AppSort

Making a script

Contents

Introduction

AppSort uses Lua scripts located at “/Library/Application Support/AppSort” to sort applications. Each script should contain a function named sort with one parameter (I usually call this parameter apps but in reality it can be called anything).

The parameter will help determine all information needed for sorting. It is a 2D Lua table. Each inner table represents an app on the home screen (it doesn't include dock icons or icons inside of folders).

The script must also return a 2D table. The returned table must also include every app that comes in on the parameter and must not repeat any items. If these requirements are not met (by syntax error or improper coding) an error message will pop up when the script is attempted to run.

This document is not intended on teaching you Lua. It assumes you know or may have some limited knowledge on it. If you do not know Lua you should start here: http://www.lua.org/pil/contents.html

The Function

The function is how we know how to sort the apps. The name of the function must be “sort.” The most basic function looks like this.

```
function sort(apps)
		return apps
end
```

This function will not do any sorting but it will work and help me explain the anatomy of it more. The parameter “apps” comes in as a 2D table (meaning its a table of tables). The “apps” table an integer table or array.

The return value is also something very important. The return value should be in the exact same format as the input parameter. It is also important that they contain the same objects and the returned value is just in a different order. If this is not true your script will throw an error.

The Parameter

In the last section we touched briefly on the parameter of the function. There is not a lot more to say about it. It is a Lua array or integer based table starting at an index of 1. This means we should use functions like "ipairs()" etc. on it. All app information available to you for sorting will be passed through this parameter. The basic structure of this table looks something like this.

```
{1 = "app one table", 2 = "app two table"}
```

Of course the real parameter will contain all apps and not just two. Also the contained value will will be a table not a string. We will look more at the contained value in the next section.

The Application Table

The application table contains all the information available to us for that exact app. The information available contains: average icon color, screen time of app, name of the app, bundle identifier of the app, badge count of the app, icon type (ex. folder, app, bookmark) and genre of the app. These values are stored in a key-value Lua table. Accessing one of these values from the parameter looks like this.

```
name = apps[1]["name"]
```

This will access the name of the of the first app in the parameter. This means the parameters real structure looks something more like this.

```
{1 ={"name" = "Example","color" =
"FFFFFF","genre" =
"Productivity","id"="com.example.exampleapp","t
ype"="Application","badge"="0","usage"="0"}
```

Again this is only one app and the parameter will contain all the apps. Below is a list of the available values and what they return.

value	return
name	string containing the icons label
color	string containing hex color code of the average color of the icon
genre	string containing the label that a folder would have if using this app to create one. Example of results include: “Social”,”Productivity”,”Games”.
id	string containing bundle identifier. In some cases where a bundle identifier is not present a number (in string format) will be used
type	string containing what type of icon it is. Possible results: “Application” for a normal application, “Folder” of a folder icon and “Webclip” for a bookmark icon.
badge	string containing the number of badges the current icon has
usage	string containing the number of seconds on screen that application has

An Example

Below is an example of a sorting script that will sort icons based on there labels first first letter alphabetically.

```
function sort(apps)
    table.sort(apps, function(app1, app2) return cap(app1["name"]) <
cap(app2["name"] end)
    return apps
end
function cap(str)
    return  string.gsub(str, "^%l", string.upper)
end
```

Conclusion

Now you should be able to build your own app sorting Lua scripts. if you want to see more examples you can always look in the directory where the scripts are stored. The whole process is fairly simple and I probably over explained it.

If anything is confusing about this document please let me know and I will do my best to fix it. Also if there are more values in the application table you would like access to, let me know about that as well. And lastly if you see that any of my scripts are pretty bad (I know they aren't the best, I never used Lua before making this tweak), please let me know.

I encourage you to make sorting scripts for AppSort. If you think your script is super awesome you can email it to me and I might add it into a release, if you want. I also think that repositories will let you put the scripts on their repo as well (as long as you know how to build packages).

Have fun sorting your apps and please do let me know about feature requests, problems, confusion or anything relevant to AppSort.