CS193E
Lecture 5

Loading Resources
Notifications
System Panels

Today's Topics

- Questions on the Favorite Things 1 assignment?
- Loading resources
  - Additional nib files
  - Arbitrary resources
- Notifications
- System Panels

Build Errors/Warnings

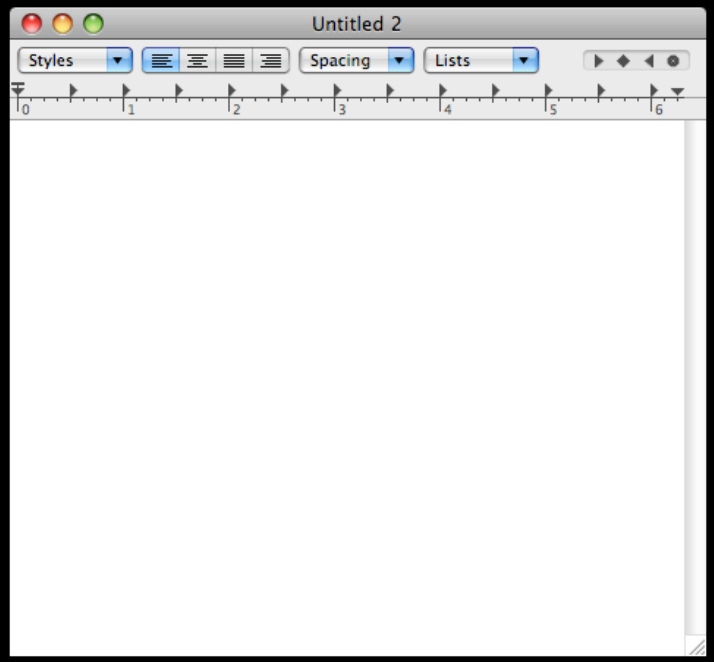
- The Debug configuration can sometimes mask build warnings/ errors with ZeroLink feature
  - *Build Clean*
    Build > Clean
  - *Use release configuration instead of debug*
    Project > Set Active Build Configuration > Release
  - *Build with new configuration*
    Build > Build

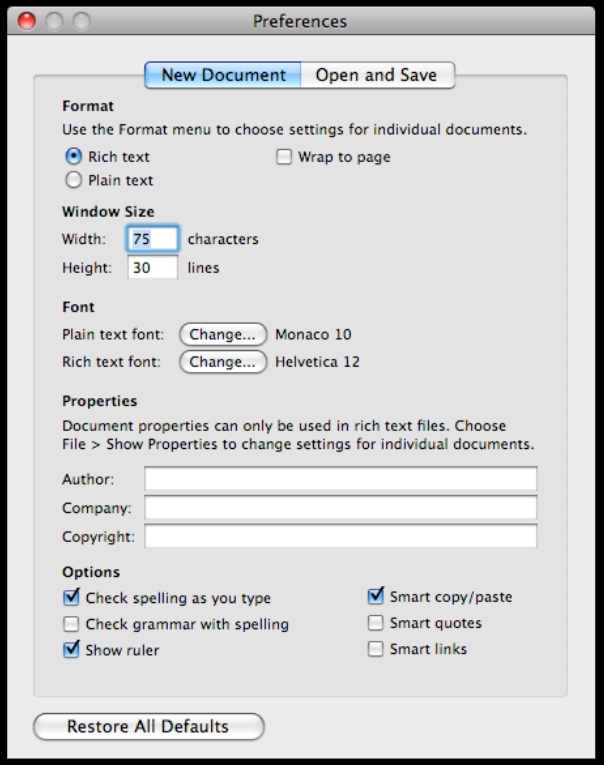
Demo

Favorite Things II

Resources and Multiple Nib Files

Applications use multiple components

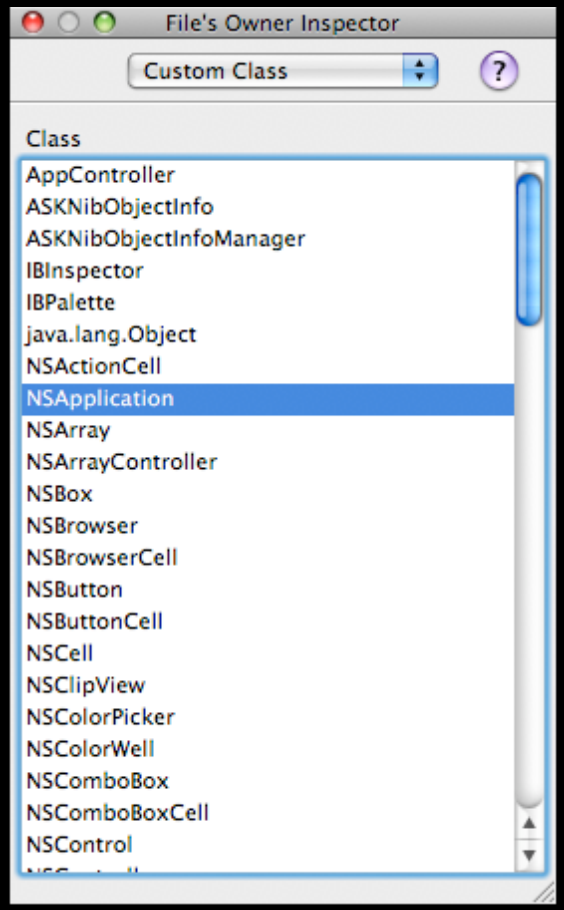




Implement using multiple nib files

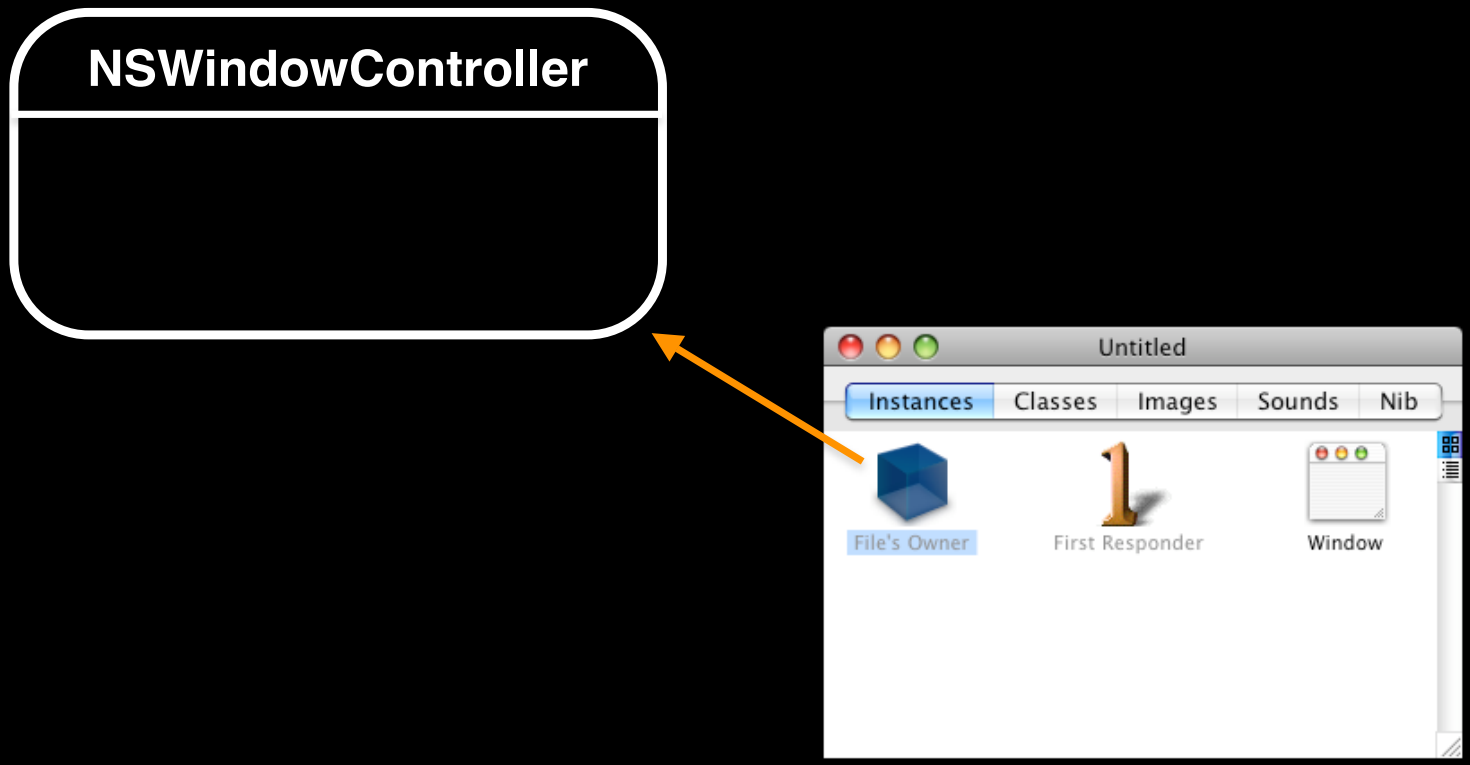
- Components—modular design and implementation
- Efficiency—lazy instantiations avoid unnecessary file I/O, resulting in decreased startup times
- Reuse and Replication
- Dynamic replacement

NSApplication loads and owns main nib



File's Owner lives outside the nib

Often is the object that loads the nib



NSWindowController

- A window controller
  - Manages a single window
  - Loads the window's nib file
  - Frees other top-level objects within the nib
- You typically subclass NSWindowController

NSWindowController methods

Frequently used

- `- (IBAction) showWindow: (id) sender`
- `- (NSWindow *) window`

Commonly overridden

- `- (NSString *) windowNibName`
- `- (void) windowWillLoad`
- `- (void) windowDidLoad`
- `- (NSString *) windowTitleForDocumentDisplayName: (NSString *) displayName`

Demo

Adding a second nib file
Loading a second window
Setting bundle identifier and icon

Dynamically-loaded resources

- Nib files
- Images and sounds
- Localized character strings
- Executable code and class implementations

NSBundle interface

```
+ (NSBundle *)mainBundle;

+ (NSBundle *)bundleWithPath:(NSString *) path;

+ (BOOL)loadNibNamed:(NSString *) name owner:
(id)owner;

- (NSString *)pathForResource:(NSString *) name
ofType:(NSString *)extension;

- (NSString *)localizedStringForKey:(NSString *) key
value:(NSString *)value
table:(NSString *)tableName;

- (Class)classNamed:(NSString *)className;
```

Notifications

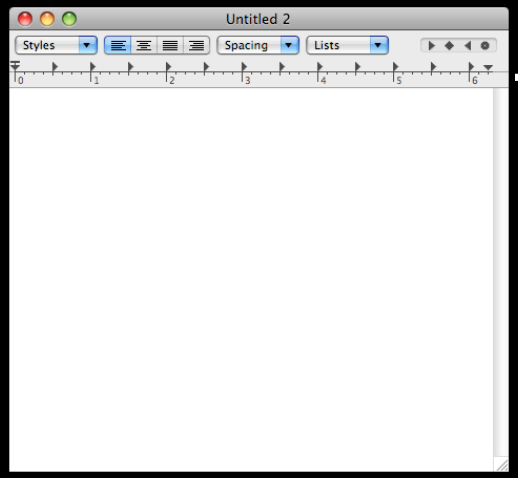
Notifications

- Facilitate loose coupling
- General purpose 1-to-N communication
- Lets you "broadcast" messages
- Notifications have a name, an object and an optional "userInfo" dictionary
- Coordinated through a communication "center"

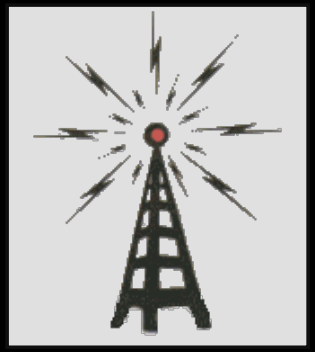
Example notification names

- NSApplicationWillFinishLaunchingNotification
- NSApplicationDidFinishLaunchingNotification
- NSSplitViewDidResizeSubviewsNotification
- NSTextDidChangeNotification
- NSWindowDidResizeNotification

Notifications



Window "posts"
a notification

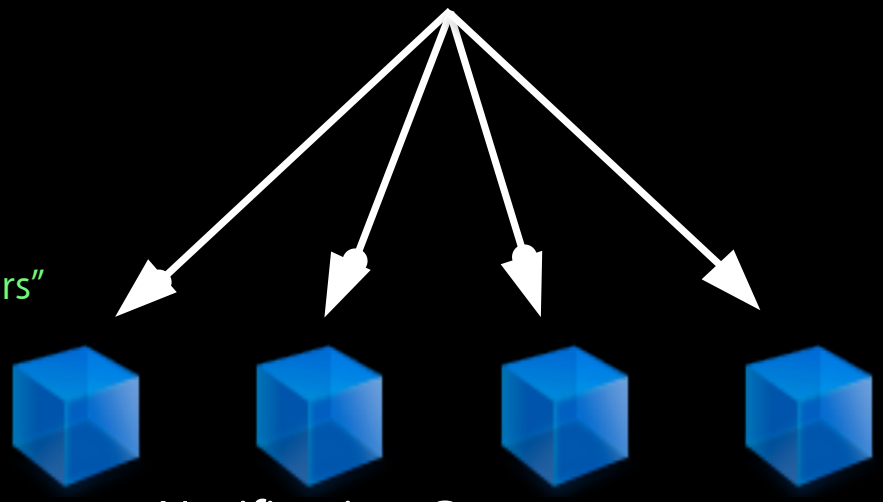


Notification Center

"Poster"

Window wants
to announce
windowDidBecomeMain:

"Observers"



Notification Center
broadcasts to observers

Registering for a notification

```
// Get the notification center
NSNotificationCenter *center =
        [NSNotificationCenter defaultCenter];

// register as an observer
    [center addObserver:self
        selector:@selector(windowBecameMain:)
            name:NSWindowDidBecomeMainNotification
          object:theWindow];
```

Observing Options

- Specific notification from a specific object

```
[center addObserver:self selector:@selector(objectDidSomething:)
      name:@"DoSomething" object:someObject];
```

- All notifications from a specific object

```
[center addObserver:self selector:@selector(objectDidSomething:)
      name:nil object:someObject];
```

- Specific notification from any object

```
[center addObserver:self selector:@selector(windowBecameMain:)
      name:@"WindowDidBecomeMain" object:nil];
```

Callback Conventions

- Notification callback methods return void and take a single argument: the notification
- - (void)windowBecameMain:(NSNotification *)notification

```
{
    NSString *name = [notification name];
    id object = [notification object];
    NSDictionary *userInfo = [notification userInfo];

    // handle the window activation...
}
```

Removing Observers

- Notification centers don't retain observers
- Before an object is dealloced, it **must** be cleared out of the notification center!

```
- (void)removeObserver:(id)observer;
```

- For example,

```
- (void)dealloc {
    [center removeObserver:self];
    [super dealloc];
}
```

Removing Options

- Specific notification from a specific object

```
[center removeObserver:self name:@"DoSomething"
      object:someObject];
```

- All notifications from a specific object

```
[center removeObserver:self name:nil object:someObject];
```

- Specific notification from any object

```
[center removeObserver:self name:@"WindowDidBecomeMain"
      object:nil];
```

- Any notification from any object

```
[center removeObserver:self];
```

Posting a notification

```
// Get the notification center
    NSNotificationCenter *center =
        [NSNotificationCenter defaultCenter];
// Post the notification
    [center postNotificationName:
        @“MyCustomNotification” object:theWindow];
```

Notification Miscellany

- Notification names are just strings
- Adding custom notifications is trivial
- Some delegate callbacks actually use notifications; delegate automatically registered when set as the delegate

Notification versus delegation

- To-many relationship
- Not used to alter behavior of the posting object

- Note: Some Application Framework classes automatically register delegates to receive selected notifications.

Panels, Sheets, and Alerts

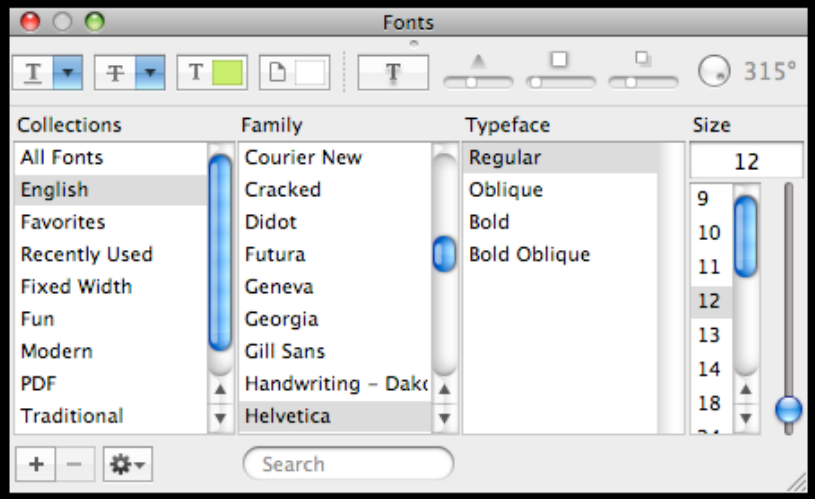
Panels, Sheets, Alerts

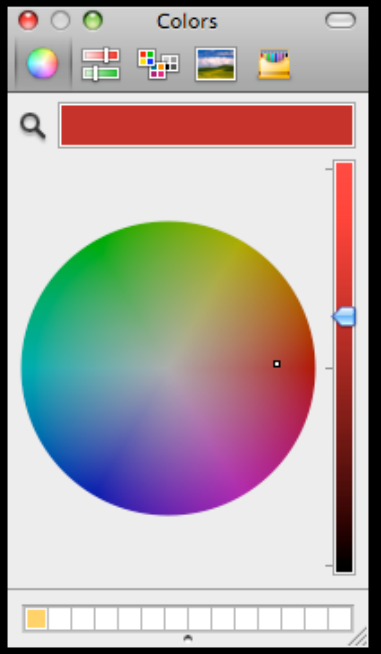
- Panels
  - An NSPanel is a subclass of NSWindow
  - Used for auxiliary windows such as inspectors
  - Some different default behaviors than windows
- Alerts
  - An NSAlert is a configurable object to present a warning or alert to the user as a modal window or as a sheet
- Sheets
  - A sheet is a window or panel run in a modal fashion in conjunction with another window
  - There is no 'NSSheet' class

Standard AppKit panels

- Provide a consistent look and feel
- New features picked up automatically by application

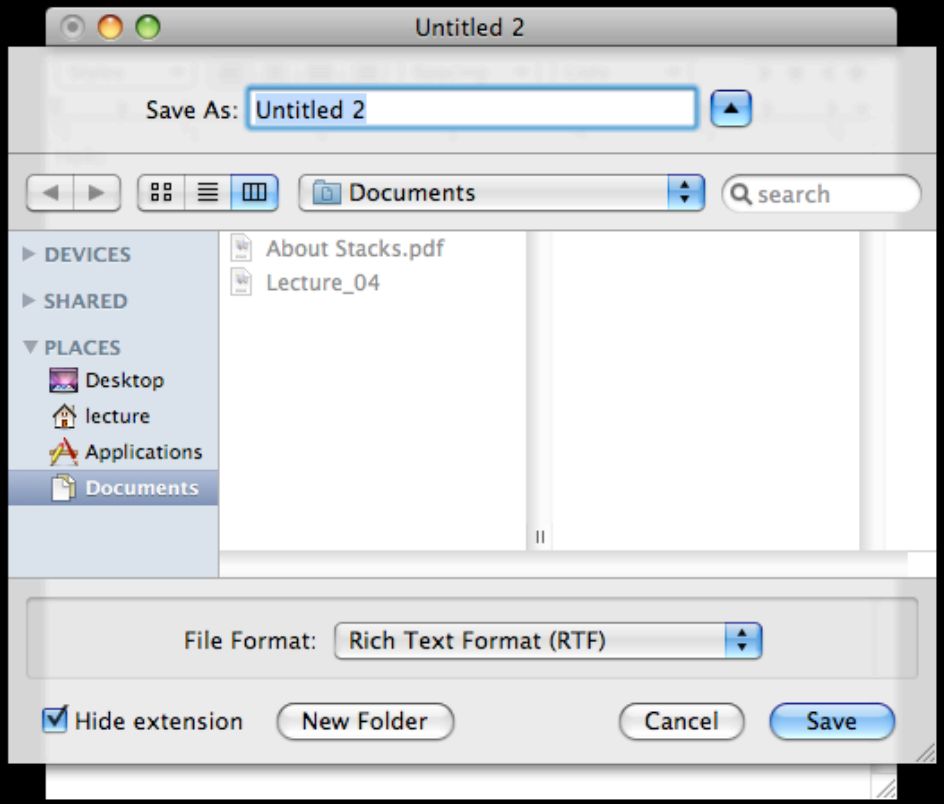
NSFontPanel and NSColorPanel





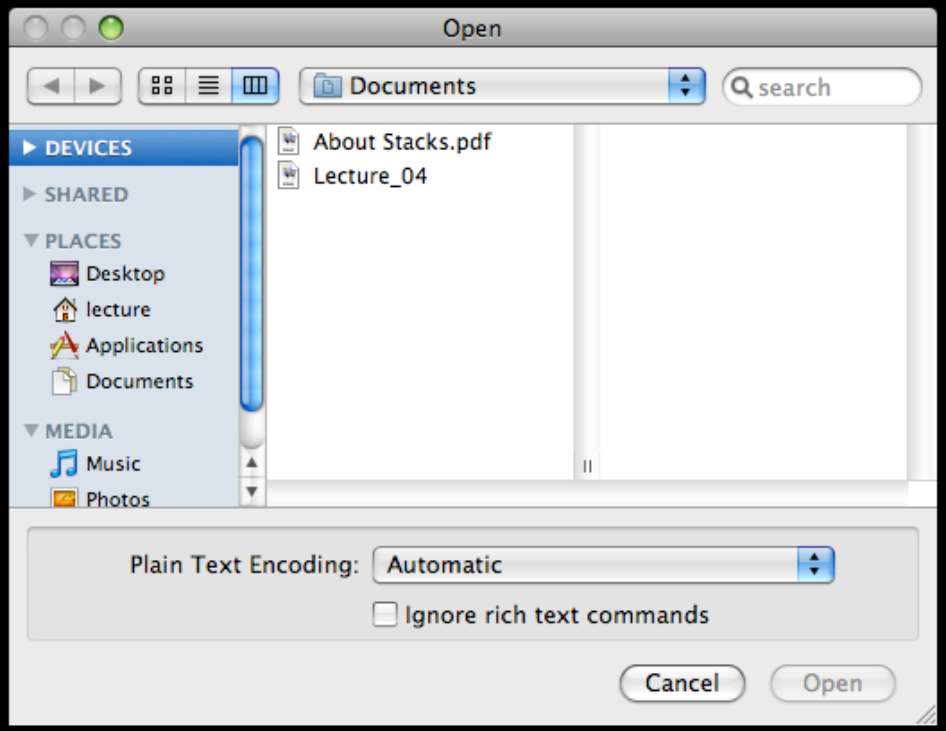
NSSavePanel

Used as window or sheet



NSOpenPanel

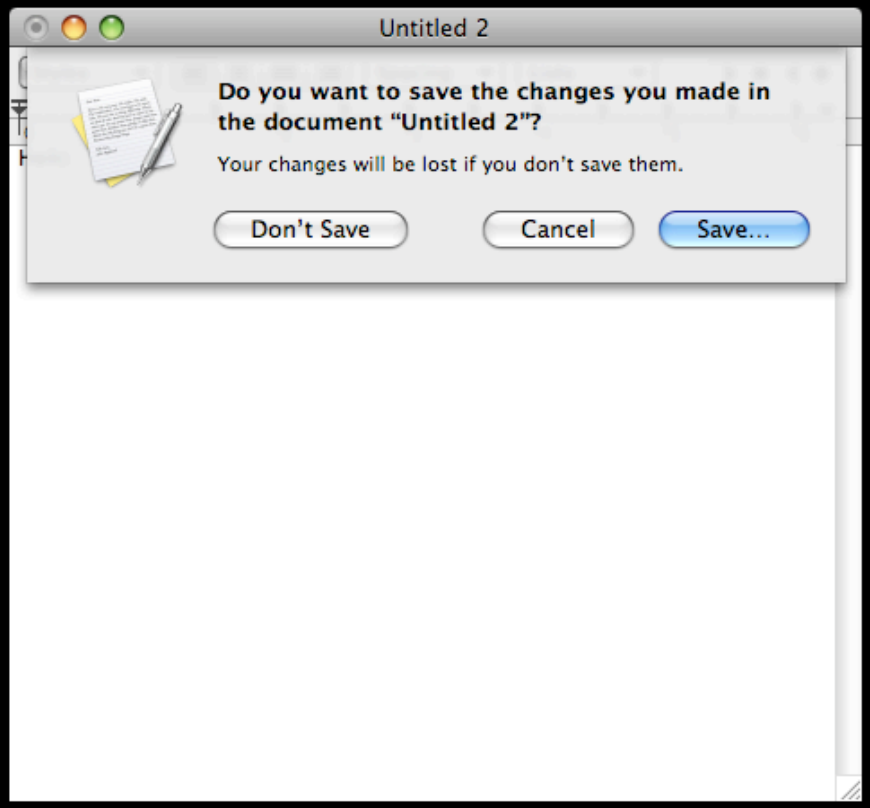
Used as window or sheet



Using NSOpenPanel

```
- (IBAction)chooseFile:(id)sender {
    NSOpenPanel *openPanel = [NSOpenPanel openPanel];
    // configure open panel
    [openPanel setAllowsMultipleSelection:NO];
    [openPanel setTitle:@"Select file to import"];
    // nil for types allows any type
    int result = [openPanel runModalForTypes:nil];
    if (result == NSOKButton) {
      NSString *filename = [openPanel fileName];
      // do something with filename
    }
}
```

Sheets provide window-level modality



Creating an alert sheet

```
- (BOOL) windowShouldClose:(id)sender {
     NSAlert *alert = [[[NSAlert alloc] init] autorelease];
    [alert addButtonWithTitle:@"Save"];
    [alert addButtonWithTitle:@"Cancel"];
    [alert addButtonWithTitle:@"Don’t Save"];
    [alert setMessageText:@"Unsaved Changes"];
    [alert setInformativeText:@"You’ll lose unsaved
    changes."];
    [alert setAlertStyle:NSWarningAlertStyle];

    [alert beginSheetModalForWindow:window
        modalDelegate: self
        didEndSelector:
           @selector(alertDidEnd:returnCode:contextInfo:)
        contextInfo: nil];

    return NO;
}
```

Alert sheet callback method

```
- (void) alertDidEnd:(NSAlert *)alert returnCode:(int) code
                                        contextInfo:(void *)context {

    switch(code) {

        case NSAlertFirstButtonReturn:
        [window close]; // note -close NOT -close:
        break;

        default:
        break;

    }

}
```

Questions?