴，感谢他对本书提出的宝贵意见。

作者团队

Contents 目　录

第 1 章 Chapter 1

搭建源代码编译环境

本章主要内容

- Android 全源码开发环境；
- Android 常用工具及相关技巧说明；
- WebKit 代码目录结构及全书内容概要；
- WebKit 源代码调试。

作为全书的第 1 章，首先要介绍的是 Android 系统以及 WebKit 全源码开发环境的背景信息。

1.1 Android全源码开发环境

Android 2.3 版本以后谷歌官方推荐在 64 位 Linux 系统上编译其源代码，推荐的编译平台是 Ubuntu LTS 10.04/12.04。

1.1.1 PC 配置建议

Android 源码包很大，源码至少 3.5GB 以上，包含 git 版本信息约增加一倍空间，编译输出的 out 目录 10GB 以上，因此交叉编译的代码量是相当大的。有人曾用 DELL Latitude E5410（i3 机型）Ubuntu 实体机（非虚拟机）做首次编译，整整花了 10 个小时。所谓“工欲善其事必先利其器”，推荐使用高配置主机，例如：

- CPU：Intel i7-4770；

- 内存：16GB DDR3-1600；
- 硬盘：500GB 以上。

1.1.2 Ubuntu 系统安装

Ubuntu 的常见安装方法有：

1）使用光盘安装最简单，将刻录有 ubuntu-12.04LTS-AMD64.iso 的光盘插入光驱，重新启动机器，按照 Ubuntu 提示安装即可。

如果不方便刻录光盘，可以采用制作 USB 系统盘的方式安装。

2）使用 iso 硬盘安装，适合 Windows 7/Ubuntu 双系统。预先对硬盘分区，为 Ubuntu 预留的总分区空间应该在 60GB 以上，安装 easybcd 启动管理工具，将 ubuntu-12.04LTS-AMD64.iso 中 casper 目录下的 vmlinuz 和 initrd.lz 两个文件解压到 C 盘根目录，启动 easybcd，添加一个 NeoGrub 条目，配置启动项目如下：

```
title Install Ubuntu
root (hd0,0)
kernel (hd0,0)/vmlinuz boot=casper
iso-scan/filename=/ubuntu-13.04-desktop-i386.iso ro quiet splash
locale=zh_CN.UTF-8
initrd (hd0,0)/initrd.lz
```

重新启动机器，选择 Ubuntu 启动项，就进入安装界面，按照提示完成安装。

3）通过 Vmware 或 VirtualBox 软件，将 Ubuntu 安装到虚拟机里，这种方法会降低 Ubuntu 的性能，但是优点是可同时运行 Windows 和 Ubuntu 两种操作系统，适合机器配置比较高的用户。

为方便开发，Ubuntu 系统安装好后一般还需要配置 SSH 和 Samba 服务，以便通过其他终端（比如安装有 Windows 系统的笔记本电脑）来控制系统以及访问系统中的文件。

1.1.3 Ubuntu 下 Android 编译环境搭建

Ubuntu 系统安装完毕后，需要更新和安装 Android 编译环境需要的系统工具包，以及配置 JDK 等。

（1）安装 sun-java6-jdk 包

由于版权问题，Ubuntu 社区不再对 sun-java6-jdk 提供支持，用户需要从 Oracle 官方网站下载安装包 jdk-6u45-linux-x64.bin。

安装过程：

```
$ chmod u+x jdk-6u45-linux-x64.bin
$ ./jdk-6u45-linux-x64.bin
```

```
$ sudo mv jdk1.6.0_45 /opt
$sudo update-alternatives --install "/usr/bin/java"
"java" "/opt/jdk1.6.0_45/bin/java" 1
$sudo update-alternatives --install "/usr/bin/javac"
"javac" "/opt/jdk1.6.0_45/bin/javac" 1
$sudo update-alternatives --install "/usr/lib/mozilla/plugins/libjavaplugin.so"
"mozilla-javaplugin.so" "/opt/jdk1.6.0_45/jre/lib/amd64/libnpjp2.so" 1
$ sudo update-alternatives --install "/usr/bin/javaws" "javaws"
"/opt/jdk1.6.0_45/bin/javaws"
```

选择默认的 JDK 版本（sun-java6-jdk）：

```
$ sudo update-alternatives --config java
$ sudo update-alternatives --config javac
```

设置环境变量：

```
export JAVA_HOME=/opt/jdk1.6.0_45/
export JRE_HOME=$JAVA_HOME/jre
export CLASSPATH=$JAVA_HOME/lib:$JRE_HOME/lib:$CLASSPATH
export PATH=$JAVA_HOME/bin:$JRE_HOME/bin:$PATH
```

（2）安装依赖包

```
$ sudo apt-get install git gnupg flex bison gperf build-essential \
  zip curl libc6-dev libncurses5-dev:i386 x11proto-core-dev \
  libx11-dev:i386 libreadline6-dev:i386 libgl1-mesa-glx:i386 \
  libgl1-mesa-dev g++-multilib mingw32 tofrodos \
  python-markdown libxml2-utils xsltproc zlib1g-dev:i386
$ sudo ln -s /usr/lib/i386-linux-gnu/mesa/libGL.so.1
/usr/lib/i386-linux-gnu/libGL.so
```

至此，环境搭建完成。

注意 如果读者使用其他的 Linux 发行版搭建编译环境，可进入 Android 官方网站 http://source.android.com/source/initializing.html 参考环境搭建方法。

1.1.4 工作目录设置

Android 源码大约占用 8.5GB 的硬盘空间，至少需要 30GB 可用空间完成编译。我们假定下载到用户的 HOME 目录下，因此 /home 目录所在分区剩余空间不能少于 30GB。

1.1.5 源代码下载

Android 的官方源码放在 Google git 服务器上，通过 Google 提供的 repo 脚本同步到本地。步骤如下：

```
$ mkdir ~/bin
$ PATH=~/bin:$PATH
$ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
```

```
$ chmod a+x ~/bin/repo
$ mkdir $HOME/android
$ cd $HOME/android
$ repo init -u https://android.googlesource.com/platform/manifest -b android-4.2_r1
$ repo sync
```

以上步骤，先通过 curl 下载 repo 脚本，再通过 repo 脚本下载 Android 源码，在 repo init 时可以指定分支，比如本次是 android-4.2_r1，说明要下载 android 4.2 release 1 的源码。如果网络不好，repo sync 花的时间非常长，而且经常中断，可以用如下简单脚本来 sync 代码：

```
repo sync
while [ $? -ne 0 ]
do
    repo sync
done
```

此外，也可以通过 git 命令实现单个 git 库下载。需要在 repo init 产生 .repo 目录后，按照 .repo/manifest.xml 中的工程描述来逐个下载：

```
$git clone https://android-review.googlesource.com/build
$git clone https://android-review.googlesource.com/abi/cpp
......
```

下载源码耗时很长，比较痛苦，可以在下载时建立本地镜像，方便下次同步代码：

```
$ mkdir -p /usr/local/aosp/mirror
$ cd /usr/local/aosp/mirror
$ repo init -u https://android.googlesource.com/mirror/manifest --mirror
$ repo sync
```

这样在本机 /usr/local/aosp/mirror 建立了源码树，下次同步源码时就不用在网络上下载了，而只需：

```
$ mkdir -p /usr/local/aosp/master
$ cd /usr/local/aosp/master
$ repo init -u /usr/local/aosp/mirror/platform/manifest.git
$ repo sync
```

1.1.6 整体编译 Android 源代码

编译步骤如下：

```
$ cd $HOME/android/android-4.2_r1 // 进入源码目录
$ source build/envsetup.sh        // 初始化环境变量
$ lunch aosp_arm-eng              // 选择目标开发板
$ make -j8                        // 编译
```

命令中 source 和 launch 两步的目的在于配置环境变量，如果不开新的终端，source 和 launch 只运行一次就可以了。编译参数 -j8 的作用是启动多核并行编译，可显著提高编译速度。-jn 中的 n 取决于 CPU 的核心数，一般设置为“CPU 核心数 * 2”。如果没有运行 launch，会提示用户选择编译目标，每一个编译目标格式是 BUILD-BUILDTYPE，其中

BUILD 是产品名称，BUILDTYPE 是编译类型。常见的 BUILD 如下：

- aosp_arm：模拟器，包含所有语言、apps、输入法。
- aosp_maguro：maguro 设备，可运行在 Galaxy Nexus GSM/HSPA+ 上。
- aosp_panda：panda 设备，运行在 PandaBoard 设备上。

常见的 BUILDTYPE 如下：

- user：最终用户机，无 root，限制访问。
- userdebug：调试测试机有 root 权限，有调试工具。
- eng：工程机，带有全部调试符号。

编译输出在 out/target/product/xx/system/，其中 xx 代表某产品，比如 generic 表示模拟器，maguro 代表 Galaxy nexus.system/app 是所有 apk 目录，lib 是 so 库的目录。与 WebKit 浏览器关联的 Browser.apk 和 libwebcore.so 就分别在这两个目录。

带调试符号的 so 库输出在 out/target/product/xx/symbol/system/lib/，调试 native 代码（比如 WebKit）时，就会用到这些带有符号的 .so 文件。

1.1.7 单个模块按需编译

编译单个模块，可以在 Android 源码主目录“make 模块名”编译，比如编译 Browser 模块：

```
$ make -j8 Browser
```

模块名可以是多个。也可以进到模块目录，用 mm 命令编译。mm 是 source build/envsetup.sh 定义的 bash 函数：

```
$ cd packages/apps/Browser
$ mm
```

如果要查看编译中 make 实际做了什么事情，要加上 showcommands 参数：

```
$ mm showcommands
```

1.1.8 编译生成本地 Android SDK

进入 Android 源码目录，设置环境，运行 make 生成 sdk：

```
make sdk
```

上述命令生成文件 out/host/linux-x86/sdk/android-sdk_eng.xxxx_linux-x86.zip。在 Android 源码编译成功的基础上重新编译带自己 API 的 android.jar 需要进行以下几步操作：

- 添加自己源代码：在 Android 源码 frameworks/base 目录下新建一个文件夹保存代码，如 tv，以包的形式命名添加文件所在的路径，如 /tv/java/android/tv，并将相应的包的名称定义在目录内 Java 文件的头部。

❑ 修改 makefile 文件。

➢ 修改 build/core/pathmap.mk 文件，在 FRAMEWORKS_BASE_SUBDIRS 变量后添加 tv\，代码如下（makefile 文件中的语句以 Tab 键空格开头）：

```
FRAMEWORKS_BASE_SUBDIRS := \
    $(addsuffix /java,\
        core\
        graphics\
        location\
        media\
        opengl\
        sax\
        telephony\
        wifi\
        vpn\
        keystore\
        tv\
    )
```

➢ 修改 frameworks/base/Android.mk 文件，在 packages_to_document 变量后添加自己源代码的包名称，如 android/tv，结果如下（makefile 文件中的语句以 Tab 键空格开头）：

```
packages_to_document := \
        android \
        javax/microedition/khronos android/tv
```

❑ 在 Ubuntu 命令行终端使用如下命令进行编译：

```
make clean
make update-api
make PRODUCT-sdk-sdk
```

最后一行也可以用如下两个命令代替：

```
make -j4 sdk
```

android.jar 文件所在的目录为：/out/target/common/obj/PACKAGING/android_jar_intermediates/android.jar，并在 /out/target/common/obj/JAVA_LIBRARIES/android_stubs_current_intermediates/src 目录下重新以 package 形式组织所有生成到 android.jar 中的源代码。

SDK 所在目录为 out/host/linux-x86/sdk/，生成的 SDK 文件目录和压缩包为：android-sdk_eng.xxx_linux-x86.zip（xxx 为自己 Android 源码所在的根目录文件名）。

1.2 Android常用工具使用及相关技巧说明

Android SDK 本身包含很多帮助开发人员设计、开发、测试和发布 Android 应用的工具，本节将讨论最常用的工具。

- 开发利器 adt-bundle，是封装 Eclipse 和 adt 的集成开发工具，编写、调试 Android 程序的 Java 代码，并集成 DDMS。
- SDK Manager，该工具包含很多重要的功能，包括管理不同的 Android SDK 版本（构建目标）。Android 的版本众多，并且 API 有些兼容性问题。另外，该工具还用于管理 Android 虚拟设备配置（AVD），用来配置模拟器，如图 1-1 所示。

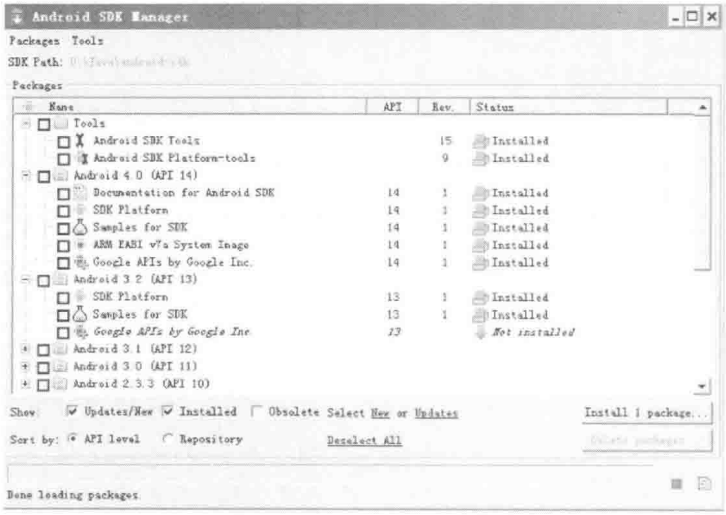

图 1-1　SDK Manager

- adb（Android Debug Bridge）是 Android 提供的一个通用的调试工具。借助这个工具，我们可以管理设备或手机模拟器的状态，快速更新设备或手机模拟器中的代码，如应用或 Android 系统升级，在设备上运行 shell 命令，管理设备或手机模拟器上的预定端口，在设备或手机模拟器上复制或粘贴文件等。
- DDMS 的全称是 Dalvik Debug Monitor Service，它提供多种调试分析手段，如测试设备截屏、logcat 输出、模拟电话呼叫、SMS、生成虚拟地理坐标、查看特定进程的线程以及堆信息等。如图 1-2 所示。
- Android 的模拟器，可以模拟不同的设备，用来运行程序，查看运行结果，测试 Android 应用的运行。如图 1-3 所示。
- logcat 是 Android 中的一个命令行工具，可以用于得到程序的 log 信息。Android 日志系统提供了记录和查看系统调试信息的功能。日志都是从各种软件和一些系统的缓冲区中记录下来的，缓冲区可以通过 logcat 命令来查看和使用。