Key Equivalent Handling in Cocoa Applications

Part I—Key Equivalents

Peter Ammon
Cocoa Frameworks Engineer

Key Equivalents and Menus

- Key equivalent basics
- Keyboard event flow and override points
- Handling key events from other apps
- Conflict handling
- Debugging conflicts
- Custom menu items
- Simulating menus

Key Equivalents Basics

- Accelerate access to buttons and menu items with the keyboard

- Same API on both NSButton and NSMenuItem

Key Equivalents Basics

- Accelerate access to buttons and menu items with the keyboard

Key equivalent

```
[menuItem setKeyEquivalent:@”n”];
```

Lowercase!

Key Equivalents Basics

- Accelerate access to buttons and menu items with the keyboard

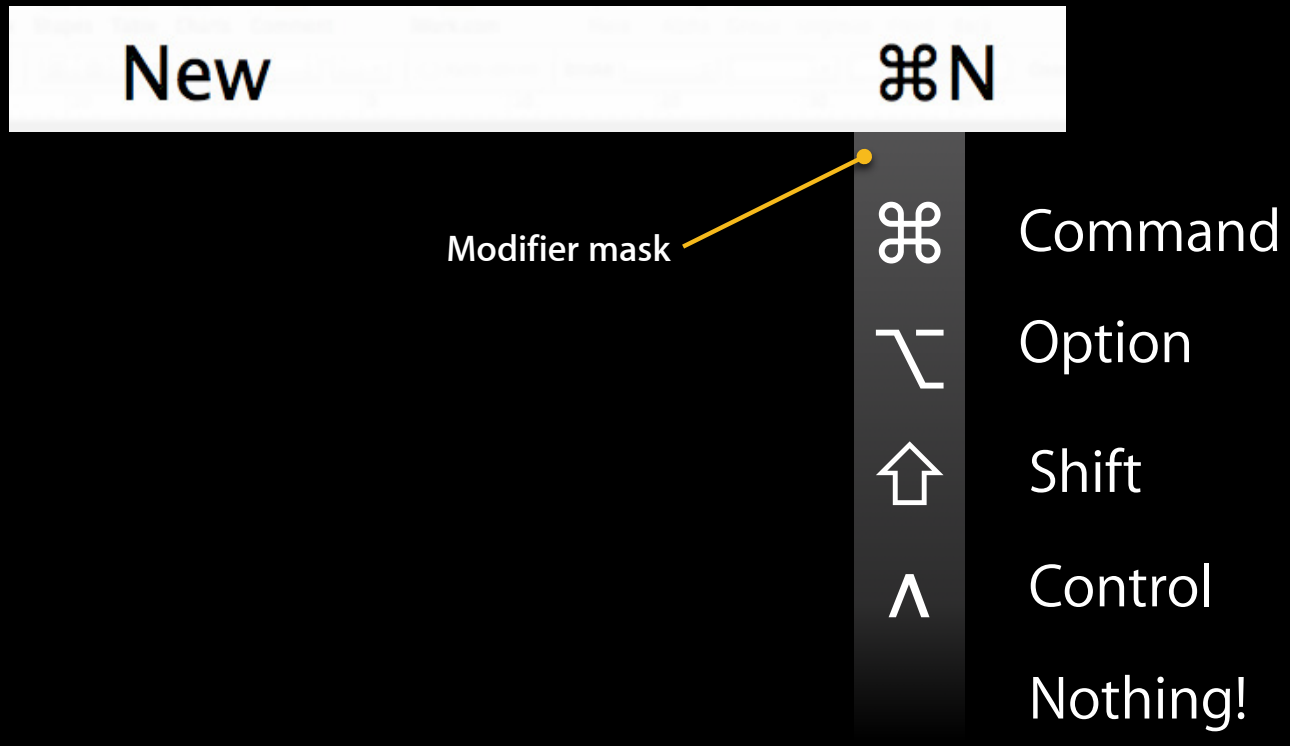
Key equivalent

```
[menuItem setKeyEquivalent:@"N"];
```

Oops!

Key Equivalents Basics

- Accelerate access to buttons and menu items with the keyboard



Key Equivalents Basics

- Accelerate access to buttons and menu items with the keyboard

Modifier mask

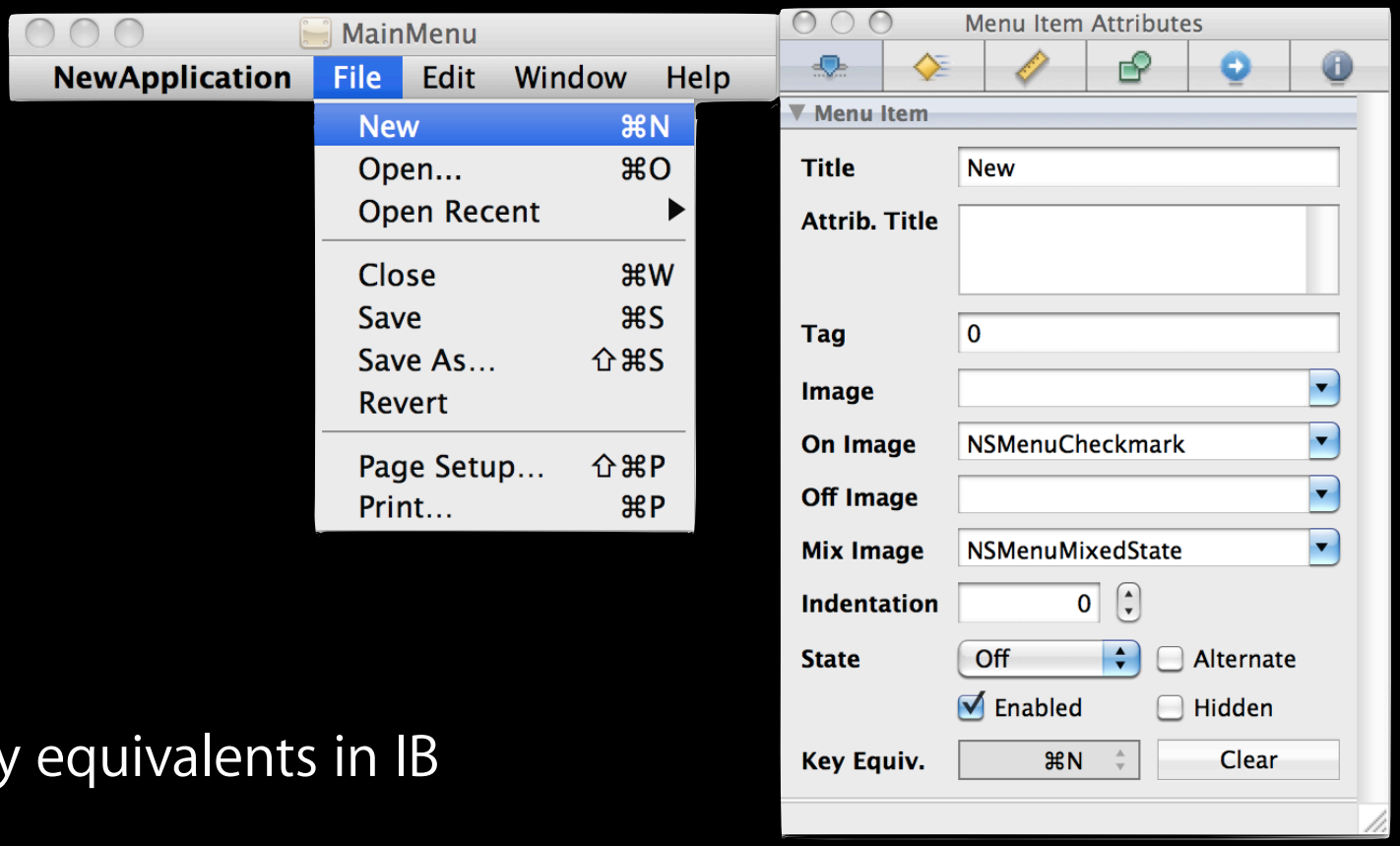
Key Equivalents Basics

- Accelerate access to buttons and menu items with the keyboard


```
[menuItem setKeyEquivalentModifierMask:NSCommandKeyMask];
```

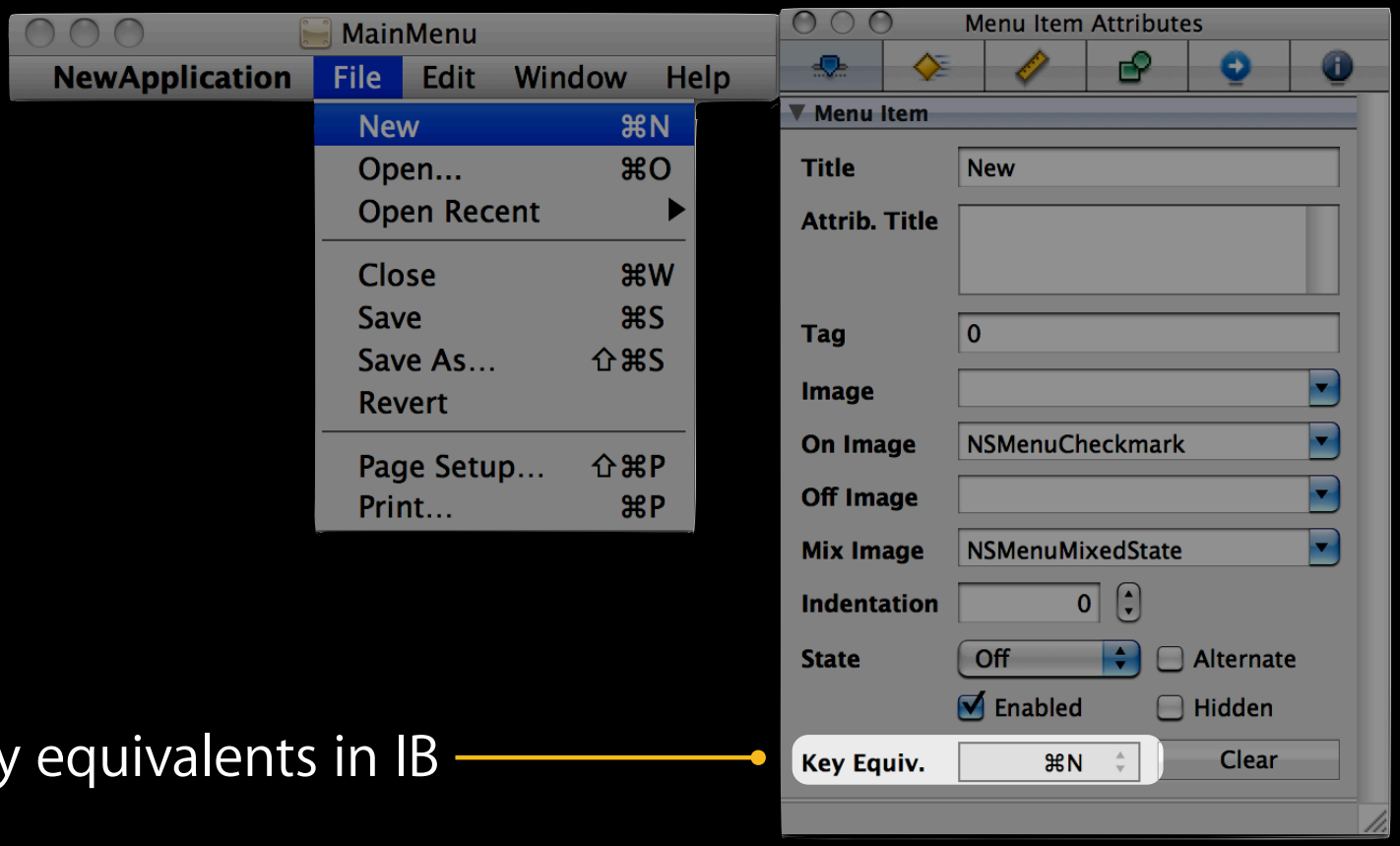
- Also on `NSButton`

Key Equivalents Basics



- Set key equivalents in IB

Key Equivalents Basics



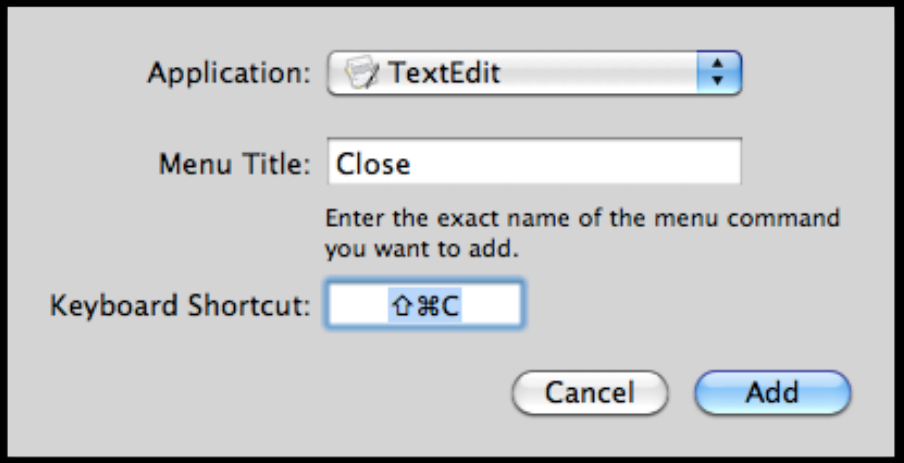
- Set key equivalents in IB

When to Use Key Equivalents

- Assign key equivalents to frequently used commands
- The Human Interface Guidelines lists a lot of key equivalents with well established meanings
- A lot
- Don’t repurpose standard or reserved key equivalents
- Use alternate items to enable power users without cluttering the UI
  - But always provide another way to accomplish the task

User Key Equivalents

- Users can customize them in System Preferences

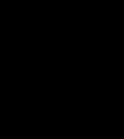


- User key equivalent

Key Equivalents and Menus

- Key equivalent basics
- Keyboard event flow and override points
- Handling key events from other apps
- Conflict handling
- Debugging conflicts
- Custom menu items
- Simulating menus

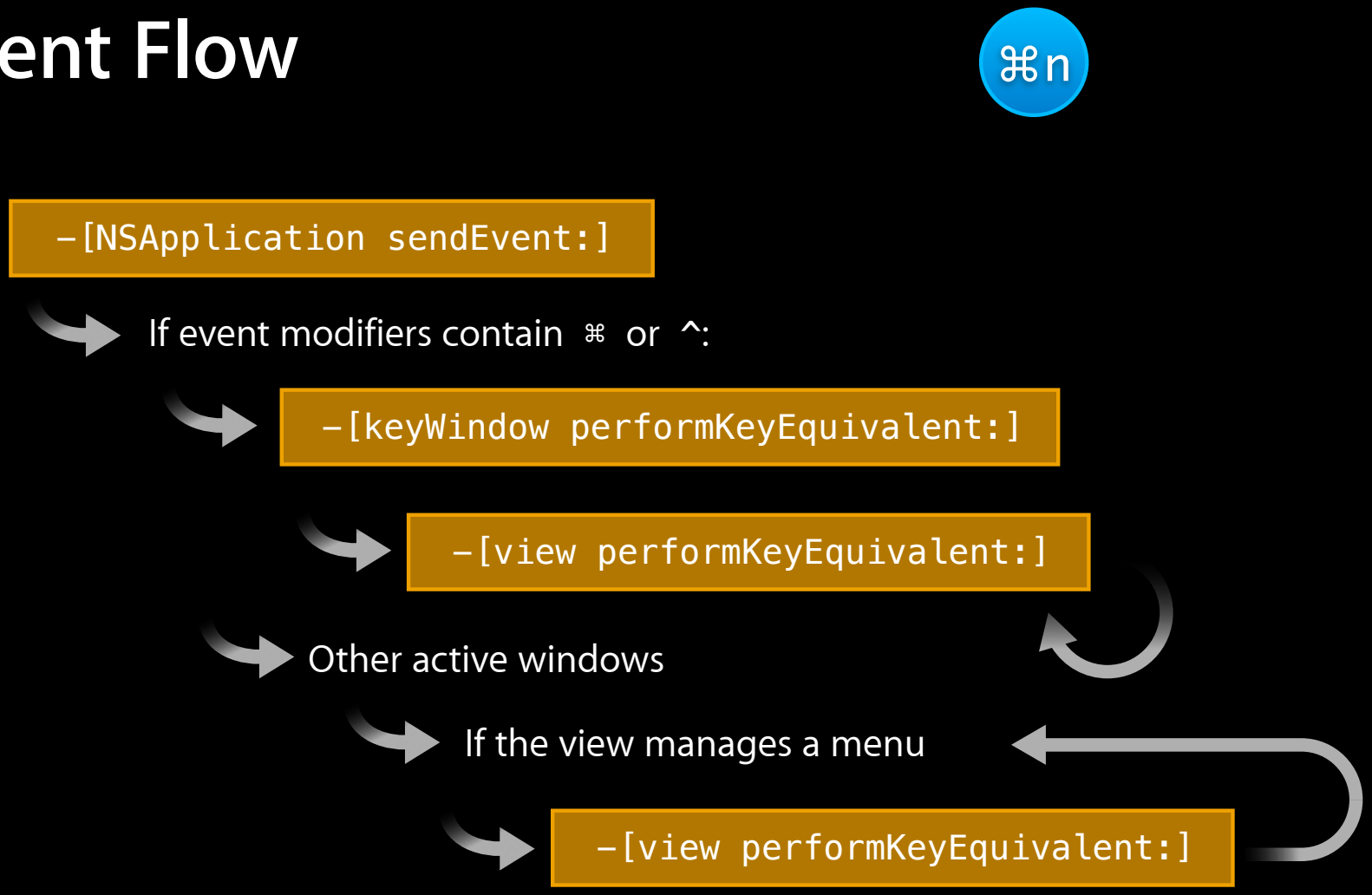
Event Flow




```
-[NSApplication sendEvent:]
```

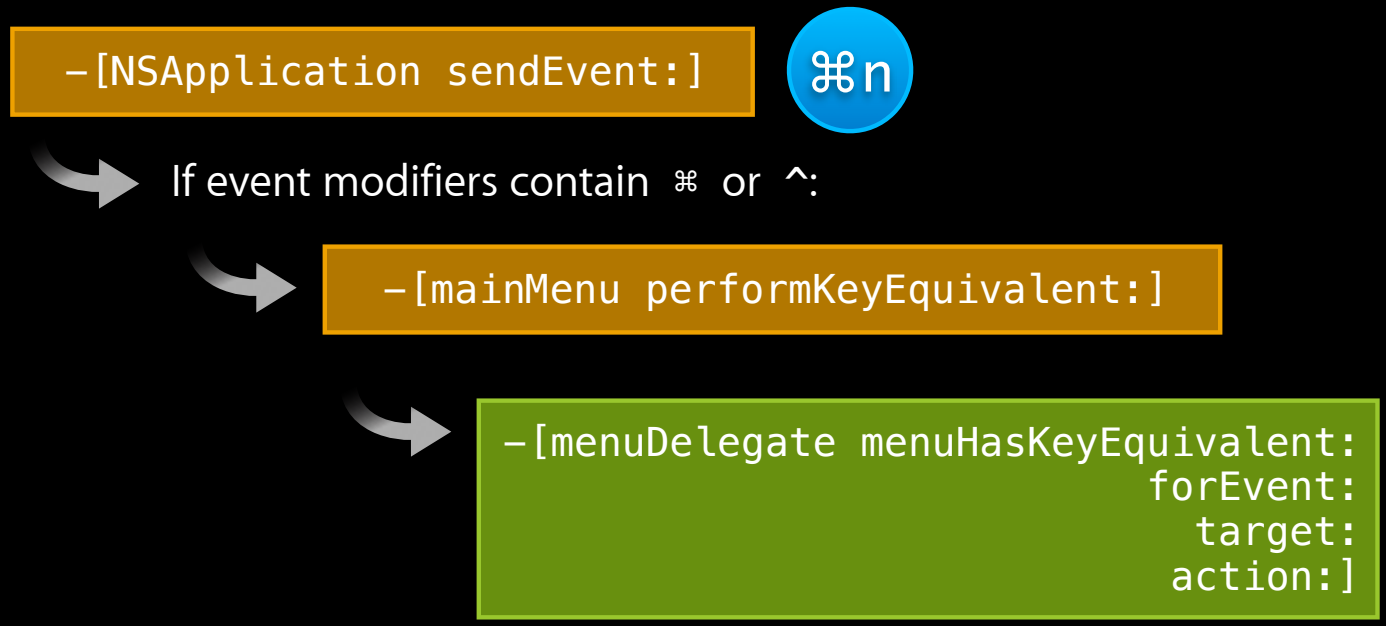
- Local event monitors get first crack at the event
- Can ignore it, handle it, or return a different event

Event Flow



- Stops when any `performKeyEquivalent:` returns YES

Event Flow

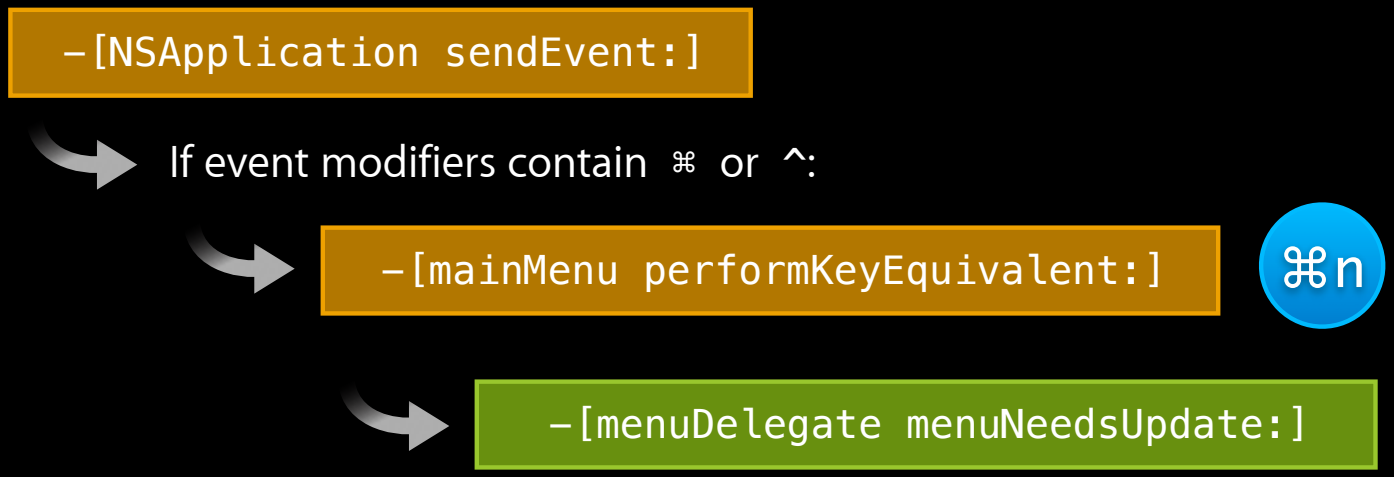

-[NSApplication sendEvent:]
⌘n
If event modifiers contain ⌘ or ^:
-[mainMenu performKeyEquivalent:]
-[menuDelegate menuHasKeyEquivalent:
forEvent:
target:
action:]

Event Flow

```
-[menuDelegate menuHasKeyEquivalent:
                            forEvent:
                              target:
                              action:]
```

- Not a very useful delegate method
  - Key equivalents don't appear in the menu
  - Doesn't get called if the menu is open
  - Doesn't work with user key equivalents
- One good use:
  - Suppress expensive population for menus without KEs
  - Just return NO

Event Flow



- Dynamically populate the menu with items and their key equivalents
- Item with a matching key equivalent is invoked

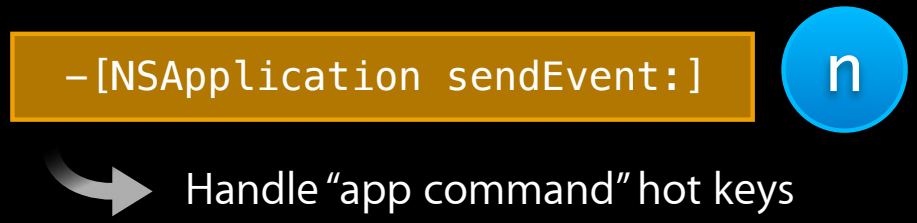
Event Flow

`-[NSApplication sendEvent:]`

- So far we've been assuming the event had ⌘ or ^

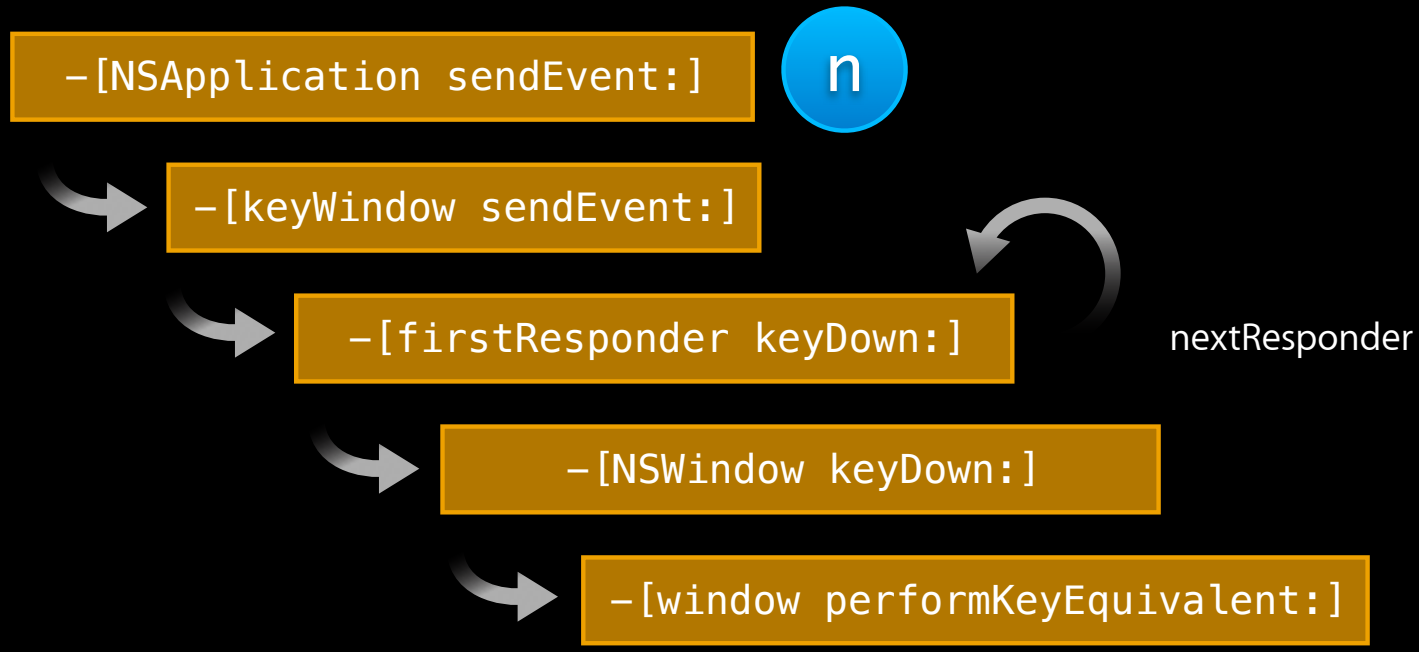
- What if it has no modifiers?
- Or what if every `performKeyEquivalent:` returned NO?

Event Flow



- Window cycling
- Focus management keys
- Other hot keys never get sent to the app

Event Flow


-[NSApplication sendEvent:]
n
-[keyWindow sendEvent:]
-[firstResponder keyDown:]
nextResponder
-[NSWindow keyDown:]
-[window performKeyEquivalent:]

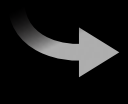
Event Flow

```
-[NSApplication sendEvent:]
```

```
-[mainMenu performKeyEquivalent:]
```

Event Flow

```
-[NSApplication sendEvent:]
```



```
Beep!
```

Override Points

- Windows handle key equivalents before the main menu
- Keyboard events with command or control try key equivalents first
- Other keyboard events try key equivalents last

Override Points

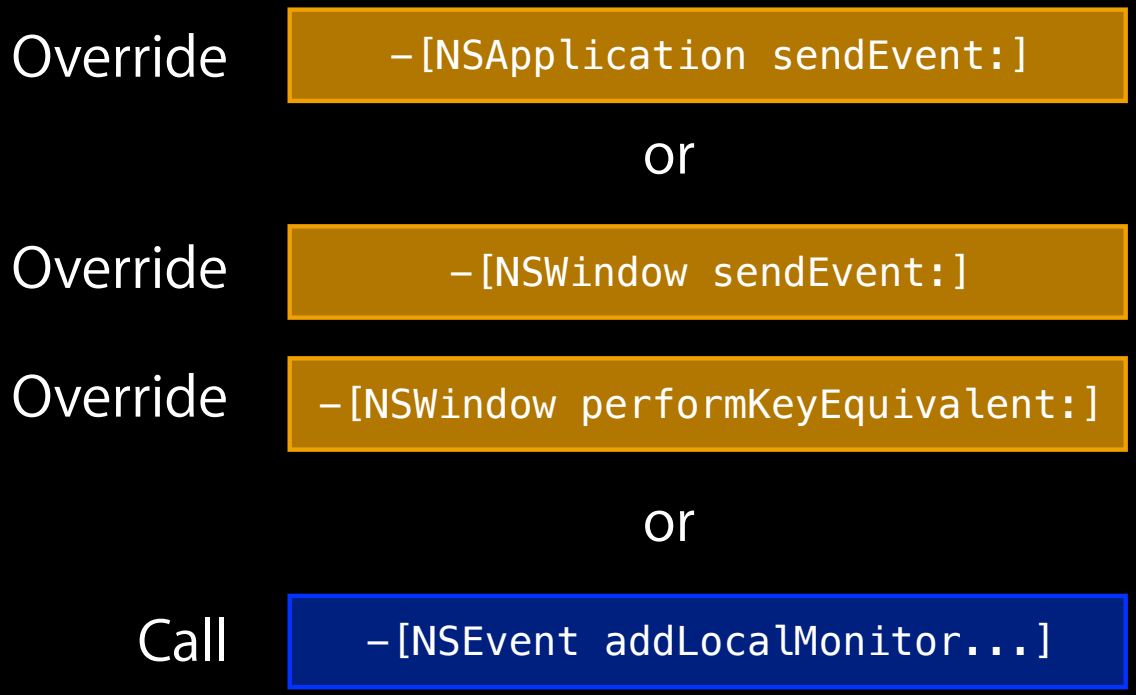
- Best practices

Implement `-[menuDelegate menuNeedsUpdate:]`

Override `-[view performKeyEquivalent:]`

Override Points

- Before normal key equivalent handling



Override Points

- After normal key equivalent handling

Override `-[NSWindow keyDown:]`

and call through to super

Key Equivalents and Menus

- Key equivalent basics
- Keyboard event flow and override points
- Handling key events from other apps
- Conflict handling
- Debugging conflicts
- Custom menu items
- Simulating menus

Handling Key Events from Other Apps

- Use a Services key equivalent

```
<key>NSExecutable</key>
<string>Stickies</string>
<key>NSKeyEquivalent</key>
<dict>
   <key>default</key>
   <string>Y</string>
</dict>
```

- Your app doesn't have to be running
- Users can customize it in System Preferences
- Normal app key equivalents have priority
- You can interact with the text selection
- `NSKeyEquivalent` key in the Service declaration

Handling Key Events from Other Apps

- Use a hot key
- Your app has to be running
- Always takes precedence
- Requires a virtual key code
  - See HIToolbox/Events.h
  - Or use `-[NSEvent keyCode]`
- `RegisterEventHotKey()` Carbon function
- Available in 64-bit

Handling Key Events from Other Apps

- Use a global event monitor
- Asynchronously responds to events
- Doesn't block them
- `+ (id)addGlobalMonitorForEventsMatchingMask:handler:`

Key Equivalents and Menus

- Key equivalent basics
- Keyboard event flow and override points
- Handling key events from other apps
- Conflict handling
- Debugging conflicts
- Custom menu items
- Simulating menus

Key Equivalents Conflicts

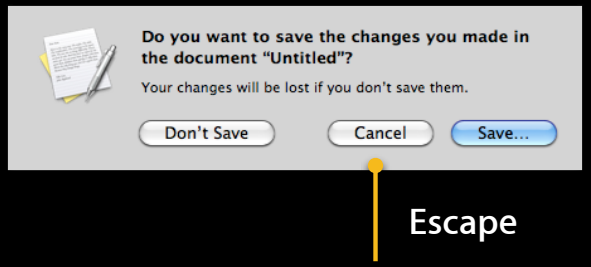
- Most KEs are always visible

Close ⌘W

- Some can be revealed

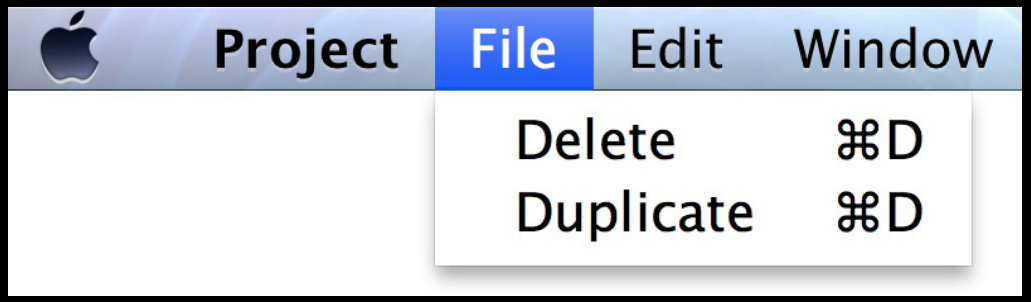
Close All ⌥⌘W

- Some are power user secrets



Key Equivalents Conflicts

- For visible key equivalents, what you see must be what you get



- `NSMenu` mediates key equivalent collisions

Key Equivalents Conflicts

- Key equivalents may come from outside your app
  - Services
  - User key equivalents
  - Plug-ins
  - Input managers
  - Future menu items added by AppKit (Close All)
- Key equivalents are first come, first serve
  - User key equivalents always win conflicts
  - Services always lose
- The `keyEquivalent` method returns the currently applicable key equivalent

Demo

Key Equivalents Conflicts

- Sometimes it's OK if two menu items share a key equivalent
- Menu items can show the same key equivalent if they have the same action
- …even if one is in a popup and the other is in the main menu

Key Equivalents and Menus

- Key equivalent basics
- Keyboard event flow and override points
- Handling key events from other apps
- Conflict handling
- Debugging conflicts
- Custom menu items
- Simulating menus

Debugging Conflicts

- Why doesn't my key equivalent show up?

```
call (void)[[NSMenuKEUniquer mainMenuUniquer] _printContents]
```

- Tells you about registered key equivalents and conflicts
- For debugging only!

Demo

Key Equivalents and Menus

- Key equivalent basics
- Keyboard event flow and override points
- Handling key events from other apps
- Conflict handling
- Debugging conflicts
- Custom menu items
- Simulating menus

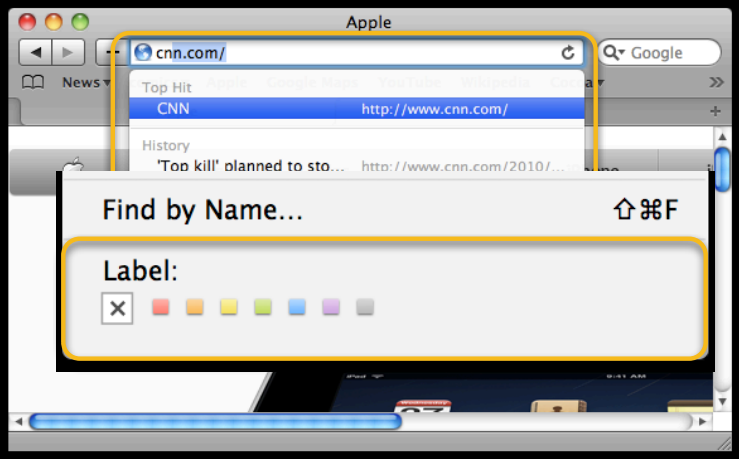
Key Equivalent Handling in Cocoa Applications

Part II—Custom Menus

Raleigh Ledet
Cocoa Frameworks Engineer

Custom Menus

- Custom menu items
- Simulating menus



Demo

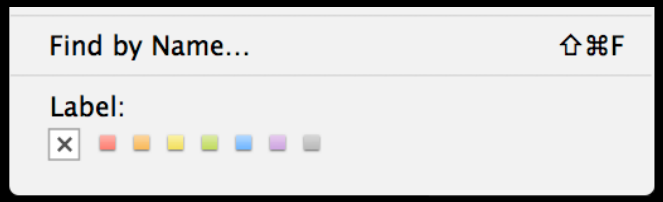
Custom Menus

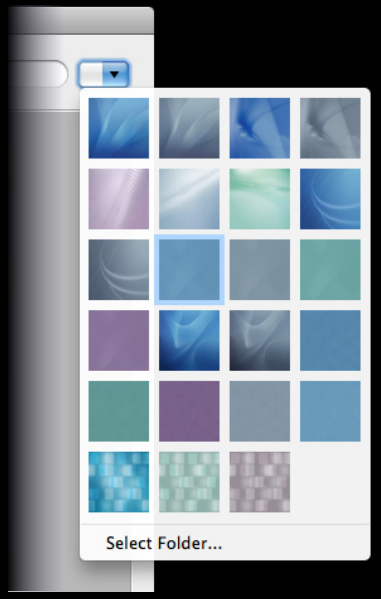
developer.apple.com/samplecode/CustomMenus_145

Key Equivalents and Menus

- Key equivalent basics
- Keyboard event flow and override points
- Handling key events from other apps
- Conflict handling
- Debugging conflicts
- Custom menu items
- Simulating menus

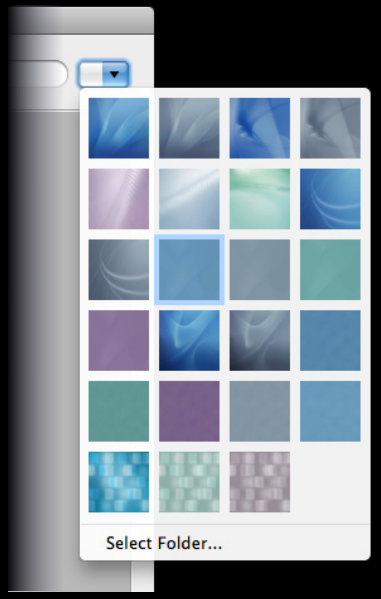
Custom View in NSMenuItem





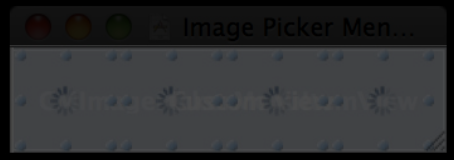
Custom View in NSMenuItem

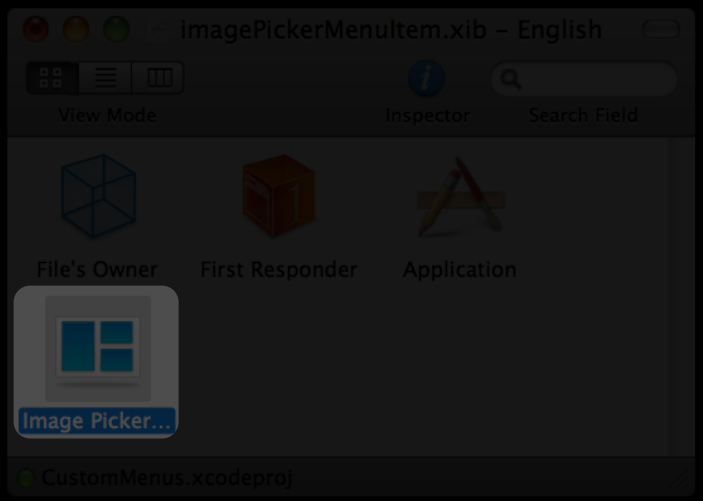
- Setting menu item view
- Mouse navigation
- Keyboard navigation
- Animations

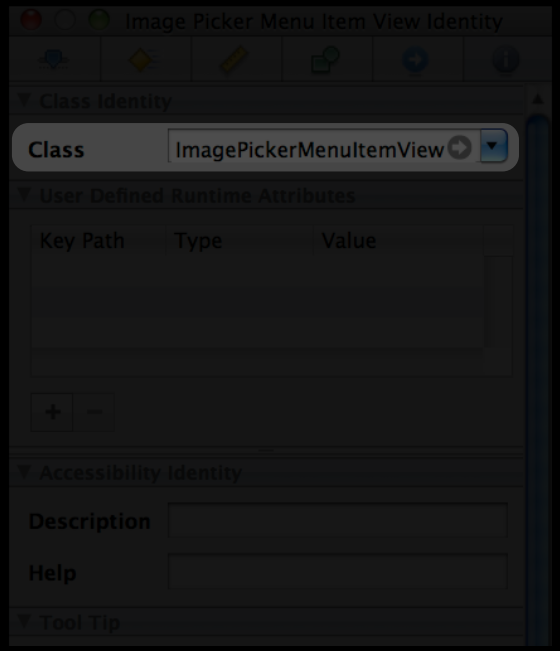


Custom View in NSMenuItem

Setting menu item view

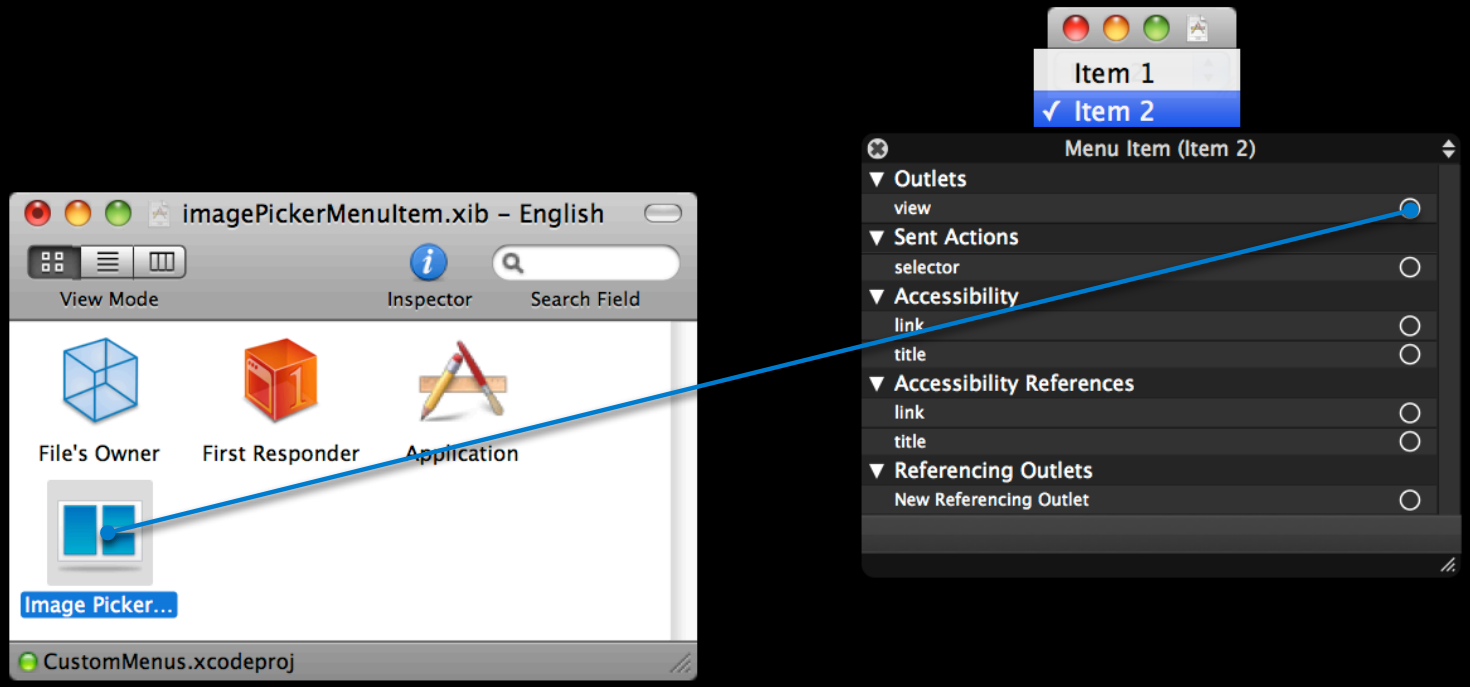






Custom View in NSMenuItem

Setting menu item view



Custom View in NSMenuItem

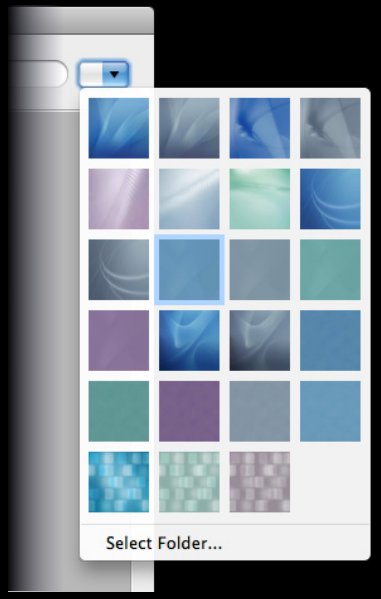
Setting menu item view

CustomMenusAppDelegate.m: -(void)setupImagesMenu

```
NSViewController *viewController = [[NSViewController alloc]
                                initWithNibName:@"imagePickerMenuItem" bundle:nil];

[menuItem setView:viewController.view];
```

Custom View in NSMenuItem

- Setting menu item view
- Mouse navigation
- Keyboard navigation
- Animations



Custom View in NSMenuItem

Mouse Navigation

```
@implementation ImagePickerMenuItemView

- (void)updateTrackingAreas {
    // Remove existing tracking areas

    for (NSInteger index = 0; index < _imageViews.count; index++) {
        NSTrackingArea *trackingArea = [self trackingAreaForIndex:index];
        [self addTrackingArea:trackingArea];
        ...
    }
}
```

Custom View in NSMenuItem

Mouse Navigation

```
@implementation ImagePickerMenuItemView

- (void)updateTrackingAreas {
  // Remove existing tracking areas

  for (NSInteger index = 0; index < _imageViews.count; index++) {
      NSTrackingArea *trackingArea = [self trackingAreaForIndex:index];
      [self addTrackingArea:trackingArea];
      ...
  }
}
```

Custom View in NSMenuItem

Mouse Navigation

```
@implementation ImagePickerMenuItemView

- (id)trackingAreaForIndex:(NSInteger)index {
    NSDictionary *trackerData = [NSDictionary dictionaryWithObjectsAndKeys:
                                 [NSNumber numberWithInteger:index], kTrackerKey, nil];
    NSView *view = [_imageViews objectAtIndex:index];
    NSRect trackingRect = [self convertRect:view.bounds fromView:view];
    NSTrackingAreaOptions trackingOptions = NSTrackingEnabledDuringMouseDrag |
                NSTrackingMouseEnteredAndExited | NSTrackingActiveInActiveApp;
    NSTrackingArea *trackingArea = [[NSTrackingArea alloc]
                initWithRect:trackingRect options:trackingOptions owner:self
                userInfo:trackerData];
    ...
```

Custom View in NSMenuItem

Mouse Navigation

```
@implementation ImagePickerMenuItemView

- (id)trackingAreaForIndex:(NSInteger)index {
    NSDictionary *trackerData = [NSDictionary dictionaryWithObjectsAndKeys:
                        [NSNumber numberWithInteger:index], kTrackerKey, nil];
    NSView *view = [_imageViews objectAtIndex:index];
    NSRect trackingRect = [self convertRect:view.bounds fromView:view];
    NSTrackingAreaOptions trackingOptions = NSTrackingEnabledDuringMouseDrag |
                NSTrackingMouseEnteredAndExited | NSTrackingActiveInActiveApp;
    NSTrackingArea *trackingArea = [[NSTrackingArea alloc]
                    initWithRect:trackingRect options:trackingOptions owner:self
                    userInfo:trackerData];
    ...
```

Custom View in NSMenuItem

Mouse Navigation

```
@implementation ImagePickerMenuItemView

- (id)trackingAreaForIndex:(NSInteger)index {
    NSDictionary *trackerData = [NSDictionary dictionaryWithObjectsAndKeys:
                                 [NSNumber numberWithInteger:index], kTrackerKey, nil];
    NSView *view = [_imageViews objectAtIndex:index];
    NSRect trackingRect = [self convertRect:view.bounds fromView:view];
    NSTrackingAreaOptions trackingOptions = NSTrackingEnabledDuringMouseDrag |
            NSTrackingMouseEnteredAndExited | NSTrackingActiveInActiveApp;
    NSTrackingArea *trackingArea = [[NSTrackingArea alloc]
                initWithRect:trackingRect options:trackingOptions owner:self
                userInfo:trackerData];
    ...
```

Custom View in NSMenuItem

Mouse Navigation

```
@implementation ImagePickerMenuItemView

- (id)trackingAreaForIndex:(NSInteger)index {
    NSDictionary *trackerData = [NSDictionary dictionaryWithObjectsAndKeys:
                        [NSNumber numberWithInteger:index], kTrackerKey, nil];
    NSView *view = [_imageViews objectAtIndex:index];
    NSRect trackingRect = [self convertRect:view.bounds fromView:view];
    NSTrackingAreaOptions trackingOptions = NSTrackingEnabledDuringMouseDrag |
                NSTrackingMouseEnteredAndExited | NSTrackingActiveInActiveApp;
    NSTrackingArea *trackingArea = [[NSTrackingArea alloc]
                    initWithRect:trackingRect options:trackingOptions owner:self
                    userInfo:trackerData];
    ...
```

Custom View in NSMenuItem

Mouse Navigation

```
@implementation ImagePickerMenuItemView
```

```
- (void)mouseEntered:(NSEvent *)event {
    NSInteger index = [[(NSDictionary*)[event userData]
                                        objectForKey:kTrackerKey] integerValue];
    self.selectedIndex = index;
}
```

```
- (void)mouseExited:(NSEvent *)event {
    self.selectedIndex = kNoSelection;
}
```

Custom View in NSMenuItem

Mouse Navigation

```
@implementation ImagePickerMenuItemView

- (void)mouseEntered:(NSEvent *)event {
    NSInteger index = [[(NSDictionary*)[event userData]
                                        objectForKey:kTrackerKey] integerValue];
    self.selectedIndex = index;
}
```

```
- (void)mouseExited:(NSEvent *)event {
    self.selectedIndex = kNoSelection;
}
```

Custom View in NSMenuItem

Mouse Navigation

```
@implementation ImagePickerMenuItemView
- (void)mouseUp:(NSEvent *)event {
    [self sendAction];
}

- (void)sendAction {
    NSMenuItem *actualMenuItem = [self enclosingMenuItem];
    [NSApp sendAction:[actualMenuItem action] to:[actualMenuItem target]
            from:self];

    NSMenu *menu = [actualMenuItem menu];
    [menu cancelTracking];
}
```

Custom View in NSMenuItem

Mouse Navigation

```
@implementation ImagePickerMenuItemView
- (void)mouseUp:(NSEvent *)event {
    [self sendAction];
}

- (void)sendAction {
    NSMenuItem *actualMenuItem = [self enclosingMenuItem];
    [NSApp sendAction:[actualMenuItem action] to:[actualMenuItem target]
           from:self];

    NSMenu *menu = [actualMenuItem menu];
    [menu cancelTracking];
}
```

Custom View in NSMenuItem

Mouse Navigation

```
@implementation ImagePickerMenuItemView
- (void)mouseUp:(NSEvent *)event {
    [self sendAction];
}

- (void)sendAction {
    NSMenuItem *actualMenuItem = [self enclosingMenuItem];
    [NSApp sendAction:[actualMenuItem action] to:[actualMenuItem target]
           from:self];

    NSMenu *menu = [actualMenuItem menu];
    [menu cancelTracking];
}
```

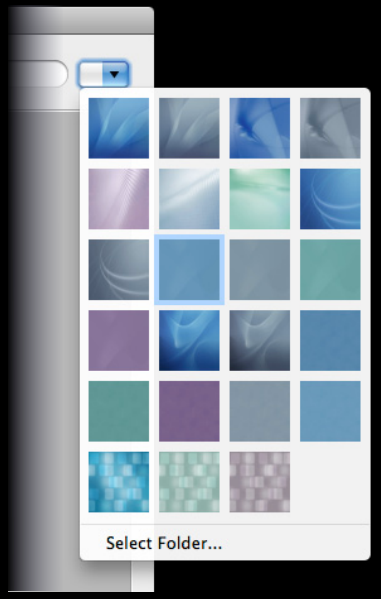
Custom View in NSMenuItem

Mouse Navigation

```
@implementation ImagePickerMenuItemView
- (void)mouseUp:(NSEvent *)event {
    [self sendAction];
}

- (void)sendAction {
    NSMenuItem *actualMenuItem = [self enclosingMenuItem];
    [NSApp sendAction:[actualMenuItem action] to:[actualMenuItem target]
            from:self];

    NSMenu *menu = [actualMenuItem menu];
    [menu cancelTracking];
}
```

Custom View in NSMenuItem

- Setting menu item view
- Mouse navigation
- Keyboard navigation
- Animations



Demo

Custom Menus

developer.apple.com/samplecode/CustomMenus_145

Custom View in NSMenuItem

Keyboard Navigation

```
@implementation ImagePickerMenuItemView
- (BOOL)acceptsFirstResponder {
    return YES;
}

- (BOOL)becomeFirstResponder {
    if (self.selectedIndex == kNoSelection) {
        self.selectedIndex = 0;
    }

    return YES;
}
```

Custom View in NSMenuItem

Keyboard Navigation

```
@implementation ImagePickerMenuItemView
- (BOOL)acceptsFirstResponder {
    return YES;
}
```

```
- (BOOL)becomeFirstResponder {
    if (self.selectedIndex == kNoSelection) {
        self.selectedIndex = 0;
    }

    return YES;
}
```

Custom View in NSMenuItem

Keyboard Navigation

```
@implementation ImagePickerMenuItemView
- (BOOL)resignFirstResponder {
    self.selectedIndex = kNoSelection;
    return YES;
}

- (void)keyDown:(NSEvent *)event {
    [self interpretKeyEvents:[NSArray arrayWithObject:event]];
    [super keyDown:event];
}
```

Custom View in NSMenuItem

Keyboard Navigation

```
@implementation ImagePickerMenuItemView
- (BOOL)resignFirstResponder {
    self.selectedIndex = kNoSelection;
    return YES;
}
```

```
- (void)keyDown:(NSEvent *)event {
    [self interpretKeyEvents:[NSArray arrayWithObject:event]];
    [super keyDown:event];
}
```

Custom View in NSMenuItem

Keyboard Navigation

```
@implementation ImagePickerMenuItemView
- (void)moveRight:(id)sender { ...
}
- (void)moveLeft:(id)sender { ...
}
- (void)moveToBeginningOfLine:(id)sender { ...
}
- (void)moveToEndOfLine:(id)sender { ...
}
```

Custom View in NSMenuItem

Keyboard Navigation

```
@implementation ImagePickerMenuItemView
- (void)moveRight:(id)sender { ...
}
- (void)moveLeft:(id)sender { ...
}
- (void)moveToBeginningOfLine:(id)sender { ...
}
- (void)moveToEndOfLine:(id)sender { ...
}
```

Custom View in NSMenuItem

Keyboard Navigation

```
@implementation ImagePickerMenuItemView
- (void)insertNewline:(id)sender {
    [self sendAction];
}

- (void)insertText:(id)insertString {
    if ([insertString isEqualToString:@" "]) {
        [self sendAction];
    }
}
```

Custom View in NSMenuItem

Keyboard Navigation

```
@implementation ImagePickerMenuItemView
- (void)insertNewline:(id)sender {
    [self sendAction];
}
```

```
- (void)insertText:(id)insertString {
    if ([insertString isEqualToString:@" "]) {
        [self sendAction];
    }
}
```

Custom View in NSMenuItem

Keyboard Navigation

```
@implementation ImagePickerMenuItemView
- (void)doCommandBySelector:(SEL)selector {
    if (   selector == @selector(moveRight:)
        || selector == @selector(moveLeft:)
        || selector == @selector(moveToBeginningOfLine:)
        || selector == @selector(moveToEndOfLine:)
        || selector == @selector(insertNewline:) )
    {
        [super doCommandBySelector:selector];
    }
    // Don't call super to prevent the system beep
}
```

Custom View in NSMenuItem

Keyboard Navigation

```
@implementation ImagePickerMenuItemView
- (void)doCommandBySelector:(SEL)selector {
    if (   selector == @selector(moveRight:)
        || selector == @selector(moveLeft:)
        || selector == @selector(moveToBeginningOfLine:)
        || selector == @selector(moveToEndOfLine:)
        || selector == @selector(insertNewline:) )
    {
        [super doCommandBySelector:selector];
    }
    // Don't call super to prevent the system beep
}
```

Custom View in NSMenuItem

Keyboard Navigation

```
@implementation ImagePickerMenuItemView
- (void)doCommandBySelector:(SEL)selector {
    if (   selector == @selector(moveRight:)
        || selector == @selector(moveLeft:)
        || selector == @selector(moveToBeginningOfLine:)
        || selector == @selector(moveToEndOfLine:)
        || selector == @selector(insertNewline:) )
    {
        [super doCommandBySelector:selector];
    }
    // Don't call super to prevent the system beep
}
```

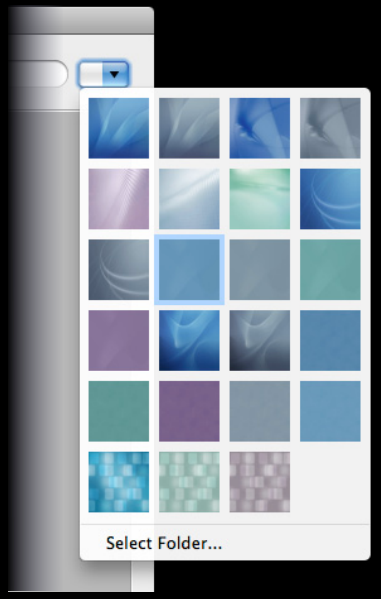
Custom View in NSMenuItem

Keyboard Navigation

```
@implementation ImagePickerMenuItemView
- (void)doCommandBySelector:(SEL)selector {
    if (   selector == @selector(moveRight:)
        || selector == @selector(moveLeft:)
        || selector == @selector(moveToBeginningOfLine:)
        || selector == @selector(moveToEndOfLine:)
        || selector == @selector(insertNewline:) )
    {
        [super doCommandBySelector:selector];
    }
    // Don't call super to prevent the system beep
}
```

Custom View in NSMenuItem

- Setting menu item view
- Mouse navigation
- Keyboard navigation
- Animations



Custom View in NSMenuItem

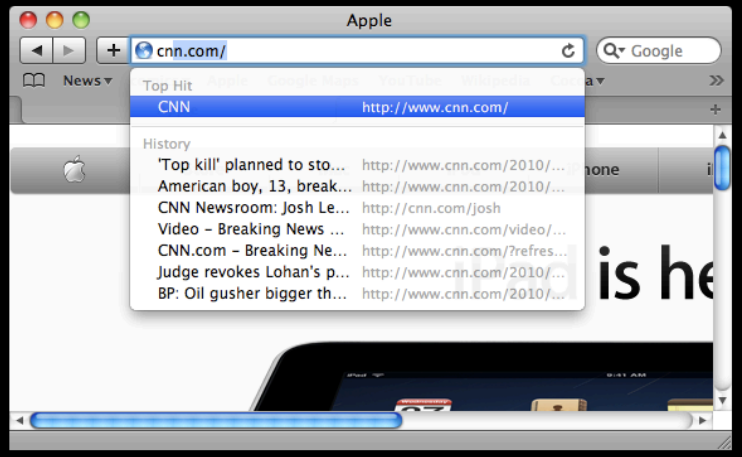
Keyboard Navigation

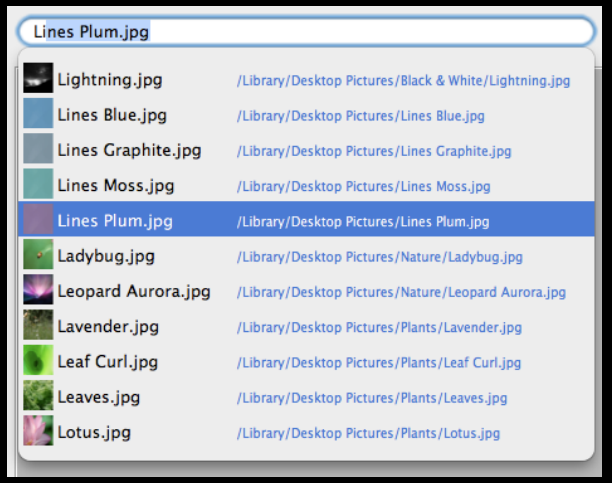
```
@implementation ImagePickerMenuItemView
- (void)viewDidMoveToWindow {
    if (self.window) {
        // start any animations here
    } else {
        // Terminate any animations
    }
}
```

Key Equivalents and Menus

- Key equivalent basics
- Keyboard event flow and override points
- Handling key events from other apps
- Conflict handling
- Debugging conflicts
- Custom menu items
- Simulating menus

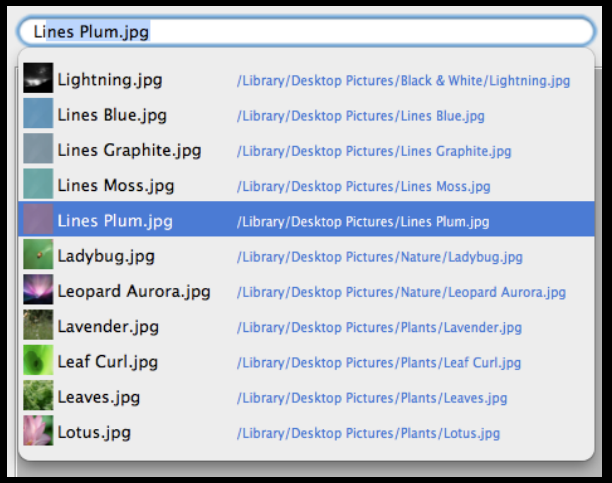
Suggestion Window





Suggestion Window

- Borderless window
- Add suggestion window as child of the parent window
- Activation
- Autocanceling
- Tracking
- Accessibility



Suggestion Window

Border window

```
@implementation SuggestionsWindow
- (id)initWithContentRect:(NSRect)contentRect styleMask:(NSUInteger)aStyle
      backing:(NSBackingStoreType)bufferingType defer:(BOOL)flag {

    self = [super initWithContentRect:contentRect
              styleMask:NSBorderlessWindowMask
              backing:bufferingType defer:flag];

    if (self) {
        [self setHasShadow:YES];
        [self setBackgroundColor:[NSColor clearColor]];
        [self setOpaque:NO];
    }

    return self;
}
```

Suggestion Window

Border window

```
@implementation SuggestionsWindow
- (id)initWithContentRect:(NSRect)contentRect styleMask:(NSUInteger)aStyle
      backing:(NSBackingStoreType)bufferingType defer:(BOOL)flag {

    self = [super initWithContentRect:contentRect
              styleMask:NSBorderlessWindowMask
              backing:bufferingType defer:flag];

    if (self) {
        [self setHasShadow:YES];
        [self setBackgroundColor:[NSColor clearColor]];
        [self setOpaque:NO];
    }

    return self;
}
```

Suggestion Window

Border window

```
@implementation SuggestionsWindow
- (id)initWithContentRect:(NSRect)contentRect styleMask:(NSUInteger)aStyle
      backing:(NSBackingStoreType)bufferingType defer:(BOOL)flag {

    self = [super initWithContentRect:contentRect
              styleMask:NSBorderlessWindowMask
              backing:bufferingType defer:flag];

    if (self) {
        [self setHasShadow:YES];
        [self setBackgroundColor:[NSColor clearColor]];
        [self setOpaque:NO];
    }

    return self;
}
```

Suggestion Window

Border window

```
@implementation SuggestionsWindow
- (id)initWithContentRect:(NSRect)contentRect styleMask:(NSUInteger)aStyle
      backing:(NSBackingStoreType)bufferingType defer:(BOOL)flag {

    self = [super initWithContentRect:contentRect
              styleMask:NSBorderlessWindowMask
              backing:bufferingType defer:flag];

    if (self) {
        [self setHasShadow:YES];
        [self setBackgroundColor:[NSColor clearColor]];
        [self setOpaque:NO];
    }

    return self;
}
```

Suggestion Window

Border window

```
@implementation SuggestionsWindow
- (id)initWithContentRect:(NSRect)contentRect styleMask:(NSUInteger)aStyle
      backing:(NSBackingStoreType)bufferingType defer:(BOOL)flag {

    self = [super initWithContentRect:contentRect
              styleMask:NSBorderlessWindowMask
              backing:bufferingType defer:flag];

    if (self) {
        [self setHasShadow:YES];
        [self setBackgroundColor:[NSColor clearColor]];
        [self setOpaque:NO];
    }

    return self;
}
```

Suggestion Window

Border window

```
@implementation SuggestionsWindow
- (id)initWithContentRect:(NSRect)contentRect styleMask:(NSUInteger)aStyle
      backing:(NSBackingStoreType)bufferingType defer:(BOOL)flag {

    self = [super initWithContentRect:contentRect
              styleMask:NSBorderlessWindowMask
              backing:bufferingType defer:flag];

    if (self) {
        [self setHasShadow:YES];
        [self setBackgroundColor:[NSColor clearColor]];
        [self setOpaque:NO];
    }

    return self;
}
```

Suggestion Window

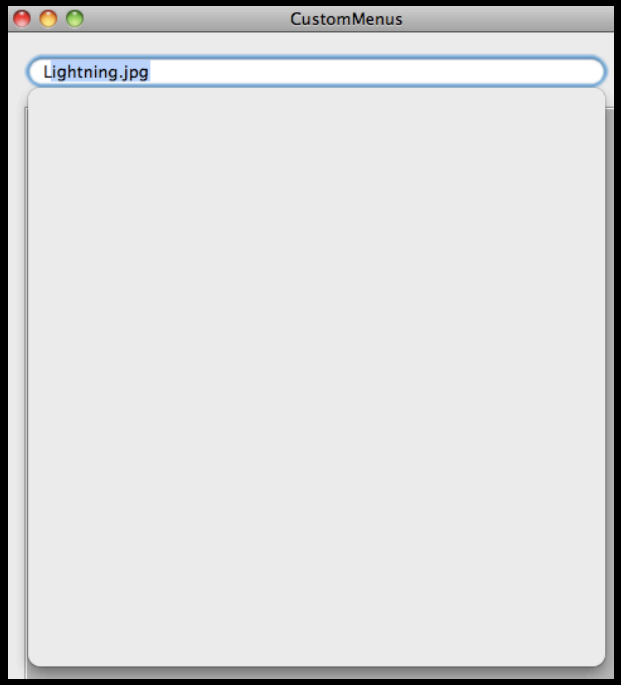
Border window

```
@implementation SuggestionsWindow
- (id)initWithContentRect:(NSRect)contentRect styleMask:(NSUInteger)aStyle
      backing:(NSBackingStoreType)bufferingType defer:(BOOL)flag {

    self = [super initWithContentRect:contentRect
              styleMask:NSBorderlessWindowMask
              backing:bufferingType defer:flag];

    if (self) {
        [self setHasShadow:YES];
        [self setBackgroundColor:[NSColor clearColor]];
        [self setOpaque:NO];
    }

    return self;
}
```

Suggestion Window

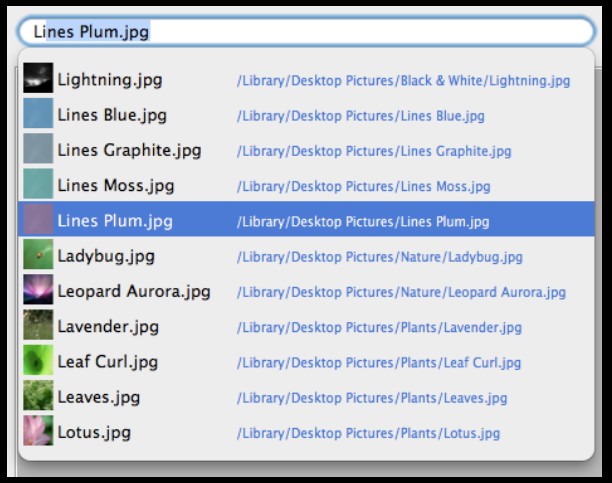
Border window

```
@implementation SuggestionsWindow
- (id)initWithContentRect:(NSRect)contentRect styleMask:(NSUInteger)aStyle
      backing:(NSBackingStoreType)bufferingType defer:(BOOL)flag {

    self = [super initWithContentRect:contentRect
              styleMask:NSBorderlessWindowMask
              backing:bufferingType defer:flag];

    if (self) {
        [self setHasShadow:YES];
        [self setBackgroundColor:[NSColor clearColor]];
        [self setOpaque:NO];
    }

    return self;
}
```



Suggestion Window

- Borderless window
- Add suggestion window as child of the parent window
- Activation
- Autocanceling
- Tracking
- Accessibility



Demo

Custom Menus

developer.apple.com/samplecode/CustomMenus_145

Suggestion Window

Add suggestion window as child of the parent window

SuggestionsWindowController.m: -(void)beginForTextField:

```
[parentWindow addChildWindow:suggestionWindow ordered:NSWindowAbove];
```

SuggestionsWindowController.m: -(void)cancelSuggestions

```
[parentWindow removeChildWindow:suggestionWindow];
[suggestionWindow orderOut:nil];
```

Suggestion Window

Add suggestion window as child of the parent window

SuggestionsWindowController.m: -(void)beginForTextField:

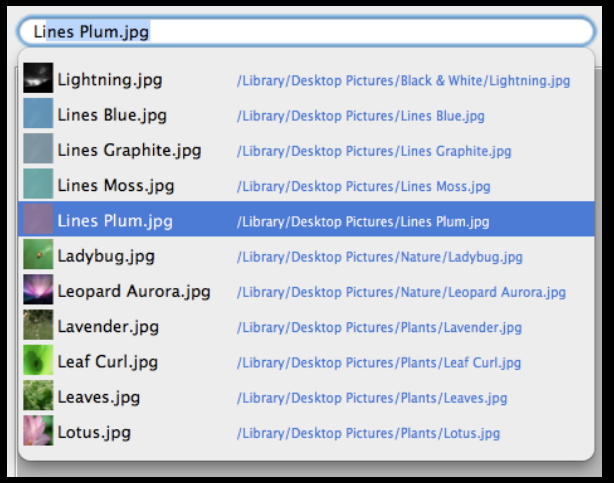
```
[parentWindow addChildWindow:suggestionWindow ordered:NSWindowAbove];
```

SuggestionsWindowController.m: -(void)cancelSuggestions

```
[parentWindow removeChildWindow:suggestionWindow];
[suggestionWindow orderOut:nil];
```

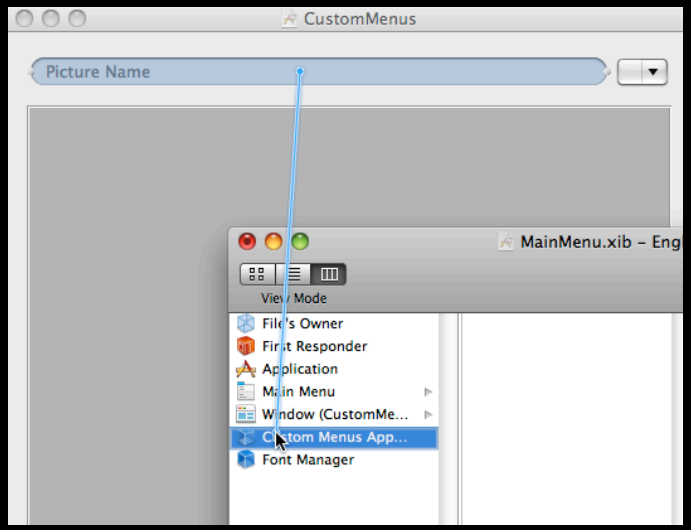
Suggestion Window

- Borderless window
- Add suggestion window as child of the parent window
- Activation
- Autocanceling
- Tracking
- Accessibility



Suggestion Window

Activation



Suggestion Window

Activation

```
@implementation CustomMenusAppDelegate <NSTextFieldDelegate>
-controlTextDidBeginEditing:
-controlTextDidChange:
-controlTextDidEndEditing:
-control:textView:doCommandBySelector:
```

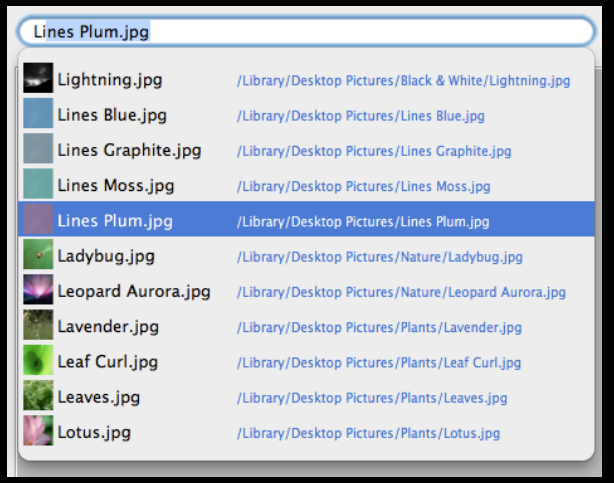
Suggestion Window

Activation

```
@implementation CustomMenusAppDelegate <NSTextFieldDelegate>
-controlTextDidBeginEditing:
-controlTextDidChange:
-controlTextDidEndEditing:
-control:textView:doCommandBySelector: { ...
    if (commandSelector == @selector(complete:)) {
        if ([_suggestionsController.window isVisible]) {
            [_suggestionsController cancelSuggestions];
        } else {
            [_suggestionsController beginForControl:control];
        }
        return YES;
    }
```

Suggestion Window

- Borderless window
- Add suggestion window as child of the parent window
- Activation
- Autocanceling
- Tracking
- Accessibility



Demo

Custom Menus

developer.apple.com/samplecode/CustomMenus_145

Suggestion Window

Autocanceling

Controlling the Mouse

SuggestionsWindowController.m: -(void)beginForTextField:

```
eventMask = NSLeftMouseDownMask | NSRightMouseDownMask | NSOtherMouseDownMask;
_localEventMonitor = [NSEvent addLocalMonitorForEventsMatchingMask:eventMask
                                                           handler:^(NSEvent *event) {
    if ([event window] != suggestionWindow) {
        if ([event window] != parentWindow) {
            [self cancelSuggestions];
        } else {
            NSView *contentView = [parentWindow contentView];
            NSPoint location = [contentView convertPoint:[event
                                                          locationInWindow] fromView:nil];
            NSView *hitView = [contentView hitTest:location];
```

Suggestion Window

Autocanceling

Controlling the Mouse

SuggestionsWindowController.m: -(void)beginForTextField:

```
eventMask = NSLeftMouseDownMask | NSRightMouseDownMask | NSOtherMouseDownMask;
_localEventMonitor = [NSEvent addLocalMonitorForEventsMatchingMask:eventMask
                                                           handler:^(NSEvent *event) {
    if ([event window] != suggestionWindow) {
        if ([event window] != parentWindow) {
            [self cancelSuggestions];
        } else {
            NSView *contentView = [parentWindow contentView];
            NSPoint location = [contentView convertPoint:[event
                                                          locationInWindow] fromView:nil];
            NSView *hitView = [contentView hitTest:location];
```

Suggestion Window

Autocanceling

Controlling the Mouse

SuggestionsWindowController.m: -(void)beginForTextField:

```
eventMask = NSLeftMouseDownMask | NSRightMouseDownMask | NSOtherMouseDownMask;
_localEventMonitor = [NSEvent addLocalMonitorForEventsMatchingMask:eventMask
                                                           handler:^(NSEvent *event) {
    if ([event window] != suggestionWindow) {
        if ([event window] != parentWindow) {
            [self cancelSuggestions];
        } else {
            NSView *contentView = [parentWindow contentView];
            NSPoint location = [contentView convertPoint:[event
                                                          locationInWindow] fromView:nil];
            NSView *hitView = [contentView hitTest:location];
```

Suggestion Window

Autocanceling

Controlling the Mouse

```
        } else {
            NSView *contentView = [parentWindow contentView];
            NSPoint location = [contentView convertPoint:[event
                                    locationInWindow] fromView:nil];
            NSView *hitView = [contentView hitTest:location];
            NSText *fie
            if (hitView != parentTextField && hitView != fieldEditor ) {
                event = nil;
                [self cancelSuggestions];
            }
        }
    }
```

Suggestion Window

Autocanceling

Controlling the Mouse

```
        } else {
            NSView *contentView = [parentWindow contentView];
            NSPoint location = [contentView convertPoint:[event
                                        locationInWindow] fromView:nil];
            NSView *hitView = [contentView hitTest:location];
            NSText *fieldEditor = [parentTextField currentEditor];
            if (hitView != parentTextField && hitView != fieldEditor ) {
                event = nil;
                [self cancelSuggestions];
            }
        }
    }
```

Suggestion Window

Autocanceling

Controlling the Mouse

```
        } else {
            NSView *contentView = [parentWindow contentView];
            NSPoint location = [contentView convertPoint:[event
                                        locationInWindow] fromView:nil];
            NSView *hitView = [contentView hitTest:location];
            NSText *fieldEditor = [parentTextField currentEditor];
            if (hitView != parentTextField && hitView != fieldEditor ) {
                    event = nil;
                    [self cancelSuggestions];
            }
        }
    }
```

Suggestion Window

Autocanceling

Controlling the Mouse

```
    return event;
}];
```

```
SuggestionsWindowController.m: -(void)cancelSuggestions
if (_localMouseDownEventMonitor) {
    [NSEvent removeMonitor:_localMouseDownEventMonitor];
    _localMouseDownEventMonitor = nil;
}
```

Suggestion Window

Autocanceling

Controlling the Mouse

```
    return event;
}];

SuggestionsWindowController.m: -(void)cancelSuggestions
if (_localMouseDownEventMonitor) {
    [NSEvent removeMonitor:_localMouseDownEventMonitor];
    _localMouseDownEventMonitor = nil;
}
```

Suggestion Window

Autocanceling

Controlling the Mouse

```
    return event;
}];

SuggestionsWindowController.m: -(void)cancelSuggestions
if (_localMouseDownEventMonitor) {
    [NSEvent removeMonitor:_localMouseDownEventMonitor];
    _localMouseDownEventMonitor = nil;
}
```

Suggestion Window

Autocanceling

Window Resigns Key Focus

SuggestionsWindowController.m: -(void)beginForTextField:

```
_lostFocusObserver = [[NSNotificationCenter defaultCenter]
                        addObserverForName:NSWindowDidResignKeyNotification
                        object:parentWindow queue:nil
                        usingBlock:^(NSNotification *arg1) {
        [self cancelSuggestions];
    }];
```

Suggestion Window

Autocanceling

Window Resigns Key Focus

SuggestionsWindowController.m: -(void)beginForTextField:

```
_lostFocusObserver = [[NSNotificationCenter defaultCenter]
                      addObserverForName:NSWindowDidResignKeyNotification
                      object:parentWindow queue:nil
                      usingBlock:^(NSNotification *arg1) {
    [self cancelSuggestions];
}];
```

Suggestion Window

Autocanceling

Window Resigns Key Focus

```
SuggestionsWindowController.m: -(void)beginForTextField:
_lostFocusObserver = [[NSNotificationCenter defaultCenter]
                        addObserverForName:NSWindowDidResignKeyNotification
                        object:parentWindow queue:nil
                        usingBlock:^(NSNotification *arg1) {
    [self cancelSuggestions];
}];
```

Suggestion Window

Autocanceling

Window Resigns Key Focus

SuggestionsWindowController.m: -(void)beginForTextField:

```
[[NSNotificationCenter defaultCenter] removeObserver:_lostFocusObserver];
_lostFocusObserver = nil;
```

Suggestion Window

Autocanceling

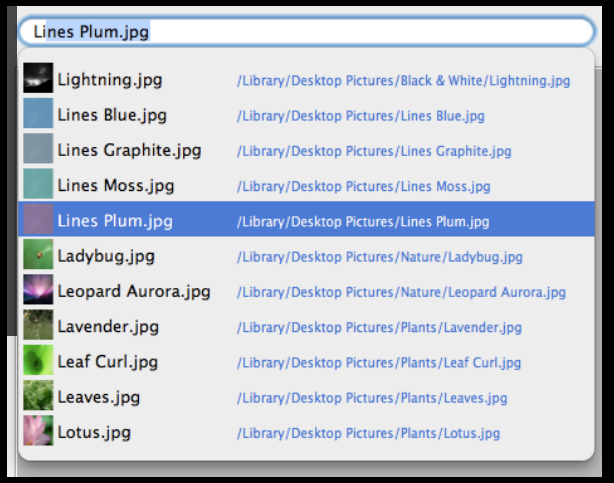
Window Resigns Key Focus

SuggestionsWindowController.m: -(void)beginForTextField:

```
[[NSNotificationCenter defaultCenter] removeObserver:_lostFocusObserver];
_lostFocusObserver = nil;
```

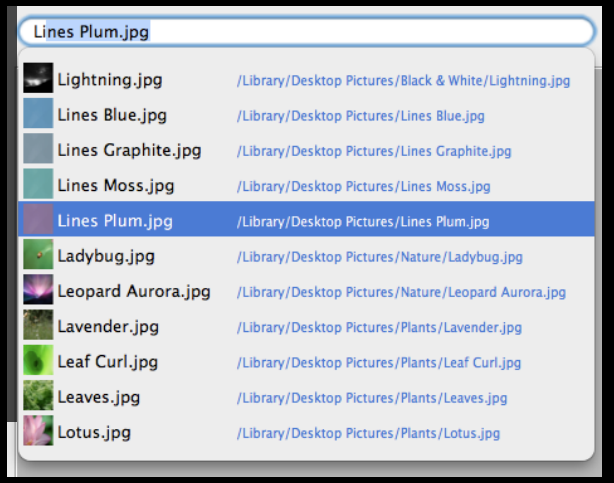
Suggestion Window

- Borderless window
- Add suggestion window as child of the parent window
- Activation
- Autocanceling
- Tracking
- Accessibility



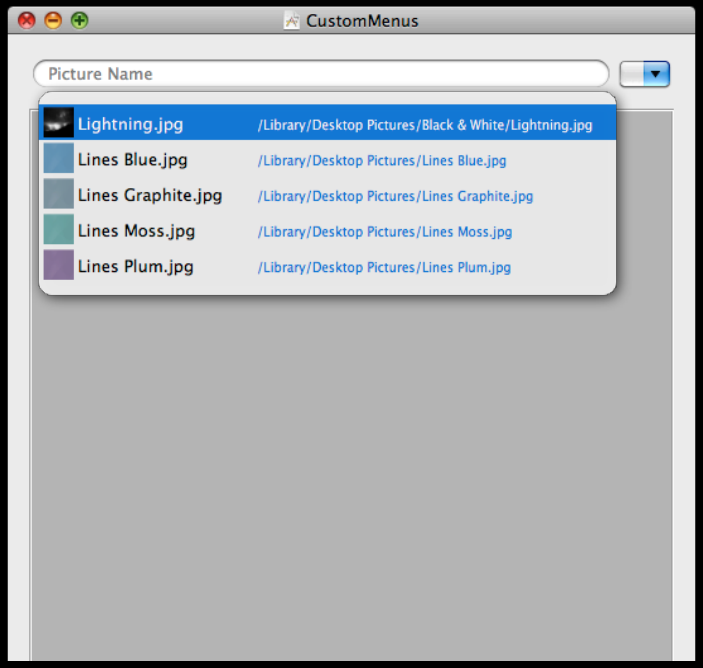
Suggestion Window

- Borderless window
- Add suggestion window as child of the parent window
- Activation
- Autocanceling
- Tracking
- Accessibility



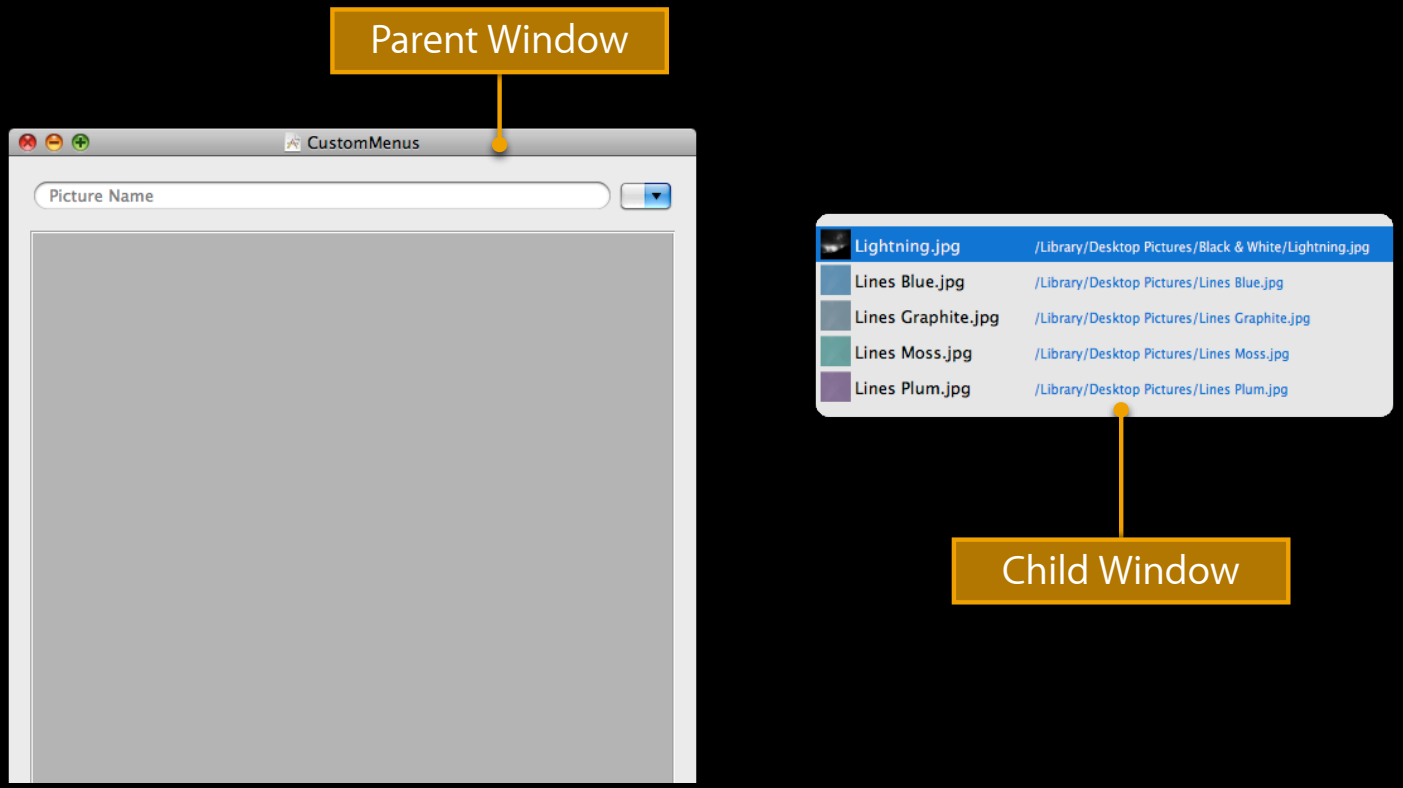
Suggestion Window

Accessibility



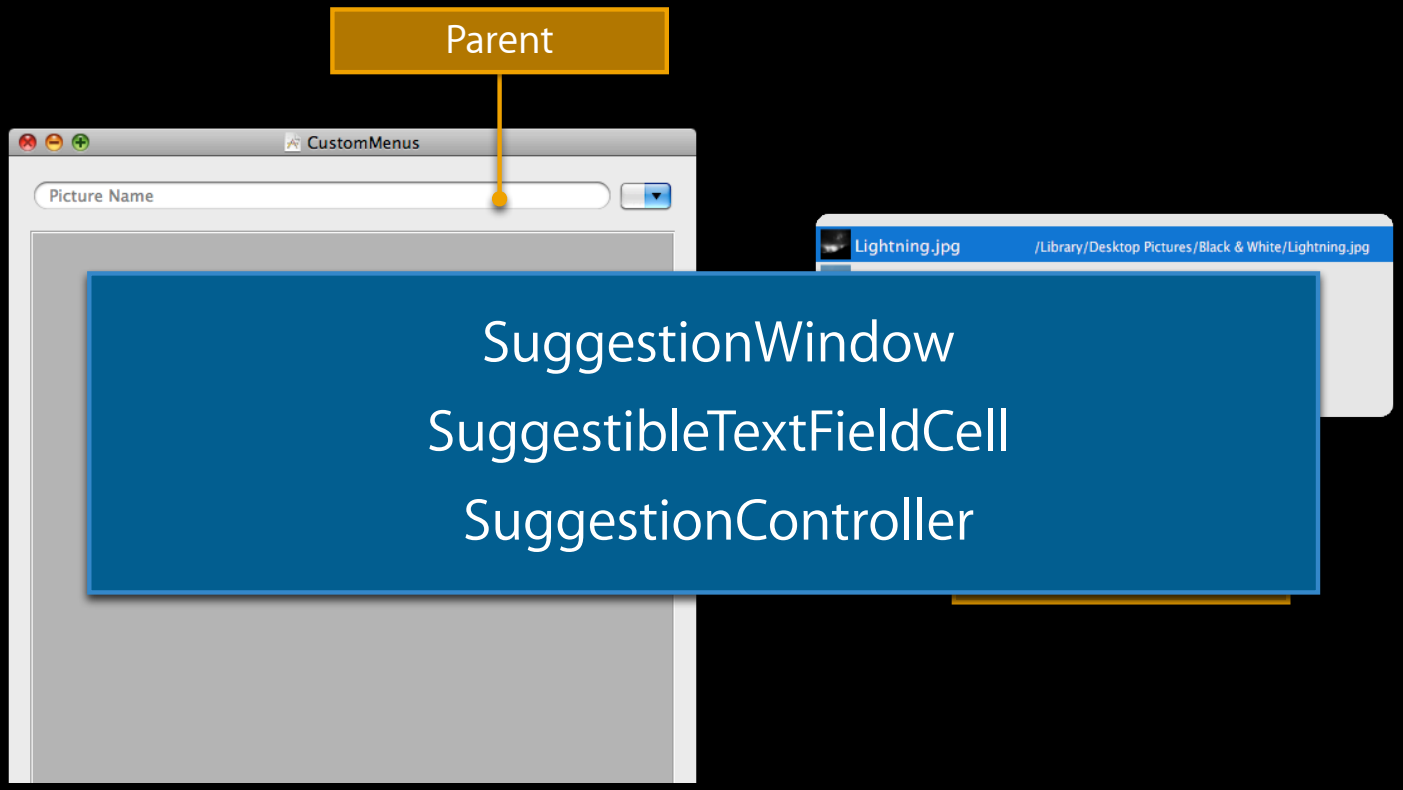
Suggestion Window

Accessibility



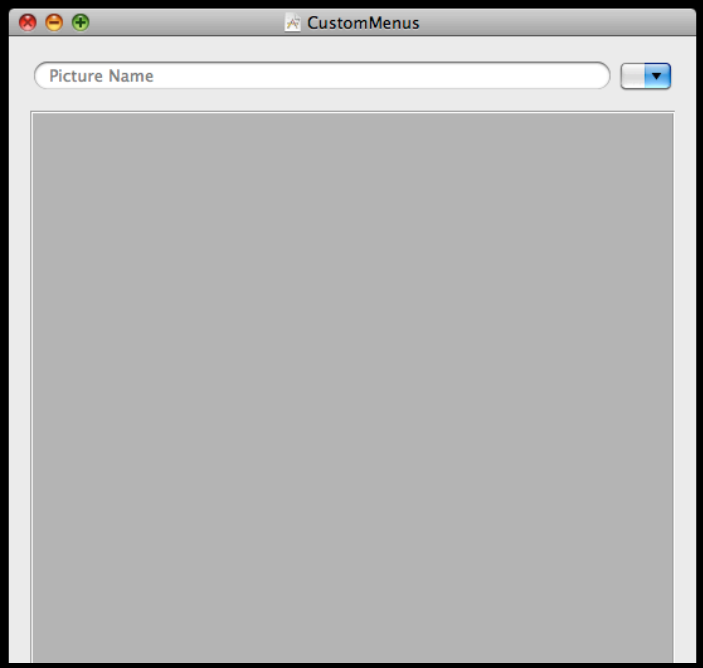
Suggestion Window

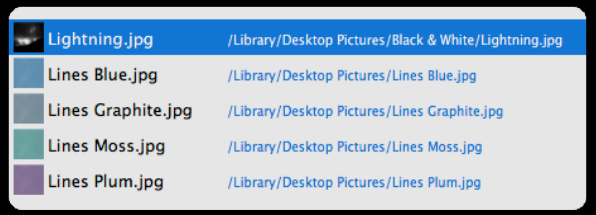
Accessibility



Suggestion Window

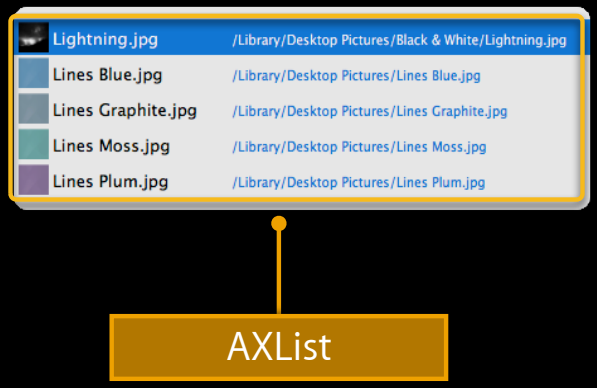
Accessibility





Suggestion Window

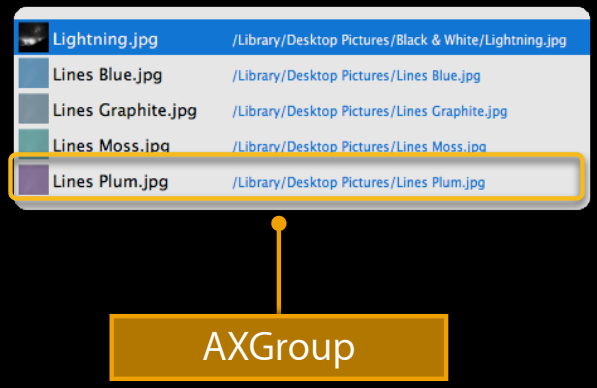
Accessibility



RoundedCornersView

Suggestion Window

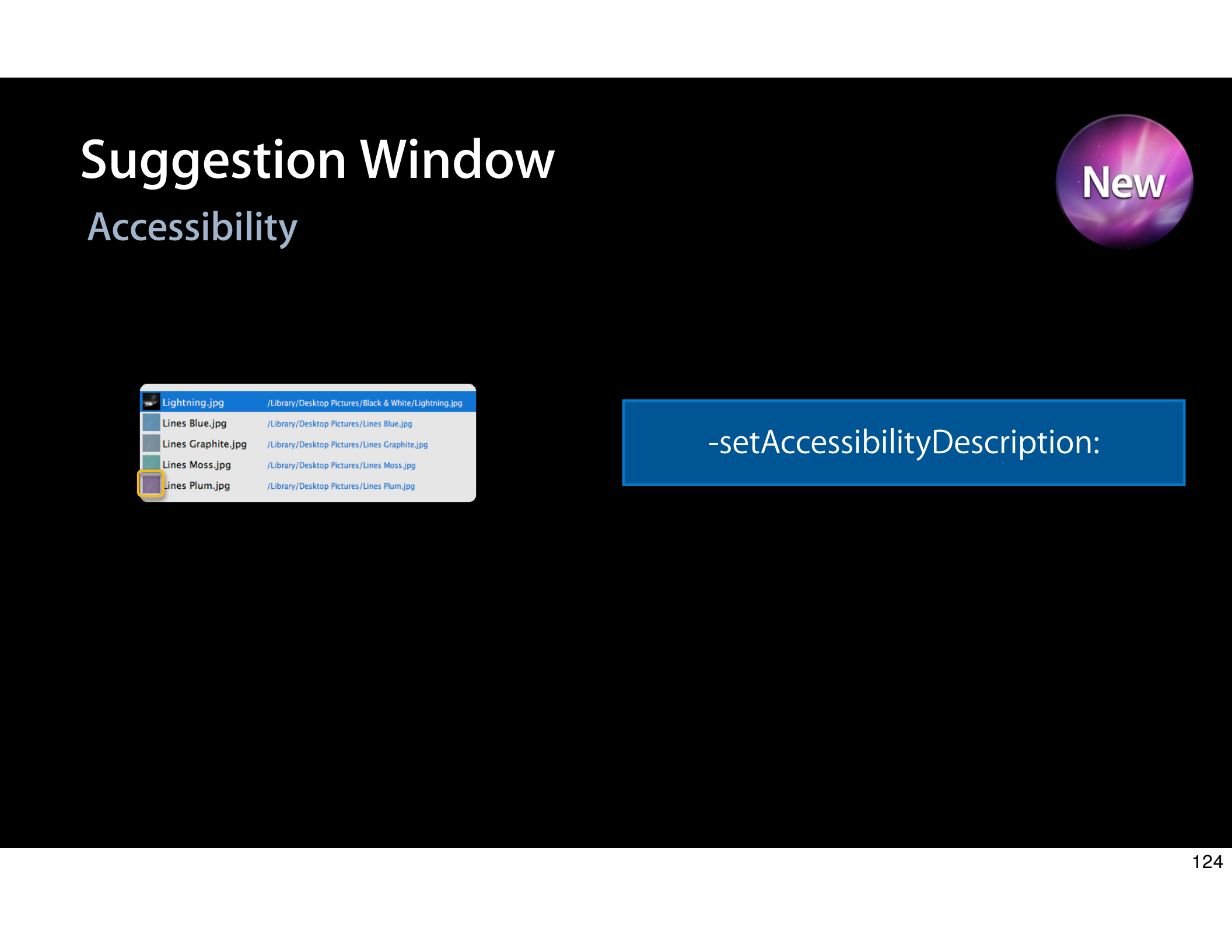
Accessibility



HighlightingView

Suggestion Window

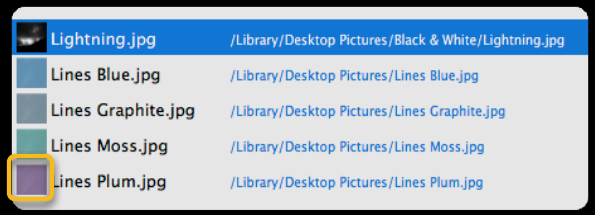
Accessibility



-setAccessibilityDescription:

Suggestion Window

Accessibility



New

Accessibility Descriptions strings

Summary

- Key equivalents accelerate using applications
- Implement dynamic key equivalents with overrides or delegates
- Conflicts are mediated by NSMenu
- Custom menu item views are easy
- Simulate menus when appropriate

More Information

Bill Dudney
Application Frameworks Evangelist
dudney@apple.com

Documentation
Mac OS X Human Interface Guidelines
http://developer.apple.com/ue

Apple Developer Forums
http://devforums.apple.com

Related Sessions

Cocoa Tips and Tricks	Marina Tuesday 2:00PM
Crafting Custom Cocoa Views	Russian Hill Friday 10:15AM

WWDC10