How to turn the Bash script into a clickable application with AppleScript

AppleScript is a user-friendly programming language that makes it easy to turn scripts into something GUI-friendly and GUI-friendly.

Suppose you are a very knowledgeable bash script, like automating tasks for friends and family members who don't know much about technology. Although you gave them a script that completely fixes the problem, they can still worry about using Terminal alone. In this case, you can also help them manually.

Fortunately, AppleScript is a user-friendly programming language that makes it easy to turn scripts into something that is shared and friendly to the GUI.

"Transform" Bash script with AppleScript tool

Start with AppleScript!

AppleScript is a long-standing programming language from the Mac OS version 7. AppleScript is designed as an easy-to-read and easy-to-use language, according to user standards in the 1980s. The main purpose of AppleScript is interaction. with Finder to automate workflows.

You can use AppleScript to perform a complex workflow, such as the bash script or modify the priority file and simplify it into a clickable button.

The **Script Editor** tool in the **Utilities** folder is the AppleScript text editor and IDE, helping you write an AppleScript function. The Help menu bar is an invaluable resource. Please use this menu to view AppleScript language guide information.

You can also go to **Window> Library** to get a list of all AppleScript commands you can use.

Make the Bash script clickable with AppleScript

Bash can help you interact with your Mac, create scripts for almost everything you want to do with your computer.

Normally, if you want to run the bash script, you must ensure that it has the right to execute. After that, you must open **Terminal,** navigate to the script path and press **Return** to run it. However, with AppleScript, you can run bash scripts by double-clicking quickly.

Command by shell script

Adding bash scripts to AppleScript is a simple process. First, go to **Applications> Utilities> Script Editor** to open the new AppleScript. Then, select a location for the script after it is completed and click **New Document.** You will see an empty editing window.

Prepare the bash script with your favorite macOS text editor or do it right in Script Editor. When ready, add it with the command **of the shell script** . You can send commands using:

```
do shell script "Command"
```

Add a semicolon (**;**) to send multiple commands, as follows:

```
do shell script "Command1; Command 2"
```

To refer to the bash script available elsewhere, just use:

```
do shell script "/path/to/your/script.sh"
```

If you want to run a command that requires admin rights, you can put them in AppleScript first, as follows:

```
do shell script "command" user name "USER" password "PASSWORD" with administrato
```

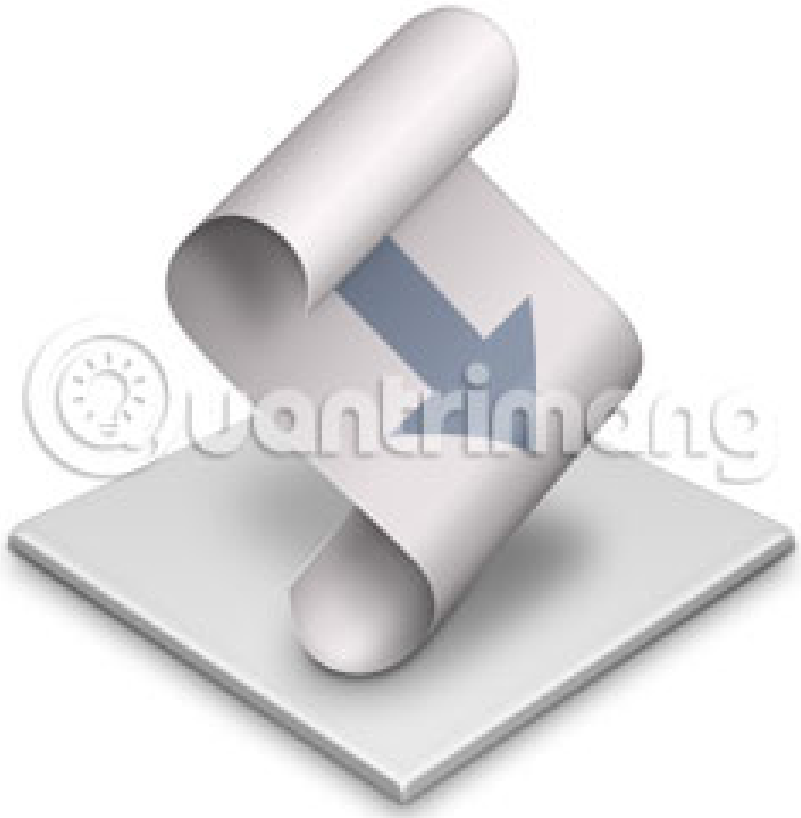
Click the **Play** button to run through AppleScript and check for syntax errors. Finally, save AppleScript. In **File Format,** select **Application.** This option will allow you to double click to run the script.

Reverse the script in Bash

On a side note, what if you want to do the opposite, ie add AppleScript commands to the bash script? The good news is that you can do that too!

In the bash script, you can add an **osascript** command **,** followed by AppleScript, to add interesting elements such as user input or warning banners.

Use AppleScript Droplet



Suppose you have a lot of resized images on your Mac to post to a blog or want to sort all files by type. You can write scripts to perform these actions. But a traditional bash script requires you to write the entire path of each file you want to manipulate, when you need to execute that script. Doing that is boring, especially when you have dozens of large files.

Fortunately, you can set AppleScript to only request to drag the target file into it, then run as input. These special AppleScript are called Droplets. Here's how to create a Droplet:

1. Open a new AppleScript with Script Editor.

2. Start AppleScript with **on open dropped_file** . You can call **drop_file** by whatever name you want. AppleScript will use it as a variable assigned to the file you have dropped.

3. Complete the rest of the script, use the **shell script** or the usual AppleScript syntax. Make sure you close the script with **end open.**

4. Save AppleScript as an application, as explained above.

You can now drag and drop files directly into your script.

An example of an AppleScript script

The example below shows both the handy features of AppleScript mentioned above in real-life situations.

Change the permissions of any drag and drop file

Suppose you want to give files, scripts or documents to a friend and want to make sure they don't make it messy. Using this script, you can easily remove the write permissions of any file, turn it into a read-only version.

Keep in mind that when you drop a file on a Droplet, the path name is read by AppleScript in HFS (Hierarchical File System) format. This format uses colons instead of spaces, so the file on the desktop reads:

```
Macintosh HD:Users:jdoe:Desktop:myfile
```

Bash uses a different path standard called POSIX, so the same bash file read is **Macintosh HD / Users / jdoe / Desktop / myfile** . The script below converts the file name to the appropriate path standard before changing the permissions on it:

```
on open dropped_file set bash_path_file to the POSIX path of dropped_file do she
```

Use Automator to automate Mac more

Now that you've grasped the basics of automation, learn more by going to Automator on a Mac. Automator and AppleScript have a lot of similarities. If you're just getting started with scripts and automation, you can see Automator is an easier tool. Instead of an easy-to-read programming language, Automator uses intuitive drag and drop bubbles and an easy-to-learn interface to simplify common tasks.

Hope you are succesful.