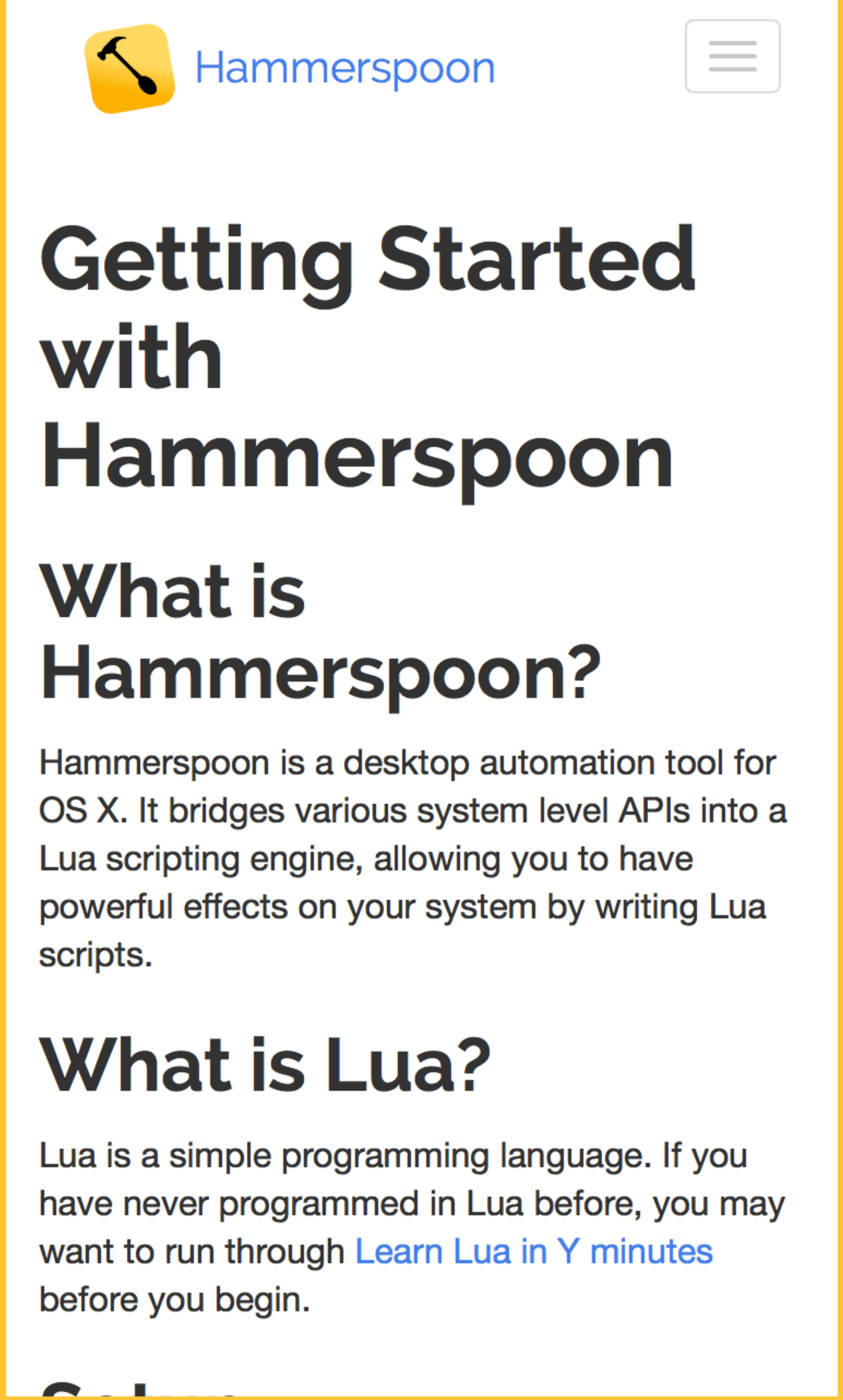
# Hammerspoon

## Staggeringly powerful desktop automation

# Hammerspoon

## Staggeringly powerful desktop automation



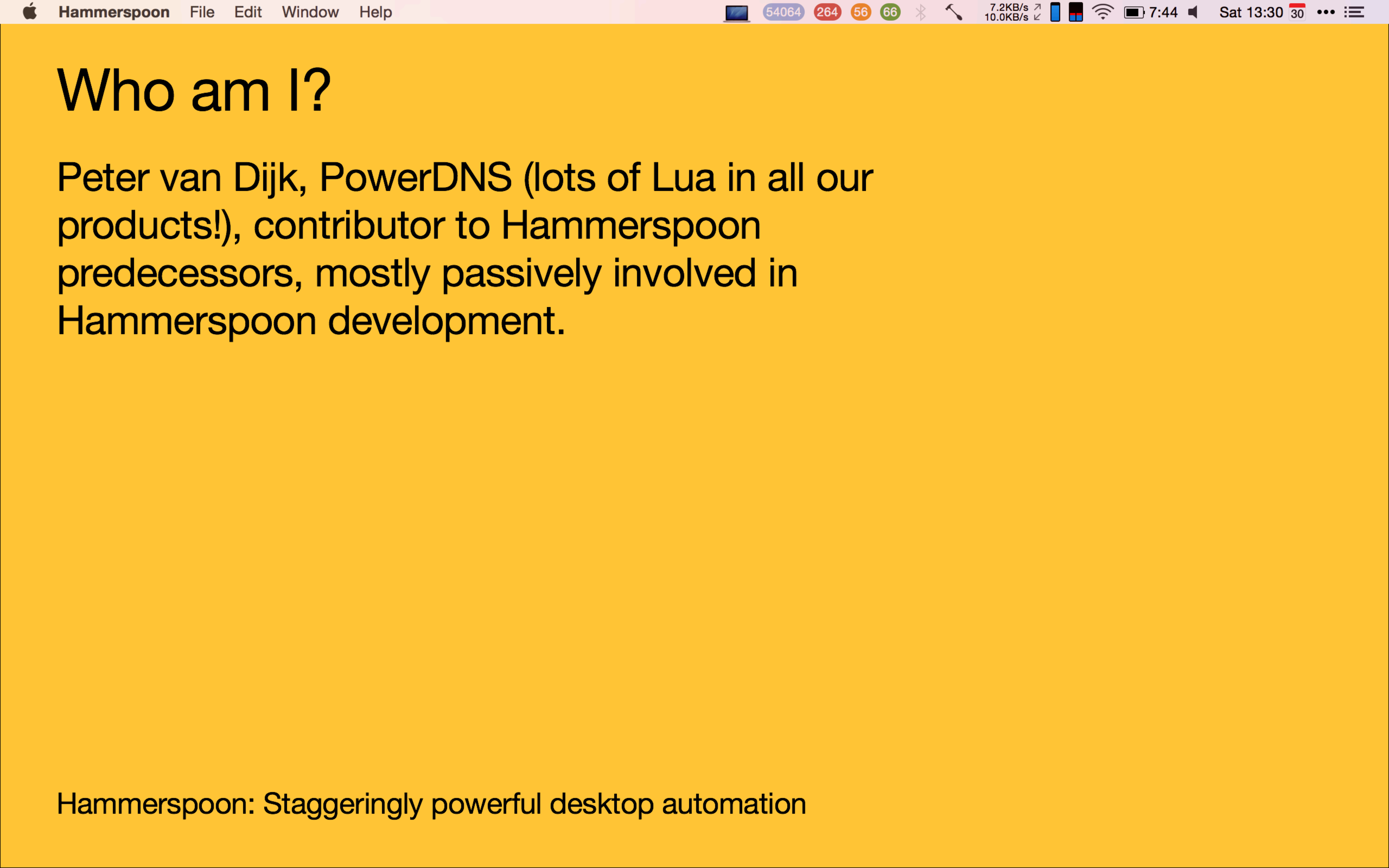
# Getting Started with Hammerspoon

## What is Hammerspoon?

Hammerspoon is a desktop automation tool for OS X. It bridges various system level APIs into a Lua scripting engine, allowing you to have powerful effects on your system by writing Lua scripts.

## What is Lua?

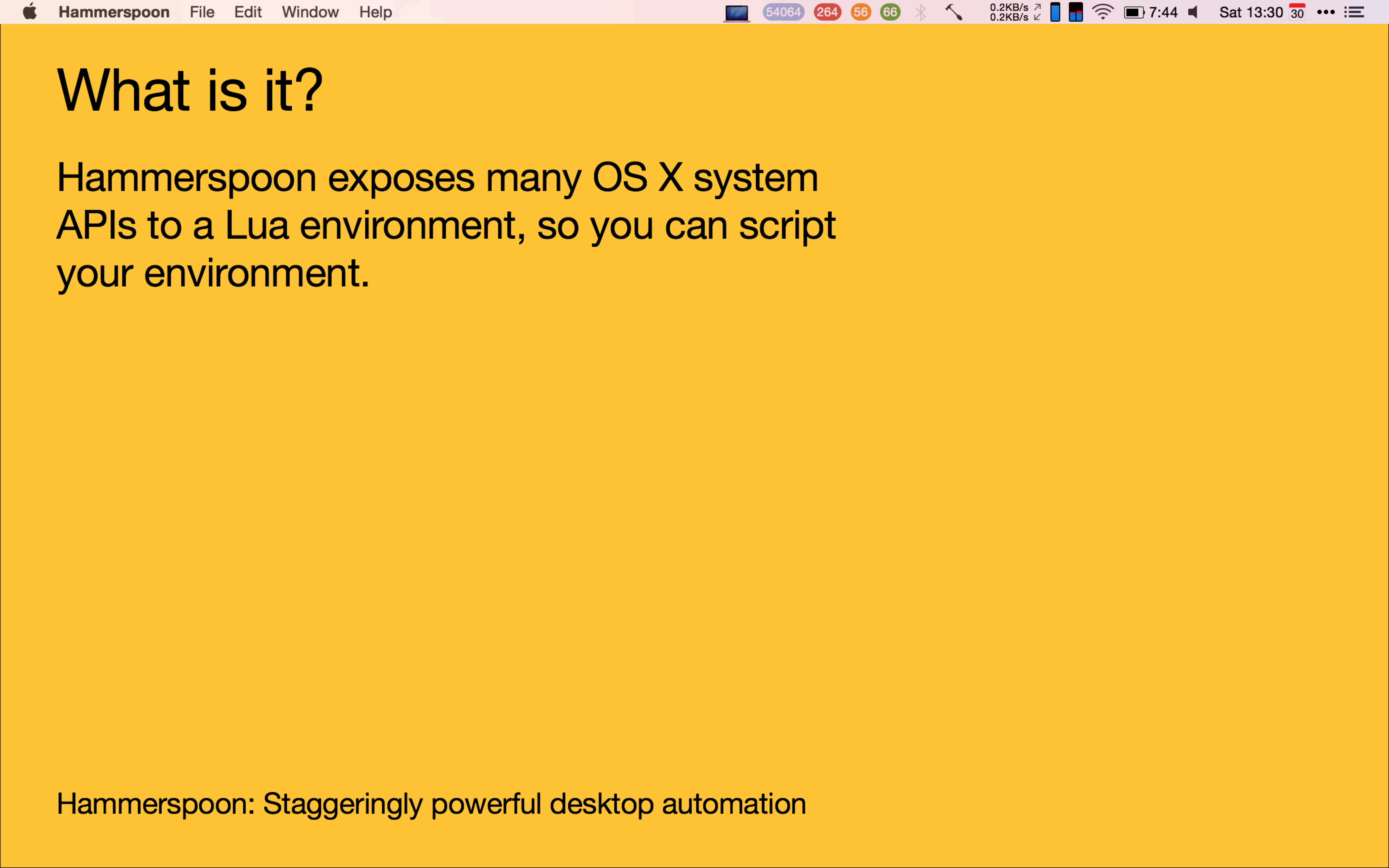
Lua is a simple programming language. If you have never programmed in Lua before, you may want to run through Learn Lua in Y minutes before you begin.



# Who am I?

Peter van Dijk, PowerDNS (lots of Lua in all our products!), contributor to Hammerspoon predecessors, mostly passively involved in Hammerspoon development.

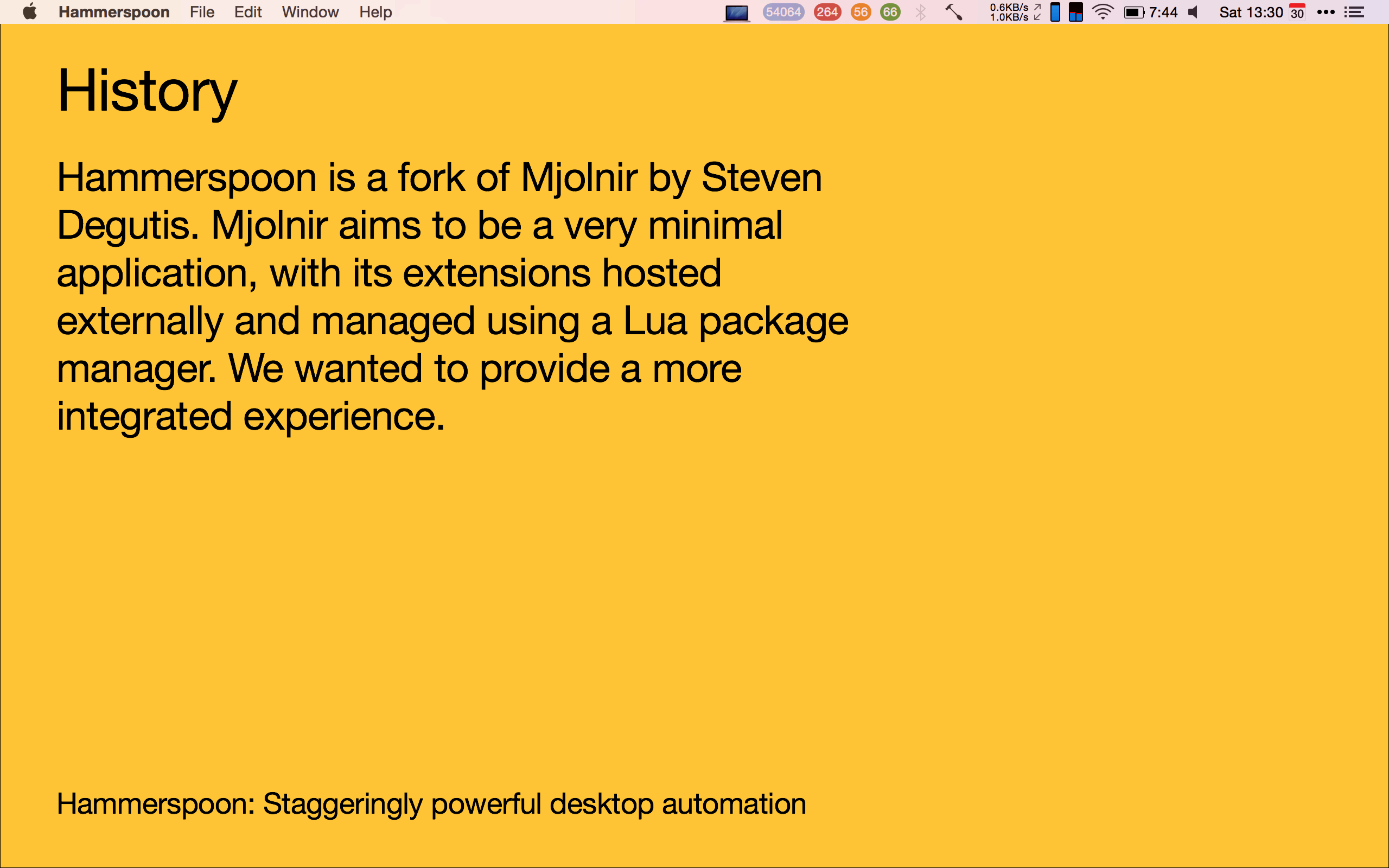
Hammerspoon: Staggeringly powerful desktop automation



# What is it?

Hammerspoon exposes many OS X system APIs to a Lua environment, so you can script your environment.

Hammerspoon: Staggeringly powerful desktop automation

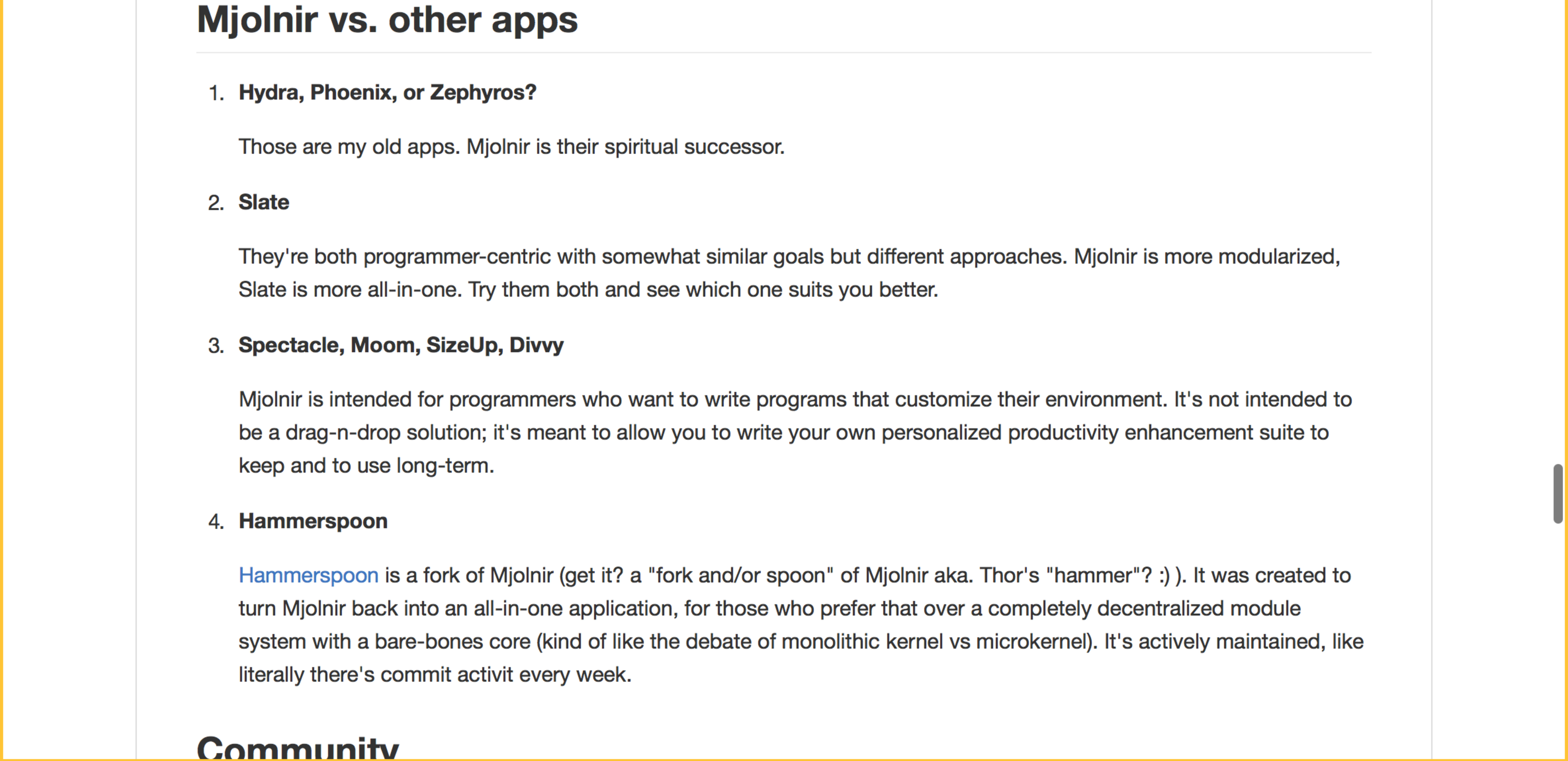


# History

Hammerspoon is a fork of Mjolnir by Steven Degutis. Mjolnir aims to be a very minimal application, with its extensions hosted externally and managed using a Lua package manager. We wanted to provide a more integrated experience.

Hammerspoon File Edit Window Help 54064 264 56 66 16.7KB/s 209.1KB/s 7:44 Sat 13:31 30

# A comparison



## Mjolnir vs. other apps

1. **Hydra, Phoenix, or Zephyros?**

   Those are my old apps. Mjolnir is their spiritual successor.

2. **Slate**

   They're both programmer-centric with somewhat similar goals but different approaches. Mjolnir is more modularized, Slate is more all-in-one. Try them both and see which one suits you better.

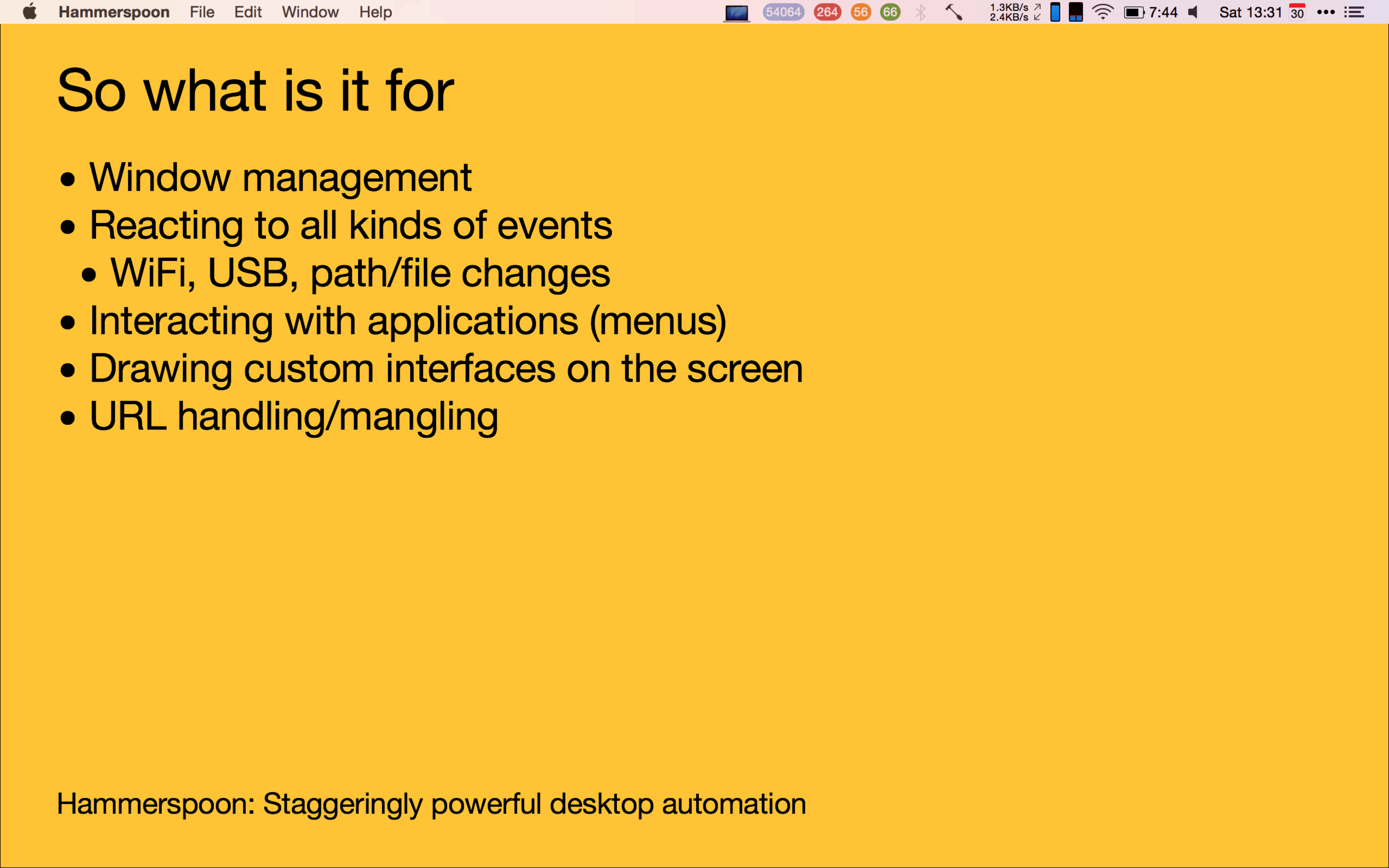
3. **Spectacle, Moom, SizeUp, Divvy**

   Mjolnir is intended for programmers who want to write programs that customize their environment. It's not intended to be a drag-n-drop solution; it's meant to allow you to write your own personalized productivity enhancement suite to keep and to use long-term.

4. **Hammerspoon**

   Hammerspoon is a fork of Mjolnir (get it? a "fork and/or spoon" of Mjolnir aka. Thor's "hammer"? :) ). It was created to turn Mjolnir back into an all-in-one application, for those who prefer that over a completely decentralized module system with a bare-bones core (kind of like the debate of monolithic kernel vs microkernel). It's actively maintained, like literally there's commit activit every week.

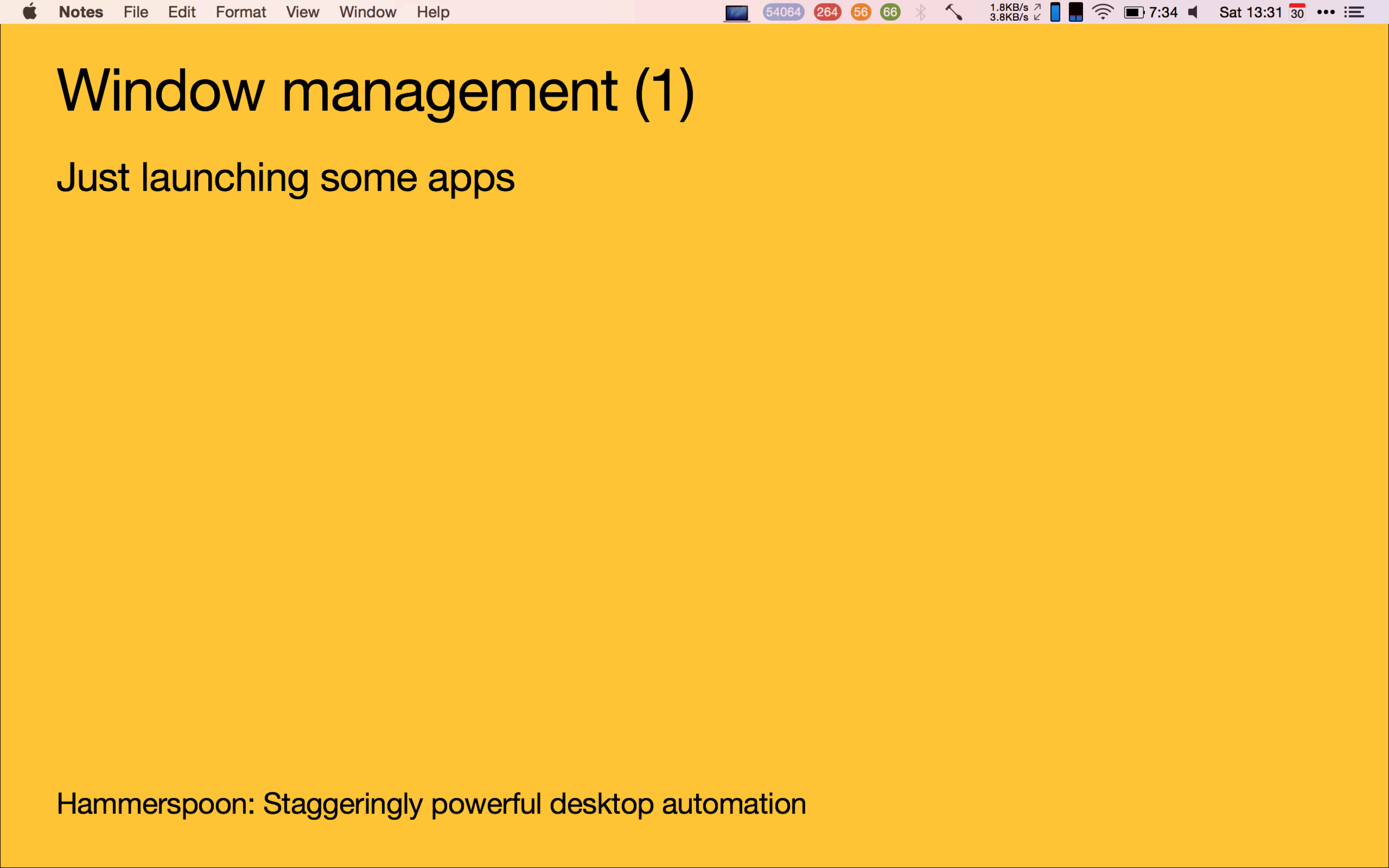
## Community



# So what is it for

- Window management
- Reacting to all kinds of events
  - WiFi, USB, path/file changes
- Interacting with applications (menus)
- Drawing custom interfaces on the screen
- URL handling/mangling

Hammerspoon: Staggeringly powerful desktop automation



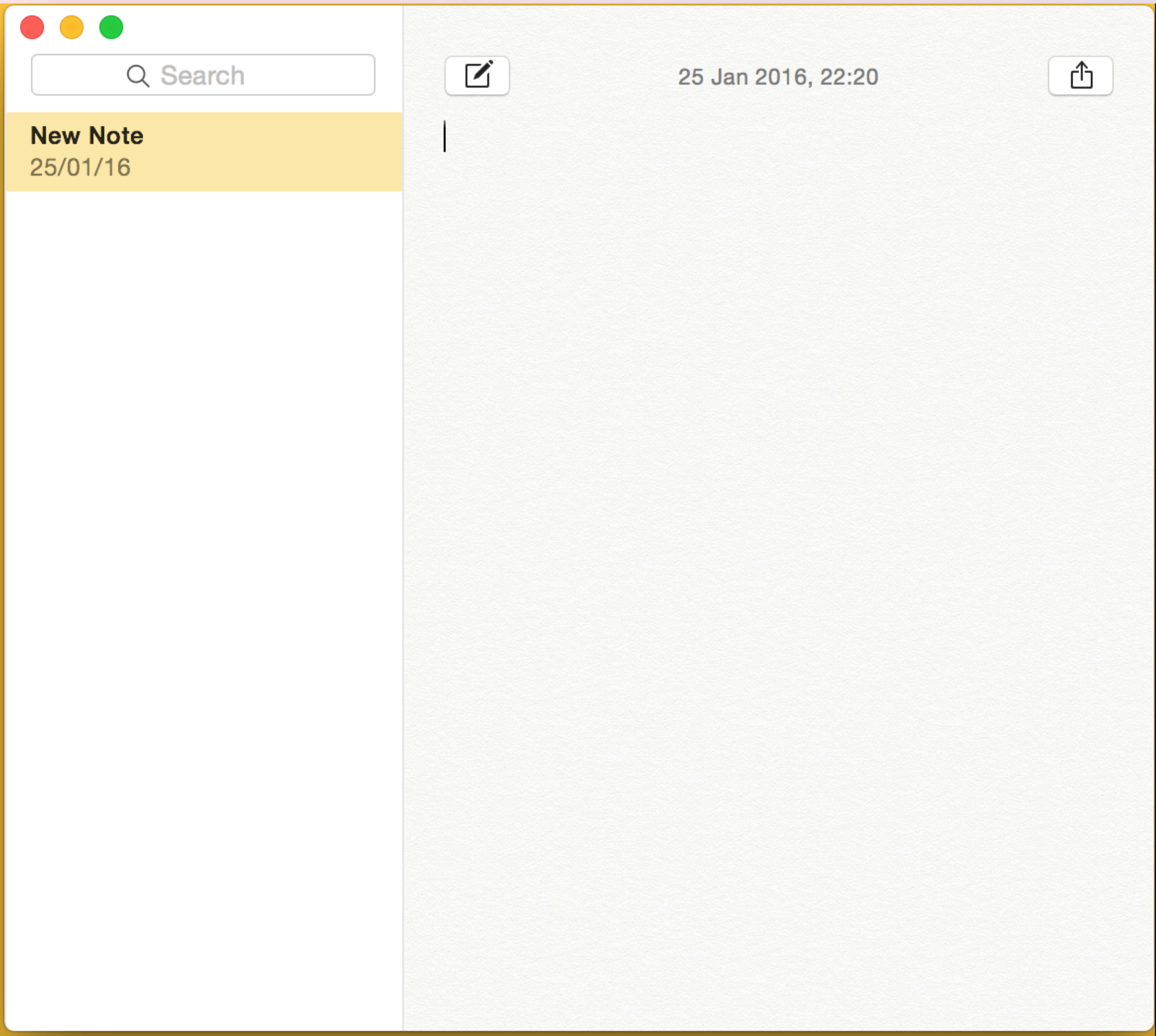
# Window management (1)

Just launching some apps

Hammerspoon: Staggeringly powerful desktop automation

# Window management (1)

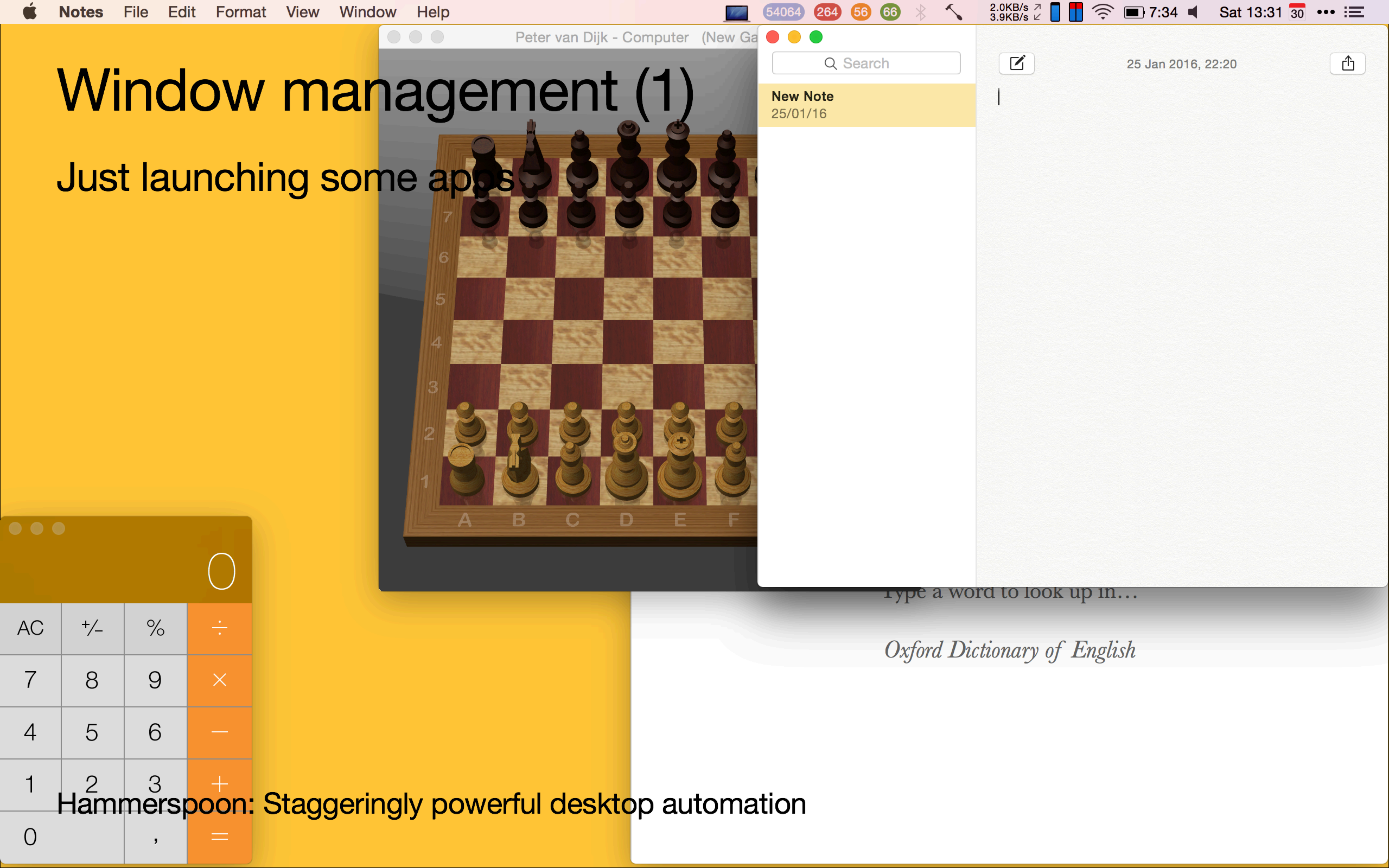
Just launching some apps

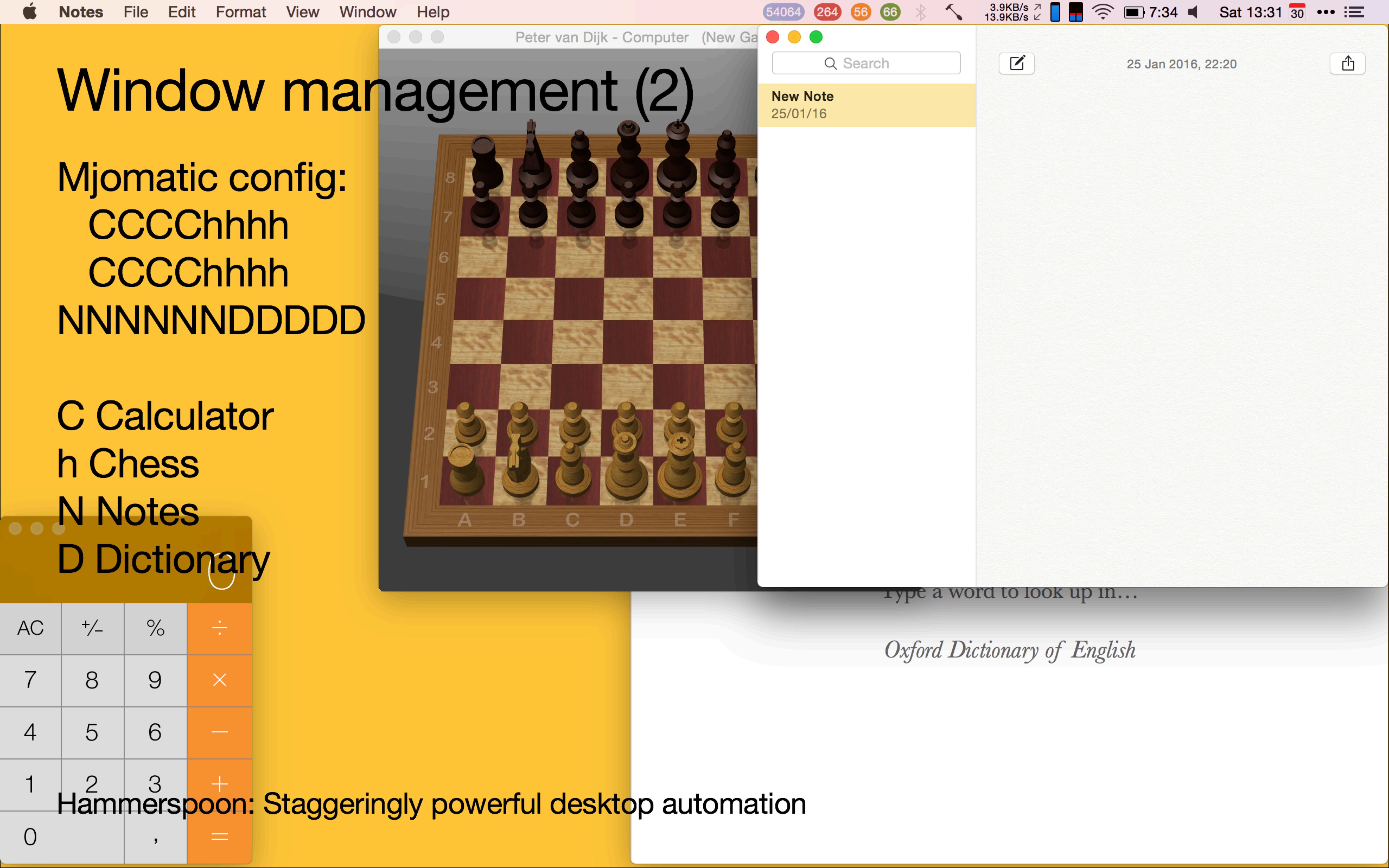


Hammerspoon: Staggeringly powerful desktop automation

# Window management (1)

Just launching some apps





# Window management (2)

Mjomatic config:

```
   CCCChhhh
   CCCChhhh
NNNNNNDDDD
```

C Calculator
h Chess
N Notes
D Dictionary

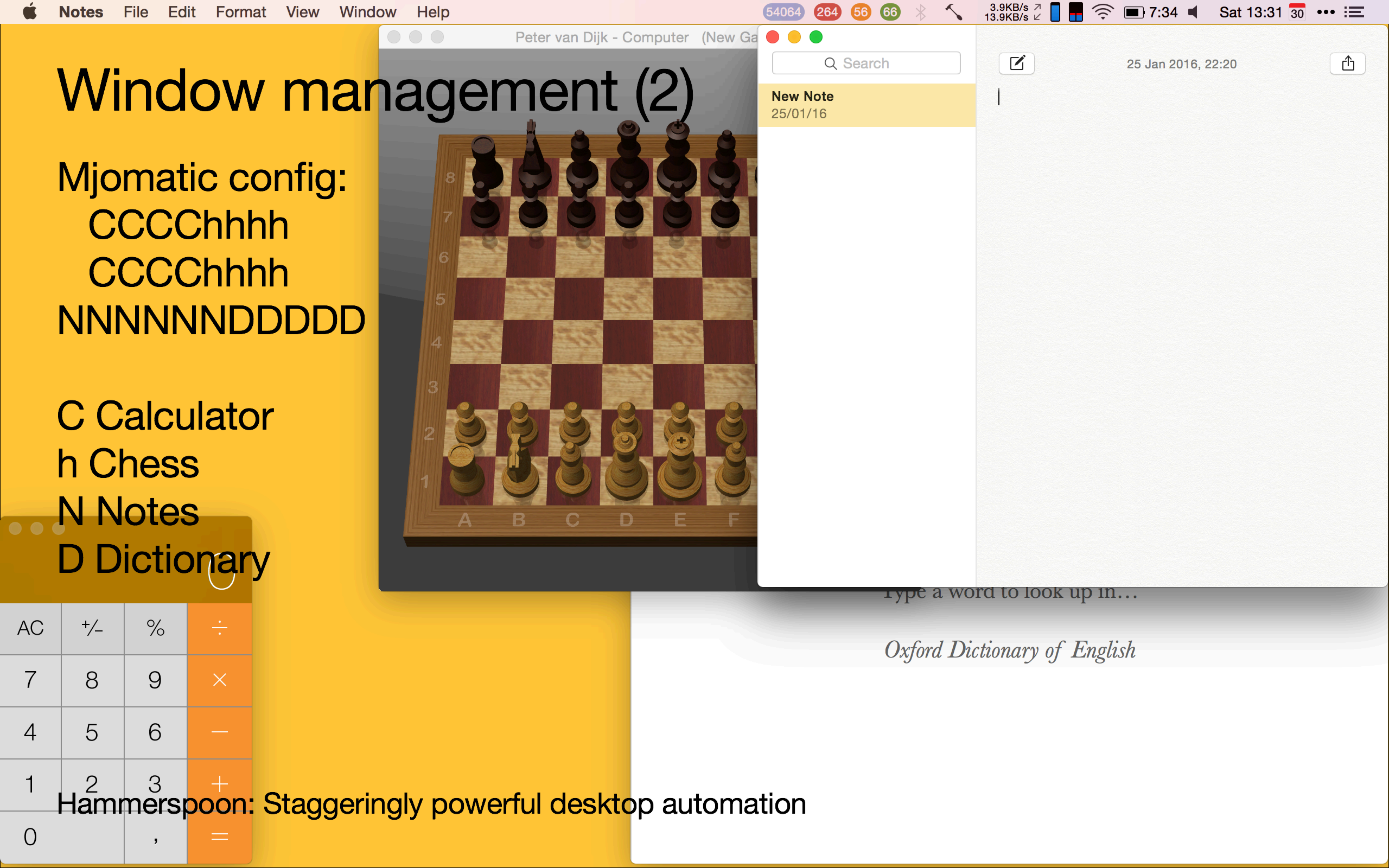
Hammerspoon: Staggeringly powerful desktop automation

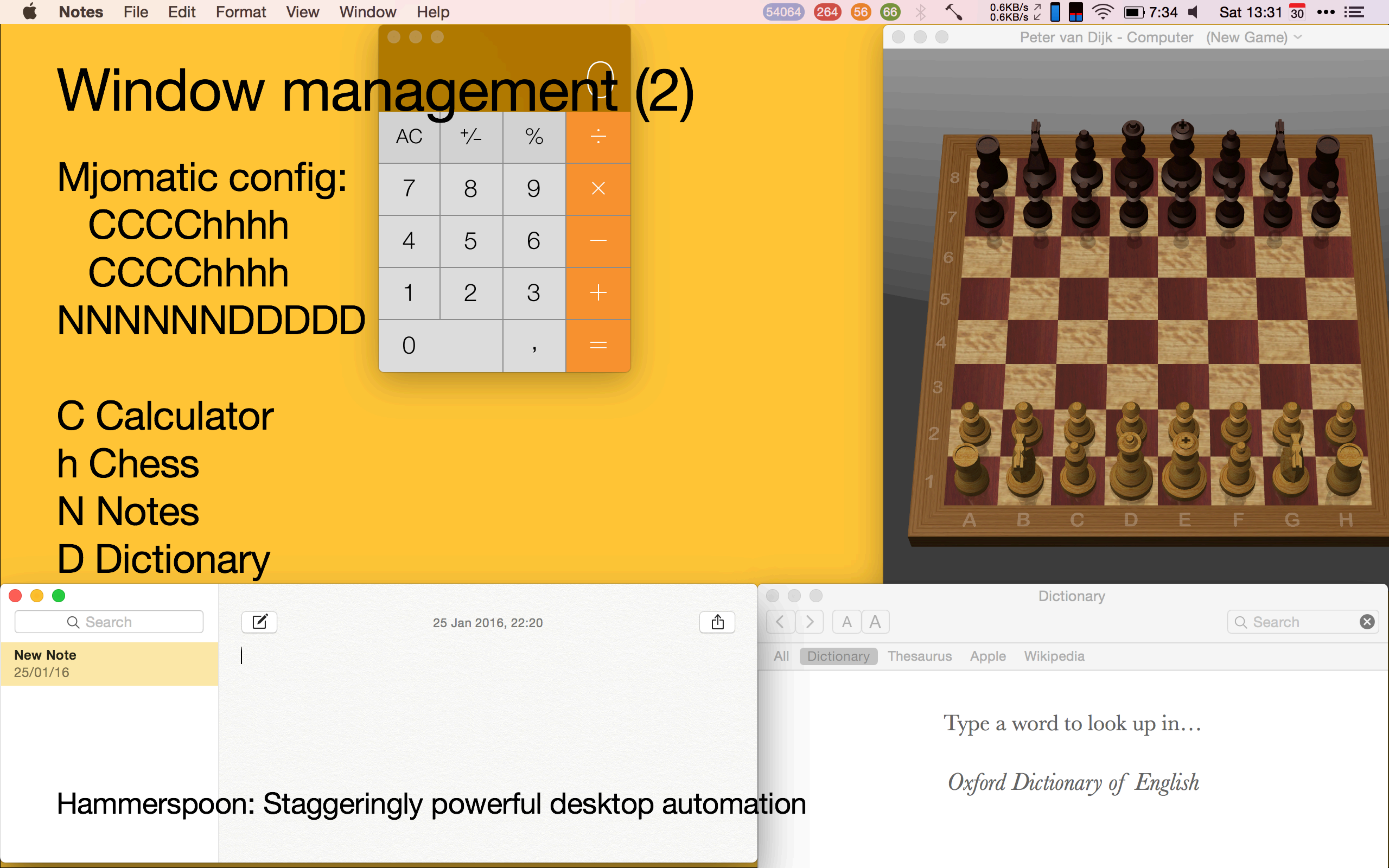
# Window management (2)

Mjomatic config:

```
  CCCChhhh
  CCCChhhh
NNNNNNDDDD
```

C Calculator
h Chess
N Notes
D Dictionary





# Window management (2)

Mjomatic config:

```
  CCCChhhh
  CCCChhhh
NNNNNNDDDD
```

C Calculator

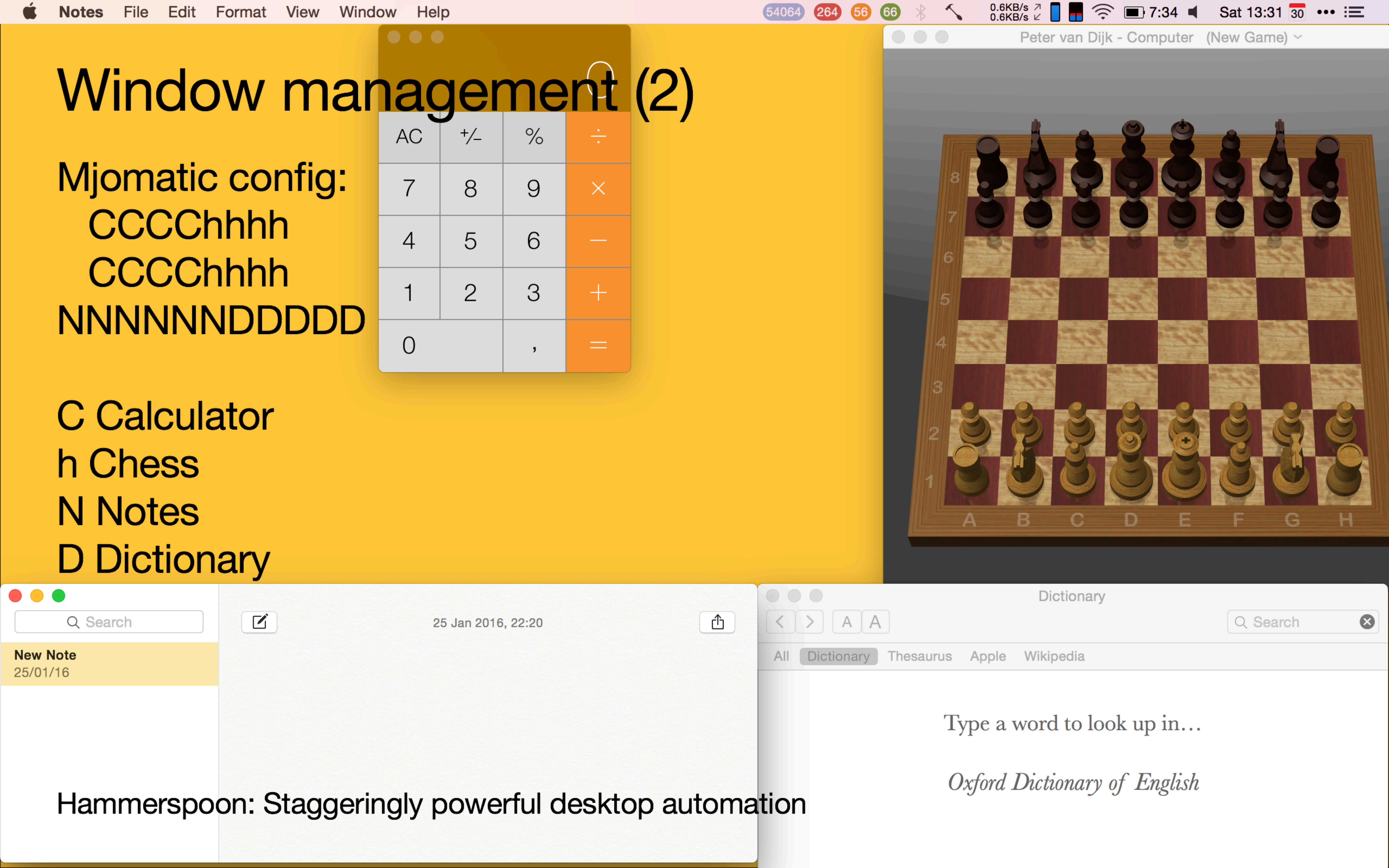
h Chess

N Notes

D Dictionary

Type a word to look up in…

*Oxford Dictionary of English*



25 Jan 2016, 22:20

Type a word to look up in…

*Oxford Dictionary of English*

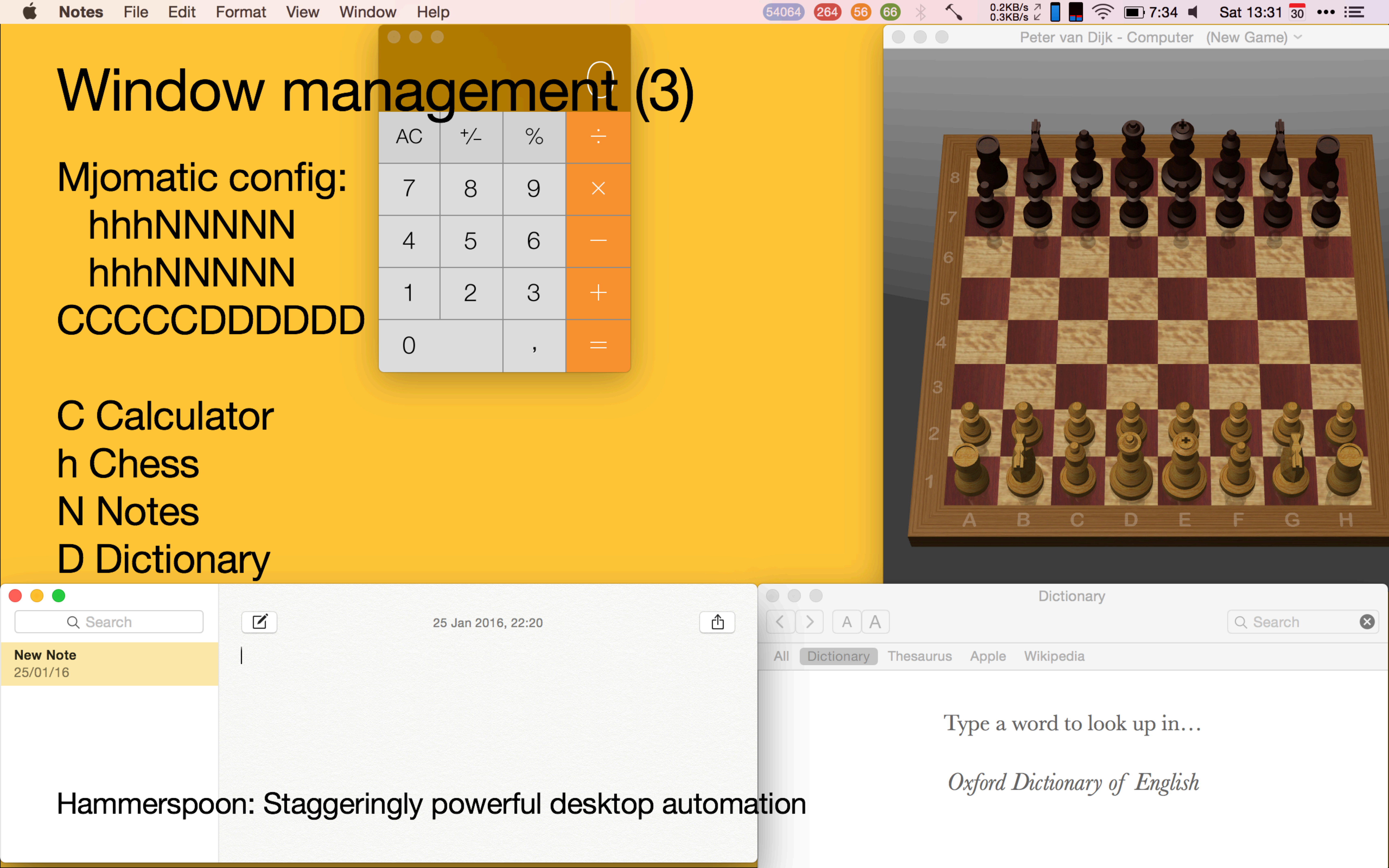
# Window management (2)

Mjomatic config:

```
  CCCChhhh
  CCCChhhh
NNNNNNDDDD
```

C Calculator
h Chess
N Notes
D Dictionary

Hammerspoon: Staggeringly powerful desktop automation



# Window management (3)

Mjomatic config:

```
   hhhNNNNN
   hhhNNNNN
CCCCCDDDDD
```

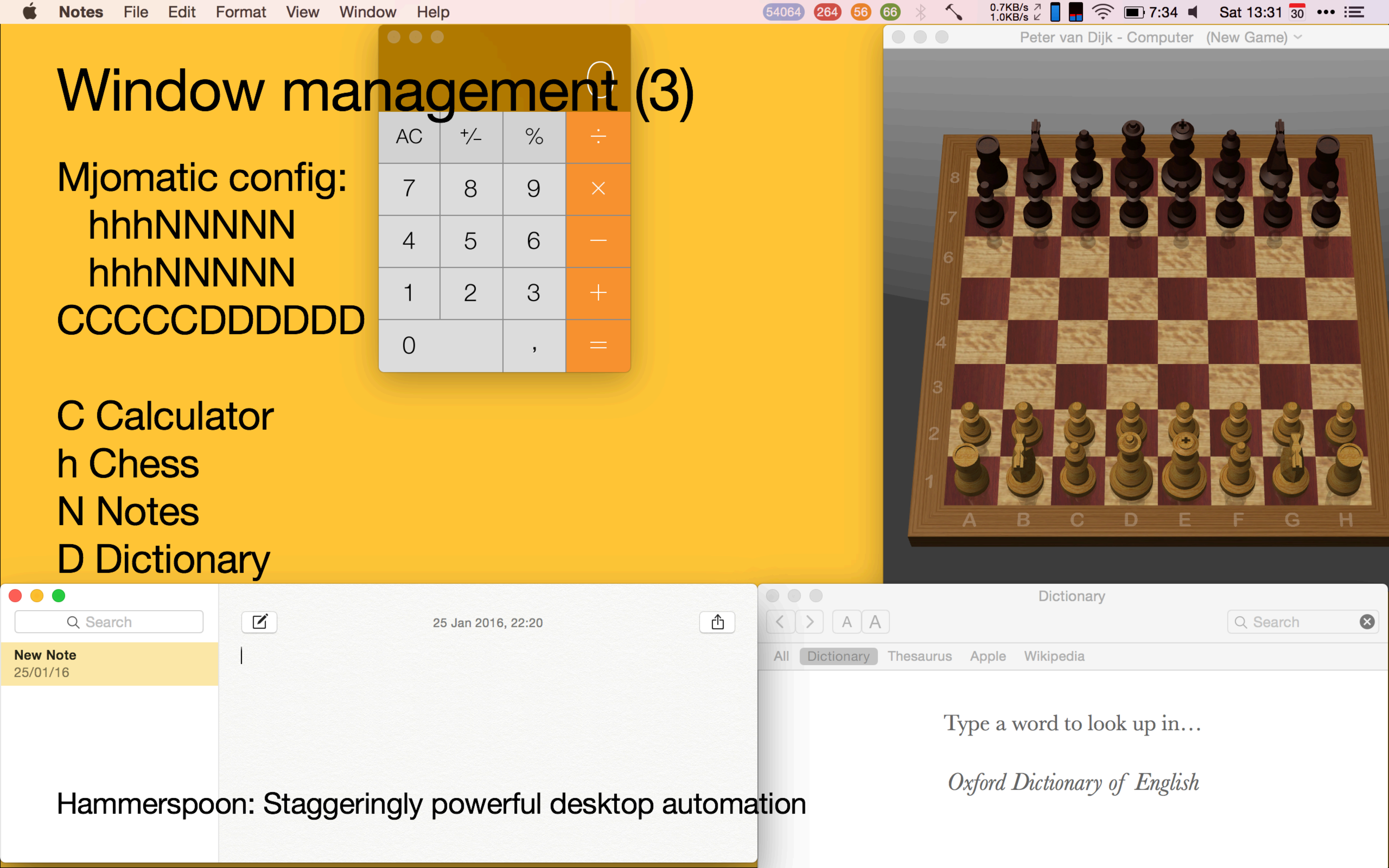
C Calculator

h Chess

N Notes

D Dictionary

Hammerspoon: Staggeringly powerful desktop automation

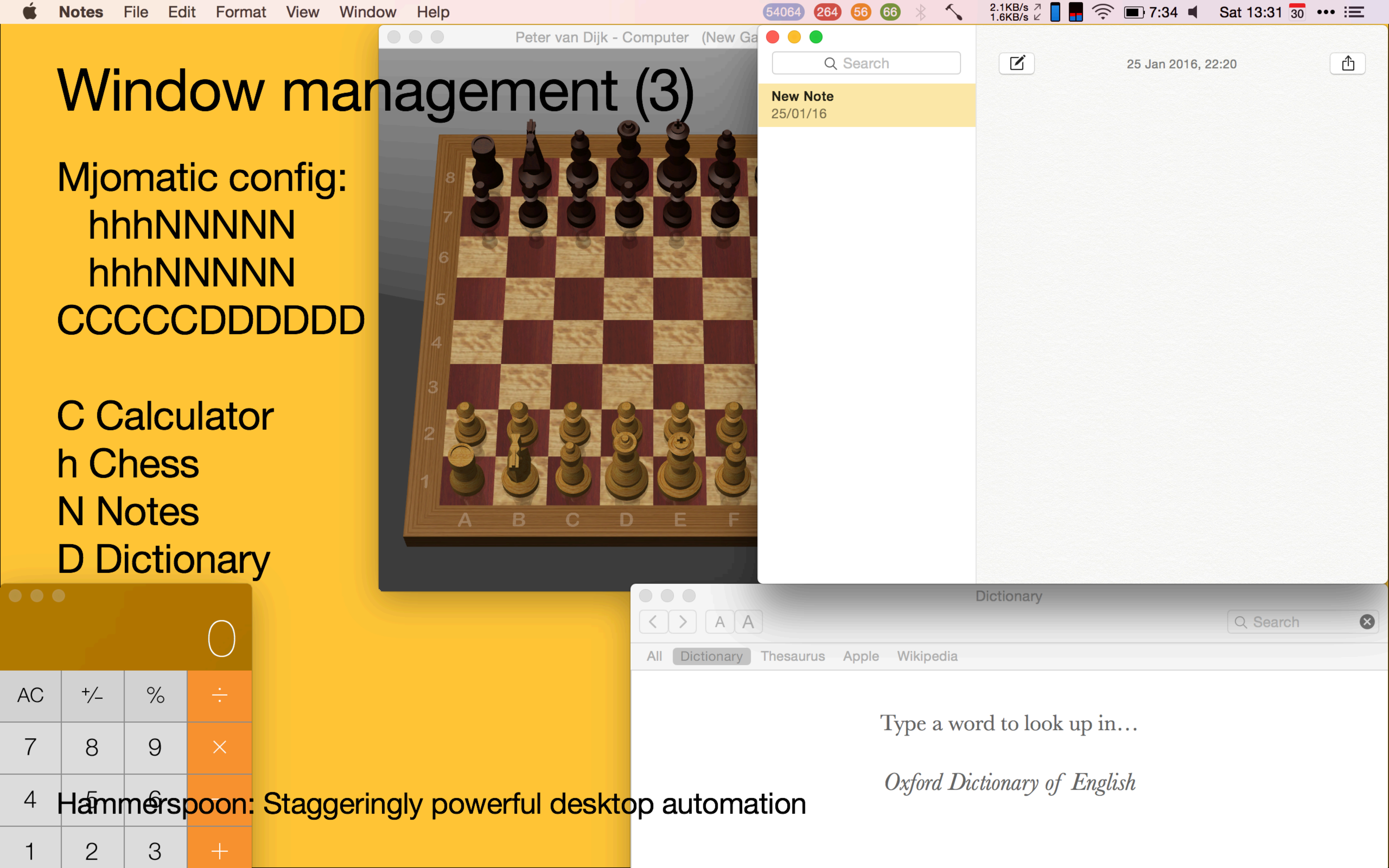


# Window management (3)

```
Mjomatic config:
  hhhNNNNN
  hhhNNNNN
CCCCCDDDDD
```

C Calculator  
h Chess  
N Notes  
D Dictionary

Hammerspoon: Staggeringly powerful desktop automation



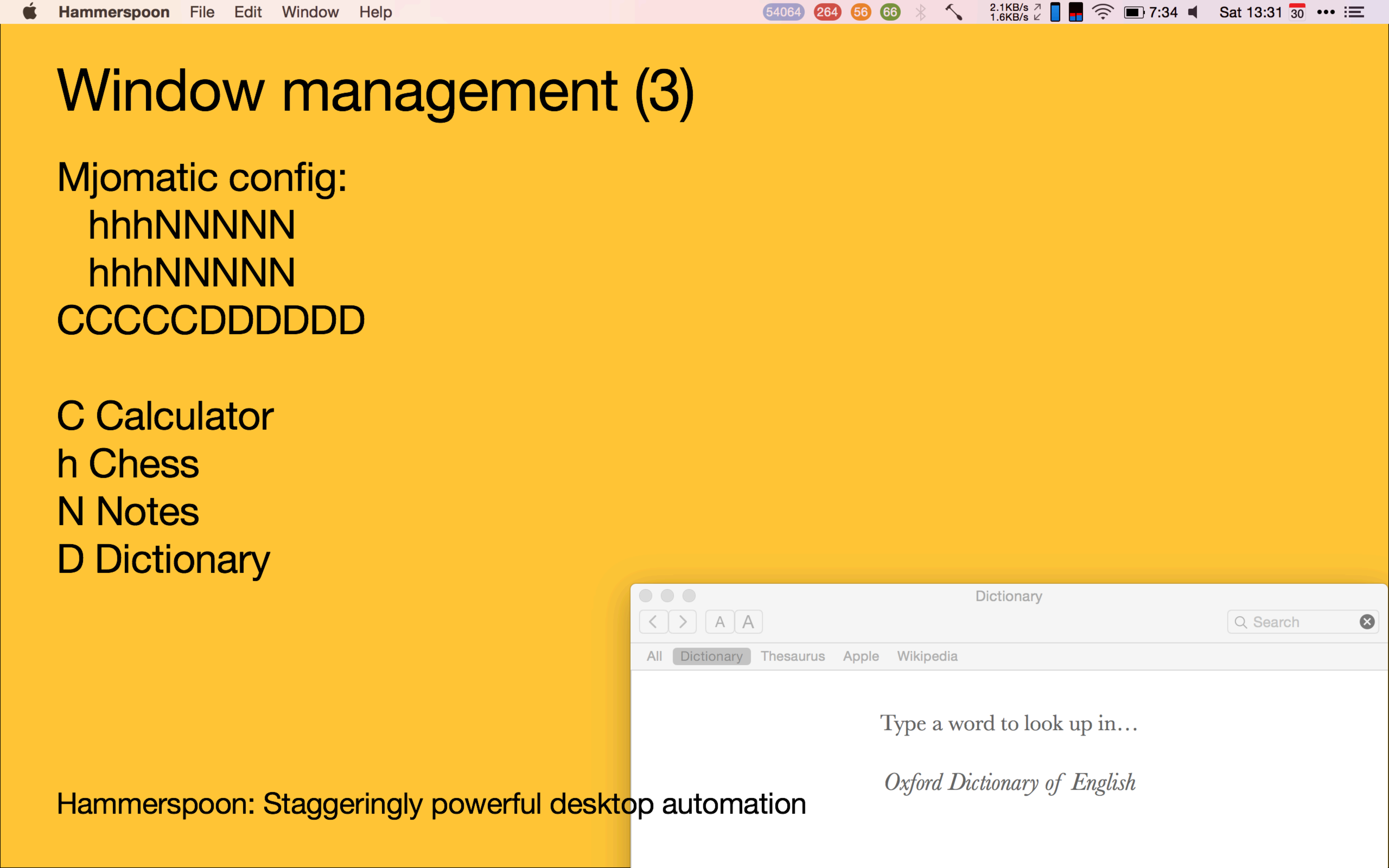
# Window management (3)

Mjomatic config:

```
  hhhNNNNN
  hhhNNNNN
CCCCCDDDDDD
```

C Calculator
h Chess
N Notes
D Dictionary

Hammerspoon: Staggeringly powerful desktop automation



# Window management (3)

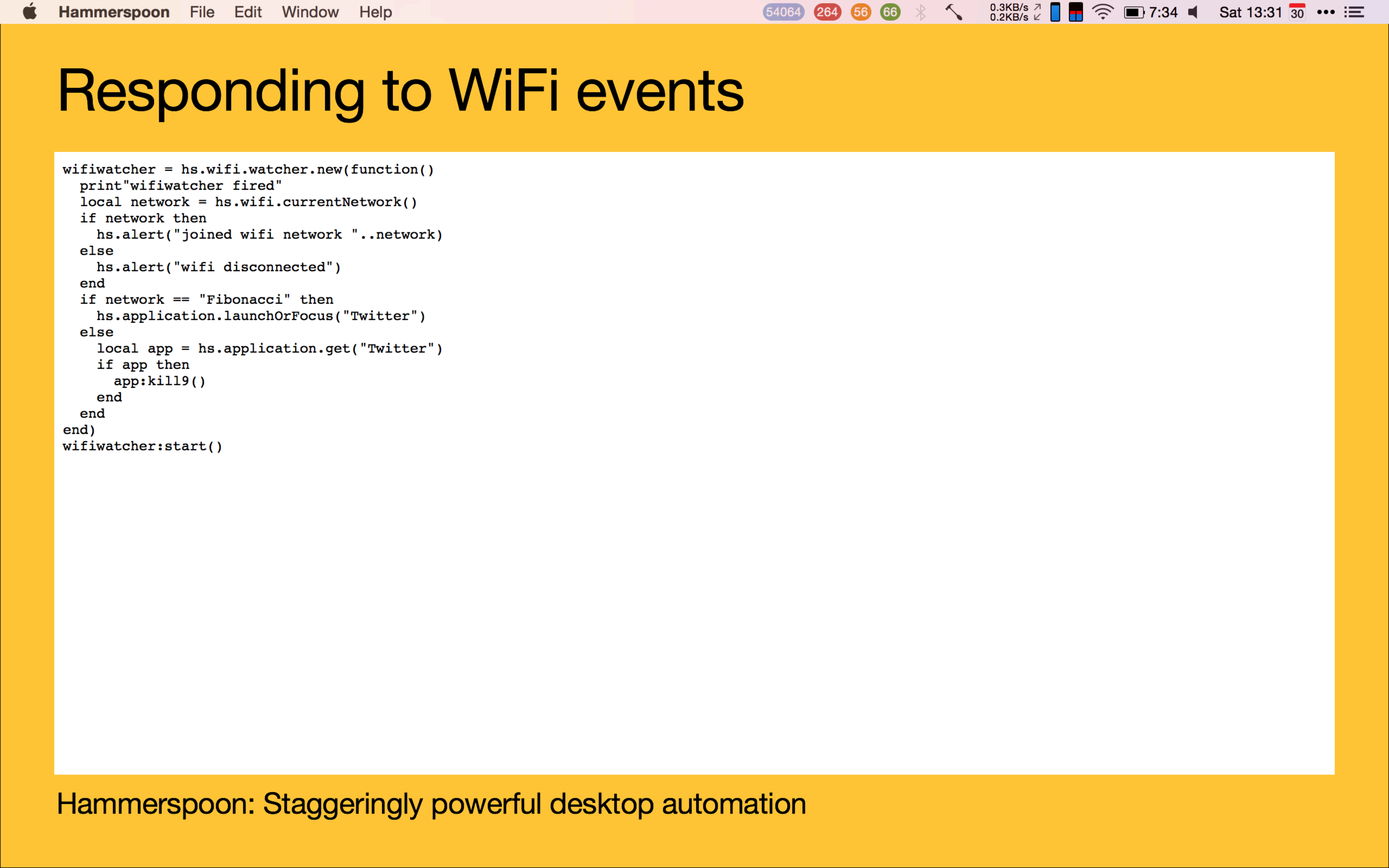
Mjomatic config:

```
   hhhNNNNN
   hhhNNNNN
CCCCCDDDDD
```

C Calculator  
h Chess  
N Notes  
D Dictionary

Type a word to look up in…

*Oxford Dictionary of English*

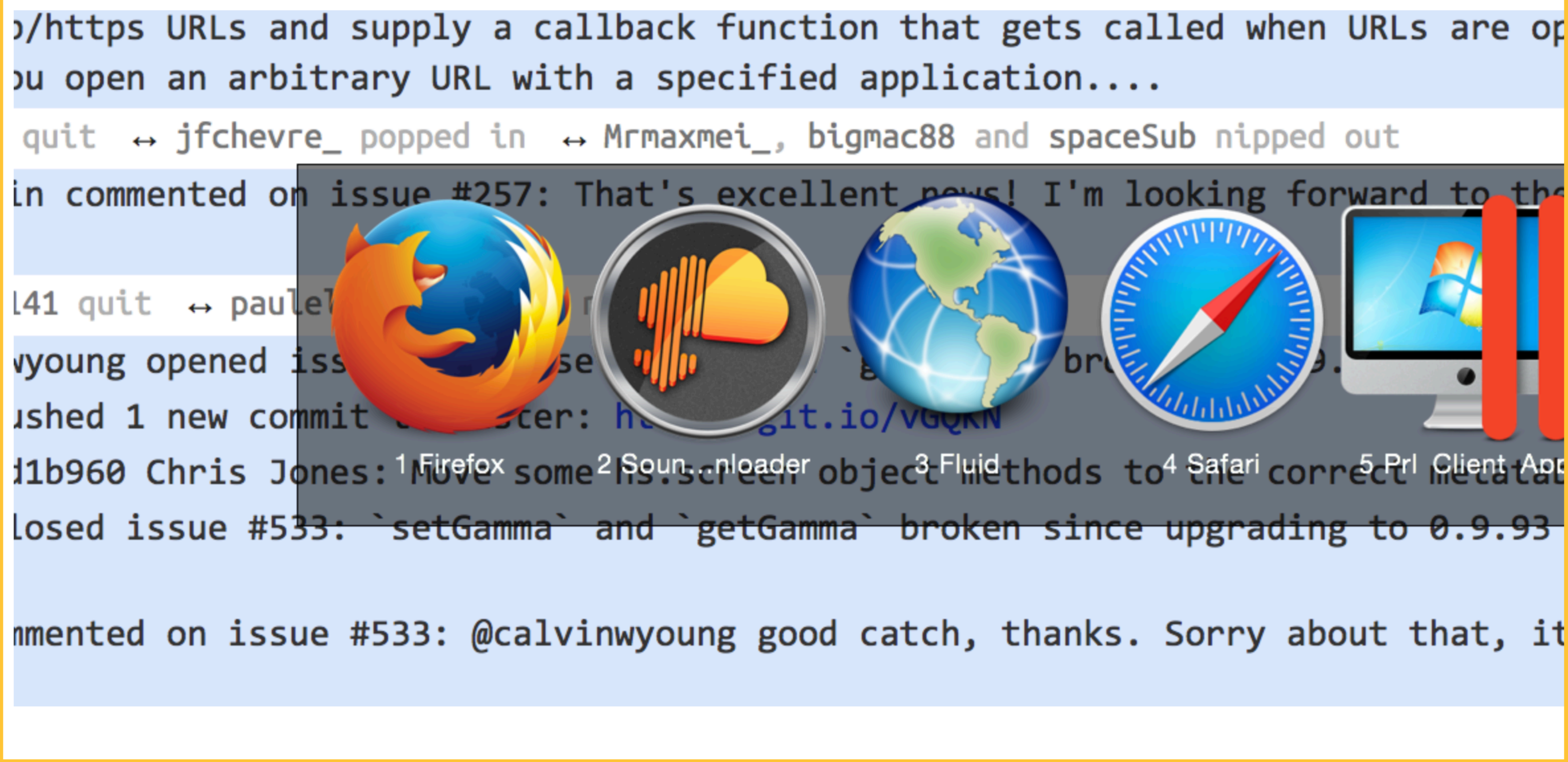


# Responding to WiFi events

```
wifiwatcher = hs.wifi.watcher.new(function()
  print"wifiwatcher fired"
  local network = hs.wifi.currentNetwork()
  if network then
    hs.alert("joined wifi network "..network)
  else
    hs.alert("wifi disconnected")
  end
  if network == "Fibonacci" then
    hs.application.launchOrFocus("Twitter")
  else
    local app = hs.application.get("Twitter")
    if app then
      app:kill9()
    end
  end
end)
wifiwatcher:start()
```

Hammerspoon: Staggeringly powerful desktop automation

# Handling URL events



/https URLs and supply a callback function that gets called when URLs are op
u open an arbitrary URL with a specified application....

quit ↔ jfchevre_ popped in ↔ Mrmaxmei_, bigmac88 and spaceSub nipped out

in commented on issue #257: That's excellent news! I'm looking forward to the

141 quit ↔ paule

young opened iss se ` g br 9.

ushed 1 new commit ter: h git.io/vGQKN

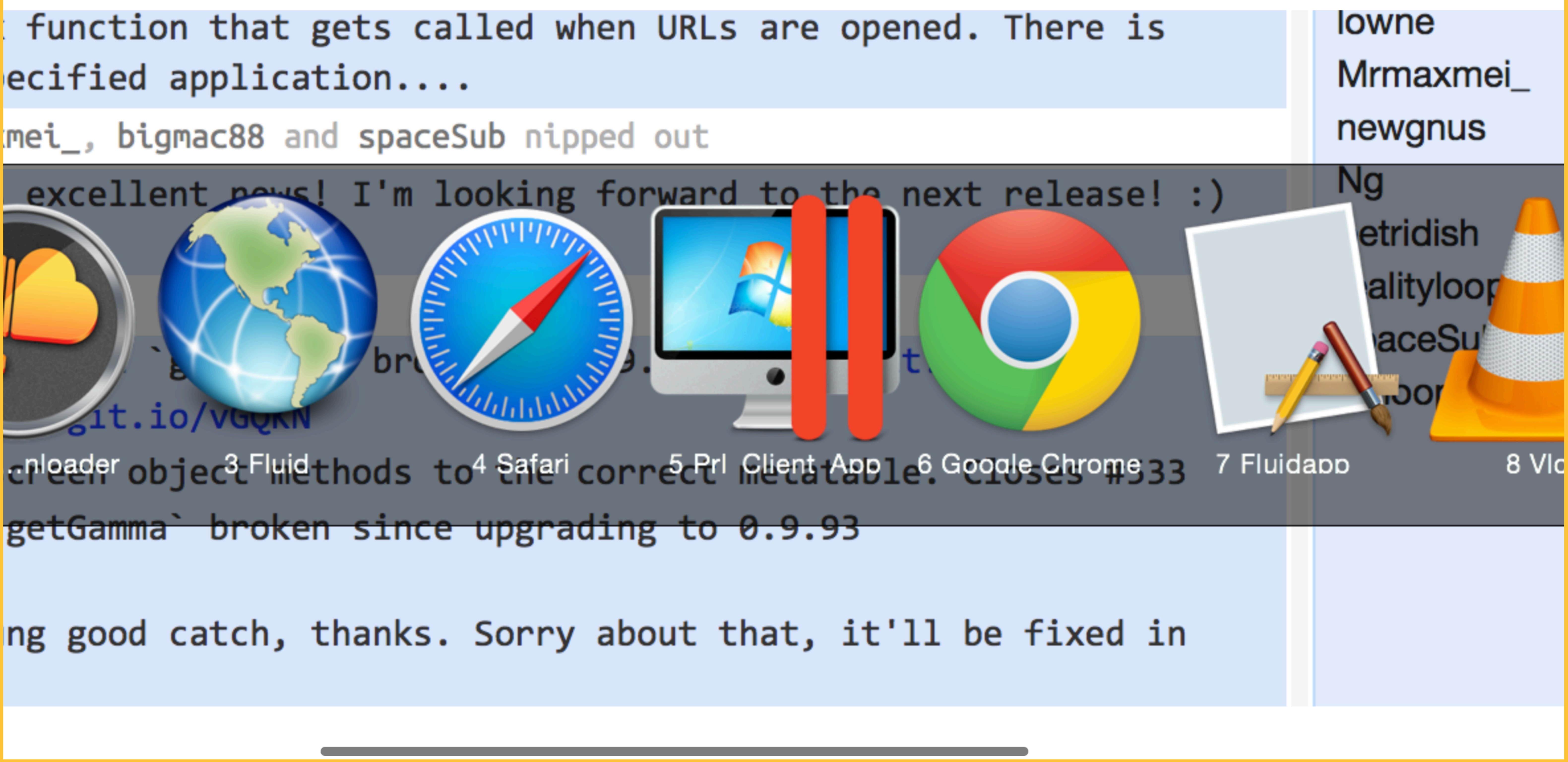
d1b960 Chris Jones: Move some hs.screen object methods to the correct metatab

losed issue #533: `setGamma` and `getGamma` broken since upgrading to 0.9.93

mmented on issue #533: @calvinwyoung good catch, thanks. Sorry about that, it

Hammerspoon: Staggeringly powerful desktop automation

# Handling URL events



function that gets called when URLs are opened. There is
ecified application....

mei_, bigmac88 and spaceSub nipped out

excellent news! I'm looking forward to the next release! :)

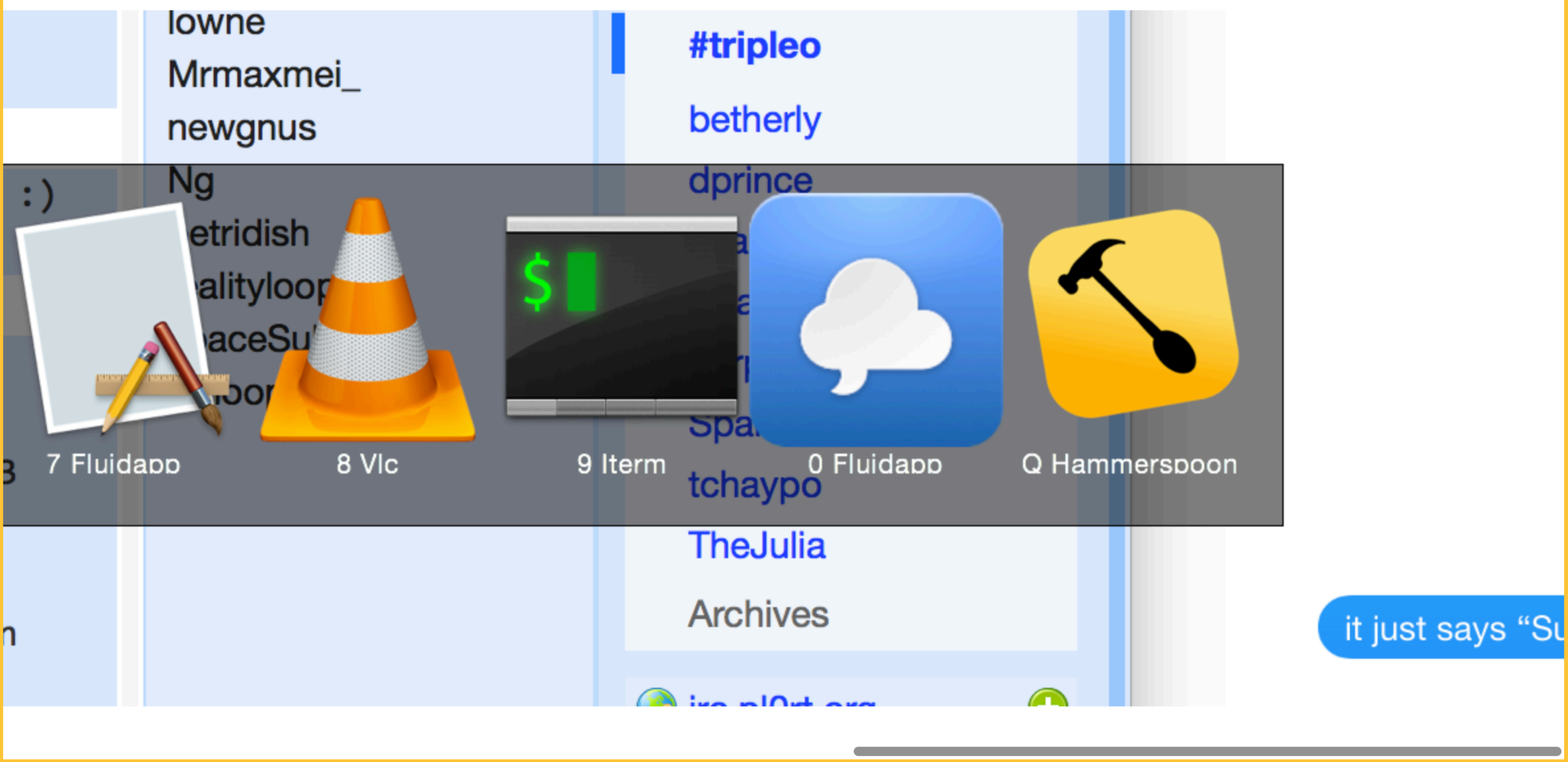
git.io/vGQKN

creen object methods to the correct metatable. Closes #333

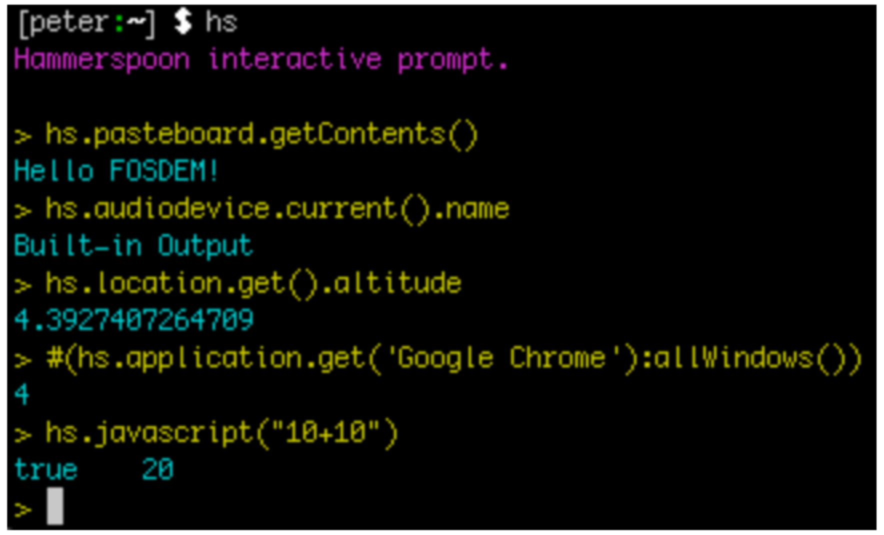
getGamma` broken since upgrading to 0.9.93

ng good catch, thanks. Sorry about that, it'll be fixed in

# Handling URL events



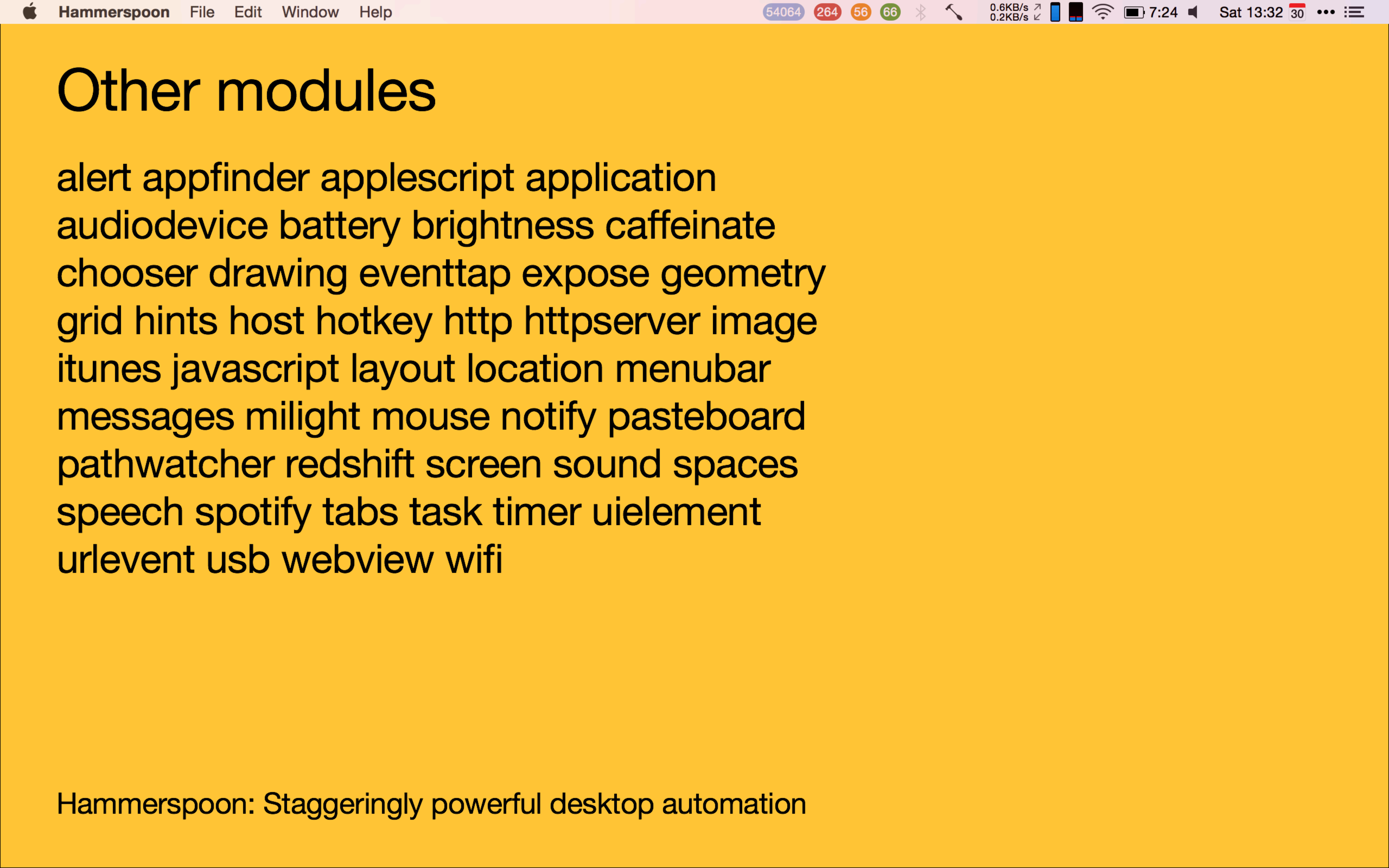
# Command line interface



```
[peter:~] $ hs
Hammerspoon interactive prompt.

> hs.pasteboard.getContents()
Hello FOSDEM!
> hs.audiodevice.current().name
Built-in Output
> hs.location.get().altitude
4.3927407264709
> #(hs.application.get('Google Chrome'):allWindows())
4
> hs.javascript("10+10")
true    20
> 
```

Hammerspoon: Staggeringly powerful desktop automation

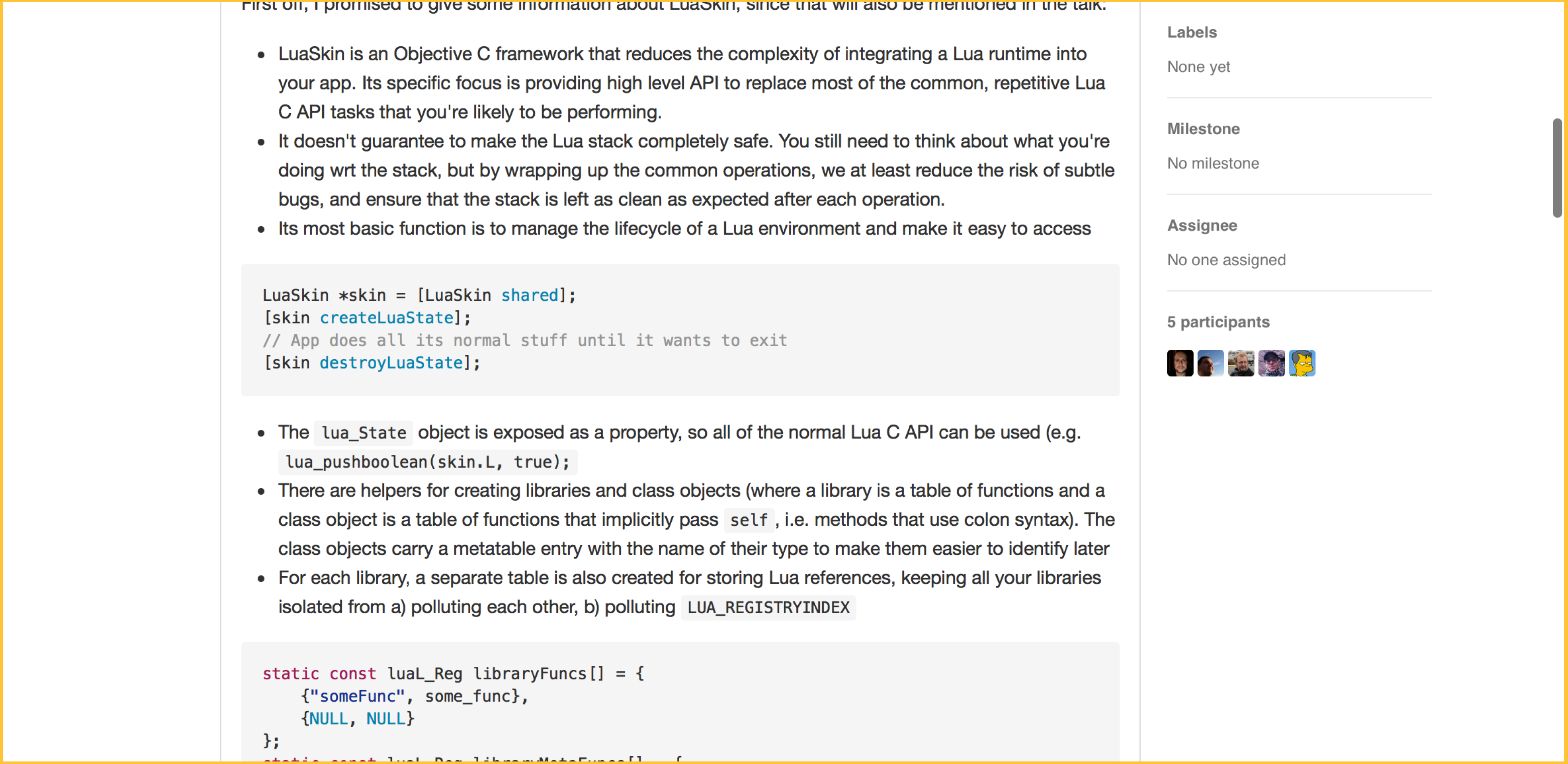


# Other modules

alert appfinder applescript application audiodevice battery brightness caffeinate chooser drawing eventtap expose geometry grid hints host hotkey http httpserver image itunes javascript layout location menubar messages milight mouse notify pasteboard pathwatcher redshift screen sound spaces speech spotify tabs task timer uielement urlevent usb webview wifi

Hammerspoon: Staggeringly powerful desktop automation

# LuaSkin



First off, I promised to give some information about LuaSkin, since that will also be mentioned in the talk:

- LuaSkin is an Objective C framework that reduces the complexity of integrating a Lua runtime into your app. Its specific focus is providing high level API to replace most of the common, repetitive Lua C API tasks that you're likely to be performing.
- It doesn't guarantee to make the Lua stack completely safe. You still need to think about what you're doing wrt the stack, but by wrapping up the common operations, we at least reduce the risk of subtle bugs, and ensure that the stack is left as clean as expected after each operation.
- Its most basic function is to manage the lifecycle of a Lua environment and make it easy to access

```
LuaSkin *skin = [LuaSkin shared];
[skin createLuaState];
// App does all its normal stuff until it wants to exit
[skin destroyLuaState];
```

- The `lua_State` object is exposed as a property, so all of the normal Lua C API can be used (e.g. `lua_pushboolean(skin.L, true);`
- There are helpers for creating libraries and class objects (where a library is a table of functions and a class object is a table of functions that implicitly pass `self`, i.e. methods that use colon syntax). The class objects carry a metatable entry with the name of their type to make them easier to identify later
- For each library, a separate table is also created for storing Lua references, keeping all your libraries isolated from a) polluting each other, b) polluting `LUA_REGISTRYINDEX`

```
static const luaL_Reg libraryFuncs[] = {
    {"someFunc", some_func},
    {NULL, NULL}
};
static const luaL_Reg libraryMetaFuncs[] = {
```

Labels

None yet

Milestone

No milestone

Assignee

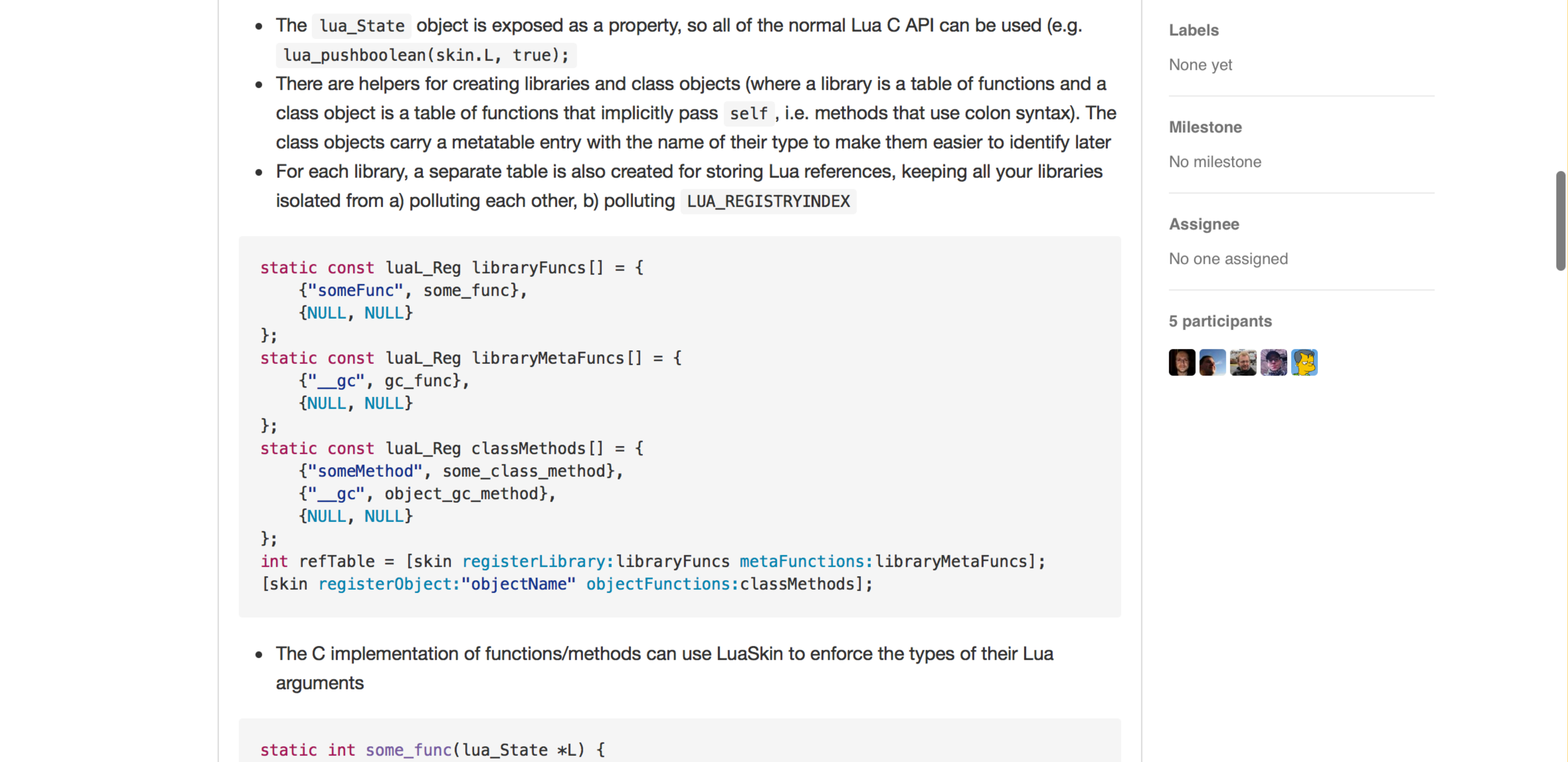
No one assigned

5 participants

Hammerspoon: Staggeringly powerful desktop automation

Hammerspoon File Edit Window Help 54064 264 56 66 3.0KB/s 3.4KB/s 7:24 Sat 13:32 30

# LuaSkin



- The `lua_State` object is exposed as a property, so all of the normal Lua C API can be used (e.g. `lua_pushboolean(skin.L, true);`
- There are helpers for creating libraries and class objects (where a library is a table of functions and a class object is a table of functions that implicitly pass `self`, i.e. methods that use colon syntax). The class objects carry a metatable entry with the name of their type to make them easier to identify later
- For each library, a separate table is also created for storing Lua references, keeping all your libraries isolated from a) polluting each other, b) polluting `LUA_REGISTRYINDEX`

```
static const luaL_Reg libraryFuncs[] = {
    {"someFunc", some_func},
    {NULL, NULL}
};
static const luaL_Reg libraryMetaFuncs[] = {
    {"__gc", gc_func},
    {NULL, NULL}
};
static const luaL_Reg classMethods[] = {
    {"someMethod", some_class_method},
    {"__gc", object_gc_method},
    {NULL, NULL}
};
int refTable = [skin registerLibrary:libraryFuncs metaFunctions:libraryMetaFuncs];
[skin registerObject:"objectName" objectFunctions:classMethods];
```

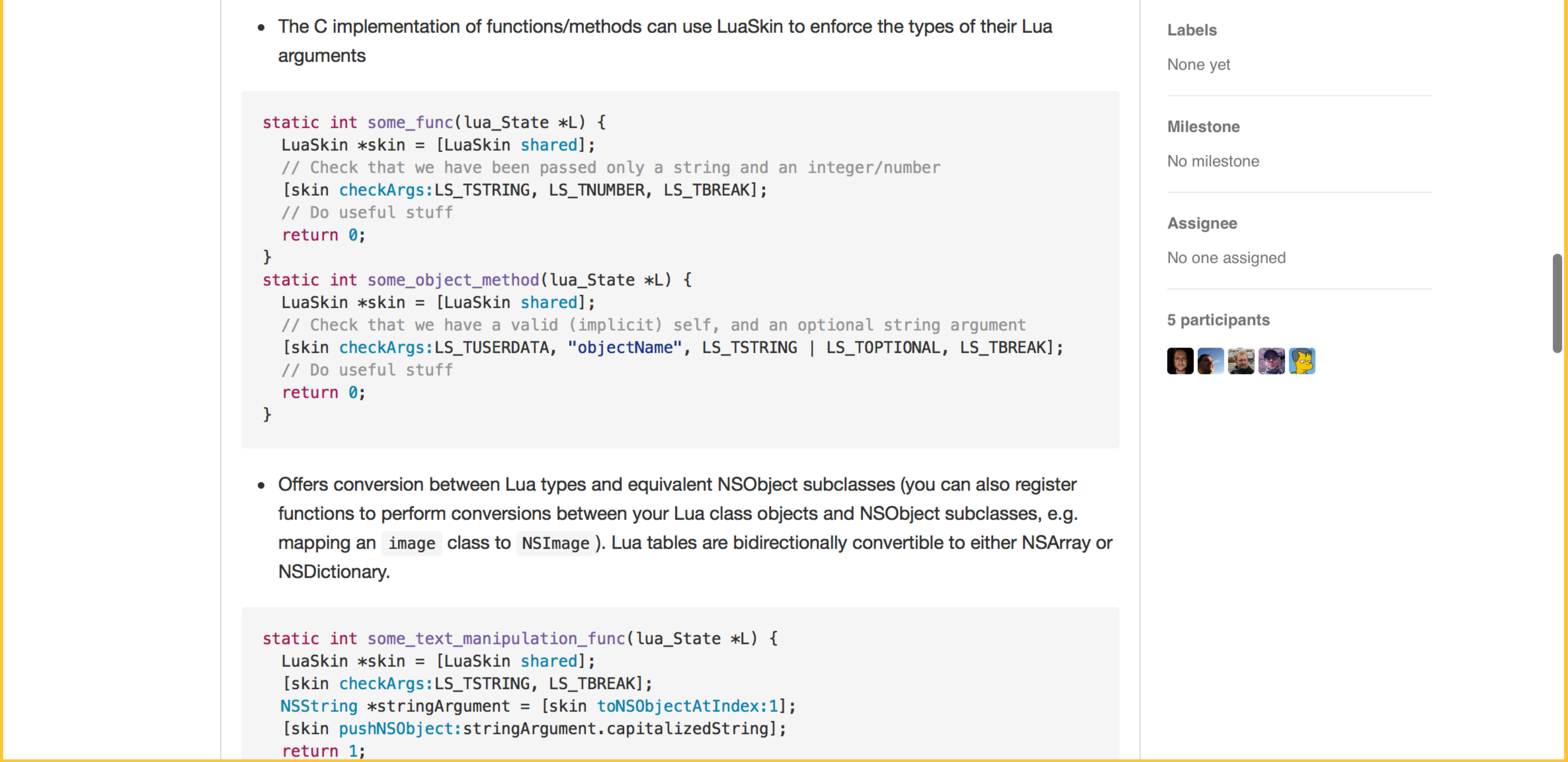
- The C implementation of functions/methods can use LuaSkin to enforce the types of their Lua arguments

```
static int some_func(lua_State *L) {
```

Hammerspoon: Staggeringly powerful desktop automation

Hammerspoon File Edit Window Help 54064 264 56 66 0.5KB/s 0.7KB/s 7:24 Sat 13:32 30

# LuaSkin



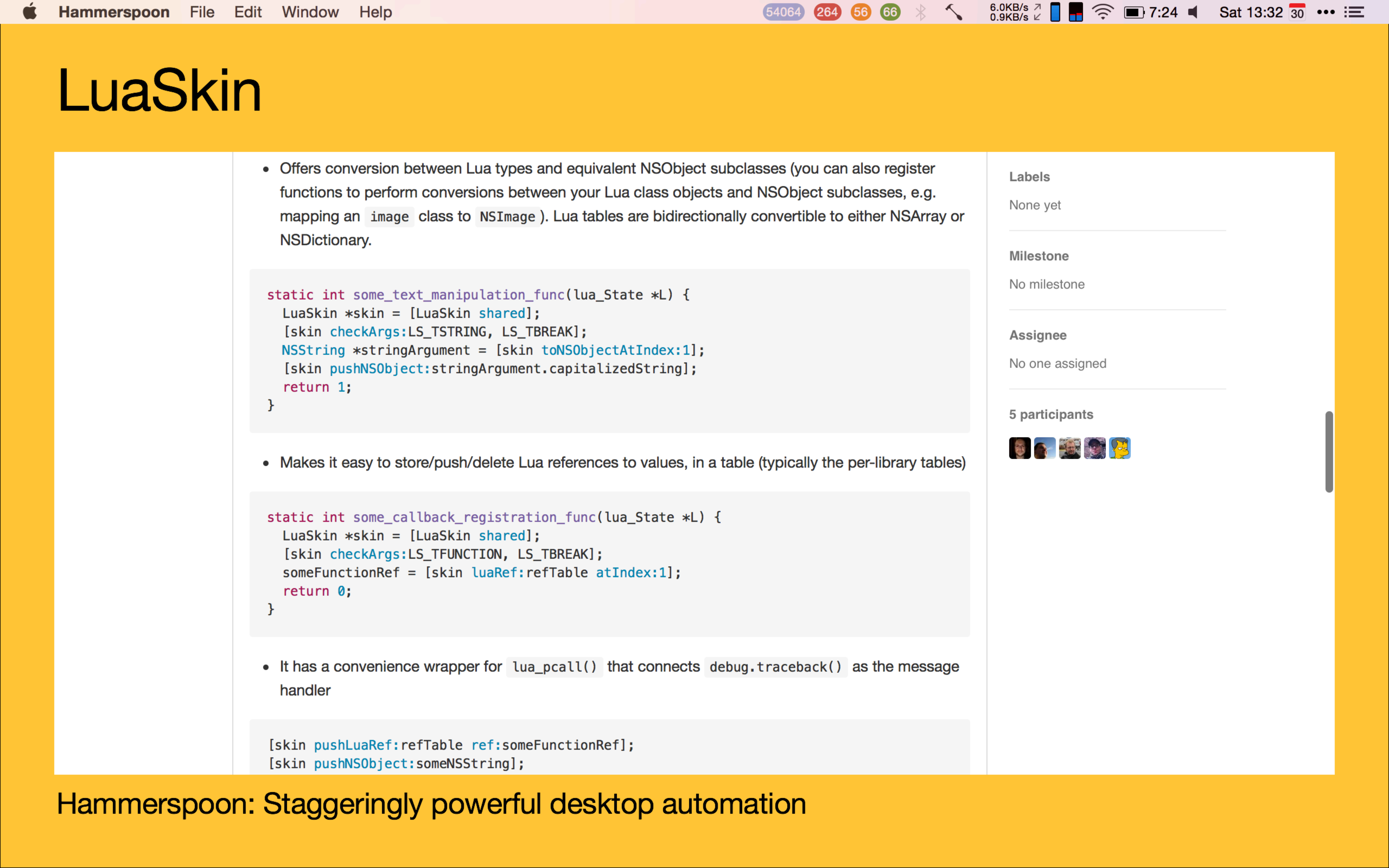
- The C implementation of functions/methods can use LuaSkin to enforce the types of their Lua arguments

```
static int some_func(lua_State *L) {
  LuaSkin *skin = [LuaSkin shared];
  // Check that we have been passed only a string and an integer/number
  [skin checkArgs:LS_TSTRING, LS_TNUMBER, LS_TBREAK];
  // Do useful stuff
  return 0;
}
static int some_object_method(lua_State *L) {
  LuaSkin *skin = [LuaSkin shared];
  // Check that we have a valid (implicit) self, and an optional string argument
  [skin checkArgs:LS_TUSERDATA, "objectName", LS_TSTRING | LS_TOPTIONAL, LS_TBREAK];
  // Do useful stuff
  return 0;
}
```

- Offers conversion between Lua types and equivalent NSObject subclasses (you can also register functions to perform conversions between your Lua class objects and NSObject subclasses, e.g. mapping an `image` class to `NSImage`). Lua tables are bidirectionally convertible to either NSArray or NSDictionary.

```
static int some_text_manipulation_func(lua_State *L) {
  LuaSkin *skin = [LuaSkin shared];
  [skin checkArgs:LS_TSTRING, LS_TBREAK];
  NSString *stringArgument = [skin toNSObjectAtIndex:1];
  [skin pushNSObject:stringArgument.capitalizedString];
  return 1;
```

Hammerspoon: Staggeringly powerful desktop automation



# LuaSkin

- Offers conversion between Lua types and equivalent NSObject subclasses (you can also register functions to perform conversions between your Lua class objects and NSObject subclasses, e.g. mapping an `image` class to `NSImage`). Lua tables are bidirectionally convertible to either NSArray or NSDictionary.

```
static int some_text_manipulation_func(lua_State *L) {
  LuaSkin *skin = [LuaSkin shared];
  [skin checkArgs:LS_TSTRING, LS_TBREAK];
  NSString *stringArgument = [skin toNSObjectAtIndex:1];
  [skin pushNSObject:stringArgument.capitalizedString];
  return 1;
}
```

- Makes it easy to store/push/delete Lua references to values, in a table (typically the per-library tables)

```
static int some_callback_registration_func(lua_State *L) {
  LuaSkin *skin = [LuaSkin shared];
  [skin checkArgs:LS_TFUNCTION, LS_TBREAK];
  someFunctionRef = [skin luaRef:refTable atIndex:1];
  return 0;
}
```

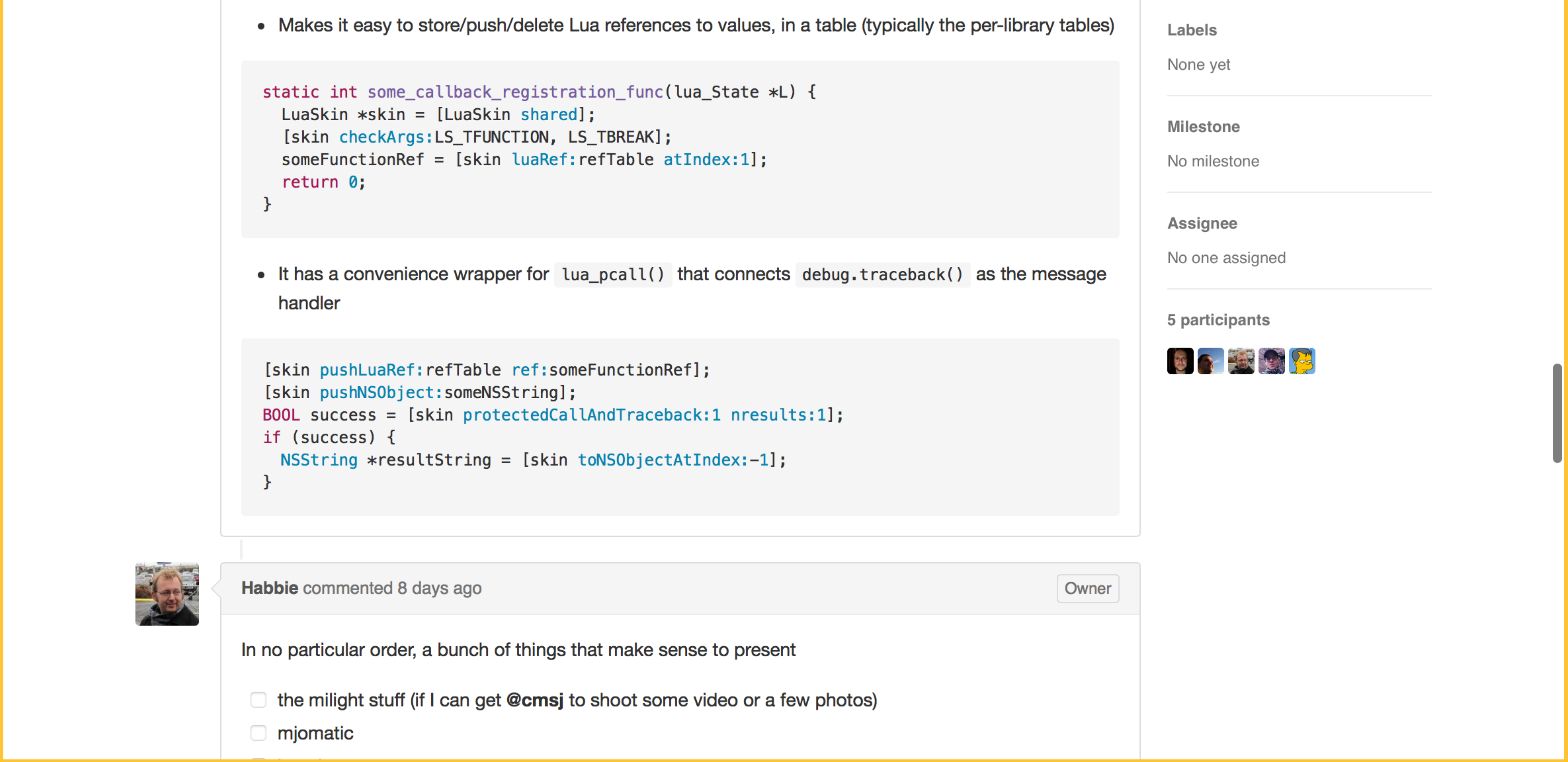
- It has a convenience wrapper for `lua_pcall()` that connects `debug.traceback()` as the message handler

```
[skin pushLuaRef:refTable ref:someFunctionRef];
[skin pushNSObject:someNSString];
```

Hammerspoon: Staggeringly powerful desktop automation

Hammerspoon File Edit Window Help

# LuaSkin



- Makes it easy to store/push/delete Lua references to values, in a table (typically the per-library tables)

```
static int some_callback_registration_func(lua_State *L) {
  LuaSkin *skin = [LuaSkin shared];
  [skin checkArgs:LS_TFUNCTION, LS_TBREAK];
  someFunctionRef = [skin luaRef:refTable atIndex:1];
  return 0;
}
```

- It has a convenience wrapper for `lua_pcall()` that connects `debug.traceback()` as the message handler

```
[skin pushLuaRef:refTable ref:someFunctionRef];
[skin pushNSObject:someNSString];
BOOL success = [skin protectedCallAndTraceback:1 nresults:1];
if (success) {
  NSString *resultString = [skin toNSObjectAtIndex:-1];
}
```

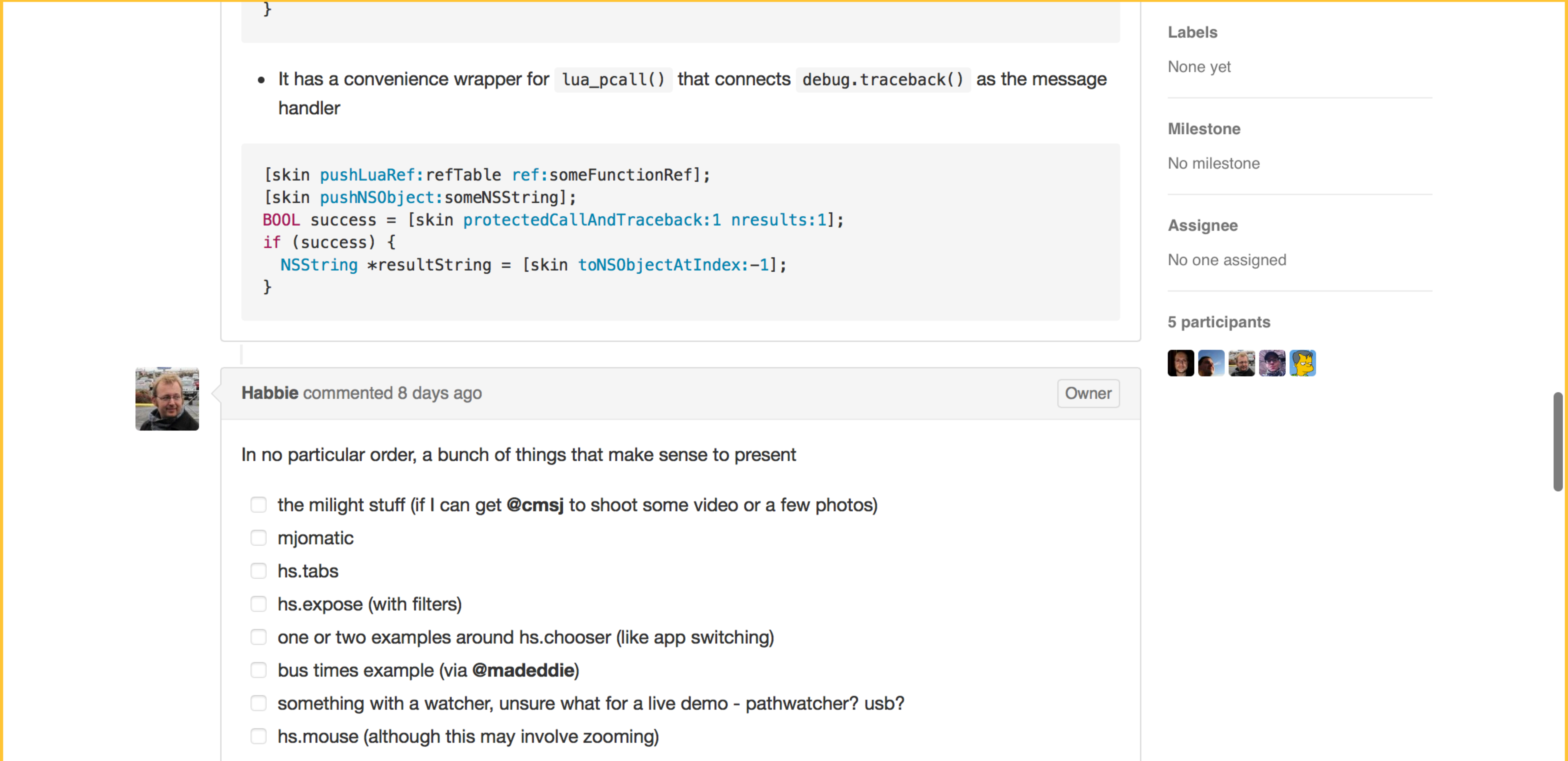
**Habbie** commented 8 days ago

In no particular order, a bunch of things that make sense to present

☐ the milight stuff (if I can get **@cmsj** to shoot some video or a few photos)
☐ mjomatic

Hammerspoon: Staggeringly powerful desktop automation

# LuaSkin



```
}
```

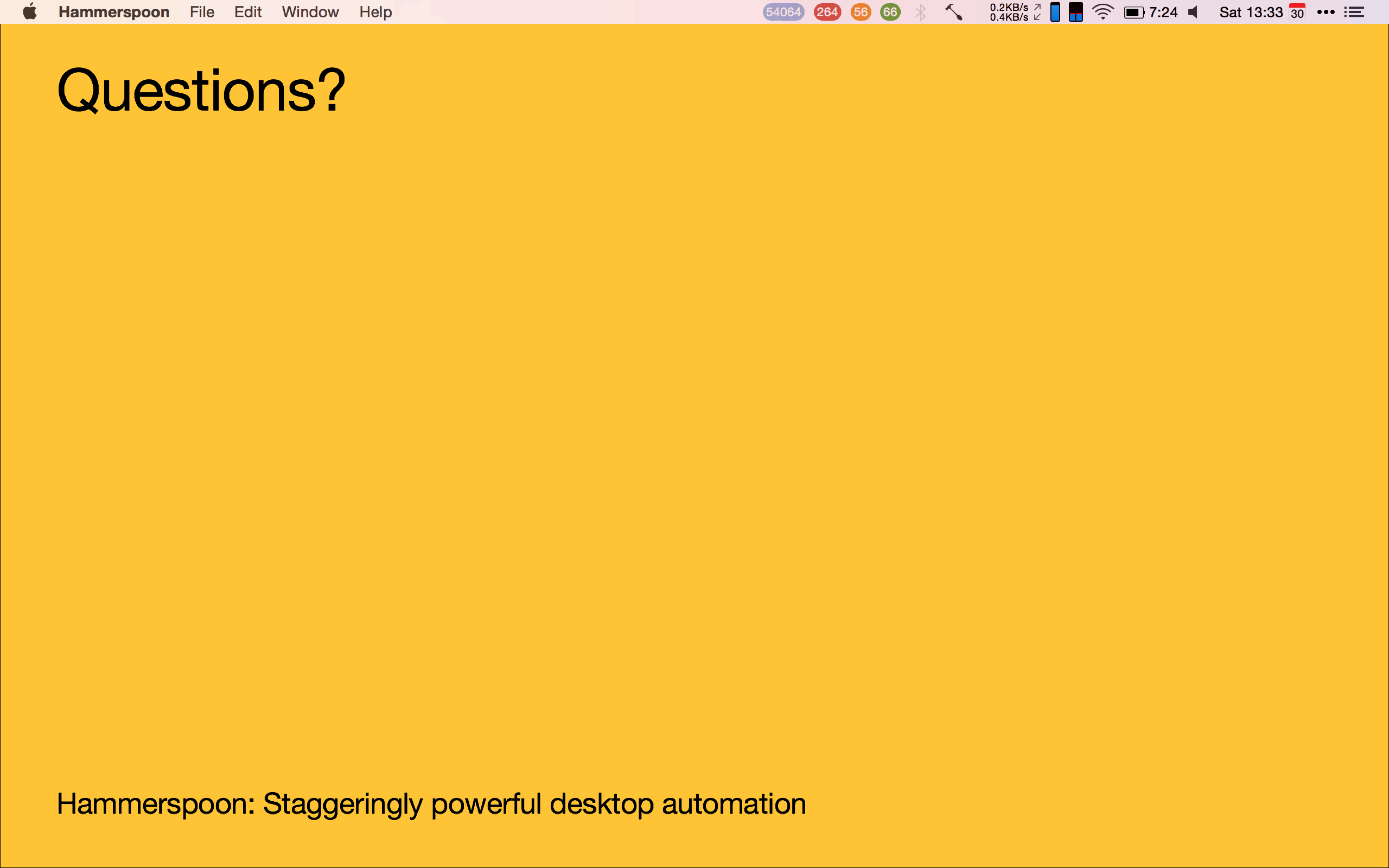
- It has a convenience wrapper for `lua_pcall()` that connects `debug.traceback()` as the message handler

```
[skin pushLuaRef:refTable ref:someFunctionRef];
[skin pushNSObject:someNSString];
BOOL success = [skin protectedCallAndTraceback:1 nresults:1];
if (success) {
  NSString *resultString = [skin toNSObjectAtIndex:-1];
}
```

**Habbie** commented 8 days ago

In no particular order, a bunch of things that make sense to present

- □ the milight stuff (if I can get **@cmsj** to shoot some video or a few photos)
- □ mjomatic
- □ hs.tabs
- □ hs.expose (with filters)
- □ one or two examples around hs.chooser (like app switching)
- □ bus times example (via **@madeddie**)
- □ something with a watcher, unsure what for a live demo - pathwatcher? usb?
- □ hs.mouse (although this may involve zooming)



# Questions?

Hammerspoon: Staggeringly powerful desktop automation