

Ben Bass

Ben has been playing with Macs since the mid 1990's, and was thrilled to discover that people would actually pay him for his hobby. He has worked as a consultant and in-house tech in New York City, Connecticut, and the Washington DC area. He is currently getting paid for his hobby by the Johns Hopkins University Applied Physics Laboratory, where he works on making sure the overall Mac experience is as smooth as possible. Ben works on lab-wide projects to make sure the Mac users are as well taken care of as possible.



Automation

Getting the Mindset for IT

Automation

Getting the Mindset for IT

28 September 2009

Copyright 2005-2009 Senseption, LLC

acmeFoo



Scott M. Neal

smn.mg@acmefoo.org

MacTech Seattle 6 April 2016

Copyright 2016 MindsetGarden

What are we doing today?

- Answer the questions:
 - What should I automate?
 - Why should I automate?
 - How do I automate?
- Present tools for building your automations
- Provide resources for inspiration

The Automating Mindset

Automating is a mindset

- Some have it naturally
- It is learnable

Many who automate are not primarily programmers, but might be a

- Linguist
- Musician
- English Major
- Accountant

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 without memorizing EVERYTHING about cars or music

- Learn the underlying concepts
- Find out how/where to find the needed specifics

Your Takeaways

Learning how to automate is a process

- You will learn the basics to continue learning
 - including knowing what questions, how to ask those questions, and where to ask the questions

Empower yourself to automate

- It is within your reach
- You do NOT need to have everything memorized
 - Learn from others
 - RTFM
 - Build your own automation library
- Automate all the things

Automation What?

Automate all the things:

- Any task you do regularly
- A task you need to perform multiple times
- A task you need done consistently by others

Automation Why?

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

Automation brings consistency to complex tasks

Automation How? Goals and Plans

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 to move emails to a certain folder with a keystroke

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

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

Focus on brainstorming, not syntax

High-level Pseudocode: Breakfast

Goal: to eat breakfast

Here is some high-level pseudocode 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

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

Flowcharts

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

- This is a plan, and plans not only have steps to execute, but also an order and flow in which to execute them
- AKA, order matters

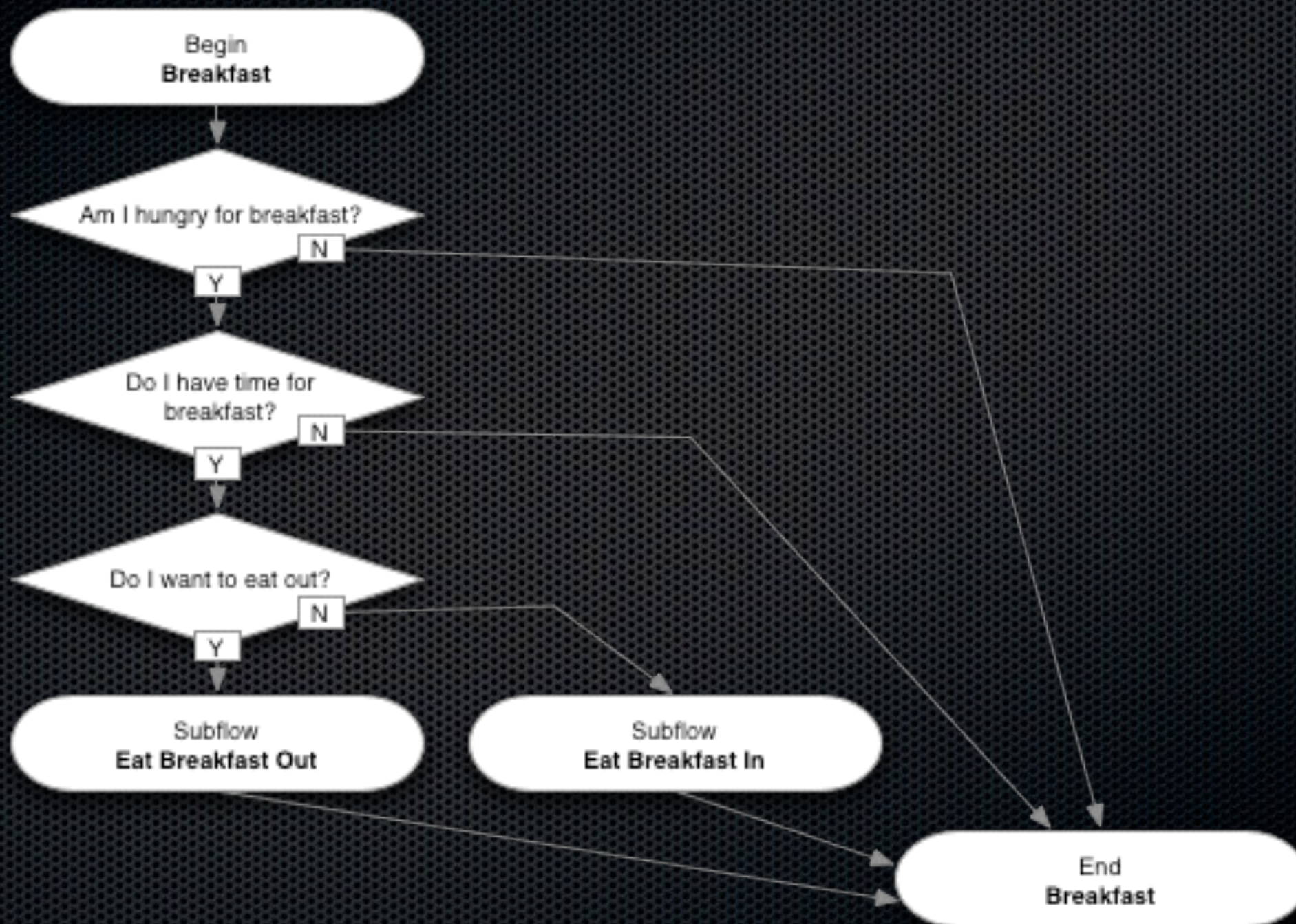
Flowcharts use specific symbols to delineate what is happening at that stage in the flow of execution

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

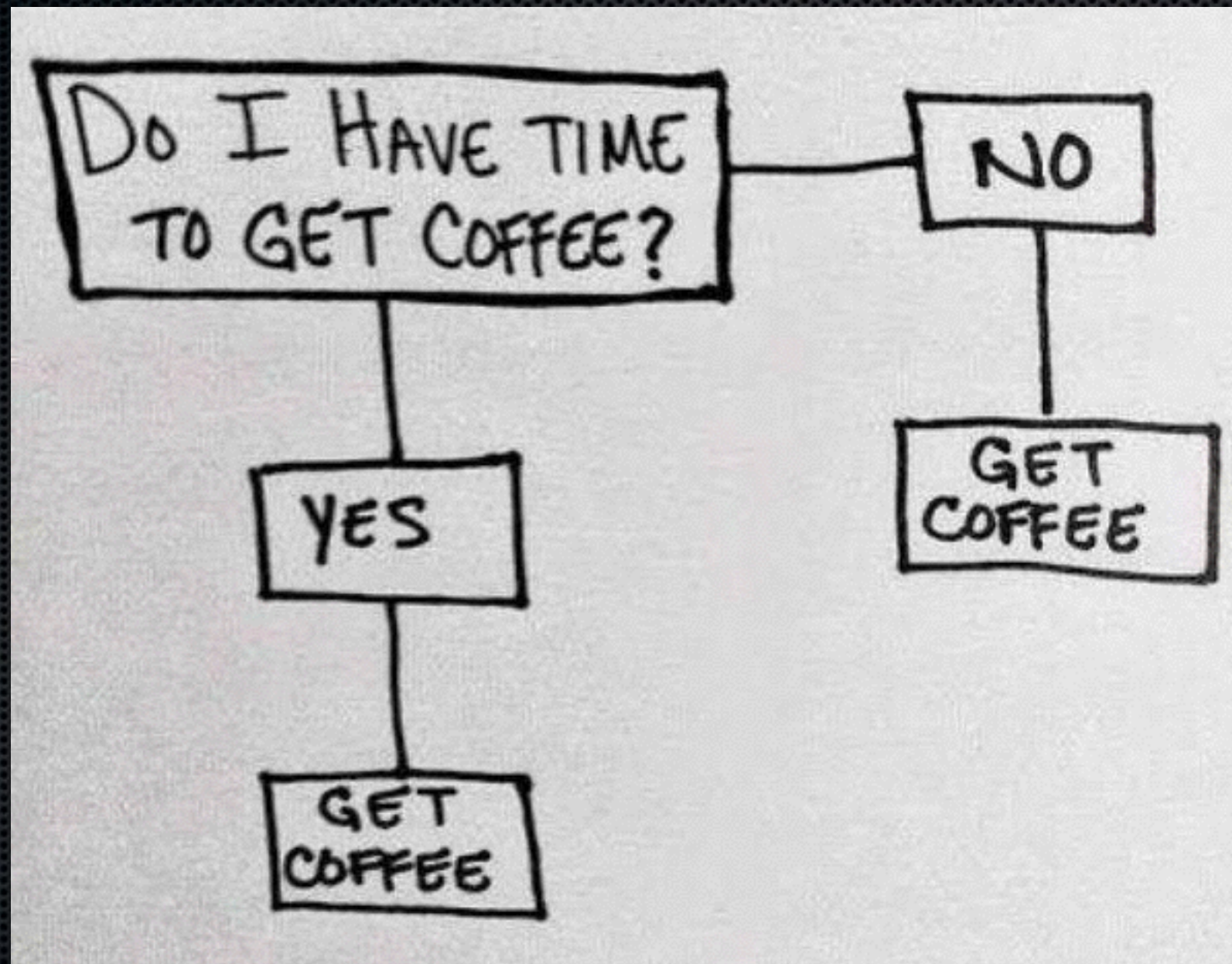
Many tools exist to create Flowcharts

- OmniGraffle
- See Wikipedia page on Flowchart for other tools

Flowchart Example



Flowchart



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!

Process like a Computer

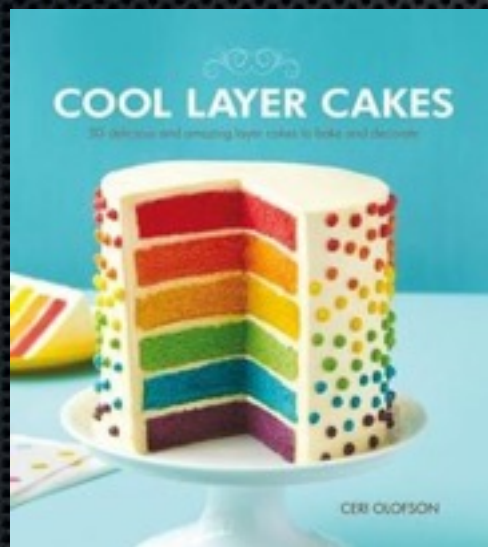
Computers don't actually think they **process**

- 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
- 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

Process 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



Events

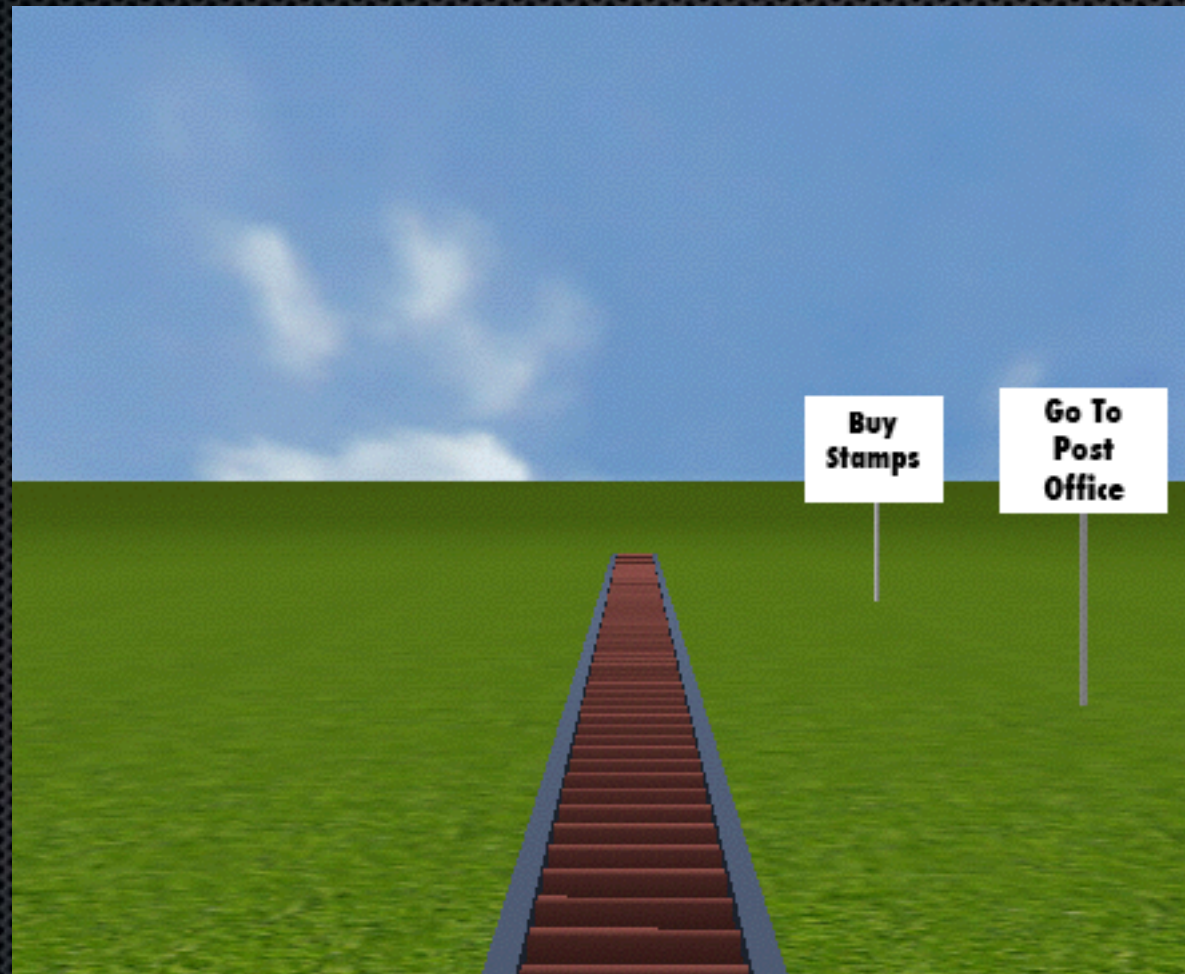
These come at you whether you want them to or not

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

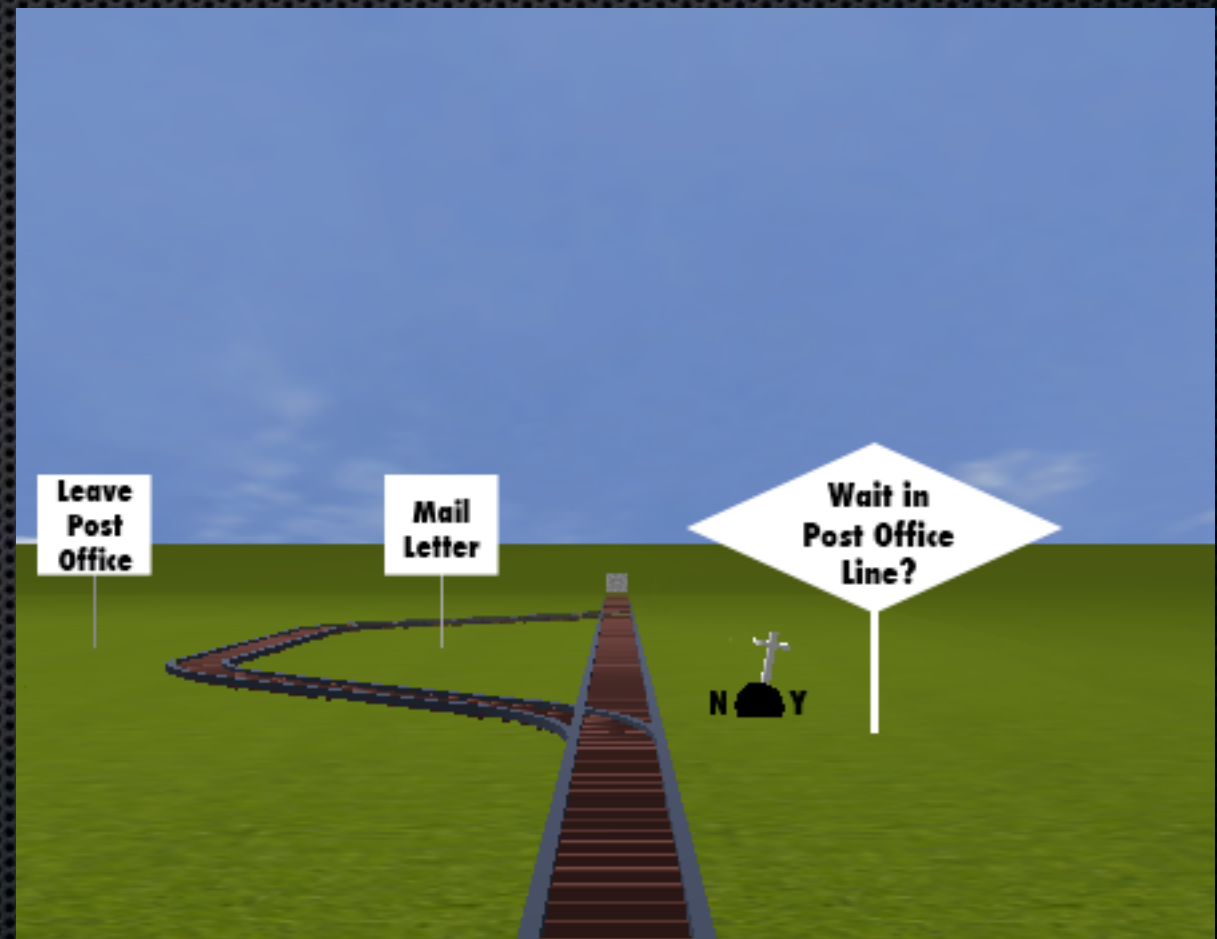
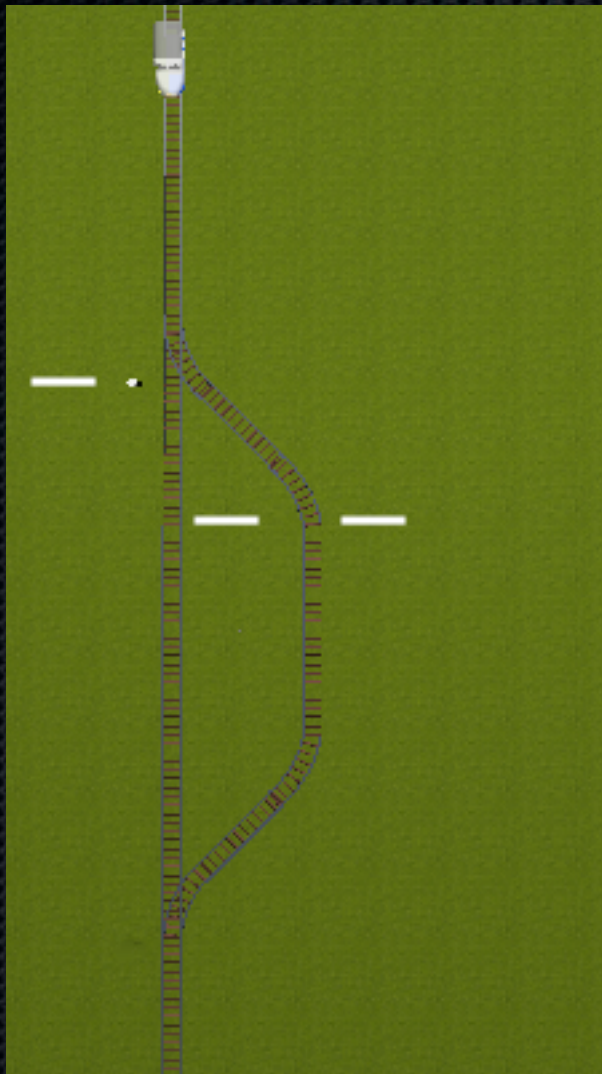
Many “crashes” in software are from incomplete handling of events



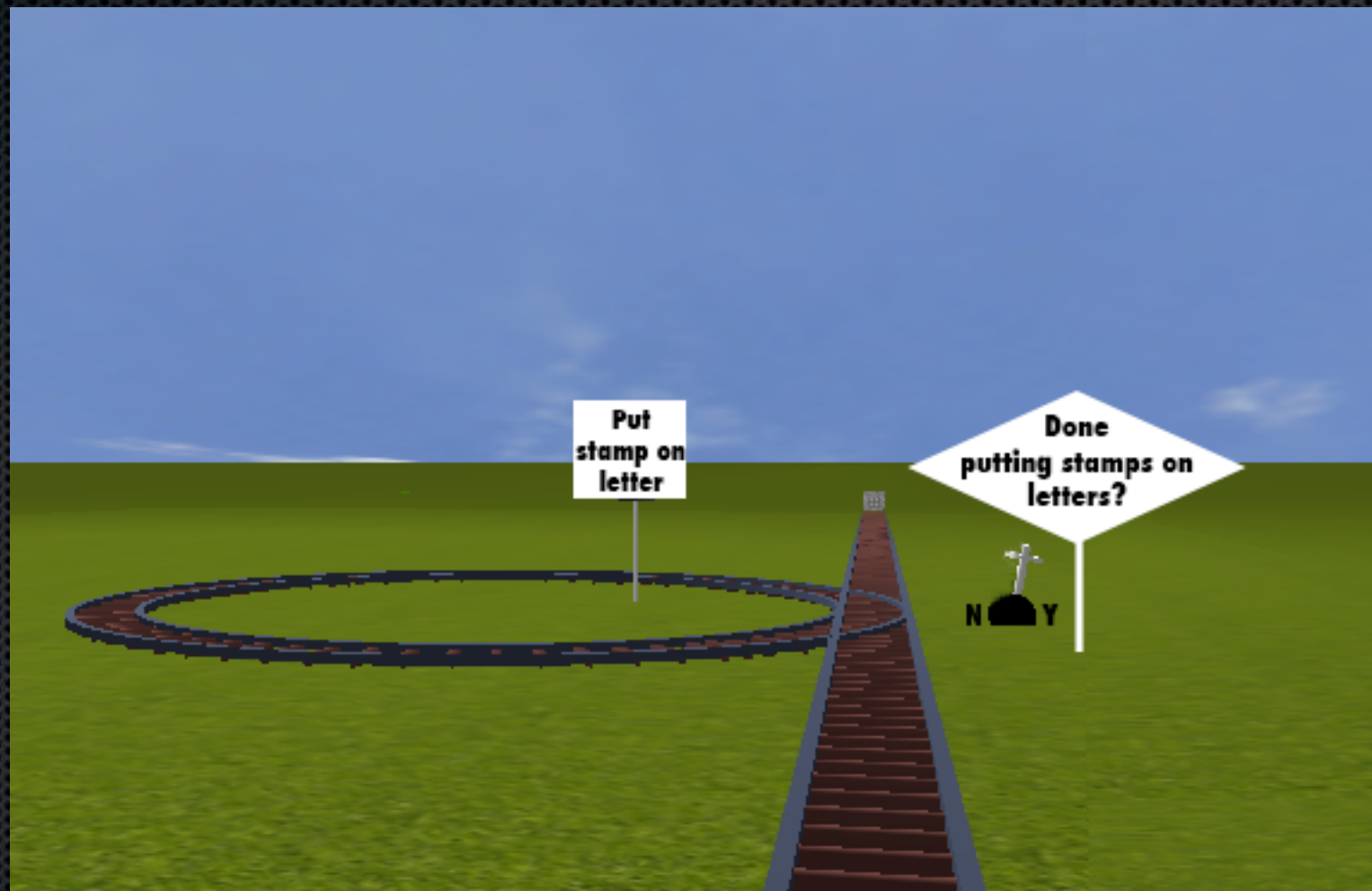
Automating is “railed”



Decisions, still “railed”



Repetition with Loops



IDE: Workflows, Scripts, Programs

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**

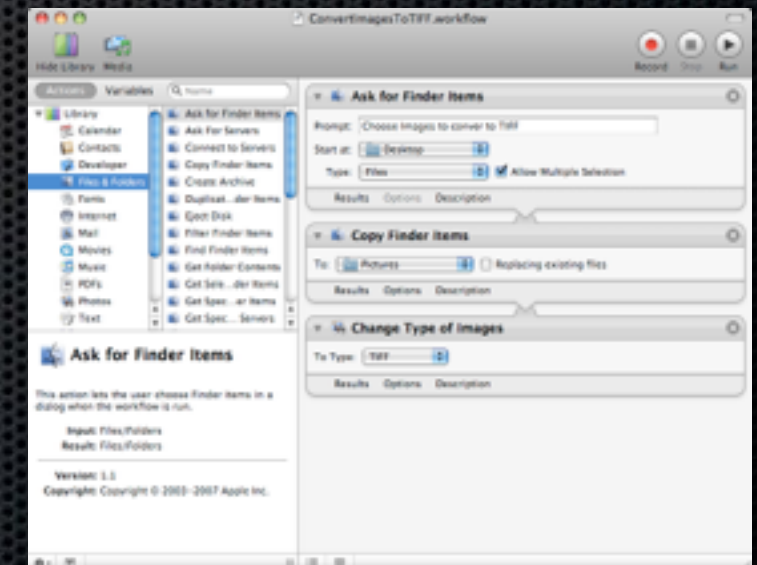
We will develop Automations using Automating-specific tools

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

Typical Mac OS X IDEs

Automator

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



Typical Mac OS X IDEs

AppleScript's Script Editor

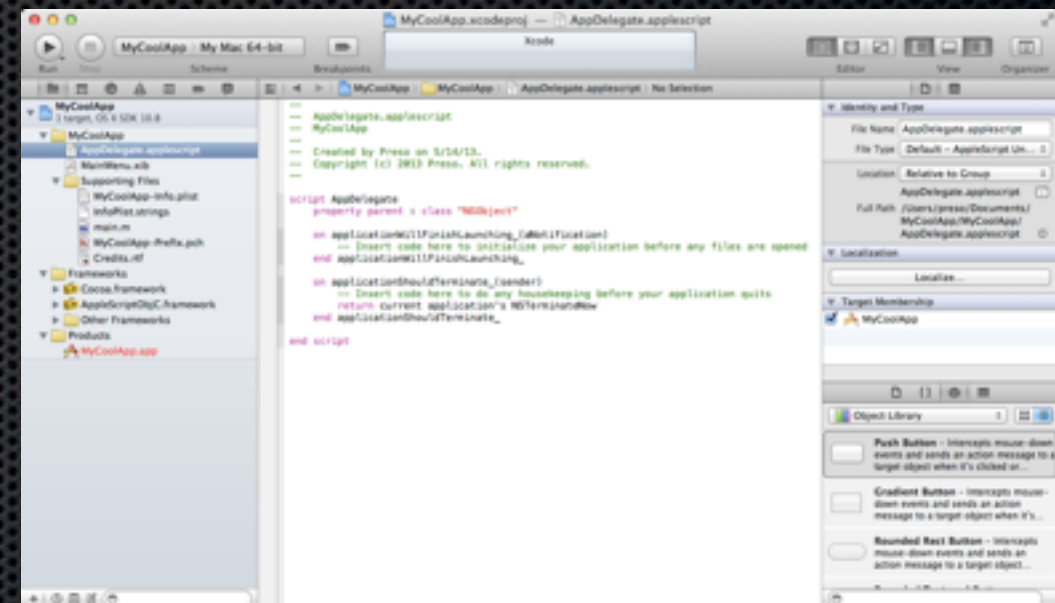
- Built-in:
 - Text editor
 - Documentation
- Automation Languages:
 - AppleScript
- Interpreted or Compiled



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
 - ...



Typical Mac OS X IDEs

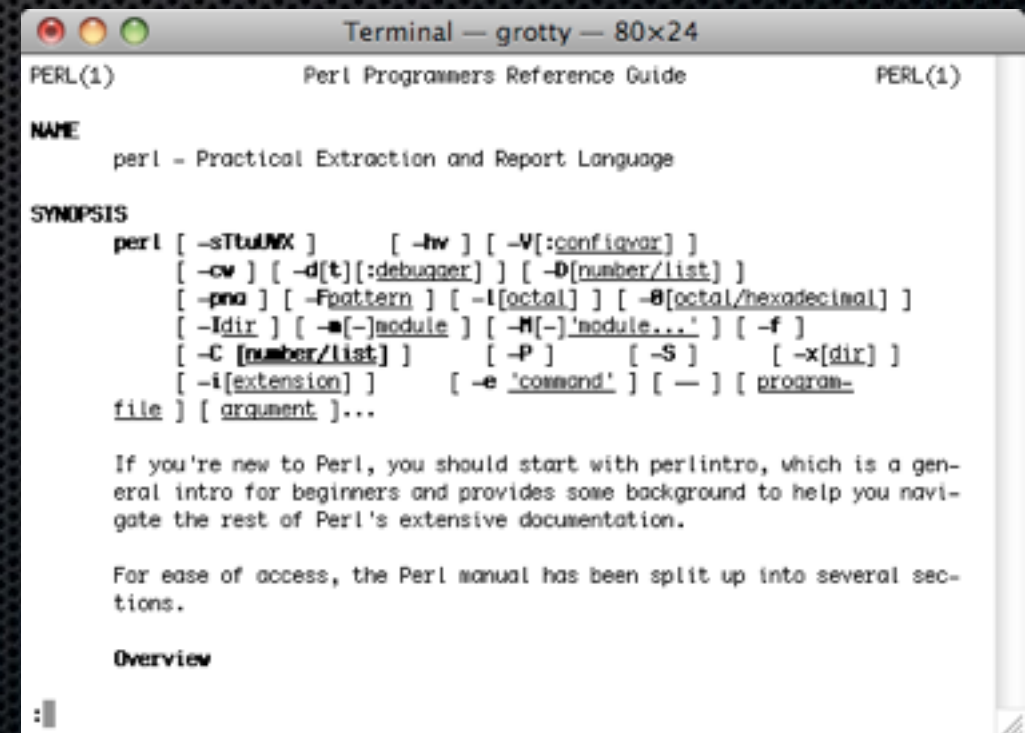
Safari/FireFox/Chrome

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

Typical Mac OS X IDEs

UNIX CLI

- Built-in:
 - Text editor (vi, nano/pico, emacs)
 - 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



The screenshot shows a Terminal window titled "Terminal — grotty — 80x24". The window displays the Perl man page, which includes the title "PERL(1)", the subtitle "Perl Programmers Reference Guide", and the name "perl - Practical Extraction and Report Language". The synopsis section lists various command-line options for the perl interpreter, such as -s, -t, -d, -D, -p, -pna, -I, -M, -C, -P, -S, -x, -i, -e, and --. The text also mentions that new users should start with perlintro and that the Perl manual is split into several sections. The word "Overview" is visible at the bottom of the page.

The Process/Methodology Synopsis

1. Have a goal
2. Make a plan
 - Document what you would do if you served your goal manually
 - pseudocode, OmniGraffle, napkin, ...
 - Make sure The Plan contains
 - events and decisions
3. Be Inspired
 - See if it's been done already, and if so, if you have permission to use that code
 - 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 the automation environment
2. Get help from others
 - MacEnterprise Mailing list, <http://www.macenterprise.org/>
 - MacAdmins Slack Channel, <https://macadmins.slack.com/>
 - Online Forums
 - Stack Exchange, <http://stackexchange.com/>
 - Stack Overflow, <http://stackoverflow.com/>
 - Give credit where it is due
 - “Be inspired” does NOT mean “plagiarize”
 - Ask about your Goal, not how to perform an individual task
 - Don't lose the forest for the trees

Questions?



Ben Bass

ben@benbass.com

Some Inspiration

Not sure where to start?

Here are some links and demos you can do on your own.

A great place to start is
www.macosxautomation.com

Demo I:

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 ↵
```


Demo I:

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

Demo 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

Demo 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

Demo 3:

Web Automation

Zapier

- Watch the Zapier video at https://m.youtube.com/watch?v=nk_zw9paux8

If This Then That (IFTTT)

- Watch the IFTTT video at <https://www.youtube.com/watch?v=FWL8QjcOc9g>