Lua on a Mac

Band 1: Hammerspoon

How to Automate Routine-Tasks

By Alfred Schilken

1. Edition, 2018 Hammerspoon – How to Automate Routine-Tasks

ISBN-13: 978-1717507907
ISBN-10: 1717507905

Translated with DeepL.com

Self-published by
Bergstr.7, 63486 Bruchköbel
alfred@schilken.de

Printed by CreateSpace

The following volumes are planned:
Band 2: CodeFlow – Programming Apps in Lua – to be published 2018
Band 3: Advanced Hammerspoon – SQLite, Screen-Scraping, FinderPopUp and more
Band 4: Developing apps for watchOS and tvOS in Lua

Version 1.0, last change: 03.05.2018, 227 pages / 44195 words / 246753 characters
Icons by https://icons8.com
Typeset in URWClassico
Edited with Papyrus Autor 9.02 on MacBook Pro.
Image processing with Photoshop Elements 13.

About the book:

You have a Mac and want to do routine tasks with as little effort as possible? Then this book shows you how Hammerspoon can help.

Practical examples serve as a starting point for your own scripts. In most cases, all you need to do is change a directory name, web address, or email address to customize the mini-programs for your purposes.

I also introduce some 'spoons'. These small ready-to-use applications are installed with two clicks and a few configuration lines.

The examples show:

- How your Mac sends a message to your iPhone for interesting events.
- How to collect several sections of text in the clipboard, re-sort them and enter them in a form in one step.
- How the spoon DeepLTranslate translates selected text sections.
- How to remotely control actions on your Mac with an email.
- How to add your own menu items to the menu bar.
- How to add hotkeys for menu items of any program.
- How to trigger mouse clicks with a hotkey.
- How to check and change your Mac's system settings.
- How each change in a directory calls a script.
- How connecting a USB device triggers an action.
- How your Mac recognizes the location using your WiFi and changes the volume.
- How your Mac checks every half hour whether your webserver is still responding.
- How a script checks the Amazon sales rank and reports a book sale.
- And much more...

About Hammerspoon and Lua:

Hammerspoon is a tool for automating macOS. It is a bridge between a Lua runtime environment and the operating system. Extensive libraries and so-called 'spoons'[1] facilitate access to all essential system functions of the macOS operating system.

About the strange name: Hammerspoon is a 'fork' of its predecessor project called Mjolnir. That is the name of the magical hammer of the god Thor in the Nordic mythology.

In case you didn't notice it in the logo: the lower end of the hammer is shaped like a spoon :-)

Lua is a scripting language that is also used to control Minecraft, Adobe Lightroom, Wireshark and 70 other programs[2]. It offers everything a full-fledged programming language needs and is particularly easy to learn. Some iOS apps[3] also use Lua as script language.

Even game development for iOS, Android and desktop is possible: Corona and Löve are popular platforms for games programmed in Lua. The CodeFlow IDE[4] enables the development of complex apps for all Apple systems–even for publication in the AppStore. It works optimally with Xcode and allows live coding: changes to the code or the storyboard are immediately visible without recompiling! Another book in the series 'Lua on the Mac' will be published in 2018.

---

1 These are easy to install hammerspoon applications.
2 see https://en.wikipedia.org/wiki/List_of_applications_using_Lua.
3 especially interesting: Trunknotes in the AppStore.
4 https://www.celedev.com

About the author:

Alfred Schilken was already enthusiastic about microprocessors in the 1980s. His first experimental board[5] had to make do with a single kilobyte of RAM and a cassette recorder as mass storage–that was in 1980, and it was a stroke of luck that his thesis at the Institute for Applied Physics in Frankfurt was about a microprocessor system.

Since then, programming has remained his hobby, but for more than 30 years he has also worked as a freelancer on projects for clients such as Telekom, Lufthansa Systems, Sybase, SAP and Deutsche Bank. He switched from assembler to C very early, then programmed in C++, later in Java, Python and Swift. Only since working with Hammerspoon has programming in Lua become very interesting for him.

[5] That was a KIM-1 development system for the 6502 microprocessor by MOS-Technology

Acknowledgements:

I would like to take this opportunity to thank some outstanding people. Without the initiative of these computer-loving colleagues the Hammerspoon community would not be as far as it is today.

My thanks to you:

- Roberto Ierusalimschy for the language Lua, which he published in 1993 and has been developing ever since.
- Steven Degutis. His project Mjolnir is the basis for Hammerspoon. The first commits in the Hammerspoon repository were also made by him.
- Chris Jones @cmsj for his over 1400 contributions to the development of the Hammerspoon App.
- Aaron Magill for 500 posts to the Hammerspoon GitHub repository.
- Mark Lowne for 300 contributions to the Hammerspoon GitHub repository.
- Diego Zamboni for his spoons and his blog[6].
- Chris Wanstrath, PJ Hyett and Tom Preston-Werner, the founders of GitHub. They make sure that GitHub remains free for OpenSource projects like Hammerspoon.
- DeepL GmbH in Cologne, Germany, who made the translation of this book into English possible with their free online translation service.

[6] http://zzamboni.org/post/

Table of Contents

Preface

Hammerspoon was for me the trigger to learn the language Lua. I had already noticed this language as an alternative to C in the embedded development of ESP6288 microcontrollers. But that was not enough to interest me in this language. While searching for a clipboard tool, I came across Hammerspoon and was impressed by the comprehensive integration into the macOS operating system. The simple triggering of a macro via a keyboard shortcut is just as possible as via a separate menu item in the menu bar. My first application was a custom clipboard tool. This way I copy the address data of an order from an e-mail and fill it into the form of my order processing in one go. Little by little I realized what else could to be done with the powerful API. I programmed a polling loop for my Bitcoin Miner that sends a message to my iPhone when it gets stuck. Then a monitor of the sales rank of my Amazon books was added and so on and so on. I was surprised that nobody had ever published a book about this useful tool. So I started collecting notes about Hammerspoon and now this book is published–the first worldwide.

In this book I give a crash course for the Lua language and show how very useful little helpers can be programmed with just a few lines of code. I will also introduce some useful 'spoons'. These are small ready-to-use applications that run in the Hammerspoon environment and are installed with two configuration lines. My tried-and-tested scripts are typical applications that can be adapted to individual needs by readers without programming experience. Usually only the names of programs, files or menu items need to be changed to achieve a desired different result.

If you are interested in more detailed programming tasks, the comments on the listings will help you. And if you want to do completely new tasks, the commented list of the Hammerspoon API gives you first clues for a possible implementation.

Preface to the English edition

Originally I didn't even think of translating the book into English. The thought just didn't come to me – probably because it seemed too time-consuming.

When I discovered the translation page https://deepl.com by chance and first samples surprised me very positively, my attitude changed. The next impulse was the Spoon PopupTranslateSelection by Diego Zamboni <diego@zzamboni.org>: His spoon uses the Google Translator, but I could change that with little effort.

With my spoon DeepLTranslate two hotkeys are now enough to translate a marked text passage from German into English. The English raw version was finished in one day. I also needed a few more days for proofreading. However, the effort is by far not as great as with a newly written version.

The DeepL translator works with AI and neural networks. This works surprisingly well, but is not (yet) perfect. There are certainly still some phrases in the book that I did not notice as a non-native speaker.

If you are a native speaker–or very good in English–I have an offer for you: By the end of 2018 at the latest I will publish a second edition of the book. If you have reported the most errors by then, I will refund the purchase price of the e-book via paypal. My main concern here is with confusing or misleading wording.

Translated with www.DeepL.com/Translator

Source code

The code of my configuration file init.lua and all Lua modules covered in this book is stored in this repository on GitHub:

https://github.com/schilken/dot-hammerspoon

The spoons I have extended are in the separate repository:

https://github.com/schilken/Spoons

This is a fork of: https://github.com/Hammerspoon/Spoons .

Web pages of the book

The book contains many links to the Internet in the footnotes and appendix, some of which are very long. These links can also be found on the book 's website. In addition, errors found, important changes to the Hammerspoon app and other tips will appear here.

Before you enter a long web address in the browser, open the web pages of the book and go to the link list under the address:

http://hammerspoon.tutorial-and-example.com .

Typographical conventions

`Source Code Pro – constant width – bold – in grey box`

All command line entries are indicated by this bold font on a grey background. If the line starts with a $, the entry is made in the terminal window of the Mac. If the line starts with >, the command must be entered in the Hammerspoon console.

`Source Code Pro – constant width – standard – in grey box`

All outputs in the Hammerspoon console and source texts appear on a grey background in non-proportional font. This font is particularly suitable for source code and data. If the line starts with a timestamp, then it is the output of a **print()** statement in the Hammerspoon console.

```
> os.execute('pwd')
true  exit  0
```

`Source Code Pro – constant width – bold – in text`

This bold font identifies names of **`files`**, **`variables`**, **`functions`**, and **`types`** in the continuous text.

All icons in this book are from https://icons8.com. Thanks to this notice I am allowed to use them without license fees.

Stumbling block: This icon marks a warning of typical errors or traps. For example, if a usual prerequisite is not fulfilled in the present case.

Hint: This icon marks a hint, useful additional information.

More technical details for advanced users.

1. Hammerspoon - what it can do

Hammerspoon is an automation tool for macOS. It provides a runtime environment for Lua scripts. Unlike 'normal' applications, these Lua scripts are not started in the Finder or via the Dock.

They are triggered by:

- keyboard shortcuts
- mouse clicks
- Selection of a menu item in the menu bar
- Changes in the file system (Watcher)
- Plugged USB devices
- Incoming HTTP packets

The script started in this way can in turn trigger far-reaching actions:

- Read or write clipboard
- Read or write files
- Simulate keystrokes
- Simulate mouse clicks
- Start programs, bring them to the foreground
- Change window size and position
- Trigger menu items of another program
- Access web servers
- Change system settings, e.g. volume, brightness
- and much more

Lua provides all the possibilities of a full-fledged programming language for processing the input:

- variables

- integer and floating point numbers
- Strings for text
- mathematical operations of all kinds, i.e. addition, subtraction and so on
- Tables for lists (arrays) and dictionaries[7]
- Logical ifs with branches
- Loops
- Functions

1.1 Install and allow access to the system

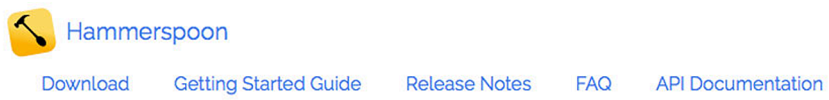

Figure 1 The Homepage of Hammerspoon

The download from **`Hammerspoon.org`** is free because it is an open source project whose source code is available on GitHub. The click on **`Download`** leads to GitHub.com. The Hammerspoon app consists of a single file, which is delivered as a ZIP archive with a size of approx. 8 MB. If the app is not unpacked automatically after the download, a double click in the Finder is enough to do this.

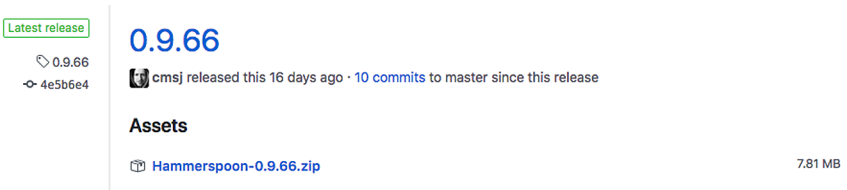

Figure 2 Download from GitHub.com

[7] dictionary is the usual name in Swift and Python. Javascript also uses the term Hashmap, some textbooks call it associative arrays.

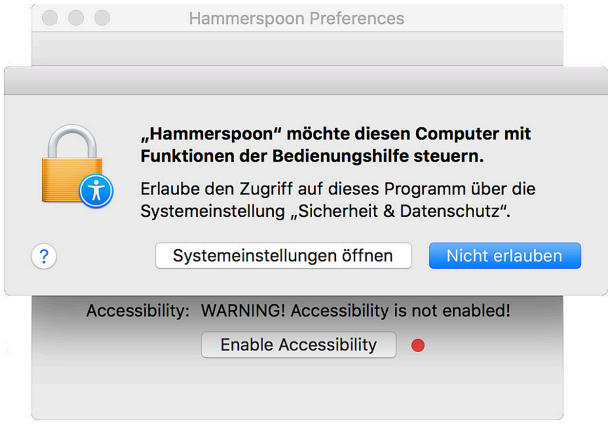

Figure 3 Open system settings to allow access

At the first start a message appears that access to the system is not allowed. Without this permission Hammerspoon could not intercept and simulate keyboard and mouse events. With a click on the **`Enable Accessibility`** button the system setting for security appears. Under the tab **`Privacy`** the Hammerspoon app is already entered in the list, but it must be allowed to control the computer. To check the box in front of the Hammerspoon entry, the lock must be unlocked and the administrator password is required.

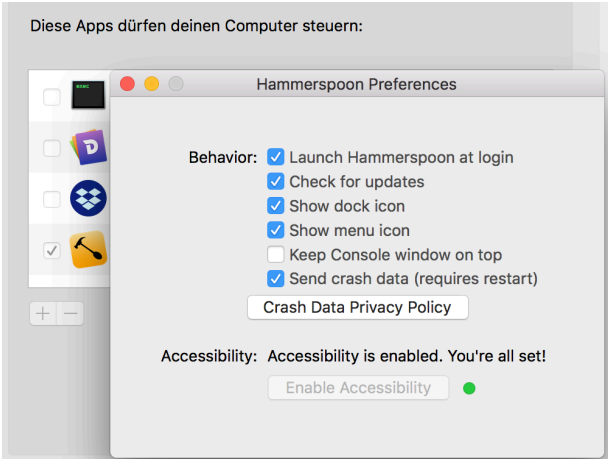

Figure 4 Now access is granted

After this permission, the Control Panel can be closed. The Hammerspoon preferences now show a green dot behind the disabled button Enable Accessability. Hammerspoon is ready to make himself useful.

`WARNING:`

Of course, abuse is also possible with this permission. As long as you run only self-made scripts, the risk is low unless you program the deletion of all files in a loop. But you could do the same damage directly in the Finder. Hammerspoon is comparable to a sharp knife: It can be very useful but also very harmful when used awkwardly.

For the beginning I recommend to use only the scripts offered and explained by me and to adapt them to your own needs. If you've become more familiar with Lua, you can browse the API yourself - looking for solutions to your automation tasks.

1.2 Hello Lua-World

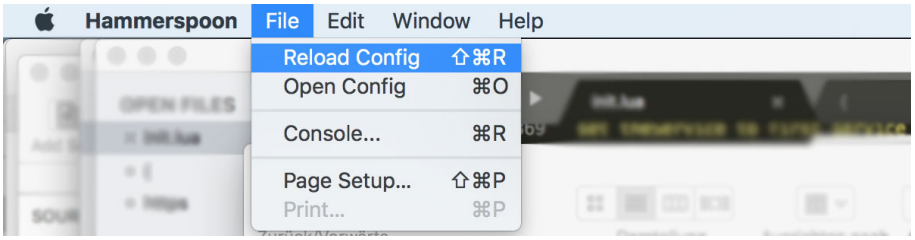

Figure 5 The menu of the Hammerspoon app

A short test ensures that the installation was successful. The Lua console, which can be opened via the menu of the Hammerspoon app, is suitable for this. A click on the menu item **`File > Console...`** (or with the hotkey **`⌘R`**) opens the Hammerspoon console, which I will use for many code examples in the following.

The illustration shows the Hammerspoon console shortly after entering the line **`hs.alert.show('Hello Lua-World')`** in the input field at the bottom. This call of a Hammerspoon function displays the given text as a message for a few seconds.

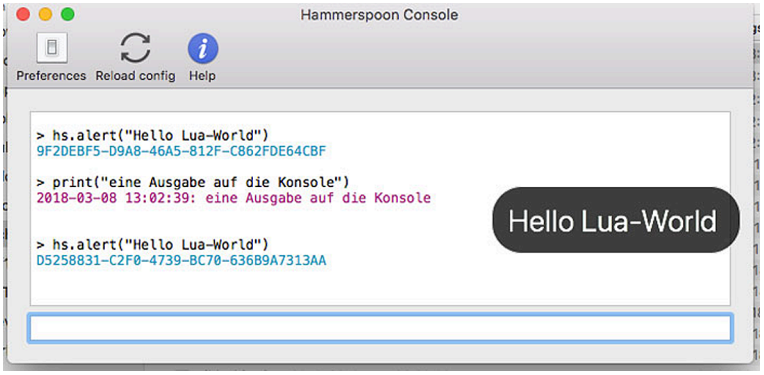

Figure 6 The output with hs.alert.show()

After pressing the Enter key, the input field is empty again and the entry is logged in black font in the output area. Below it in blue are the return values of the function. The number displayed here is the identification with which a permanent message (this is also possible) could be removed. Before this last entry I had already made two further entries, which can also be seen. A very helpful function for interactive experiments in the console is the **`print()`** statement. It outputs its parameters in red font with a preceding time stamp.

The message box with the rounded corners is the screen output of the **`alert()`** function. This box is positioned randomly above the console, it appears centered on the screen by default. However, the position can be changed with the corresponding parameter, more on this in *chapter 6.3 Output messages*.

1.3 Typical applications

The Hello World example has already demonstrated how easy it is to display a message. The following example shows how this display can be triggered with a hotkey. The connection of a hotkey with a Lua function is a special feature of Hammerspoon and is used very often. For such links to be permanently active, they must be defined in a configuration file. To keep this configuration as simple as possible, Hammerspoon requires some conventions. One of these conventions is where the Lua code for the configuration must be located: The first start of the Hammerspoon app after installation creates a directory with the name .hammerspoon in the home directory of the user. This directory is empty except for an empty directory named Spoons.

At the first start, a short notification appears, which can then be found in the notofocation center. It says here:

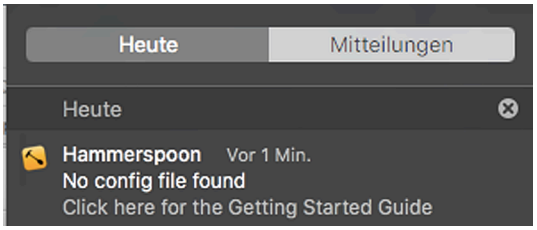

Figure 7 The entry in the User Notification Center after the first start of the Hammerspoon-App

A click on this message opens the tutorial page[8] in the web browser. More than 20 small examples are explained here. The first one shows the Lua code for a hammerspoon-typical 'Hello World':

```
hs.hotkey.bind({'ctrl', 'alt', 'cmd'}, 'W', function()
  hs.alert.show('Hello World')
end)
```

With the start of the Hammerspoon app also the hammer icon appears in the menu bar. A click on this icon shows a menu with the following entries:

`Reload Config`

`Open Config`

`Console...`

`...`

With Open Config the app opens the file init.lua in the default editor[9]. Copy the three lines of the Hello World script into the empty file and save it with ⌘S This creates the configuration file init.lua in the directory ~/.hammerspoon.

The menu command Reload Config activates the new configuration. This happens without any visible effects. The three lines of code have registered a new shortcut key and linked it with the function **`hs.alert.show('Hello World')`**. To check this, I open the Hammerspoon console with the menu command Console..... The last line

[8] http://www.hammerspoon.org/go/

[9] TextEdit, or in another editor if it has been reconfigured.

of the console output confirmes: The hotkey **⌘^⌥W** is active. The character ^ stands for the ctrl key.

```
2018-03-29 12:01:17: -- Loading ~/.hammerspoon/init.lua
2018-03-29 12:01:17: -- Loading extension: hotkey
2018-03-29 12:01:17: 12:01:17     hotkey: Enabled hotkey ⌘^⌥W
```

Listing 1 – The Hammerspoon console logs all activated hotkeys

This can easily proven in experiment: If you hold down the special keys **ctrl**⌥⌘ and the letter W, the message appears in a black window for one second. I use these alerts very often while writing a script or adapting it to other conditions.

I use all three special keys for all Hammerspoon hotkeys. This has the advantage that I don't have to check whether another program already uses the hotkey and I have to remember less details.

Key Binding – Shortcuts – Makros

Most programs provide hotkeys for the most frequently used menu items. These hotkeys appear in the menu to the right of the menu name. If no abbreviation is provided for a menu item, Hammerspoon can supply it very easily. The example in *chapter 10.7 Set up hotkeys for menu items* shows a simple case for the papyrus program with which I write this book.

However, the possibilities are far from exhausted. Because the newly registered abbreviation starts a Lua script, this script can also execute several menu items in succession. The script can also simulate keystrokes or mouse clicks. In this way, programs that do not offer this option themselves can be automated. Hammerspoon implements macros for programs that cannot do this themselves. If a certain action of a program can only be performed with the mouse, Hammerspoon may be able to provide a hotkey for it. However, this only works reliably if the position of the mouse click is always at the same place. The example in *chapter 6.9 Simulate mouse clicks* shows how to open a context menu with a simulated right mouse click.

Custom menu items in the menu bar

The API function **`hs.menubar.new()`** is used to insert a menu item in the menu bar. This menu item can appear as an icon or text. A click on it executes the registered Lua function. *Chapter 10.6 Extending the menu bar* shows this in practice. The Hammerspoon API also offers functions to change the text or the icon in the menu. A script can display its current state directly in the menu. This is quite useful for some applications. More extensive Lua programs with several commands can also display submenus, which appear when clicking on the menu entry and allow a more detailed selection of the desired action. For example, a collection of frequently used Lua scripts can be made available via a menu.

Periodic query of certain states

The Hammerspoon API provides a timer that repeatedly calls a Lua function with a configured time period.

A typical usage for this is: Every 10 minutes, a web server is queried and the HTTP response code is checked. If this results in a time-out, i.e. the server does not respond at all, or not with the OK code 200, the Lua function signals this with a warning. Such a warning can be:

- A macOS message that appears as a popup on the screen and in the notification center at the top right.
- A message (iMessage) sent to an iOS device, typically an iPhone or iPad.
- An e-mail to any recipient.

Positive monitoring is also possible: For example, the sales rank of your own books can be queried at Amazon. This sales rank is a number that gets bigger and bigger as long as no book is sold. So the book moves further and further back in the ranking. After a book sale, the rank jumps forward a few places, so the sales rank is reduced by a leap. Amazon seems to recalculate the sales rank every 2 hours. A Lua function, which is called with this period and compares the new rank with the previous one,

can recognize each book sale and report it by message on the iPhone (→ Practical example in *chapter 10.11 Periodically check Amazon rank*).

`Monitoring the File System`

Another API function is used to perform an action when a file is created or modified in a directory. I use this option with a database program that stores all reports of a day under the same name. As long as only one report is generated per day, this is not a problem. But if several of them are written, only the last saved content remains at the end of the day.

In the example of *chapter 10.4 Rename file automatically - Append time* I show how the name of each newly created file is extended by a time stamp.

EVA: Input – Processing – Output

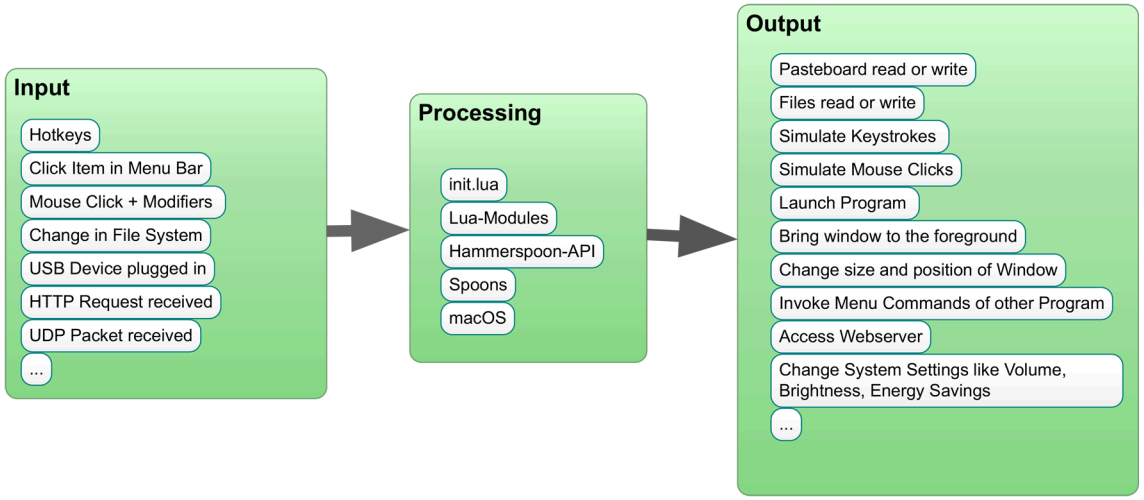

Figure 8 Overview of input and output options

As mentioned above, Hammerspoon scripts are not programs that are started in the Finder or in the terminal. They cannot be assigned to a file abbreviation either, so that they are automatically started when a file is double-clicked and receive the file name as a parameter.

To be useful, Hammerspoon scripts must also follow the classic IT rule:

```
Input - Processing - Output
```

or for short:

```
I - P - O.
```

Input

A very rough input possibility is the control of a script over different keyboard combinations. Depending on the combination, the script then performs different actions, but these are in a similar range. Such additional input details can also be specified by mouse clicks in combination with modifier keys. For example, right-clicking at the bottom left while holding down the Shift key could start a specific program. The current contents in the clipboard is also a popular input option. In this case, a script reads its input data directly from there. The file name from which the script is to read its input can also be located here.

However, it is more elegant to use the API for selecting one or more files. The script opens the standard dialog for selecting a file or directory. Depending on the user's choice, the script aborts his action or uses the selected file for his input.

When it comes to offering a choice of several options, the **hs.chooser** module is used. This module offers a list of freely configurable text entries. Each line can additionally contain an icon and a hotkey. At the top a text field is used to filter the entries, if there are too many. To make this even more versatile, a context menu for the selected line can offer more actions to be executed. In this way, very powerful control options can be implemented with just a few basic components.

Among the input possibilities are also eleven different watchers, all of which work according to the same principle. They monitor the following system components:

- file system
- Program status
- USB
- audio settings
- battery state
- Wifi

For the complete list see *chapter 6.6 Watcher*.

In the broadest sense, the HTTP server is also a watcher. Hammerspoon scripts can wait on several ports for HTTP requests or UDP packets. In this way, the input data comes directly from a distance to the configured Lua script and causes the programmed reaction and output. (→ *chapter 6.6*)

An experimental input option is the control with noises. Two easy to make noises trigger a Lua script: Lip Popping and the 'sssss' noise. See the module **`hs.noise`** in the Hammerspoon API documentation.

I haven't tried the module **`hs.speech.listener`** yet, but it sounds exciting.

Processing

Hammerspoon uses the little-known script language Lua to process the input. But Lua is by no means new. It was developed in Brazil as early as 1993 and is increasingly gaining acceptance as an embedded macro language in larger programs. The best known example is Adobe's Lightroom. All Lightroom plugins are coded in Lua. Also the network tool Wireshark or the game Minecraft use Lua. Lua offers all the possibilities of a full-fledged programming language. It is very minimalistic with only 22 reserved keywords. But it is so cleverly designed that even very demanding concepts can be realized with the means offered. For example, it is not inherently object-oriented, which could be considered a weakness. However, all essential properties of

an object-oriented language can be simulated very easily with a few conventions. This has the advantage that the functionality of classes, objects and inheritance are not hidden behind a facade. All these features are realized in Lua with very few basic functions. I also became aware of many automagic operations of the compiler that are hidden in **`Java`**, **`C++`** or **`Swift`**.

The concept of modularization is also realized with little effort. Overall, Lua is a language that is so small that it is easy to keep a full overview. In my opinion, this is no longer the case with **`C++`** or **`Swift`**. These languages are so complex that I sometimes have to try a Google search for special constructions to find a clarifying example. Lua is considered a 'glue language', which means that its main task is to combine the functions of powerful libraries to build complex applications. Such libraries are often highly optimized and programmed in a 'real' native programming language. These basic components do the main work and the Lua code combines the basic functions with each other. Lua forms the thin intermediate layer that tells the native high-performance components what to do.

In the case of the Hammerspoon app, Lua-Code coordinates the calls of the system frameworks encoded in **`Objective-C`**. To make this even easier, Hammerspoon provides an intermediate layer in the form of its own API. On the one hand, this hs API has a pleasant Lua interface and on the other hand, it uses all required macOS frameworks. For example, a single Lua line is enough to send a message to an iPhone. All necessary calls to the macOS frameworks are encapsulated by the API class **`hs.message`**.

With the Hammerspoon API, however, not all possibilities are exhausted. The so-called spoons are Lua modules that offer a standardized plugin interface and can be installed as required. A directory of all spoons can be found in the official spoon repository (http://www.hammerspoon.org/Spoons/). The source code of these plugins is open source on GitHub. Spoons can contain not only Lua code, but also native C code. Thus even complex drivers and adapters for Hammerspoon scripts, which actually only have a C interface, can become useful.

Output

In addition to the many input options, Hammerspoon also excels with its extensive output options. Normal programs' cannot simply bring another program to the foreground and control it remotely. For Hammerspoon this is no problem: It can trigger any menu item of another program, control the mouse position, perform clicks, enter text in dialog boxes of another program and much more.

Hammerspoon scripts can also produce the 'usual' output of a program, for example:

- Create a new file in the file system
- Write text or binary data to a file
- Show text in popup window
- Output standard messages
- Play mp3 or system sound effects
- Display window with HTML-formatted contents
- Send HTTP Requests
- Control USB devices
- Interact with SQLite databases
- Access the Speech Synthesizer component of OS X
- Use MIDI, serial interface and much more

2. Lua Overview

I hope the previous chapter has made clear what interesting possibilities lie dormant in the combination of Lua, Hammerspoon app and macOS frameworks. Before these possibilities can be used, it is worth learning more about Lua. Because of the small language range, this is quickly done.

2.1 Similarities with Javascript and C

If you have already gained experience with another programming language, you will quickly get used to the simple syntax of Lua. A typical Lua program looks like this:

```
-- a comment
local stringVariable = 'abc'  -- no semicolon required

globalIntegerVariable = 123

function sum(a, b)
  return a + b
end

if stringVariable == 'Adams' then
  globaleIntegerVariable = 42
else
  globaleIntegerVariable = 0
end

local summe = sum(42, globaleIntegerVariable)

print('summe: ' .. summe)
```

Listing 2 – A typical Lua program

- This small program shows how minimalistic Lua code looks like:
- No semicolon at the end of the line is necessary, but it doesn't disturb either. The newer languages **`Swift`** and **`Go`** behave in the same way. No type must be specified when defining variables.