Beyond Web & Firefox OS

GAIA FROM ABOVE

Introduce One Module A Week About FirefoxOS Gaia Modules and Apps

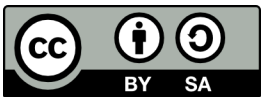
Beyond Web and Firefox OS - GAIA from above

□□Gaia

gasolin, PoYuChen, Tzu-Lin Huang, Arthur Chen, EragonJ, Steve Chung, Evan Xd, John Hu, Yuren Ju, Greg Weng, Luke Chang and Cervantes Yu

This book is for sale at http://leanpub.com/gaiafromabove

This version was published on 2015-08-16



Tweet This Book!

Please help gasolin, PoYuChen, Tzu-Lin Huang, Arthur Chen, EragonJ, Steve Chung, Evan Xd, John Hu, Yuren Ju, Greg Weng, Luke Chang and Cervantes Yu by spreading the word about this book on Twitter!

The suggested hashtag for this book is #gaiafromabove.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

https://twitter.com/search?q=#gaiafromabove

Introduce FirefoxOS Frontend projects – Gaia – that use true web technology to build the mobile operating system.

Contents

Preface

Always bet on web[1]

```
"You don't have to know if you bet on Web-based applications.
No one can break that without breaking browsing.
The Web may not be the only way to deliver software,
but it's one that works now and will continue to work for a long time.
Web-based applications are cheap to develop,
and easy for even the smallest startup to deliver."
-- Paul Graham on `why web applications are the other road ahead`.
```

Firefox OS, the frontier of web technologies, is the voyage of the Mozillans to explore strange new worlds, to seek out emerging and wild uninvented web technologies, and to boldly go where no man has been before!

Firefox OS is the fantacy land. It shows that everything is achievable via web technologies, and you, a web developer, can live with it happily ever after ... oneday. The good thing is Firefox OS is open source: you can be the one make it a reality!

`Beyond Web - GAIA FROM Above` will introduce the Front End of Firefox OS - GAIA, by module and functionalities. The material in this book is mostly based on the studies of and sharing sessions from Taipei GAIA Platform(PE) and Device porting(DPE) team.

Since Firefox OS is evolving, this book is based on the master branch of Gaia (from v1.4 at 2014/12 to 3.0 2015/3).

Why I write this book

To develop a Mobile OS is a challenging task. To make Mobile OS by web technology is an ambitious and exciting goal. As the B2G and Gaia project evolve, I found the lack of documentation to help developers understand how constituents of B2G fit together. Many aspects of how the system works, such as how the system is booted or an app is launched, remain undocumented. The situation might scare developers away from start getting hands on this interesting project. As around top #30 gaia contributor, I take the action to write this book to help myself and potential contributors to learn deeply about how Gaia works.

[1] http://bergie.iki.fi/blog/5185282727/

Target Audience

Why bother reading this book

FIrefox OS is a truely open source platform based on a sophisticated web engine, Gecko, and is developed by the great open minds from Mozilla and the community. Many new Web APIs, tools, design patterns are invented and applied while web technology is unleashed to utilize mobile device capabilities. You may not want to miss the tide of modern mobile web platform.

Prerequisites

To make this book concise and readable, we assume readers to have background knowledge of Javascript, CSS, and HTML. How to develop webapps is out of scope of this book as well. A reader can learn how to develop a Firefox OS webapp from reading the articles in Build Apps for Firefox OS[2], watching the videos series App Basics for Firefox OS[3], or reading the book Quick Guide For Firefox OS App Development[4].

Let's start the journey.

Credit

Thanks Mozilla to bring the web as the platform to the community. Carriers and device vendors also play the key roles to make this open source platform delivered to amount of people.

The main author is Fred Lin who coordinates the book structure and the writing.

Related chapters are based on presentations and documents from:

- Gaia under the hood - Kanru Chen
- Build script - Yuren Ju
- System and window management - Alive kuo and Luke Chang
- Settings - Arthur chen

The awesome book cover is designed by Vit Lai.

Since the book is CC 2.0 shared-alike license, part of this book is available on MDN[5]. Thanks Chris Mills for editing.

[2] https://developer.mozilla.org/en-US/Apps

[3] https://developer.mozilla.org/en-US/Firefox_OS/Screencast_series:_App_Basics_for_Firefox_OS

[4] https://leanpub.com/quickguidefirefoxosdevelopment

[5] https://developer.mozilla.org/en-US/Firefox_OS/Platform/Gaia/Gaia_apps

Discussion

If you have any issue or any comments about this book, please go to the feedback[6] section in book landing page and leave your comment there.

[6] https://leanpub.com/gaiafromabove/feedback

Chapter 1 - Overview

1.1 Firefox OS architecture

The main Firefox OS architecture is

- Application Layer : known as Gaia[7]
- Platform Layer : known as Gecko[8]
- Infrastructure Layer : known as Gonk[9]

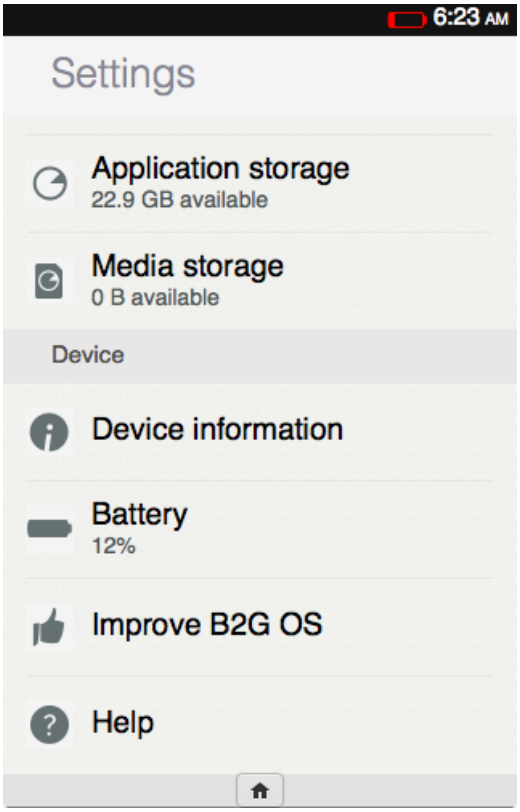
You may want to know the Firefox OS architecture better before you start developing Gaia. Here are introductory slides of the Firefox OS and Gaia development flow: Developing Firefox OS[10] and FirefoxOS for developers[11]. Also there's an Overview and High Level Architecture[12] that elaborates the Firefox OS Architecture more concisely.

Boot2Gecko (B2G)

Boot2Gecko (B2G) is the project name of Firefox OS. You can see it anywhere through Firefox OS related projects. `Firefox OS` and `Firefox` are branded names from Mozilla Foundation. Consult Mozilla before making a production device with such names. In unofficial version of Firefox OS, you will see `B2G OS` instead of `Firefox OS` on the settings app `device` section if you are using the simulator or the development build.

---

[7] https://github.com/mozilla-b2g/gaia
[8] http://hg.mozilla.org/mozilla-central
[9] https://github.com/mozilla-b2g/B2G
[10] http://www.slideshare.net/gasolin/develop-firefox-os1-28348187
[11] http://slid.es/davidflanagan/firefoxos_for_developers
[12] http://www.mozilla-hispano.org/archivos/docs/fxosappdaysvall/arquitectura.pdf



Simulator shows b2g os in Device section

1.2 Gecko Overview

Gecko is the core engine powering all Firefox and FirefoxOS versions. You may not know the fact that `every browser is like gecko`. Since the sentence `like gecko` is a build-in keyword in every internet connection messages. Search `[user-agent](http://en.wikipedia.org/wiki/User_agent#Format_for_human-operated_web_browsers)` of any browser, you will find:

```
1 Mozilla/5.0 (Windows NT 6.2; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) \
2 Chrome/32.0.1667.0 Safari/537.36
```

Since Gecko is the browser engine from the very first popular Netscape navigator. Internet Explorer and other browsers put this keyword in their user-agent messages. Now it's in the W3C standard.

Gecko supports web open standards: HTML, CSS, and JavaScript, and some experimental APIs to push web to the new frontier. Gecko also includes a networking stack, graphics stack, layout engine, a JavaScript virtual machine, and porting layers to makes sure those APIs work well on every operating system Gecko supports.

You can download Gecko source by following the guide[13]. (Note: To build full Firefox OS you don't have to download gecko separately)

Gecko is ported to several platforms via `port` layer. For example, the Firefox Windows version adapts the `gecko port` for Windows to build the browser with native look and feel.

In Gecko source there's a `b2g/` folder that contains the adapter to gonk (gonk port) that unleashes mobile hardware capabilities to the web. Mozilla and the community will keep extending it to push the capability of Firefox OS and web standards.

1.3 Gonk Overview

Gonk is the device porting layer, which is an adapter between hardware and gecko. Gonk is a relatively simple Linux distribution that can be treated as a Gecko Port paired with Gecko porting layers.

Gonk includes the Linux kernel, userspace hardware abstraction layer (HAL), and other OEM specific libraries. Since different devices may have their specific chipsets and varied hardware specs, the Gonk layer may look pretty different for devices.

https://github.com/mozilla-b2g/B2G contains official support gonk ports for a variety of devices. You can treat it as a gonk repository. The list of supported devices is available in B2G/config.sh[14].

General gonk work includes porting to the specific board and make sure Gecko can work well on the device.

1.4 Gaia Overview

Gaia is the front-end of Firefox OS, which contains system administration and build-in apps shipped with a Firefox OS device. All Gaia source, including the System, Keyboard IMEs, and everything you can see are implemented with HTML5 (HTML+CSS+Javascript) & Open WebAPIs.

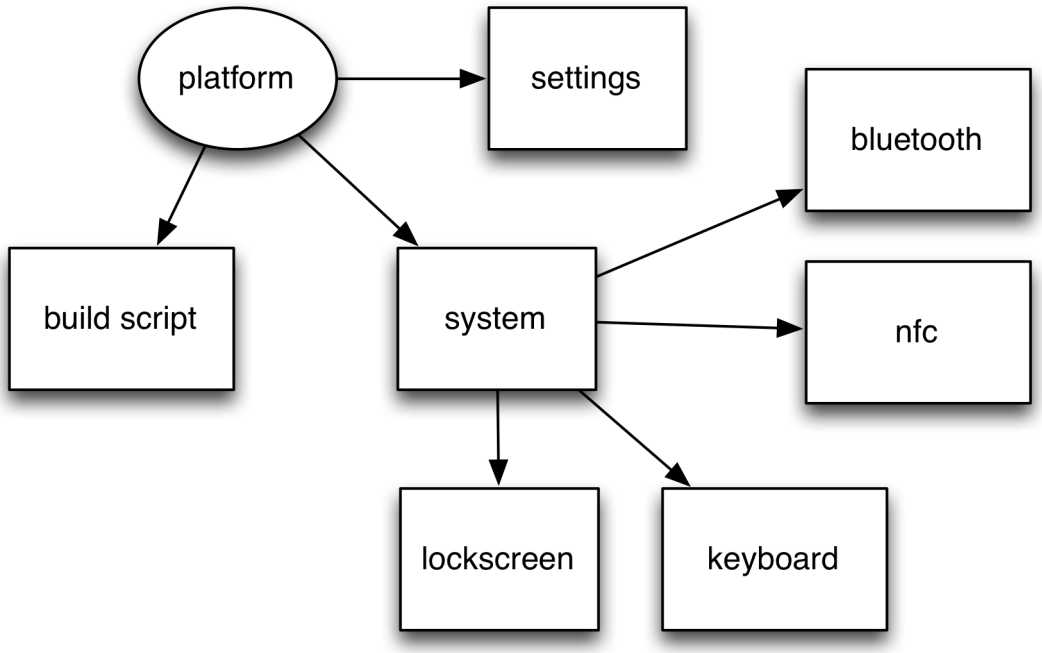
The Gaia functionalities can be roughly categorized into the following groups:

Platform

Including System, Settings, Lockscreen, Build script, Keyboard and IME, and Bluetooth apps.

[13] https://developer.mozilla.org/zh-TW/docs/Developer_Guide/Source_Code/Mercurial

[14] https://github.com/mozilla-b2g/B2G/blob/master/config.sh#L158



platform

System

The System app is the first web app loaded by Gecko during the Firefox OS bootup procedure. It bears numerous responsibilities that are essential for running the system, and are therefore out of scope of individual web apps.

Window Management

Firefox OS's window management functionality — including app life cycle and interaction, notifications, animations and much more — is handled by a specific part of the System app. This article looks at Firefox OS Window Management in detail.

Settings

The Settings app allows Firefox OS users to configure device settings, and responds to incoming activities (Web activities named `configure`), which allows other apps to jump to different panels inside the Settings app to handle configuration as required (such as showing the wifi settings panel if no data connection is available.)

Keyboard

Keyboard is a special app which works closely with system. It handles layout and multiple input-method editors (IME).

FrontEnd

Homescreen

Like other mobile platforms, Homescreen shows currently installed webapps and bookmarks.

Browser

The System Browser provides browser-like functionality where it is needed — including page navigation, search and bookmarks.

Gaia Elements

Gaia element are reusable UI components that presents Firefox OS look and feel.

BuildingBlock and Gaia Components

BuildingBlock[15] is the reusable UI components set that presents Firefox OS look and feel. It's being replaced by Gaia elements.

From 2.1, the BuildingBlock is progressively replaced by gaia-components[16], which use web component to make UI components more reusable. You can preview available components at http://gaia-components.github.io/gaia-components/ .

If you want to preview the Gaia components on desktop, you could browse to 'about:config' in Firefox, search for 'dom.webcomponents.enabled' preference and set value to `true`. Then the required `Document.registerElement` API is enabled.

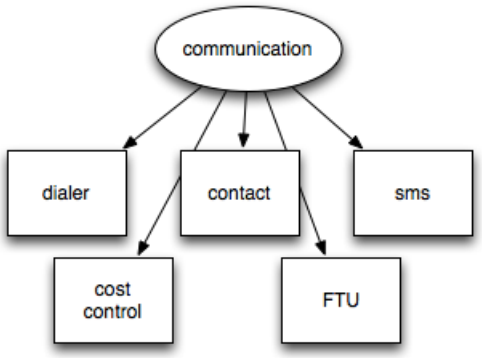
WebIDE (was App Manager)

The WebIDE[17] and App Manager[18] is the development tool in Firefox for Desktop to help you test, deploy and debug HTML5 web apps on Firefox OS phones and the Firefox OS Simulator, directly from your browser.

Communications

The communications apps include Dialer, Contact, Messages, Emergency Call, Cost Control and FTU. Some telephony related functionality such as Cell Broadcast, Voice Mail are also in this scope.

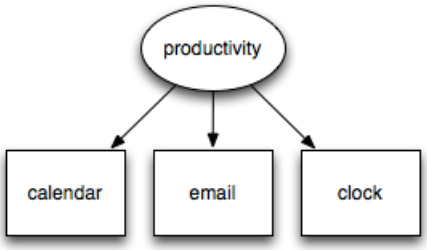
[15] https://developer.mozilla.org/en-US/Apps/Tools_and_frameworks/Building_Blocks
[16] https://github.com/gaia-components
[17] https://developer.mozilla.org/en-US/docs/Tools/WebIDE
[18] https://developer.mozilla.org/en-US/Firefox_OS/Using_the_App_Manager



communications

Productivity

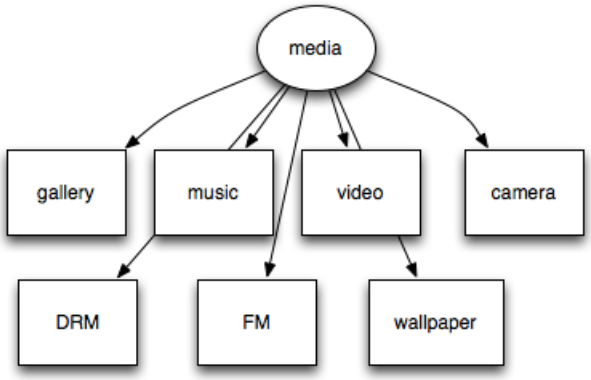
Email, Calendar, Clock apps are in the productivity category.



productivity

Media

The media apps consist of Camera, Gallery, Music, Video, FM Radio, Ringtones, and some media related functions such as forward lock DRM(fl) and wallpapers.



media

In addition to these functions, there are several major features such as Marketplace, test framework, pdf.js, which are developed closely internally or externally from Gaia.

1.5 WebAPI

Mozilla has been working actively to expose web capability to mobile devices. It's done by defining web native APIs that make use of mobile sensors, telephony, wifi, batteries and so on.

The public APIs are documented on the MDN WebAPI[19] page. Some APIs in progress are tracked in WebAPI proposed to W3C[20] wiki page.

Note that these APIs not only are designed make Firefox OS work, but also are proposed to become standards for all web browsers. When implemented on a conforming browser, these APIs can be used immediately and Web developers don't need to 'port' their webapp to other platform or browser.

1.6 Permission and security model

Some WebAPIs are ready for use in FirefoxOS and Firefox for Android. Since some are essential to provide a full 'smartphone' experience, such as telephony, but the the development of the APIs is not finalized yet, some of those APIs can only be used by gaia and build-in apps. Those are called 'certified' API. Some APIs, such as TCP socket, are useful in some circumstances but may have security concerns. Webapp using those APIs need to be reviewed and signed by Firefox Marketplace or some other trusted organization to make sure the user won't be offended. These APIs are called 'privileged' APIs.

In MDN WebAPI[21] page shows 'certified' and 'privileged' tags after each WebAPI. Generally if you are a webapp developer, you can't access to 'certified' APIs at this moment.

[19] https://developer.mozilla.org/en-US/docs/WebAPI
[20] https://wiki.mozilla.org/WebAPI
[21] https://developer.mozilla.org/en-US/docs/WebAPI

Here is the list of permissions[22] that available.

1.7 Release Cycle

Since Firefox OS 1.2, Mozilla tries to align the Firefox OS release cycles with that of the Firefox desktop version (that is, 6 weeks). Firefox OS releases a new version every 3 months. So every Firefox OS release will bypass a Gecko (Firefox browser core) release.

For example. Firefox 1.4 is bundled with Gecko 30 and Firefox 2.0 is bundled with Gecko 32, which skips Gecko 31.

Check https://wiki.mozilla.org/RapidRelease/Calendar for the Firefox OS versions and the correspondent gecko versions.

1.8 Where to discuss and get reference

The main resource of Gaia is available on Mozilla Developer Network[23] and Mozilla Wiki[24]. The secondary source is in Mozilla Wiki[25]. Or you can find some cutting edge features from Bugzilla[26].

You can use irc://irc.mozilla.org #gaia to discuss with other gaia developers instantly. And feel free to find answers and ask/reply questions on StackOverflow[27] with the tag 'firefox-os'.

dev-gaia[28] is the official mailing list of Gaia Development. Once you act as a gaia developer, you can learn a lot and participate in to change the world of web.

Please keep the article How To Ask Questions The Smart Way[29] in mind before you start asking any question there.

1.9 Build the entire Firefox OS from source code

This book is about Firefox OS application layer: Gaia. We'll just sketch the process of building the entire Firefox OS from source code.

Follow the instructions from build prerequisites[30], fetch code[31] in Mozilla Developer Network to download the full Firefox OS source code.

[22] https://developer.mozilla.org/en-US/docs/Web/Apps/App_permissions
[23] https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS/Platform/Gaia
[24] https://wiki.mozilla.org/Gaia
[25] http://wiki.mozilla.org
[26] http://bugzilla.mozilla.org
[27] http://stackoverflow.com/questions/tagged/firefox-os
[28] https://lists.mozilla.org/listinfo/dev-gaia
[29] http://www.catb.org/esr/faqs/smart-questions.html
[30] https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS/Firefox_OS_build_prerequisites
[31] https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS/Preparing_for_your_first_B2G_build

Here's a good evaluation choice called foxbox[32] that helps you create a configured VM environment without affecting your desktop environment (disclosure: this tool is developed by the author). Your desktop should have hardware virtualization(VT-x or AMD-V) support. It's available on Mac/Windows 8. Prepare to have on your workstation at least 4GB RAM and about 40GB disk space (for full OS, 5∼10GB for gaia).

Remember to backup the phone system partition[33] before the first time you try to flash your device. Then you are ready to follow build[34] instructions to configure and flash the device or B2G simulator.

Generally you can config the target device with the following command:

```
ANDROIDFS_DIR=<absolute path to parent dir of system dir> ./config.sh <target>
```

Once your environment is settled down, command `config.sh <target>` can help you config your build environment for the target device. `build.sh` helps you build the device image, and `flash.sh` flashes the ROM on to the device.

Firefox OS reuses the AOSP(Android Open Source Project) build system but does not build the java related parts. Instead of `build/envsetup.sh` and `lunch` in android, Firefox OS wrap them in `config.sh`.

The java related parts are build with `fake-jdk-tools` folder. So Firefox OS developer do not have to modify much of AOSP build process.

You can read gonk-misc, build Firefox OS through AOSP build system[35] (written in Chinese, you may use web translation tool to read it if you don't read Chinese) to gain more detail about the Firefox OS build process.

1.10 Get a reference phone

FirefoxOS Reference Phone (Flame) is available[36] with global free shipping for $170. It has 4 inch screen, and the RAM is configurable from 256MB to 1GB in the bootloader. This enables you to use same device to check your webapp with limited or rich memory.

I'd recommand you to get one for FirefoxOS development.

1.10.1 Flash the device

There's a tool that help developers grab nightly builds and flash to the reference phone

You can check B2G-flash-tool[37] for more detail.

[32] https://github.com/gasolin/foxbox

[33] https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS/Firefox_OS_build_prerequisites#Backup_the_phone_system_partition

[34] https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS/Building

[35] http://tech.mozilla.com.tw/posts/3433/gonk-misc-a-bridge-from-aosp-build-system-to-building-firefox-os

[36] http://www.everbuying.com/product549652.html

[37] https://github.com/Mozilla-TWQA/B2G-flash-tool

1.11 Reference

- Public UI Spec https://mozilla.app.box.com/s/44utizl9oz4eupyu3fuu

Chapter 2 - Get in touch with Gaia

2.1 How to try Gaia

The quickest way to try Gaia is via Firefox WebIDE[38] (for Firefox 33 above) or Firefox App Manager[39].

Firefox WebIDE[40] is a developer tool available in Firefox for Desktop. It provides a number of useful tools to help you test, deploy and debug HTML5 web apps on Firefox OS phones and the Firefox OS Simulator, directly from your browser.

In desktop Firefox Browser 33+, open the WebIDE from MENU > Tools > Web Developer > WebIDE. Select a runtime from the menu and you are ready to experience Firefox OS.

The whole viewable and responsive things in the Simulator are done by Gaia.

2.2 How to get Gaia Source

It's possible to only get Gaia source, run it with a nightly browser or simulator, and flash it without modifying Gecko. First, you need to have a Windows/Mac/Linux PC or laptop with `git` installed.

You should fork the main Gaia[41] repository to your own. It's better to register a github[42] account so you can contribute your code back to Gaia and keep code in a maintainable way.

Click the 'fork' button in top right corner to fork to your own repository from https://github.com/mozilla-b2g/gaia[43]. Now you can clone the code from your forked repository:

```
1 $ git clone https://github.com/gasolin/gaia.git
```

And remember to run the following command in that folder:

```
1 $ git remote add upstream https://github.com/mozilla-b2g/gaia.git
```

to add remote `mozilla-b2g/gaia` as `upstream` to your git remote repository.

2.2.1 How to keep gaia up to date

Once you have `mozilla-b2g/gaia` in your git remote repository, you can run the command

[38] https://developer.mozilla.org/en-US/docs/Tools/WebIDE
[39] https://developer.mozilla.org/en-US/Firefox_OS/Using_the_App_Manager
[40] https://developer.mozilla.org/en-US/docs/Tools/WebIDE
[41] https://github.com/mozilla-b2g/gaia
[42] https://github.com/
[43] https://github.com/mozilla-b2g/gaia

```
$ git remote update
```

to update all remote repository.

Then run the command

```
$ git merge upstream/master
```

to make your local file up to date.

2.3 Run Gaia via Firefox mulet

Get and install the Firefox nightly version (codenane: mulet) from http://ftp.mozilla.org/pub/mozilla.org/b2g/nightly mozilla-central/[44]. The file is named like `firefox-xxx.en-US.mac64.dmg`.

Then execute the following commands in the gaia directory:

```
$ make
$ [app path]/firefox-bin -new-instance -profile [gaia absolute path]/profile-deb\
ug
```

It will bring up a new firefox nightly window, with a Firefox OS frame in it.

Gaia provides a desktop-helper addon, which has a mockup of the Firefox OS device environment and provides some useful tools. The addon allows us to test and debug Gaia code in the browser and reuse all web tools.

2.4 How to install Gaia on a real device

Make sure your device is rooted. Then, you have to enable the remote debug console before installing Gaia on the device.

2.4.1 Enable the remote debug console

Firefox OS reuses the Hardware Abstraction Layer (HAL) of Android Open Source project (AOSP) and accordingly inherits some low-level debug tools from AOSP, such as `adb (Android Debug Bridge)`.

To install Gaia on a real device, you need to have `adb` tool installed. `adb` is a command line tool that lets you communicate with the device.

On your device, check `settings > Device Information > More information > Developer Menu`. Then go back to top level of settings, and you'll see the developer panel. Click it, then select the `Remote Debugging` option as `ADB and Devtools`. It will enable the remote debug console and enable you to debug your device with the App manager.

[44] http://ftp.mozilla.org/pub/mozilla.org/b2g/nightly/latest-mozilla-central/

2.4.2 Setup adb

WebIDE already wraps the adb helper addon[45] for Gaia developers. Connect your device via USB, and then tap `connect` button in WebIDE to connect with your device.

If you like to use adb via command line, you can install adb manually. Open http://developer.android.com/sdk/index.l click `DOWNLOAD FOR OTHER PLATFORMS` and download SDK Tools only for your platform. Unzip it in your path and You can find the adb tool in `<sdk>/platform-tools/`.

To make it work with Gaia build tools, edit PATH environment in windows, edit `~/.bash_profile` in mac OS X, or the correspondent files in Linux distributions.

Let's take OS X for example, edit ~/.bash_profile:

```
export PATH=${PATH}:<absolute sdk path>/platform-tools
```

Save and reload the environment. Then, you can type 'adb' in command line to test if it works.

Once adb is settled, you can install Gaia into a real device.

2.4.3 Install Gaia on a real device

First, connect your device to your computer with a USB cable.

To check if there is any connected device, you can type:

```
$ adb devices
List of devices attached
emulator-5554  device
```

If there's a device attached, you can proceed with the running following commands. If you see no connected device with the above command, you (Windows or Linux distributions user) may need to check OEM USB Drivers[47] page to correctly set up the USB driver on your computer.

Clean Install

Run the following command to reset your phone with your gaia source code

```
$ make reset-gaia
```

In 2.2 or above, we can add an extra parameter `P=1` to enable multiprocess compiling.

[45] https://developer.mozilla.org/en-US/Firefox_OS/Using_the_App_Manager#Adb_Helper_Add-on
[46] http://developer.android.com/sdk/index.html
[47] http://developer.android.com/tools/extras/oem-usb.html

```
$ make reset-gaia P=1
```

It will be way faster.

Install apps without reboot

To test non-system apps, you can install them without rebooting the device. Just run the command:

```
$ make install-gaia
```

Install a specific app within Gaia

If you want to install a specific app only, you can pass it through the APP variable:

```
$ make install-gaia APP=browser
```

2.5 Debug with a real device

You still need the APP manager or adb tool to remote debug the apps on a Firefox OS Device.

Firefox OS shares the same console.log() command as normal web pages do. You can see the same log as in the Firefox Web Console

```
$ adb logcat
```

To clean up the log entries

```
$ adb logcat -c
```

To show the Error log entries only:

```
$ adb logcat | grep "Error"
```

or filter with -s option to achieve the same output result:

```
$ adb logcat -s "Error"
```

The full adb tools usage is available on Android Debug Bridge[48] page. Note that some android specific command may not work on Firefox OS devices.

[48] http://developer.android.com/tools/help/adb.html

2.6 Gaia source structure

Here's the quick overview of the code structure and functionality.

apps/

All main Gaia apps, including the apps shown in homescreen, such as calendar, camera, and fundamental apps such as system, homescreen, and keyboard

build/

Contains build scripts

dogfood_apps/, external-apps/, showcase_apps/, test_apps/

Folders containing other apps that will be included in customization.

Firefox `marketplace` is located in `external-apps`. It's developed separately and is included as a packaged app.

keyboard/

Keyboard dictionaries and layouts for different languages.

locales/

Localization files for different languages.

shared/

Files that are shared over the Gaia and will be included in build time

tools/

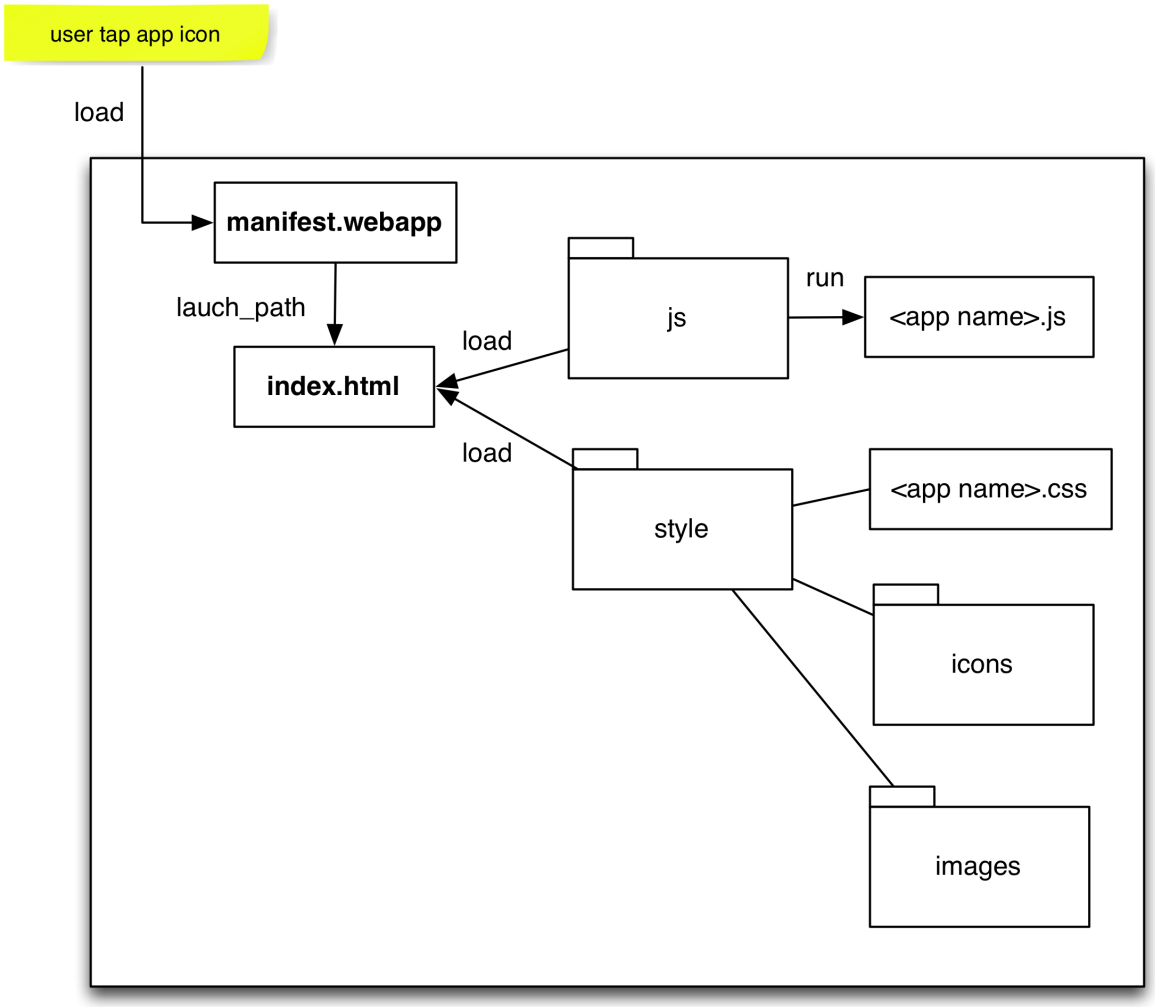
Tools for build script and test.

2.7 How a WebApp gets started

All build-in WebApps are packaged[49] webapps, which are essentially zip files containing all application assets: HTML, CSS, JavaScript, images, manifest, etc. Each WebApp in Gaia is organized with the following structure:

[49]

```
apps/[app name]/
 - js
 - styles
 - locales
 - test
 - index.html
 - manifest.webapp
```



webapp load process

Once the system app tries to launch a built-in web app, it will try to open the `manifest://[app name].gaiamobile.org:8080` URL. Gecko will try to parse the `manifest.webapp` within a webapp. `manifest.webapp` defines the launch path, and by convention it's `index.html` for all build-in webapps. The `index.html` file will load the required styles and javascript.

You can refer to the manifest format from MDN[50].

2.7.1 js

All javascripts are located in the `/js` folder.

As an informal convention, the major javascript entrypoint will be named as `[app name].js` or `main.js`.

2.7.2 styles

All app related styles are located in the `/styles` folder.

2.7.3 locales

App-related language/localization strings are located in the `/locales` folder.

The localization is done with `shared/js/l10n.js`, which wraps the `navigator.mozl10n` API and changes the strings to the translated versions in the locales/*.properties files.

A general string will be

```
1  <span data-l10n-id="hello">hello</span>
```

The `data-l10n-id` attribute specifies a key which maps to a value inside of a properties file. This content is changed after the page loads.

You can refer to Localizing Firefox OS Apps[51] for more detail.

The official translations are maintained separately in https://hg.mozilla.org/gaia-l10n.

2.7.4 tests

App-related test cases are located within the `/test` folder. Check Chapter 4 for more about running tests.

2.8 How to contribute to Gaia

Gaia is an Open Source project. You can help this project with a variety of skills you'd like to perform such as documentation, testing, coding, marketing and so forth.

If you are not sure if Gaia is the project you'd like to work on, it's ok. There're plenty of Mozilla related projects that can help you do good with other people. Consult the What can I do[52] wizard first.

[50] https://developer.mozilla.org/en-US/Apps/Developing/Manifest

[51] https://hacks.mozilla.org/2013/08/localizing-firefox-os-apps/

[52] http://www.whatcanidoformozilla.org/

2.8.1 Find a bug to contribute

If you expect to fix something in Gaia, check the Bugzilla to see if it has been done before you actually do it.

Mozilla uses the Bugzilla[53] issue tracking system to track issues and bugs. It's an essential tool to manage large-scale projects.

Bugzilla is the single place to track all Mozilla-related bugs, including Firefox OS/Gaia bugs[54]

Here's a good reference of using bugzilla[55] and a more tips and hacks[56] of using bugzilla.

You can check the good first bug[57] list, which are picked by experienced Mozilla developers. They are willing to mentor and help you fix some first easy bugs, to get familiar with the developing environment and how to contribute to these open source projects.

2.8.2 File a bug

The best ways to get contributing to an Open Source project is to file a bug.

Filing a bug allowing you to tell people that something is broken or a new thing you would like the project to do. It also introduces yourself to the existing community.

In the guide of writing a good bug[58], bugs can be categorized into two categories:

- Reporting errors
- Asking for new features or enhancements

There's a specific https://developer.mozilla.org/en-US/Firefox_OS/Developing_Firefox_OS/Filing_bugs_against_Firefox_OS[59] that shows how to file a Firefox OS bug.

Talk more on IRC & bugzilla to update your findings and how to analyze the issue.

Once you are committed to contribute to the Gaia project, there're addon tools featuring github integration:

- https://addons.mozilla.org/en-US/firefox/addon/github-tweaks-for-bugzilla/?src=search[60]
- https://addons.mozilla.org/en-US/firefox/addon/bugzillajs/[61]

[53] https://bugzilla.mozilla.org/

[54] https://bugzilla.mozilla.org/describecomponents.cgi?product=Firefox%20OS

[55] http://blog.johnath.com/2010/02/04/bugzilla-for-humans/

[56] http://flailingmonkey.com/bugzilla-tips-hacks-and-etiquette

[57] https://bugzilla.mozilla.org/buglist.cgi?resolution=---&status_whiteboard_type=allwordssubstr&status_whiteboard=mentor&product=Firefox%20OS&list_id=9166857

[58] http://www.thecssdiv.co.uk/2013/03/writing-a-good-bug/

[59] documentonMDN

[60] https://addons.mozilla.org/en-US/firefox/addon/github-tweaks-for-bugzilla/?src=search

[61] https://addons.mozilla.org/en-US/firefox/addon/bugzillajs/

2.8.3 Submit a Pull Request for review

Once you have taken a bug and have a working fix in your own branch, you could submit a Pull Request for review.

To properly get the commit reviewed, please use 'Add an attachment' and paste your Pull Request on file field. Then set review or feedback flag to the proper module owner[62].

If you have worked on your branch for a while, you may need to rebase your branch and check the recent update did not conflict with your changes.

Run the commands:

```
# update remote repositories
$ git remote update
# put your commit on top
$ git rebase upstream/master
```

to make your local file up to date and keep your commits above recent Gaia changes.

2.8.4 Use Git rebase to squish commits into one commit

The reviewer or feedbacker may give you some suggestions. Once you finish the change and commit again into your code base, please squish your commits into single commit. Therefore the commit log can have more clear while the Pull Request is merged.

Here is the reference about how to squish commit via git rebase command: http://gitready.com/advanced/2009/02/10/commits-with-rebase.html

2.8.5 Modify existing commit log

Commit logs are a general reference when people try to glance the App changelog to find out what's happen with recent build. So a meaningful commit log is essential for a healthy open source project. Rather than just copy the Bugzilla issue statement, it's more helpful to put `what problem you really solved`, `what you have done in this patch` into your commit log.

You can change your existing commit's log via following command:

```
$ git commit --amend
```

[62] https://wiki.mozilla.org/Modules/FirefoxOS

2.9 Reference

- Gaia Development Cycle[63]
- Different ways to run Gaia (https://developer.mozilla.org/en-US/Firefox_OS/Developing_Gaia/Different_ways_to_run_Gaia)
- Developing Gaia[64]

[63] https://github.com/bocoup/gaia-notes/blob/master/development-cycle.md
[64] http://www.slideshare.net/gasolin/develop-firefox-os1-28348187

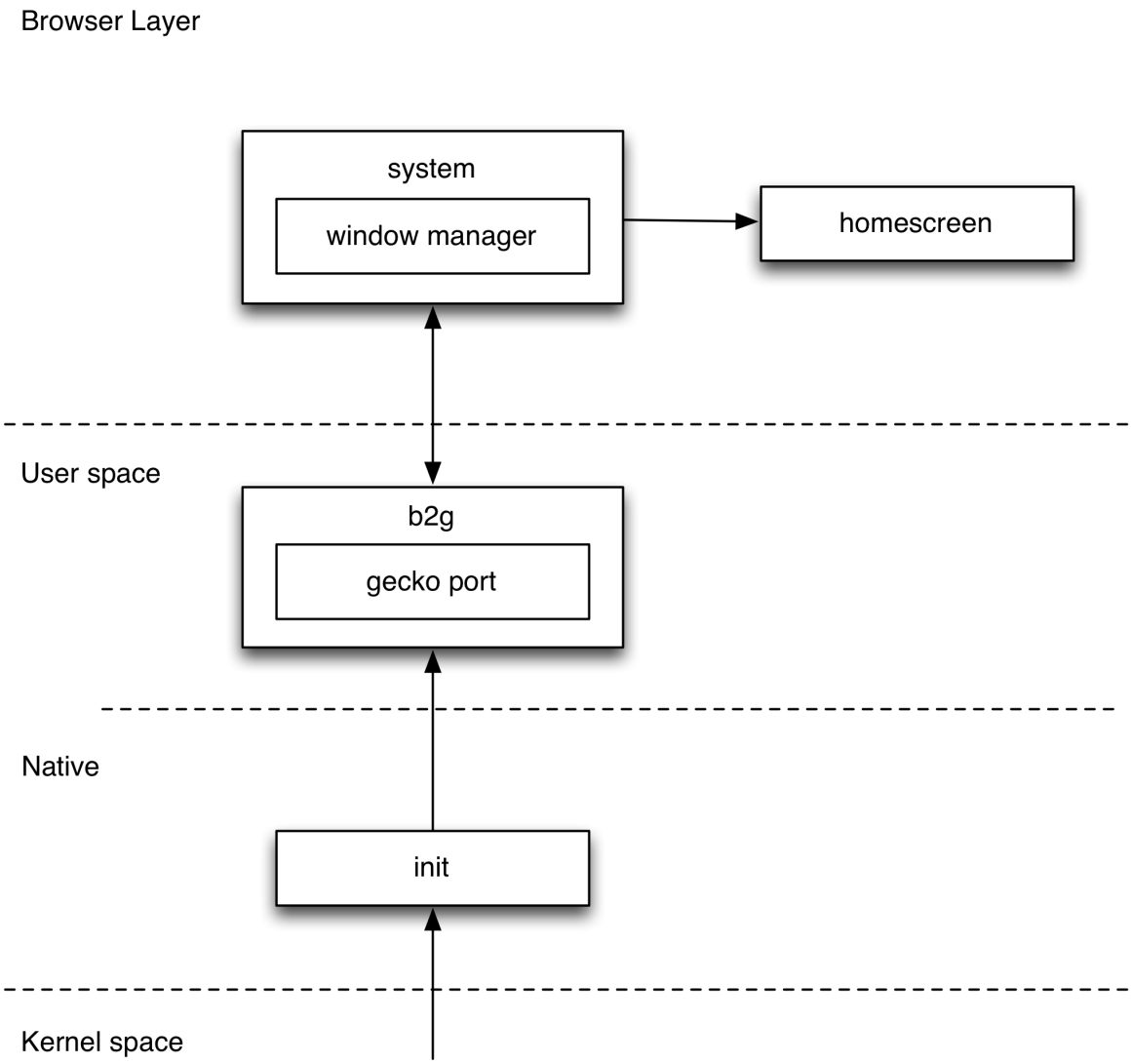
Chapter 3 - Gaia under the hood

3.1 Booting path

Before looking Gaia from above, we can take some time to know how we get to Gaia when we power on a device.

Gaia is launched from Gecko. On a Firefox OS device, Gecko is booting from the linux kernel. The Architecture overview[65] on Mozilla Developer Network (MDN) describes the bootstrapping process of how FirefoxOS boots to Gecko.

[65] https://developer.mozilla.org/en-US/Firefox_OS/Platform/Architecture



System bootup flow

3.1.1 Bootloader

The bootloader brings up the Firefox OS device, performs basic hardware checks and then hands off to the Linux Kernel to bring up the devices and run essential processes.

3.1.2 Linux Kernel process

The Linux Kernel used for Firefox OS is based on Android Open Source Project (AOSP). You can reference Android boot process[66] to understand how to bring the device up to kernel.

[66] http://bootloader.wikidot.com/linux:boot:android

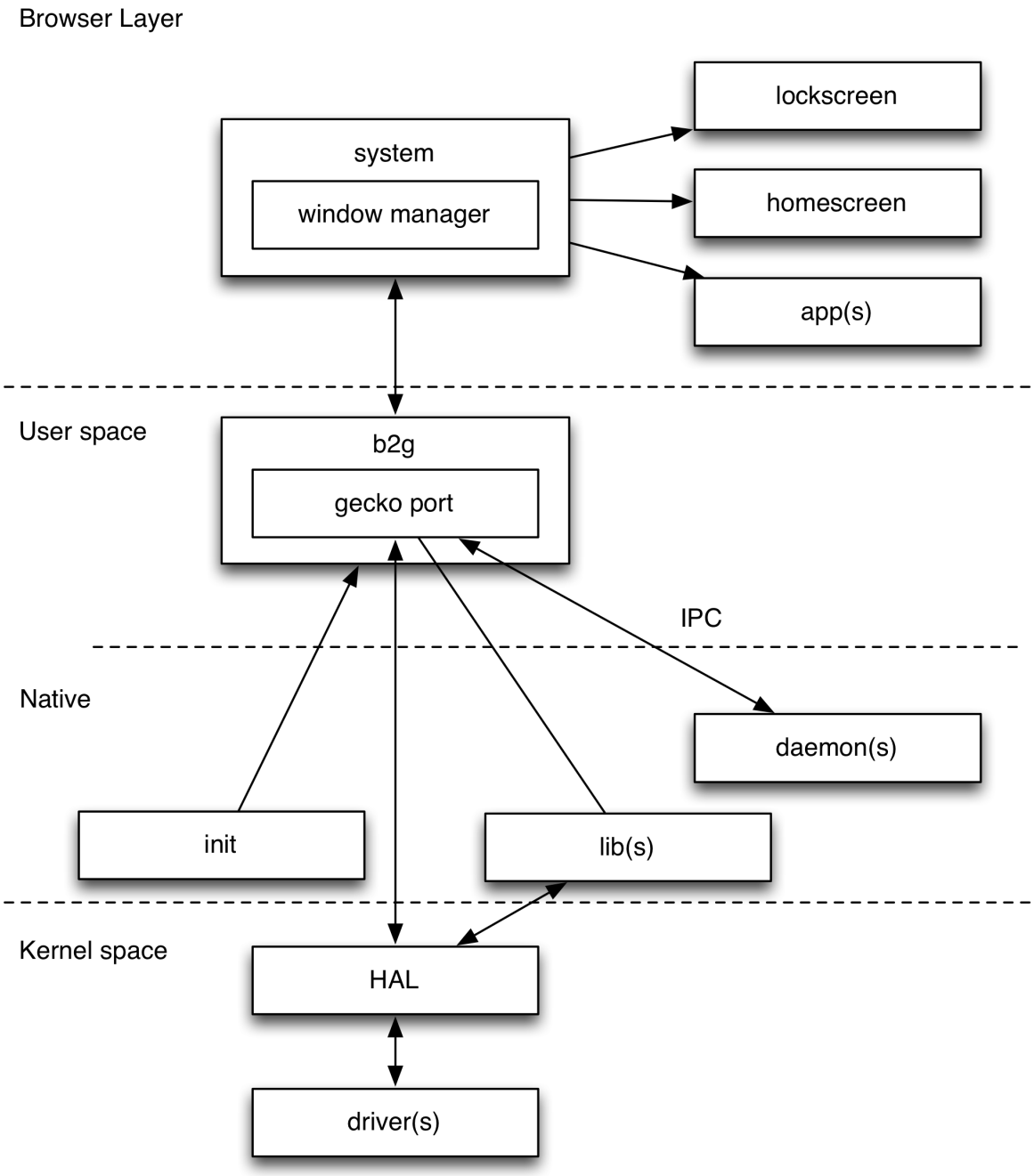
The kernel will execute processes defined in `init.rc` and the successor init.b2g.rc[67] to launch essential process such as `b2g` (the main FirefoxOS process, containing Gecko), `rild` (telephony daemon process that might contain proprietary functionalities specific to the chipset).

After the processes in init.rc are launched, we successfully boot to Gecko (the `b2g` process).

3.1.3 In b2g process (Gecko)

The `b2g` process runs `libxul`, which reference to `b2g/chrome/app/b2g.js` to get the default preferences. From the preferences it will open the described html file called `b2g/chrome/content/shell.html`, which is packed in the `omni.ja` file. The shell.html includes `b2g/chrome/content/shell.js` file, which will trigger the Gaia `system` app.

[67] https://github.com/mozilla-b2g/gonk-misc/blob/master/init.b2g.rc



FirefoxOS structure

3.2 Search and Reference Gecko code

To search the Gecko code, you can use http://dxr.mozilla.org[68]. It's neat and provides good cross-reference features, but is resreicted a limited set of repositories. Other than that, you can try http://mxr.mozilla.org[69], which is less fancy than the MXR, but contains more Mozilla projects.

3.3 Gecko files that are related to Gaia

The following folders contains files that are relevant to gaia

b2g/

The b2g folder contains mainly FirefoxOS related functions.

b2g/chrome/content

Contains Javascript files that run below the system app

b2g/chrome/content/shell.html

The entry point, the first html to load in gecko to launch the Gaia system app.

shell.html includes `settings.js` and `shell.js`:

```
<script type="application/javascript;version=1.8"
      src="chrome://browser/content/settings.js"> </script>
<script type="application/javascript;version=1.8"
      src="chrome://browser/content/shell.js"> </script>
```

`settings.js` contains system default setting parameters.

b2g/chrome/content/shell.js

`shell.js` is the first script to load the Gaia `system` app.

`shell.js` imports all required modules, registers key listeners, defines `sendCustomEvent` and `sendChromeEvent` to communicate with Gaia, and provides webapp install helper. It also contains miscellaneous functions including indexedDB quota, RemoteDebugger, keyboard helper, screenshot, etc. Among them, the most important one is that `shell.js` launchs the Gaia `system` app and then hands over the overall system related management work to the Gaia `system` app.

[68] http://dxr.mozilla.org
[69] http://mxr.mozilla.org

```
let systemAppFrame =
  document.createElementNS('http://www.w3.org/1999/xhtml', 'html:iframe');
....
container.appendChild(systemAppFrame);
```

b2g/chrome/app/b2g.js

Predefined references, like `about:config` in the Firefox browser, same as Gaia `pref.js`. Some preference values can be changed in the settings app. It can be overridden by Gaia's user.js in Gaia build script.

mozilla-central/dom/{API}

Web API implementation are placed under the dom/ directory. Some older APIs are located in dom/base, ex navigator.cpp

mozilla-central/dom/apps

Contains DOM APIs related to web apps, including permissions, installation, etc.

mozilla-central/dom/apps/src/

All Web app permissions are defined in PermissionsTable.jsm[70]

mozilla-central/dom/webidl

Contains Web API definition files. Consult MDN WebIDL_bindings[71] for supported attributes.

mozilla-central/hal/gonk

The hardware abstraction layer in Gecko interfacing with Gonk.

3.4 Generated files

modules/libpref/src/init/all.js

Contains all configs.

[70] http://mxr.mozilla.org/mozilla-central/source/dom/apps/src/PermissionsTable.jsm
[71] https://developer.mozilla.org/en-US/docs/Mozilla/WebIDL_bindings

/system/b2g/omni.ja

Contains the pack of Gecko JavaScript files, styles, resources on the device.

3.5 Contribute to Gecko bug

Though this books focuses on Gaia, you can refer to How to submit a patch[72] if you are interested and have a chance to patch gecko code.

Before that, you are also required to become a mozilla committer[73].

3.6 Reference

- FirefoxOS and its use of linux http://www.slideshare.net/aimeemaree/firefoxos-and-its-use-of-linux-a-deep-dive-into-gonk-architecture

[72] https://developer.mozilla.org/en-US/docs/Developer_Guide/How_to_Submit_a_Patch
[73] http://www.mozilla.org/hacking/committer/

Chapter 4 - Build Script

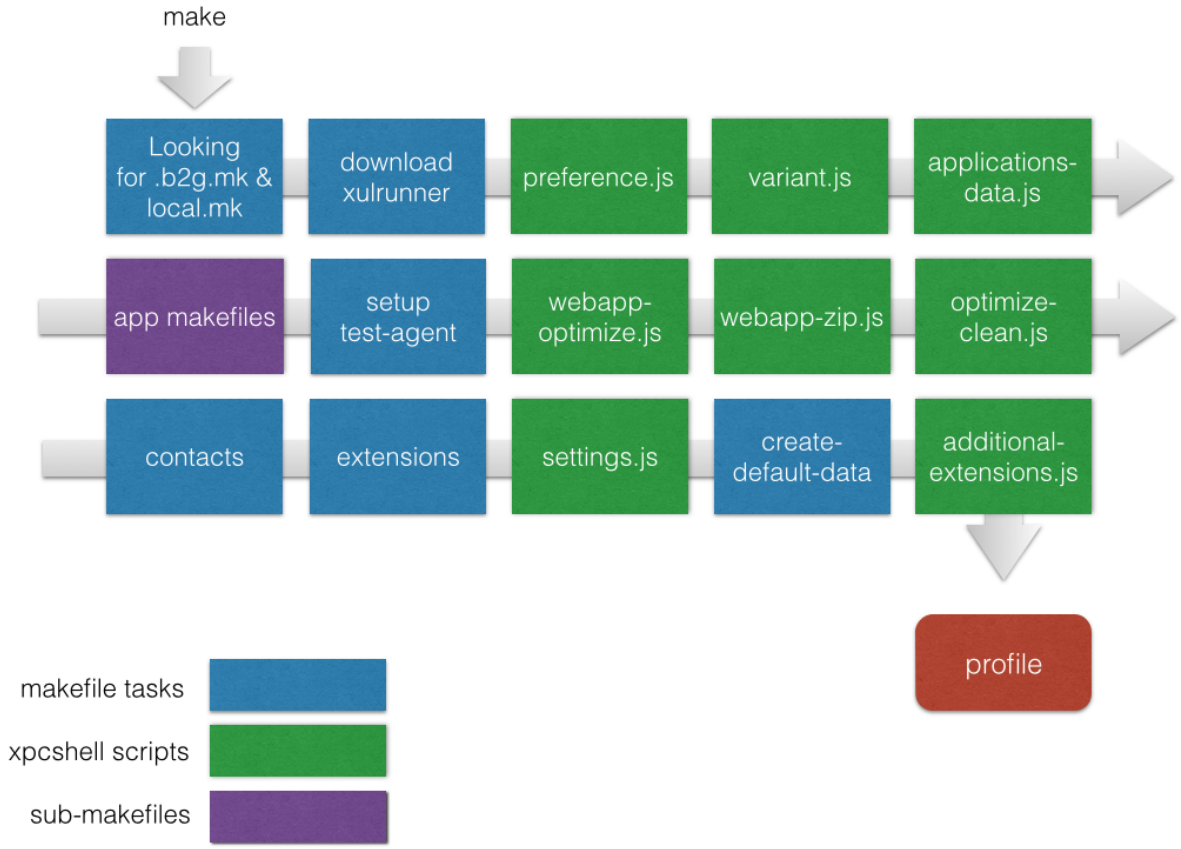
Gaia plays many tricks in build time. Build scripts are originally written in bash (Makefile), python, and Javascript. After Firefox OS 1.4, build scripts are mostly written in Javascript and executed by `xpcshell`. `xpcshell` is similar to node.js but it's capable of running some mozilla flavored javascript. It even allows core Gaia build scripts to be run in a Firefox extension.

Gaia Building Script contains many helper tools to help install, test, localize, and package webapps onto a real device. It also allows developer to customize Gaia, like changing default wallpaper, ringtone, apps, settings, for different markets.

4.1 The build process

The Gaia build system performs multiple steps whhen you run a mere `make` command in the Gaia folder. First, the shell will download `xpcshell` and extract it into your environment. `xpcshell` functions are wrapped in a commonjs-like API in `build/xpcshell-common.js` for developers to get hands on more easily. Check `build/utils-xpc.js` and `build/test` for more detail.

The following diagram provides an overview of the Gaia build process:



credit[74]

In earlier realease of Gaia, the build script handle each Gaia app's custom build process. The build script will process the Gaia app from their source folder, place the output to the profile folder, and leave intermediate files under source folder.

From 2.1 on, app specific build processes are moved to separate `apps/<app>/build/build.js` files. The build process now simplified to three stages: `pre-app`, `in-app`, and `post-app` processes. Now intermediate files are processed from source folder to `stage` folder, and then the final results are copied to the `profile` folder.

New build process

preference.js

Generates the default settings for FirefoxOS. It will generate `user.js` and put onto the device to be read by Gecko.

[74] https://mdn.mozillademos.org/files/6747/gaia-build-system.001.png

settings.js

Generates the default settings for FirefoxOS. It will be read by Gaia.

webapp-shared.js

Files in the `shared/` folder that are declared in built-in webapp's html will be included in this step.

webapp-optimize.js

Minifies app Javascripts and l10n files.

webapp-zip.js

Zips apps and puts them under the `profile/` folder.

From 2.1 on, per app build script are introduced. If `build.js` is found in per app's `build/` folder, it will take the responsibility of building this app instead of center `app.js`.

Also there are several parts that are handled by build script. You can refer to Build System Primer[75] for more information.

4.2 Per-commit coding style check

Though Javascript doesn't force you to write organized code, Gaia project follows the Google javascript coding style[76] to ensure the coding style is consistent. Gaia uses `gjslint` and `jshint` to check JS coding styles before each commit (done by git pre-commit hook). Once you submit a Pull Request to the Gaia repository, the TBPL server (Mozilla's continue integration server) will run these linters to check that all styles are correctly followed.

The precommit hook script is in `tools/pre-commit` and will be copied to project `.git/hooks` folder once a make command is executed.

Install gjslint[77] and jshint separately to check coding styles by your own.

jshint code quality check has been introduced in Gaia from 1.4 to complement with gjslint, the code style check tool. gjslint is planned to be replaced by jscs[78].

You can install it via npm:

[75] https://developer.mozilla.org/en-US/Firefox_OS/Platform/Gaia/Build_System_Primer
[76] http://google-styleguide.googlecode.com/svn/trunk/javascriptguide.xml
[77] https://developers.google.com/closure/utilities/docs/linter_howto
[78] http://jscs.info/

```
$ npm install -g jshint
```

You can check `build/jshint` for more detail about jshint in Gaia.

Gaia also provide the build target for you. You can run

```
$ make lint
```

to automatically run gjslint and jshint style checks.

Or run

```
$ make hint
```

to run jshint style checks.

4.3 Testing

The build system provides tools to set up a variety of tests, including unit test, integration test, performance test, and UI test.

4.3.1 Unit Test

Gaia uses mocha[79] as test framework, chai[80] as assert library, sinon.js[81] as mock&stub library, blanket.js[82] as test coverage tool.

You can run the following command to host a unittest server:

```
$ make test-agent-server
```

Then open the test-agent app in nightly:

```
$ make DEBUG=1
$ [FirefoxNightly path]/firefox --profile [absolute path]/gaia/profile-debug h\
ttp://test-agent.gaiamobile.org:8080
```

or use `pwd` shortcut in *nix system:

[79] http://visionmedia.github.io/mocha/
[80] http://chaijs.com/api/assert/
[81] http://sinonjs.org/
[82] http://blanketjs.org/

```
$ firefox -profile `pwd`/profile-debug http://test-agent.gaiamobile.org:8080
```

Then run all tests with the following command

```
$ make test-agent-test
```

You can add argument `APP=<app folder name>` (such as `APP=settings`) to test a single app.

4.3.2 Integration test with marionette js

Gaia Integration tests are driven by marionette scripts written in Javascript and Python. It can communicate with Gecko and is capable of controlling both browser and Firefox OS device and interacting with each other.

You can run the following command to trigger the integration test:

```
$ make test-integration
```

Again, you can also append `APP=<app folder name>` to test a single app, such as `APP=calendar`.

```
creating an event
  - should make an event visible in the month day view
  - should display the created event in read-only view

launch calendar
  ✓ should make calendar active

week view
  ✓ should have a space between months (920ms)


2 passing (10s)
2 pending
```

integration test result for calendar app

After running the `test-integration` command, you can load the profile to b2g with:

```
$ b2g/Contents/MacOS/b2g-bin -profile <absolute path>/gaia/profile-test
```

And with the `VERBOSE=1` option:

```
$ make test-integration VERBOSE=1
```

The developer can see the log on console.

If you've run `test-integration` once, you can use the `test-integration-test` command next time since `test-integration-test` command won't build the profile every time.

You can use a real device (like flame) to run the marionette test.

```
$ adb forward tcp:2828 tcp:2828
$ MARIONETTE_RUNNER_HOST=marionette-device-host make test-integration
```

4.3.3 performance test

For Gaia performance test with Raptor[83], it's only meaningful if you run the test on a real device.

You can install it independently via `sh $ npm install -g @mozilla/raptor`

The performance test will boot up the device and launch the app several times to get an average app load time. Memory usages of both the app and the system process (b2g) are also collected and reported after the test completes.

```
$ make raptor
$ APP=music node tests/raptor/launch_test RUNS=30
```

You can append `APP=<app folder name>` to test a single app, such as `APP=music`. And append `RUNS=<number>` for test rounds.

The test result may looks like below:

```
Metric Mean Median Min Max StdDev p95
------------------------------ -------- -------- -------- -------- ------ ---
coldlaunch.navigationLoaded 2360.000 2360.000 2360.000 2360.000 0.000 n/a
coldlaunch.navigationInteractive 2503.000 2503.000 2503.000 2503.000 0.000 n/a
coldlaunch.visuallyLoaded 2618.000 2618.000 2618.000 2618.000 0.000 n/a
coldlaunch.contentInteractive 2618.000 2618.000 2618.000 2618.000 0.000 n/a
coldlaunch.fullyLoaded 3525.000 3525.000 3525.000 3525.000 0.000 n/a
coldlaunch.rss 38.200 38.200 38.200 38.200 0.000 n/a
coldlaunch.uss 17.800 17.800 17.800 17.800 0.000 n/a
coldlaunch.pss 23.200 23.200 23.200 23.200 0.000 n/a
```

`Mean` denotes the average app load time, which should be less than 1 second for better user experience.

The golden rule for `raptor` is that smaller numbers are better. We won't mention too much about the detail of what this memory usage indicates, but just come with a simple note here:

- uss = unique set size
- pss = proportional set size
- rss = resident set size

[83] https://developer.mozilla.org/en-US/Firefox_OS/Automated_testing/Raptor

Generally rss >= pss >= uss . `rss` don't accurately reflect a process's usage of pages shared with other processes. So the two numbers you want to watch are the `pss` and `uss`. You can find more detailed descriptions of them online.

`uss` is the memory that is completely unique to that process. This is the amount of memory that would be freed if the application was terminated right now. It's a key value for evaluation.

`pss` reports the proportional size of its shared libraries. It includes memory that is not released if the process was terminated.

Other than `make raptor`, another alternative for performance test is b2g-perf[84]. To run it, first flash Gaia to the device with production configuration:

```
1     $ GAIA_OPTIMIZE=1 make production
```

After the device boots up, run the `b2g-perf` command:

```
1     $ pip install b2gperf
2     $ adb forward tcp:2828 tcp:2828
3     $ b2gperf --delay=10 Settings
```

The result will look like:

```
1  2014-03-28 17:51:27,349 B2GPerfRunner INFO     | Results for Settings, cold_load_\
2  time: median:1658, mean:1664, std: 58, max:1931, min:1578, all:1665,1624,1687,16\
3  47,1628,1622,1654,1648,1663,1629,1619,1641,1664,1669,1578,1632,1705,1658,1725,16\
4  61,1623,1699,1653,1621,1682,1693,1680,1659,1668,1931
```

The `mean:1664` denotes **on this device** the current average load time of the settings app is 1664ms.

More details underneath `Firefox OS App launch performance` can be found in this talk[85]

4.4 API document

Gaia doc uses the jsdoc3[86] format.

You can run the following command to generate Gaia docs:

[84] https://github.com/mozilla/b2gperf

[85] http://presentboldly.com/eliperelman/firefox-os-app-launch-performance-condensed

[86] http://usejsdoc.org/

```
$ make docs
```

4.5 Customization

Gaia supports customization for your own need, such as changing default wallpaper, ringtone, apps, settings.

You can specify the location with the GAIA_DISTRIBUTION_DIR environment variable, like this:

```
$ GAIA_DISTRIBUTION_DIR=<DISTRIBUTION_PATH> make production
```

Read Customization Guide[87] for more information.

4.6 Device type

Gaia contains some built-in device type distribution such as phone, tablet, or TV. Run the command:

```
make GAIA_DEVICE_TYPE=tablet
```

to load default tablet-related settings and configurations. The real operation is done by per app build script, which is located in `apps/<app name>/build` folder.

4.7 Keyboard IME

To get keyboard IME layout and dictionary enabled, use the following command:

```
$ GAIA_KEYBOARD_LAYOUTS=en,zh-Hant-Zhuyin,el,de,fr,zh-Hans-Pinyin make
```

4.8 Localization

You have to download locale files separately. Take Traditional Chinese for example, you can get the locale file via mercurial(hg):

```
$ hg clone https://hg.mozilla.org/gaia-l10n/zh-TW/ B2G/gaia/locales/zh-TW
```

Run the command for to build:

[87] https://wiki.mozilla.org/B2G/MarketCustomizations

```
$ LOCALE_BASEDIR=locales/ LOCALES_FILE=locales/languages_mine.json make
```

The locale files should be listed in the languages_mine.json file.

The file can be as simple as

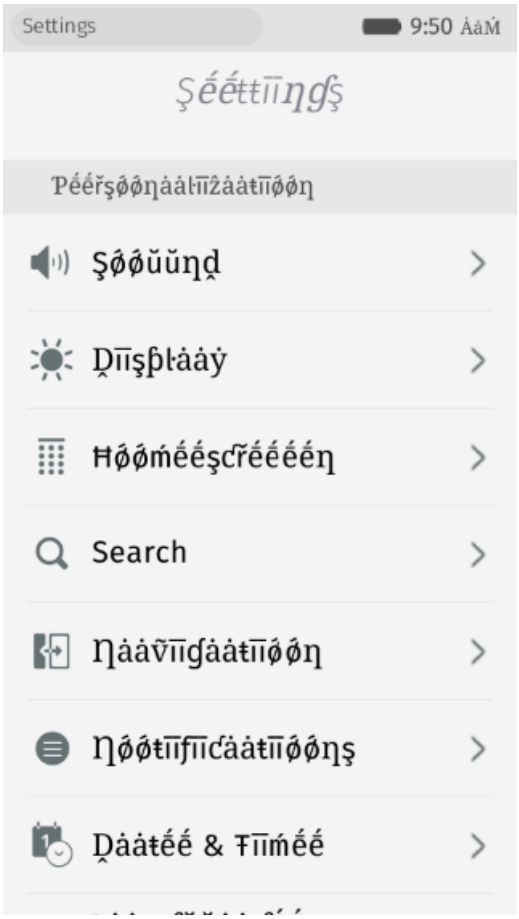
```
{
"en-US" : "English (US)",
"zh-TW" : "□□□□ (□□)"
}
```

Refer to this gist[88] for locale build automation script.

4.8.1 test localization via pseudolocales

You can enable pseudo-localization in your device's developer panel. Then choose one of them in Settings->Languages (they'relocated at the very beginning of the list). After that you may test your patch. Every element that has l10n-enabled will look like mixed code.

[88] https://gist.github.com/timdream/7716684



pseudo locale

There are some best practices[89] about how to perform localization.

4.9 Beyond Build Script

4.9.1 Tools

There are helpful tools located in gaia/tools/ folder. Let's take `png_recompress.sh` for example.

Pick an image named 'bg.png' and run the command:

```
1    $ ./tools/png_recompress.sh -v bg.png
```

It will remove unnecessary metadata and further compress the png so that it may save up to 99.5% of size, which has a very measurable impact on ZIP file size and memory footprint.

[89] https://developer.mozilla.org/en-US/docs/Mozilla/Localization/Localization_best_practices

4.9.2 Per commit tests via Treeherder

Gaia project is hosted on github. It used to use Travis CI[90] to do per-commit continuous integration and pull request test. Now the tests are moved to Mozilla's Treeherder[91] service.

Million of Gaia builds are run via Treeherder[92]. In addition to lint, unit-test, marionette js test, Gaia also runs buildscript test and gaia_ui_tests(marionette python version) on Treeherder.

4.9.3 Run Gaia ui test locally

If you see the `gaia_ui_tests` error on Treeherder and want to reproduce it locally. You can refer to `tests/travis_ci/gaia_ui_tests/`.

Follow `tests/travis_ci/gaia_ui_tests/install` to setup you local test environment:

```
$ virtualenv --no-site-packages ENV
$ source ENV/bin/activate
$ ROOT=$PWD
$ cd tests/python/gaia-ui-tests/
$ python setup.py develop
$ cd $ROOT
$ make b2g
$ DESKTOP_SHIMS=1 NOFTU=1 make
```

The script enables standalone python virtual environment `virtualenv` to run the tests so the related configurations won't affect the entire host system. It fetches new b2g desktop version and then builds a desktop version of Gaia.

Then run `tests/travis_ci/gaia_ui_tests/script`

```
$ export GAIATEST_ACKNOWLEDGED_RISKS=true
$ export GAIATEST_SKIP_WARNING=true
$ root=tests/python/gaia-ui-tests/gaiatest
$ b2g=`find b2g -follow -name "b2g-bin" | tail -n 1`
$ python $root/cli.py --app=b2gdesktop \
  --binary=$b2g \
  --profile=profile \
  --type=b2g \
  --timeout=10000 \
  tests/python/gaia-ui-tests/gaiatest/tests/accessibility/lockscreen/test_a1\
1y_unlock_to_homescreen.py
```

You can enter the following command to exit to normal console:

[90] https://travis-ci.org/

[91] https://developer.mozilla.org/en-US/Firefox_OS/Treeherder

[92] https://treeherder.mozilla.org/#/jobs?repo=gaia-try

```
$ deactivate
```

You can refer to MDN[93] for more information about gaia UI test.

4.10 Reference

- https://developer.mozilla.org/en-US/Firefox_OS/Platform/Gaia/Build_System_Primer
- https://speakerdeck.com/yurenju/gaia-build-system-introduction
- https://developer.mozilla.org/en-US/Firefox_OS/TVs_connected_devices

[93] https://developer.mozilla.org/en-US/Firefox_OS/Platform/Automated_testing/Endurance