Closed devices powered by open source software?
The IoT Paradox.

Peter Hoddie
Vice President Kinoma, Marvell
www.kinoma.com  @phoddie
I’m a software engineer.
Darwin Streaming Server

Apple Releases Darwin 1.0 Open Source

New Mac OS Core and QuickTime Streaming Server Released to Tens of Thousands of Developers

INTERNET WORLD, LOS ANGELES—April 5, 2000—Apple® today announced the release of Darwin 1.0, the advanced operating system core at the heart of Mac OS X, and the release of an updated Darwin Streaming Server. Darwin’s open source model allows the tens of thousands of registered Darwin developers to modify, customize and extend key Apple software, including the modern mach kernel and BSD layers found in Apple’s next generation operating system, Mac OS X.

"The core of Mac OS X is the only mainstream operating system following an open source model," said Philip Schiller, Apple’s vice president of Worldwide Product Marketing. "The new Darwin 1.0 posting includes some of the most advanced operating system technology available, and it’s open to our customers and developers so that we may collaborate on the future of the Mac OS."

The new Darwin kernel is based on FreeBSD and Mach 3.0 technologies and supports the Kernel Extension Developer Kit (KDK) for developing drivers and loadable modules. Darwin 1.0 gives developers access to essential Mac OS X source code. This allows developers to enhance the feature set, performance and quality of Mac OS X products in partnership with Apple engineers. Darwin 1.0 is processor-independent and is built for PowerPC and Intel platforms, enabling Open Source developers to work on Darwin projects on the widest choice of computer systems.

In addition to Darwin 1.0, Apple also announced an update to the Darwin Streaming Server, the open source version of Apple’s QuickTime™ Streaming Server software, which is used to stream high-quality audio and video over the Internet. The Darwin Streaming Server is the only open source, standards-based and free Internet streaming server. This update makes the Darwin Streaming Server more accessible for developers and enhances the platform's ability to send data over the Internet.
Kinoma on GitHub
(https://github.com/Kinoma)
Kinoma Create

Santa Clara, California, United States
Technology

Story | Updates 17 | Comments 39 | Funders 552

Kinoma Create — The JavaScript-Powered Internet of Things Construction Kit

$52,287 USD
raised by 552 people in 1 month

523% funded
No time left

$10,000 USD goal
Flexible Funding

CAMPAIGN CLOSED
This campaign ended on April 16, 2014
Outline of talk

• Lament
  • Complain about IoT being closed
• Pivot
  • Surely it could be open
• Solution
  • Code
Internet of Things is a big deal.
Dear GitHub Copyright Agent:

As explained further below, I am writing to request that you remove proprietary and copyrighted material belonging to Apple Inc. (“Apple”).

I, the undersigned, state UNDER PENALTY OF PERJURY that:

[1] I am a person injured, or an agent authorized to act on behalf of a person injured, by the violation of copyright laws, in particular section 501 of Title 17 of the United States Code, the Copyright Act, or “DMCA”;

[2] I May Be Contacted At:

Name of Injured Party : APPLE INC.
Name and Title: [private], ASSOCIATE
Company: KILPATRICK TOWNSEND AND STOCKTON LLP
Address: TWO EMBARCADERO CENTER, EIGHTH FLOOR

Public Law 105-304
105th Congress
An Act

To amend title 17, United States Code, to implement the World Intellectual Property Organization Copyright Treaty and Performances and Phonograms Treaty, and for other purposes.

Be it enacted by the Senate and House of Representatives of the United States of America in Congress assembled,

SECTION 1. SHORT TITLE.
This Act may be cited as the “Digital Millennium Copyright Act”.

SEC. 2. TABLE OF CONTENTS.

Sec. 1. Short title.
Sec. 2. Table of contents.

TITLE I—WIPO TREATIES IMPLEMENTATION

Be it enacted by the Senate and House of Representatives of the United States of America in Congress assembled,
GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007


Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable (since such users cannot opt out). For more information, see GNU General Public License section 6.

The GNU General Public License does not guarantee that you have sufficient resources to copy the software. Chapter 3, Distribution of a Combined Work, is intended to ensure this.

The GNU General Public License does not make it illegal to accept funds for financial support. It does not mean that you cannot accept such funds.

In other words, you are not required to accept this License, but doing so is strongly encouraged and suggested. If you wish to avoid any potential consequences arising from rejection of this License, you may choose to add other terms to your own license, for example, requiring end-users to accept your terms along with the GNU GPL.

However, do not so modify the License, or choose some other name for it, so as to cause confusion at home to people who would not otherwise know of the General Public License. Thus, public use of the program in a modified version counts as use of it by modification as well.

When you are distributing modified versions of the GNU GPL, you must give the recipients both the source code and the ability to run, copy, modify, distribute, and place modified versions on their public servers. Similarly, source code must be included when you distribute an executable version.

You must make sure that they, too, receive a copy of this License. And you must give them all of the opportunities to copy you also received. If you modify the software, you must clearly inform the users of such change.

You should also get your employer to sign a Corporate Source Code License. Then all the work you did is covered by the GPL. This is especially important if your employer is a well-known company, because it makes it much easier for your employer to convey the GPL effectively.

If the modified program normally reads commands interactively when run, you should also make it possible to run it non-interactively. This requires that you offer all the ordinary command-line options.

Most of these requirements and some additional ones are described in section 2, General Terms of Modification and Distribution.

In summary, what the GPL protects is not the program itself, but users' ability to modify the program. The other rare cases where it may seem inappropriate to apply the GPL are described in section 7, Additional General Terms. These include:

• Easy-to-distribute, low-cost versions
• Open source libraries
• Special and limited use of the program

There are situations when it is useful to have a shorter, simpler license that covers only part of the General Public License.

For example, modified versions of the distribution are not required to include source code.

For programs that fit these conditions, the GPL offers a short form called the GNU Lesser General Public License.

The main difference is that, under the Lesser General Public License, developers do not have to make source code available; they can subject some parts of the program to different licensing terms.

This is intended to allow production of free libraries without the limitation of having to make the entire library free. However, whether it is used on its own or with the General Public License, these versions are covered by this license.

The Lesser General Public License, even though it is shorter than the GPL, is intended to ensure that the entire work is free for all users.

There are other ways to use free software. If you are an individual developer, you may use your own license and retain all your rights.

Another possibility is to run programs of one kind with scripts of another kind under the terms of the General Public License. This is the kind of"
Openness

- Open to all apps.
- Extensible OS.
- Open to some apps
- Not open at all

Computers

Phones

Things
Our Goal

• We can run our code on the devices we own

• Our code can access all major functions of the device

• We will tolerate that…

• …our code runs in a sandbox – just like apps on iOS or Android

• We believe the device owner should define the sandbox boundaries, not the manufacturer.
Some portability needed

- Different hardware runs different operating systems and CPUs
- Let’s not make coding for every IoT device completely different
  - Too much to learn each time
- Let’s use scripts for that code
  - Easier to write
  - More portable
  - More secure (generally)
Making your IoT device scriptable

- Assumption: you have an open IoT device
  - …or hack a closed one. ;)
- Reality: an IoT device is mostly just an application
  - Everything here works for apps too.
- Adding the XS6 JavaScript engine to your application
- Connecting your application’s functionality to scripts
Source on GitHub
(https://github.com/Kinoma)
kinomajs
xs6
doc
includes
makefiles
linux
mac
win
sources
fsk
tool
tools
xsbug
Build the library

• Define “XS6” environment variable with path to source code

• We’ll build the library, xs6lib, first.
  • The default build creates both static and dynamic linking versions of the library

• cd $XS6/makefiles/linux
  make debug -f xslib.make
Build xsbug, the debugger

- Source level debugger in a simple user interface
- Runs on Linux/GTK, Mac OS X, and Windows
- `cd $XS6/makefiles/linux`  
  `make debug -f xsbug.make`
- To run the debugger on a different machine from your target device, set the IP address of the machine running xsbug in XSBUG_HOST on the target device
xsbug on Mac OS X
Configure the runtime

• xs6 is designed to run on many different operating systems

• Good news: ports already exist for most common platforms including Linux, Mac OS X, Windows, Android, and iOS

• To configure xs6 for a different host, modify only these files:
  • xs6Host.c - essential platform interfaces for the virtual machine
  • xs6Platform.h - bindings for POSIX APIs (malloc, strlen, etc.)
  • xs6Platform.c - implementations of Posix APIs for non-POSIX hosts
Running scripts from the command line

- Here’s our Hello World

```javascript
/* test.js */
trace("Hello, world!\n");
```

- Here’s how we invoke it:

  ./test test.js
Running scripts from the command line

/* test.c */
#include <xs.h>
int main(int argc, char* argv[])
{
    xsCreation creation = {
        128 * 1024 * 1024,  /* initial chunk size */
        16 * 1024 * 1024,   /* incremental chunk size */
        8 * 1024 * 1024,    /* initial heap slot count */
        1 * 1024 * 1024,    /* incremental heap slot count */
        4 * 1024,           /* stack slot count */
        12 * 1024,          /* key slot count */
        1993,               /* name modulo */
        127                 /* symbol modulo */
    };
    xsMachine* machine = xsCreateMachine(&creation, NULL,"my virtual machine", NULL);
    xsBeginHost(machine);
    xsRunProgram(argv(1));
    xsEndHost(machine);
    xsDeleteMachine(machine);
    return 0;
}
JavaScript is a sandbox

• The language defines a small number of objects
  • Boolean, Number, String, Array, Date, RegExp, etc.

• Built-in objects have no access to operating system services
  • No files, timers, sockets, environment variables

• You decide what scripts can do by adding objects and functions
  • Be generous!
Reading environment variables

• To allow a script to do this

  ```c
  trace(getenv("XS6") + "\n");
  trace(getenv("XSBUG_HOST") + "\n");
  ```

• Add the getenv function to the VM in the tool

  ```c
  xsResult = xsNewHostFunction(xs_getenv, 1);
  xsSet(xsGlobal, xsID("getenv"), xsResult);
  ```

• Implements xs_getenv in C in the tool

  ```c
  void xs_getenv(xsMachine* the)
  {
    xsStringValue result = getenv(xsToString(xsArg(0)));
    if (result)
      xsResult = xsString(result);
  }
  ```
Notifications

• When something changes, we want to tell the script
  • Light switch turned on, network up/down
• C code can call into the virtual machine
  • Beware: JavaScript is single threaded.
• Always call your xs6 virtual machine from the same thread.
Notifications –1

• To allow a script to do this

```javascript
device.onValueChanged = function(name, value) {
    trace("onValueChanged" + name + " " + value + "\n");
};
```

• Keep a reference to the VM in your application

```javascript
machine = xsCreateMachine(&creation, NULL, "test", NULL);
```

• Add the “device” object to the virtual machine as a global

```javascript
xsBeginHost(machine);
xsResult = xsNewInstanceOf(xsObjectPrototype);
xsSet(xsGlobal, xsID("device"), xsResult);
xsEndHost(machine);
```
• When your C code is notified of a change, pass it along to the script

```c
void c_onValueChanged(char* name, double value)
{
    xsBeginHost(machine);
    xsResult = xsGet(xsGlobal, xsID("device"));
    xsCall2(xsResult, xsID("onValueChanged"), xsString(name), xsNumber(value));
    xsEndHost(machine);
}
```
setTimeout

- setTimeout is not part of JavaScript language
- We can implement it in xs6
- To allow a script to do this
  ```javascript
  setTimeout(function(it) {trace(it + "\n");}, 1000, "WOW!");
  ```
- Or in ES6 syntax
  ```javascript
  setTimeout(it => {trace(it + "\n")}, 1000, "WOW!");
  ```
• Implement setTimeout

    typedef struct xs_setTimeout_data {
        xsMachine* machine;
        xsSlot function;
        xsSlot argument;
    };

    xs_setTimeout(xsMachine* the) {
        struct xs_setTimeout_data* data = malloc(sizeof(xs_setTimeout_data));
        data->machine = the;
        data->function = xsArg(0);
        xsRemember(data->function);
        data->reference = xsArg(2);
        xsRemember(data->reference);
        c_setTimeout(xs_setTimeout_callback, xsToNumber(xsArg(1)), data);
    }
Implementing setTimeout – 2

- When native timer fires, call the script function

```c
void xs_setTimeout_callback(void* it)
{
    xs_setTimeout_data* data = it;
    xsBeginHost(data->machine);
    xsCallFunction1(xsAccess(data->function), xsUndefined, xsAccess(data->argument))
    xsForget(data->argument);
    xsForget(data->function);
    xsEndHost(data->machine);
    free(data);
}
```
• Sounds interesting? Try it.
  
• https://github.com/kinoma/kinomajs

• Making your device (or app) JavaScript-able isn’t difficult

• You may even discover that it is easier for you to add new features through scripts
Resolving the IoT paradox

• Open source can lead to more open devices…
  • …just as it has enabled countless closed device.
• It goes beyond making code open
  • Create code that implements the future we want to live in
  • Show it is possible, useful, desired by customers
• Then the manufacturers will listen
Closed devices powered by open source software?
The IoT Paradox.

Peter Hoddie
Vice President Kinoma, Marvell
www.kinoma.com  @phoddie