Automation
Getting the Mindset for IT



acmeFoo

Scott M. Neal
smn.mg@acmefoo.org
MacTech SF 16 March 2016
Copyright 2016 MindsetGarden

Goals for this Session

- Present a step-by-step method by which you can create your own automations
- Present some pre-existing automations as “inspiration”

Goals

Empower non-programmers to utilize developer tools to solve problems

- Obtain the correct mindset for developing automations while learning the the basics of tools that enable you to automate
  - Automator
  - AppleScript
  - Web automation (time permitting)
- Provide a foundation for learning automating techniques
  - future courses
  - on your own

Non-programmers may be

- programmers-of-the-future
- managers



The Automating Mindset

Have you ever noticed that programmers seem to know MANY languages?

- This is (almost) unheard of in Human linguistics

Programming is a mindset

- Some have it naturally
- It is learnable
  - ...provided you can be organized!

Many programmers/scripters/automaters started out NOT with Computer Science but as

- Linguists
- Musicians

A programmer is worth his/her weight in gold

- ...and some charge that!
- But wouldn't you rather know what they know?



4

Goals

This session is designed to:

- Briefly present some basic concepts:
- Show some beyond-basic examples that you can take home with you

The focus will be on making sure you can automate independently

- Learn mindset
- Automate in multiple environments
- Read and understand documentation
  - because no programmer knows how to do EVERYTHING off the top of his/her head
  - Many "introductory" books aren't as introductory as they need to be (they don't teach mindset)
    - but are a GOLDMINE of info once you have the mindset

This is NOT a "dump demos and facts" presentation



Automating Mindset: Agenda

Goal:

- To remove (as many as possible) impediments that prevent you from creating your own automations

Materials derived from

- 3-5 day acmeFoo Auto101/201 courses and 2 day CLI101 course
- 1-day MacTech Conference seminar (Fall)
  - To have you start working on your own automations
    - Organizing your plan
    - Converting that plan into an actual automation
  - You will NOT have this stuff stick in your brain until you actually write an automation to solve one of your own problems!



Goals

You will NOT become automating experts here… But:

- You will have the basics you will need to do so
  - including knowing how and where to ask questions
- You will have access to an Automating expert to ask questions

Don't all of you want to be empowered to automate?

- It's within your reach
  - …unlike creating matter with your bare hands or levitating above traffic
- "Real programmers" (whatever that means) do NOT have everything memorized
  - "Be inspired"



Why learn about Automating?

People don't automate because they are lazy, but because they are EFFICIENT!

Manually managing hundreds/thousands of devices is tedious and prone to error

You may not write your own complicated automations from scratch, but it is CRITICAL that you understand them so you can tweak to your specific needs



Real Goal

More spare time for you!



Automation Goal and a Plan:

The Automating Mindset

A Car Mechanic:

- doesn't learn how to fix one car--learns how to fix "cars"
- applies that mindset to specific makes & models

A Musician

- doesn't memorize every song in a vacuum
  - Keys
  - Tunings

You can be a successful musician or car mechanic (or whatever) without memorizing EVERYTHING about cars or music

- Car mechanic doesn't need to know the physics of tires
- Musician doesn't need to know the physics of sound

‘‘Wax-On Wax-Off’’

Please be patient MacTech-san… You WILL get Automating time!



Think like a Computer

Computers don't actually "think", they "process"

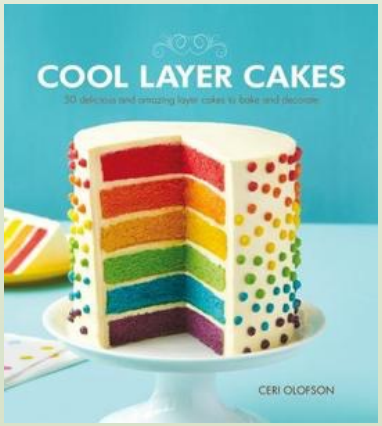
- Step-by-step sequential execution of statements
  - Same order as English reading: start at the top left, work your way to the right of a statement one character/word at a time, and then when one statement is done, execute the next statement
  - Flow Control constructs modify linear execution of statements
- Predictable decisions
  - Bad: "Do we have enough room in the car?"
  - Good: "If I need room for 7 passengers, and I can find out how many seats are available, is the number of seats available greater than or equal to 7?"
- Can't be vague
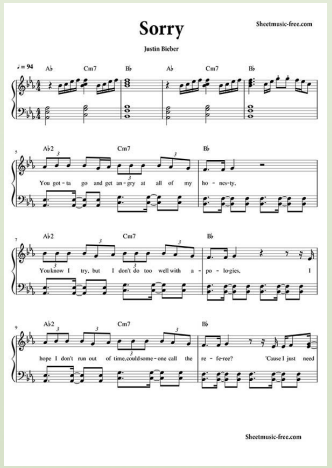


Think like a Computer

Railroads are a great analogy for how computers work

- You aren't riding the train, you are laying the track
  - Based on the "building blocks" that the IDE gives you
- You may prefer to think of it as
  - "following a recipe" vs. "making a recipe"
  - The musician or musical device vs. MIDI or sheet music









If you can organize your thoughts, create plans, and execute them, you can automate!

Decisions


IF THERE IS SOMEONE BEHIND YOU AND NO ONE IN FRONT OF YOU, THEN GET OUT OF THIS LANE
STAY IN THIS LANE UNLESS YOU ARE GOING TO PASS
EXIT 9W
IF YOU CAN'T FIGURE OUT WHY, THEN TAKE THIS LANE

Events

These come at you whether you want them to or not

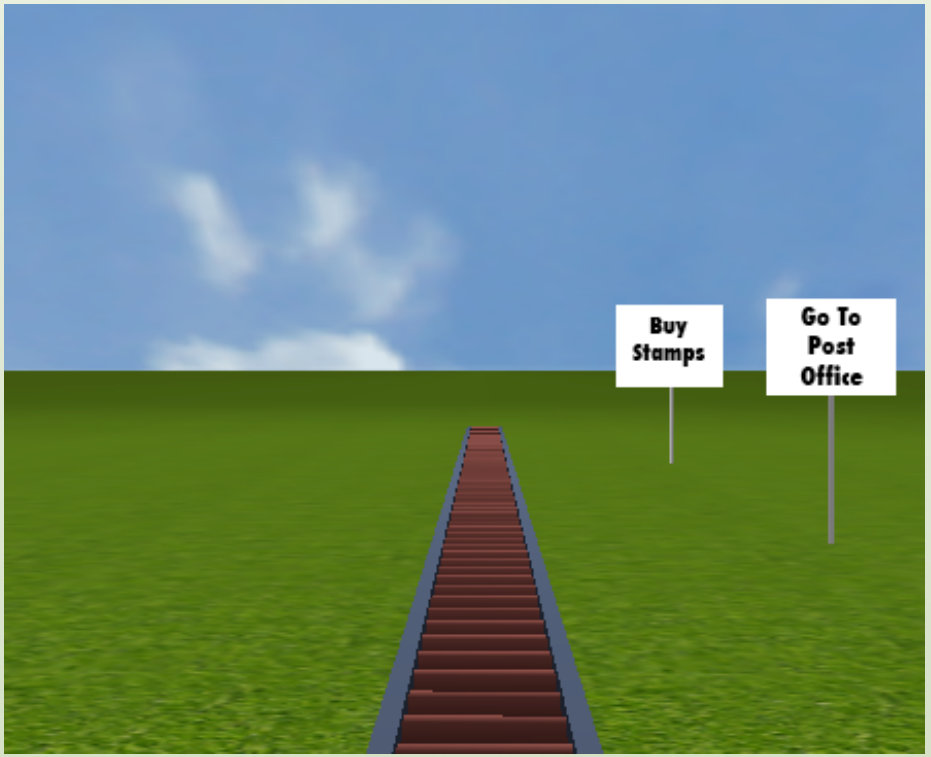
- Keyboard/Mouse click
- Volume gets mounted
- Drive gets unplugged

Many “crashes” in software are from incomplete handling of all events

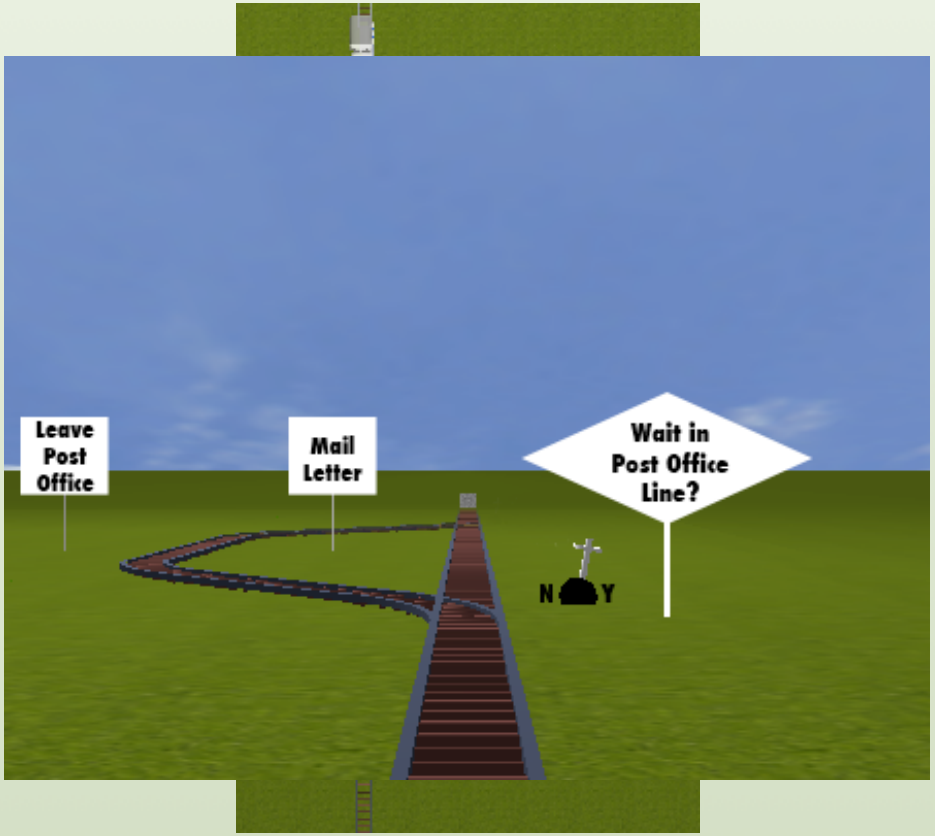




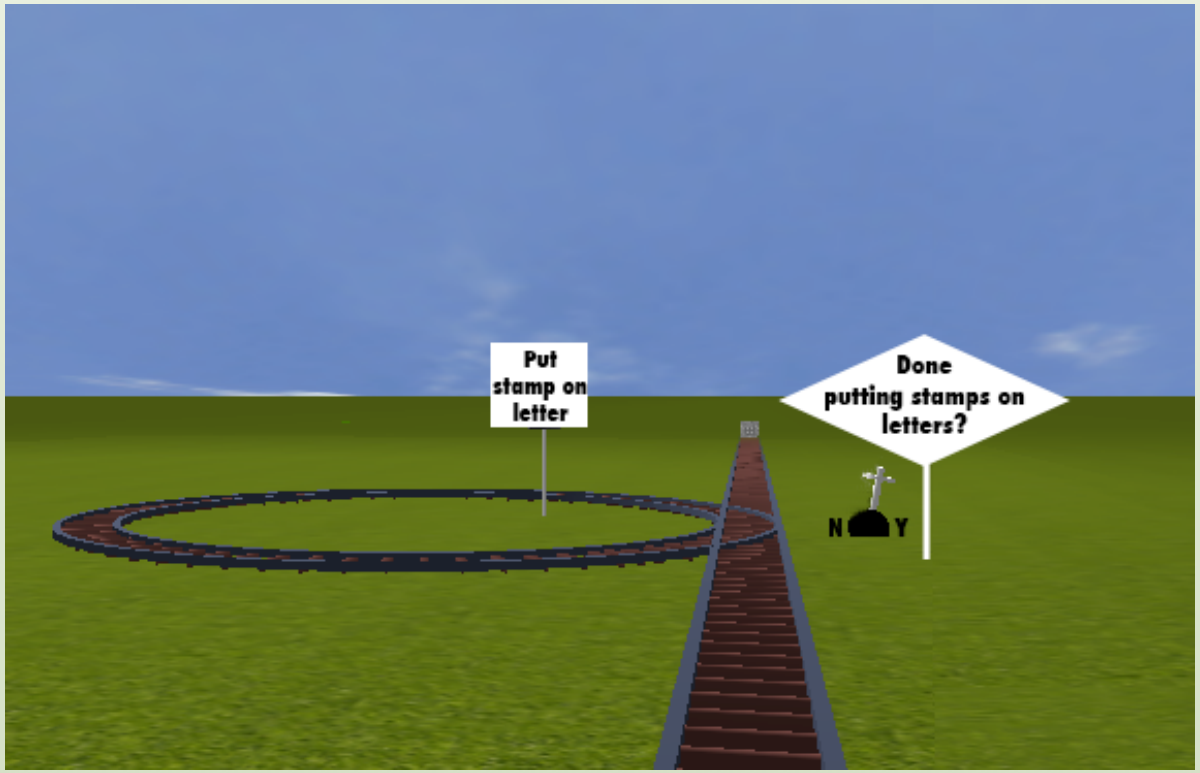
Automating is ‘‘railed’’



Decisions, still ‘‘railed’’



Repetition with Loops



IDE: Workflows, Scripts, Programs

Depending on the tools you use, you will be developing Workflows, Scripts, or Programs (or a combination) to turn your project plan into reality

- We will use the umbrella term **Automation**(s) with a capital A

We will develop Automations using Automating-specific tools

- Work together in an **Integrated Development Environment** (**IDE**)



MACTECH

20

IDE: Integrated Development Environment

Provides way of importing or creating information

- Text Editor(s)
  - Syntax coloring

Provides an interface to documentation

Provides **debugging** assistance

Trying to do all of this by yourself would be MASSIVELY overwhelming



Turning an Automation into an OS Process

An Automation, the great culmination of your plan, typically consists of one or more input files called **source code**

Your source code is linked with pre-existing libraries/ frameworks

The running OS Process created from your Automation can be executed two ways

- **Interpreted**
- **Compiled**



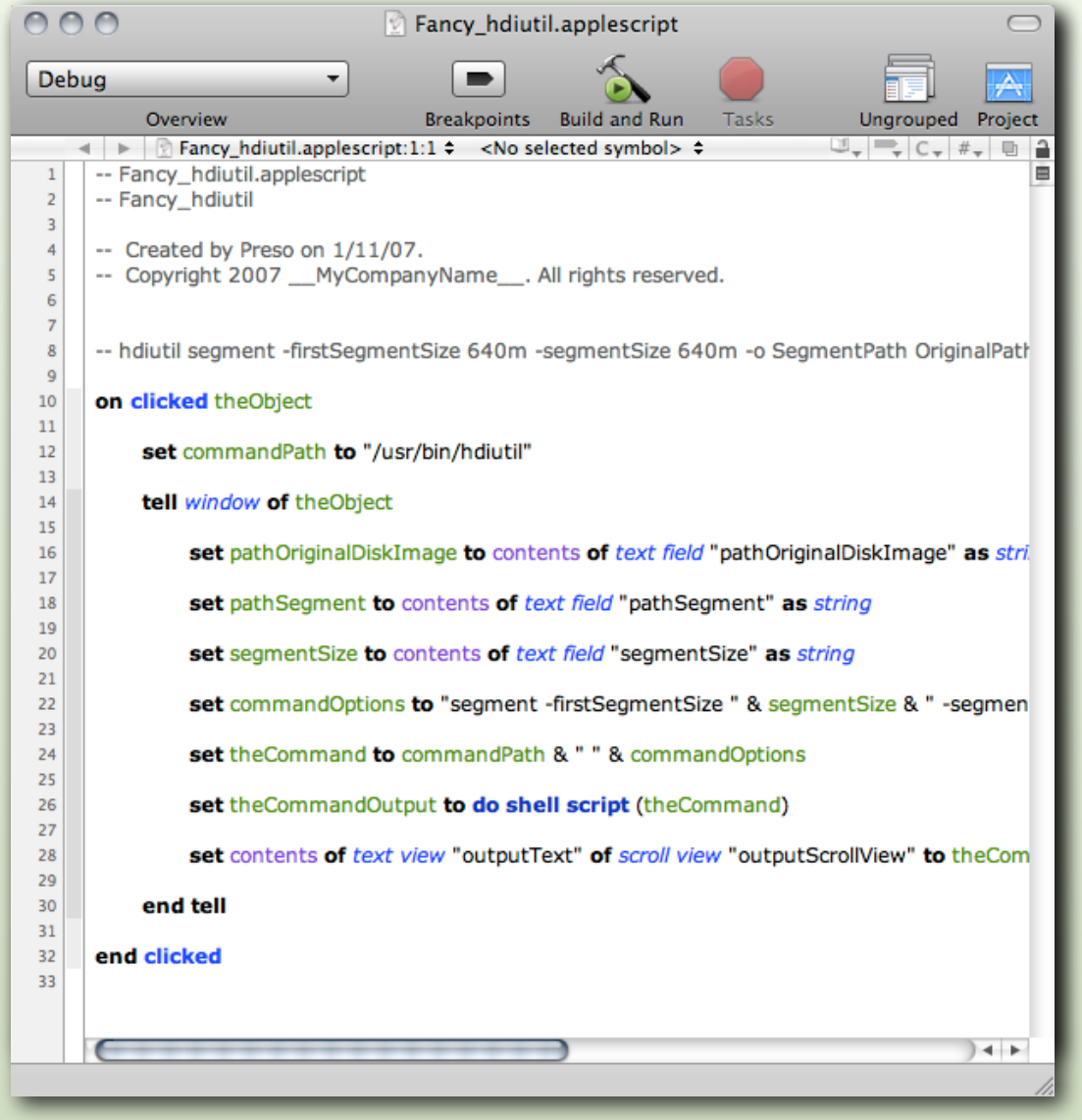
MACTECH

22

IDE Text Editor

IDE Provides a text editor

- A place to create and edit your source code
- Some feature **Syntax Coloring**
- Some highlight code blocks and nesting



Interpreted vs. Compiled

Code/Resources can be interpreted or compiled

- Interpreted
  - Process runs in native OS executable format
  - Done when Automation is executed
- Compiled
  - Process runs in native OS executable format
  - Components
    - Source Code
    - Object Code
    - Linking

However it's done, your code/resources need to be converted into something that the OS can actually execute



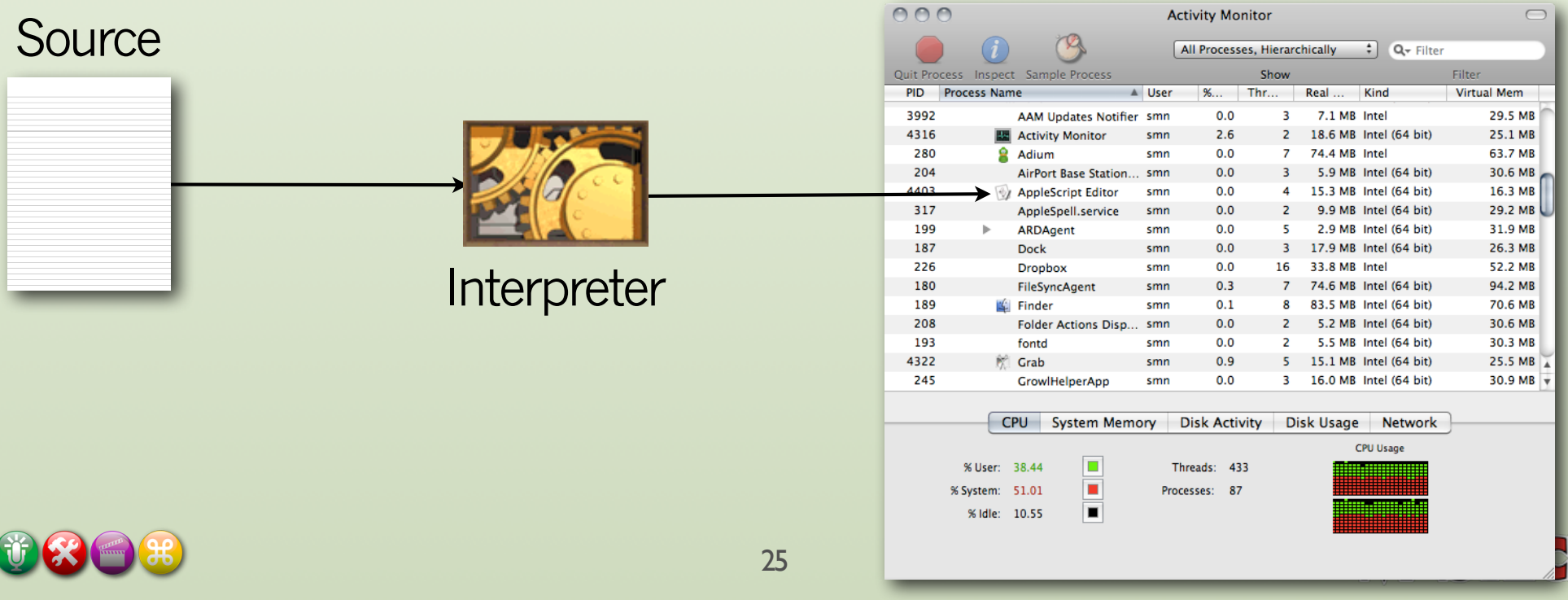
MACTECH

24

Interpreted Automation

With an Interpreted Automation, the Automation process is the language or IDE itself

- Source code is interpreted on-the-fly, line-by-line
  - Non optimized, slower
- Process is seen by the OS as the language/IDE, not the name of your Automation

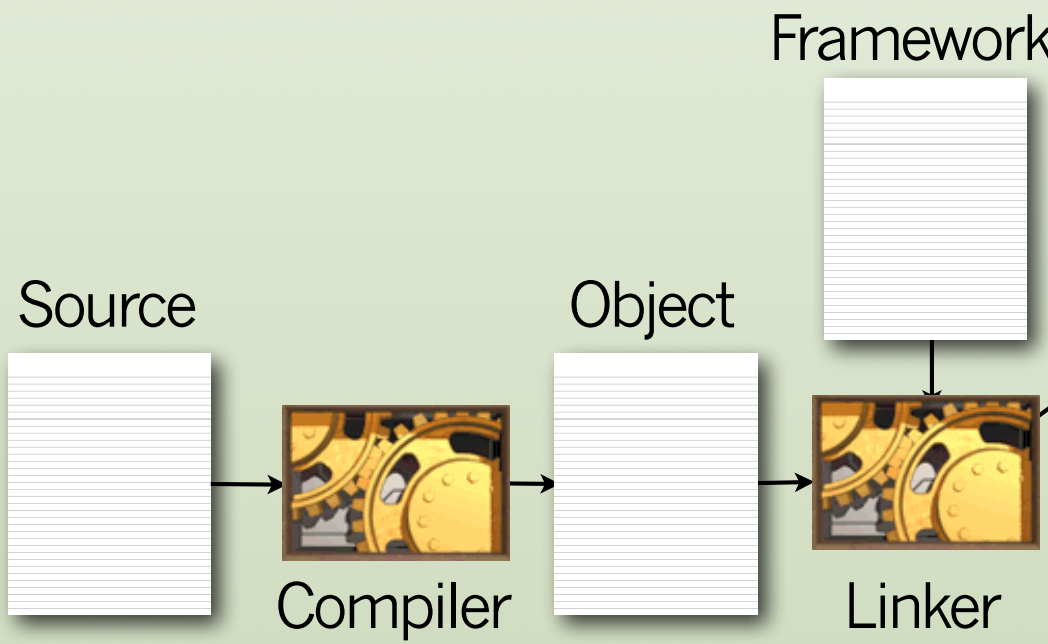


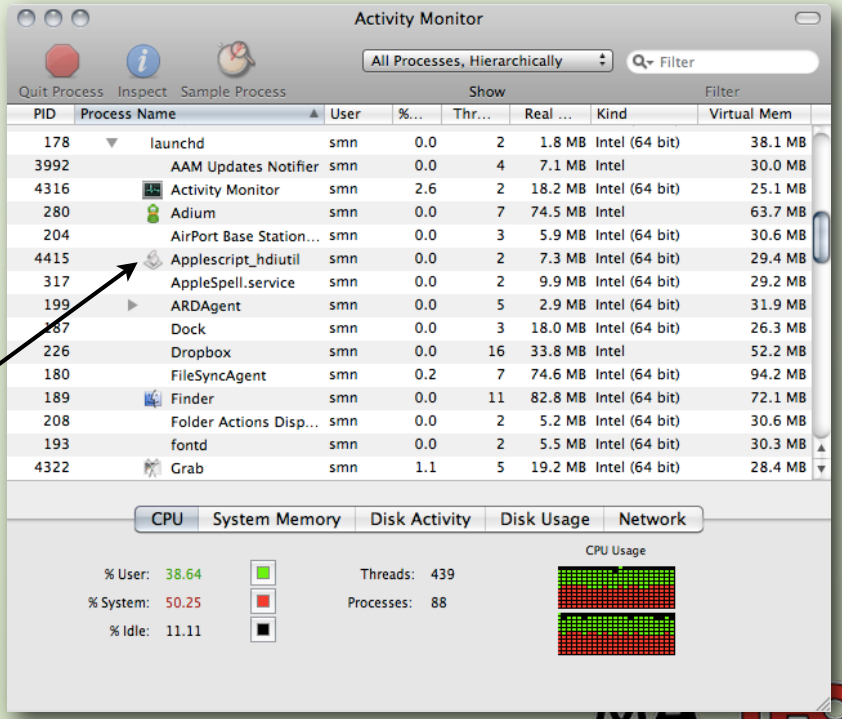
25

Compiled Automation

With a Compiled Automation, the Automation is transformed into an OS-native executable by a **compiler** and then run as its own process

- Source code is compiled by the IDE into **object code**
- Object code is turned into an executable via **linking**
- Process is seen by the native OS as a separate, unique executable with IDE-given name





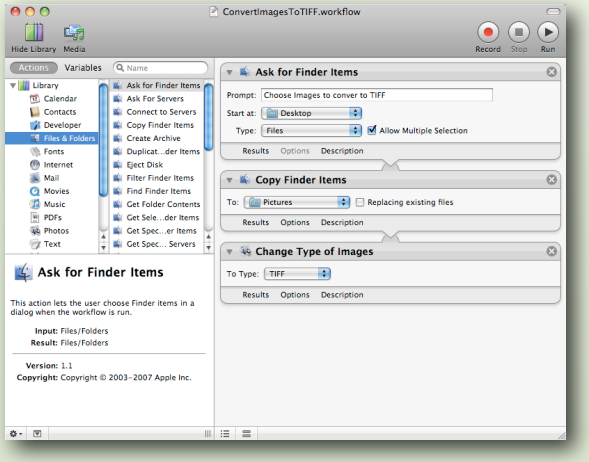
26

Typical Mac OS X IDEs

Automator



- Built-in:
  - GUI editor and organizer
  - Documentation
- Automation Languages:
  - Automator itself
  - AppleScript (behind-the-scenes)
- Interpreted



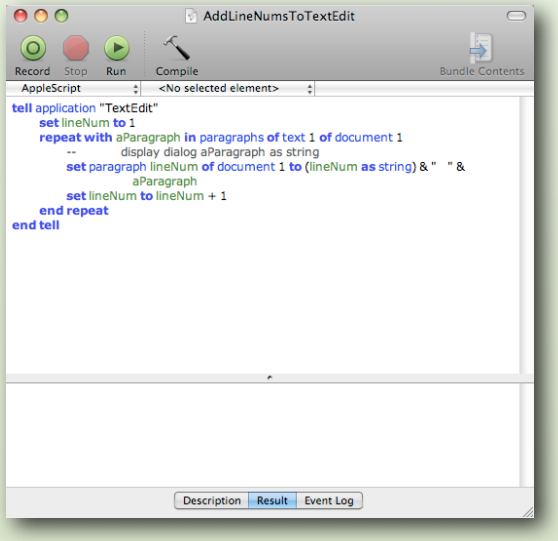
27

Typical Mac OS X IDEs

AppleScript's Script Editor



- Built-in:
  - Text editor
  - Documentation
- Automation Languages:
  - AppleScript
- Interpreted or Compiled



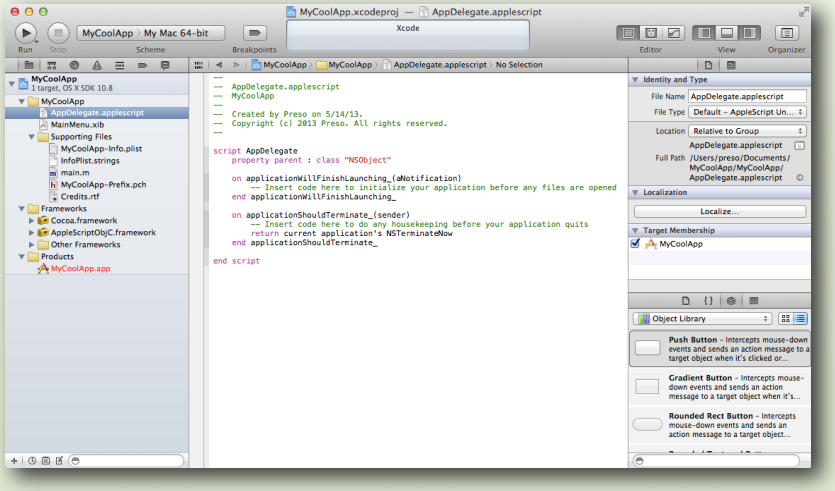
28

Typical Mac OS X IDEs



Xcode

- Built-in:
  - Text editor
  - Source code organizer
  - Object code compiling/linking
  - Documentation
- Automation Languages:
  - AppleScript, Python, Ruby,
  - Objective-C, C, C++, Java
  - Interface Builder (pre-Xcode 4)
  - ...
- Compiled Projects
  - AppleScript
  - Cocoa
  - Carbon
  - Frameworks/Libraries, Kernel Extensions, Plug-ins
  - ...



29

Typical Mac OS X IDEs

Safari/FireFox/Chrome

- Built-in:
  - Debugger
- NOT built-in
  - Text editor
  - Source code organizer
  - Documentation
- Automation Languages:
  - JavaScript
  - HTML
  - CSS (Cascading Style Sheets)
- Interpreted

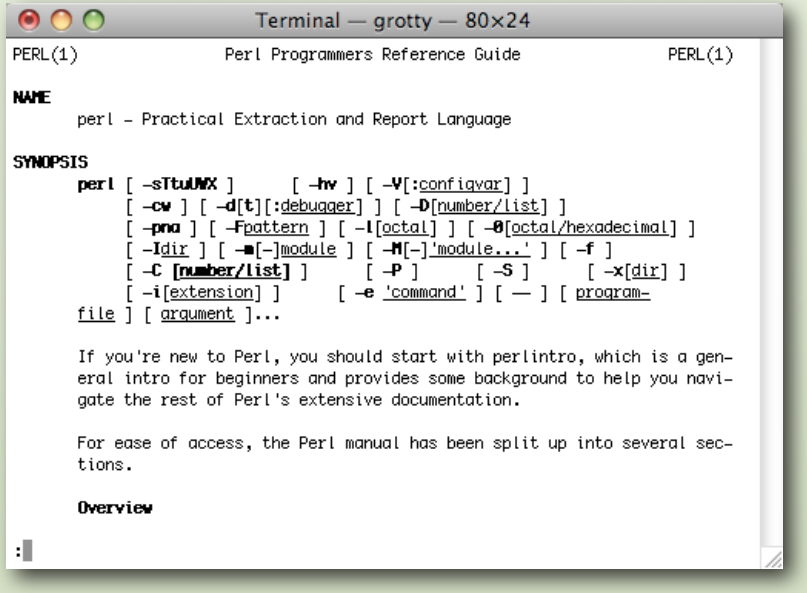


30

Typical Mac OS X IDEs

UNIX CLI

- Built-in:
  - Text editor (`vi`, `nano`/`pico`)
  - Documentation (`man` pages)
- NOT built-in
  - Source code organizer
  - Object code/linking organizer
- Automation Languages:
  - Perl
  - Shells: `bash`, `csh`, `ksh`, etc.
  - Objective-C, C++, Java
  - Python, Ruby
  - ...
- Interpreted or Compiled



A Goal, and a Plan

In order to want to automate a workflow, you probably already have a **Goal** in mind. Examples:

- Create an email by selecting an email address using Services
- Name your screenshots as they are created using Automator Folder Actions
- Create your own Service using Automator
- (you tell me!)

The Goal you have in automating will drive a **Plan** to perform the automation

- Let's make a list of three goals you want to accomplish



MACTECH

32

Creating a Plan

Once we have a Goal, how do we go about creating an Automation-friendly plan?

"Write" out your plan before getting ANYWHERE NEAR an IDE

- You can use a piece of paper, a napkin, or best of all, a computer itself
- Use an automating mindset while creating your plan

Do NOT think too deeply about how you're going to automate it

- Create your plan first, then tune it into the most appropriate specific IDE



PseudoCode

A useful (and popular) way to “write down” a plan is with **pseudocode**

Pseudocode is Automation language and IDE-agnostic

- “perfect” syntax
- analogous to human language Esperanto

There is no standardized definition of pseudocode

We will develop a pseudocode language for us to use

- start off very informally (high-level)
- migrate to a formal pseudocode

“Focus on brainstorming, not syntax”



High-level PseudoCode: Breakfast

Goal: to eat breakfast

Here is some high-level pseudo code showing the steps involved for breakfast (the most important meal of the day!):

```
Am I hungry enough for breakfast?
     NO:    forget about breakfast, this plan is done
     YES:   Do I have time for breakfast?
          NO:   forget about breakfast, this plan is done
          YES:  Do I want to eat out?
               YES: eat out
               NO:  make breakfast
```

The entry point into a plan is often called the **Main** part of the plan

- Top-level starting point for your plan



MACTECH

35

Planning

We now have a high-level description of the main part of our plan

- we have not addressed the actual meal yet, or whether we are making it or if we are eating out

PseudoCode is intuitive to some people, but others like graphics better



MACTECH

36

Flowcharts

Flowcharts are a formal way of graphically showing the flow of our plan

- Remember, this is a plan, and plans not only have steps to execute, but also an order and flow in which to execute them and of the data between steps

Flowcharts use specific symbols to delineate what is happening at that stage in the flow of execution

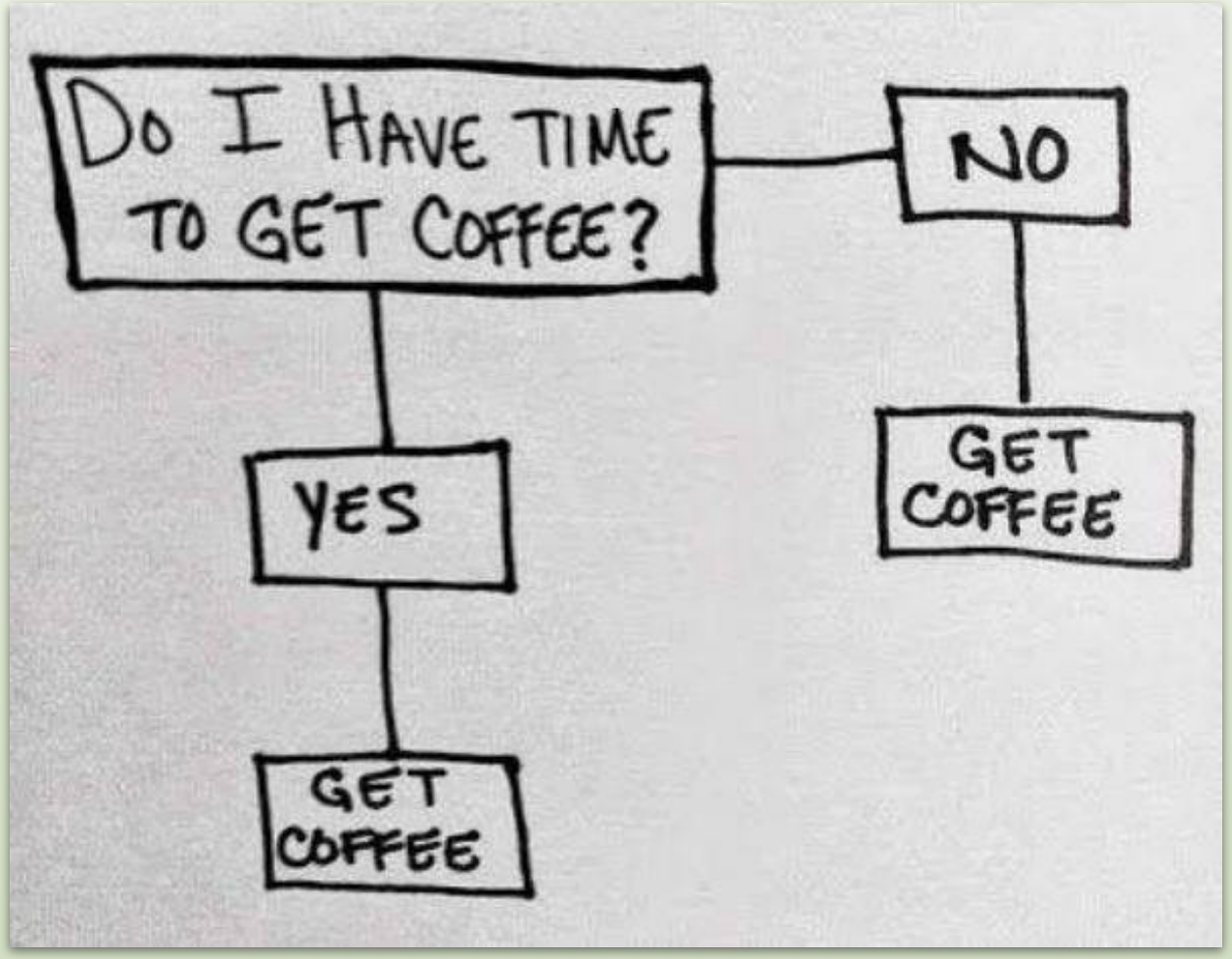
- Directional arrows specify Automation flow
- Symbols represent what happens at steps in the flow

Many tools exist to create Flowcharts

- OmniGraffle
- See Wikipedia page on Flowchart for other tools

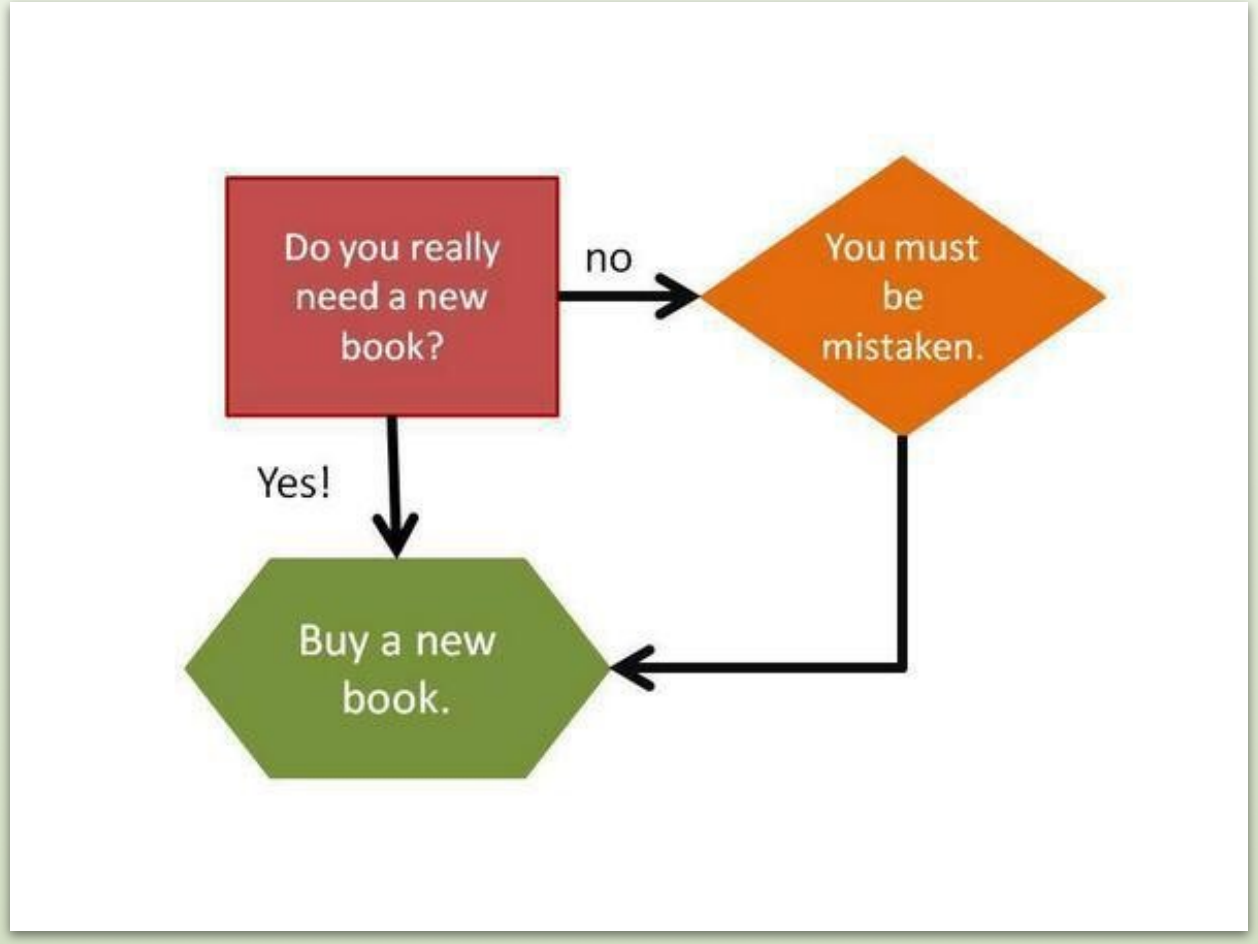


Flowchart



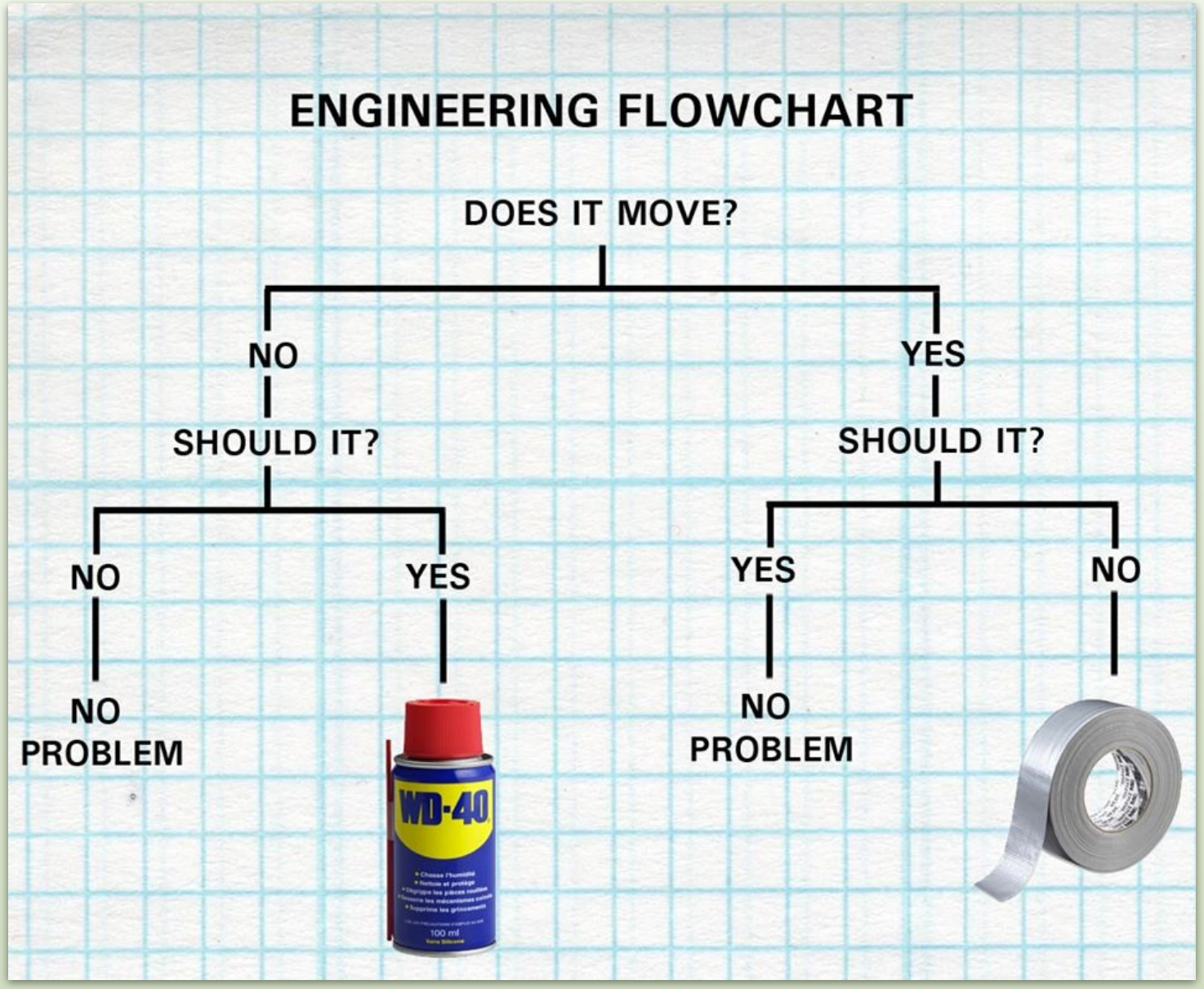
38

Flowchart



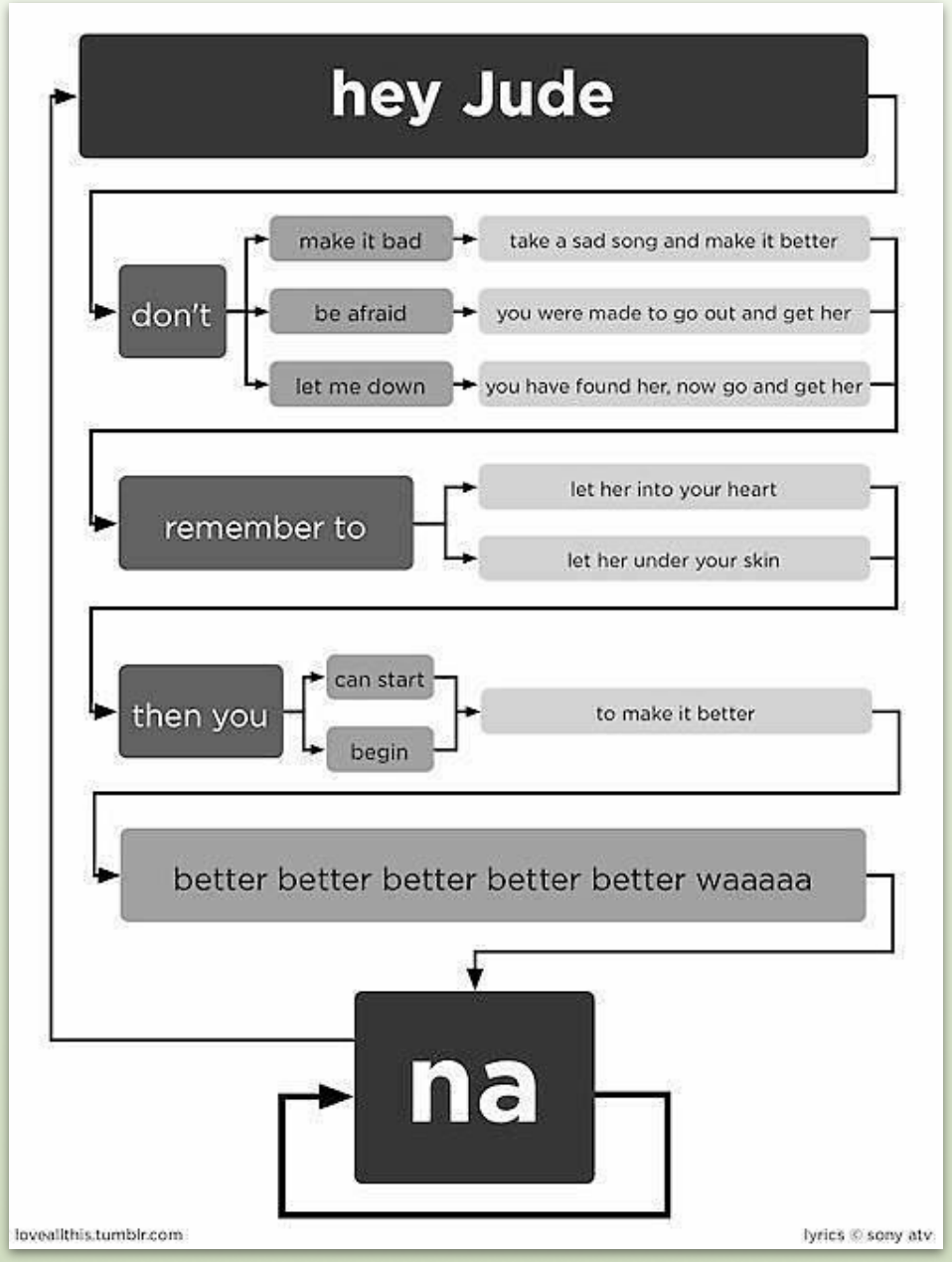
39

Flowchart



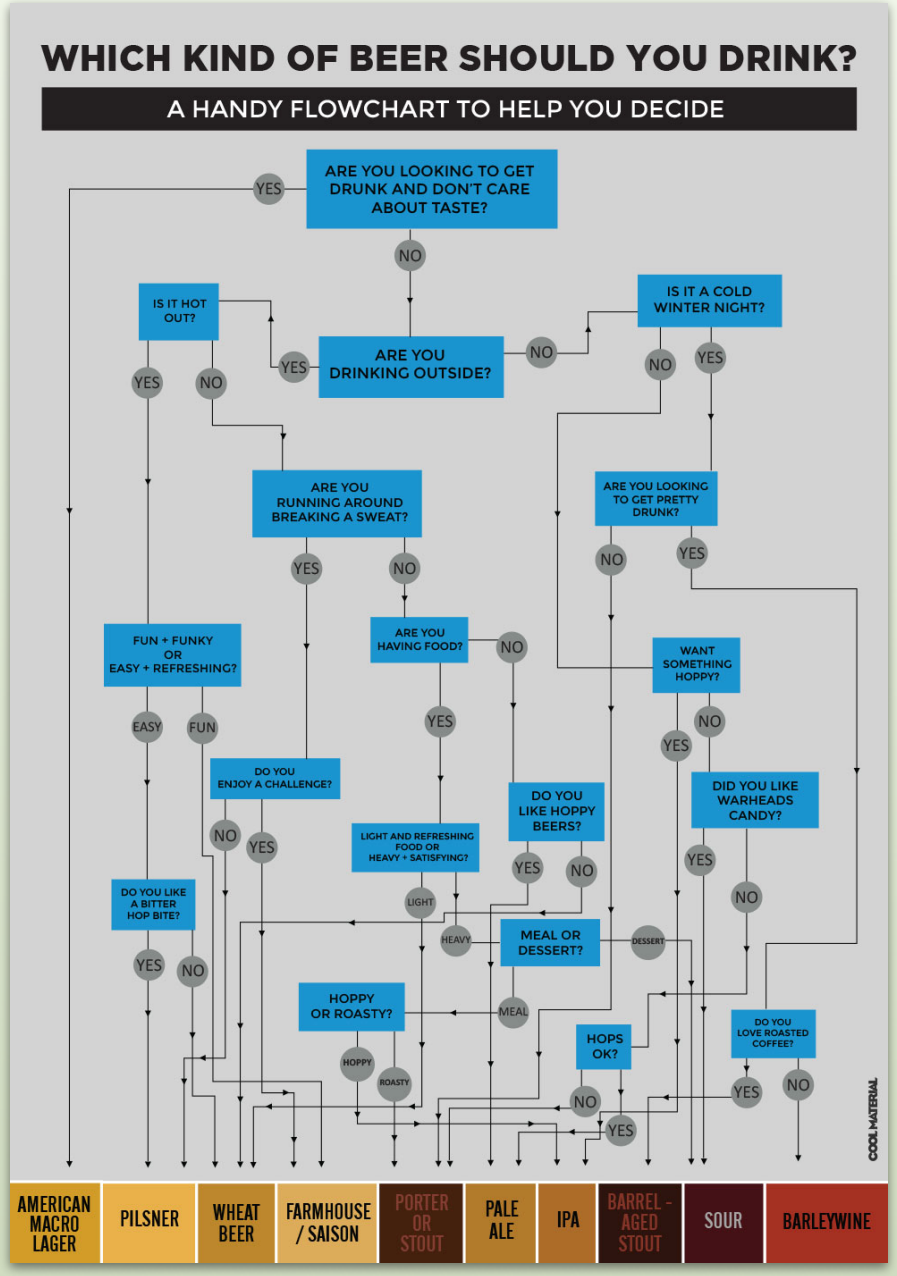
40

Flowchart



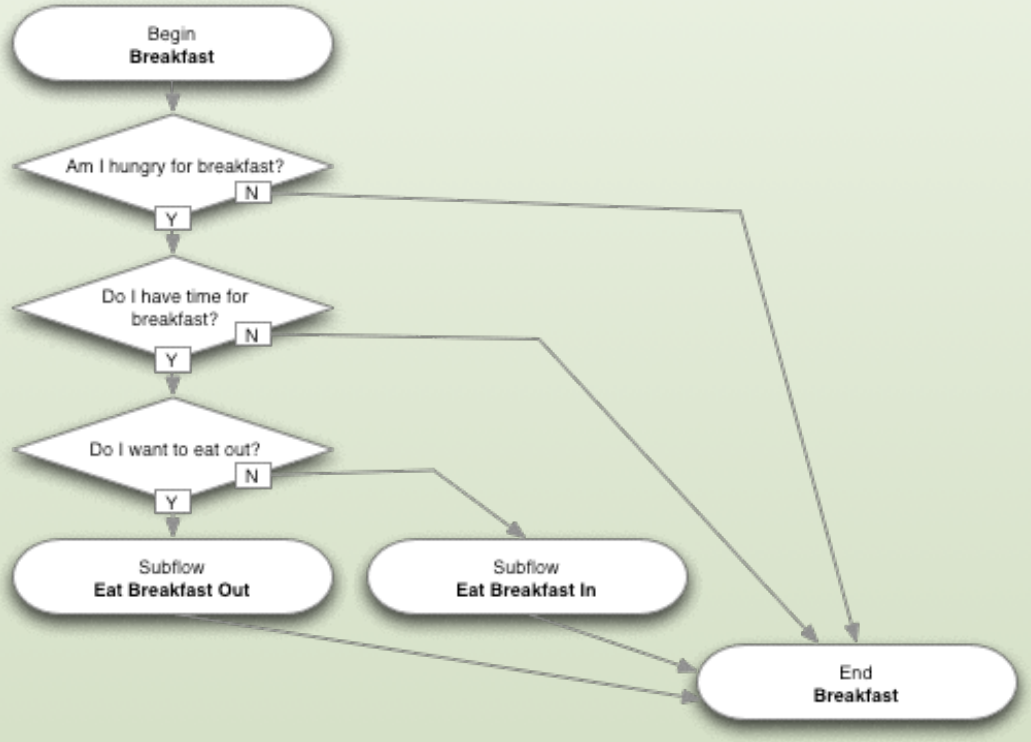
41

Flowchart



42

Flowchart Example



47

Flowcharts & Pseudocode

Flowcharts and pseudocode help make your plan Automation-friendly

- Flow is predictable, the steps are clear, decisions are not ambiguous

Flowcharts are good for conceptualizing the flow at a high-level before heading straight to an IDE for implementation

PseudoCode should be written for everything but the most trivial Automations

The clearer the goal, and the better the plan, the easier it is to implement an Automation in an IDE!



‘‘Be Inspired’’: Pro Edition

In my classes, I let the students create their own automations and I mentor them.

We don’t have time in this short session, so I’m going to “be inspired” by

www.macosxautomation.com

(and some other things)

Goal 1: Rename Screenshot

"Inspired by"

`http://apple.blogoverflow.com/2012/06/folder-actions-tutorial-automation-meet-the-filesystem/`

Takes advantage of OS X

- Command-Shift-3: Take a screenshot of the screen, and save it as a file on the desktop

Utilizes Automator Folder Actions

For those that wish to follow along:

```
$ defaults write com.apple.screencapture location \ ↵
/tmp/screenshots ↵
$ killall SystemUIServer ↵
```



Goal 1: Rename Screenshot

Plan:

1. Save a reference to the new file so we can access it later
2. Ask the user to type a name for the new screenshot
3. Save that name in a variable so we can access it later
4. Rename the added file (we'll retrieve the saved reference to it) to the name the user entered (which we'll also retrieve)
5. Move the renamed file to the Desktop



MACTECH

52

Goal 2: Services

"Inspired by"

`http://www.macosxautomation.com/services/`

specifically

`http://www.macosxautomation.com/services/learn/tut01/index.html`

We will

- Launch an application using Services
- Assign a keyboard shortcut to it



MACTECH

53

Goal 2:
Application Launch Service

Plan:

1. Create an Automator Service workflow
2. Keep It Simple: have one action, and not accept input
3. Launch a specific application in that one action
4. Assign a keyboard shortcut within System Preferences



MACTECH

54

Goal 3:
Download & Run an AppleScript

"Inspired by":
You choose!
For those that wish to follow along:

1. Open ScriptEditor
   In Applications->Utilities
2. Go to menu
   File->Open Dictionary
3. Explore macosxautomation.com



55

Goal 4:
Web Automation

"Inspired by":
Zapier
For those that wish to follow along:

1. Watch the Zapier video at

`http://`

2. (that's all we have time for…)



MACTECH

56

The Process/Methodology Synopsis

1. Have a goal
2. Make a plan
   - Document what you would do if you served your goal manually
     - pseudo code, OmniGraffle, napkin, …
   - Make sure The Plan contains
     - events and decisions
3. "Be Inspired"
   1. See if it's been done already, and if so, if you have permission to use that code
   2. If not, know how to look in documentation to see if you have the tools you need



The Process/Methodology Synopsis

1. Adjust your plan based on the reality of automation environments
2. Get help from others
    - "programmer" forums are MUCH less the newbie-hazing entities they were in the past
    - Give credit where it is due
        - "Be inspired" does NOT mean "plagiarize"

Remember, you're not starting your automation quest by written a long novel

- Think of it more as a newsletter, or blog post, or haiku



Automation
Getting the Mindset for IT



acmeFoo

Scott M. Neal
smn.mg@acmefoo.org
MacTech SF 16 March 2016
Copyright 2016 MindsetGarden