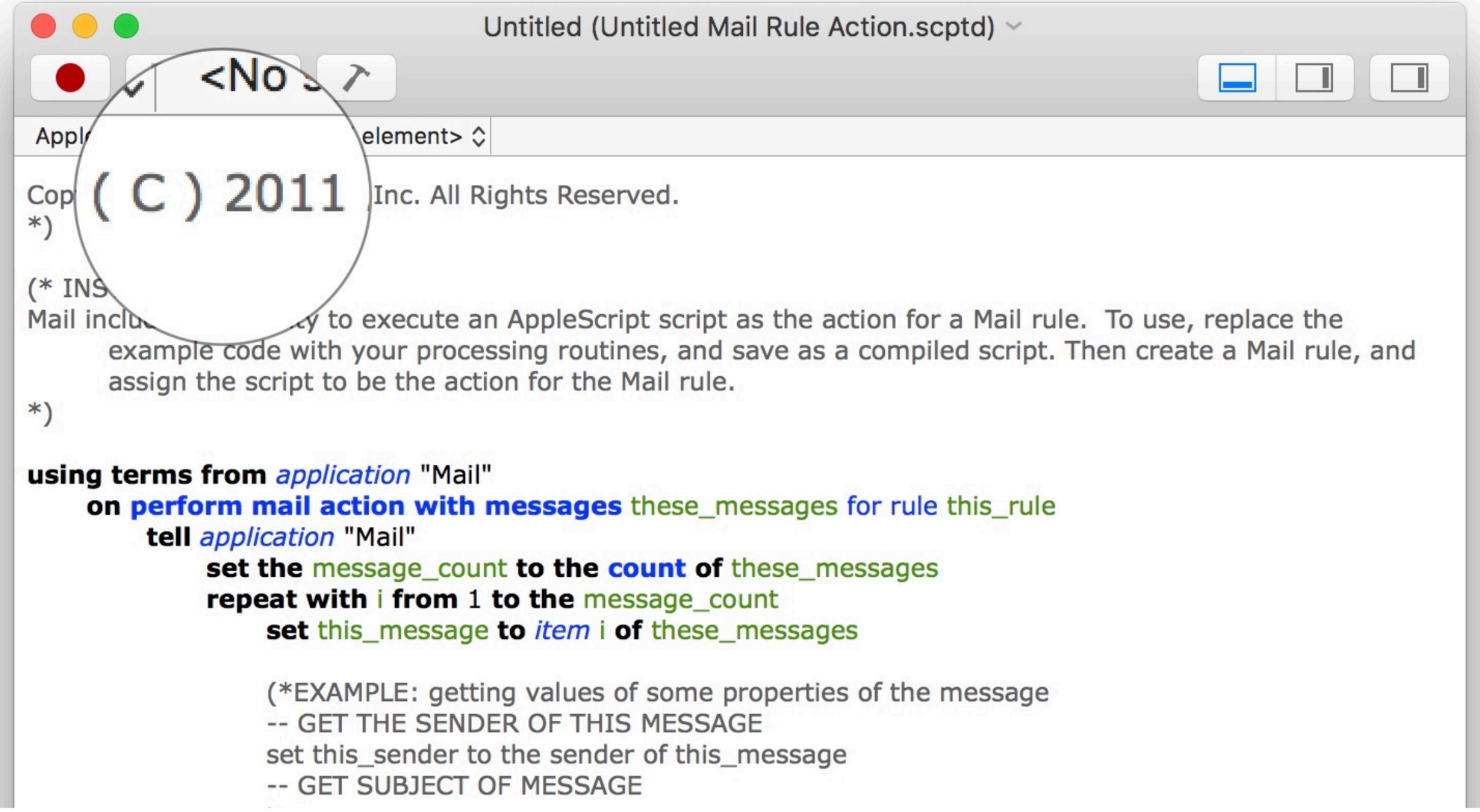
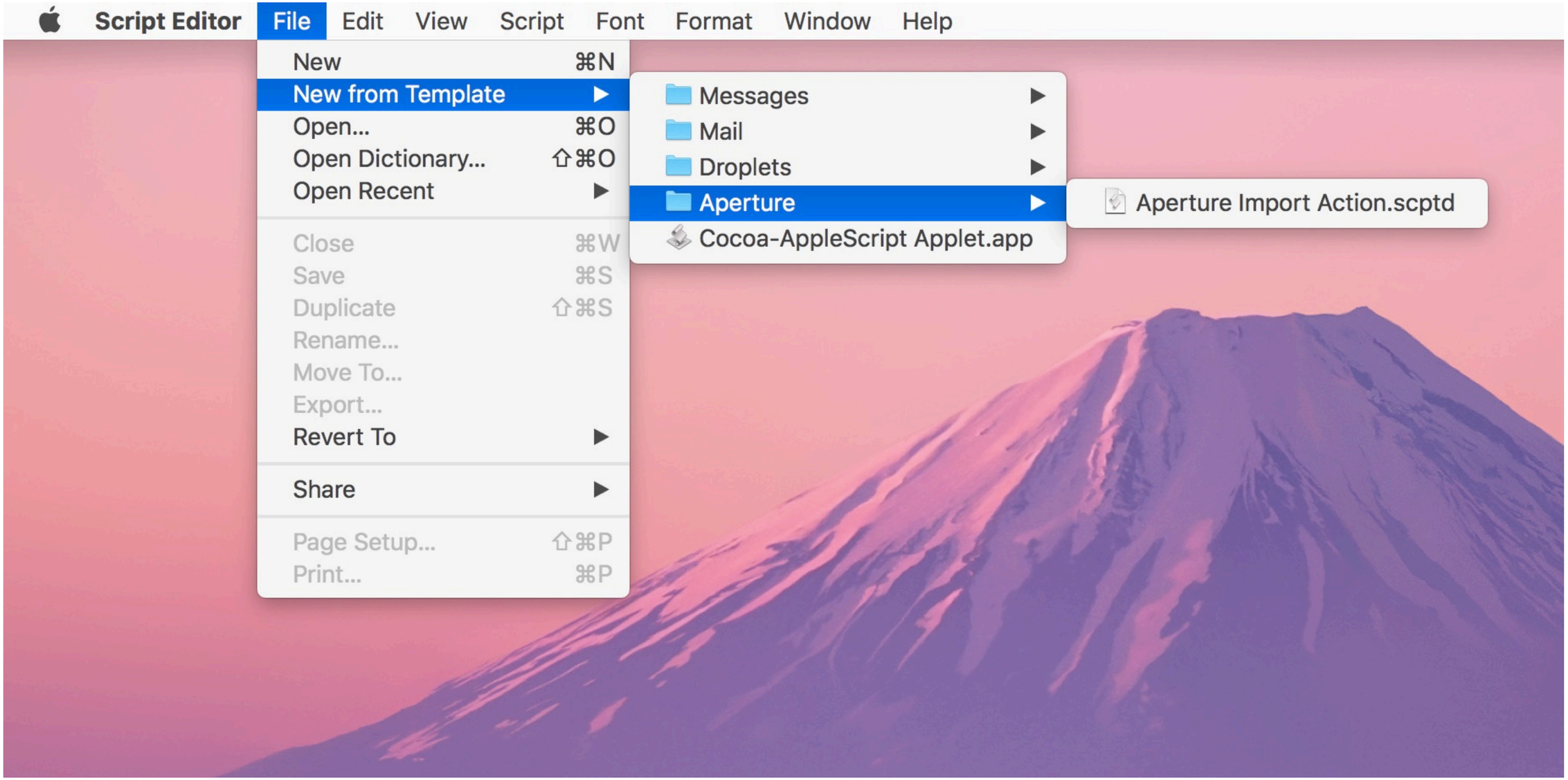
JavaScript for Automation (JXA)

@joshparnham


Untitled (Untitled Mail Rule Action.scptd)
(C) 2011
Inc. All Rights Reserved.
*)
(* INS
Mail inclu y to execute an AppleScript script as the action for a Mail rule. To use, replace the
example code with your processing routines, and save as a compiled script. Then create a Mail rule, and
assign the script to be the action for the Mail rule.
*)
using terms from application "Mail"
on perform mail action with messages these_messages for rule this_rule
tell application "Mail"
set the message_count to the count of these_messages
repeat with i from 1 to the message_count
set this_message to item i of these_messages
(*EXAMPLE: getting values of some properties of the message
-- GET THE SENDER OF THIS MESSAGE
set this_sender to the sender of this_message
-- GET SUBJECT OF MESSAGE


Script Editor
File
Edit
View
Script
Font
Format
Window
Help
New ⌘N
New from Template
Open... ⌘O
Open Dictionary... ⇧⌘O
Open Recent
Close ⌘W
Save ⌘S
Duplicate ⇧⌘S
Rename...
Move To...
Export...
Revert To
Share
Page Setup... ⇧⌘P
Print... ⌘P
Messages
Mail
Droplets
Aperture
Cocoa-AppleScript Applet.app
Aperture Import Action.scptd

Mac Automation ?

- Automating tasks on macOS
  - Scripting supported Applications
  - UI element automation
  - System APIs

AppleScript

AppleScript

"If you're used to a programming language, AppleScript will drive you crazy"

— *Sal Soghoian*

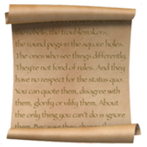
JavaScript

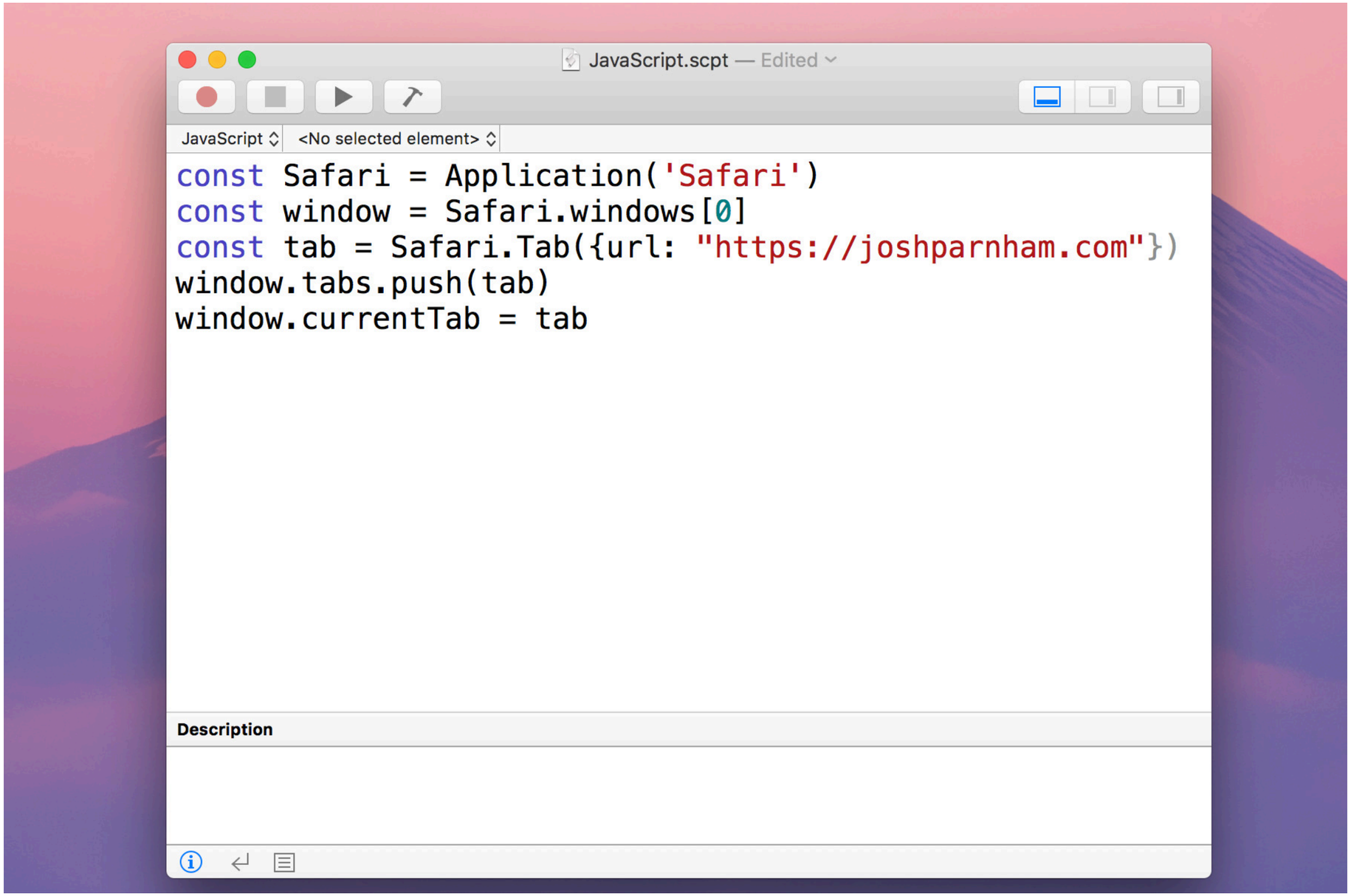
History 📚

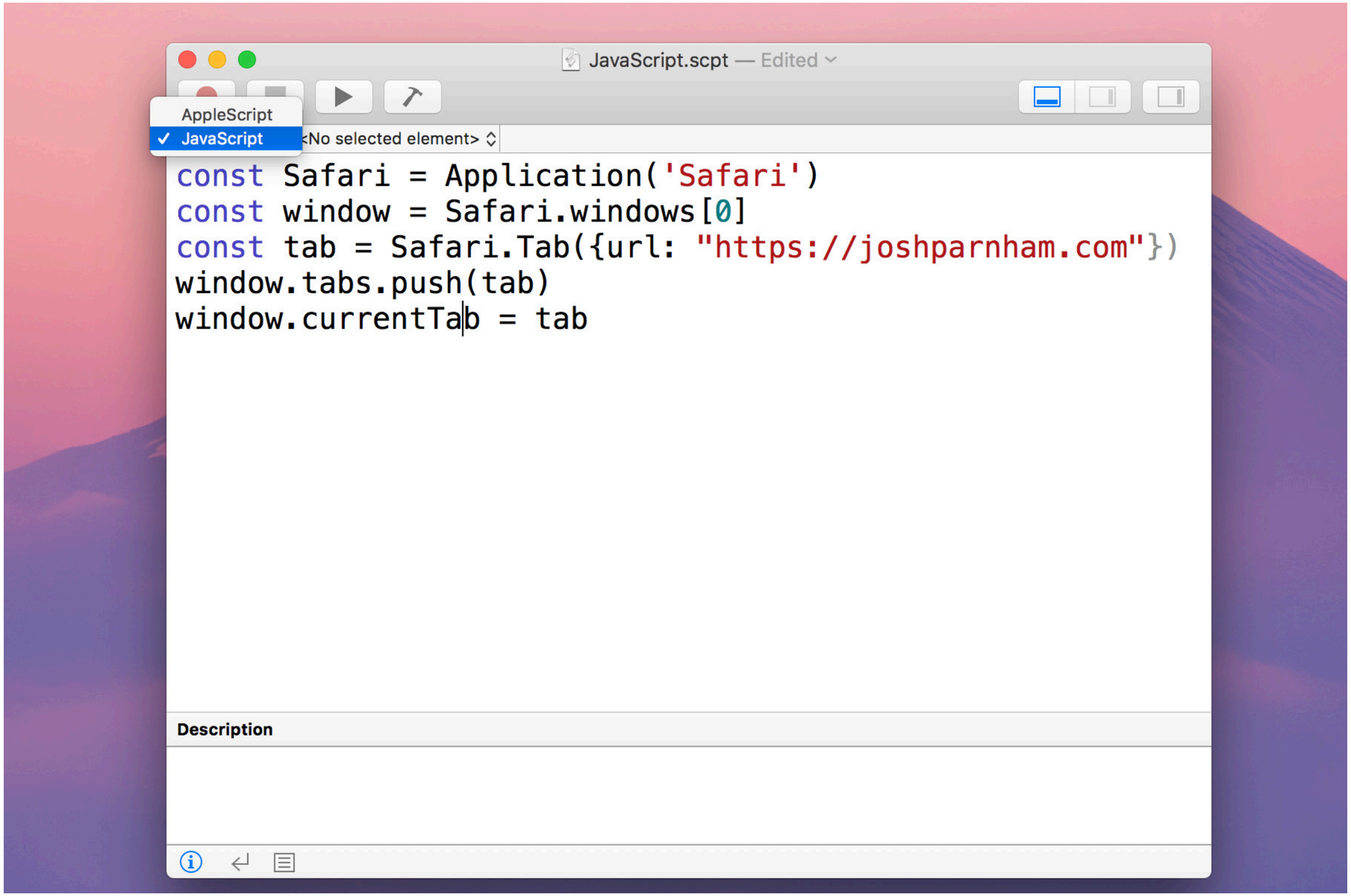
- OS X 10.10 Yosemite (2014)
- Open Scripting Architecture
  - `osascript/osacompile` and friends
- JavaScript implementation based on `JavaScriptCore`

Basics 👩‍💻

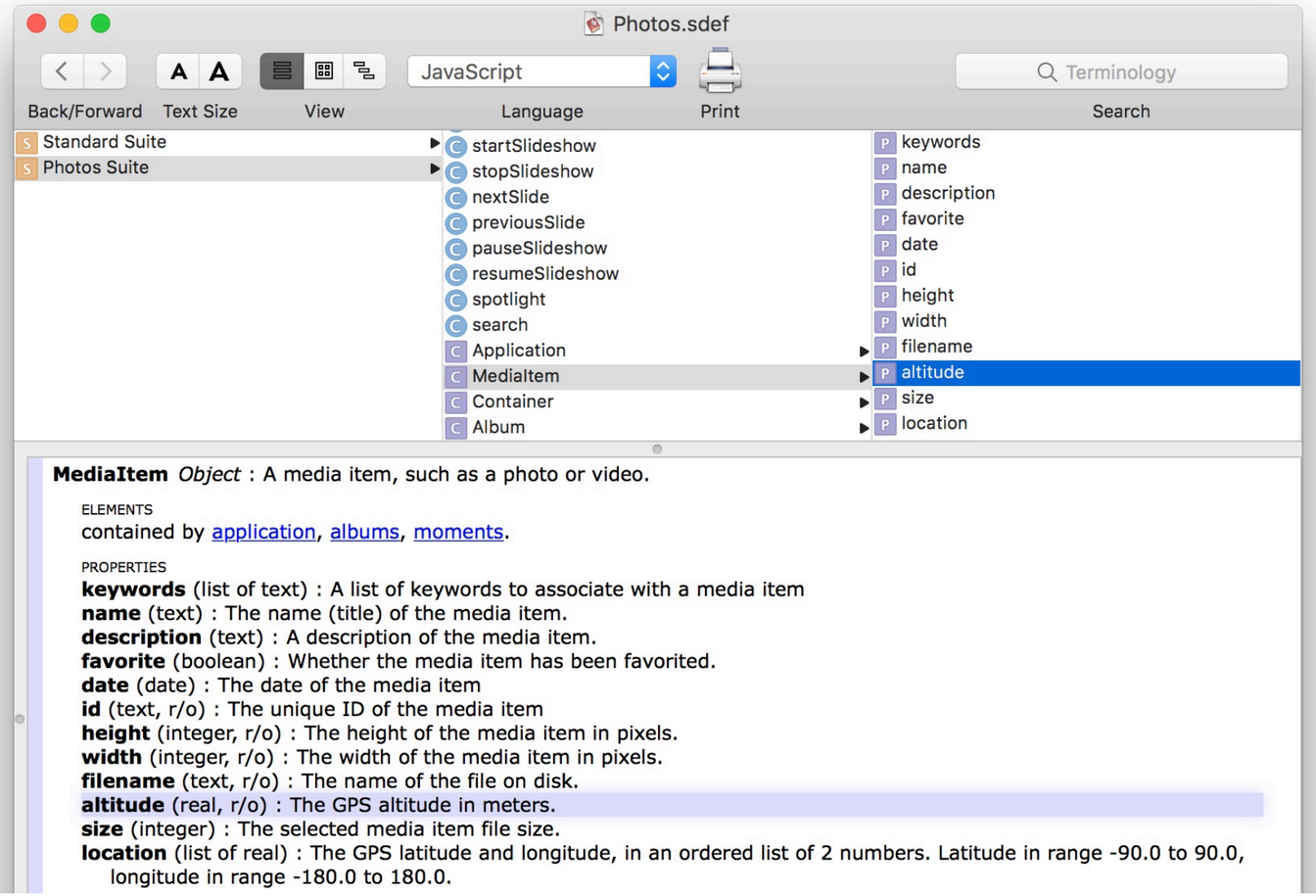
```
const Safari = Application('Safari')
const firstWindow = Safari.windows[0]
const tab = Safari.Tab({url: "https://joshparnham.com"})
firstWindow.tabs.push(tab)
firstWindow.currentTab = tab
```

Script Editor.app 

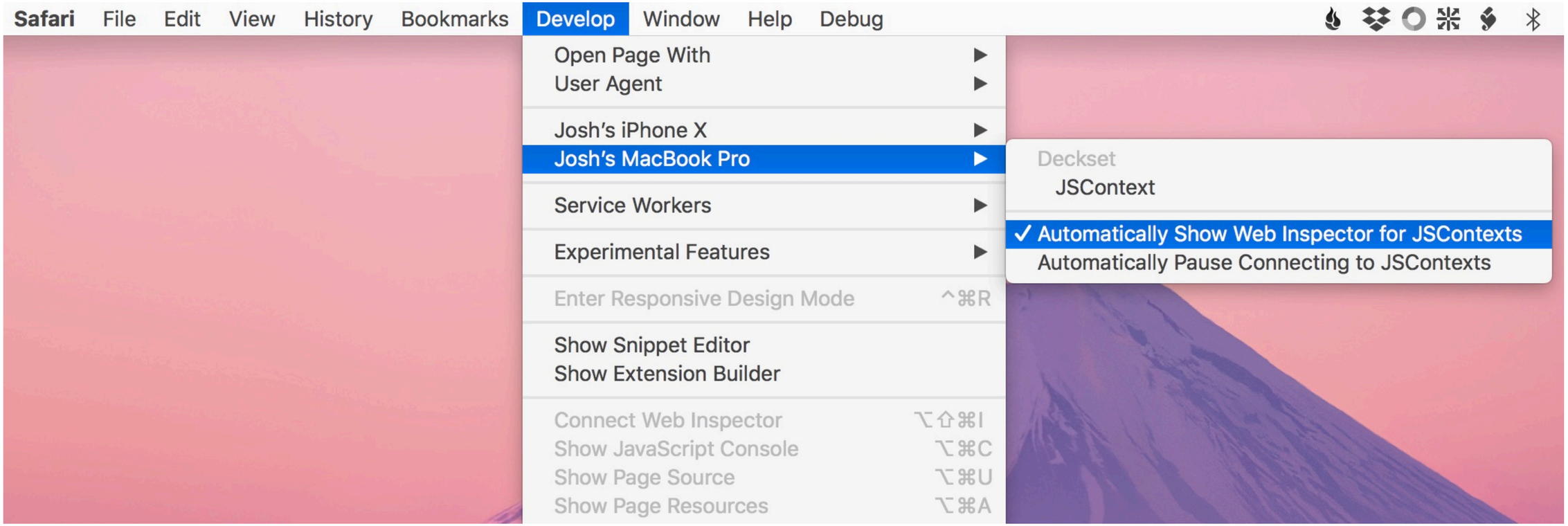

JavaScript.scpt — Edited
JavaScript
<No selected element>
const Safari = Application('Safari')
const window = Safari.windows[0]
const tab = Safari.Tab({url: "https://joshparnham.com"})
window.tabs.push(tab)
window.currentTab = tab
Description


JavaScript.scpt — Edited
AppleScript
JavaScript
<No selected element>
const Safari = Application('Safari')
const window = Safari.windows[0]
const tab = Safari.Tab({url: "https://joshparnham.com"})
window.tabs.push(tab)
window.currentTab = tab
Description

Documentation 📕



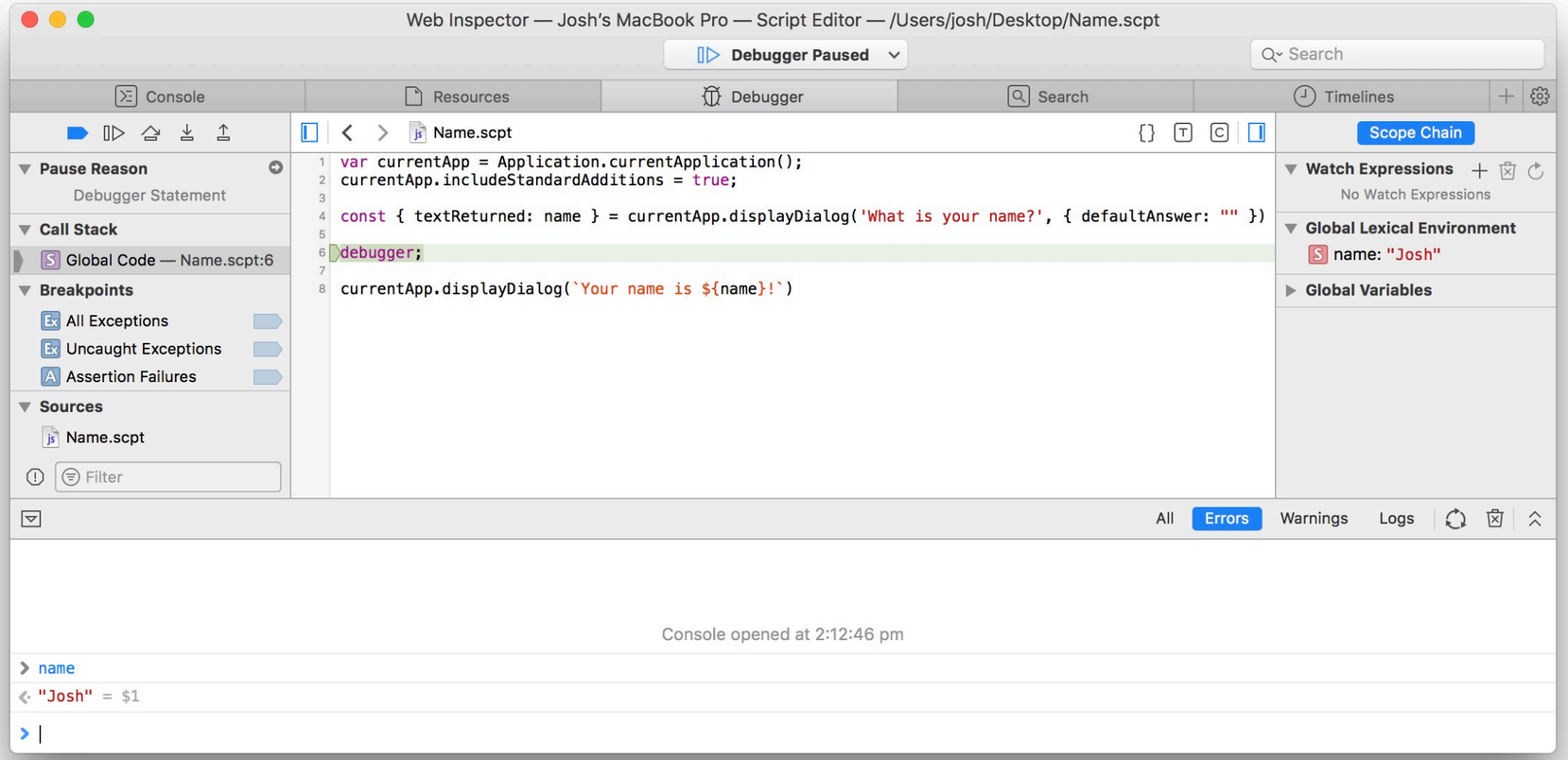
Debugging 🐛


Safari File Edit View History Bookmarks Develop Window Help Debug
Open Page With
User Agent
Josh's iPhone X
Josh's MacBook Pro
Service Workers
Experimental Features
Enter Responsive Design Mode ^⌘R
Show Snippet Editor
Show Extension Builder
Connect Web Inspector ⌥⇧⌘I
Show JavaScript Console ⌥⌘C
Show Page Source ⌥⌘U
Show Page Resources ⌥⌘A
Deckset
JSContext
✓ Automatically Show Web Inspector for JSContexts
Automatically Pause Connecting to JSContexts

Debugging 🐛

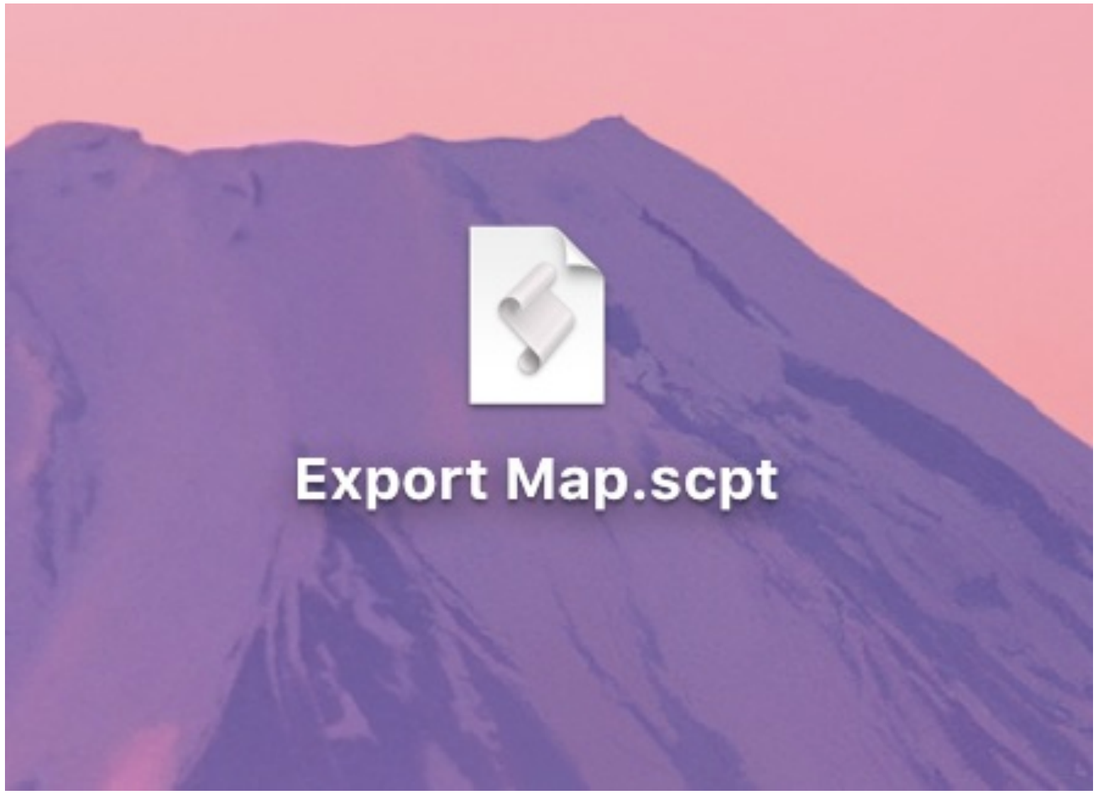
- `debugger;` statement in Script Editor

Debugging 🐛


Web Inspector — Josh's MacBook Pro — Script Editor — /Users/josh/Desktop/Name.scpt
Debugger Paused
Search
Console
Resources
Debugger
Search
Timelines
Name.scpt
Scope Chain
Pause Reason
Debugger Statement
Call Stack
Global Code — Name.scpt:6
Breakpoints
All Exceptions
Uncaught Exceptions
Assertion Failures
Sources
Name.scpt
Filter
1 var currentApp = Application.currentApplication();
2 currentApp.includeStandardAdditions = true;
3
4 const { textReturned: name } = currentApp.displayDialog('What is your name?', { defaultAnswer: "" })
5
6 debugger;
7
8 currentApp.displayDialog(`Your name is ${name}!`)
Watch Expressions
No Watch Expressions
Global Lexical Environment
name: "Josh"
Global Variables
All
Errors
Warnings
Logs
Console opened at 2:12:46 pm
> name
< "Josh" = $1
>

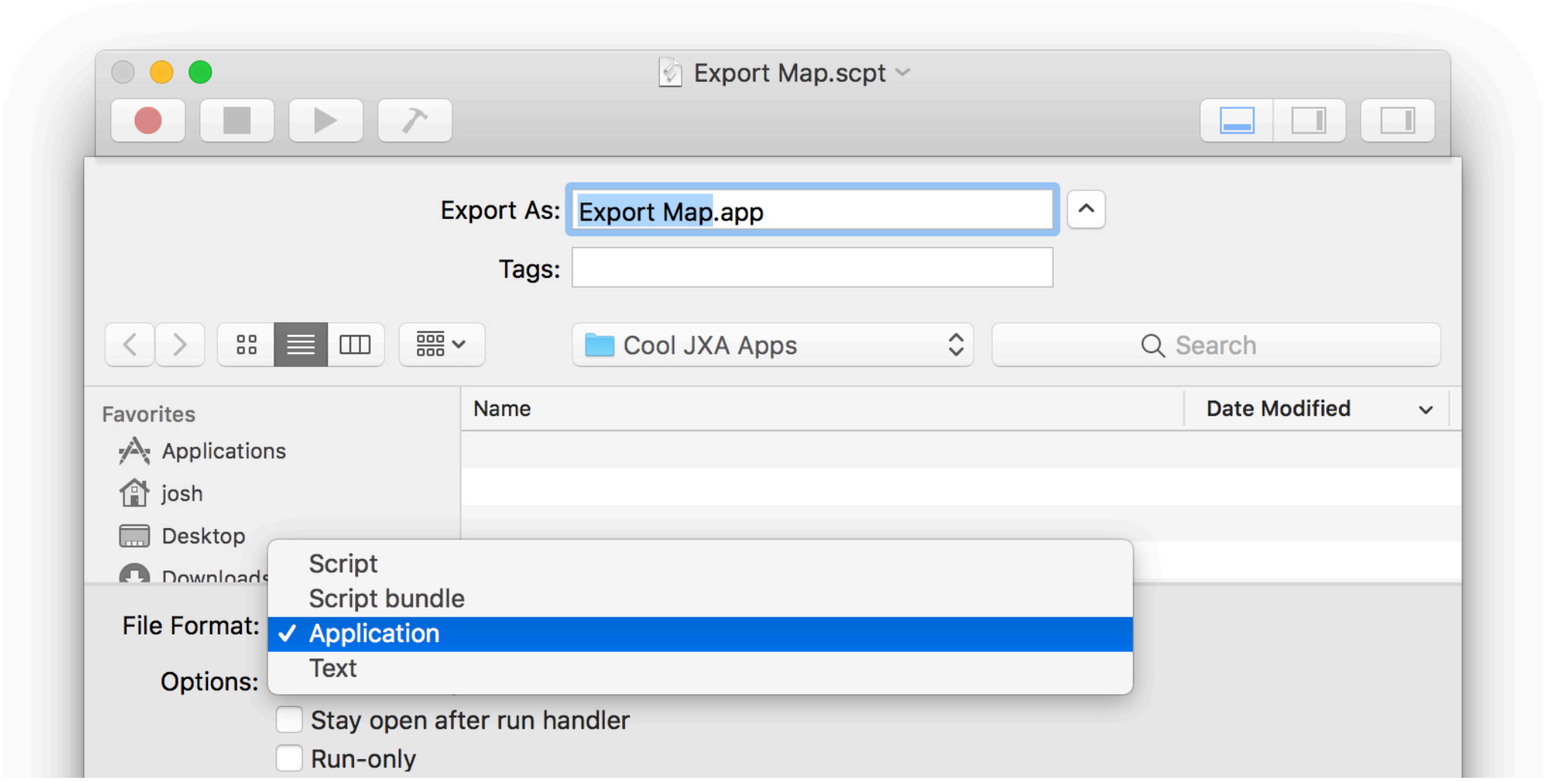
Invocation ✅

- `.scpt` file



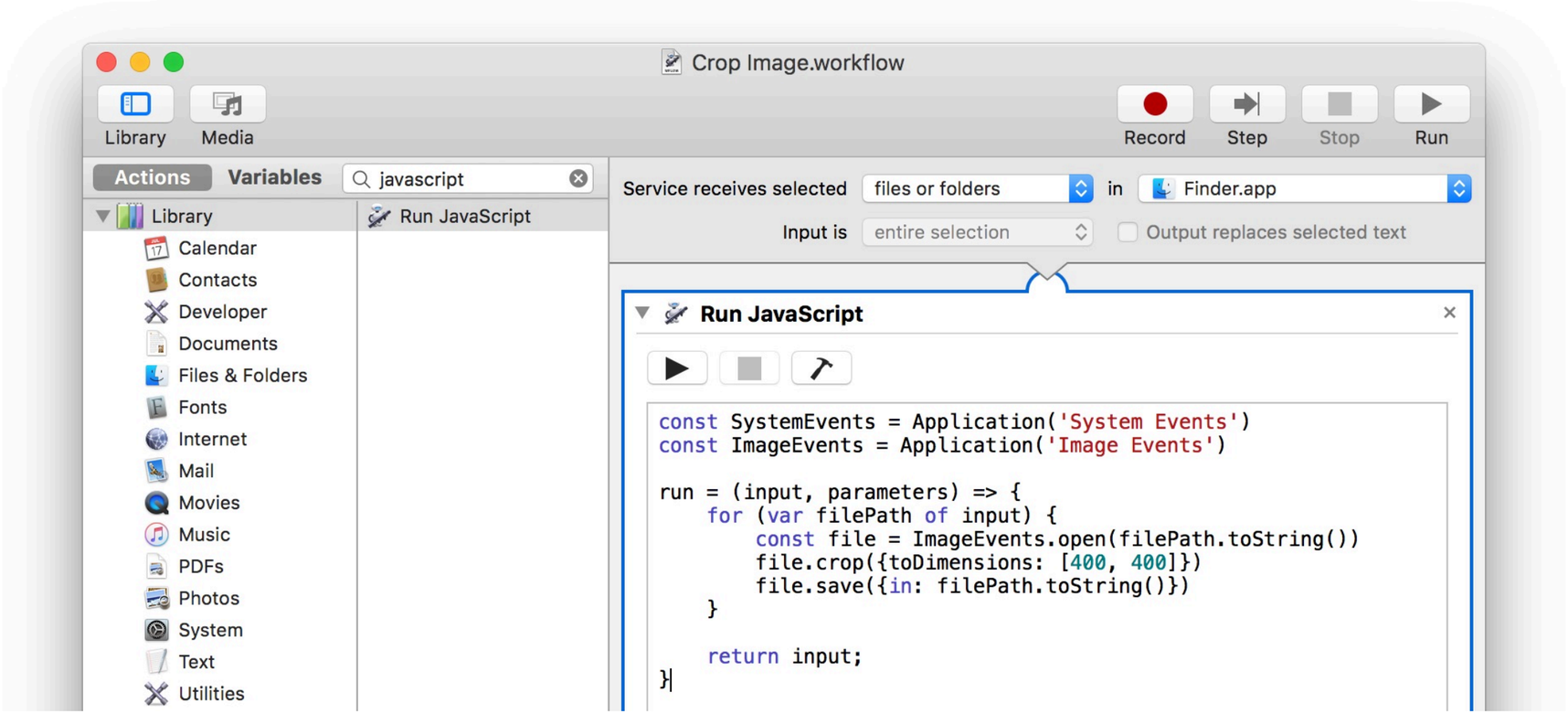
Invocation ✳️

- Applet

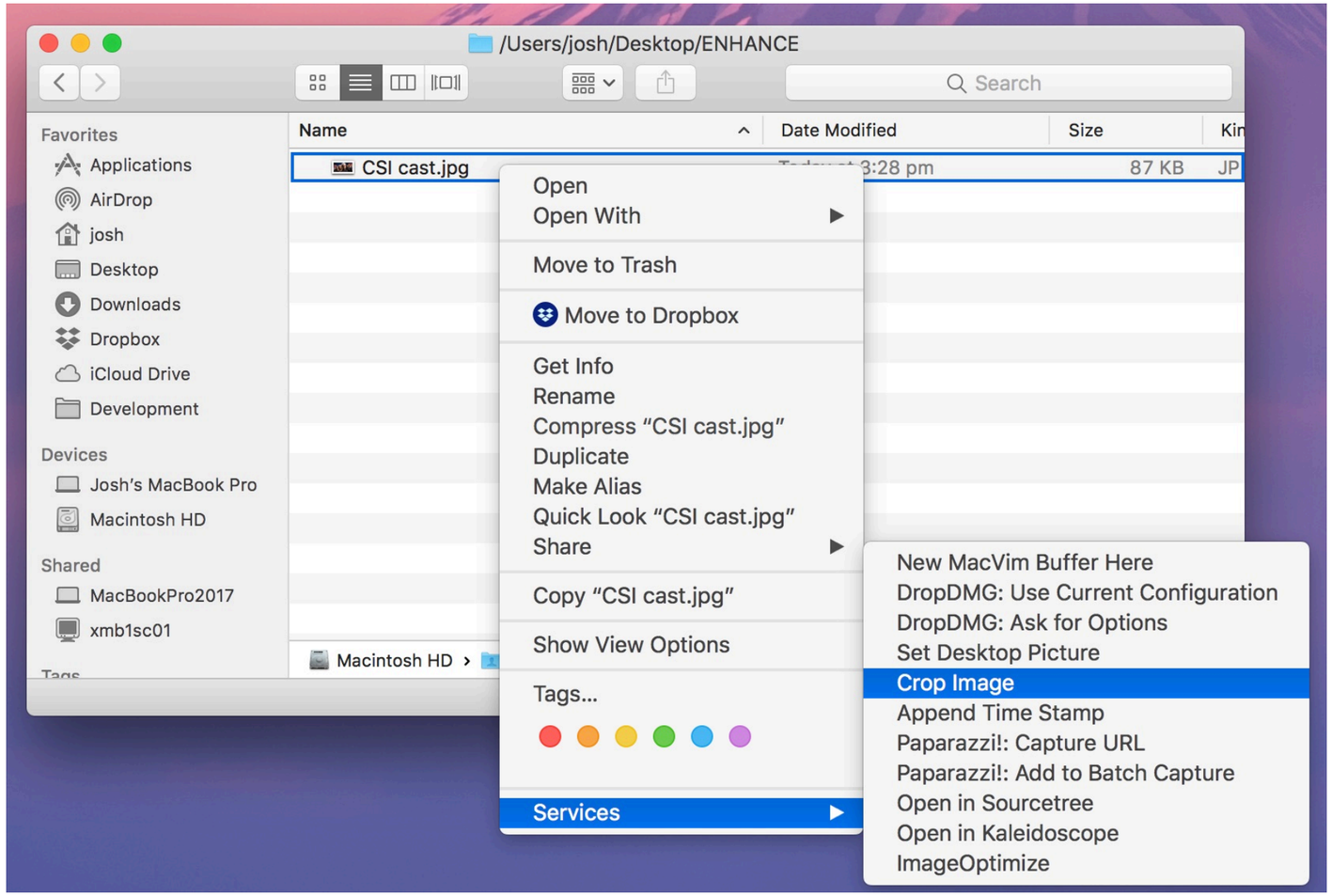


Invocation ✨

- Services menu

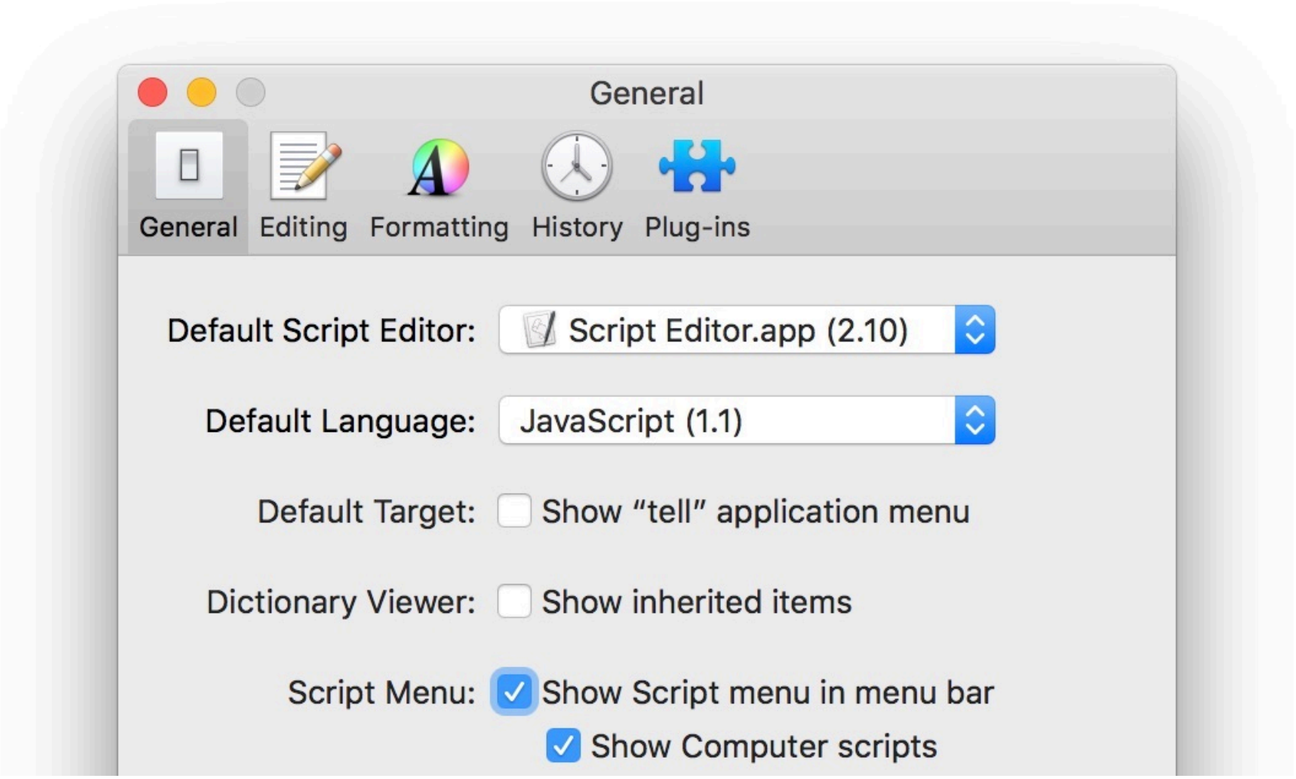


Invocation ✨


/Users/josh/Desktop/ENHANCE
Search
Favorites
Applications
AirDrop
josh
Desktop
Downloads
Dropbox
iCloud Drive
Development
Devices
Josh's MacBook Pro
Macintosh HD
Shared
MacBookPro2017
xmb1sc01
Name
Date Modified
Size
CSI cast.jpg
87 KB
Macintosh HD
Open
Open With
Move to Trash
Move to Dropbox
Get Info
Rename
Compress "CSI cast.jpg"
Duplicate
Make Alias
Quick Look "CSI cast.jpg"
Share
Copy "CSI cast.jpg"
Show View Options
Tags...
Services
New MacVim Buffer Here
DropDMG: Use Current Configuration
DropDMG: Ask for Options
Set Desktop Picture
Crop Image
Append Time Stamp
Paparazzi!: Capture URL
Paparazzi!: Add to Batch Capture
Open in Sourcetree
Open in Kaleidoscope
ImageOptimize

Invocation ✳️

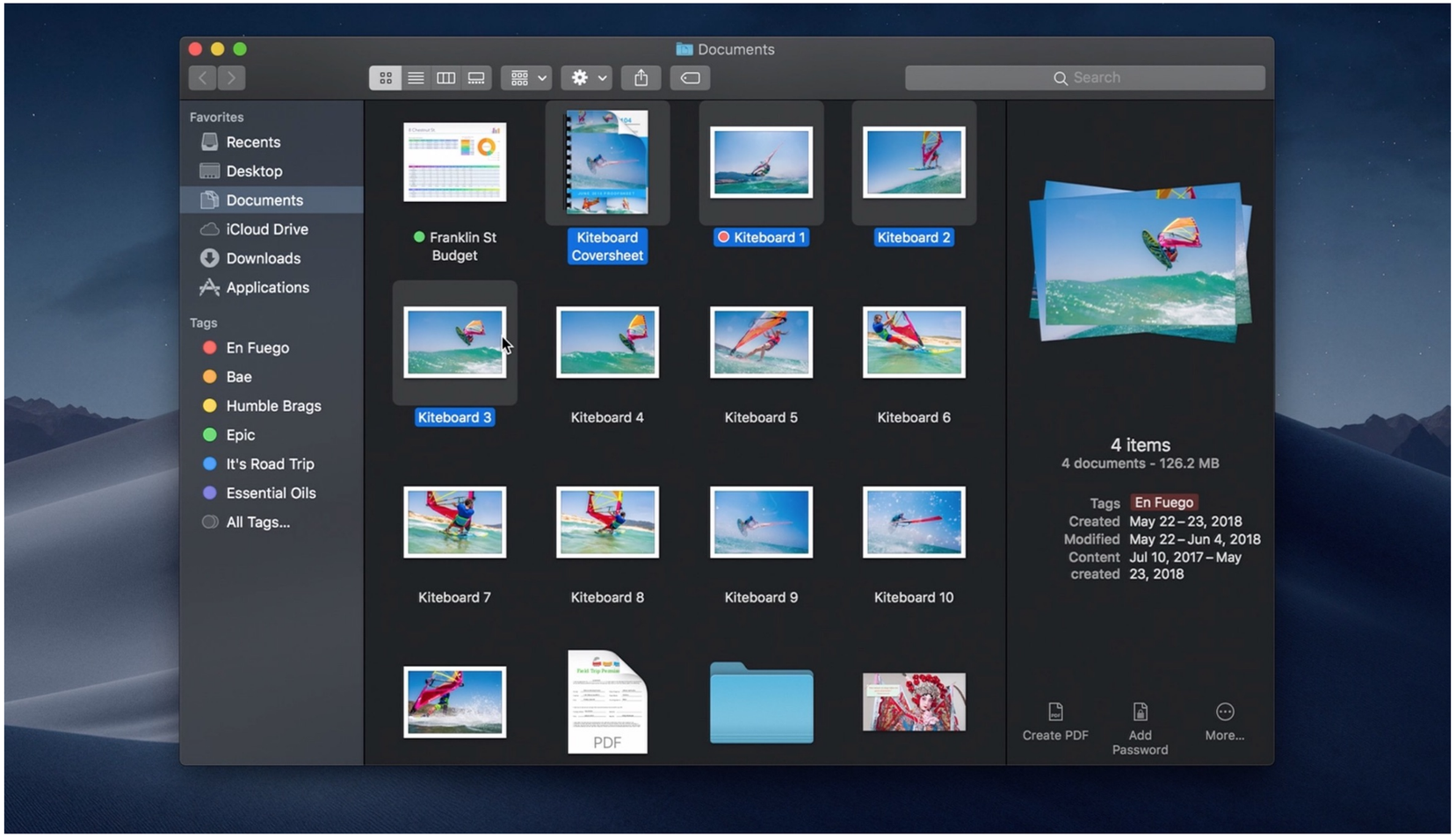
- System-wide Scripts menu



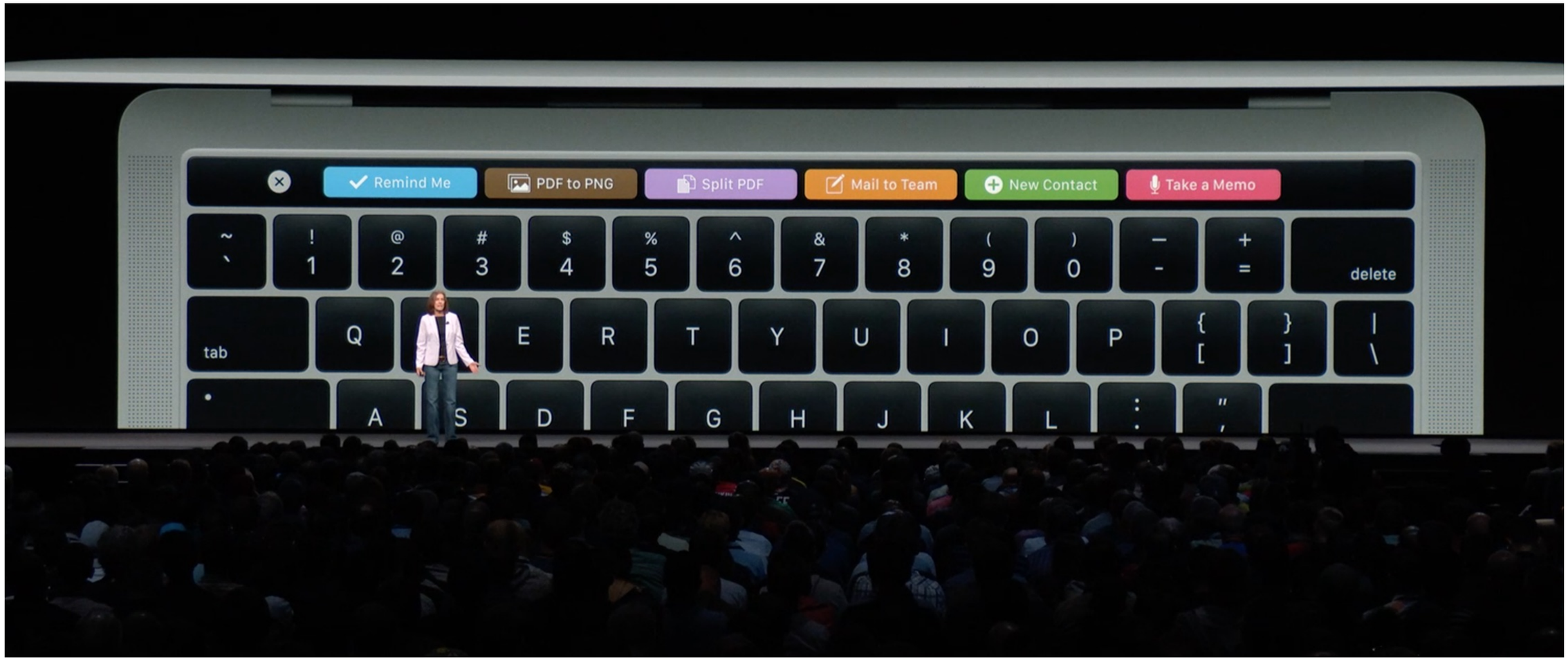
Invocation ✅

- Folder contents changed
- Calendar alarms
- Dictation command
- and others 🚀

Even Better Invocation 🏜

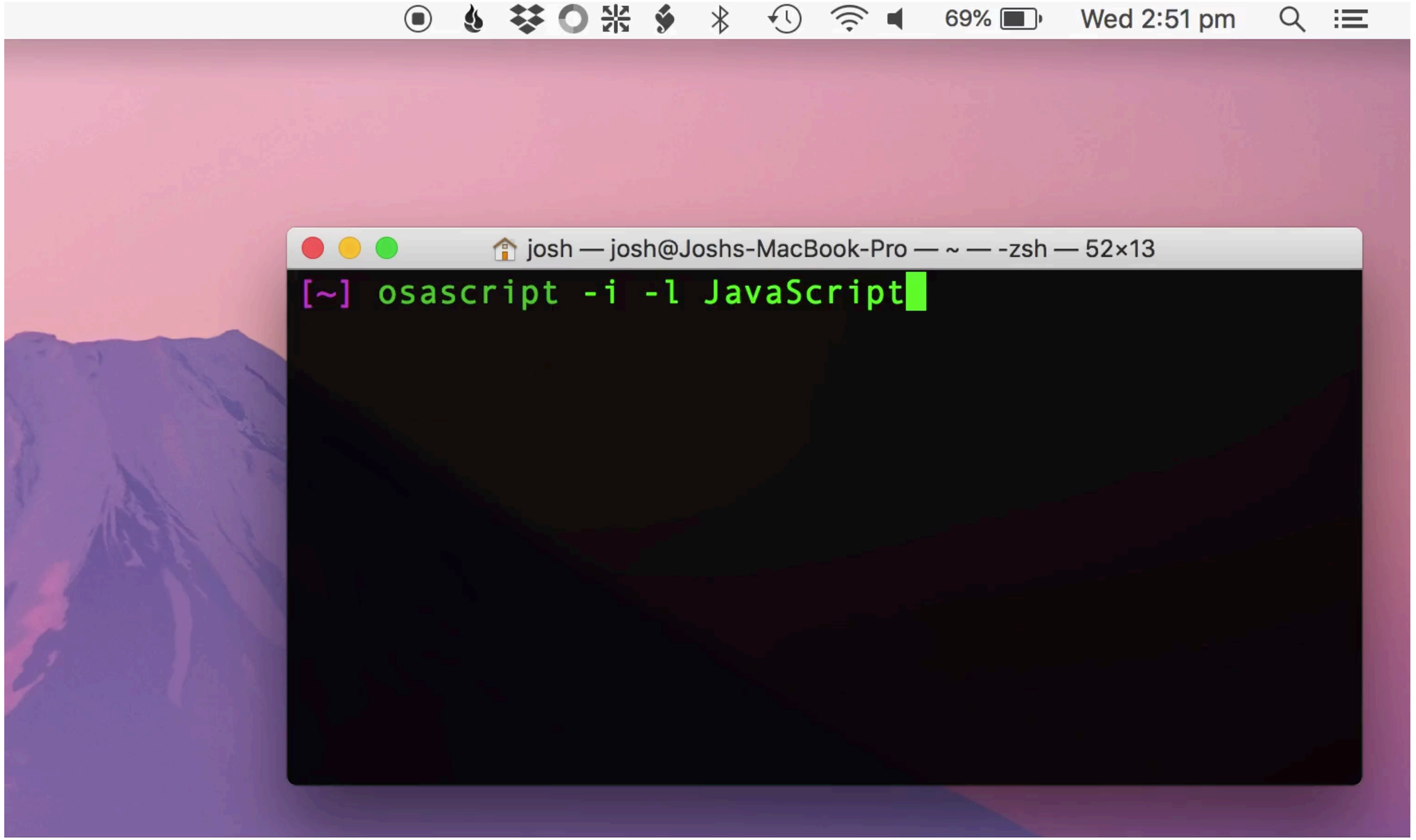


Even Better Invocation 🏜



REPL ⌨️

```
$ osascript -i -l JavaScript
```


69% Wed 2:51 pm
josh — josh@Joshs-MacBook-Pro — ~ — -zsh — 52×13
[~] osascript -i -l JavaScript

Scripting ⚡

Example: `args.sh`

```
#!/usr/bin/env osascript -l JavaScript

run = argv => console.log(
  argv.map(x => x.padStart(10)).join('\n')
)
```

```
$ ./args.sh Hello there MelbJS! 👋

    Hello
    there
  MelbJS!
       👋
```

Interoperability ☕

```
const result = 'hello'.replace('e', '3')
```

Interoperability ☕

```
const result = ObjC.wrap('hello')
.stringByReplacingOccurrencesOfStringWithString('e', '3').js
```

Interoperability Examples ☕

```
#!/usr/bin/env osascript -l JavaScript

ObjC.import('DictionaryServices')

run = argv => {
    const word = argv[0]
    const range = { 'location': 0, 'length': word.length }
    let definition = $.DCSCopyTextDefinition(null, word, range)
    console.log(definition.js.split(" | ").join("\n"))
}
```

Interoperability Examples ☕

```
$ ./define.sh JavaScript

JavaScript
ˈdʒɑːvəˌskrɪpt
noun [mass noun] trademark an object-oriented computer programming
language commonly used to create interactive effects within web browsers.
ORIGIN 1990s: from Java2 + script1.
```

Interoperability ☕

- Called the `JavaScriptObjC` bridge

JavaScript Context ➡️ Objective-C Context

- Convert primitive JS type to ObjC object

```
ObjC.wrap(...)
```

```
$(...)
```

Objective-C Context ➡️ JavaScript Context

- Convert ObjC object to JS type

`ObjC.unwrap(...)`

`.js`

Applications 🖥️

- Interoperability with `AppKit`
- Native UIs purely in (bridged) JavaScript

```
ObjC.import('Cocoa')

const resultString = (result) => `Result: ${ result ? result.toString() : ''}`

ObjC.registerSubclass({
    name: 'AppDelegate',
    methods: {
        'calculate:': {
            types: ['void', ['id']],
            implementation: function (sender) {
                var total =
                    Number(textField1.stringValue.js) +
                    Number(textField2.stringValue.js)
                resultTextFieldLabel.stringValue = resultString(total)
            }
        }
    }
})
```

```
var appDelegate = $.AppDelegate.alloc.init

const styleMask =
    $.NSTitledWindowMask |
    $.NSClosableWindowMask |
    $.NSMiniaturizableWindowMask

var window =
    $.NSWindow.alloc.initWithContentRectStyleMaskBackingDefer(
        $.NSMakeRect(0, 0, 250, 120),
        styleMask,
        $.NSBackingStoreBuffered,
        false
    )
```

```
var calculateButton =
    $.NSButton.alloc.initWithFrame(
        $.NSMakeRect(25, 20, 200, 25)
    )
calculateButton.title = 'Calculate 🧙‍♂️'
calculateButton.bezelStyle = $.NSRoundedBezelStyle
calculateButton.buttonType = $.NSMomentaryLightButton
calculateButton.target = appDelegate
calculateButton.action = 'calculate:'
calculateButton.keyEquivalent = '\r'
window.contentView.addSubview(calculateButton)
```

```
var resultTextFieldLabel =
    $.NSTextField.alloc.initWithFrame(
        $.NSMakeRect(25, 45, 200, 24)
    )
resultTextFieldLabel.stringValue = resultString()
resultTextFieldLabel.drawsBackground = false
resultTextFieldLabel.editable = false
resultTextFieldLabel.bezeled = false
resultTextFieldLabel.selectable = true
window.contentView.addSubview(resultTextFieldLabel)
```

```
var textField1 =
    $.NSTextField.alloc.initWithFrame(
        $.NSMakeRect(25, 80, 90, 24)
    )
textField1.stringValue = ''
window.contentView.addSubview(textField1)
```

```
var plusTextFieldLabel =
    $.NSTextField.alloc.initWithFrame(
        $.NSMakeRect(120, 80, 24, 24)
    )
plusTextFieldLabel.stringValue = '+'
plusTextFieldLabel.drawsBackground = false
plusTextFieldLabel.editable = false
plusTextFieldLabel.bezeled = false
plusTextFieldLabel.selectable = true
window.contentView.addSubview(plusTextFieldLabel)
```

```
var textField2 =
    $.NSTextField.alloc.initWithFrame(
        $.NSMakeRect(135, 80, 90, 24)
    )
textField2.stringValue = ''
window.contentView.addSubview(textField2)
```

```
window.center
window.title = 'Scientific Calculator 🙈'
window.makeKeyAndOrderFront(window)
```

Packaging 📦

Either through `Script Editor.app` or `osacompile`.

```
osacompile -l JavaScript -o Calculator.app -s Calculator.js
```

Demo

PS 🐙

QLScpt

```
brew cask install josh-/qlscpt/qlscpt
```

Caveats ✋

- Limited documentation
- Buggy
- Majority of automation code is still AppleScript
- 🐲 **Here be dragons** 🐉

Thanks ✌️

@joshparnham

github.com/josh-