Bringing Your iOS Apps to OS X

Session 216

Cortis Clark
Software Engineer

At a Glance

At a Glance

1. Rethink your design

At a Glance

1. Rethink your design
2. Restructure your code

At a Glance

1. Rethink your design
2. Restructure your code
3. Get started

Rethink Your Design

Embrace the Platform

Rethink your app for OS X

- Displays and windows

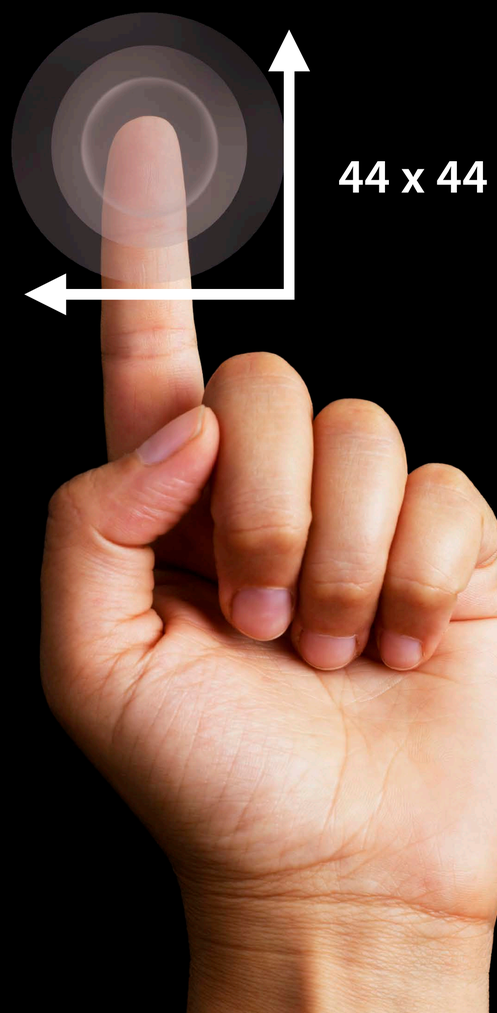
iOS

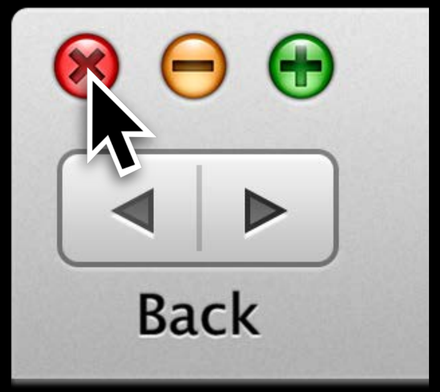
OS X

Embrace the Platform

Rethink your app for OS X

- Input devices





Much More Precise

Embrace the Platform

Rethink your app for OS X

Embrace the Platform

Rethink your app for OS X

- Menus and keyboard shortcuts

Embrace the Platform

Rethink your app for OS X

- Menus and keyboard shortcuts
- Undo and redo

Embrace the Platform

Rethink your app for OS X

- Menus and keyboard shortcuts
- Undo and redo
- Drag and drop

Embrace the Platform

Rethink your app for OS X

- Menus and keyboard shortcuts
- Undo and redo
- Drag and drop
- Quick Look

Embrace the Platform

Rethink your app for OS X

- Menus and keyboard shortcuts
- Undo and redo
- Drag and drop
- Quick Look
- Spotlight

Restructure Your Code

Leverage Existing Knowledge

Leverage Existing Knowledge

- Design patterns

Leverage Existing Knowledge

- Design patterns
- Xcode

Leverage Existing Knowledge

- Design patterns
- Xcode
- Languages and frameworks

Leverage Existing Knowledge

- Design patterns
- Xcode
- Languages and frameworks
- Resources

Leverage Existing Knowledge

- Design patterns
- Xcode
- Languages and frameworks
- Resources
- Localizations

Technology Stack

iOS and OS X

	iOS	Equivalence	OS X
Foundation	Core Foundation	=	Core Foundation
	Foundation	=	Foundation
	Core Data	=	Core Data
Text	Core Text	=	Core Text
Media	Core Graphics	=	Core Graphics
	Core Animation	=	Core Animation
	Core Image	<	Core Image
	Core Audio	<	Core Audio
	AV Foundation	<	AV Foundation
Cocoa	UIKit	~	AppKit

Technology Stack

iOS and OS X

	iOS	Equivalence	OS X
Foundation	Core Foundation	=	Core Foundation
	Foundation	=	Foundation
	Core Data	=	Core Data
Text	Core Text	=	Core Text
Media	Core Graphics	=	Core Graphics
	Core Animation	=	Core Animation
	Core Image	<	Core Image
	Core Audio	<	Core Audio
	AV Foundation	<	AV Foundation
Cocoa	UIKit	~	AppKit

Technology Stack

iOS and OS X

	iOS	Equivalence	OS X
Foundation	Core Foundation	=	Core Foundation
	Foundation	=	Foundation
	Core Data	=	Core Data
Text	Core Text	=	Core Text
Media	Core Graphics	=	Core Graphics
	Core Animation	=	Core Animation
	Core Image	<	Core Image
	Core Audio	<	Core Audio
	AV Foundation	<	AV Foundation
Cocoa	UIKit	~	AppKit

Technology Stack

iOS and OS X

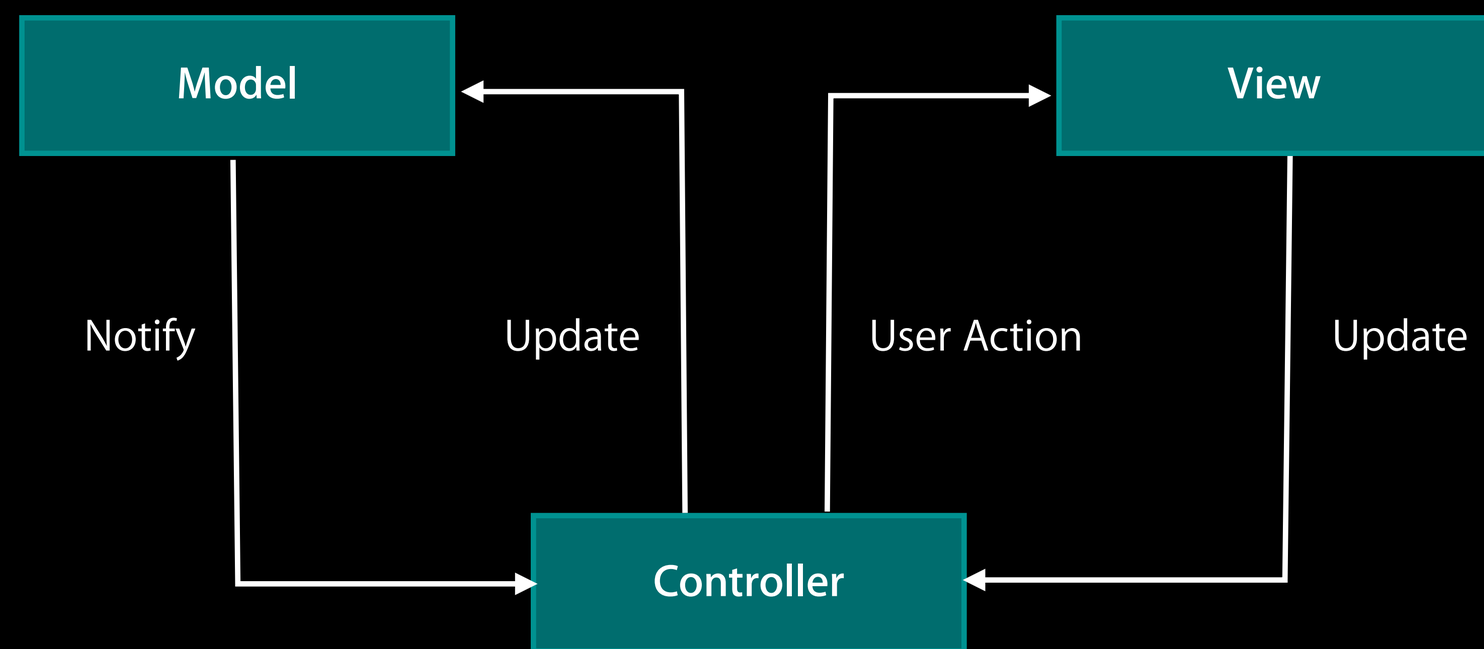
	iOS	Equivalence	OS X
Foundation	Core Foundation	=	Core Foundation
	Foundation	=	Foundation
	Core Data	=	Core Data
Text	Core Text	=	Core Text
Media	Core Graphics	=	Core Graphics
	Core Animation	=	Core Animation
	Core Image	<	Core Image
	Core Audio	<	Core Audio
	AV Foundation	<	AV Foundation
Cocoa	UIKit	~	AppKit

Technology Stack

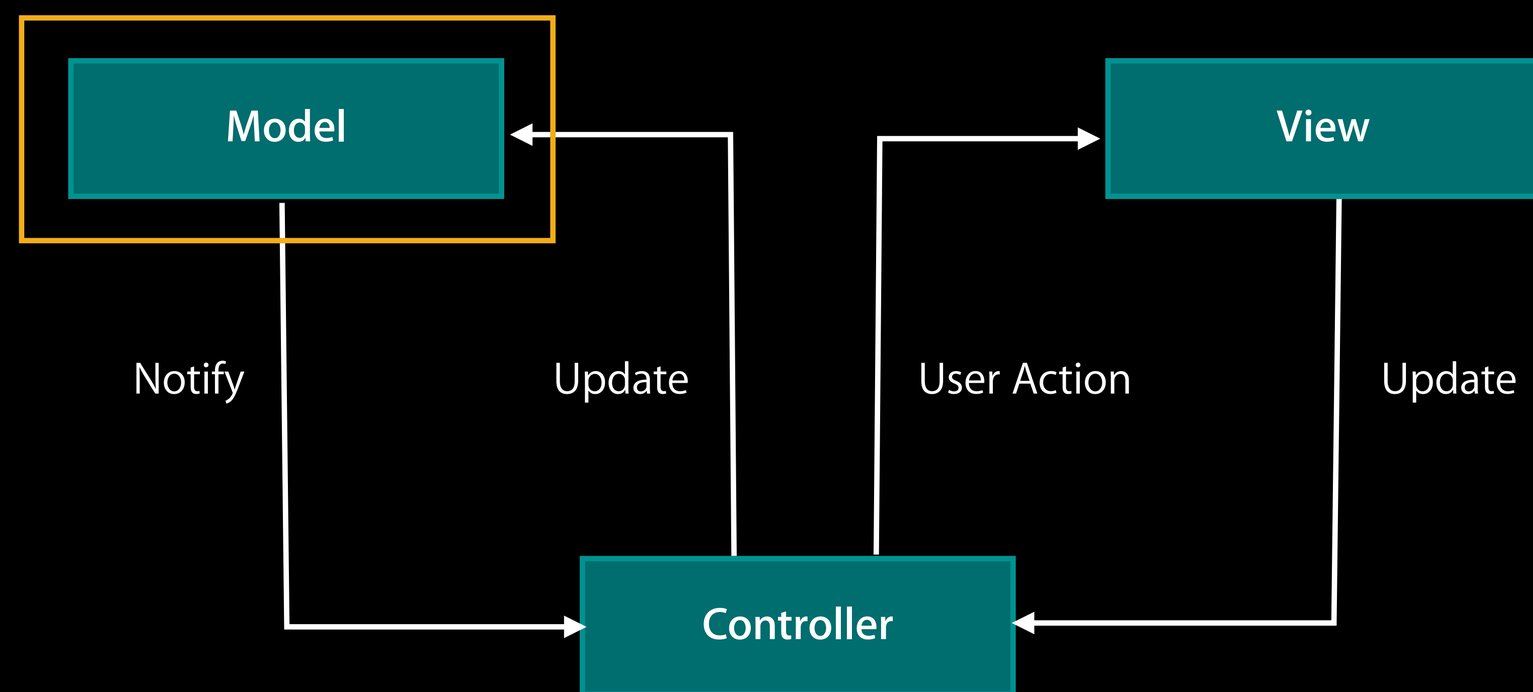
iOS and OS X

	iOS	Equivalence	OS X
Games	Open GL ES	<	Open GL
	Game Center	=	Game Center
	SpriteKit	=	SpriteKit

Model View Controller



Model View Controller



Migrating the Model

- Ensure clean model boundaries
  - Mostly re-usable
  - Model frameworks are cross-platform

Model Frameworks

Foundation, Core Foundation, Core Data etc.

Model Frameworks

Foundation, Core Foundation, Core Data etc.

- iOS

```
NSMutableArray *myArray = [NSMutableArray arrayWithCapacity:10];
[myArray addObject:@”WWDC 2013”];
```

Model Frameworks

Foundation, Core Foundation, Core Data etc.

- iOS

```
NSMutableArray *myArray = [NSMutableArray arrayWithCapacity:10];
[myArray addObject:@”WWDC 2013”];
```

- OS X

```
NSMutableArray *myArray = [NSMutableArray arrayWithCapacity:10];
[myArray addObject:@”WWDC 2013”];
```

32 Bit vs. 64 Bit

32 Bit vs. 64 Bit

- Variably sized types

```
NSInteger
NSUInteger
```

32 Bit vs. 64 Bit

- Variably sized types

```
NSInteger
NSUInteger
```

- Types of guaranteed size

```
uint32_t
int32_t
uint64_t
int64_t
```

Platform Specific Code

Platform Specific Code

- iOS only

Platform Specific Code

- iOS only

```
#if TARGET_OS_IPHONE
```

Platform Specific Code

- iOS only

```
#if TARGET_OS_IPHONE
```

- OS X only

Platform Specific Code

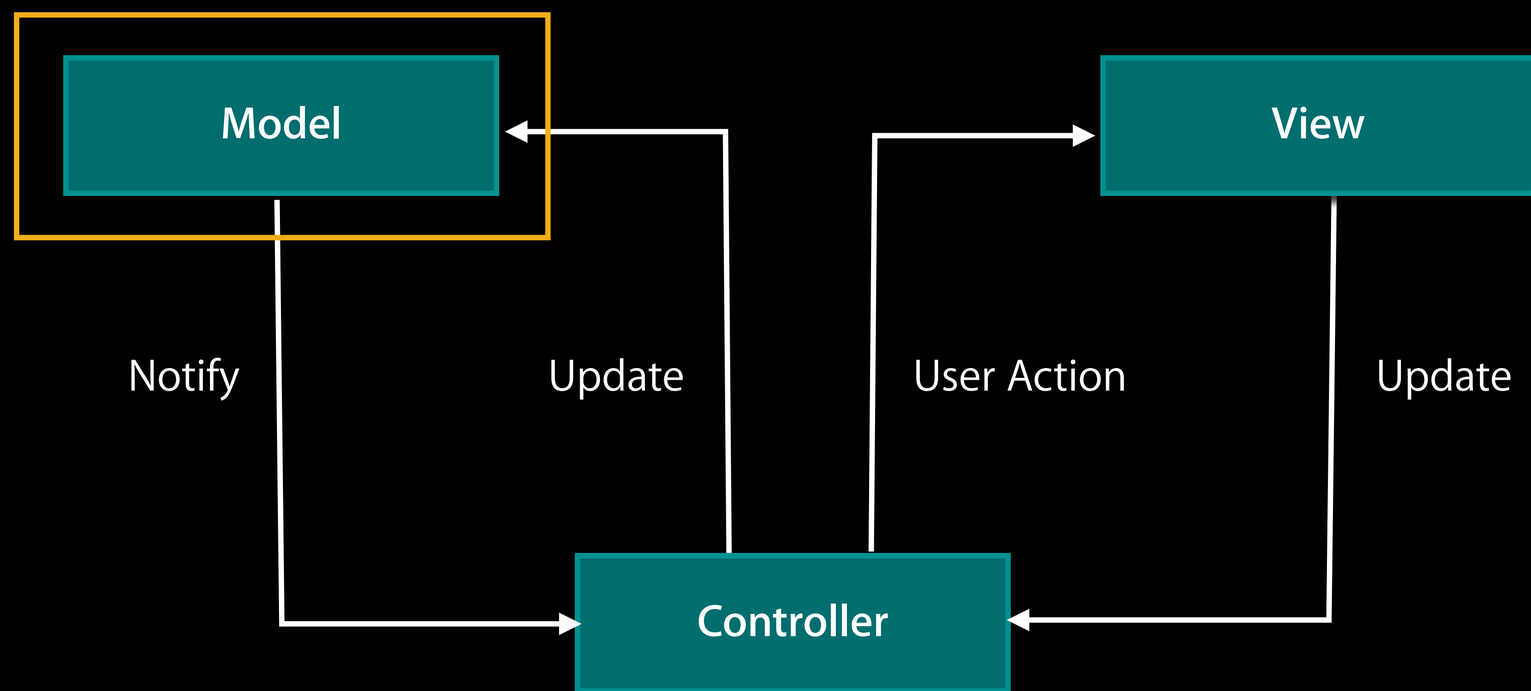
- iOS only

```
#if TARGET_OS_IPHONE
```

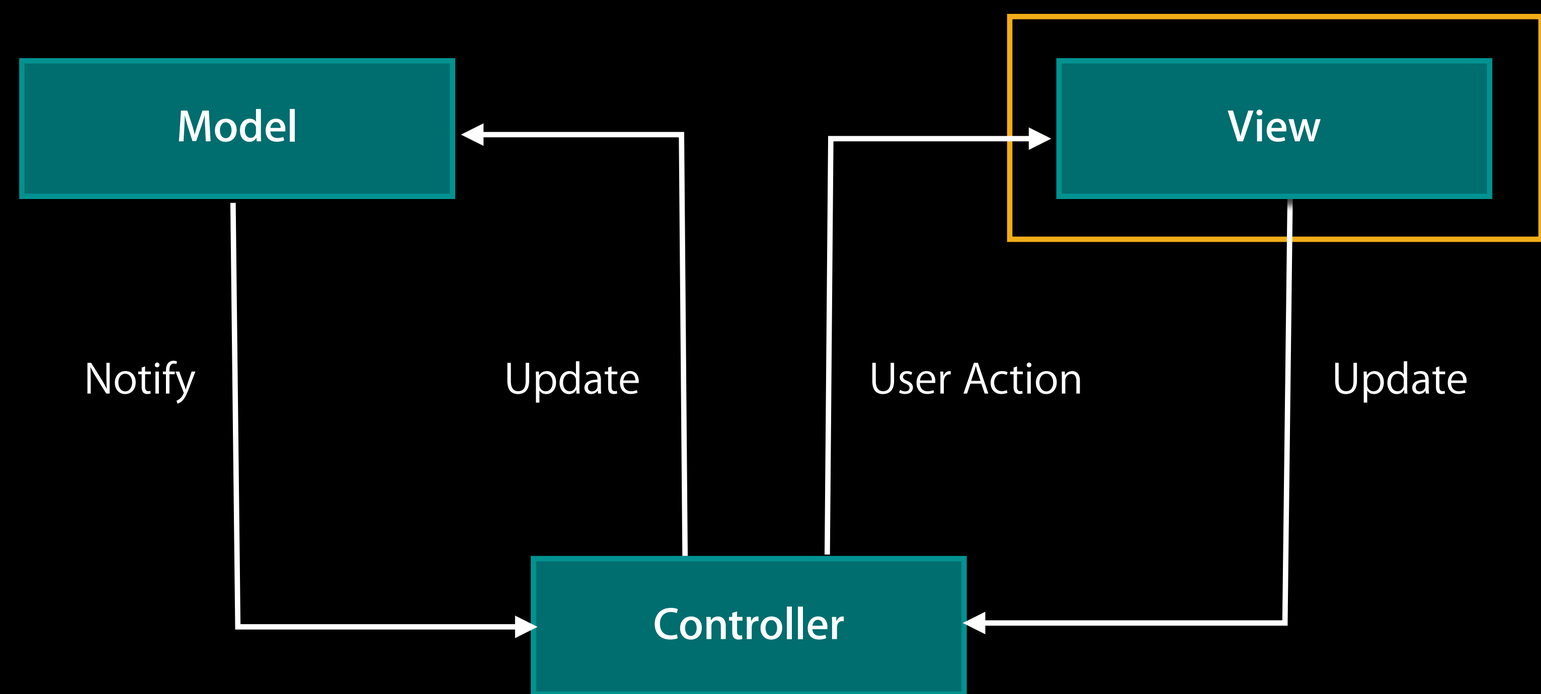
- OS X only

```
#if TARGET_OS_MAC && !TARGET_OS_IPHONE
```

Model View Controller

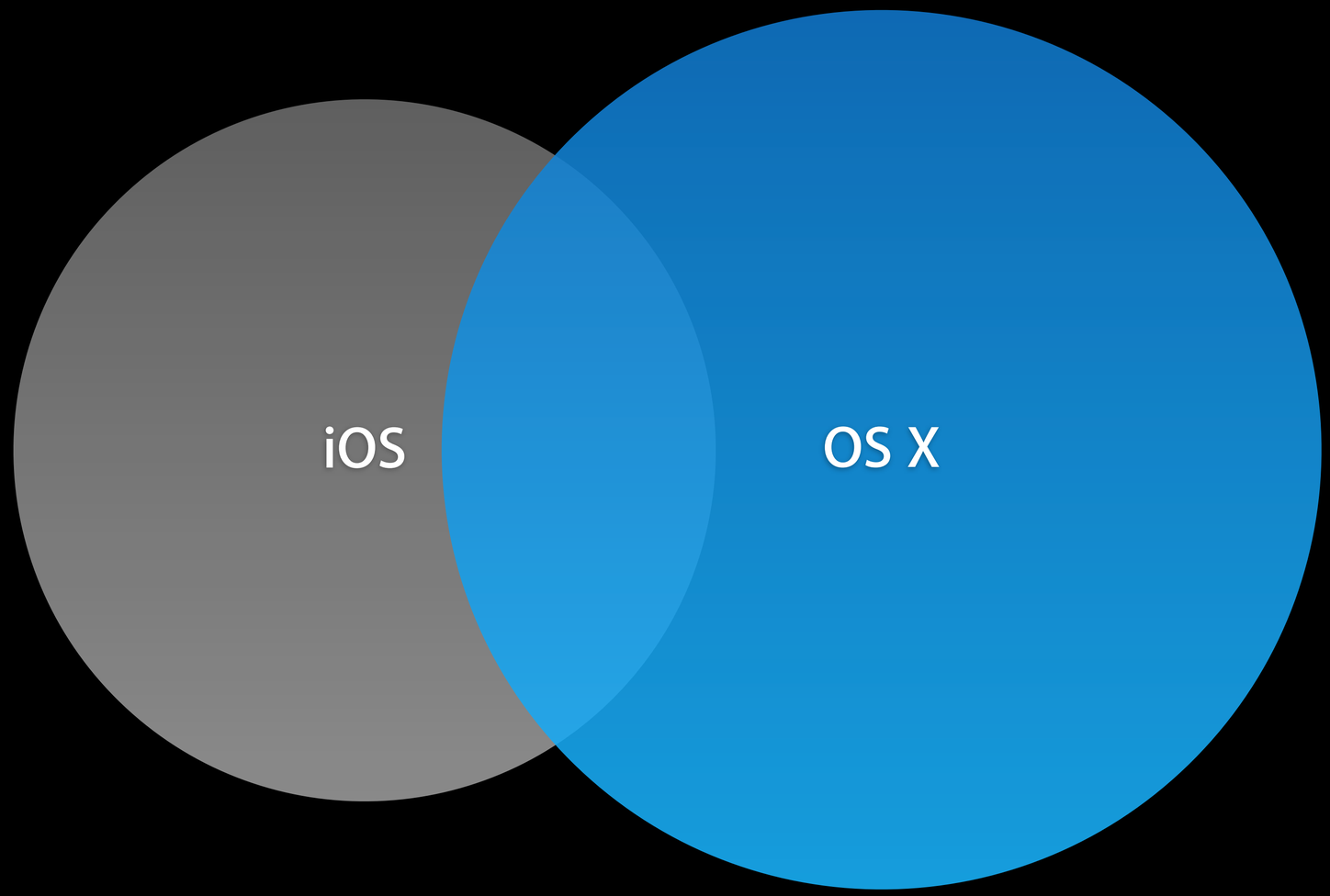


Model View Controller



Migrating the View

Built-in control set



UITableView vs. NSTableView

UITableView vs. NSTableView

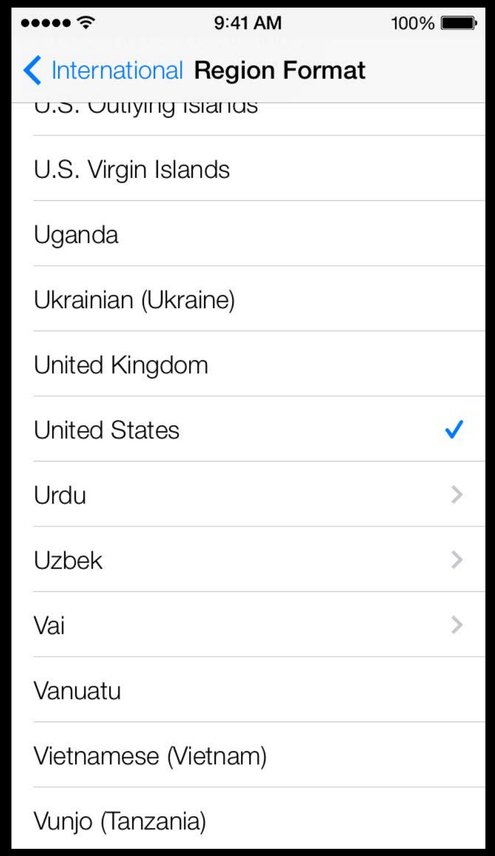
- Similarities
  - Data sources
  - Reusable cells
  - Animation

UITableView vs. NSTableView

- Similarities
  - Data sources
  - Reusable cells
  - Animation
- Differences
  - NSTableView has two variants (use view-based)
  - Multiple columns
  - Another control may be a better fit

TableView Differences

Select an item in a list

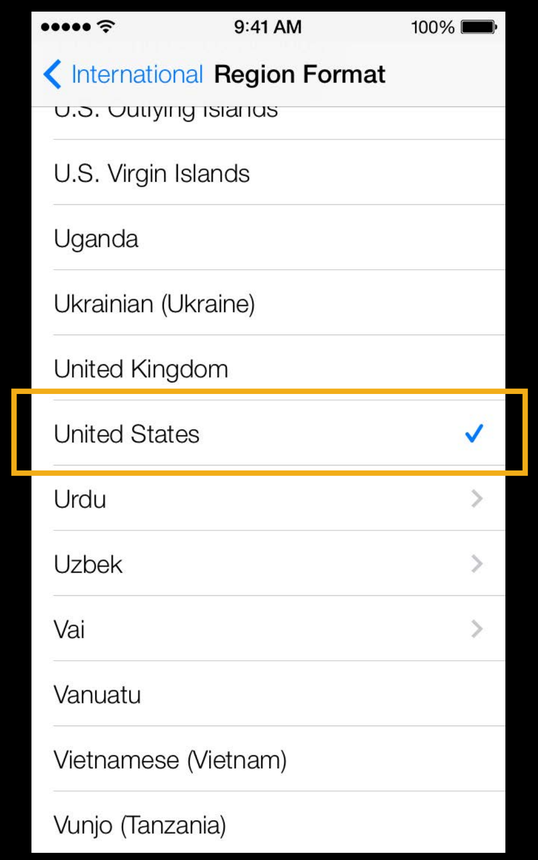


iOS UITableView

OS X NSPopupButton

TableView Differences

Select an item in a list

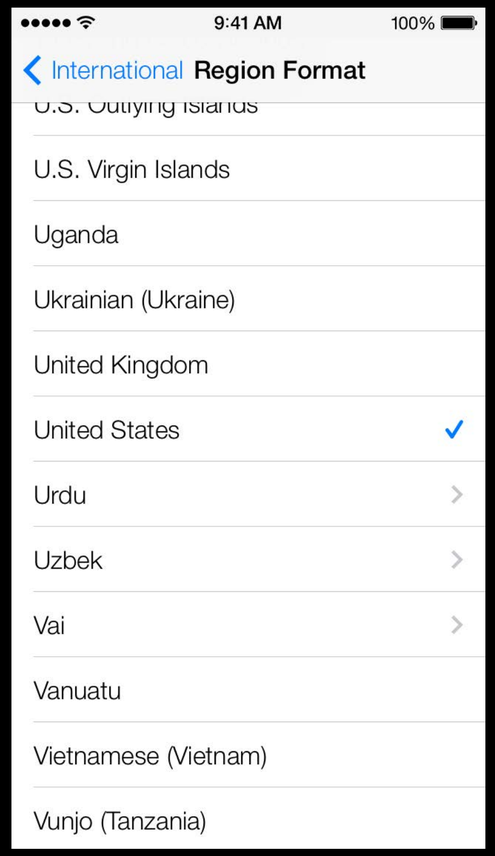


iOS UITableView

OS X NSPopupButton

TableView Differences

Select an item in a list

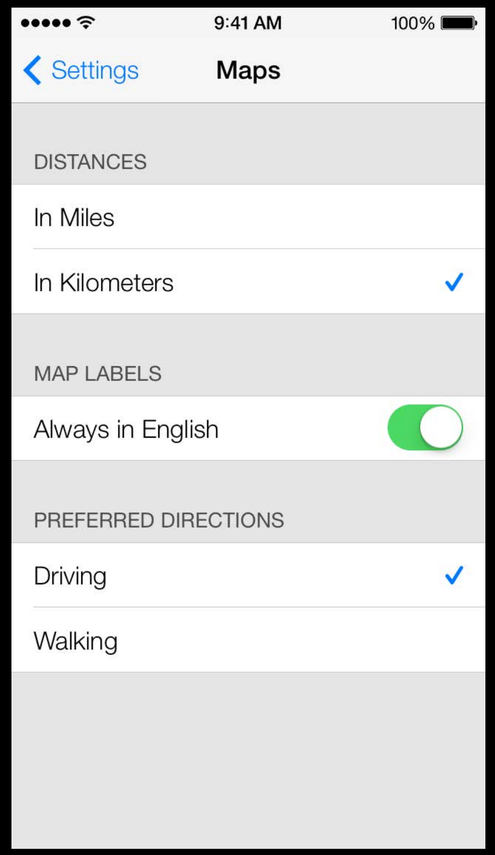


iOS UITableView

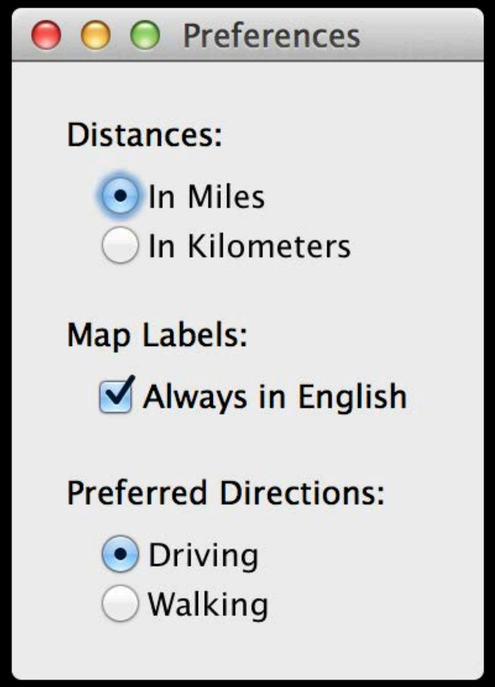
OS X NSPopupButton

TableView Differences

Select an item in a list



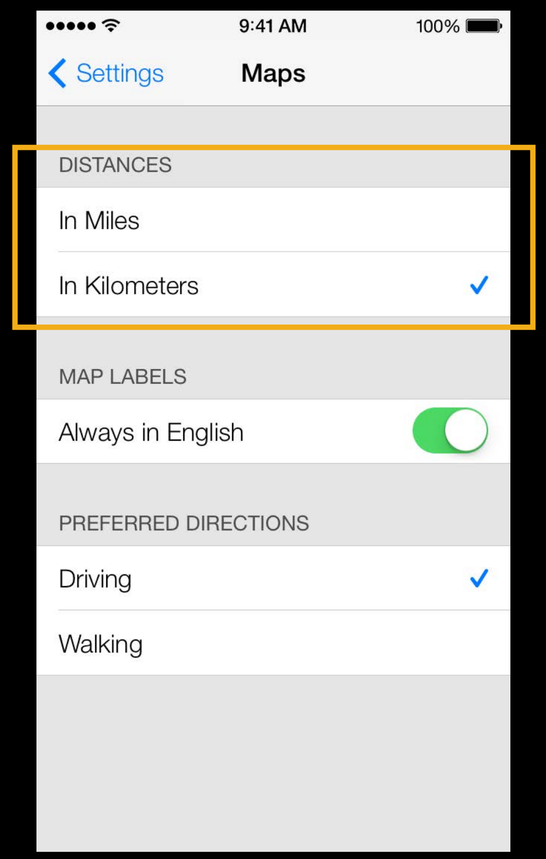
iOS UITableView



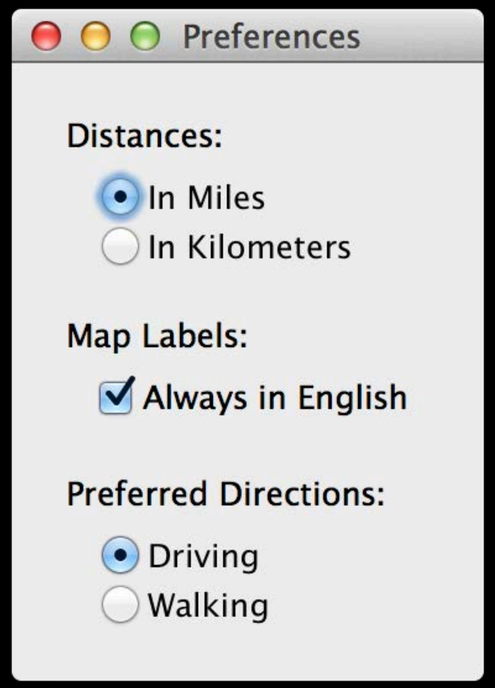
OS X NSButton
(or StackView)

TableView Differences

Select an item in a list



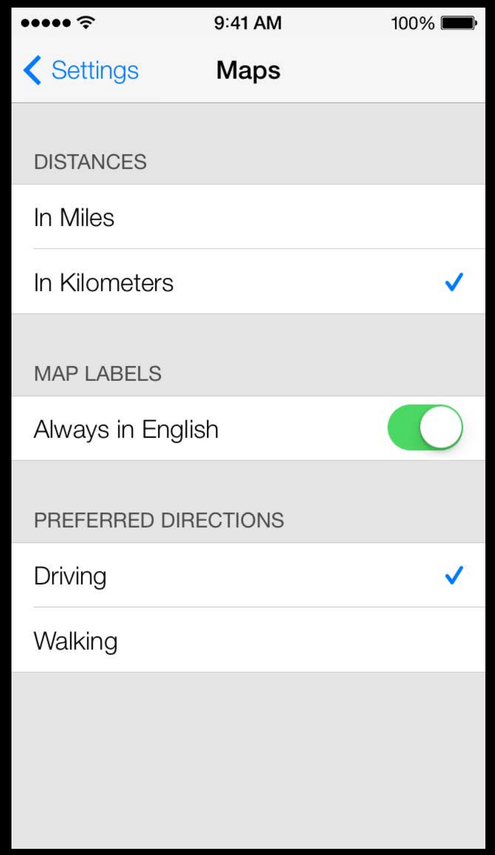
iOS UITableView



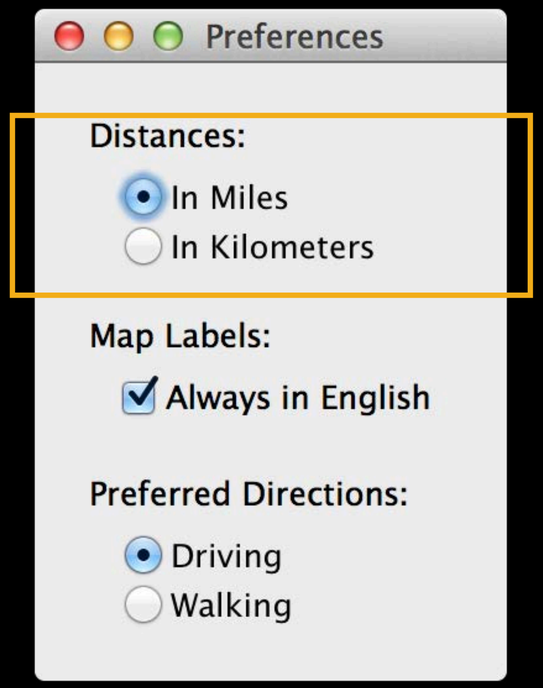
OS X NSButton
(or StackView)

TableView Differences

Select an item in a list



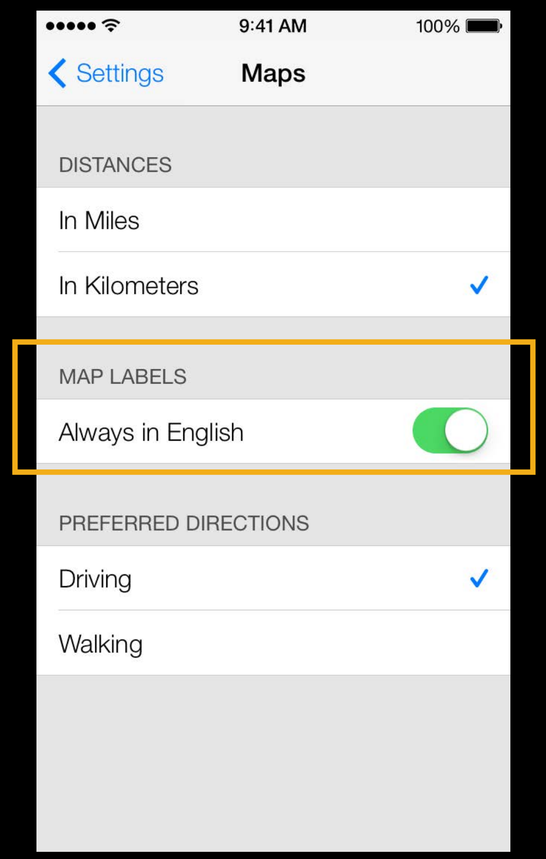
iOS UITableView



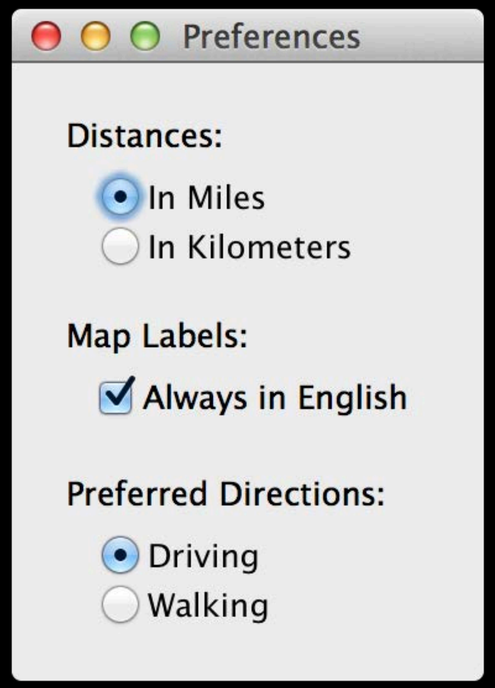
OS X NSButton
(or StackView)

TableView Differences

Select an item in a list



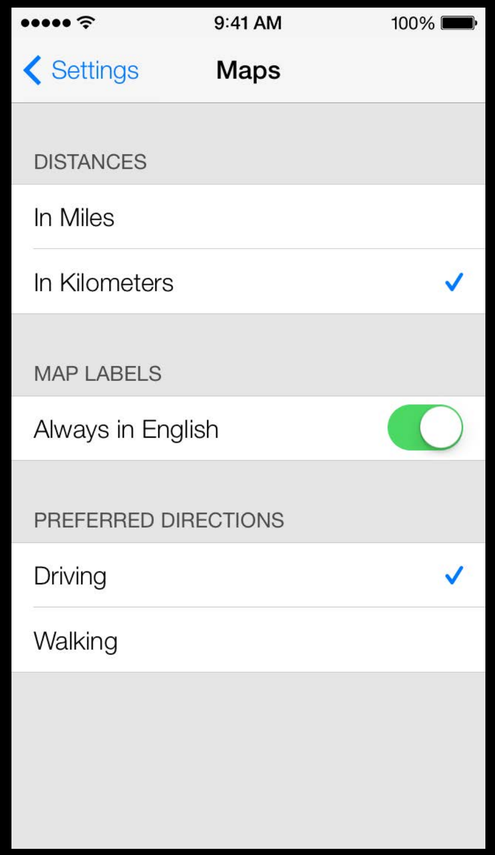
iOS UITableView



OS X NSButton
(or StackView)

TableView Differences

Select an item in a list



iOS UITableView

OS X NSButton
(or StackView)

UIView

NSView

UIView

NSView

Receives and handles events

UIView	NSView
Receives and handles events	
Responsible for drawing	

UIView	NSView
Receives and handles events	
Responsible for drawing	
(0, 0) in upper left	(0, 0) in lower left

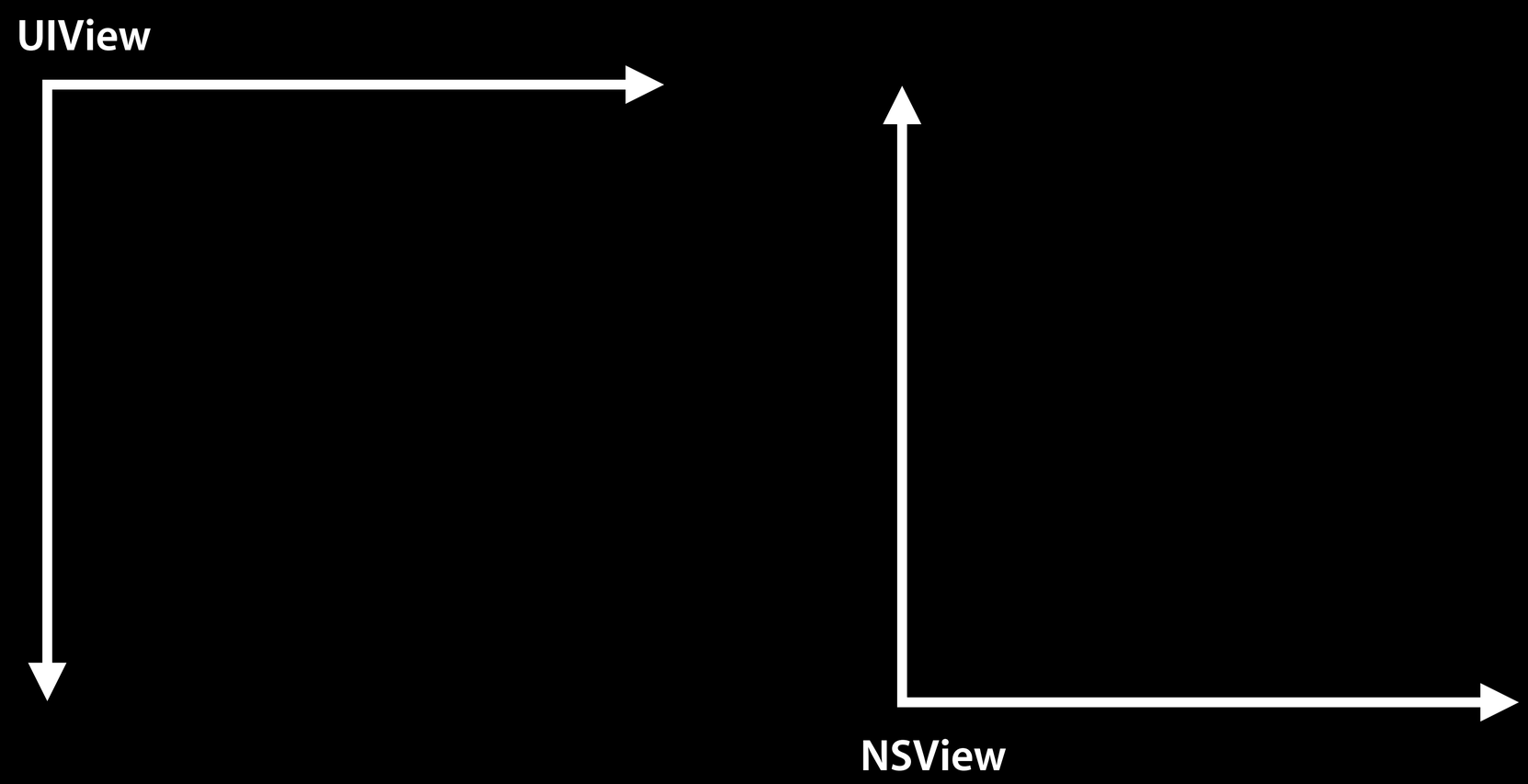
UIView	NSView
Receives and handles events	
Responsible for drawing	
(0, 0) in upper left	(0, 0) in lower left
Always has layer	Opt-in to layers

UIView	NSView
Receives and handles events	
Responsible for drawing	
(0, 0) in upper left	(0, 0) in lower left
Always has layer	Opt-in to layers
Subviews can draw outside view bounds	Subviews clip to view bounds

Drawing Origins

To match iOS

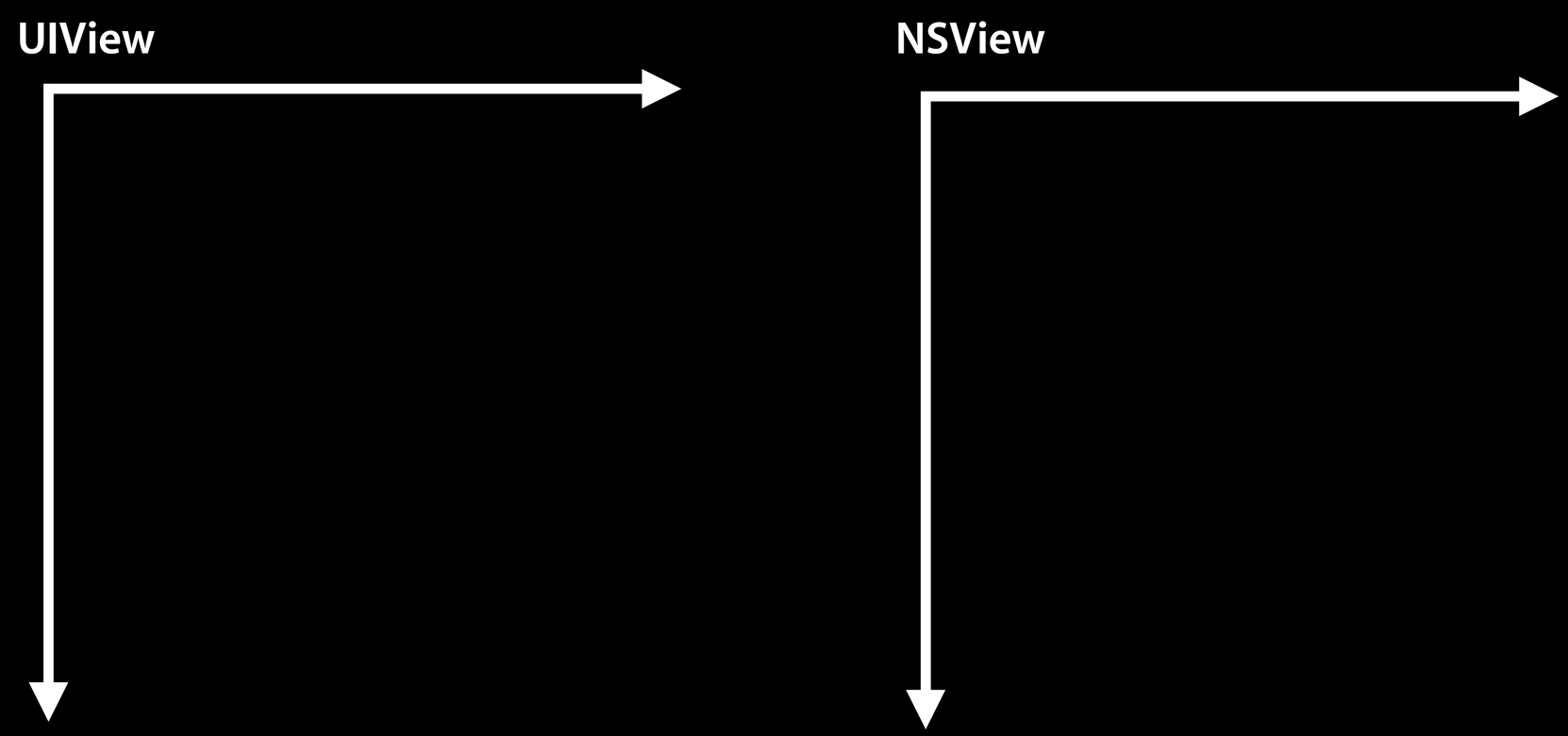
```
- (BOOL)isFlipped {
    return YES;
}
```



Drawing Origins

To match iOS

```
- (BOOL)isFlipped {
    return YES;
}
```



AppKit Flipped Views

Views that are flipped by default

NSButton
NSScrollView
NSSplitView
NSTabView
NSTableView

Layer Backed Views

Layer Backed Views

- Pros
  - Smoother animation
  - CAFilter
- Cons
  - More resource intensive

Layer Backed Views

Layer Backed Views

- UIView backed by layer

Layer Backed Views

- UIView backed by layer
- NSView can opt-in to layer backing

Layer Backed Views

- UIView backed by layer
- NSView can opt-in to layer backing
  - In code

Layer Backed Views

- UIView backed by layer
- NSView can opt-in to layer backing
  - In code

```
[aView setWantsLayer:YES];
```

Layer Backed Views

- UIView backed by layer
- NSView can opt-in to layer backing
  - In code

    ```
    [aView setWantsLayer:YES];
    ```

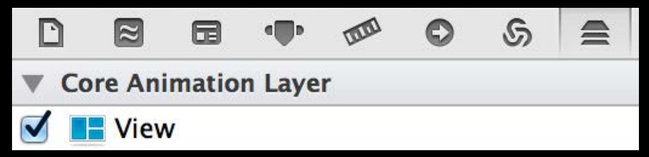
  - Using Xcode

Layer Backed Views

- UIView backed by layer
- NSView can opt-in to layer backing
  - In code

```
[aView setWantsLayer:YES];
```

  - Using Xcode

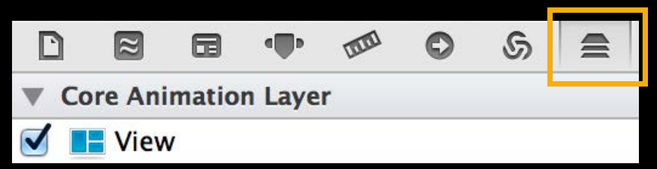


Layer Backed Views

- UIView backed by layer
- NSView can opt-in to layer backing
  - In code

```
[aView setWantsLayer:YES];
```

  - Using Xcode



Animation

Animation

- iOS

Animation

- iOS

```
[UIView animateWithDuration:1.0 animations:^{
```

Animation

- iOS

```
[UIView animateWithDuration:1.0 animations:^{
    aView.frame = CGRectMake(100.0,100.0,300.0,300.0);
```

Animation

- iOS

```
[UIView animateWithDuration:1.0 animations:^{
     aView.frame = CGRectMake(100.0,100.0,300.0,300.0);
}];
```

Animation

- iOS

```
[UIView animateWithDuration:1.0 animations:^{
     aView.frame = CGRectMake(100.0,100.0,300.0,300.0);
}];
```

- OS X uses animation proxies

Animation

- iOS

```
[UIView animateWithDuration:1.0 animations:^{
     aView.frame = CGRectMake(100.0,100.0,300.0,300.0);
}];
```

- OS X uses animation proxies

```
aView.animator.frame = NSMakeRect(100.0,100.0,300.0,300.0);
```

Events

Events

UITapGestureRecognizer replacement

Events

UITapGestureRecognizer replacement

```
- (void)mouseUp:(NSEvent *)theEvent {
   // Handle click
}
```

Events

UILongPressGestureRecognizer replacement

```
- (void)mouseDown:(NSEvent *)theEvent {
   self.longPressTimer = [NSTimer scheduledTimerWithTimeInterval:0.5
                                                target:self
                                                selector:@selector(doSomething:)
                                                userInfo:nil repeats:NO];
}

- (void)mouseUp:(NSEvent *)theEvent {
    [self.longPressTimer invalidate];
}

- (void)doSomething:(NSTimer*)theTimer {
   // perform long press action
}
```

Events

UILongPressGestureRecognizer replacement

99%

Use a Right Click Instead

Events

UILongPressGestureRecognizer replacement

```
- (NSMenu *)menuForEvent:(NSEvent *)theEvent {
   // return contextual menu
}
```

Events

UIPanGestureRecognizer replacement

```
- (void)mouseDown:(NSEvent *)theEvent {
    // Record drag start location
}

- (void)mouseDragged:(NSEvent *)theEvent {
    // Move view to new location
}

- (void)mouseUp:(NSEvent *)theEvent {
    // Cleanup drag code
}
```

Events

UIPanGestureRecognizer replacement

```
- (void)mouseDown:(NSEvent *)theEvent
{
    NSSize dragOffset = NSMakeSize(0.0, 0.0);  // parameter ignored

    NSPasteboard *pb = [NSPasteboard pasteboardWithName:NSDragPboard];
    [pb clearContents];
    [pb writeObjects:@[self.image]];

    [self dragImage:self.image at:self.imageLocation offset:dragOffset
        event:theEvent pasteboard:pb source:self slideBack:YES];

    return;
}
```

Events

UIPanGestureRecognizer replacement

```
- (void)mouseDown:(NSEvent *)theEvent
{
    NSSize dragOffset = NSMakeSize(0.0, 0.0);  // parameter ignored

    NSPasteboard *pb = [NSPasteboard pasteboardWithName:NSDragPboard];
    [pb clearContents];
    [pb writeObjects:@[self.image]];

    [self dragImage:self.image at:self.imageLocation offset:dragOffset
        event:theEvent pasteboard:pb source:self slideBack:YES];

    return;
}
```

Events

UIPanGestureRecognizer replacement

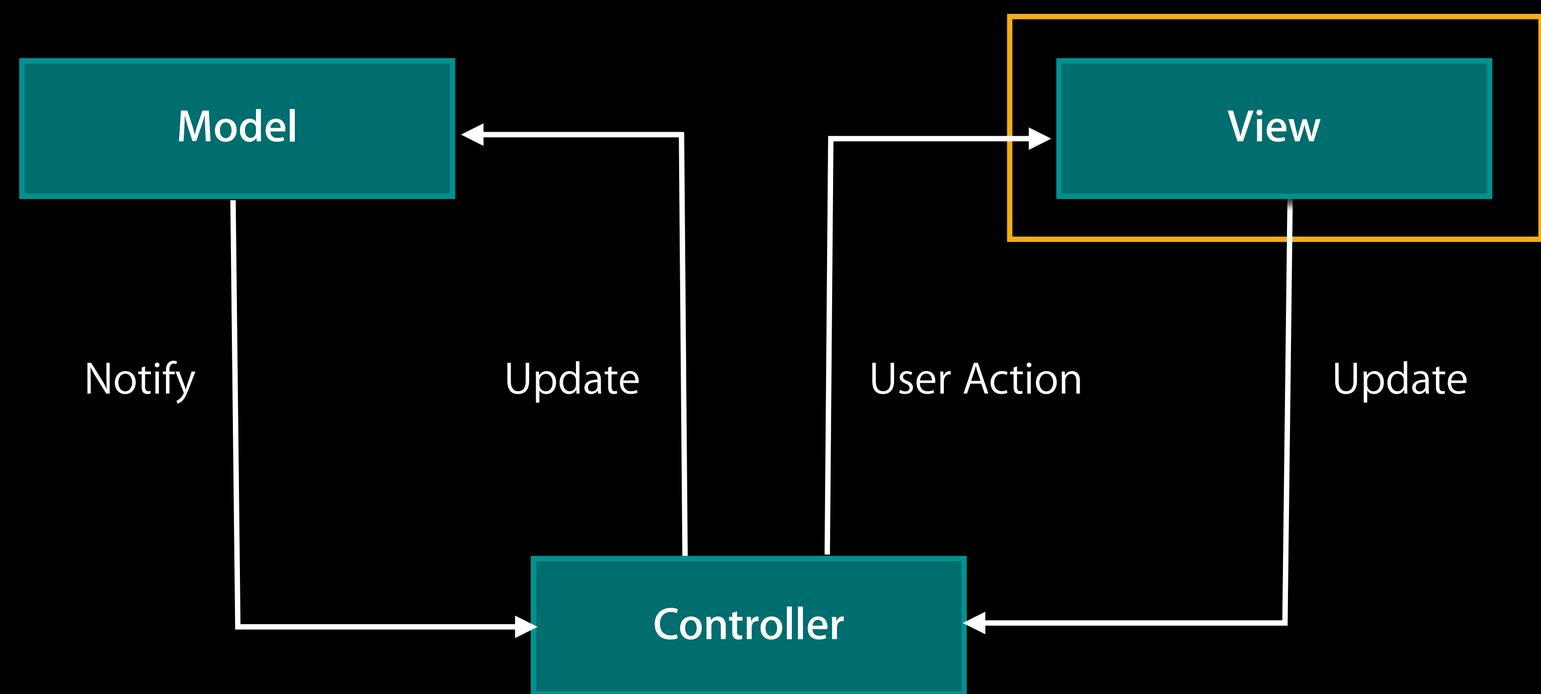
```
- (void)mouseDown:(NSEvent *)theEvent
{
    NSSize dragOffset = NSMakeSize(0.0, 0.0);  // parameter ignored

    NSPasteboard *pb = [NSPasteboard pasteboardWithName:NSDragPboard];
    [pb clearContents];
    [pb writeObjects:@[self.image]];

    [self dragImage:self.image at:self.imageLocation offset:dragOffset
        event:theEvent pasteboard:pb source:self slideBack:YES];

    return;
}
```

Events

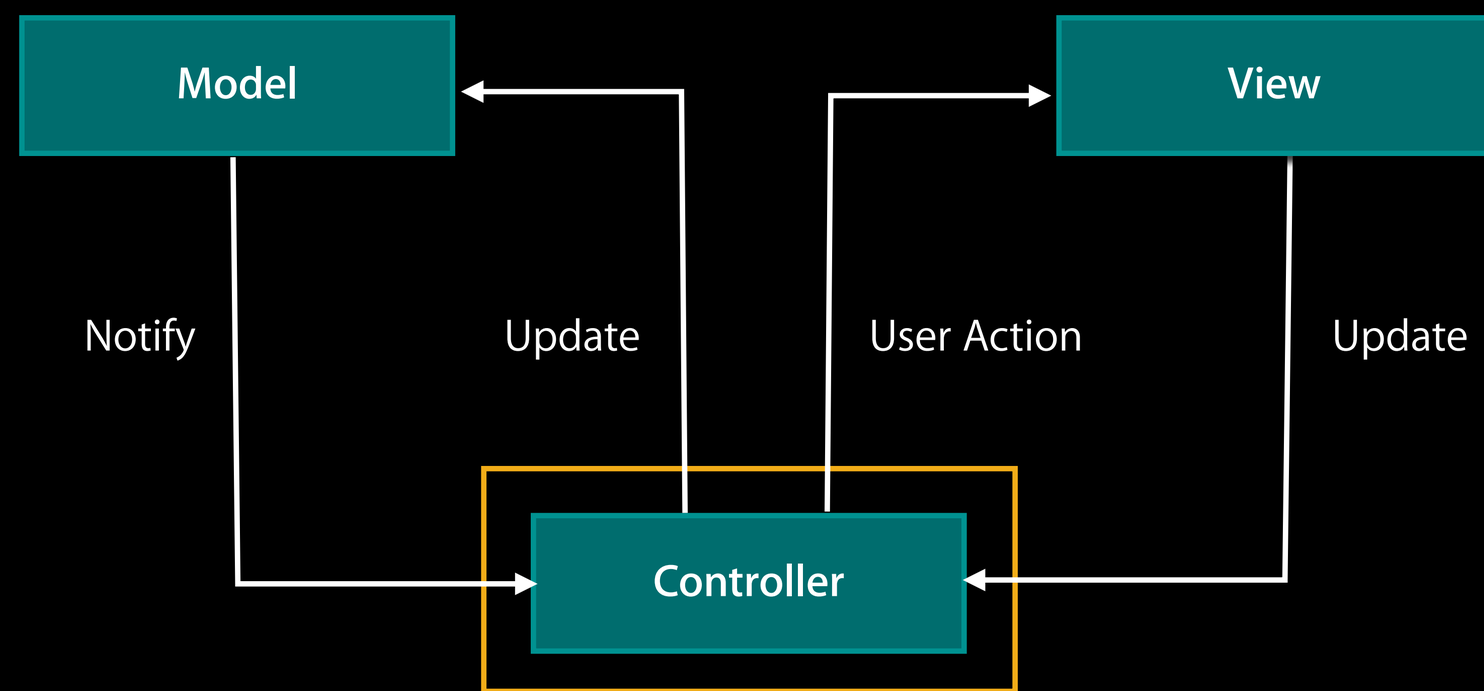
UIPanGestureRecognizer replacement

```
- (void)mouseDown:(NSEvent *)theEvent
{
    NSSize dragOffset = NSMakeSize(0.0, 0.0);  // parameter ignored

    NSPasteboard *pb = [NSPasteboard pasteboardWithName:NSDragPboard];
    [pb clearContents];
    [pb writeObjects:@[self.image]];

    [self dragImage:self.image at:self.imageLocation offset:dragOffset
        event:theEvent pasteboard:pb source:self slideBack:YES];

    return;
}
```

Model View Controller



Model View Controller



Migrating the Controller

Migrating the Controller

- UIViewController ≠ NSViewController

Migrating the Controller

- UIViewController ≠ NSViewController
- No NSNavigationController

Migrating the Controller

- UIViewController ≠ NSViewController
- No NSNavigationController
- Bindings

Migrating the Controller

- UIViewController ≠ NSViewController
- No NSNavigationController
- Bindings
- NSDocument awesomeness

UIDocument vs. NSDocument

UIDocument vs. NSDocument

- Similarities

UIDocument vs. NSDocument

- Similarities
  - Saves and loads

UIDocument vs. NSDocument

- Similarities
  - Saves and loads
  - Undo support

UIDocument vs. NSDocument

- Similarities
  - Saves and loads
  - Undo support
  - iCloud

UIDocument vs. NSDocument

- Similarities
  - Saves and loads
  - Undo support
  - iCloud
- NSDocument extras

UIDocument vs. NSDocument

- Similarities
  - Saves and loads
  - Undo support
  - iCloud
- NSDocument extras
  - File, Edit, and Window menus

UIDocument vs. NSDocument

- Similarities
  - Saves and loads
  - Undo support
  - iCloud
- NSDocument extras
  - File, Edit, and Window menus
  - UI for opening and saving files

UIDocument vs. NSDocument

- Similarities
  - Saves and loads
  - Undo support
  - iCloud
- NSDocument extras
  - File, Edit, and Window menus
  - UI for opening and saving files
  - Versions

UIDocument vs. NSDocument

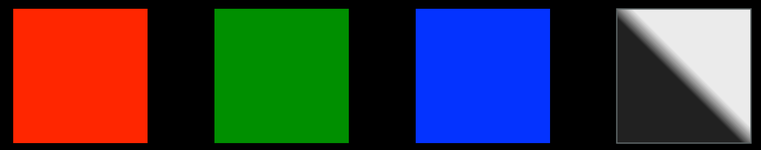
- Similarities
  - Saves and loads
  - Undo support
  - iCloud
- NSDocument extras
  - File, Edit, and Window menus
  - UI for opening and saving files
  - Versions
  - Plus more…

Demo

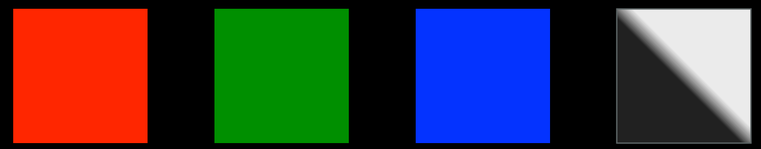
ShapeArt—A case study

Dan Schimpf
Demo Monkey

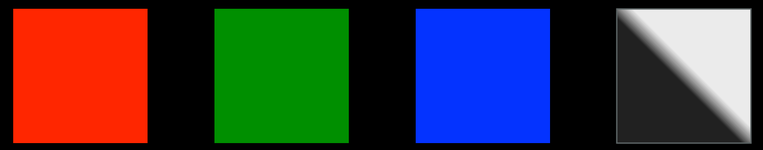
Migration Strategies



UIColor



NSColor



Migration Strategies

Mirrored code

```
UIColor *aColor = [UIColor redColor];

[aColor set];
```

```
NSColor *aColor = [NSColor redColor];

[aColor set];
```

iOS

OS X

Migration Strategies

Mirrored code

```
UIColor *aColor = [UIColor redColor];

[aColor set];
```

```
NSColor *aColor = [NSColor redColor];

[aColor set];
```

iOS

OS X

Migration Strategies

Mirrored code

```
UIColor *aColor = [UIColor redColor];

[aColor set];
```

```
NSColor *aColor = [NSColor redColor];

[aColor set];
```

iOS

OS X

Migration Strategies

Mirrored code

```
// Center new view on self
CGFloat x =
   (self.bounds.width - w)/2;
CGFloat y =
   (self.bounds.height - h)/2;

CGRect frame = CGRectMake(x,y,w,h);
UIView *view = [[UIView alloc]
  initWithFrame:frame];

// Place new view below others
[self insertSubview:view atIndex:0];
```

iOS

```
// Center new view of on self
CGFloat x =
   (self.bounds.width - w)/2;
CGFloat y =
   (self.bounds.height - h)/2;

NSRect frame = NSMakeRect(x,y,w,h);
NSView *view = [[NSView alloc]
  initWithFrame:frame];

// Place new view below others
[self addSubview:view
      positioned:NSWindowBelow
      relativeTo:nil];
```

OS X

Migration Strategies

Mirrored code

```
// Center new view on self
CGFloat x =
   (self.bounds.width - w)/2;
CGFloat y =
   (self.bounds.height - h)/2;

CGRect frame = CGRectMake(x,y,w,h);
UIView *view = [[UIView alloc]
  initWithFrame:frame];

// Place new view below others
[self insertSubview:view atIndex:0];
```

iOS

```
// Center new view of on self
CGFloat x =
   (self.bounds.width - w)/2;
CGFloat y =
   (self.bounds.height - h)/2;

NSRect frame = NSMakeRect(x,y,w,h);
NSView *view = [[NSView alloc]
  initWithFrame:frame];

// Place new view below others
[self addSubview:view
      positioned:NSWindowBelow
      relativeTo:nil];
```

OS X

Migration Strategies

Mirrored code

```
// Center new view on self
CGFloat x =
   (self.bounds.width - w)/2;
CGFloat y =
   (self.bounds.height - h)/2;

CGRect frame = CGRectMake(x,y,w,h);
UIView *view = [[UIView alloc]
  initWithFrame:frame];

// Place new view below others
[self insertSubview:view atIndex:0];
```

iOS

```
// Center new view of on self
CGFloat x =
   (self.bounds.width - w)/2;
CGFloat y =
   (self.bounds.height - h)/2;

NSRect frame = NSMakeRect(x,y,w,h);
NSView *view = [[NSView alloc]
  initWithFrame:frame];

// Place new view below others
[self addSubview:view
      positioned:NSWindowBelow
      relativeTo:nil];
```

OS X

Migration Strategies

Mirrored code

```
// Center new view on self
CGFloat x =
   (self.bounds.width - w)/2;
CGFloat y =
   (self.bounds.height - h)/2;

CGRect frame = CGRectMake(x,y,w,h);
UIView *view = [[UIView alloc]
  initWithFrame:frame];

// Place new view below others
[self insertSubview:view atIndex:0];
```

iOS

```
// Center new view of on self
CGFloat x =
   (self.bounds.width - w)/2;
CGFloat y =
   (self.bounds.height - h)/2;

NSRect frame = NSMakeRect(x,y,w,h);
NSView *view = [[NSView alloc]
  initWithFrame:frame];

// Place new view below others
[self addSubview:view
      positioned:NSWindowBelow
      relativeTo:nil];
```

OS X

Migration Strategies

Mirrored code

```
// Center new view on self
CGFloat x =
   (self.bounds.width - w)/2;
CGFloat y =
   (self.bounds.height - h)/2;

CGRect frame = CGRectMake(x,y,w,h);
UIView *view = [[UIView alloc]
  initWithFrame:frame];

// Place new view below others
[self insertSubview:view atIndex:0];
```

iOS

```
// Center new view of on self
CGFloat x =
   (self.bounds.width - w)/2;
CGFloat y =
   (self.bounds.height - h)/2;

NSRect frame = NSMakeRect(x,y,w,h);
NSView *view = [[NSView alloc]
  initWithFrame:frame];

// Place new view below others
[self addSubview:view
      positioned:NSWindowBelow
      relativeTo:nil];
```

OS X

Migration Strategies

Mirrored code

```
// Center new view on self
CGFloat x =
   (self.bounds.width - w)/2;
CGFloat y =
   (self.bounds.height - h)/2;

CGRect frame = CGRectMake(x,y,w,h);
UIView *view = [[UIView alloc]
  initWithFrame:frame];

// Place new view below others
[self insertSubview:view atIndex:0];
```

iOS

```
// Center new view of on self
CGFloat x =
   (self.bounds.width - w)/2;
CGFloat y =
   (self.bounds.height - h)/2;

NSRect frame = NSMakeRect(x,y,w,h);
NSView *view = [[NSView alloc]
  initWithFrame:frame];

// Place new view below others
[self addSubview:view
      positioned:NSWindowBelow
      relativeTo:nil];
```

OS X

Migration Strategies

Mirrored code

```
// Center new view on self
CGFloat x =
   (self.bounds.width - w)/2;
CGFloat y =
   (self.bounds.height - h)/2;

CGRect frame = CGRectMake(x,y,w,h);
UIView *view = [[UIView alloc]
  initWithFrame:frame];

// Place new view below others
[self insertSubview:view atIndex:0];
```

iOS

```
// Center new view of on self
CGFloat x =
   (self.bounds.width - w)/2;
CGFloat y =
   (self.bounds.height - h)/2;

NSRect frame = NSMakeRect(x,y,w,h);
NSView *view = [[NSView alloc]
  initWithFrame:frame];

// Place new view below others
[self addSubview:view
      positioned:NSWindowBelow
      relativeTo:nil];
```

OS X

Migration Strategies

Mirrored code

Migration Strategies

Mirrored code

- Pros
  - Flexibility

Migration Strategies

Mirrored code

- Pros
  - Flexibility
- Cons
  - Code duplication
    - Greater maintenance cost
    - Greater testing cost

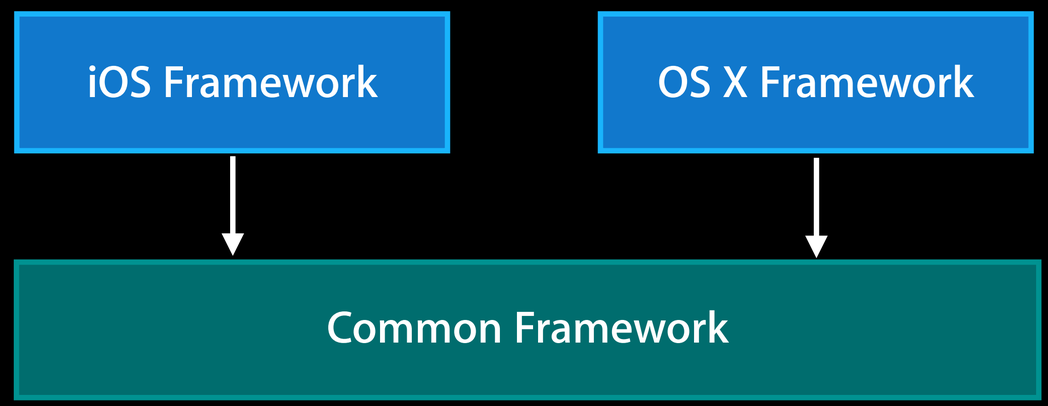
Migration Strategies

Mirrored code

- Pros
  - Flexibility
- Cons
  - Code duplication
    - Greater maintenance cost
    - Greater testing cost
- When to use?
  - Heavily platform dependent code

Migration Strategies

Use common framework



Migration Strategies

Use common framework

```
CIColor *aColor = [CIColor colorWithRed:1.0 green:0.0 blue:0.0 alpha:1.0];

CIImage *image = [CIImage imageWithColor:aColor];
```

iOS and OS X

Migration Strategies

Use common framework

Migration Strategies

Use common framework

- Lower level frameworks are cross-platform

Migration Strategies

Use common framework

- Lower level frameworks are cross-platform
- Pros
  - Maximizes code reuse
  - Most robust
  - Minimal maintenance

Migration Strategies

Use common framework

- Lower level frameworks are cross-platform
- Pros
  - Maximizes code reuse
  - Most robust
  - Minimal maintenance
- Cons
  - Refactoring
  - Less functionality

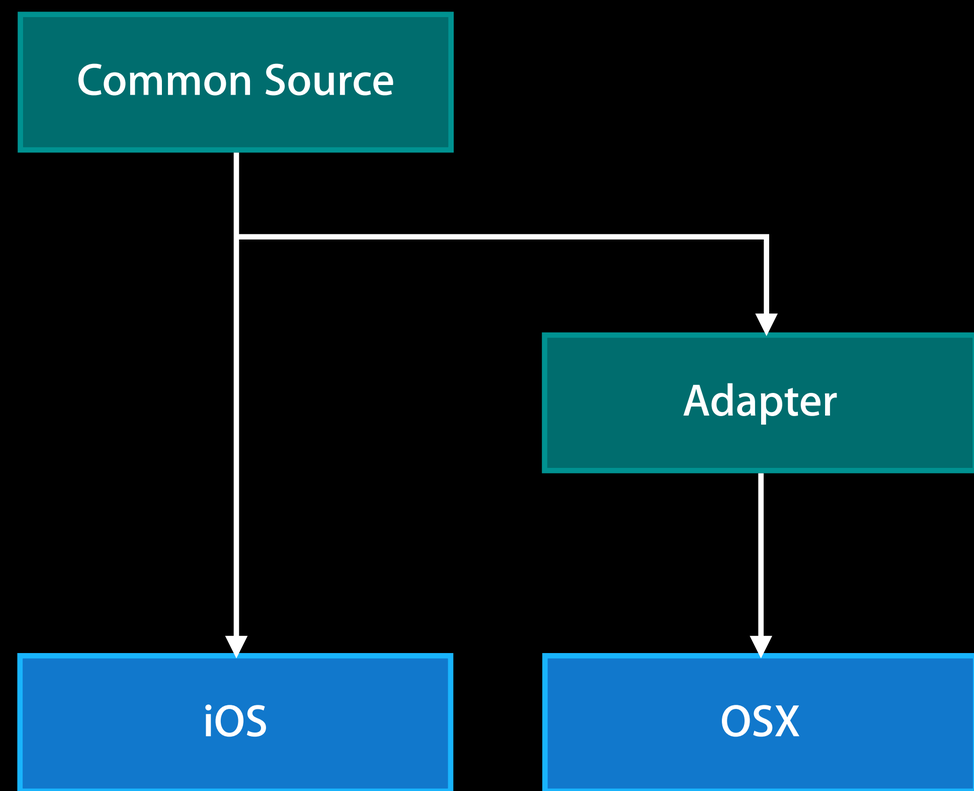
Migration Strategies

Use common framework

- Lower level frameworks are cross-platform
- Pros
  - Maximizes code reuse
  - Most robust
  - Minimal maintenance
- Cons
  - Refactoring
  - Less functionality
- When to use?
  - If common framework provides needed functionality

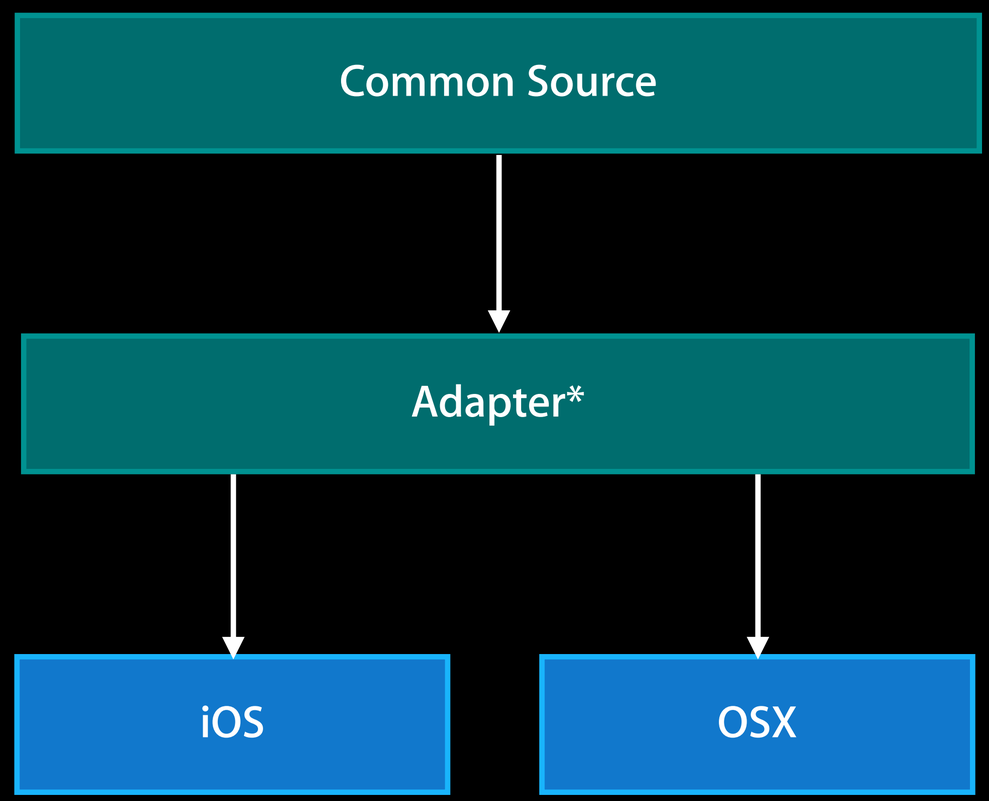
Migration Strategies

Adapter pattern



Migration Strategies

Adapter pattern*



Migration Strategies

Adapter pattern

```
// XPlatformColor.h

#if TARGET_OS_IPHONE                          // iOS

@interface XPlatformColor : NSObject {
   UIColor *_underlyingColor;
}

#elif TARGET_OS_MAC && !TARGET_OS_IPHONE   // OS X

@interface XPlatformColor : NSObject {
   NSColor *_underlyingColor;
}

#endif
```

Migration Strategies

Adapter pattern

```
// XPlatformColor.h

#if TARGET_OS_IPHONE                          // iOS

@interface XPlatformColor : NSObject {
   UIColor *_underlyingColor;
}

#elif TARGET_OS_MAC && !TARGET_OS_IPHONE   // OS X

@interface XPlatformColor : NSObject {
   NSColor *_underlyingColor;
}

#endif
```

Migration Strategies

Adapter pattern

```
// XPlatformColor.h

#if TARGET_OS_IPHONE                          // iOS

@interface XPlatformColor : NSObject {
   UIColor *_underlyingColor;
}

#elif TARGET_OS_MAC && !TARGET_OS_IPHONE   // OS X

@interface XPlatformColor : NSObject {
   NSColor *_underlyingColor;
}

#endif
```

Migration Strategies

Adapter pattern

Migration Strategies

Adapter pattern

- Pros

Migration Strategies

Adapter pattern

- Pros
  - Flexible

Migration Strategies

Adapter pattern

- Pros
  - Flexible
  - Maximizes code reuse

Migration Strategies

Adapter pattern

- Pros
  - Flexible
  - Maximizes code reuse
  - Simplified interface

Migration Strategies

Adapter pattern

- Pros
  - Flexible
  - Maximizes code reuse
  - Simplified interface
  - Requires less maintenance

Migration Strategies

Adapter pattern

- Pros
  - Flexible
  - Maximizes code reuse
  - Simplified interface
  - Requires less maintenance
- Cons

Migration Strategies

Adapter pattern

- Pros
  - Flexible
  - Maximizes code reuse
  - Simplified interface
  - Requires less maintenance
- Cons
  - Additional code

Migration Strategies

Adapter pattern

- Pros
  - Flexible
  - Maximizes code reuse
  - Simplified interface
  - Requires less maintenance
- Cons
  - Additional code
- When to use?

Migration Strategies

Adapter pattern

- Pros
  - Flexible
  - Maximizes code reuse
  - Simplified interface
  - Requires less maintenance
- Cons
  - Additional code
- When to use?
  - Underlying APIs significantly different

Migration Strategies

Adapter using #define

```
#if TARGET_OS_IPHONE

#define XPlatformColor UIColor  // iOS

#elif TARGET_OS_MAC && !TARGET_OS_IPHONE

#define XPlatformColor NSColor  // OS X

#endif
```

Migration Strategies

Adapter using #define

```
#if TARGET_OS_IPHONE
#define XPlatformColor UIColor  // iOS
#elif TARGET_OS_MAC && !TARGET_OS_IPHONE
#define XPlatformColor NSColor  // OS X
#endif
```

Migration Strategies

Adapter using #define

```
#if TARGET_OS_IPHONE
#define XPlatformColor UIColor  // iOS
#elif TARGET_OS_MAC && !TARGET_OS_IPHONE
#define XPlatformColor NSColor  // OS X
#endif
```

Migration Strategies

Adapter using #define

Migration Strategies

Adapter using #define

- Pros

Migration Strategies

Adapter using #define

- Pros
  - Almost no new code

Migration Strategies

Adapter using #define

- Pros
  - Almost no new code
  - Compile time error checking

Migration Strategies

Adapter using #define

- Pros
  - Almost no new code
  - Compile time error checking
- Cons

Migration Strategies

Adapter using #define

- Pros
  - Almost no new code
  - Compile time error checking
- Cons
  - Only for supported classes!

Migration Strategies

Adapter using #define

- Pros
  - Almost no new code
  - Compile time error checking
- Cons
  - Only for supported classes!
    - UIColor and NSColor

Migration Strategies

Adapter using #define

- Pros
  - Almost no new code
  - Compile time error checking
- Cons
  - Only for supported classes!
    - UIColor and NSColor
    - UIFont and NSFont

Migration Strategies

Adapter using #define

- Pros
  - Almost no new code
  - Compile time error checking
- Cons
  - Only for supported classes!
    - UIColor and NSColor
    - UIFont and NSFont
    - UIImage and NSImage

Migration Strategies

Adapter using #define

- Pros
  - Almost no new code
  - Compile time error checking
- Cons
  - Only for supported classes!
    - UIColor and NSColor
    - UIFont and NSFont
    - UIImage and NSImage
    - UIBezierPath and NSBezierPath

Migration Strategies

Adapter using #define

- Pros
  - Almost no new code
  - Compile time error checking
- Cons
  - Only for supported classes!
    - UIColor and NSColor
    - UIFont and NSFont
    - UIImage and NSImage
    - UIBezierPath and NSBezierPath
  - Limited API coverage within supported classes

Migration Strategies

Adapter using #define

- Pros
  - Almost no new code
  - Compile time error checking
- Cons
  - Only for supported classes!
    - UIColor and NSColor
    - UIFont and NSFont
    - UIImage and NSImage
    - UIBezierPath and NSBezierPath
  - Limited API coverage within supported classes
  - Requires custom archiving

Migration Testing

Migration Testing

- Compiling code ≠ correct code

Migration Testing

- Compiling code ≠ correct code
- Unit testing

Migration Testing

- Compiling code ≠ correct code
- Unit testing
- Manual testing

Migration Testing

- Compiling code ≠ correct code
- Unit testing
- Manual testing
- Beta testing

Migration Testing

- Compiling code ≠ correct code
- Unit testing
- Manual testing
- Beta testing
- Testing plans

Demo

Finished OS X version

Dan Schimpf
Demo Monkey

Take Home Message

Take Home Message

1. Rethink your design

Take Home Message

1. Rethink your design
2. Restructure your code

Take Home Message

1. Rethink your design
2. Restructure your code
3. Get started

More Information

Jake Behrens
App Frameworks Evangelist
behrens@apple.com

Documentation
AppKit
http://developer.apple.com/mac

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

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

Related Sessions

Taking Control of Auto Layout in Xcode 5	Presidio Wednesday 10:15AM
Introduction to Sprite Kit	Presidio Wednesday 11:30AM
Best Practices for Cocoa Animation	Marina Wednesday 2:00PM
Introducing Text Kit	Presidio Wednesday 2:00PM

Labs

NSTableView, NSView, and Cocoa Lab	Frameworks Lab A Thursday 10:15AM
iOS to OS X Conversion Lab	Frameworks Lab A Thursday 11:30AM
Cocoa Animations, Drawing, and Cocoa Lab	Frameworks Lab A Friday 9:00AM

WWDC2013