



# Cocoa Tips and Tricks

Cool Cocoa code

**Corbin Dunn**

Cocoa Software Engineer

# Introduction

- Quick and cool Cocoa tips and tricks
- Learn new things you may not know
- Focused mostly on AppKit classes

# What You'll Learn

- TableView tips
- Nib tips
- Accessibility tips
- Image tips
- Document tips
- PathControls, SplitViews, and CollectionViews
- ...and more

**DON'T PANIC**

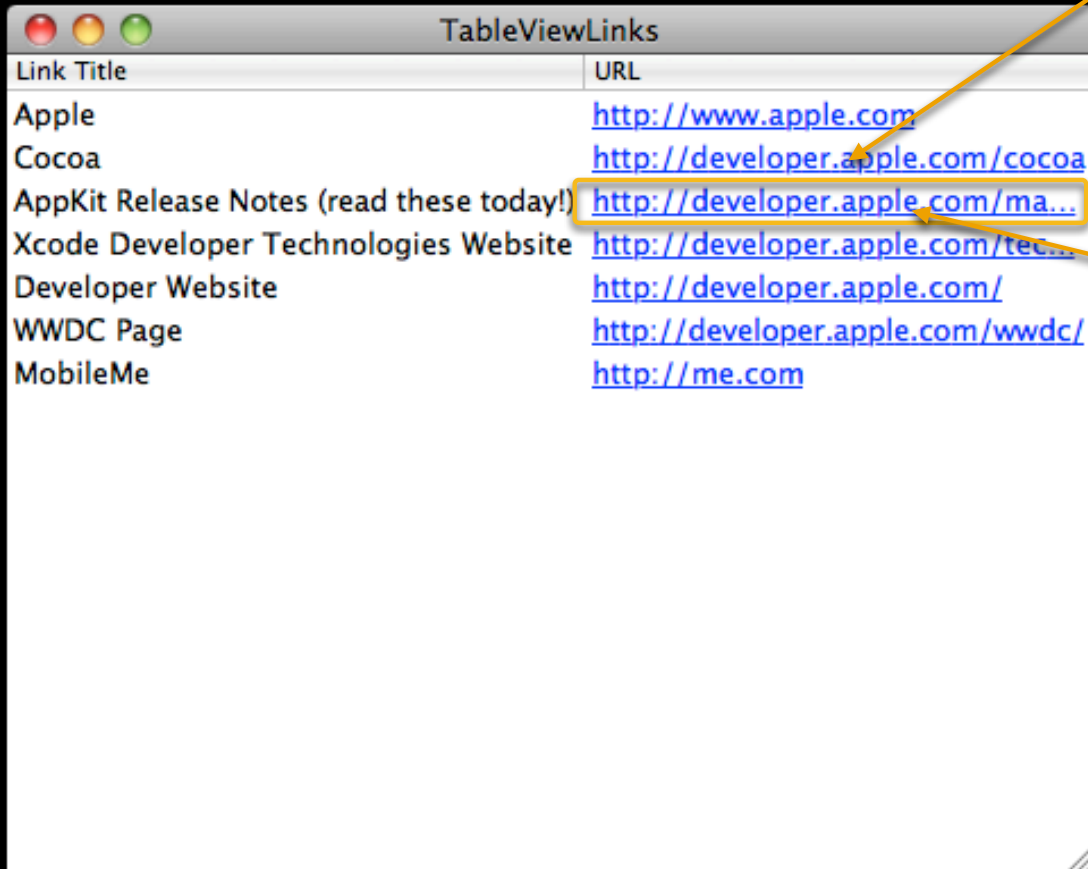
# TableView Tips

**Corbin Dunn**

Cocoa Software Engineer

# Links in a TableView

NSAttributedString



Link Title	URL
Apple	<a href="http://www.apple.com">http://www.apple.com</a>
Cocoa	<a href="http://developer.apple.com/cocoa">http://developer.apple.com/cocoa</a>
AppKit Release Notes (read these today!)	<a href="http://developer.apple.com/ma...">http://developer.apple.com/ma...</a>
Xcode Developer Technologies Website	<a href="http://developer.apple.com/tec...">http://developer.apple.com/tec...</a>
Developer Website	<a href="http://developer.apple.com/">http://developer.apple.com/</a>
WWDC Page	<a href="http://developer.apple.com/wwdc/">http://developer.apple.com/wwdc/</a>
MobileMe	<a href="http://me.com">http://me.com</a>

Custom Cell

# Creating a Link

-attributedStringValue for the NSTextFieldCell

- NSAttributedString with the NSLinkAttributeName attribute

```
NSMutableAttributedString *attrString =  
    [[NSMutableAttributedString alloc] initWithString:@"Title"];
```

```
[attrString addAttribute:NSLinkAttributeName value:aURL range:range];
```

```
[attrString addAttribute:NSForegroundColorAttributeName  
    value:[NSColor blueColor] range:range];
```

```
[attrString addAttribute:NSUnderlineStyleAttributeName  
    value:[NSNumber numberWithInt:NSSingleUnderlineStyle]  
    range:range];
```

# Custom Cell

## LinkTextFieldCell in the sample code

```
- (BOOL)trackMouse:(NSEvent *)theEvent inRect:(NSRect)cellFrame  
    ofView:(NSView *)controlView untilMouseUp:(BOOL)flag {
```

```
    BOOL result = [super trackMouse:theEvent inRect:cellFrame  
                  ofView:controlView untilMouseUp:flag];
```

```
    NSAttributedString *attrValue = [self attributedStringValue];
```

```
    NSURL *link = [attrValue attribute:NSLinkAttributeName  
                  atIndex:0 effectiveRange:NULL];
```

```
    if (link != nil) {
```

```
        // We do have a link -- open it!  
        // Okay, but how?
```

```
    }
```

```
    return result;
```

```
}
```



# Using Blocks as a Click Handler

```
@interface LinkTextFieldCell : NSTextFieldCell {  
@private
```

```
void (^_linkClickedHandler)(NSURL *url, id sender);
```

```
}
```

```
@property(copy) void (^linkClickedHandler)(NSURL *url, id sender);
```

```
@end
```

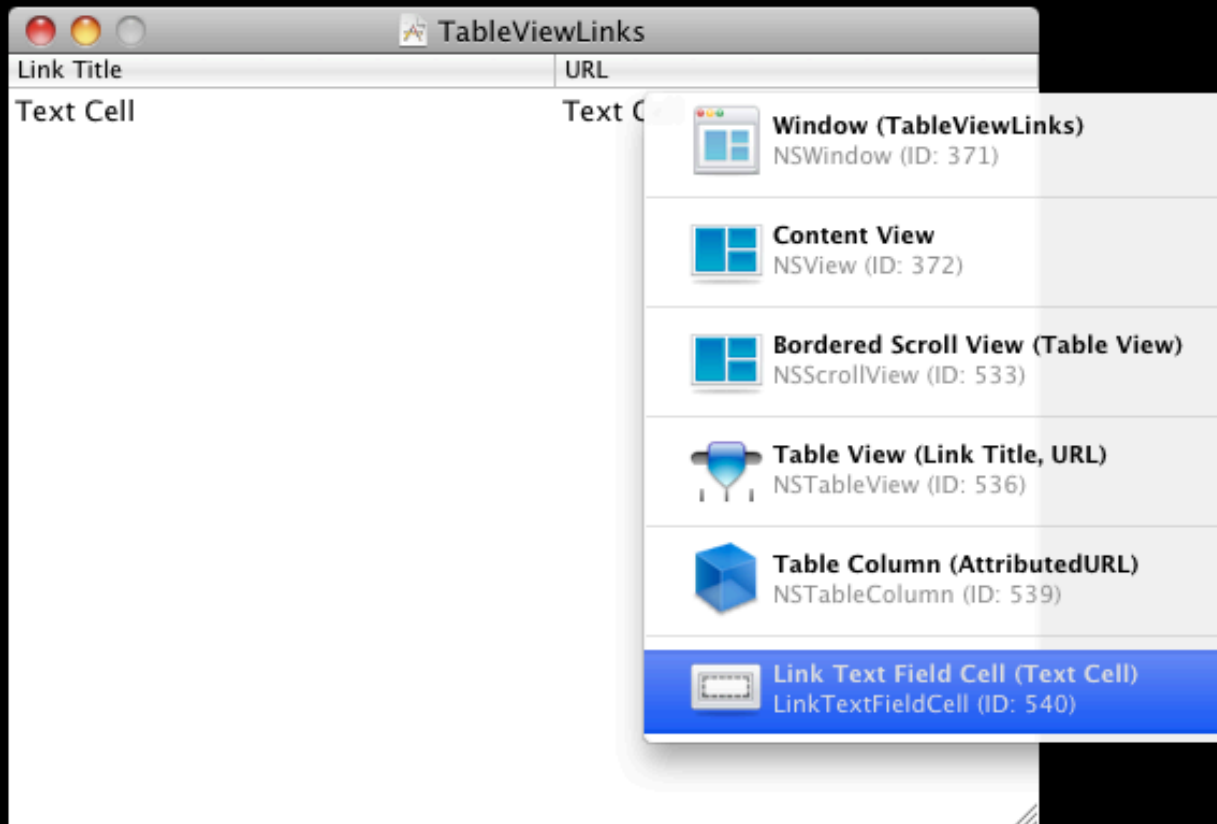
# Custom Cell

## LinkTextFieldCell in the sample code

```
- (BOOL)trackMouse:(NSEvent *)theEvent inRect:(NSRect)cellFrame
    ofView:(NSView *)controlView untilMouseUp:(BOOL)flag {

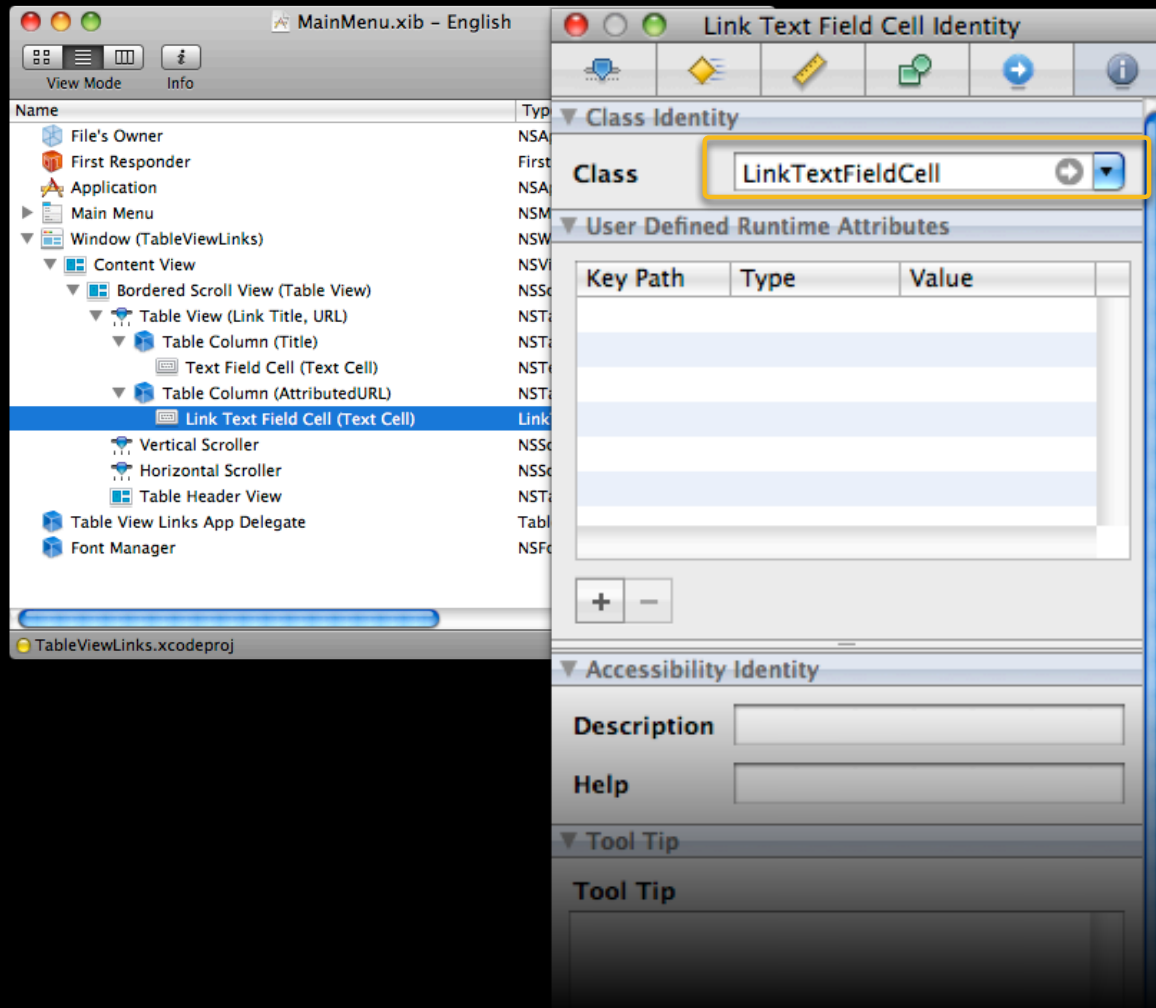
    BOOL result = [super trackMouse:theEvent inRect:cellFrame
                    ofView:controlView untilMouseUp:flag];
    NSAttributedString *attrValue = [self attributedStringValue];
    NSRange effectiveRange;
    NSURL *link = [attrValue attribute:NSLinkAttributeName
                               atIndex:0 effectiveRange:&effectiveRange];
    if (link != nil && _linkClickedHandler != nil) {
        _linkClickedHandler(link, self);
    }
    return result;
}
```

# Ctrl-Shift Click in Interface Builder



# Using a Custom Cell in IB

Selection in the nib



# Using Blocks as a Click Handler

```
- (void)tableView:(NSTableView *)tableView willDisplayCell:(id)cell
    forTableColumn:(NSTableColumn *)tableColumn
    row:(NSInteger)row {
    if ([cell isKindOfClass:[LinkTextFieldCell class]]) {
        LinkTextFieldCell *linkCell = (LinkTextFieldCell *)cell;
        // Setup the work to be done when a link is clicked
        linkCell.linkClickedHandler = ^(NSURL *url, id sender) {
            [[NSWorkspace sharedWorkspace] openURL:url];
        };
    }
}
```

# Using Blocks as a Click Handler

```
- (void)tableView:(NSTableView *)tableView willDisplayCell:(id)cell
    forTableColumn:(NSTableColumn *)tableColumn
    row:(NSInteger)row {
    if ([cell isKindOfClass:[LinkTextFieldCell class]]) {
        LinkTextFieldCell *linkCell = (LinkTextFieldCell *)cell;
        // Setup the work to be done when a link is clicked
        linkCell.linkClickedHandler = ^(NSURL *url, id sender) {
            [[NSWorkspace sharedWorkspace] openURL:url];
        };
    }
}
```

# Document Tips

Mark Piccirelli  
Cocoa Software Engineer

# Overriding NSDocument Correctly

- Try to override as little as possible
  - Don't override a `-save...` method when you can override a `-write...` method
- See "Message Flow in the Document Architecture"
  - See `<AppKit/NSDocument.h>` comments too
- We keep adding features
  - The less customization you do, the more your app gets for free



# Overriding NSDocument Correctly

- –write... and –save... methods are passed the URL and type
- Use those instead of [self fileURL] and [self fileType]

```
- (BOOL)writeToURL:(NSURL *)url ofType:(NSString *)typeName  
    error:(NSError **)outError;
```

- Don't set values in –write... methods

# Use Blocks Even with Pre-Block API

- AppKit has a lot of methods that look like this

```
- (void)saveDocumentWithDelegate:(id)delegate  
    didSaveSelector:(SEL)didSaveSelector  
    contextInfo:(void *)contextInfo;
```

- You have to add a method just to find out when the work is done
- With blocks you can add a few generic methods and use them over and over

# Use Blocks Even with Pre-Block API

- Add generic block-invoking methods like this

```
+ (void)document:(NSDocument *)document succeeded:(BOOL)didSucceed  
withCompletionHandler:(void (^)(BOOL didSucceed))completionHandler {  
    completionHandler(didSucceed);  
    Block_release(completionHandler);  
}
```

- And reuse them like this

```
[self saveDocumentWithDelegate:[self class]  
    didSaveSelector:@selector(document:succeeded:withCompletionHandler:)  
    contextInfo:Block_copy(^{BOOL didSave} {  
        // Saving is done here.  
    })];
```

# Use File Packages

- Good for new document formats that support lots of attachments
- NSFileWrapper!
  - Much improved in Snow Leopard
  - Uses hard links as an optimization
  - See the Mac OS 10.6 AppKit release notes

# Autosaving

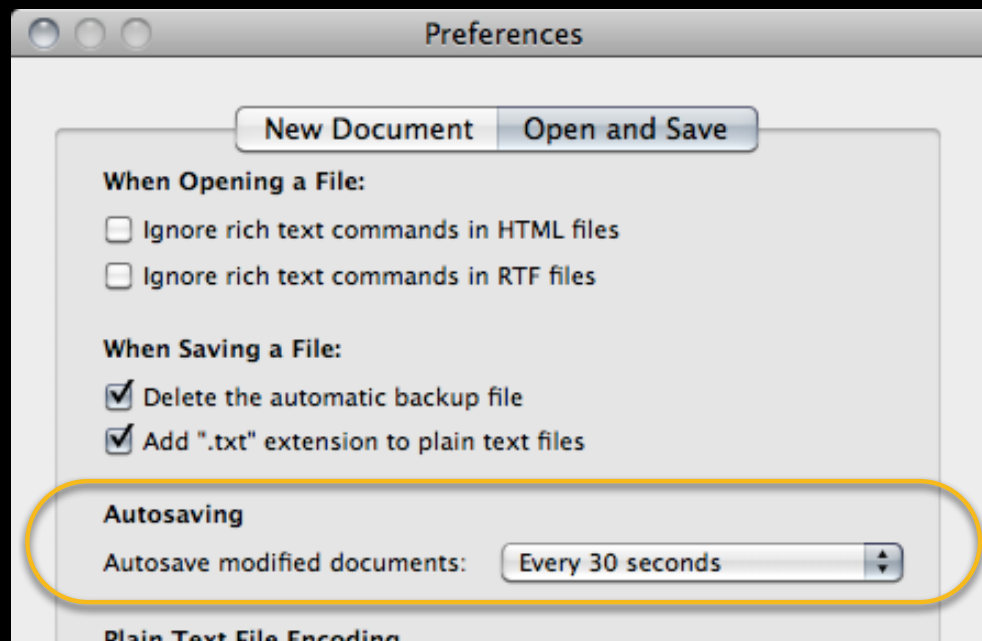
- Turn on autosaving in NSDocumentController

```
[[NSDocumentController sharedDocumentController] setAutosavingDelay:30];
```

- See the Mac OS 10.4 AppKit release notes for NSDocument methods to override and invoke

# Autosaving

- See TextEdit for example



- `/Developer/Examples/TextEdit`

# Exceptions and Error Handling

# Exceptions Are for Programming Errors

- In general, exceptions shouldn't be used for normal control flow
  - Your tests should run with no exceptions thrown
  - Unless you're using them within your own subsystems
- Be careful about catching and not rethrowing
  - Good at some subsystem boundaries
    - Grand Central Dispatch (libdispatch)
  - Bad almost everywhere else
  - At least log

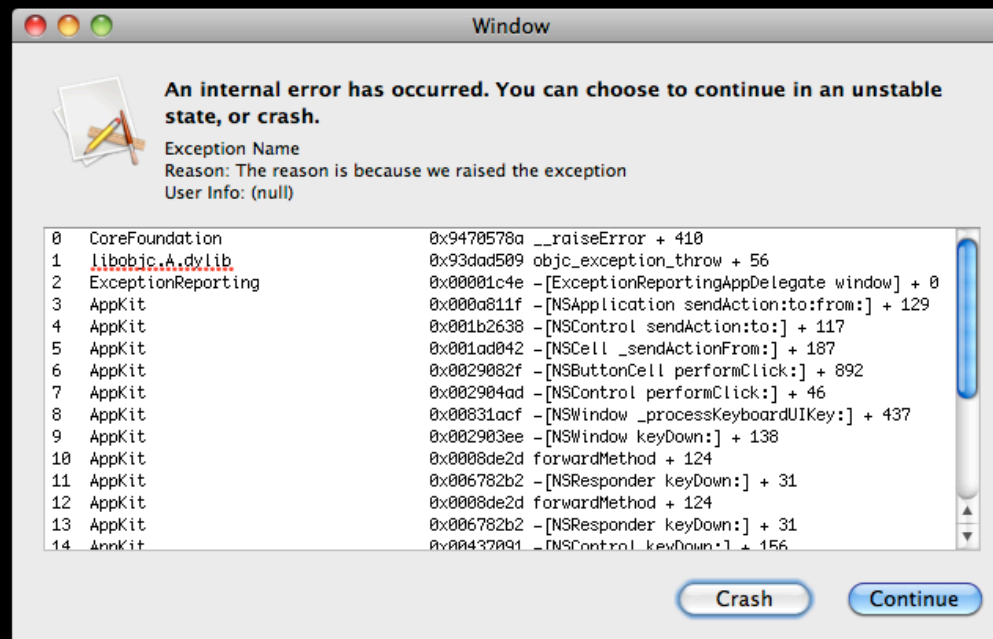


# Catching Exceptions in the Debugger

- Breaking on `objc_exception_throw` allows you to catch programming errors
  - “Stop on Objective-C Exceptions”
  - `NSAccessibilityException`

# Reporting Exceptions

Let the user see them and report bugs



# Reporting Exceptions

- Override `-[NSApplication reportException:]`

```
@implementation MyApplication

- (void)reportException:(NSException *)exception {
    @try {
        // Show [exception reason] and [exception callStackSymbols] to the user
        // See demo source code for an example
    } @catch (NSException *e) {
        // Suppress exceptions from the above code
    }
}

@end
```

# Reporting Exceptions

## Setting the custom NSObject subclass

Key	Value
▼ Information Property List	(13 items)
Localization native development re	English
Executable file	\${EXECUTABLE_NAME}
Icon file	
Bundle identifier	com.yourcompany.\${PRODUCT_NAME:rfc1034identifier}
InfoDictionary version	6.0
Bundle name	\${PRODUCT_NAME}
Bundle OS Type code	APPL
Bundle creator OS Type code	????
Bundle versions string, short	1.0
Minimum system version	\${MACOSX_DEPLOYMENT_TARGET}
Bundle version	1
Main nib file base name	MainMenu
Principal class	MyApplication

# Errors Are for Showing to the User

- Used in a lot of places in Cocoa now
- Meant to be “presentable” to the user
- NSResponder has methods for presenting them

```
- (void)presentError:(NSError *)error  
    modalForWindow:(NSWindow *)window  
        delegate:(id)delegate  
didPresentSelector:(SEL)didPresentSelector  
    contextInfo:(void *)contextInfo
```

```
- (BOOL)presentError:(NSError *)error;
```

# Nib Tips

**Corbin Dunn**  
Cocoa Software Engineer

# NSWindowController and NSViewController

```
MyWindowController *myWindowController =  
    [[MyWindowController alloc] initWithWindowNibName:@"Window"];  
  
[_myWindowController.window makeKeyAndOrderFront:nil];
```

- Top level objects automatically released for you

# Manually Loading Nibs

- An NSNib can be reused to create multiple instances from a nib

```
NSNib *nib = [[NSNib alloc] initWithNibNamed:@"NibName" bundle:nil];
```

```
// self has an IBOutlet called _secondWindow  
[nib instantiateNibWithOwner:self topLevelObjects:nil];  
[_secondWindow makeKeyAndOrderFront:nil];
```

```
// The IBOutlet is overwritten on the second call  
[nib instantiateNibWithOwner:self topLevelObjects:nil];  
[_secondWindow makeKeyAndOrderFront:nil];
```



# Manually Loading Nibs

- An NSNib can be reused to create multiple instances from a nib

```
NSNib *nib = [[NSNib alloc] initWithNibNamed:@"NibName" bundle:nil];
```

```
// self has an IBOutlet called _secondWindow
```

```
[nib instantiateNibWithOwner:self topLevelObjects:nil];  
[_secondWindow makeKeyAndOrderFront:nil];
```

```
// The IBOutlet is overwritten on the second call
```

```
[nib instantiateNibWithOwner:self topLevelObjects:nil];  
[_secondWindow makeKeyAndOrderFront:nil];
```

# Manually Loading Nibs

- An NSNib can be reused to create multiple instances from a nib

```
NSNib *nib = [[NSNib alloc] initWithNibNamed:@"NibName" bundle:nil];

// self has an IBOutlet called _secondWindow
[nib instantiateNibWithOwner:self topLevelObjects:nil];
[_secondWindow makeKeyAndOrderFront:nil];

// The IBOutlet is overwritten on the second call
[nib instantiateNibWithOwner:self topLevelObjects:nil];
[_secondWindow makeKeyAndOrderFront:nil];
```

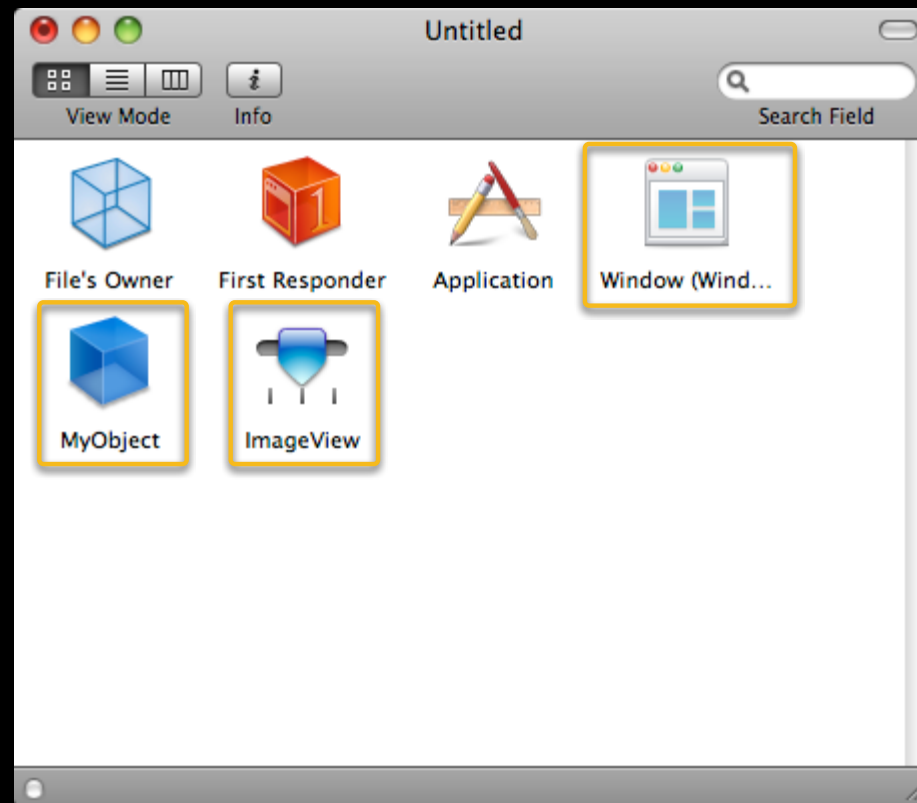
# The Owner and -awakeFromNib

```
- (void)awakeFromNib {  
    static NSInteger awakeFromNibCount = 0;  
    NSLog(@"awakeFromNib called %ld", (long)++awakeFromNibCount);  
}
```

```
2010-05-12 14:49:42.948 NibLoading[11709:a0f] awakeFromNib called 1  
2010-05-12 14:49:44.871 NibLoading[11709:a0f] awakeFromNib called 2  
2010-05-12 14:49:44.878 NibLoading[11709:a0f] awakeFromNib called 3
```

# Tips on Manually Loading Nibs

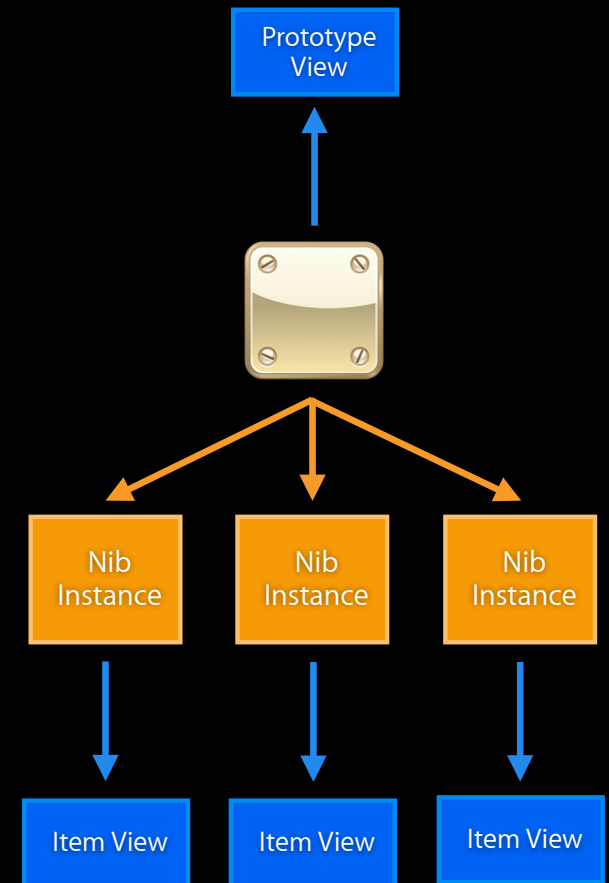
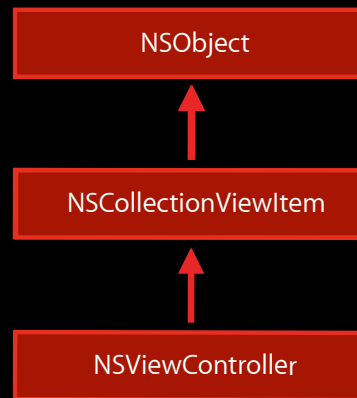
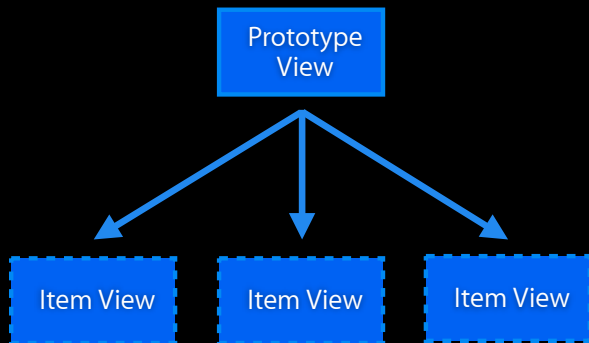
- You must release all top-level objects



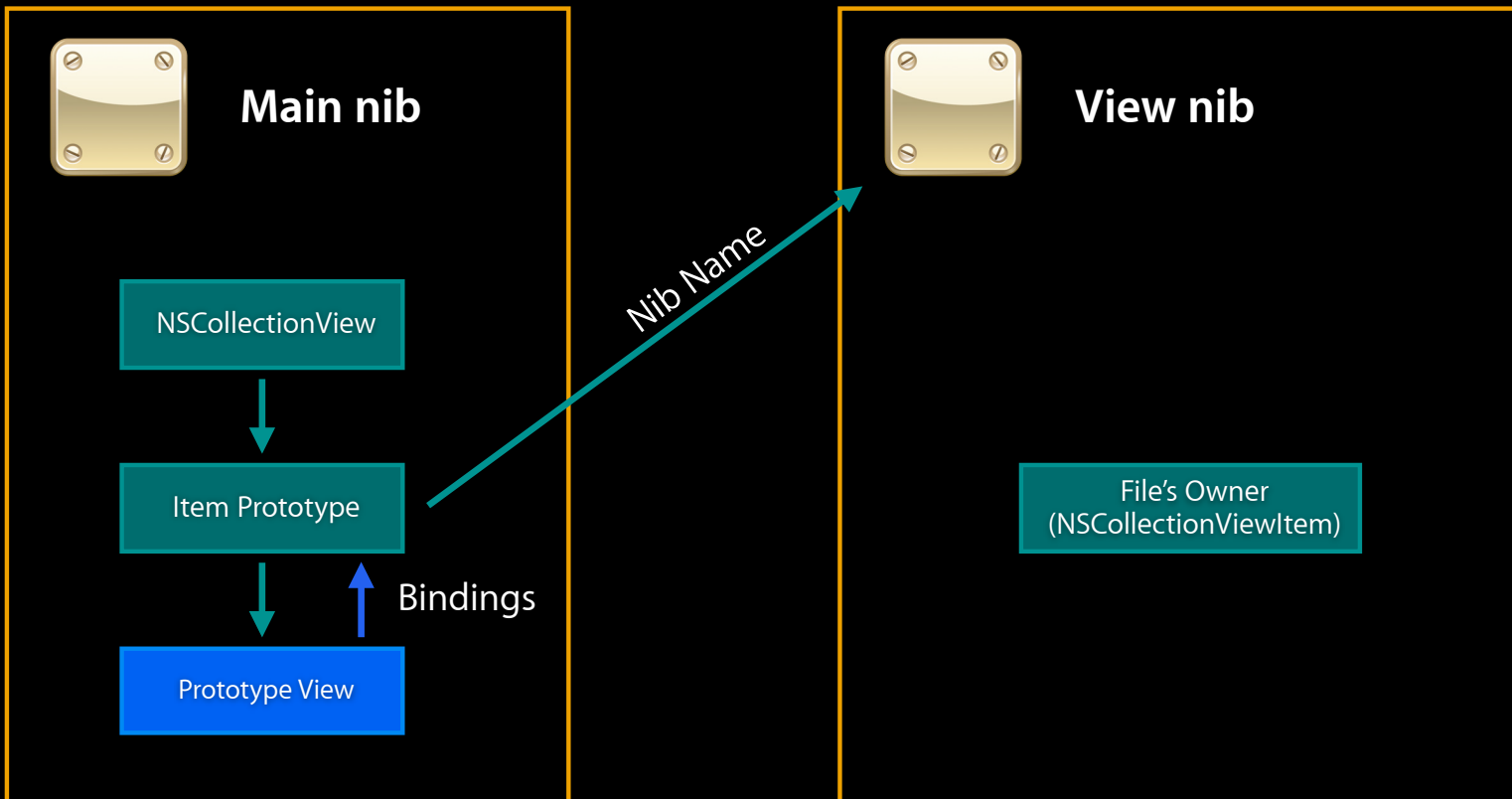
# SplitViews, PathControls, and NSCollectionView

Kevin Perry  
Cocoa Software Engineer

# Nib-Based NSCollectionViewItem



# Nib-Based NSCollectionViewItem



# Nib-Based NSCollectionViewItem

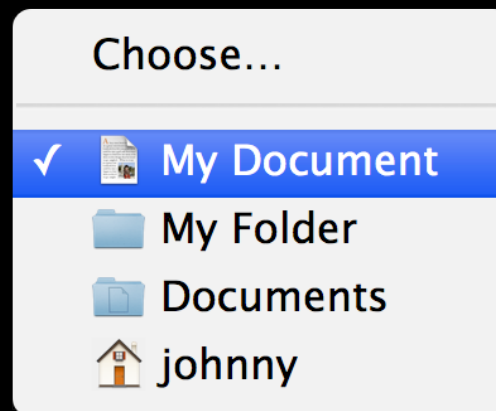
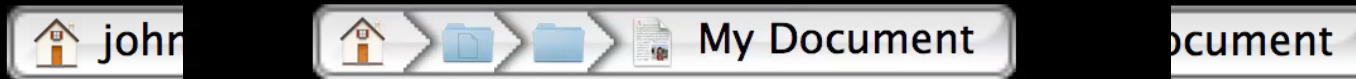
- Leopard

```
- (id)copyWithZone:(NSZone *)zone {  
    id copy = [[[self class] alloc] init];  
    NSNib *viewNib = // ...  
    [viewNib instantiateNibWithOwner:copy topLevelObjects:NULL];  
    return copy;  
}
```

```
- (void)loadView {  
    // Do nothing.  
}
```



# NSPathComponent



# NSPathControl

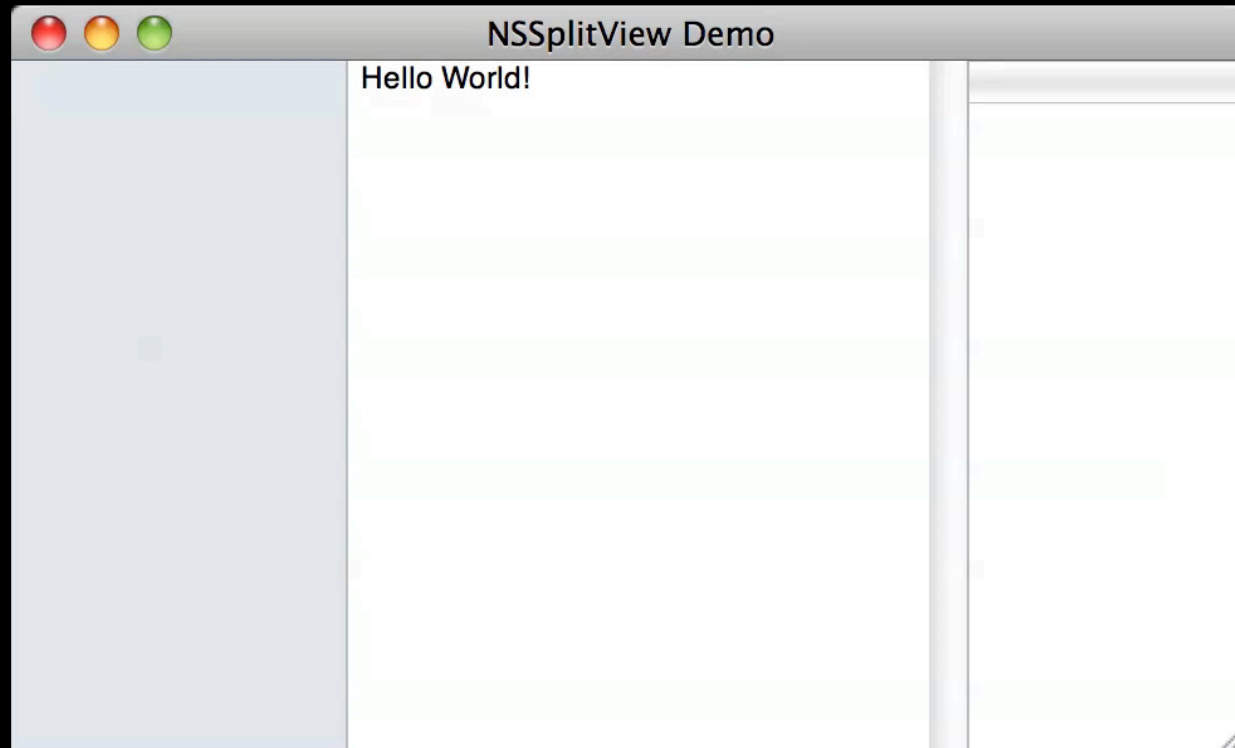
- More than file paths

- [NSPathControl `setStringValue:`]
  - [NSPathComponentCell `setImage:`]

- Custom drawing

- + [NSPathCell `PathComponentCellClass`]
  - [NSPathCell `rectOfPathComponentCell:withFrame:inView:`]

# NSSplitView



# NSSplitView

- Leopard and before

```
- (void)splitView:(NSSplitView *)sv
    resizeSubviewsWithOldSize:(NSSize)size {
    CGFloat deltaWidth = NSWidth([sv bounds]) - size.width;
    CGFloat height = NSHeight([sv bounds]);
    NSView *firstView = [[sv subviews] objectAtIndex:0];
    NSSize firstSize = [firstView frame].size;
    firstSize.height = height;
    [firstView setFrameSize:firstSize];
    NSView *middleView = [[sv subviews] objectAtIndex:1];
    NSSize middleSize = [middleView frame].size;
    middleSize.width += deltaWidth;
    middleSize.height = height;
    [middleView setFrameSize:middleSize];
    NSView *lastView = [[sv subviews] lastObject];
    NSRect lastFrame = [lastView frame];
    lastFrame.origin.x += deltaWidth;
    lastFrame.size.height = height;
    [lastView setFrame:lastFrame];
}
```

# NSSplitView

- Snow Leopard

```
- (BOOL)splitView:(NSSplitView *)sv  
    shouldAdjustSizeOfSubview:(NSView *)view {  
    return view == middleView;  
}
```

# Objective-C Tips

**Corbin Dunn**

Cocoa Software Engineer

# Static “Methods” in Objective-C

## Wait a minute—what and why?

```
@implementation AppDelegate
```

```
static void _processData(AppDelegate *self, NSInteger i) {  
    // We have a private _window ivar  
    NSLog(@"Processing %d in window %@", i, self->_window);  
}
```

```
- (void)applicationDidFinishLaunching:(NSNotification *)aNotification {  
    for (NSInteger i = 0; i < 200; i++) {  
        _processData(self, i);  
    }  
}
```

```
@end
```

# OBJC\_HELP

```
export OBJC_HELP=YES
```

```
corbin@neeb:/Applications/TextEdit.app/Contents/MacOS/TextEdit
objc[2569]: Objective-C runtime debugging. Set variable=YES to enable.
objc[2569]: OBJC_HELP: describe available environment variables
objc[2569]: OBJC_PRINT_OPTIONS: list which options are set
objc[2569]: OBJC_PRINT_IMAGES: log image and library names as they are loaded
objc[2569]: OBJC_PRINT_LOAD_METHODS: log calls to class and category +load methods
objc[2569]: OBJC_PRINT_INITIALIZE_METHODS: log calls to class +initialize methods
objc[2569]: OBJC_PRINT_RESOLVED_METHODS: log methods created by
    +resolveClassMethod: and +resolveInstanceMethod:
objc[2569]: OBJC_PRINT_CLASS_SETUP: log progress of class and category setup
objc[2569]: OBJC_PRINT_PROTOCOL_SETUP: log progress of protocol setup
objc[2569]: OBJC_PRINT_IVAR_SETUP: log processing of non-fragile ivars
objc[2569]: OBJC_PRINT_VTABLE_SETUP: log processing of class vtables
objc[2569]: OBJC_PRINT_VTABLE_IMAGES: print vtable images showing overridden
    methods
objc[2569]: OBJC_PRINT_CACHE_SETUP: log processing of method caches
objc[2569]: OBJC_PRINT_FUTURE_CLASSES: log use of future classes for toll-free
    bridging
objc[2569]: OBJC_PRINT_RTP: log initialization of the Objective-C runtime pages
objc[2569]: OBJC_PRINT_GC: log some GC operations
objc[2569]: OBJC_PRINT_PREOPTIMIZATION: log preoptimization courtesy of dyld shared
```

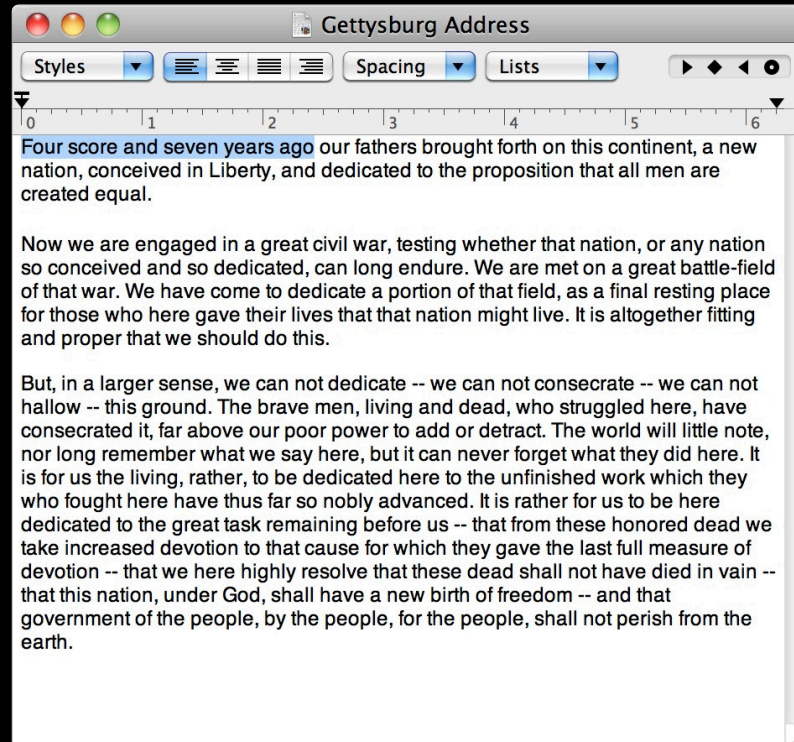


# Demo: Xcode Cocoa Tricks

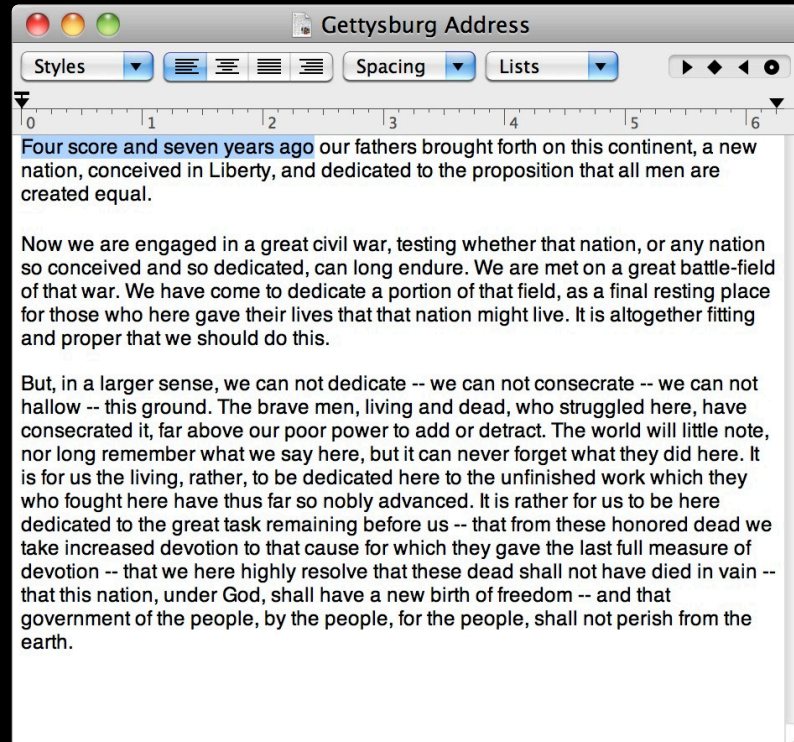
**Corbin Dunn**  
Cocoa Software Engineer

# The Secret Life of Your App's User Interface

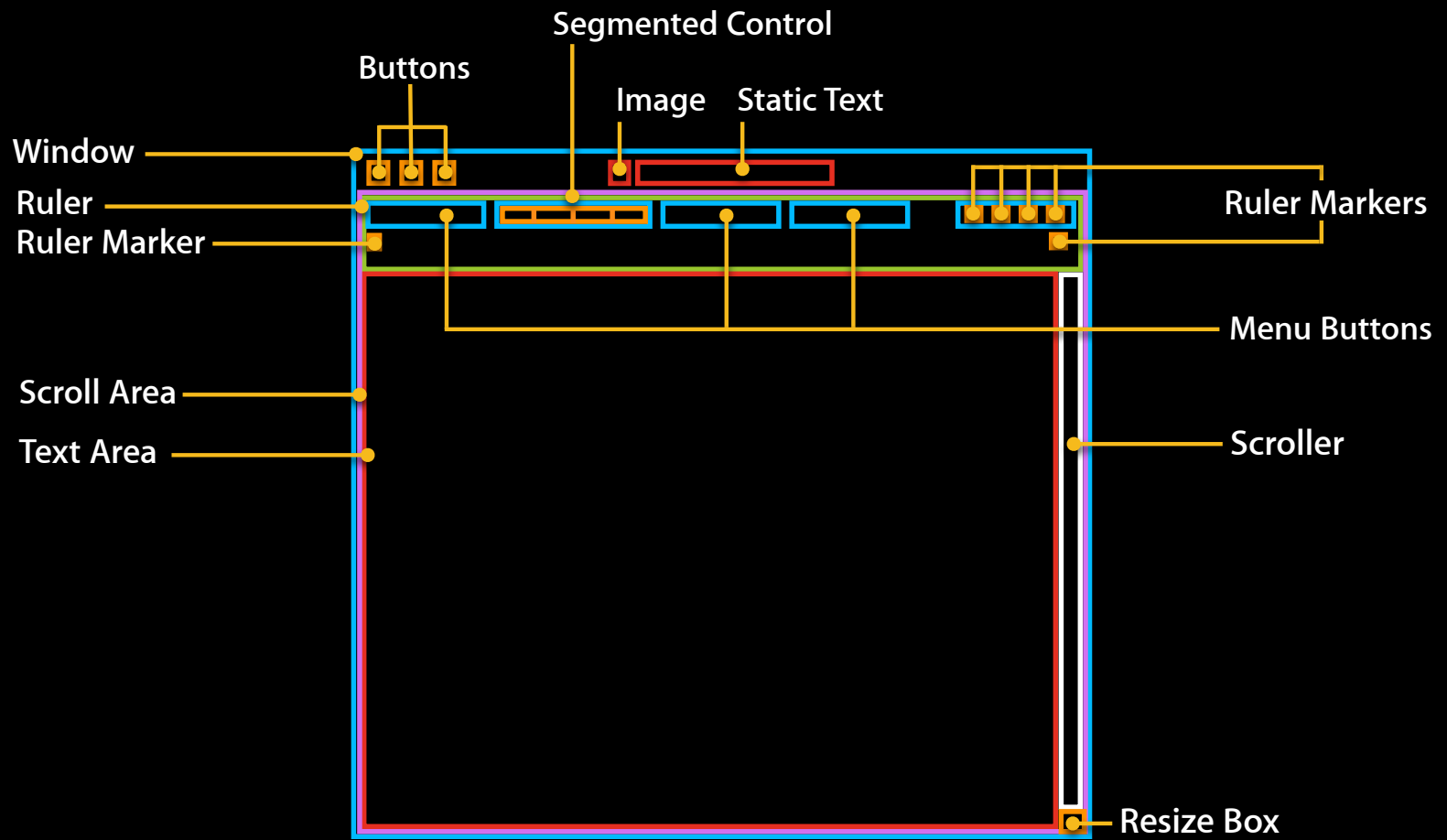
**James Dempsey**  
Cocoa Software Engineer

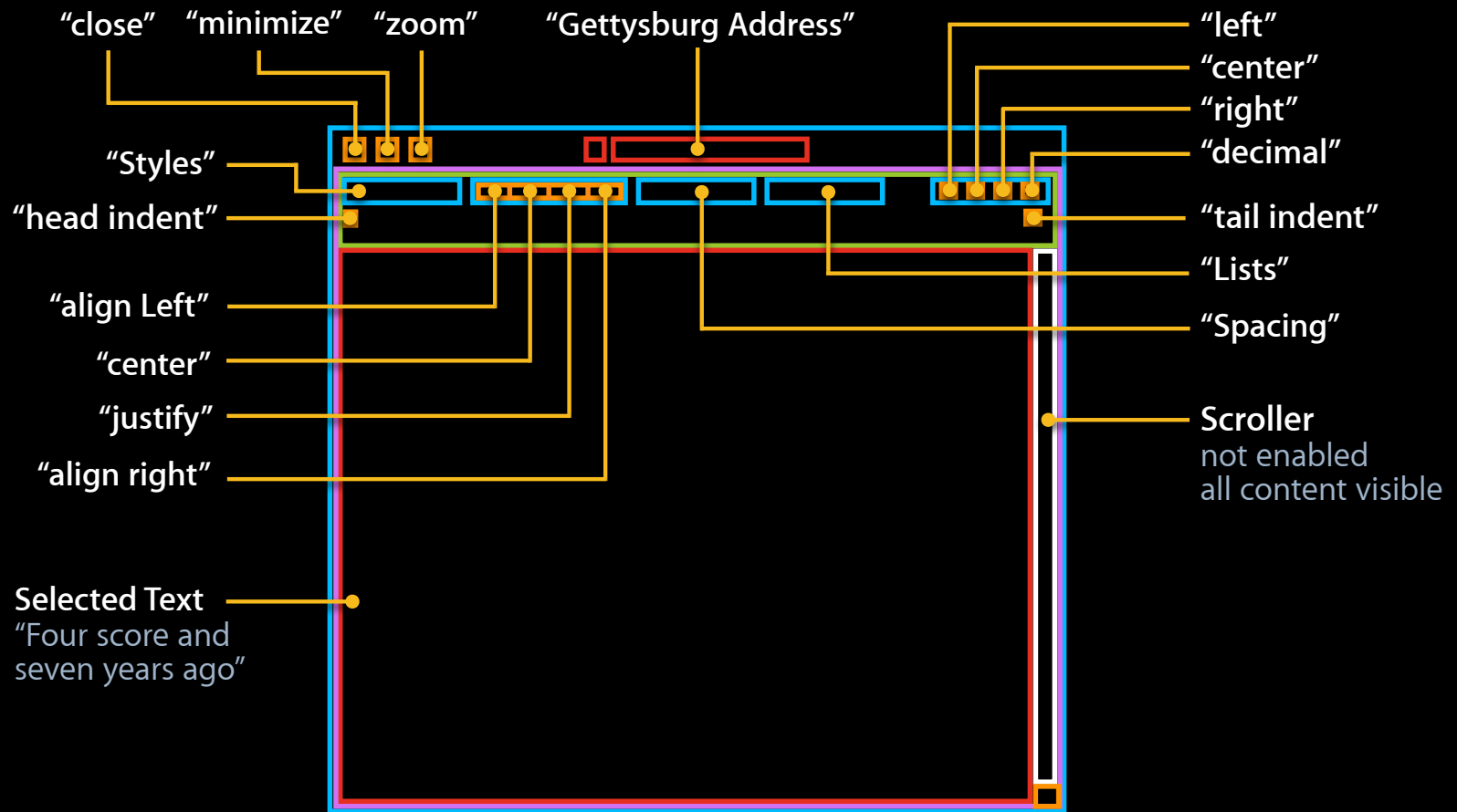




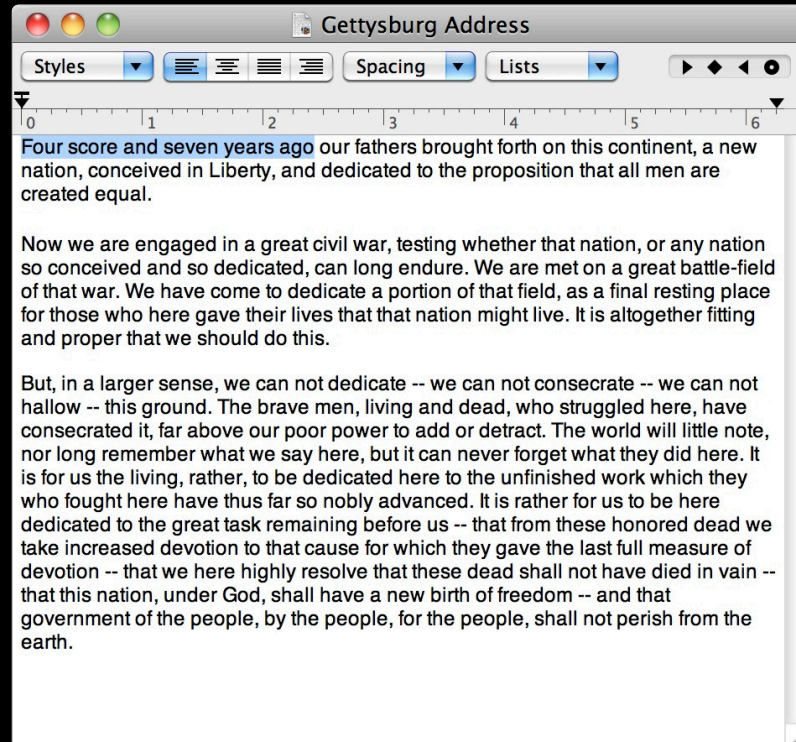


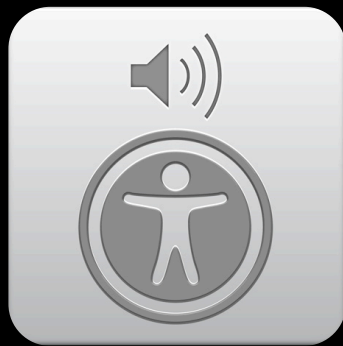












VoiceOver



Your App

- ✓ Use Standard Controls
- ✓ Describe UI Elements
- ✓ Reveal Custom Views

# Demo: Easy Image Descriptions

**James Dempsey**  
Cocoa Software Engineer

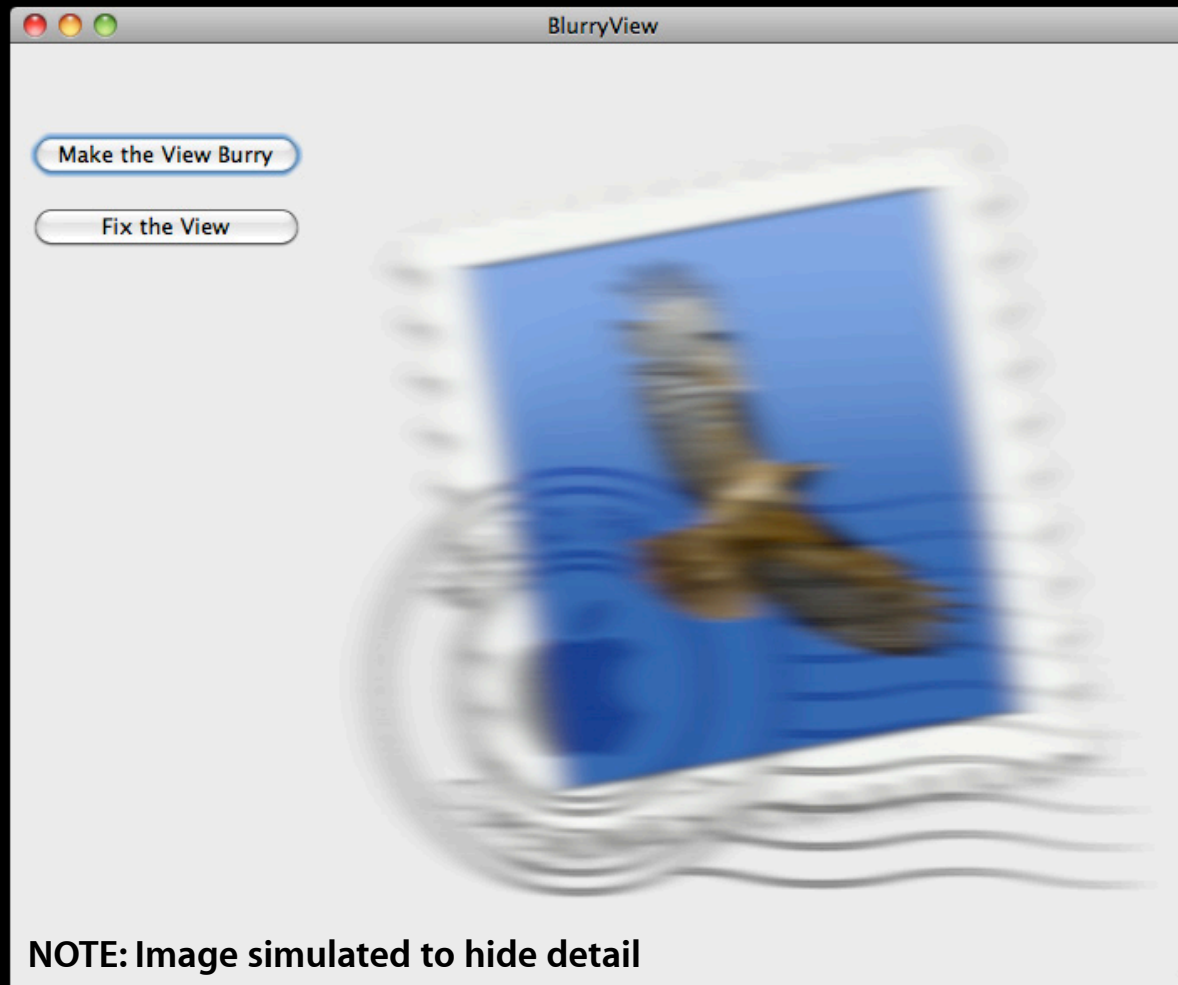
# Demo: Automated UI Testing

**James Dempsey**  
Cocoa Software Engineer

# Blurry Views

**Corbin Dunn**  
Cocoa Software Engineer

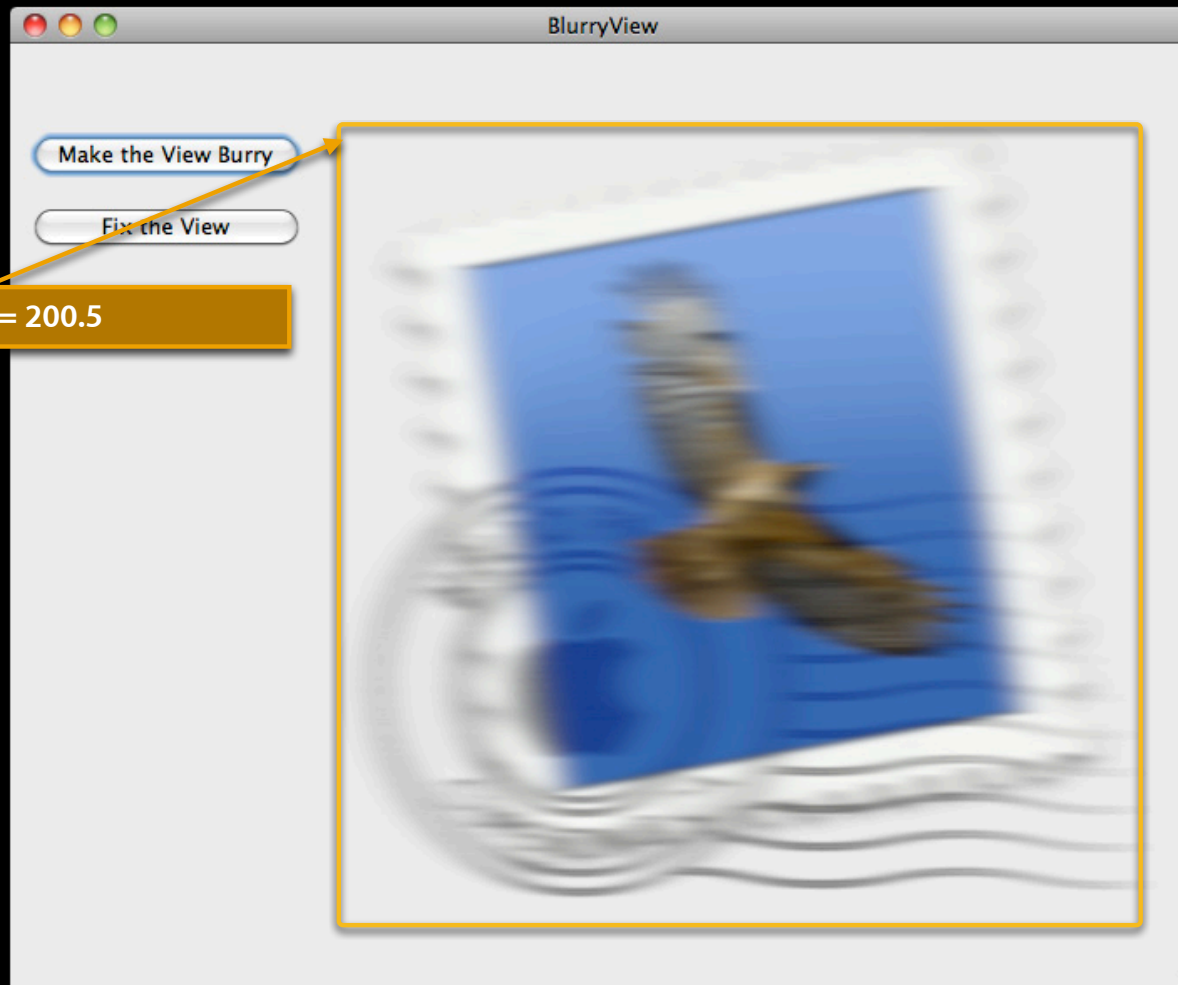
# Have You Seen Blurry Views?



# Have You Seen Blurry Views?



# Cause: Non Integral Pixel Alignment





# Two Solutions: First—centerScanRect:

```
CGRect frame = [_imageView frame];
```

```
frame = [[_imageView superview] centerScanRect:frame];
```

```
[_imageView setFrame:frame];
```

# Two Solutions: Second—Explicit

```
CGRect frame = [_imageView frame];
```

```
// Convert to pixel space
```

```
frame = [_imageView.superview convertRectToBase:frame];
```

```
// At this point we can floor/round/ceil
```

```
frame.origin.x = floor(frame.origin.x);
```

```
frame.origin.y = floor(frame.origin.y);
```

```
// Convert back to the view's coordinate space
```

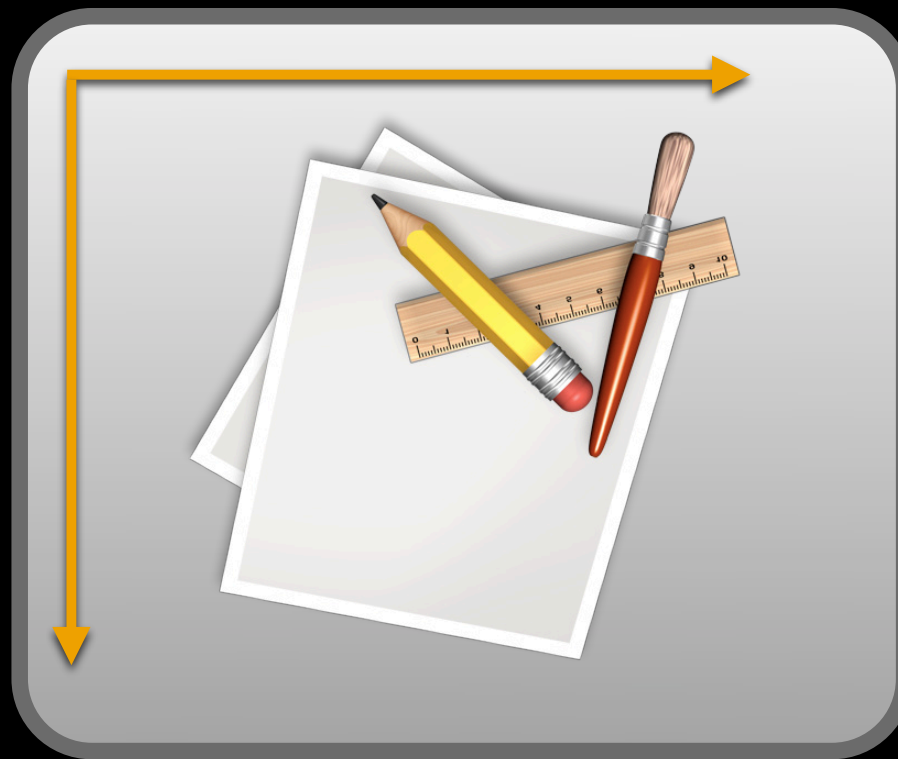
```
frame = [_imageView.superview convertRectFromBase:frame];
```

```
[_imageView setFrame:frame];
```

# Image Tips

**Ken Ferry**  
Cocoa Software Engineer

# Flippedness



# Flippedness

Drawing images right side up

```
–[UIImage drawInRect:fromRect:operation:fraction:respectFlipped:hints:]
```

```
–[UIImage compositeToPoint:fromRect:operation:fraction:]
```

```
–[UIImage setFlipped:]
```

# Mutation

Non-local effects

[image setFlipped:YES]



# Mutation

Mutate as part of set up

```
[[UIImage alloc] init]
```

```
set...
```

```
set...
```

```
set...
```

**FREEZE**



```
[image copy]
```

```
set...
```

```
set...
```

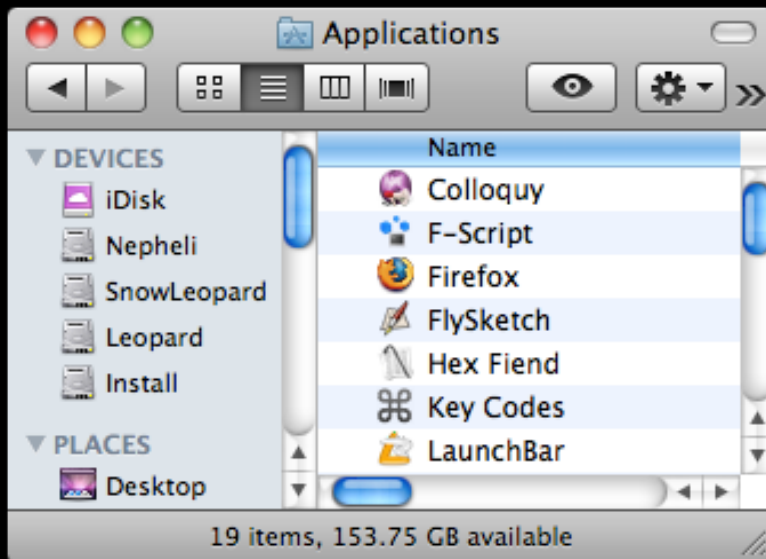
```
set...
```

**FREEZE**



# Performance

Same drawing, same image



```
-drawRect: {  
    [[NSImage alloc] initWithImage:  
}
```



```
[cell setImage:]
```





# NSImage Wrap Up

- Use -draw methods respecting flippedness
- Mutate only as part of setup
- Perform identical drawing with identical images

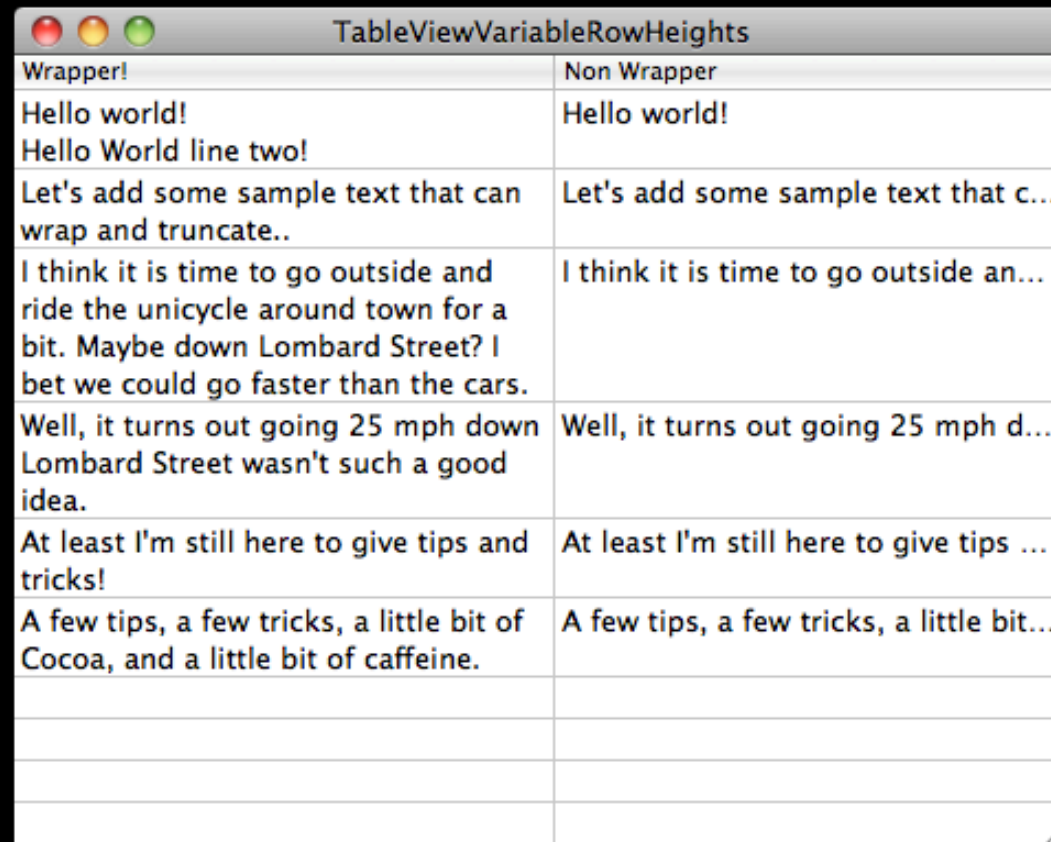
10.6 AppKit release notes  
NSImage in SnowLeopard, WWDC 2009

# More Table Tips

**Corbin Dunn**

Cocoa Software Engineer

# Variable Row Heights



Wrapper!	Non Wrapper
Hello world! Hello World line two!	Hello world!
Let's add some sample text that can wrap and truncate..	Let's add some sample text that c...
I think it is time to go outside and ride the unicycle around town for a bit. Maybe down Lombard Street? I bet we could go faster than the cars.	I think it is time to go outside an...
Well, it turns out going 25 mph down Lombard Street wasn't such a good idea.	Well, it turns out going 25 mph d...
At least I'm still here to give tips and tricks!	At least I'm still here to give tips ...
A few tips, a few tricks, a little bit of Cocoa, and a little bit of caffeine.	A few tips, a few tricks, a little bit...

# Variable Row Heights Delegate

## Use the cell to find the preferred height

```
- (CGFloat)tableView:(NSTableView *)tableView heightForRow:(NSInteger)row {
```

```
    NSCell *cell = [tableView preparedCellAtColumn:0 row:row];
```

```
    CGRect constrainedBounds = CGRectMake(0, 0, _tableViewColumnWidth, CGFLOAT_MAX);
```

```
    CGSize naturalSize = [cell cellSizeForBounds:constrainedBounds];  
    return naturalSize.height;
```

```
}
```

# Variable Row Heights Delegate

## Cache the Table Column's width

```
- (void)tableViewColumnDidResize:(NSNotification *)notification {  
    _tableColumnWidth = [[_tableView tableViewColumnWithIdentifier:@"ID"] width];  
    NSIndexPath *allRows = [NSIndexPath indexSetWithIndexesInRange:  
                            NSRange(0, _tableView.numberOfRows)]  
    [_tableView noteHeightOfRowsWithIndexesChanged:allRows];  
}
```

# Summary

- Lots of tips
- Hopefully you learned a few new things
- Download the sample code that illustrates most of these tips

# Labs

Cocoa Lab

Application Frameworks Lab C  
Tuesday 3:15PM–6:30PM

Cocoa Lab

Application Frameworks Lab D  
Thursday 2:00PM–4:15PM

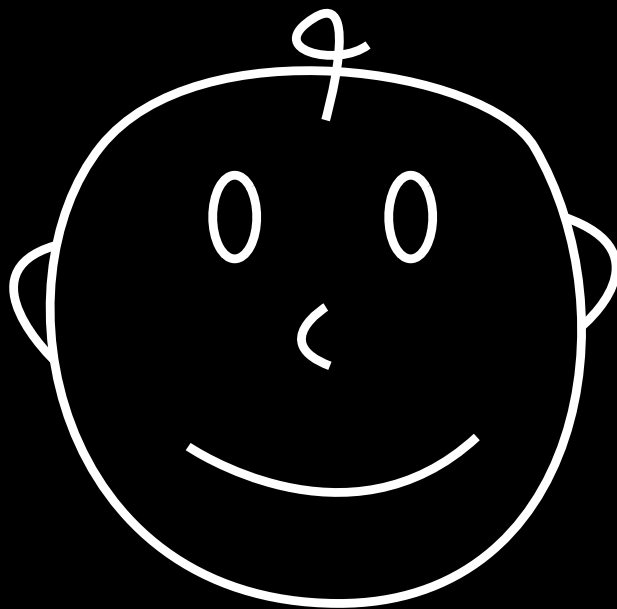
# Song

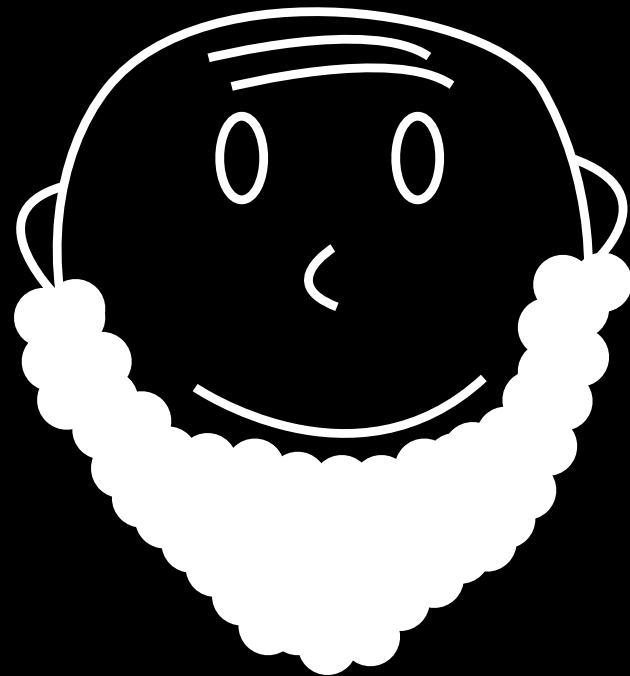
James Dempsey and the Breakpoints





# Anti-Patterns

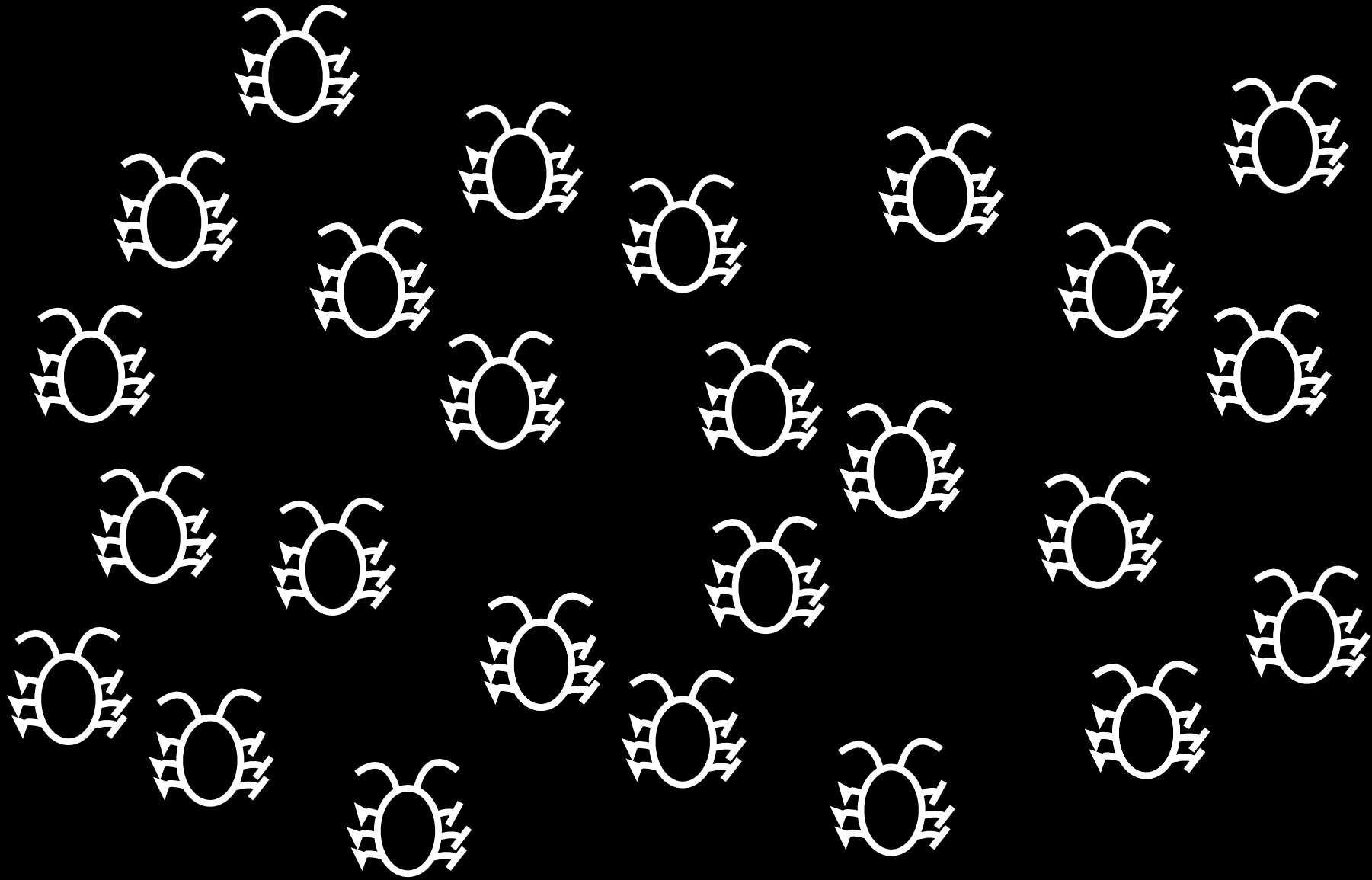


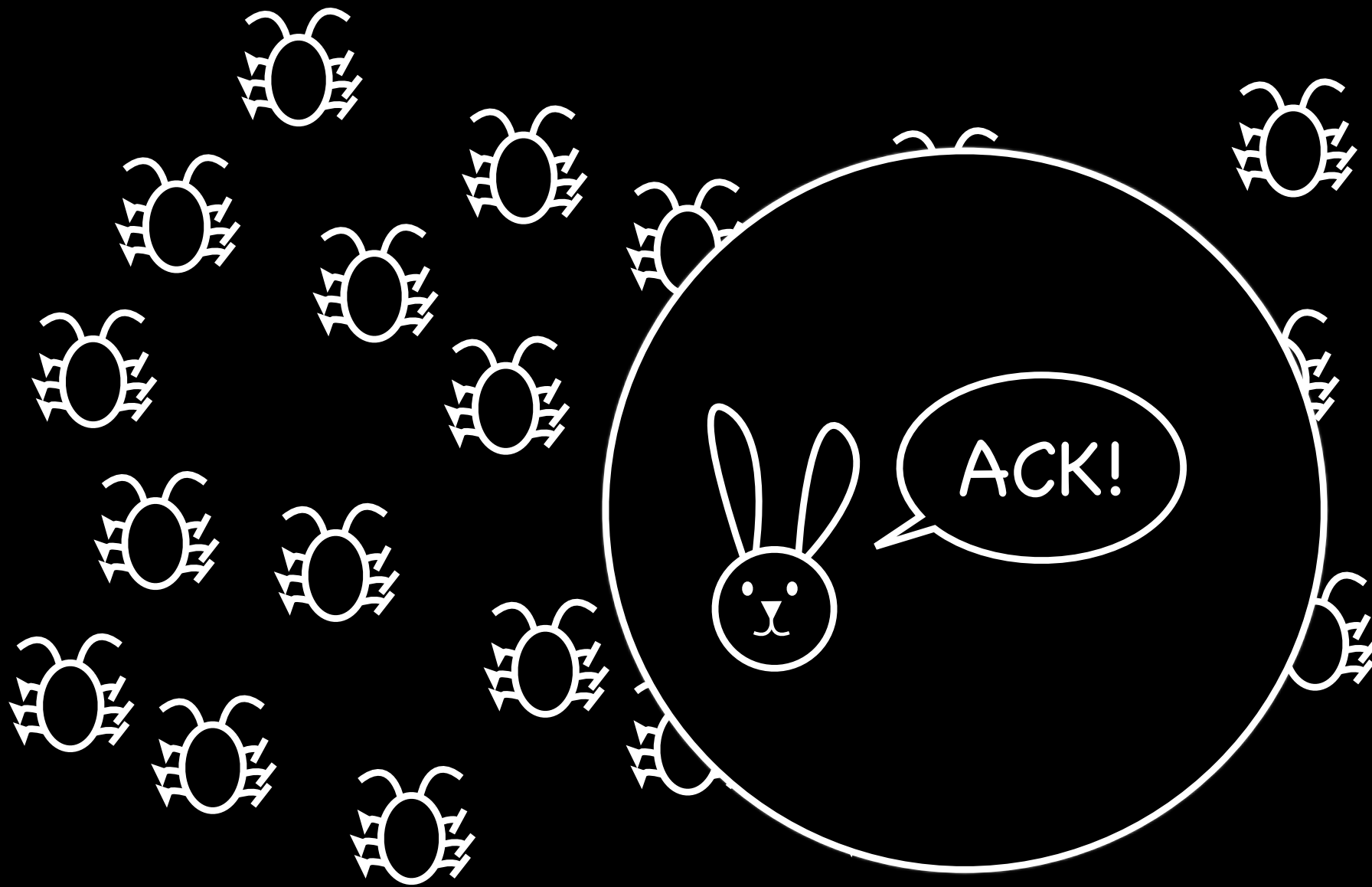




(Book Learnin' Too)



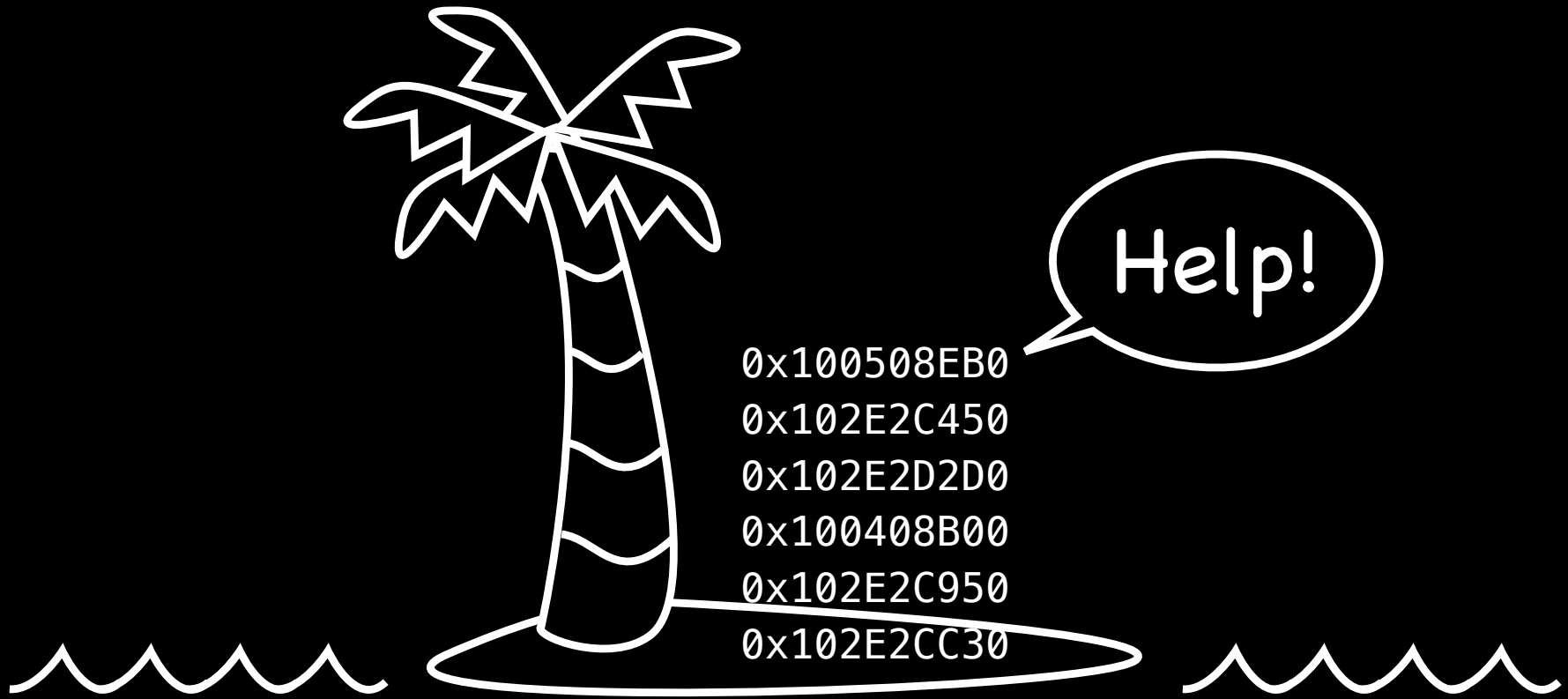




# Anti-Patterns



**“When in doubt...  
-~~explain~~”**



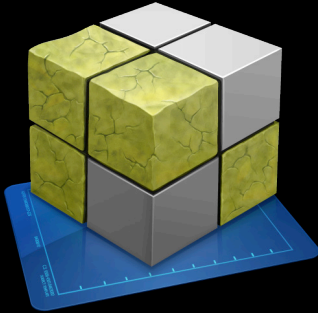
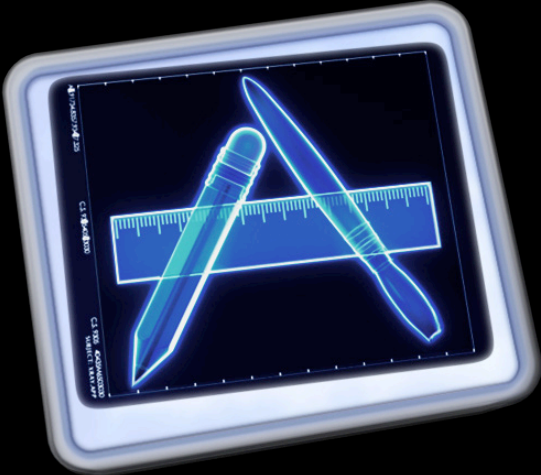
```
while ([self retainCount] > 0) {  
    [self release]  
}
```



---

# Memory Management Programming Guide for Cocoa

Performance



```
// Hold Me  
[myObj retain];
```

```
// Hold Me  
[myObj retain];
```

```
// Use Me  
[myObj doStuff];
```

```
// Hold Me  
[myObj retain];  
  
// Use Me  
[myObj doStuff];  
  
// Release Me  
[myObj release];
```



```
// Hold Me  
[myObj retain];  
  
// Use Me  
[myObj doStuff];  
  
// Release Me  
[myObj release];
```

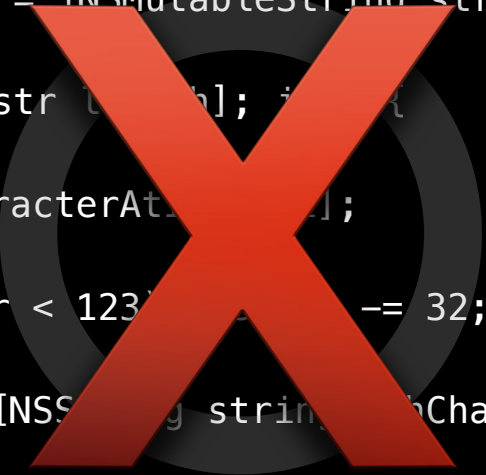
```
// Hold Me  
[myObj retain];  
  
// Use Me  
[myObj doStuff];  
  
// Release Me  
[myObj release];
```

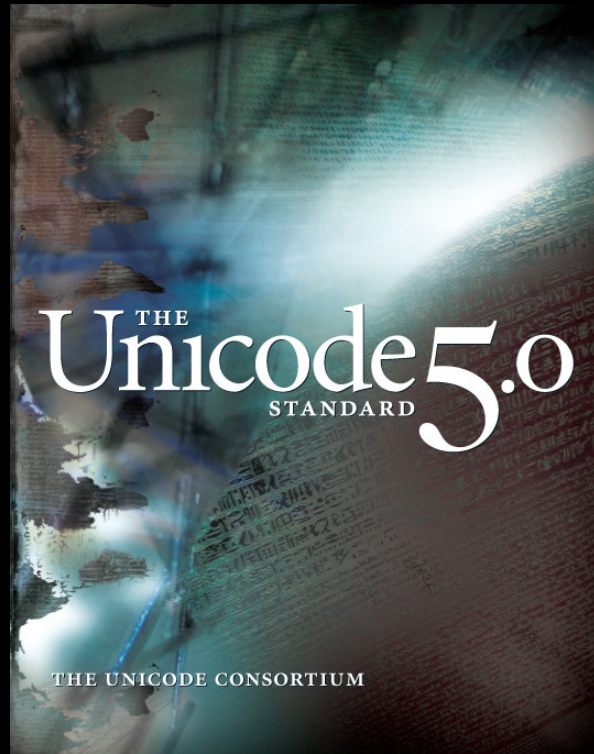
```
// Hold Me  
[myObj retain];  
  
// Use Me  
[myObj doStuff];  
  
// Release Me  
[myObj release];
```

# Anti-Patterns



```
NSMutableString *upperString = [NSMutableString string];
for (NSUInteger i = 0; i < [string length]; i++) {
    unichar theChar = [string characterAtIndex:i];
    if (theChar > 96 && theChar < 123) theChar -= 32;
    [upperString appendString:[NSString stringWithCharacters:&theChar length:1]];
}
```





1472 page printed document

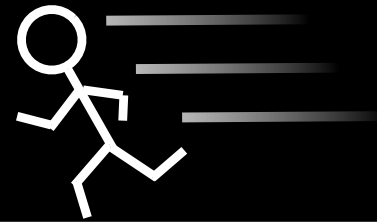
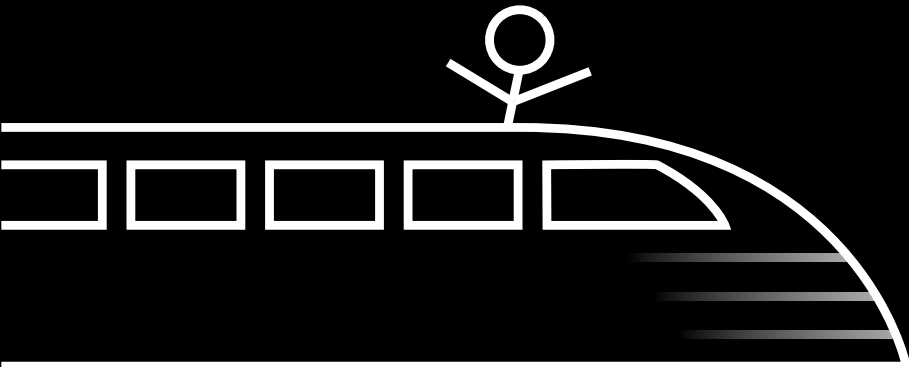
# Anti-Patterns



```
[myObject __secretPrivateMethod];
```

**CENSORED**

**I BREAK FOR  
ENCAPSULATION**



# Code Snippet Hall of Shame (Abridged)

-subviews returns  
NSArray

```
[(NSMutableArray *) [myView subviews] usingSelector:@selector(sortViews:)];
```

**This violates the  
contract in a big way**

```
// 1. Get a menu  
NSMenuItem *aMenu = [mainMenu itemAtIndex:index];
```

```
// 2. Pretend its not a menu  
NSMenu *someMenu = (NSMenu *)aMenu;
```

```
// 3. Hilarity ensues:  
-[NSMenuItem superme : unrecognized selector sent  
to instance 0x10010ca70
```



Don't overestimate  
the power of  
typecasting

```
static NSDictionary *sharedDict = nil;
```

```
@synchronized (sharedDict) {  
    sharedDict =  
        [[NSDictionary alloc] initWithObjectsAndKeys:  
         @"foo", forKey:@"foo",  
         @"bar", forKey:@"bar"];  
}
```



Did this really just synchronize on nil?





```
[self lock];
```

-release perhaps?

# Anti-Patterns

```
NSObject *objectID = 0;
```

```
for (NSUInteger i=0; i < count; ++i) {
```

Looping back to the head of the loop

```
    NSObject *object = [trackedElements objectAtIndex:i];
```

```
    if ([object isKindOfClass:[NSString class]])
```

```
    {
```

```
        objectID = [[NSString alloc] initWithString:aString];
```

Method returns an Objective-C object with a +1 retain count (owning reference)

```
        if (objectID != nil)
```

```
        {
```

```
            [objectID release];
```

Object released

Reference-counted object is used after it is released

```
        }
```

```
    }
```

# Anti-Patterns

# More Information

## Bill Dudney

Application Frameworks Evangelist  
[dudney@apple.com](mailto:dudney@apple.com)

## Apple Developer Forums

<http://devforums.apple.com>





