Cocoa For Hackers
or
Don’t Try This At Home

Chris Kane & Mike Ferris

Introduction

- Cocoa is a great environment for hacking
  - An open and extremely dynamic language (Objective C)
  - Frameworks that have been designed from the start to provide powerful abstractions that are customizable and extensible
  - Emphasis on “primitive” methods

What You'll Learn

- What mechanisms exist for customizing and extending built-in behaviors
- How to get your code loaded into all Cocoa applications
- How to do truly scary things with the Objective-C runtime environment

What You'll Promise

- I will not use private API or make undocumented assumptions in applications that I expect to work tomorrow
- I will not use gross hacks to do things that can be done in supported ways
- I will not destabilize Mac OS X by distributing buggy code in Input Manager bundles

Topics

- Interesting methods to override
- Notifications
- Categories and Posing
- Input Manager bundles
- Objective-C Runtime
- Being as safe as possible

Interesting Overrides

- There are lots of methods in the Cocoa frameworks
- For hacking purposes the interesting ones for overriding are sometimes methods that you would not usually need to override in a normal application
- Often these end up being general funnel methods that are involved in large general areas of functionality

Event Handling

- -[NSApplication sendEvent:]
- -[NSWindow sendEvent:]
- -[NSApplication nextEventMatchingMask: untilDate: inMode: dequeue:]
- -[NSApplication updateWindows]
- -[NSWindow update]

Target/Action & Field Editors

- -[NSApplication sendAction: to: from:]
- -[NSApplication targetForAction: to: from:]
- -[NSWindow fieldEditor: forObject:]

General View Stuff

- -[NSView hitTest:]
- -[NSView becomeFirstResponder]
- -[NSView resignFirstResponder]
- -[NSView viewWillMoveToWindow:]
- -[NSView viewWillMoveToSuperview:]

Text System

- -[NSTextView insertText:]
- -[NSTextView doCommandBySelector:]
- -[NSTextView setSelectedRange: affinity: stillSelecting:]
- -[NSTextView selectionRangeForProposedRange: granularity:]

More Text System

- -[NSTextView shouldChangeTextInRange: replacementString:]
- -[NSTextView didChangeText]
- -[NSTextView rangeForUserTextChange]
- -[NSTextView rangeForUserCharacterAttributeChange]
- -[NSTextView rangeForUserParagraphAttributeChange]

Notifications

- Notifications are broadcast messages that can be received by many independent observers
- Cocoa classes define many notifications that are sent when important things happen or are about to happen
- Because observers are independent, a new observer can be added at any time and will not affect the proper functioning of other existing observer

Application Notifications

- NSApplicationDidFinishLaunchingNotification
- NSApplicationDidBecomeActiveNotification
- NSApplicationDidResignActiveNotification
- NSApplicationDidUpdateNotification
- NSApplicationWillTerminateNotification
- NSApplicationDidChangeScreenParametersNotification

Window Notifications

- NSWindowDidBecomeKeyNotification
- NSWindowDidResignKeyNotification
- NSWindowDidMoveNotification
- NSWindowDidResizeNotification
- NSWindowDidUpdateNotification
- NSWindowWillCloseNotification

View Notifications

- NSViewFrameDidChangeNotification
- NSViewBoundsDidChangeNotification
- NSViewFocusDidChangeNotification

Text Notifications

- NSTextDidBeginEditingNotification
- NSTextDidEndEditingNotification
- NSTextDidChangeNotification
  - Editable NSControls have variants of these too
- NSTextViewDidChangeSelectionNotification
- NSTextStorageWillProcessEditingNotification
- NSTextStorageDidProcessEditingNotification

Other AppKit Notifications

- Menus: items added, removed, actions sent
- PopUps: menu will pop
- Drawers: opening, closing
- Split Views: resizing subviews
- Tables/Outlines: selection changes, column manipulation, expanding/collapsing

Foundation Notifications

- NSBundleDidLoadNotification
- NSUserDefaultsDidChangeNotification
- NSWillBecomeMultiThreadedNotification

Categories

- Objective-C categories add methods to an existing class
- Categories can be used to
  - Extend a class, adding new capabilities
  - Help maintain a layered design while still allowing methods to be on the natrual type of receiver
  - Patch a buggy method in a class you do not have the code for

Extending Functionality with Categories

- Does some class you're using need a new method to make your job easier?
- You can add it
  - -[NSString stringByEscapingHTML]
  - -[NSMutableArray removeAllOccurrencesOfObject:]
  - -[NSObject valueForKey:], -[NSObject takeValue:forKey:]

Adding NSTextView Key Binding Methods

- NSTextView has a key binding mechanism that maps key sequences to method invocations
- You can add new methods to NSTextView with a category and then create key bindings that invoke them
- Key-binding methods are like actions: they take a single “sender” argument which is almost always ignored

Layered Design with Categories

- NSString is a Foundation class and Foundation cannot draw
- AppKit defines how to draw
- Applications often want to draw NSStrings
- So AppKit adds drawing methods to NSString
  - -[NSString sizeWithAttributes:]
  - -[NSString drawInRect:withAttributes:]

Patching Existing Methods with Categories

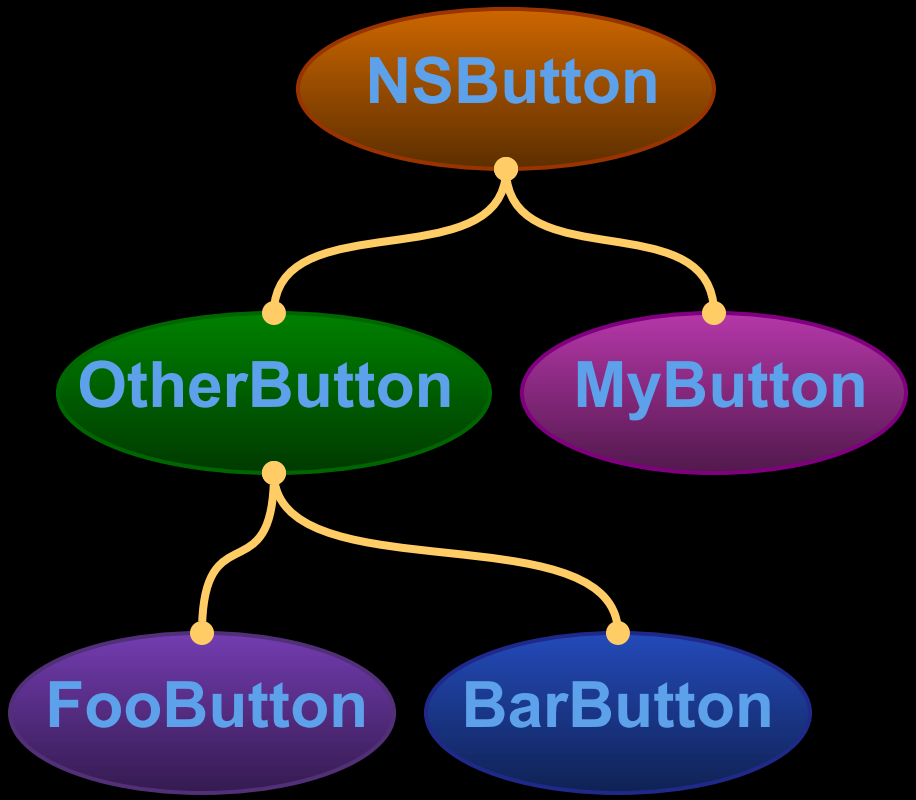
- Replacing methods can be dangerous
- Only guarantee is that category method overrides base class method
- If multiple categories define same method, behavior is undefined
- Cannot (easily) call the implementation you are replacing
- But, sometimes this can save your butt

Posing

- Posing allows a class to “become” its superclass
  - [MyButton poseAsClass:[NSButton class]]
- Any use of the superclass will actually use the posing class
- Other subclasses of the superclass “become” subclasses of the posing class
- Posing is incredibly dangerous and almost always unnecessary

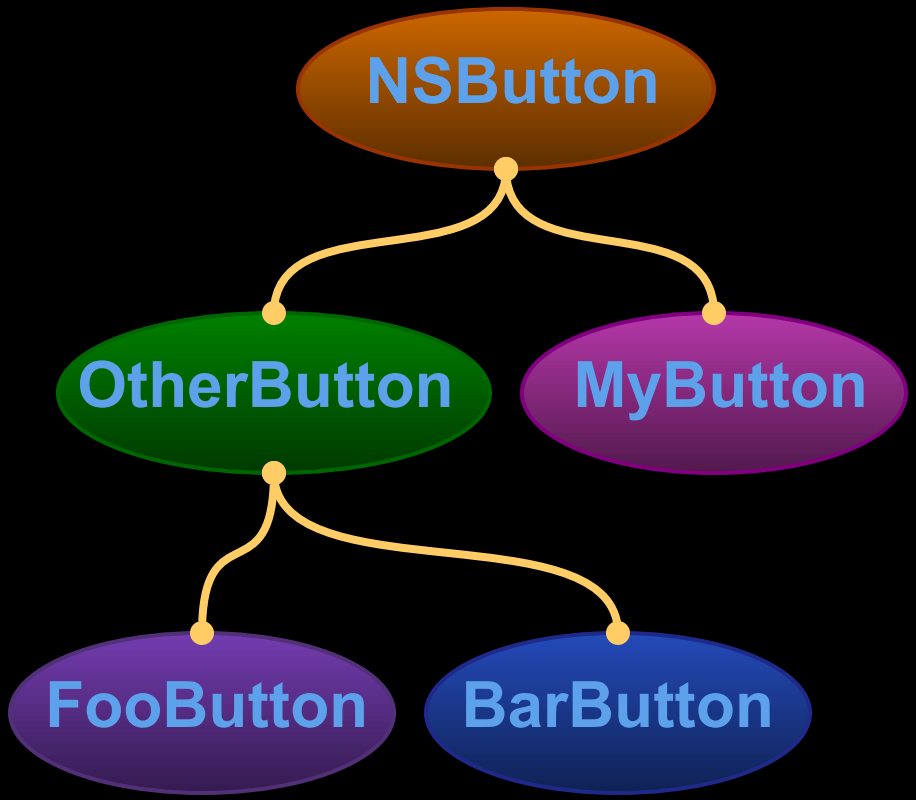
How Posing Works

Before

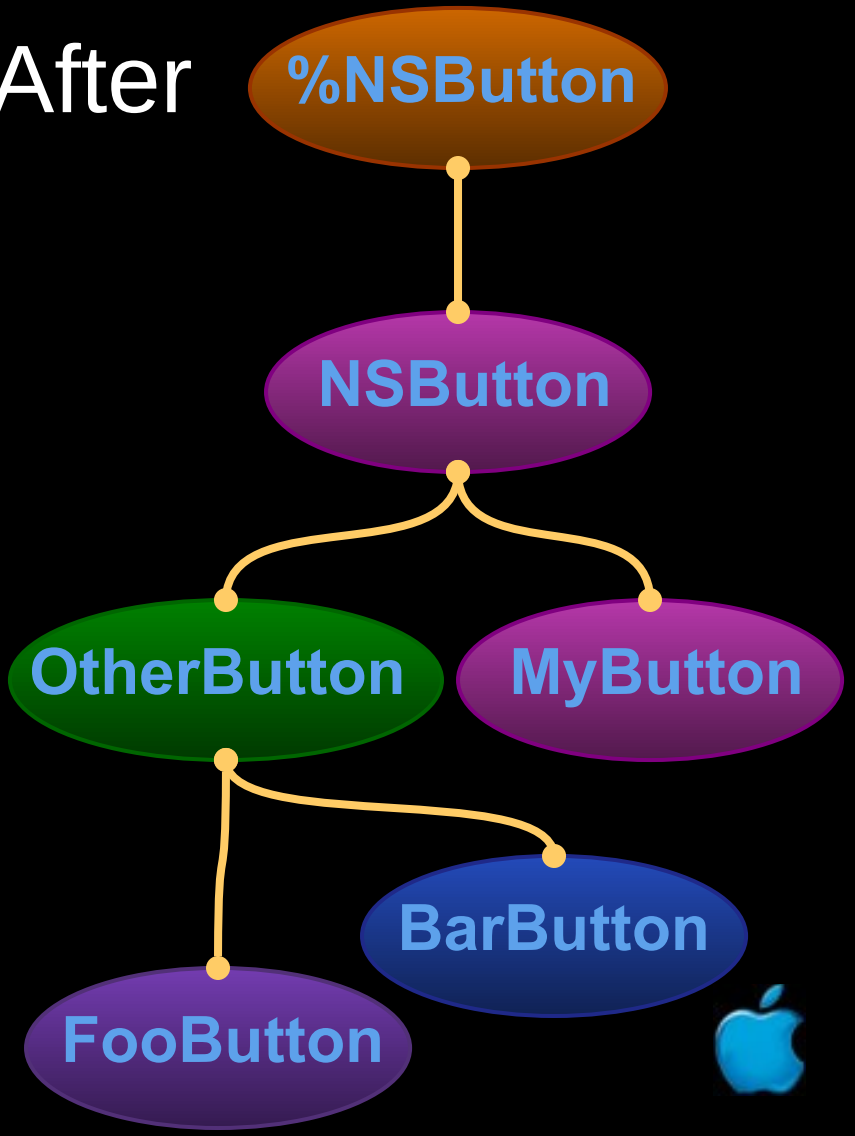


How Posing Works

Before



After



Inflicting Your Hacks on Unsuspecting Applications

- Cocoa applications will load Input Manager bundles when they launch
- Input Managers are intended to provide support for complex text input requirements of languages like Japanese
- But, in fact, they can contain any code you want

Installing Input Managers

- Input Managers are bundles that live in standard locations
  - ~/Library/InputManagers, /Network/Library/InputManagers, ...
- Each Input Manager bundle lives in its own folder in one of these locations and, in addition to the bundle, each of these folders has an “Info” file that describes certain attributes of the bundle.

HackMac Skeletal Input Manager

- If you want to start playing with Input Manager bundles, ask me
- I have a project that builds a skeletal Input Manager called HackMac for your hacking pleasure

DEMO
TextExtras Input Manager

Scary Objective-C Runtime Tricks

- Objective C headers are “public”
- But, still undocumented after 14 years
- To the astute coder, this should send a signal

Manipulating classes

- Getting all the known classes
  - int objc_getClassList(Class *buffer, int bufferLen)
- Having a Class, you can walk up the class hierarchy through the super class pointers
- Adding classes to the runtime
  - Initialize an objc_class structure
  - objc_addClass(Class cls)

Method lists

- Walking the method lists of a class
  - Struct objc_method_list
  - Triplets of selector, encoded method param types, and pointer to method code
- Add/remove methods from classes
  - Initialize an objc_method_list structure
  - Void class_addMethods(Class, objc_method_list *)
  - Void class_removeMethods(Class, objc_method_list *)

Instance variables

- Walking the ivars for an object
  - Get the struct objc_ivar_list from object's class
  - Triplet of name, encoded type, and offset into object
- Get/set ivars directly
  - objc_setInstanceVariable(object, name, &value)
  - objc_getInstanceVariable(object, name, &value)

Allocation strategy

- Can be changed by setting global function pointers
  - id (*_alloc)(Class, size)
  - id (*_copy)(id, size)
  - id (*_realloc)(id, size) [unuseful]
  - id (*_dealloc)(id)

Other goodies

- Registering new selectors
  - sel_registerName(char *)
- Send messages directly
  - objc_msgSend(id, selector, parameters…)
  - objc_msgSend_stret(void *, id, selector, parameters…)
  - objc_msgSendv
    - For “varargs”, but really, not useful
    - Marg_list format is undocumented, and different for each platform and architecture

DEMO
Isa & Class Swizzling

Being Safe

- Although we have discussed some dangerous (but powerful) features, there are things you can do to use them as safely as possible
- This is just minimizing the risk, though, not removing it

Safely Using Private API

- Use -respondsToSelector: to protect against private API disappearing and be prepared to do something intelligent if the method is not there
- Use NSClassFromString() instead of mentioning private class names directly, and if it returns nil, be prepared to do something else

Safe Categories

- Prefix your method names if you are adding methods to existing classes, especially common ones
- Prefix your category names
- Do not override methods with a category

Using Undocumented Features

- You cannot create a robust high-quality product by using undocumented features
  - Unknown side effects can make for fragile applications
  - Your app is more likely to break when the system is updated
- Much of what we have talked about today is undocumented

Q&A


Computer
Home
Apps
Docs
Hard Drive
Network
find
People
View
Search:
TIFF
Aqua
Mail
PDF

Hack Different