Best Practices for Cocoa Animation

Because a stationary target is an easy target

Session 213

Chris Dreessen
AppKit Engineer

Peter Ammon
AppKit Engineer

Basic Animation

Custom Property Animations

Overriding Default Animations

Chaining Animations

Implicit Animation

Core Animation

NSStackView

Animate View Positions

Animate Constraints Directly

Animate Window Size Changes

Background

- See NSAnimatablePropertyContainer protocol in NSAnimation.h

```
@protocol NSAnimatablePropertyContainer

- (instancetype)animator;
- (NSDictionary *)animations;
- (void)setAnimations:(NSDictionary *)animations;
- (id)animationForKey:(NSString *)key;
+ (id)defaultAnimationForKey:(NSString *)key;

@end
```

Basic Animation

Custom Property Animations

Overriding Default Animations

Chaining Animations

Implicit Animation

Core Animation

NSStackView

Animate View Positions

Animate Constraints Directly

Animate Window Size Changes

Basic Animation

Custom Property Animations

Overriding Default Animations

Chaining Animations

Implicit Animation

Core Animation

NSStackView

Animate View Positions

Animate Constraints Directly

Animate Window Size Changes

Basic Animation

Animating a single property of a view

Basic Animation

- NSAnimatablePropertyContainer protocol in NSAnimation.h

```
@protocol NSAnimatablePropertyContainer

- (instancetype)animator;
- (NSDictionary *)animations;
- (void)setAnimations:(NSDictionary *)animations;
- (id)animationForKey:(NSString *)key;
+ (id)defaultAnimationForKey:(NSString *)key;

@end
```

Basic Animation

Animating a single property of a view

Basic Animation

Animating a single property of a view

```
view.animator.alphaValue = 0;
```

Basic Animation

Animating a single property of a view

Basic Animation

Animating a single property of a view

Basic Animation

Animating a single property of a view

Basic Animation

Animating a single property of a view

```
view.animator.frameOrigin = NSMakePoint(...);
```

Basic Animation

Animating a single property of a view

Basic Animation

Animating a single property of a view

Basic Animation

Animating a single property of a view

```
view.animator.frame = NSMakeRect(...);
```

Basic Animation

Custom Property Animations

Overriding Default Animations

Chaining Animations

Implicit Animation

Core Animation

NSStackView

Animate View Positions

Animate Constraints Directly

Animate Window Size Changes

Basic Animation

Custom Property Animations

Overriding Default Animations

Chaining Animations

Implicit Animation

Core Animation

NSStackView

Animate View Positions

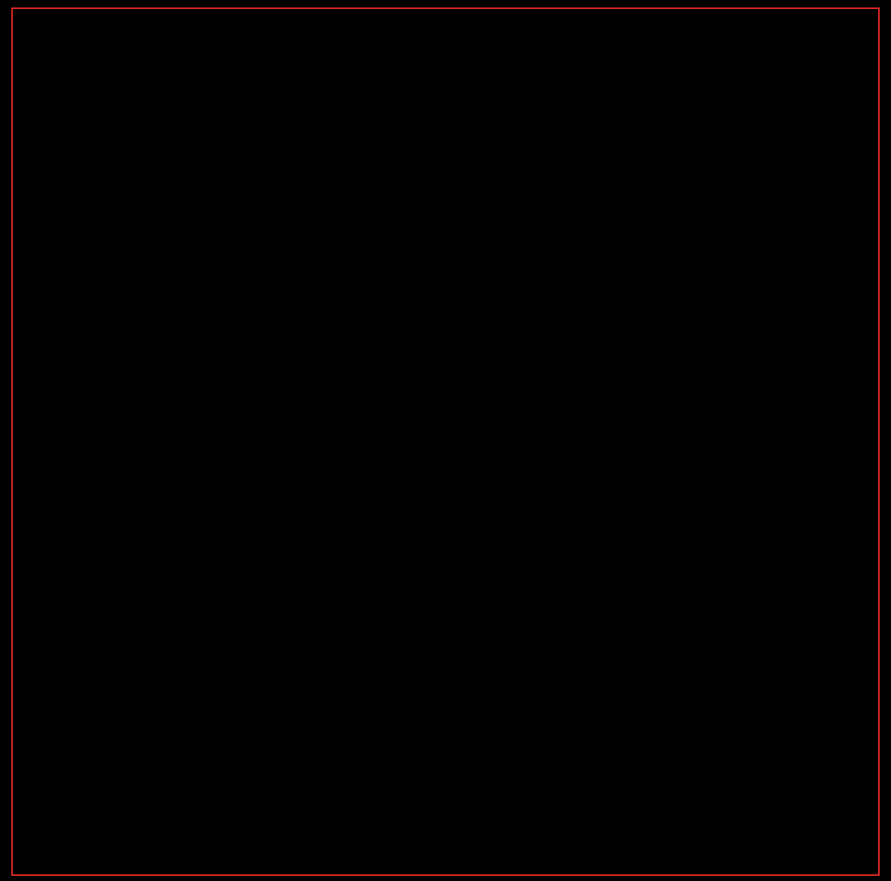
Animate Constraints Directly

Animate Window Size Changes

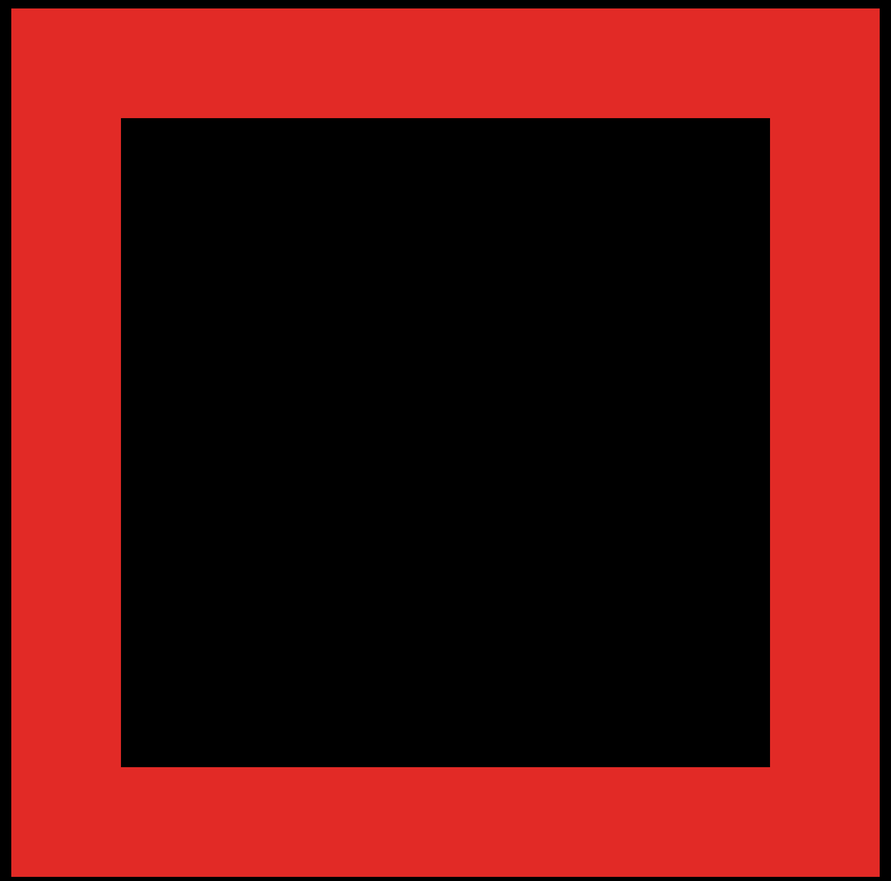
Custom Property Animations

How to make your own properties animatable

Custom Animatable Properties



Custom Animatable Properties



Custom Animatable Properties

```
@property CGFloat lineThickness;

- (void)drawRect:(NSRect) rect {
    [[NSColor redColor] set];
    NSFrameRectWithWidth(self.bounds, self.lineThickness);
}
```

```
- (void)setLineThickness:(CGFloat) thickness {
    _lineThickness = thickness;
    [self setNeedsDisplay:YES];
}

- (CGFloat)lineThickness {
    return _lineThickness;
}
```

Custom Animatable Properties

```
@property CGFloat lineThickness;

- (void)drawRect:(NSRect) rect {
    [[NSColor redColor] set];
    NSFrameRectWithWidth(self.bounds, self.lineThickness);
}

- (void)setLineThickness:(CGFloat) thickness {
    _lineThickness = thickness;
    [self setNeedsDisplay:YES];
}

- (CGFloat)lineThickness {
    return _lineThickness;
}
```

Custom Animatable Properties

```
@property CGFloat lineThickness;
```

```
- (void)drawRect:(NSRect) rect {
    [[NSColor redColor] set];
    NSFrameRectWithWidth(self.bounds, self.lineThickness);
}
```

```
- (void)setLineThickness:(CGFloat) thickness {
    _lineThickness = thickness;
    [self setNeedsDisplay:YES];
}

- (CGFloat)lineThickness {
    return _lineThickness;
}
```

Custom Animatable Properties

```
@property CGFloat lineThickness;

- (void)drawRect:(NSRect) rect {
    [[NSColor redColor] set];
    NSFrameRectWithWidth(self.bounds, self.lineThickness);
}
```

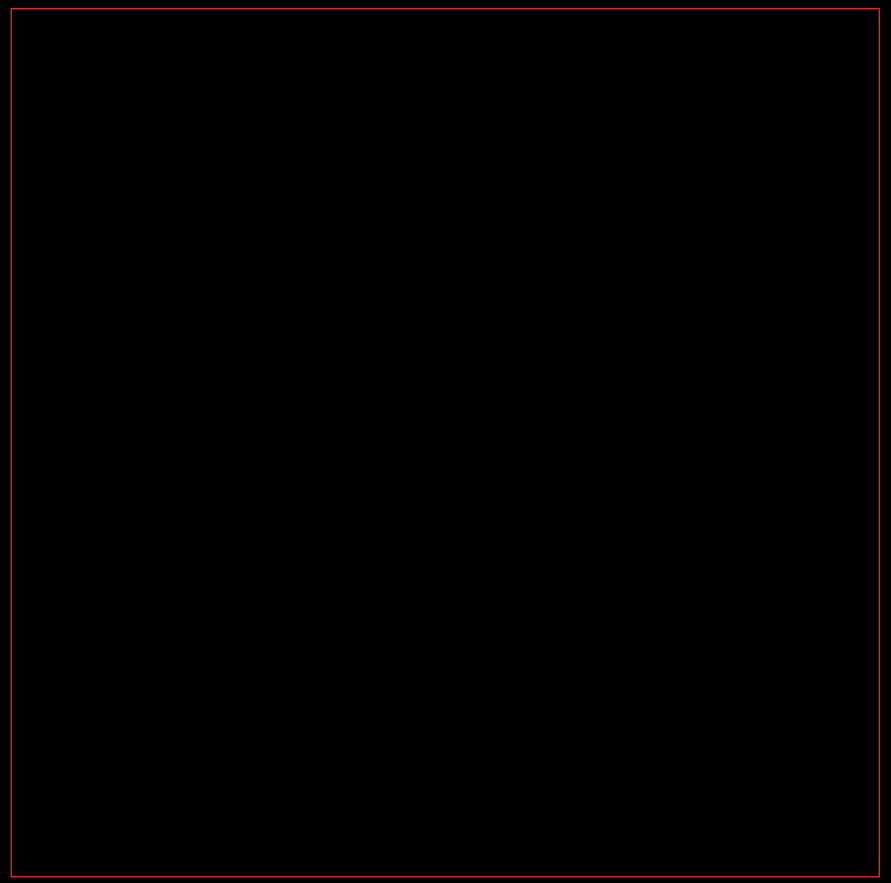
```
- (void)setLineThickness:(CGFloat) thickness {
    _lineThickness = thickness;
    [self setNeedsDisplay:YES];
}
```

```
- (CGFloat)lineThickness {
    return _lineThickness;
}
```

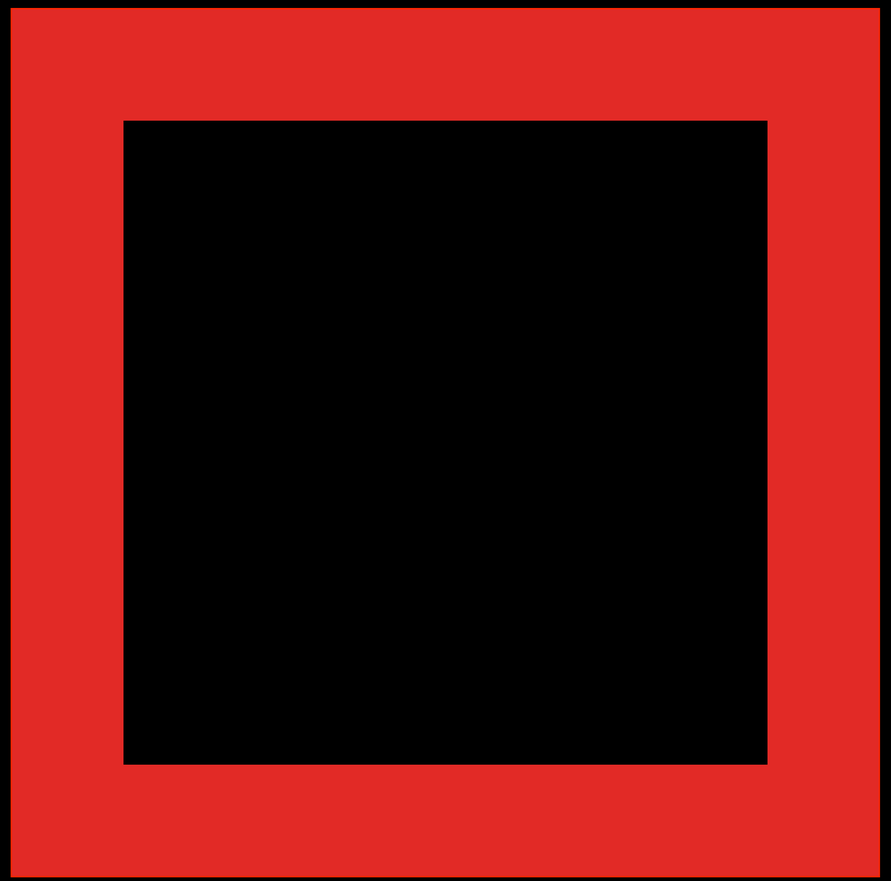
Custom Animatable Properties

```
view.animator.lineThickness = 10;
```

Custom Animatable Properties



Custom Animatable Properties



Custom Animatable Properties

- Back to NSAnimatablePropertyContainer

```
@protocol NSAnimatablePropertyContainer

- (instancetype)animator;
- (NSDictionary *)animations;
- (void)setAnimations:(NSDictionary *)animations;
- (id)animationForKey:(NSString *)key;
+ (id)defaultAnimationForKey:(NSString *)key;

@end
```

Custom Animatable Properties

- Back to NSAnimatablePropertyContainer

```
@protocol NSAnimatablePropertyContainer

- (instancetype)animator;
- (NSDictionary *)animations;
- (void)setAnimations:(NSDictionary *)animations;
- (id)animationForKey:(NSString *)key;
+ (id)defaultAnimationForKey:(NSString *)key;

@end
```

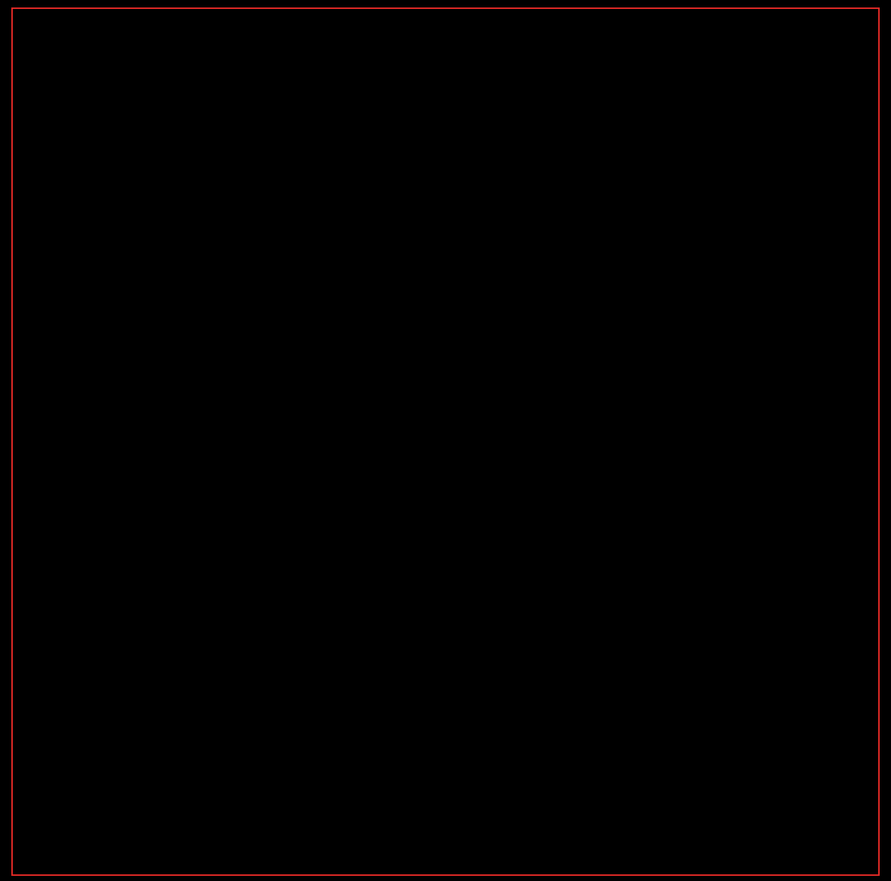
Custom Animatable Properties

```
+ (id)defaultAnimationForKey:(NSString *)key {
    if ([key isEqualToString:@”lineThickness”]) {
        return [CABasicAnimation animation];
    }

    return [super defaultAnimationForKey:key];
}
```

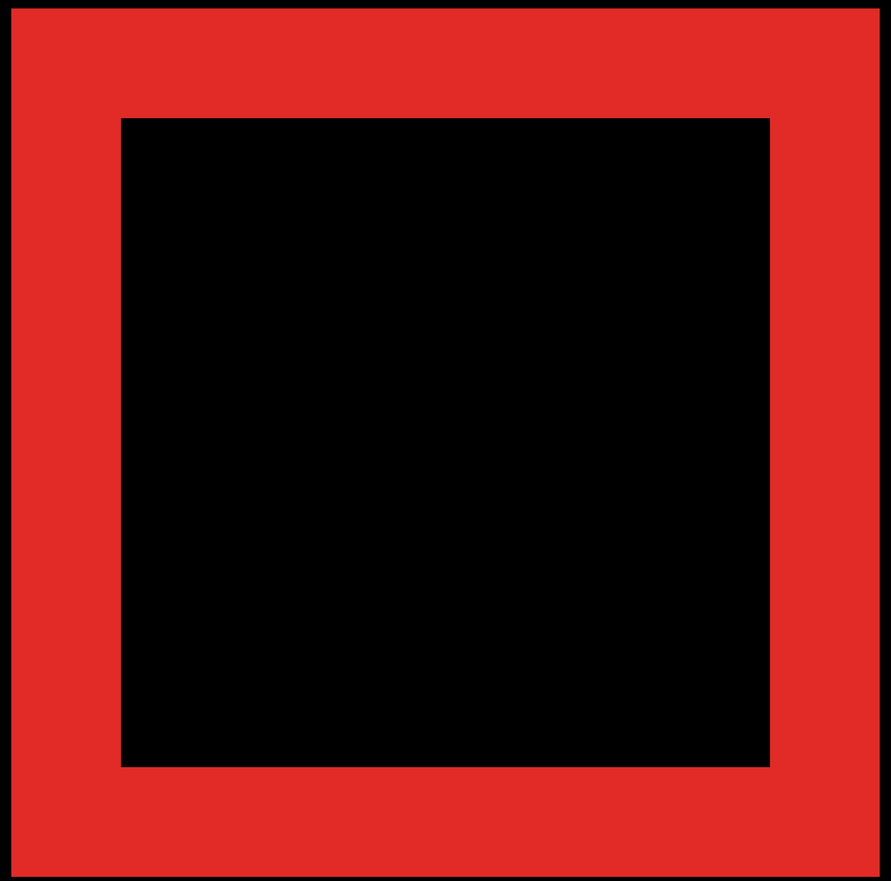
Custom Animatable Properties

```
+ (id)defaultAnimationForKey:(NSString *)key {
    if ([key isEqualToString:@"lineThickness"]) {
        return [CABasicAnimation animation];
    }

    return [super defaultAnimationForKey:key];
}
```

Custom Animatable Properties



Custom Animatable Properties



Basic Animation

Custom Property Animations

Overriding Default Animations

Chaining Animations

Implicit Animation

Core Animation

NSStackView

Animate View Positions

Animate Constraints Directly

Animate Window Size Changes

Basic Animation

Custom Property Animations

Overriding Default Animations

Chaining Animations

Implicit Animation

Core Animation

NSStackView

Animate View Positions

Animate Constraints Directly

Animate Window Size Changes

Overriding Default Animations

Because nothing is good the way it is

Overriding Default Animations

- Back to NSAnimatablePropertyContainer

```
@protocol NSAnimatablePropertyContainer

- (instancetype)animator;
- (NSDictionary *)animations;
- (void)setAnimations:(NSDictionary *)animations;
- (id)animationForKey:(NSString *)key;
+ (id)defaultAnimationForKey:(NSString *)key;

@end
```

Overriding Default Animations

- Back to NSAnimatablePropertyContainer

```
@protocol NSAnimatablePropertyContainer

- (instancetype)animator;
- (NSDictionary *)animations;
- (void)setAnimations:(NSDictionary *)animations;
- (id)animationForKey:(NSString *)key;
+ (id)defaultAnimationForKey:(NSString *)key;

@end
```

Overriding Default Animations

```
@property CGFloat lineThickness;

 + (id)defaultAnimationForKey:(NSString *)key {
   if ([key isEqualToString:@”lineThickness”]) {
      return [CABasicAnimation animation];
   }
   return [super defaultAnimationForKey:key];
}
@end
```

Overriding Default Animations

```
@property CGFloat lineThickness;

+ (id)defaultAnimationForKey:(NSString *)key {
   if ([key isEqualToString:@"lineThickness"]) {
      return [CABasicAnimation animation];
   }

   return [super defaultAnimationForKey:key];
}
@end
```

Overriding Default Animations

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[@0, @10, @20, @40];

view.animations = @{@”lineThickness”:kfa};

view.animator.lineThickness = 40;
```

Overriding Default Animations

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[@0, @10, @20, @40];

view.animations = @{@”lineThickness”:kfa};

view.animator.lineThickness = 40;
```

Overriding Default Animations

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[@0, @10, @20, @40];

view.animations = @{@”lineThickness”:kfa};

view.animator.lineThickness = 40;
```

Overriding Default Animations

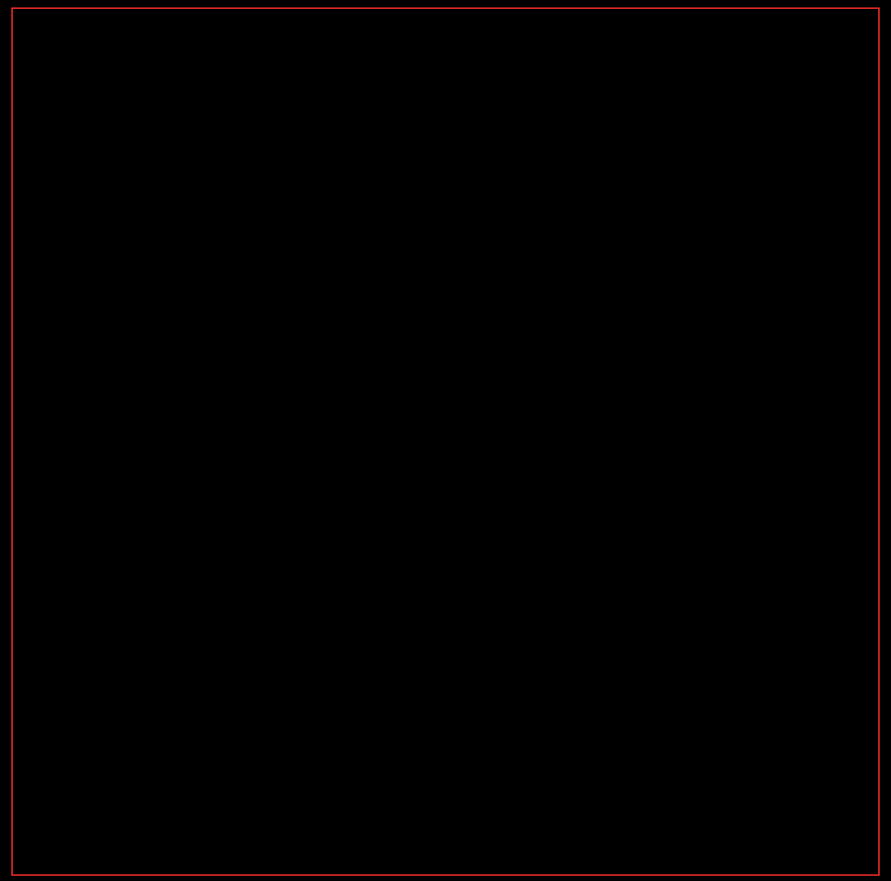
```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[@0, @10, @20, @40];

view.animations = @{@”lineThickness”:kfa};

view.animator.lineThickness = 40;
```

Overriding Default Animations

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[@0, @10, @20, @40];

view.animations = @{@”lineThickness”:kfa};

view.animator.lineThickness = 40;
```

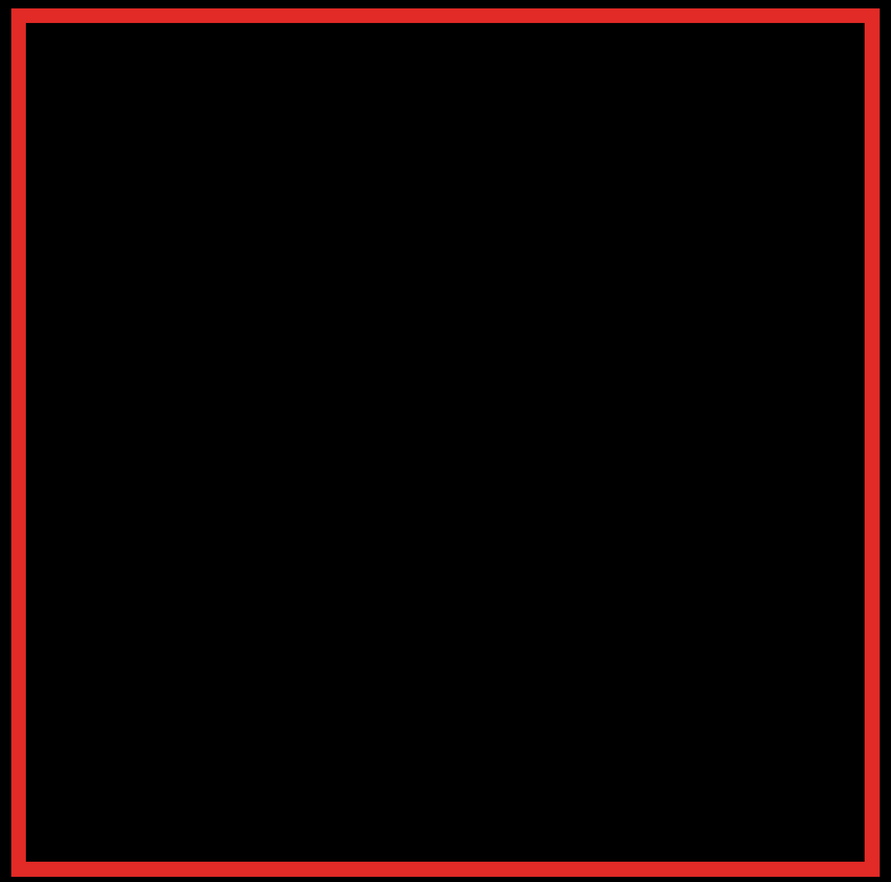
Overriding Default Animations

Animating a single property of a view



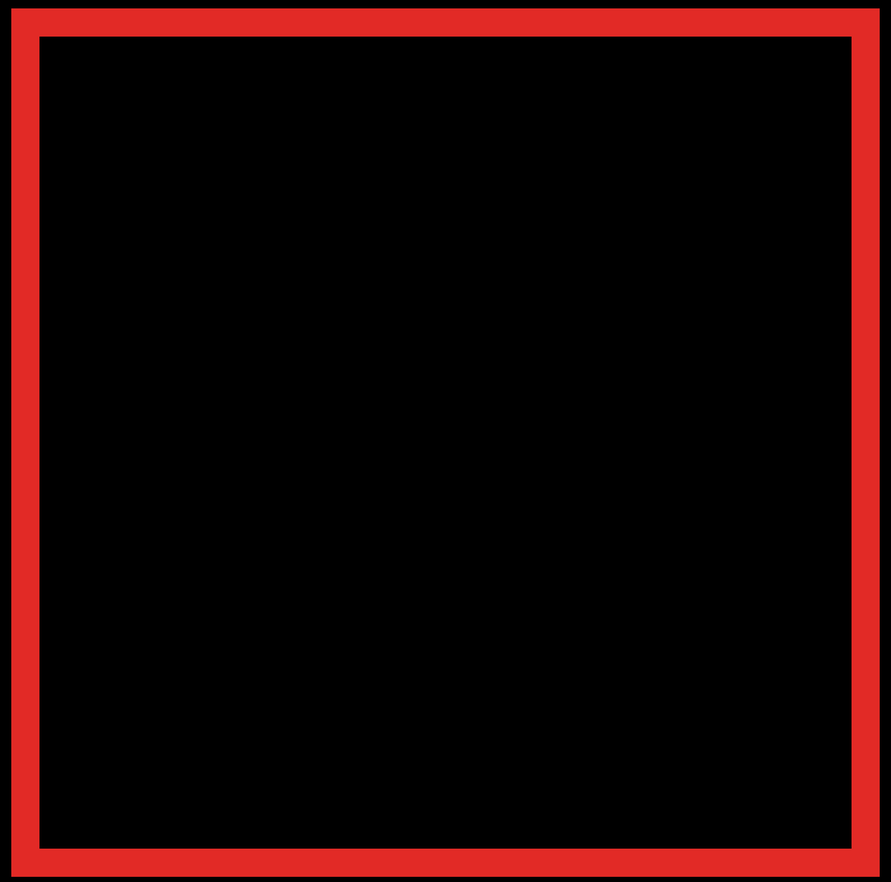
Overriding Default Animations

Animating a single property of a view



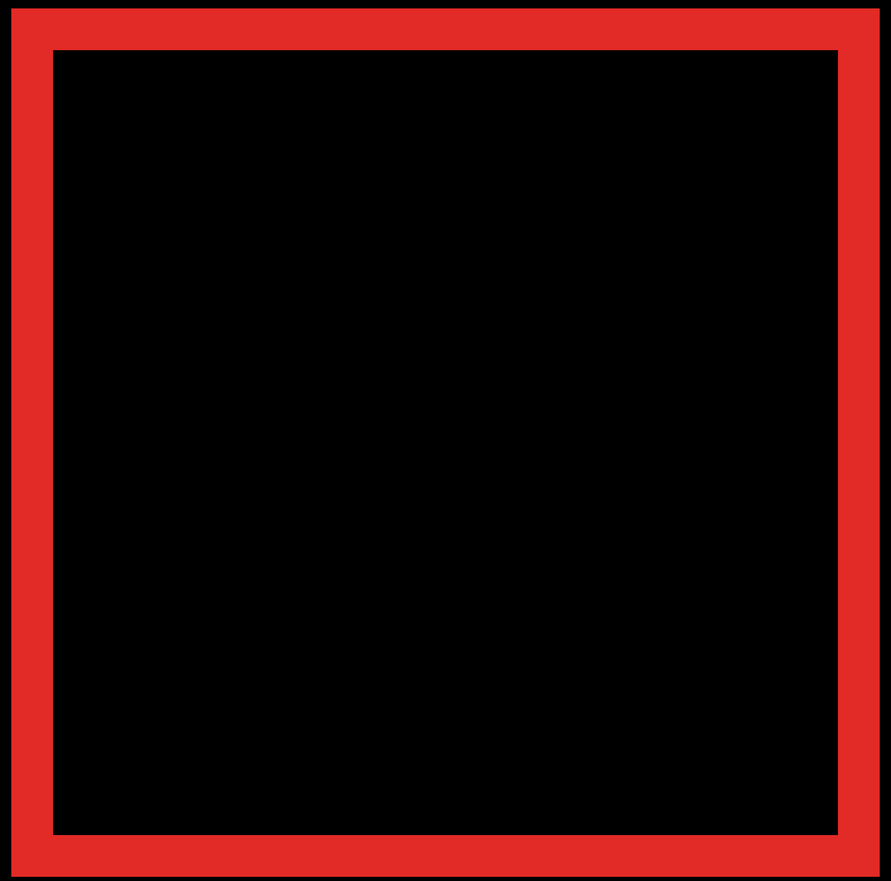
Overriding Default Animations

Animating a single property of a view



Overriding Default Animations

Animating a single property of a view



Overriding Default Animations

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[
    [NSImage imageNamed:@”iMac”],
    [NSImage imageNamed:@”iPhone”],
    [NSImage imageNamed:@”MacBook”],
    [NSImage imageNamed:@”iPad”],
    [NSImage imageNamed:@”MacMini”]
];

view.animations = @{@”image”:kfa};
view.animator.image = [NSImage imageNamed:@”AppleTV”];
```

Overriding Default Animations

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[
    [NSImage imageNamed:@”iMac”],
    [NSImage imageNamed:@”iPhone”],
    [NSImage imageNamed:@”MacBook”],
    [NSImage imageNamed:@”iPad”],
    [NSImage imageNamed:@”MacMini”]
];

view.animations = @{@”image”:kfa};
view.animator.image = [NSImage imageNamed:@”AppleTV”];
```

Overriding Default Animations

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[
    [NSImage imageNamed:@”iMac”],
    [NSImage imageNamed:@”iPhone”],
    [NSImage imageNamed:@”MacBook”],
    [NSImage imageNamed:@”iPad”],
    [NSImage imageNamed:@”MacMini”]
];

view.animations = @{@”image”:kfa};
view.animator.image = [NSImage imageNamed:@”AppleTV”];
```

Overriding Default Animations

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[
    [NSImage imageNamed:@”iMac”],
    [NSImage imageNamed:@”iPhone”],
    [NSImage imageNamed:@”MacBook”],
    [NSImage imageNamed:@”iPad”],
    [NSImage imageNamed:@”MacMini”]
];

view.animations = @{@”image”:kfa};
view.animator.image = [NSImage imageNamed:@”AppleTV”];
```

Overriding Default Animations

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[
    [NSImage imageNamed:@”iMac”],
    [NSImage imageNamed:@”iPhone”],
    [NSImage imageNamed:@”MacBook”],
    [NSImage imageNamed:@”iPad”],
    [NSImage imageNamed:@”MacMini”]
];

view.animations = @{@”image”:kfa};
view.animator.image = [NSImage imageNamed:@”AppleTV”];
```

Overriding a Default Animation

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[
    [NSImage imageNamed:@”iMac”],
    [NSImage imageNamed:@”iPhone”],
    [NSImage imageNamed:@”MacBook”],
    [NSImage imageNamed:@”iPad”],
    [NSImage imageNamed:@”MacMini”]
];

view.animations = @{@”image”:kfa};
view.animator.image = [NSImage imageNamed:@”AppleTV”];
```

Overriding a Default Animation

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[
    [NSImage imageNamed:@”iMac”],
    [NSImage imageNamed:@”iPhone”],
    [NSImage imageNamed:@”MacBook”],
    [NSImage imageNamed:@”iPad”],
    [NSImage imageNamed:@”MacMini”]
];

view.animations = @{@”image”:kfa};
view.animator.image = [NSImage imageNamed:@”AppleTV”];
```

Overriding a Default Animation

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[
    [NSImage imageNamed:@”iMac”],
    [NSImage imageNamed:@”iPhone”],
    [NSImage imageNamed:@”MacBook”],
    [NSImage imageNamed:@”iPad”],
    [NSImage imageNamed:@”MacMini”]
];

view.animations = @{@”image”:kfa};
view.animator.image = [NSImage imageNamed:@”AppleTV”];
```

Overriding a Default Animation

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[
    [NSImage imageNamed:@”iMac”],
    [NSImage imageNamed:@”iPhone”],
    [NSImage imageNamed:@”MacBook”],
    [NSImage imageNamed:@”iPad”],
    [NSImage imageNamed:@”MacMini”]
];

view.animations = @{@”image”:kfa};
view.animator.image = [NSImage imageNamed:@”AppleTV”];
```

Overriding a Default Animation

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[
    [NSImage imageNamed:@”iMac”],
    [NSImage imageNamed:@”iPhone”],
    [NSImage imageNamed:@”MacBook”],
    [NSImage imageNamed:@”iPad”],
    [NSImage imageNamed:@”MacMini”]
];

view.animations = @{@”image”:kfa};
view.animator.image = [NSImage imageNamed:@”AppleTV”];
```

Overriding a Default Animation

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[
    [NSImage imageNamed:@”iMac”],
    [NSImage imageNamed:@”iPhone”],
    [NSImage imageNamed:@”MacBook”],
    [NSImage imageNamed:@”iPad”],
    [NSImage imageNamed:@”MacMini”]
];

view.animations = @{@”image”:kfa};
view.animator.image = [NSImage imageNamed:@”AppleTV”];
```

Overriding Default Animations

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[
    @”Seattle”,
    @”Chicago”,
    @”New York”,
    @”Boston”,
    @”Philadelphia”
];

textfield.animations = @{@”stringValue”:kfa};
textfield.animator.stringValue = @”San Francisco”;
```

Overriding Default Animations

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[
    @”Seattle”,
    @”Chicago”,
    @”New York”,
    @”Boston”,
    @”Philadelphia”
];

textfield.animations = @{@”stringValue”:kfa};
textfield.animator.stringValue = @”San Francisco”;
```

Overriding Default Animations

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[
    @”Seattle”,
    @”Chicago”,
    @”New York”,
    @”Boston”,
    @”Philadelphia”
];

textfield.animations = @{@”stringValue”:kfa};
textfield.animator.stringValue = @”San Francisco”;
```

Overriding Default Animations

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[
    @”Seattle”,
    @”Chicago”,
    @”New York”,
    @”Boston”,
    @”Philadelphia”
];

textfield.animations = @{@”stringValue”:kfa};
textfield.animator.stringValue = @”San Francisco”;
```

Overriding Default Animations

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[
    @”Seattle”,
    @”Chicago”,
    @”New York”,
    @”Boston”,
    @”Philadelphia”
];

textfield.animations = @{@”stringValue”:kfa};
textfield.animator.stringValue = @”San Francisco”;
```

Overriding Default Animations

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[
    @”Seattle”,
    @”Chicago”,
    @”New York”,
    @”Boston”,
    @”Philadelphia”
];

textfield.animations = @{@”stringValue”:kfa};
textfield.animator.stringValue = @”San Francisco”;
```

Overriding Default Animations

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[
    @”Seattle”,
    @”Chicago”,
    @”New York”,
    @”Boston”,
    @”Philadelphia”
];

textfield.animations = @{@”stringValue”:kfa};
textfield.animator.stringValue = @”San Francisco”;
```

Overriding Default Animations

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[
   @”Seattle”,
   @”Chicago”,
   @”New York”,
   @”Boston”,
   @”Philadelphia”
];

textfield.animations = @{@”stringValue”:kfa};
textfield.animator.stringValue = @”San Francisco”;
```

Overriding Default Animations

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[
    @”Seattle”,
    @”Chicago”,
    @”New York”,
    @”Boston”,
    @”Philadelphia”
];

textfield.animations = @{@”stringValue”:kfa};
textfield.animator.stringValue = @”San Francisco”;
```

Overriding Default Animations

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[
   @”Seattle”,
   @”Chicago”,
   @”New York”,
   @”Boston”,
   @”Philadelphia”
];

textfield.animations = @{@”stringValue”:kfa};
textfield.animator.stringValue = @”San Francisco”;
```

Philadelphia

Overriding Default Animations

```
CAKeyframeAnimation *kfa = [CAKeyframeAnimation animation];
kfa.values = @[
    @”Seattle”,
    @”Chicago”,
    @”New York”,
    @”Boston”,
    @”Philadelphia”
];

textfield.animations = @{@”stringValue”:kfa};
textfield.animator.stringValue = @”San Francisco”;
```

Overriding Default Animations

```
@interface MyFormatter : NSFormatter
@property NSDateFormatter *dateFormatter;
@end

@implementation MyFormatter
- (NSString *)stringForObjectValue:(id)obj
{
    NSTimeInterval tisrd = [(NSNumber *)obj doubleValue];
    NSDate *date = [NSDate dateWithTimeIntervalSinceReferenceDate:tisrd];

    return [self.dateFormatter stringForObjectValue:date];
}
@end
```

Overriding Default Animations

```
@interface MyFormatter : NSFormatter
@property NSDateFormatter *dateFormatter;
@end

@implementation MyFormatter
- (NSString *)stringForObjectValue:(id)obj
{
    NSTimeInterval tisrd = [(NSNumber *)obj doubleValue];
    NSDate *date = [NSDate dateWithTimeIntervalSinceReferenceDate:tisrd];

    return [self.dateFormatter stringForObjectValue:date];
}
@end
```

Overriding Default Animations

```
@interface MyFormatter : NSFormatter
@property NSDateFormatter *dateFormatter;
@end

@implementation MyFormatter
- (NSString *)stringForObjectValue:(id)obj
{
    NSTimeInterval tisrd = [(NSNumber *)obj doubleValue];
    NSDate *date = [NSDate dateWithTimeIntervalSinceReferenceDate:tisrd];

    return [self.dateFormatter stringForObjectValue:date];
}
@end
```

Overriding Default Animations

```
@interface MyFormatter : NSFormatter
@property NSDateFormatter *dateFormatter;
@end

@implementation MyFormatter
- (NSString *)stringForObjectValue:(id)obj
{
    NSTimeInterval tisrd = [(NSNumber *)obj doubleValue];
    NSDate *date = [NSDate dateWithTimeIntervalSinceReferenceDate:tisrd];

    return [self.dateFormatter stringForObjectValue:date];
}
@end
```

Overriding Default Animations

```
@interface MyFormatter : NSFormatter
@property NSDateFormatter *dateFormatter;
@end

@implementation MyFormatter
- (NSString *)stringForObjectValue:(id)obj
{
    NSTimeInterval tisrd = [(NSNumber *)obj doubleValue];
    NSDate *date = [NSDate dateWithTimeIntervalSinceReferenceDate:tisrd];

    return [self.dateFormatter stringForObjectValue:date];
}
@end
```

Overriding Default Animations

```
@interface MyFormatter : NSFormatter
@property NSDateFormatter *dateFormatter;
@end

@implementation MyFormatter
- (NSString *)stringForObjectValue:(id)obj
{
    NSTimeInterval tisrd = [(NSNumber *)obj doubleValue];
    NSDate *date = [NSDate dateWithTimeIntervalSinceReferenceDate:tisrd];

    return [self.dateFormatter stringForObjectValue:date];
}
@end
```

Overriding Default Animations

```
@interface MyFormatter : NSFormatter
@property NSDateFormatter *dateFormatter;
@end

@implementation MyFormatter
- (NSString *)stringForObjectValue:(id)obj
{
    NSTimeInterval tisrd = [(NSNumber *)obj doubleValue];
    NSDate *date = [NSDate dateWithTimeIntervalSinceReferenceDate:tisrd];

    return [self.dateFormatter stringForObjectValue:date];
}
@end
```

Overriding Default Animations

```
- (void)animate:(id)sender {
    [NSAnimationContext runAnimationGroup:^(NSAnimationContext *context) {
        context.duration = 20;
        self.textField.animations = @{
            @"doubleValue" : [CABasicAnimation animation]
        };
        self.textField.doubleValue = 0;
        self.textField.animator.doubleValue = 1000000;
    } completionHandler:NULL];
}
```

Overriding Default Animations

```
- (void)animate:(id)sender {
    [NSAnimationContext runAnimationGroup:^(NSAnimationContext *context) {
        context.duration = 20;
        self.textField.animations = @{
            @"doubleValue" : [CABasicAnimation animation]
        };
        self.textField.doubleValue = 0;
        self.textField.animator.doubleValue = 1000000;
    } completionHandler:NULL];
}
```

Overriding Default Animations

```
- (void)animate:(id)sender {
    [NSAnimationContext runAnimationGroup:^(NSAnimationContext *context) {
        context.duration = 20;
        self.textField.animations = @{
            @"doubleValue" : [CABasicAnimation animation]
        };
        self.textField.doubleValue = 0;
        self.textField.animator.doubleValue = 1000000;
    } completionHandler:NULL];
}
```

Overriding Default Animations

```
- (void)animate:(id)sender {
    [NSAnimationContext runAnimationGroup:^(NSAnimationContext *context) {
        context.duration = 20;
        self.textField.animations = @{
            @"doubleValue" : [CABasicAnimation animation]
        };
        self.textField.doubleValue = 0;
        self.textField.animator.doubleValue = 1000000;
    } completionHandler:NULL];
}
```

Overriding Default Animations

```
- (void)animate:(id)sender {
    [NSAnimationContext runAnimationGroup:^(NSAnimationContext *context) {
        context.duration = 20;
        self.textField.animations = @{
            @"doubleValue" : [CABasicAnimation animation]
        };
        self.textField.doubleValue = 0;
        self.textField.animator.doubleValue = 1000000;
    } completionHandler:NULL];
}
```

Overriding Default Animations

```
- (void)animate:(id)sender {
    [NSAnimationContext runAnimationGroup:^(NSAnimationContext *context) {
        context.duration = 20;
        self.textField.animations = @{
            @"doubleValue" : [CABasicAnimation animation]
        };
        self.textField.doubleValue = 0;
        self.textField.animator.doubleValue = 1000000;
    } completionHandler:NULL];
}
```

Overriding Default Animations

```
- (void)animate:(id)sender {
    [NSAnimationContext runAnimationGroup:^(NSAnimationContext *context) {
        context.duration = 20;
        self.textField.animations = @{
            @"doubleValue" : [CABasicAnimation animation]
        };
        self.textField.doubleValue = 0;
        self.textField.animator.doubleValue = 1000000;
    } completionHandler:NULL];
}
```

Sunday, December 31, 2000 at 4:00:00 PM Pacific Standard Time

Basic Animation

Custom Property Animations

Overriding Default Animations

Chaining Animations

Implicit Animation

Core Animation

NSStackView

Animate View Positions

Animate Constraints Directly

Animate Window Size Changes

Basic Animation

Custom Property Animations

Overriding Default Animations

Chaining Animations

Implicit Animation

Core Animation

NSStackView

Animate View Positions

Animate Constraints Directly

Animate Window Size Changes

Chaining Animations

When one animation isn’t good enough

Chaining animations

```
- (void)windowDidLoad {
    [NSAnimationContext runAnimationGroup:^(NSAnimationContext *){
        view1.animations = frameAnimations;
        view1.animator.frame = toValue;
    } completionHandler:^{
        [NSAnimationContext runAnimationGroup:^(NSAnimationContext *){
            view2.animations = frameAnimations;
            view2.animator.frame = toValue;
        } completionHandler:^{
            [NSAnimationContext runAnimationGroup:^(NSAnimationContext *){
                view3.animations = frameAnimations;
                view3.animator.frame = toValue;
            } completionHandler:^{
            }];
        }];
    }];
}
```

Chaining animations

```
- (void)windowDidLoad {
    [NSAnimationContext runAnimationGroup:^(NSAnimationContext *){
        view1.animations = frameAnimations;
        view1.animator.frame = toValue;
    } completionHandler:^{
        [NSAnimationContext runAnimationGroup:^(NSAnimationContext *){
            view2.animations = frameAnimations;
            view2.animator.frame = toValue;
        } completionHandler:^{
            [NSAnimationContext runAnimationGroup:^(NSAnimationContext *){
                view3.animations = frameAnimations;
                view3.animator.frame = toValue;
            } completionHandler:^{
            }];
        }];
    }];
}
```

Chaining animations

```
- (void)windowDidLoad {
    [NSAnimationContext runAnimationGroup:^(NSAnimationContext *){
        view1.animations = frameAnimations;
        view1.animator.frame = toValue;
    } completionHandler:^{
        [NSAnimationContext runAnimationGroup:^(NSAnimationContext *){
            view2.animations = frameAnimations;
            view2.animator.frame = toValue;
        } completionHandler:^{
            [NSAnimationContext runAnimationGroup:^(NSAnimationContext *){
                view3.animations = frameAnimations;
                view3.animator.frame = toValue;
            } completionHandler:^{
            }];
        }];
    }];
}
```

Chaining animations

```
- (void)windowDidLoad {
    [NSAnimationContext runAnimationGroup:^(NSAnimationContext *){
        view1.animations = frameAnimations;
        view1.animator.frame = toValue;
    } completionHandler:^{
        [NSAnimationContext runAnimationGroup:^(NSAnimationContext *){
            view2.animations = frameAnimations;
            view2.animator.frame = toValue;
        } completionHandler:^{
            [NSAnimationContext runAnimationGroup:^(NSAnimationContext *){
                view3.animations = frameAnimations;
                view3.animator.frame = toValue;
            } completionHandler:^{
            }];
        }];
    }];
}
```

Chaining animations

```
- (void)windowDidLoad {
    [NSAnimationContext runAnimationGroup:^(NSAnimationContext *){
        view1.animations = frameAnimations;
        view1.animator.frame = toValue;
    } completionHandler:^{
        [NSAnimationContext runAnimationGroup:^(NSAnimationContext *){
            view2.animations = frameAnimations;
            view2.animator.frame = toValue;
        } completionHandler:^{
            [NSAnimationContext runAnimationGroup:^(NSAnimationContext *){
                view3.animations = frameAnimations;
                view3.animator.frame = toValue;
            } completionHandler:^{
            }];
        }];
    }];
}
```

Chaining animations

```
- (void)windowDidLoad {
    [NSAnimationContext runAnimationGroup:^(NSAnimationContext *){
        view1.animations = frameAnimations;
        view1.animator.frame = toValue;
    } completionHandler:^{
        [NSAnimationContext runAnimationGroup:^(NSAnimationContext *){
            view2.animations = frameAnimations;
            view2.animator.frame = toValue;
        } completionHandler:^{
            [NSAnimationContext runAnimationGroup:^(NSAnimationContext *){
                view3.animations = frameAnimations;
                view3.animator.frame = toValue;
            } completionHandler:^{
            }];
        }];
    }];
}
```

Chaining animations

```
- (void)windowDidLoad {
    [NSAnimationContext runAnimationGroup:^(NSAnimationContext *){
        view1.animations = frameAnimations;
        view1.animator.frame = toValue;
    } completionHandler:^{
        [NSAnimationContext runAnimationGroup:^(NSAnimationContext *){
            view2.animations = frameAnimations;
            view2.animator.frame = toValue;
        } completionHandler:^{
            [NSAnimationContext runAnimationGroup:^(NSAnimationContext *){
                view3.animations = frameAnimations;
                view3.animator.frame = toValue;
            } completionHandler:^{
            }];
        }];
    }];
}
```

Chaining animations

```
- (void)windowDidLoad {
    [NSAnimationContext runAnimationGroup:^(NSAnimationContext *){
        view1.animations = frameAnimations;
        view1.animator.frame = toValue;
    } completionHandler:^{
        [NSAnimationContext runAnimationGroup:^(NSAnimationContext *){
            view2.animations = frameAnimations;
            view2.animator.frame = toValue;
        } completionHandler:^{
            [NSAnimationContext runAnimationGroup:^(NSAnimationContext *){
                view3.animations = frameAnimations;
                view3.animator.frame = toValue;
            } completionHandler:^{
            }];
        }];
    }];
}
```

Demo

Basic Animation

Custom Property Animations

Overriding Default Animations

Chaining Animations

Implicit Animation

Core Animation

NSStackView

Animate View Positions

Animate Constraints Directly

Animate Window Size Changes

Basic Animation

Custom Property Animations

Overriding Default Animations

Chaining Animations

Implicit Animation

Core Animation

NSStackView

Animate View Positions

Animate Constraints Directly

Animate Window Size Changes

Implicit Animation

Do what I mean, not what I say

Implicit Animation

- 10.8 added NSAnimationContext.allowsImplicitAnimation

Implicit Animation

- 10.8 added NSAnimationContext.allowsImplicitAnimation

```
view.animator.frame = NSMakeRect(...);
```

Implicit Animation

- 10.8 added NSAnimationContext.allowsImplicitAnimation

```
NSAnimationContext.currentContext.allowsImplicitAnimation = YES;
view.frame = NSMakeRect(...);
```

Implicit Animation

- 10.8 added NSAnimationContext.allowsImplicitAnimation
- Animation is a side effect of directly setting a property

Implicit Animation

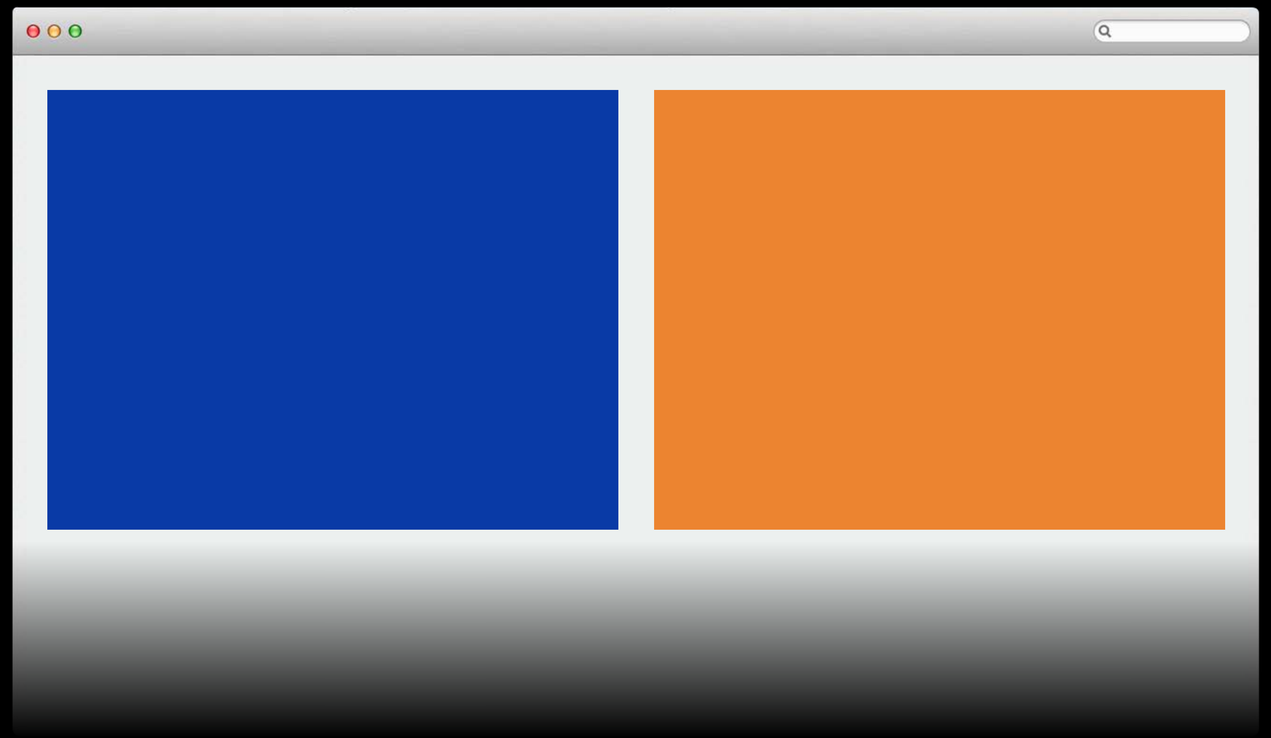
- 10.8 added NSAnimationContext.allowsImplicitAnimation
- Animation is a side effect of directly setting a property

```
- (void)swapSubviewFrames:(id)sender
{
    NSRect frame0 = self.view.subviews[0].frame;
    NSRect frame1 = self.view.subviews[1].frame;
    self.view.subviews[0].frame = frame1;
    self.view.subviews[1].frame = frame0;
}
```

Implicit Animation

- 10.8 added NSAnimationContext.allowsImplicitAnimation
- Animation is a side effect of directly setting a property

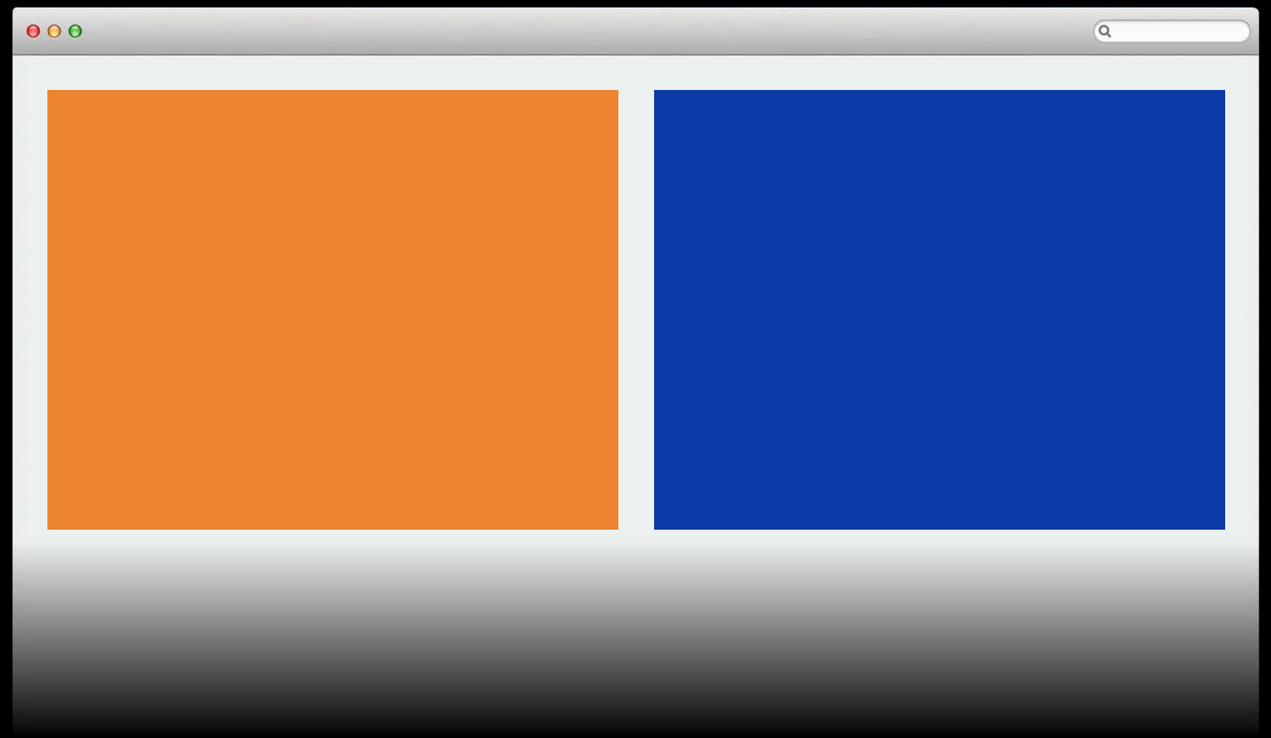
```
[myViewController swapSubviewsFrames:nil];
```



Implicit Animation

- 10.8 added NSAnimationContext.allowsImplicitAnimation
- Animation is a side effect of directly setting a property

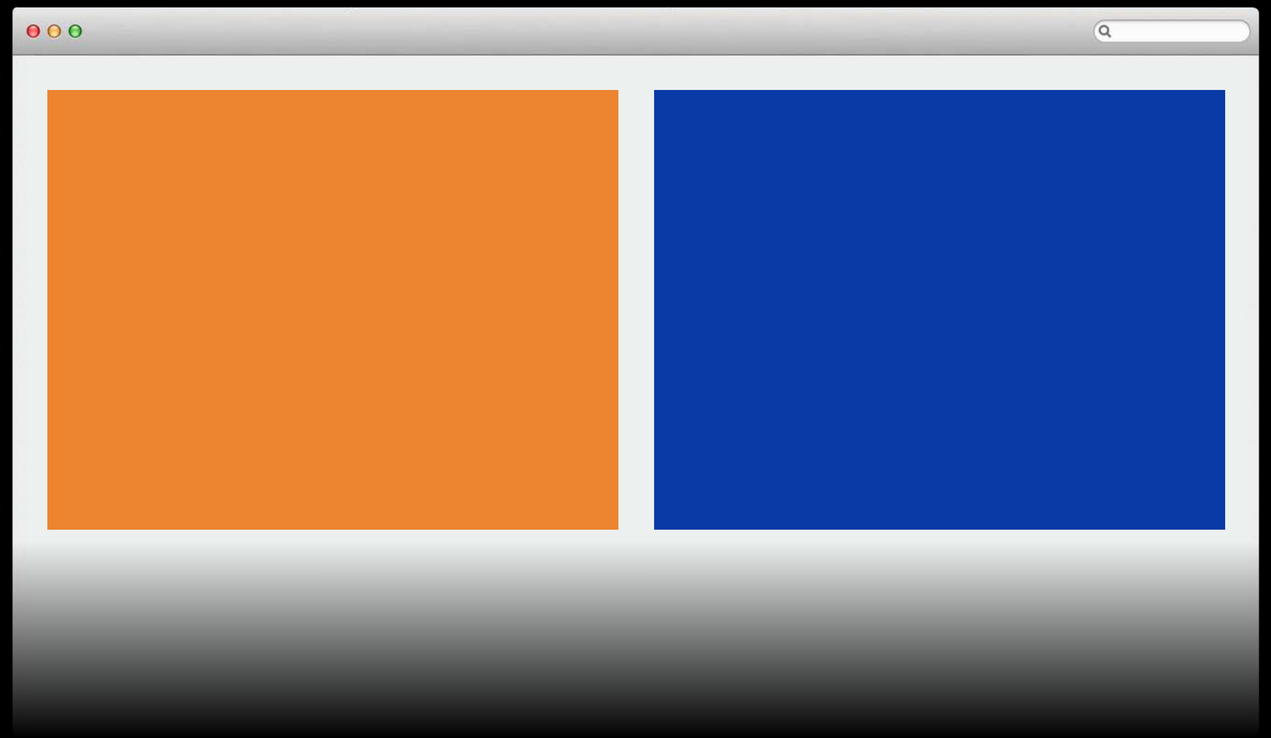
```
[myViewController swapSubviewsFrames:nil];
```



Implicit Animation

- 10.8 added NSAnimationContext.allowsImplicitAnimation
- Animation is a side effect of directly setting a property

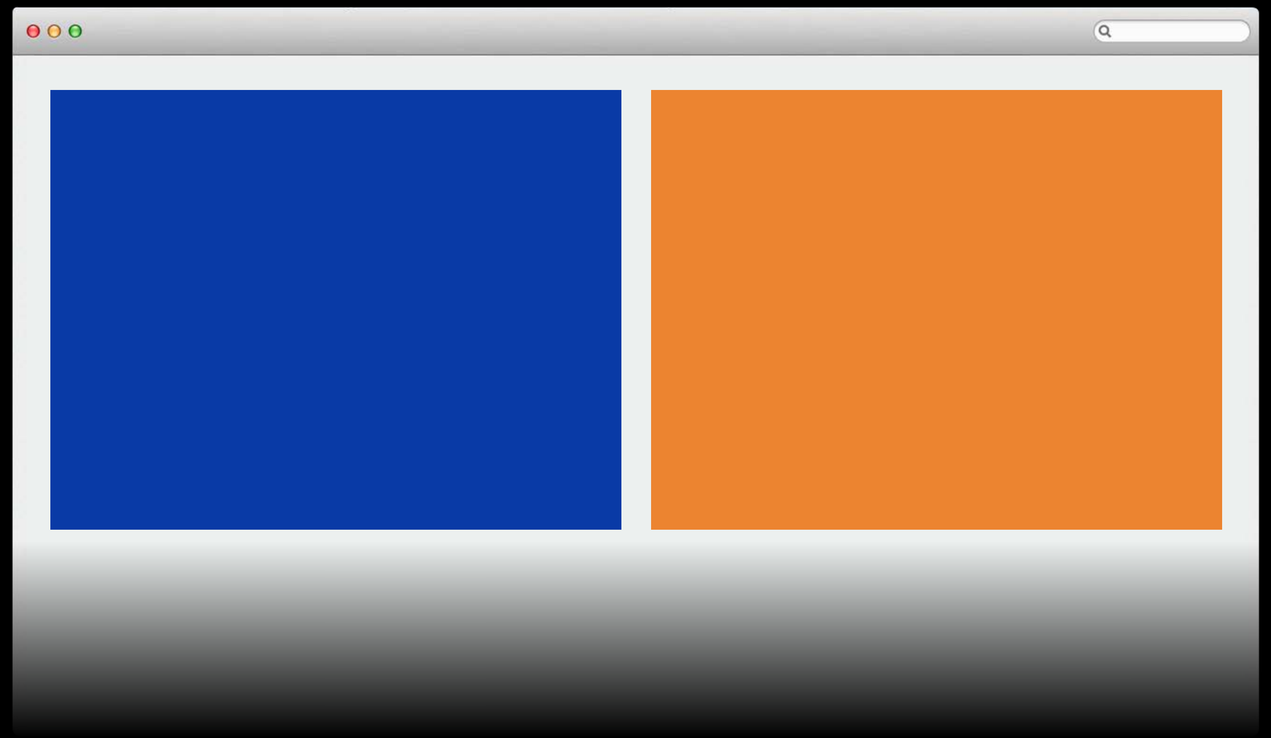
```
NSAnimationContext.currentContext.allowsImplicitAnimation = YES;
[myViewController swapSubviewsFrames:nil];
```



Implicit Animation

- 10.8 added NSAnimationContext.allowsImplicitAnimation
- Animation is a side effect of directly setting a property

```
NSAnimationContext.currentContext.allowsImplicitAnimation = YES;
[myViewController swapSubviewsFrames:nil];
```



Implicit Animation

- 10.8 added NSAnimationContext.allowsImplicitAnimation
- Animation is a side effect of directly setting a property
- Can be used to animate properties not accessible to the animator proxy

Implicit Animation

- 10.8 added NSAnimationContext.allowsImplicitAnimation
- Animation is a side effect of directly setting a property
- Can be used to animate properties not accessible to the animator proxy
- Only works for some properties
  - frame
  - frameSize
  - frameOrigin
- Works for more properties when a view is “layer-backed”

Basic Animation

Custom Property Animations

Overriding Default Animations

Chaining Animations

Implicit Animation

Core Animation

NSStackView

Animate View Positions

Animate Constraints Directly

Animate Window Size Changes

Basic Animation

Custom Property Animations

Overriding Default Animations

Chaining Animations

Implicit Animation

Core Animation

NSStackView

Animate View Positions

Animate Constraints Directly

Animate Window Size Changes

Core Animation

Animate your core!

Core Animation

Explicit animation

Core Animation

Explicit animation

- CALayer contains many properties, most of which are animatable
  - bounds, position, opacity, etc.

Core Animation

Explicit animation

- CALayer contains many properties, most of which are animatable
  - bounds, position, opacity, etc.
- Animations can be added explicitly

```
–[CALayer addAnimation:forKey:]
```

Core Animation

Explicit animation

- CALayer contains many properties, most of which are animatable
  - bounds, position, opacity, etc.
- Animations can be added explicitly
  - `[CALayer addAnimation:forKey:]`

```
CABasicAnimation *animation = [CABasicAnimation animation];
animation.fromValue = ...;
animation.toValue = ...;

[layer addAnimation:animation forKey:@”property”];
```

Core Animation

Explicit animation

Core Animation

Explicit animation

- Animations are nondestructive

Core Animation

Explicit animation

- Animations are nondestructive
- Animations temporarily override properties of a layer for rendering

Core Animation

Explicit animation

- Animations are nondestructive
- Animations temporarily override properties of a layer for rendering

```
CABasicAnimation *animation = [CABasicAnimation animation];
animation.fromValue = @.75;
animation.toValue = @.25;

[layer addAnimation:animation forKey:@”opacity”];


NSLog(@”%f”, layer.opacity);
```

Core Animation

Explicit animation

- Animations are nondestructive
- Animations temporarily override properties of a layer for rendering

```
CABasicAnimation *animation = [CABasicAnimation animation];
animation.fromValue = @.75;
animation.toValue = @.25;
```

```
[layer addAnimation:animation forKey:@”opacity”];
```

```
NSLog(@”%f”, layer.opacity); ➔ 1.0
```

Core Animation

Implicit animation

Core Animation

Implicit animation

- Animations are added in response to property changes

Core Animation

Implicit animation

- Animations are added in response to property changes
- If you write this:

```
layer.opacity = 0.5;
```

Core Animation

Implicit animation

- Animations are added in response to property changes
- If you write this:

```
layer.opacity = 0.5;
```

- You're really getting all of this:

```
layer.opacity = 0.5;
CABasicAnimation *animation =[CABasicAnimation animation];
animation.toValue = @.5;
[layer addAnimation:animation forKey:opacity];
```

Core Animation

Implicit animation

- Animations are added in response to property changes
- If you write this:

```
layer.opacity = 0.5;
```

- You're really getting all of this:

```
layer.opacity = 0.5;
CABasicAnimation *animation =[CABasicAnimation animation];
animation.toValue = @.5;
[layer addAnimation:animation forKey:opacity];
```

- The details of this are governed by the CAAction protocol

Core Animation

Layer-backed views

Core Animation

Layer-backed views

- Views can delegate compositing and animating to Core Animation

Core Animation

Layer-backed views

- Views can delegate compositing and animating to Core Animation
- Setting the wantsLayer property of an NSView to YES opts in

Core Animation

Layer-backed views

- Views can delegate compositing and animating to Core Animation
- Setting the wantsLayer property of an NSView to YES opts in
  - The view now manages a layer

Core Animation

Layer-backed views

- Views can delegate compositing and animating to Core Animation
- Setting the wantsLayer property of an NSView to YES opts in
  - The view now manages a layer
  - All descendants of a view manage their own layers, too

Core Animation

Layer-backed views

- Views can delegate compositing and animating to Core Animation
- Setting the wantsLayer property of an NSView to YES opts in
  - The view now manages a layer
  - All descendants of a view manage their own layers, too
- This offers different performance and memory tradeoffs

Core Animation

Layer-backed views

- Views can delegate compositing and animating to Core Animation
- Setting the wantsLayer property of an NSView to YES opts in
  - The view now manages a layer
  - All descendants of a view manage their own layers, too
- This offers different performance and memory tradeoffs
- Things behave…differently

Core Animation

Layer-backed views

- Views can delegate compositing and animating to Core Animation
- Setting the wantsLayer property of an NSView to YES opts in
  - The view now manages a layer
  - All descendants of a view manage their own layers, too
- This offers different performance and memory tradeoffs
- Things behave…differently

Core Animation

How AppKit runs an animation

Core Animation

How AppKit runs an animation

- The animation is stored in addition to the property in the view

Core Animation

How AppKit runs an animation

- The animation is stored in addition to the property in the view
- AppKit periodically wakes up on the main thread

Core Animation

How AppKit runs an animation

- The animation is stored in addition to the property in the view
- AppKit periodically wakes up on the main thread
- It evaluates the animation for the current time, and applies that value to the object being animated

Core Animation

How AppKit runs an animation

- The animation is stored in addition to the property in the view
- AppKit periodically wakes up on the main thread
- It evaluates the animation for the current time, and applies that value to the object being animated
- The value for that property in the view is replaced

Core Animation

How AppKit runs an animation

- The animation is stored in addition to the property in the view
- AppKit periodically wakes up on the main thread
- It evaluates the animation for the current time, and applies that value to the object being animated
- The value for that property in the view is replaced
- The regular NSView drawing cycle draws the values currently stored in the view

Core Animation

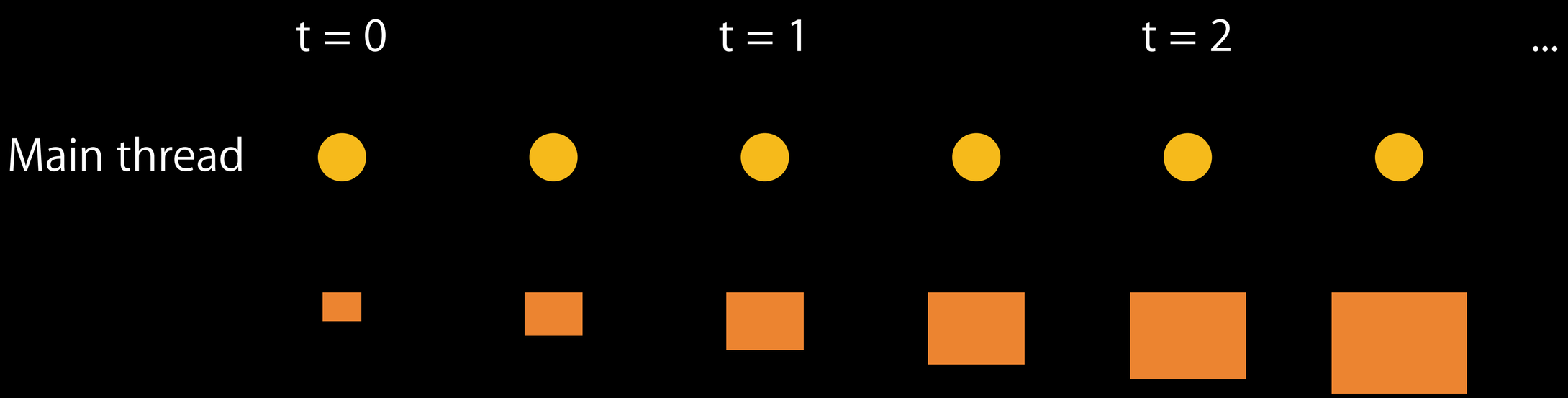
How AppKit runs an animation

t = 0 t = 1 t = 2 ...

Main thread

Core Animation

How AppKit runs an animation



Core Animation

How Core Animation runs an animation

Core Animation

How Core Animation runs an animation

- The animation is stored in addition to the property in the layer

Core Animation

How Core Animation runs an animation

- The animation is stored in addition to the property in the layer
- Core Animation periodically wakes up on a background thread

Core Animation

How Core Animation runs an animation

- The animation is stored in addition to the property in the layer
- Core Animation periodically wakes up on a background thread
- It evaluates the animation as part of rendering

Core Animation

How Core Animation runs an animation

- The animation is stored in addition to the property in the layer
- Core Animation periodically wakes up on a background thread
- It evaluates the animation as part of rendering
- The value for that property in the layer is unchanged

Core Animation

How Core Animation runs an animation

t = 0 t = 1 t = 2 ...

Main thread

Core Animation

How Core Animation runs an animation

t = 0 t = 1 t = 2 ...

Main thread

Core Animation

Core Animation

- NSView tries to let Core Animation drive things

Core Animation

- NSView tries to let Core Animation drive things
- frame, frameOrigin, and frameSize properties are important exceptions

Core Animation

- NSView tries to let Core Animation drive things
- frame, frameOrigin, and frameSize properties are important exceptions
- The layerContentsRedrawPolicy of NSView determines whether Core Animation is used

Core Animation

- NSView tries to let Core Animation drive things
- frame, frameOrigin, and frameSize properties are important exceptions
- The layerContentsRedrawPolicy of NSView determines whether Core Animation is used

Optimizing Drawing and Scrolling on OS X

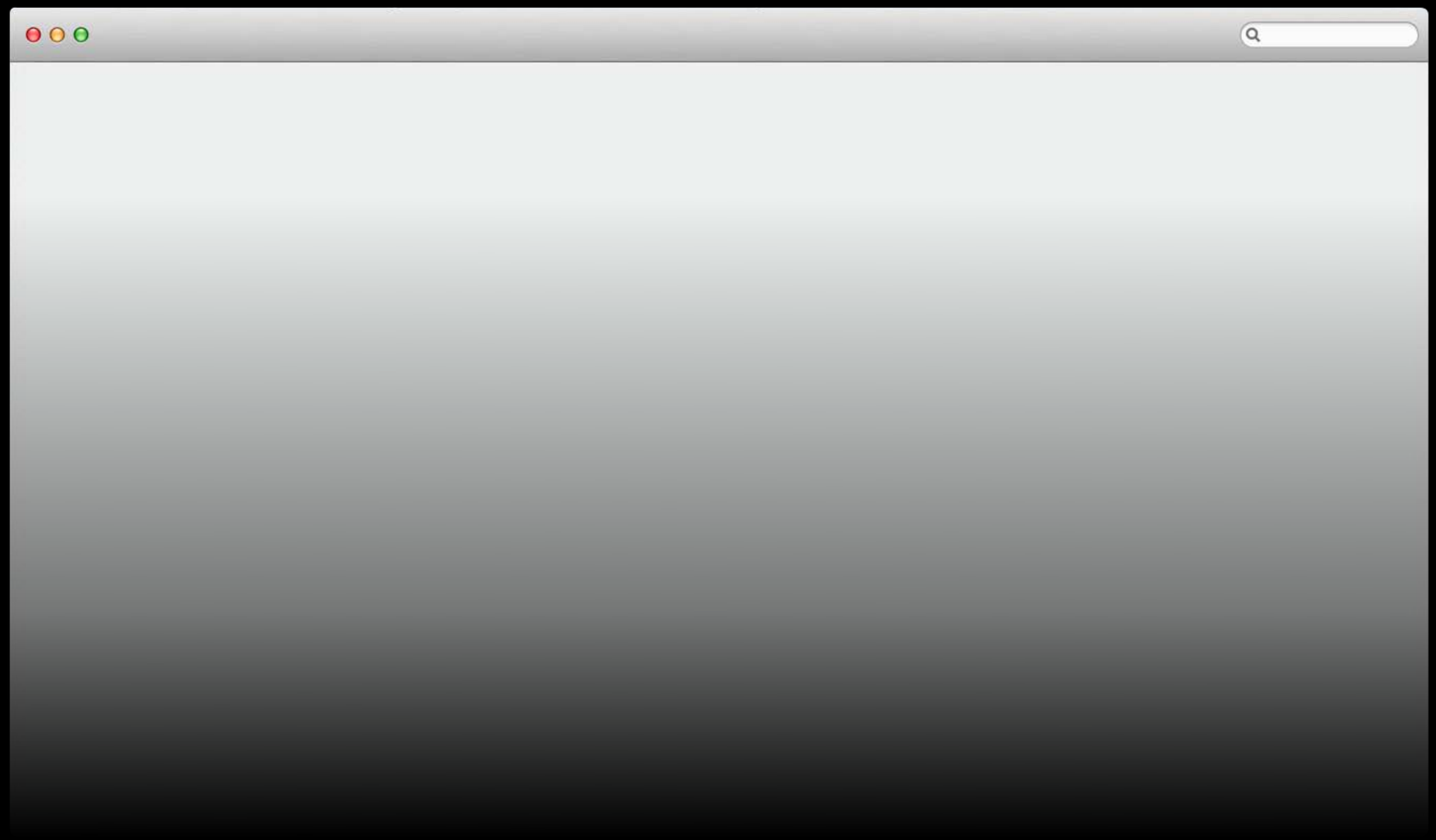
Marina
Wednesday 3:15AM

Core Animation

This code is simple

```
- (void)windowDidLoad {
    view.frame = fromValue;
    view.animator.frame = toValue;
}
```

Core Animation



Core Animation

Core Animation

This code is still simple, but broken

```
- (void)windowDidLoad {
    view.wantsLayer = YES;
    view.frame = fromValue;
    view.animator.frame = toValue;
}
```

Core Animation

This code is still simple, but broken

```
- (void)windowDidLoad {
    view.wantsLayer = YES;
    view.frame = fromValue;
    view.animator.frame = toValue;
}
```

Core Animation

This code is still simple, but broken

```
- (void)windowDidLoad {
    view.wantsLayer = YES;
    view.frame = fromValue;
    view.animator.frame = toValue;
}
```

Core Animation

Core Animation

Core Animation

- Changes grouped into transactions

Core Animation

- Changes grouped into transactions
- Implicit animation interpolates from the onscreen value to the new value

Core Animation

- Changes grouped into transactions
- Implicit animation interpolates from the onscreen value to the new value
- There was no onscreen value, so the view displays at the final position

Core Animation

- Changes grouped into transactions
- Implicit animation interpolates from the onscreen value to the new value
- There was no onscreen value, so the view displays at the final position
- There are two ways to fix this

Core Animation

Fix 1 - explicit “from” value

```
- (void)windowDidLoad {
    view.wantsLayer = YES;
    CABasicAnimation *animation = [CABasicAnimation animation];
    animation.fromValue = [NSValue valueWithRect:fromValue];
    animation.toValue = [NSValue valueWithRect:toValue]; // optional
    view.animations = @{ @”frame” : animation };
    view.animator.frame = toValue;
}
```

Core Animation

Fix 1 - explicit “from” value

```
- (void)windowDidLoad {
    view.wantsLayer = YES;
    CABasicAnimation *animation = [CABasicAnimation animation];
    animation.fromValue = [NSValue valueWithRect:fromValue];
    animation.toValue = [NSValue valueWithRect:toValue]; // optional
    view.animations = @{ @”frame” : animation };
    view.animator.frame = toValue;
}
```

Core Animation

Fix 1 - explicit “from” value

```
- (void)windowDidLoad {
    view.wantsLayer = YES;
    CABasicAnimation *animation = [CABasicAnimation animation];
    animation.fromValue = [NSValue valueWithRect:fromValue];
    animation.toValue = [NSValue valueWithRect:toValue]; // optional
    view.animations = @{ @”frame” : animation };
    view.animator.frame = toValue;
}
```

Core Animation

Fix 1 - explicit “from” value

```
- (void)windowDidLoad {
    view.wantsLayer = YES;
    CABasicAnimation *animation = [CABasicAnimation animation];
    animation.fromValue = [NSValue valueWithRect:fromValue];
    animation.toValue = [NSValue valueWithRect:toValue]; // optional
    view.animations = @{ @”frame” : animation };
    view.animator.frame = toValue;
}
```

Core Animation

Fix 1 - explicit “from” value

```
- (void)windowDidLoad {
    view.wantsLayer = YES;
    CABasicAnimation *animation = [CABasicAnimation animation];
    animation.fromValue = [NSValue valueWithRect:fromValue];
    animation.toValue = [NSValue valueWithRect:toValue]; // optional
    view.animations = @{ @”frame” : animation };
    view.animator.frame = toValue;
}
```

Core Animation

Fix 2 - completion handler

```
- (void)windowDidLoad {
    view.wantsLayer = YES;
    [NSAnimationContext runAnimationGroup:^(NSAnimationContext*) {
        view.frame = fromValue;
    } completionHandler:^{
        view.animator.frame = toValue;
    }];
}
```

Core Animation

Fix 2 - completion handler

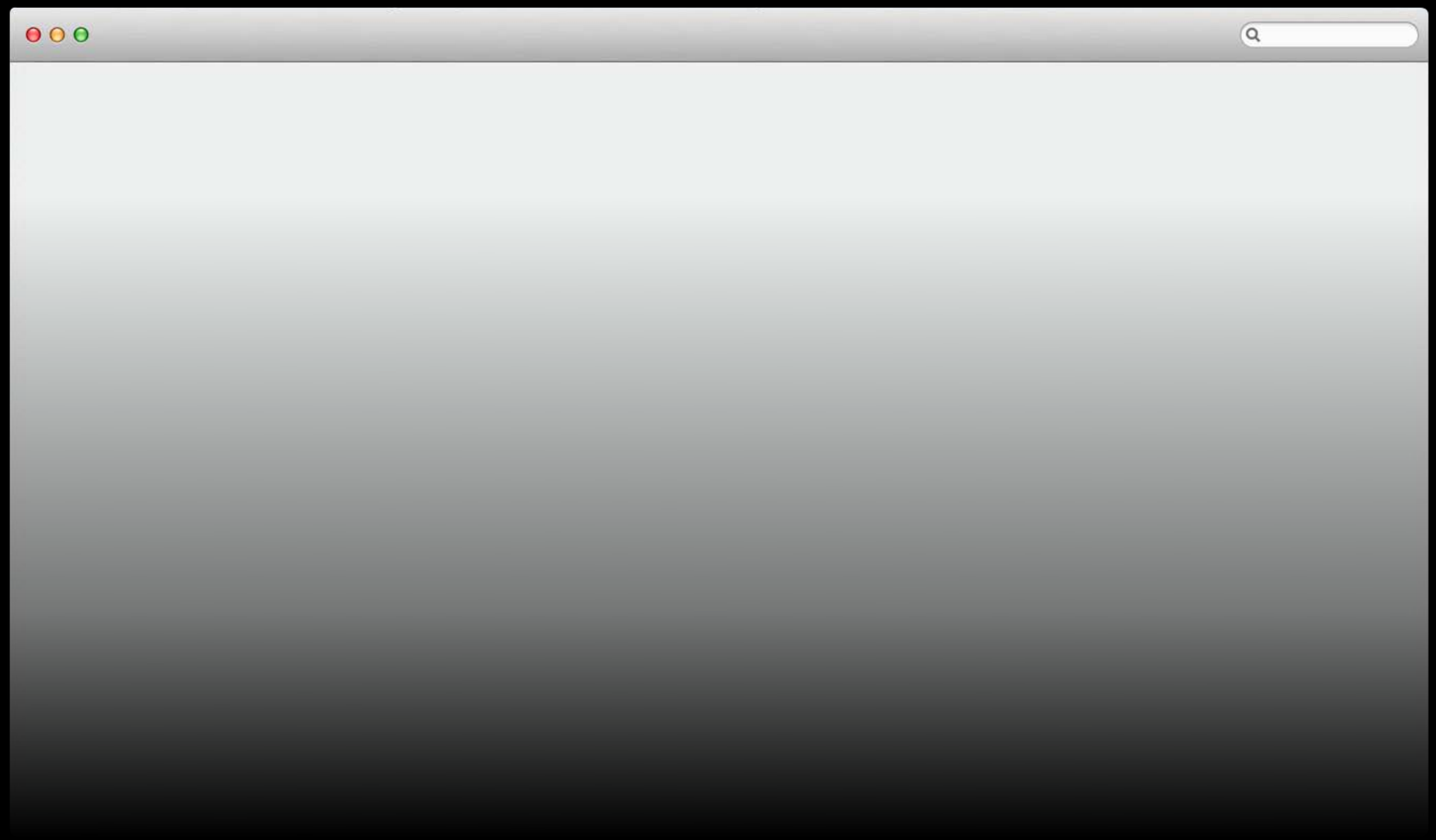
```
- (void)windowDidLoad {
    view.wantsLayer = YES;
    [NSAnimationContext runAnimationGroup:^(NSAnimationContext*) {
        view.frame = fromValue;
    } completionHandler:^{
        view.animator.frame = toValue;
    }];
}
```

Core Animation

Fix 2 - completion handler

```
- (void)windowDidLoad {
    view.wantsLayer = YES;
    [NSAnimationContext runAnimationGroup:^(NSAnimationContext*) {
        view.frame = fromValue;
    } completionHandler:^{
        view.animator.frame = toValue;
    }];
}
```

Core Animation



Core Animation

Basic Animation

Custom Property Animations

Overriding Default Animations

Chaining Animations

Implicit Animation

Core Animation

NSStackView

Animate View Positions

Animate Constraints Directly

Animate Window Size Changes

Basic Animation

Custom Property Animations

Overriding Default Animations

Chaining Animations

Implicit Animation

Core Animation

NSStackView

Animate View Positions

Animate Constraints Directly

Animate Window Size Changes

Animation with Auto Layout

Make it more fun with constraints

Basic Animation

Custom Property Animations

Overriding Default Animations

Chaining Animations

Implicit Animation

Core Animation

NSStackView

Animate View Positions

Animate Constraints Directly

Animate Window Size Changes

Basic Animation

Custom Property Animations

Overriding Default Animations

Chaining Animations

Implicit Animation

Core Animation

NSStackView

Animate View Positions

Animate Constraints Directly

Animate Window Size Changes

NSStackView

NSStackView

Name:

Submit

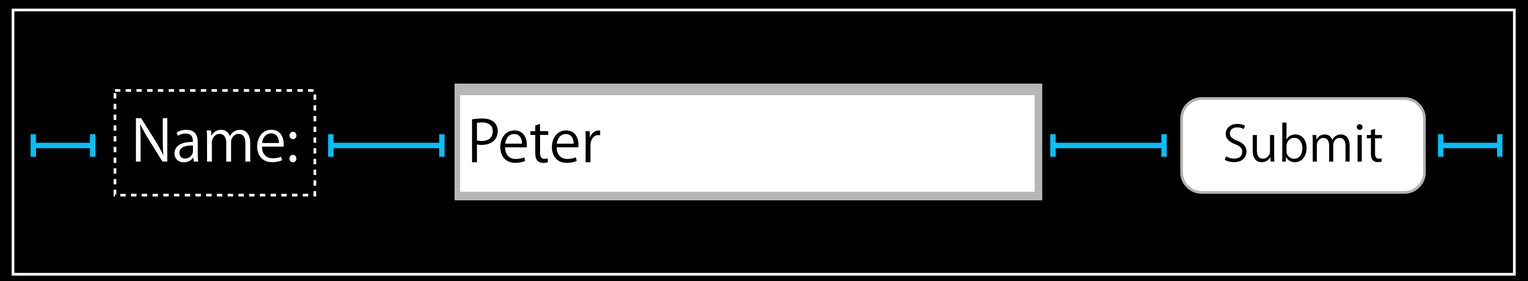
Peter

NSStackView

NSStackView

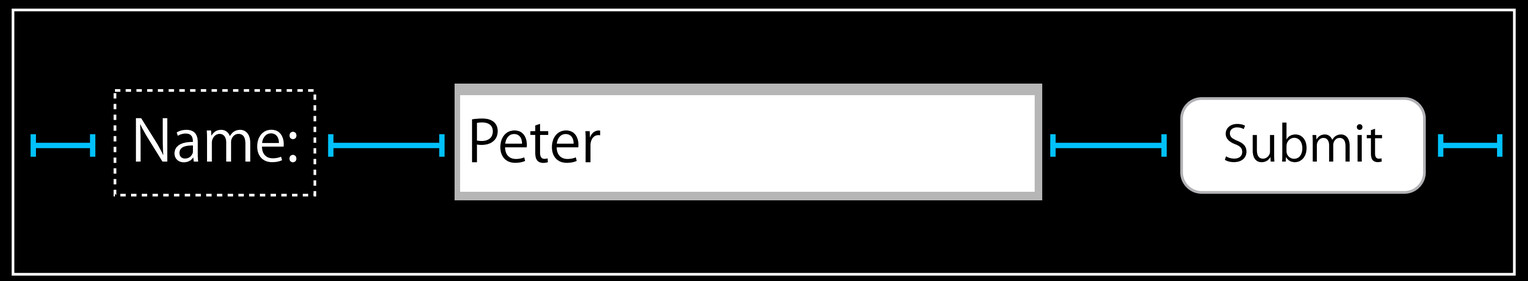
- Uses Auto Layout

NSStackView



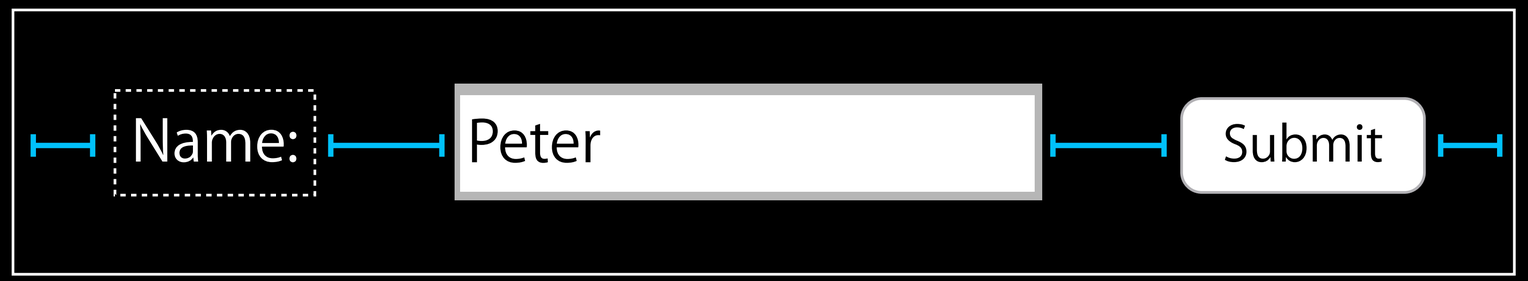
- Uses Auto Layout

NSStackView



- Uses Auto Layout
- Knows how to size things

NSStackView



- Uses Auto Layout
- Knows how to size things
- Knows how to align things
  - Baseline, center, top, bottom…

NSStackView

- Uses Auto Layout
- Knows how to size things
- Knows how to align things
  - Baseline, center, top, bottom…

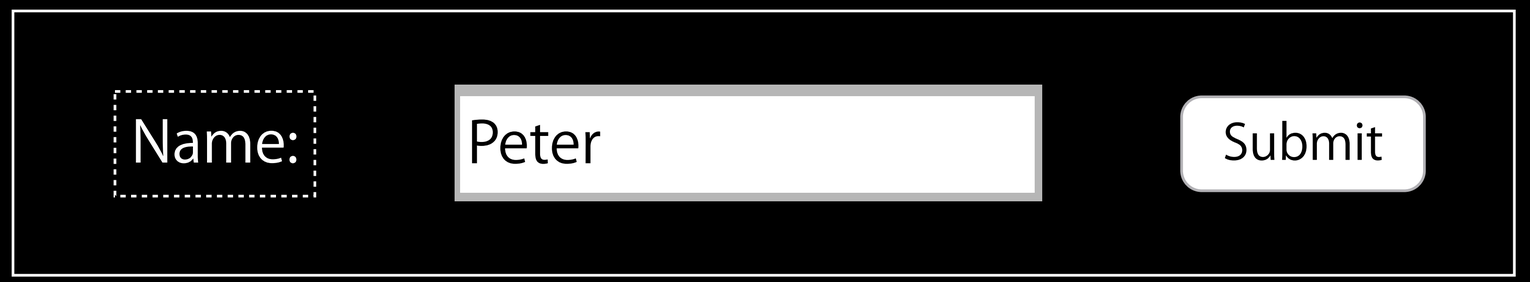
NSStackView

- Uses Auto Layout
- Knows how to size things
- Knows how to align things
  - Baseline, center, top, bottom…
- Interacts well with window resizing

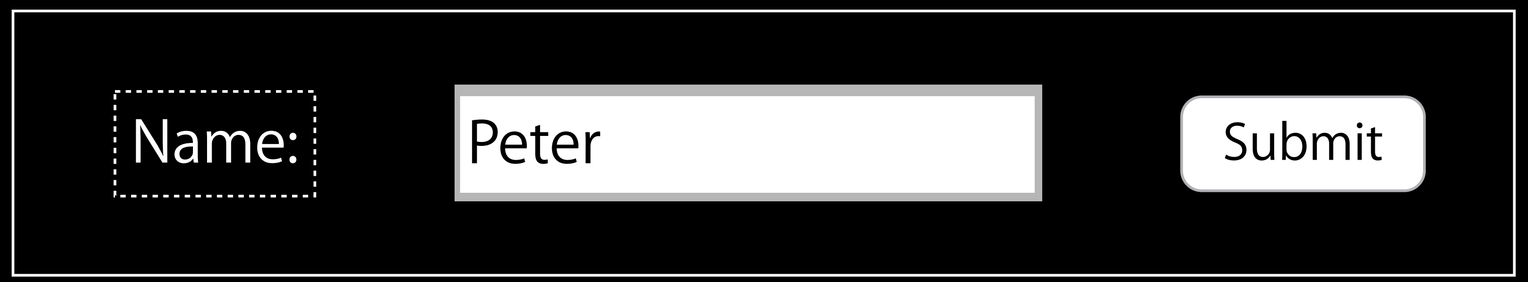
NSStackView

- Uses Auto Layout
- Knows how to size things
- Knows how to align things
  - Baseline, center, top, bottom…
- Interacts well with window resizing
- Makes it easy to create common layouts

NSStackView

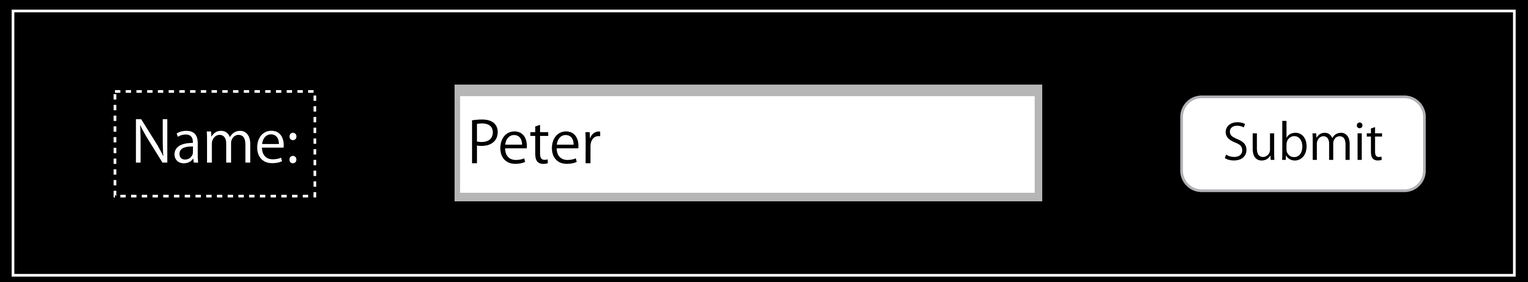


NSStackView



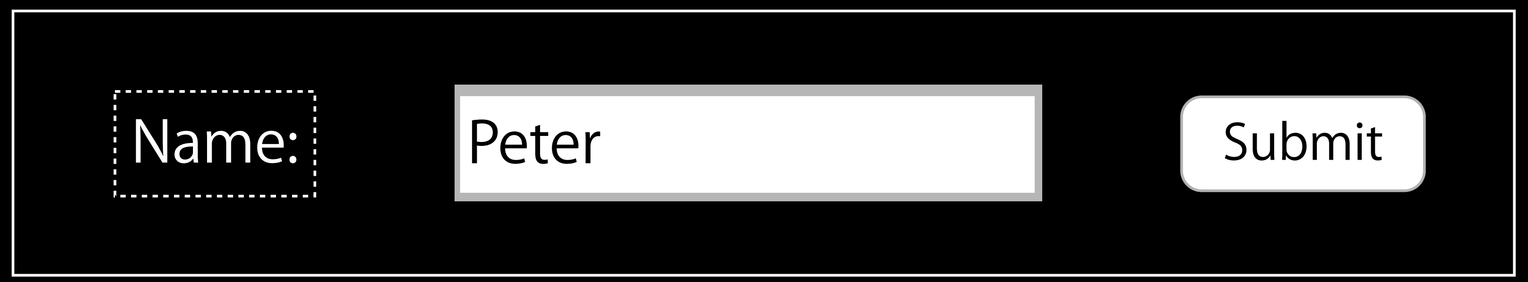
- Horizontal or vertical

NSStackView



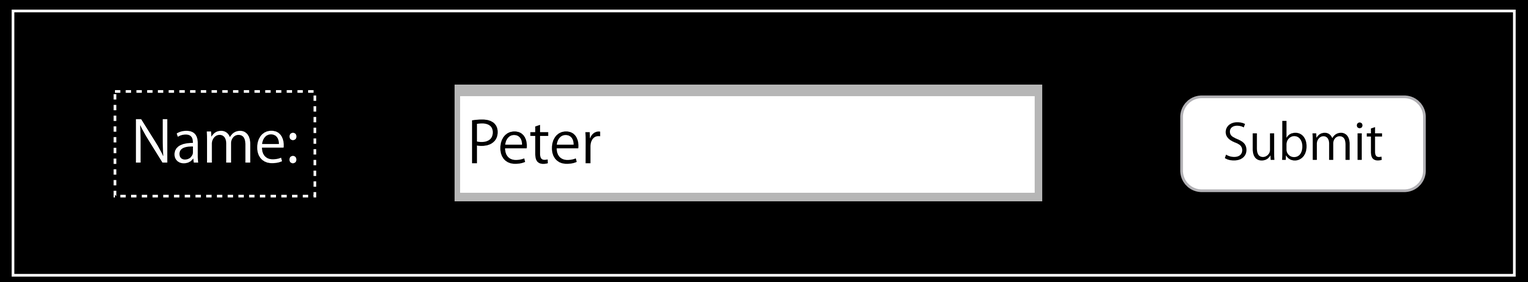
- Horizontal or vertical
- Flexible or equal spacing

NSStackView



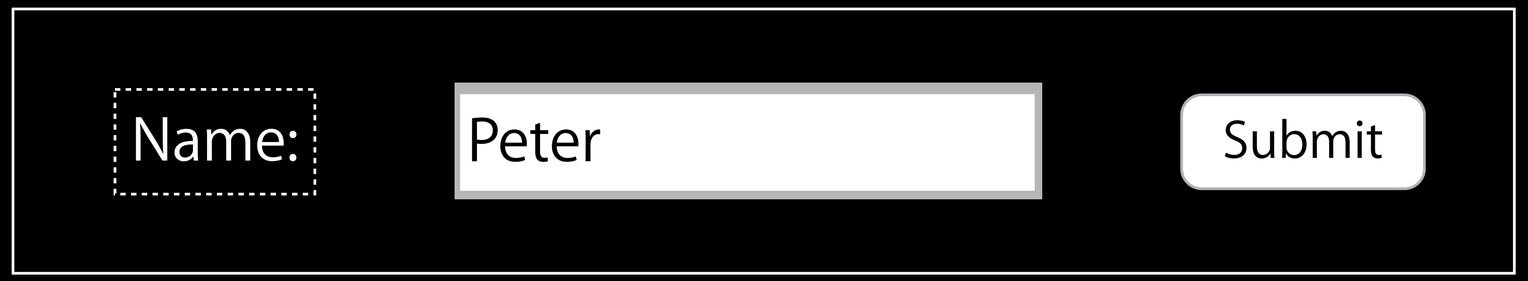
- Horizontal or vertical
- Flexible or equal spacing
- Nestable

NSStackView



- Horizontal or vertical
- Flexible or equal spacing
- Nestable
- Automatic view detaching on overflow

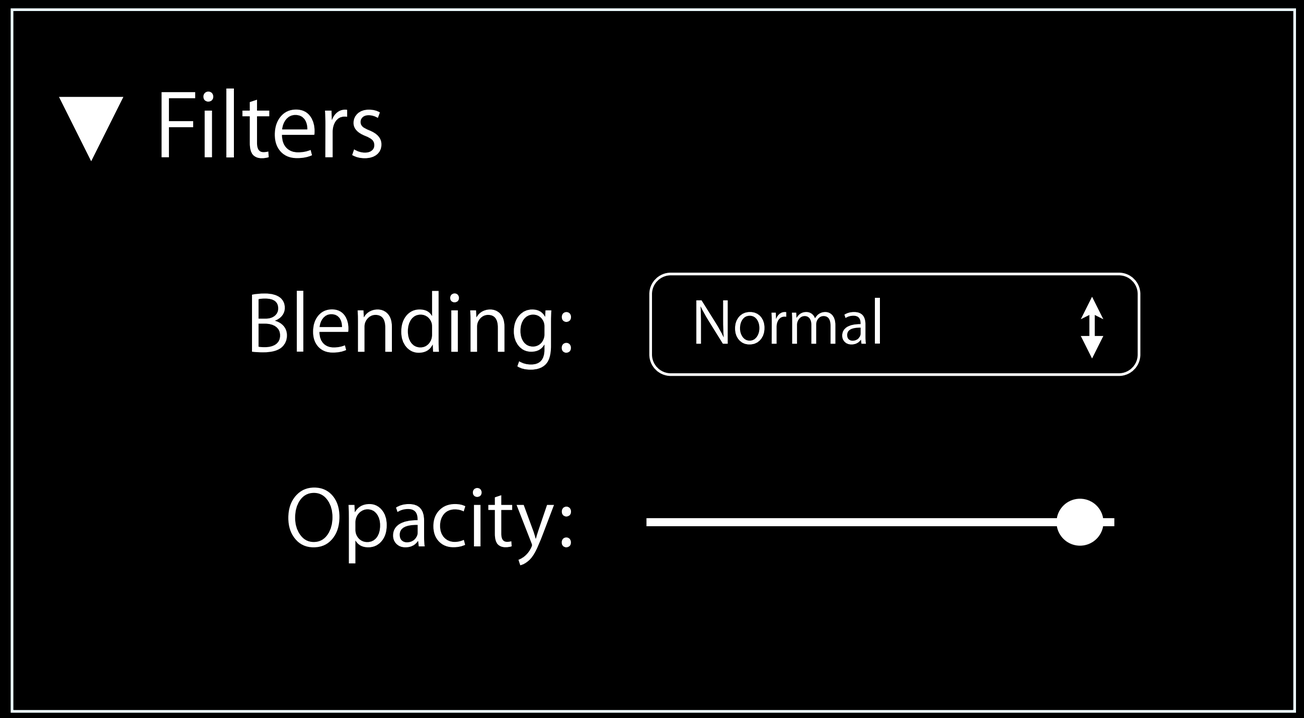
NSStackView



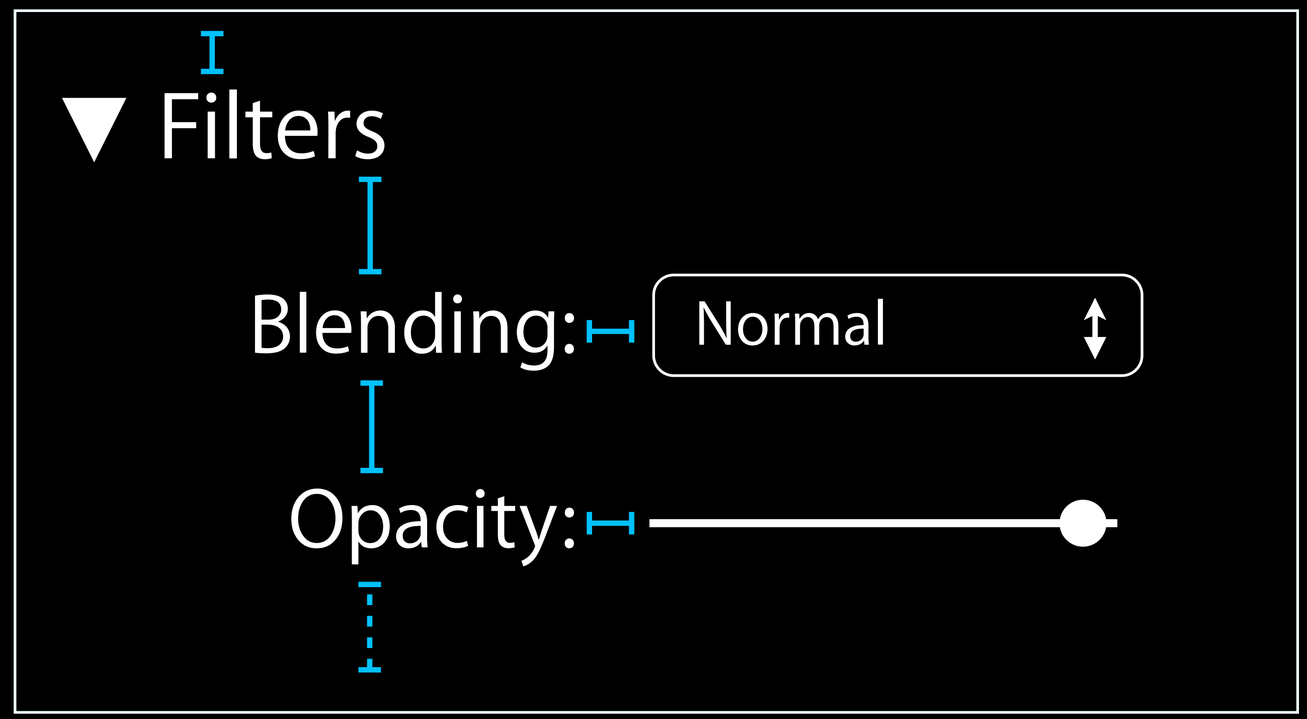

```
[NSStackView stackViewWithViews: @[label, field, button] ]
```

Inspector Window

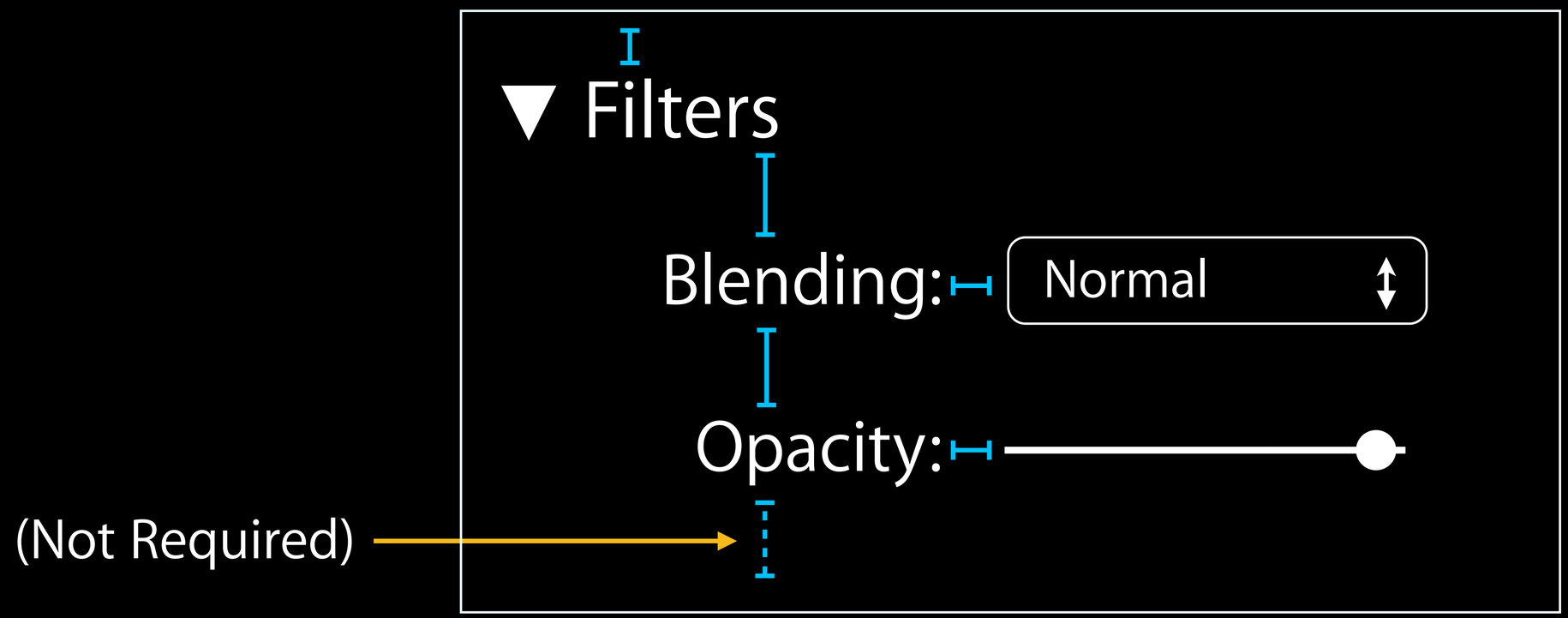
Inspector Window



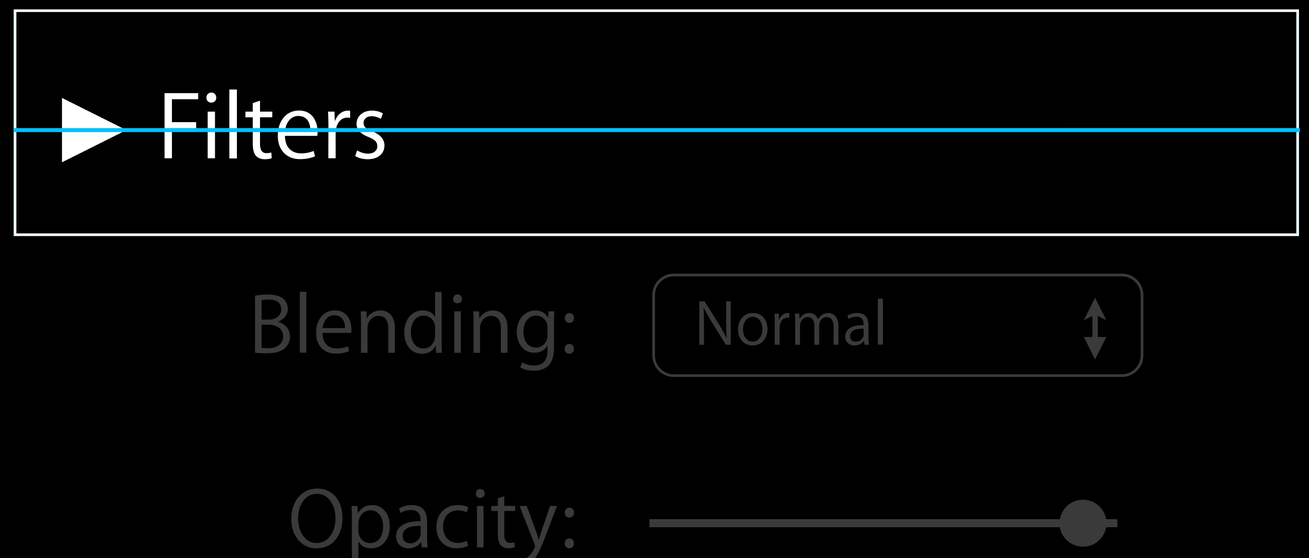
Inspector Window



Inspector Window



Inspector Window



Inspector Window

Demo

Basic Animation

Custom Property Animations

Overriding Default Animations

Chaining Animations

Implicit Animation

Core Animation

NSStackView

Animate View Positions

Animate Constraints Directly

Animate Window Size Changes

Basic Animation

Custom Property Animations

Overriding Default Animations

Chaining Animations

Implicit Animation

Core Animation

NSStackView

Animate View Positions

Animate Constraints Directly

Animate Window Size Changes

Animate View Positions via Constraints

- Change constraints directly from old layout to new layout
- Views should animate to new positions

Animate View Positions via Constraints

- Animation by modifying frames directly:

```
[NSAnimationContext runAnimationBlock:^(NSAnimationContext *context){

        view.animator.frame = NSMakeRect(...);

    context.allowsImplicitAnimation = YES;
    view.frame = NSMakeRect(...);

}];
```

Animate View Positions via Constraints

- Animation by modifying frames directly:

```
[NSAnimationContext runAnimationBlock:^(NSAnimationContext *context){

        view.animator.frame = NSMakeRect(...);
```

```
    context.allowsImplicitAnimation = YES;
    view.frame = NSMakeRect(...);

}];
```

Animate View Positions via Constraints

- Animation by modifying frames directly:

```
[NSAnimationContext runAnimationBlock:^(NSAnimationContext *context){

        view.animator.frame = NSMakeRect(...);
```

```
    context.allowsImplicitAnimation = YES;
    view.frame = NSMakeRect(...);
```

```
}];
```

Animate View Positions via Constraints

- Animation by modifying frames directly:

```
[NSAnimationContext runAnimationBlock:^(NSAnimationContext *context){

        view.animator.frame = NSMakeRect(...);

    context.allowsImplicitAnimation = YES;
    view.frame = NSMakeRect(...);

}];
```

- How to do this with constraints?

Animate View Positions via Constraints

- How to NOT animate with constraints:

```
[NSAnimationContext runAnimationBlock:^(NSAnimationContext *context){

}];
```

Animate View Positions via Constraints

- How to NOT animate with constraints:

```
[NSAnimationContext runAnimationBlock:^(NSAnimationContext *context){

        [view addConstraint:constraint];

}];
```

Animate View Positions via Constraints

- How to NOT animate with constraints:

```
[NSAnimationContext runAnimationBlock:^(NSAnimationContext *context){

    [view addConstraint:constraint];

}];
```

Animate View Positions via Constraints

- How to NOT animate with constraints:

```
[NSAnimationContext runAnimationBlock:^(NSAnimationContext *context){

    [view addConstraint:constraint];

        constraint.constant = 17;



}];
```

Animate View Positions via Constraints

- How to NOT animate with constraints:

```
[NSAnimationContext runAnimationBlock:^(NSAnimationContext *context){

    [view addConstraint:constraint];

    constraint.constant = 17;


}];
```

Animate View Positions via Constraints

- How to NOT animate with constraints:

```
[NSAnimationContext runAnimationBlock:^(NSAnimationContext *context){

    [view addConstraint:constraint];

    constraint.constant = 17;

    [view layout];

}];
```

Animate View Positions via Constraints

- How to NOT animate with constraints:

```
[NSAnimationContext runAnimationBlock:^(NSAnimationContext *context){

    [view addConstraint:constraint];

    constraint.constant = 17;

    [view layout];

}];
```

Animate View Positions via Constraints

- The underlying setFrame: call must be in an animation block
  - …with allowsImplicitAnimation = YES

Animate View Positions via Constraints

- How to animate view positions with constraints:

```
[view addConstraint:constraint];
constraint.constant = 17;
[NSAnimationContext runAnimationBlock:^(NSAnimationContext *context){
    context.allowsImplicitAnimation = YES;

    [view layoutSubtreeIfNeeded];
    /* OR */
    [window layoutIfNeeded];
}];
```

Animate View Positions via Constraints

- How to animate view positions with constraints:

```
[view addConstraint:constraint];
constraint.constant = 17;
[NSAnimationContext runAnimationBlock:^(NSAnimationContext *context){
    context.allowsImplicitAnimation = YES;

    [view layoutSubtreeIfNeeded];
    /* OR */
    [window layoutIfNeeded];
}];
```

Animate View Positions via Constraints

- How to animate view positions with constraints:

```
[view addConstraint:constraint];
constraint.constant = 17;
[NSAnimationContext runAnimationBlock:^(NSAnimationContext *context){
    context.allowsImplicitAnimation = YES;

    [view layoutSubtreeIfNeeded];
    /* OR */
    [window layoutIfNeeded];
}];
```

Animate View Positions via Constraints

- How to animate view positions with constraints:

```
[view addConstraint:constraint];
constraint.constant = 17;
[NSAnimationContext runAnimationBlock:^(NSAnimationContext *context){
    context.allowsImplicitAnimation = YES;

    [view layoutSubtreeIfNeeded];
    /* OR */
    [window layoutIfNeeded];
}];
```

Animate View Positions via Constraints

- How to animate view positions with constraints:

```
[view addConstraint:constraint];
constraint.constant = 17;
[NSAnimationContext runAnimationBlock:^(NSAnimationContext *context){
    context.allowsImplicitAnimation = YES;

    [view layoutSubtreeIfNeeded];
    /* OR */
    [window layoutIfNeeded];
}];
```

Animate View Positions via Constraints

- How to animate view positions with constraints:

```
[view addConstraint:constraint];
constraint.constant = 17;
[NSAnimationContext runAnimationBlock:^(NSAnimationContext *context){
    context.allowsImplicitAnimation = YES;

    [view layoutSubtreeIfNeeded];
    /* OR */
    [window layoutIfNeeded];
}];
```

Animate View Positions via Constraints

- How to animate view positions with constraints:

```
[view addConstraint:constraint];
constraint.constant = 17;
[NSAnimationContext runAnimationBlock:^(NSAnimationContext *context){
    context.allowsImplicitAnimation = YES;

    [view layoutSubtreeIfNeeded];
    /* OR */
    [window layoutIfNeeded];
}];
```

Demo

Basic Animation

Custom Property Animations

Overriding Default Animations

Chaining Animations

Implicit Animation

Core Animation

NSStackView

Animate View Positions

Animate Constraints Directly

Animate Window Size Changes

Basic Animation

Custom Property Animations

Overriding Default Animations

Chaining Animations

Implicit Animation

Core Animation

NSStackView

Animate View Positions

Animate Constraints Directly

Animate Window Size Changes

Animate Constraints Directly

Animate Constraints Directly

- Constraints have only one mutable property

```
@property CGFloat constant
```

Animate Constraints Directly

- Constraints have only one mutable property

```
@property CGFloat constant
```

- You can animate it via the animator proxy

```
constraint.animator.constant = 17
```

Animate Constraints Directly

- Constraints have only one mutable property

```
@property CGFloat constant
```

- You can animate it via the animator proxy

```
constraint.animator.constant = 17
```

- Constraints do NOT respect allowsImplicitAnimation

Demo

Animate Window Size Changes

Animate Window Size Changes

- Core Animation is asynchronous (background thread)

Animate Window Size Changes

- Core Animation is asynchronous (background thread)
- Window resize is synchronous (main thread)

Animate Window Size Changes

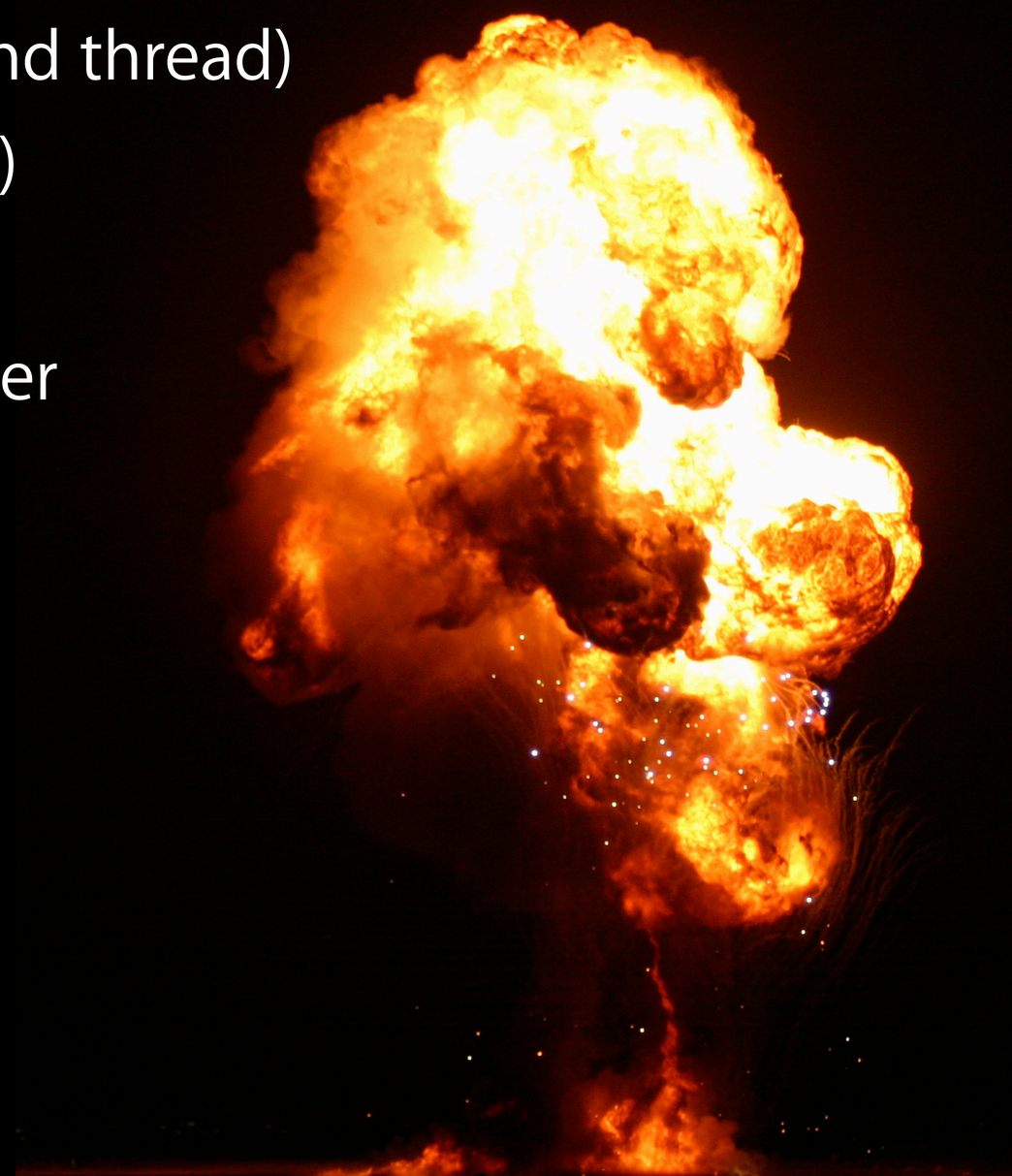
- Core Animation is asynchronous (background thread)
- Window resize is synchronous (main thread)
- Don’t cross the streams

Animate Window Size Changes

- Core Animation is asynchronous (background thread)
- Window resize is synchronous (main thread)
- Don’t cross the streams

Animate Window Size Changes

- Core Animation is asynchronous (background thread)
- Window resize is synchronous (main thread)
- Don’t cross the streams
- When you have both, you get drifting or jitter



Animate Window Size Changes

- Core Animation is asynchronous (background thread)
- Window resize is synchronous (main thread)
- Don’t cross the streams
- When you have both, you get drifting or jitter
- Solutions:

  constraint.animator

  [NSWindow setFrame:display:animate:YES]

Basic Animation

Custom Property Animations

Overriding Default Animations

Chaining Animations

Implicit Animation

Core Animation

NSStackView

Animate View Positions

Animate Constraints Directly

Animate Window Size Changes

Basic Animation

Custom Property Animations

Overriding Default Animations

Chaining Animations

Implicit Animation

Core Animation

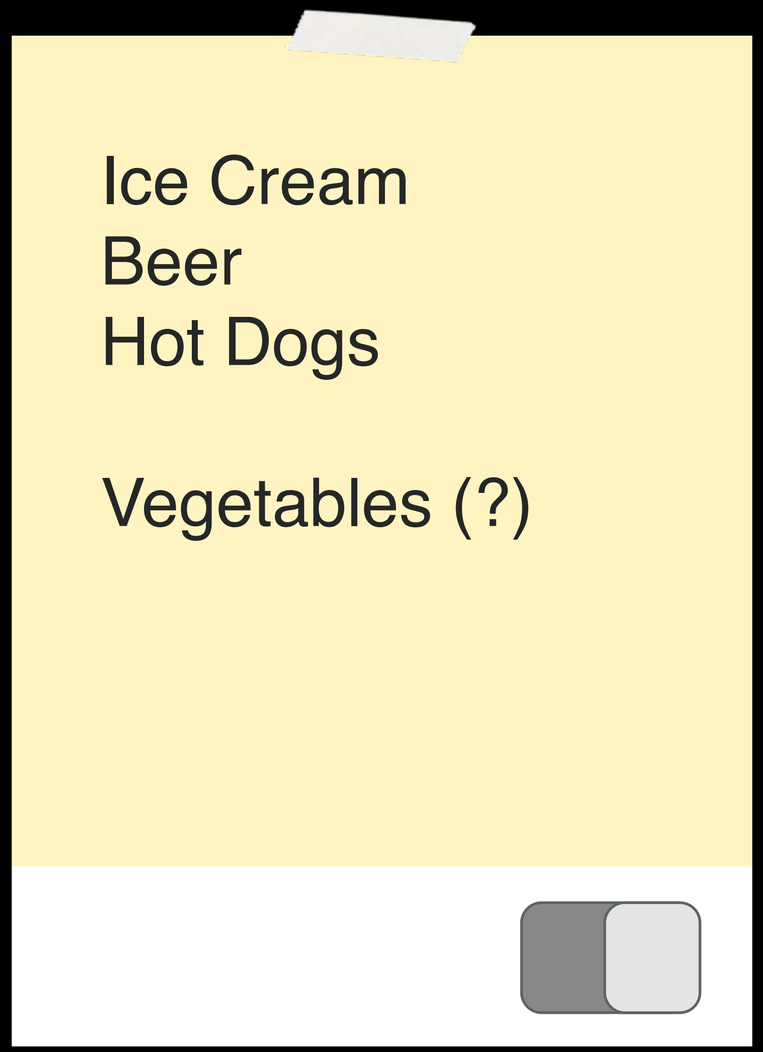
NSStackView

Animate View Positions

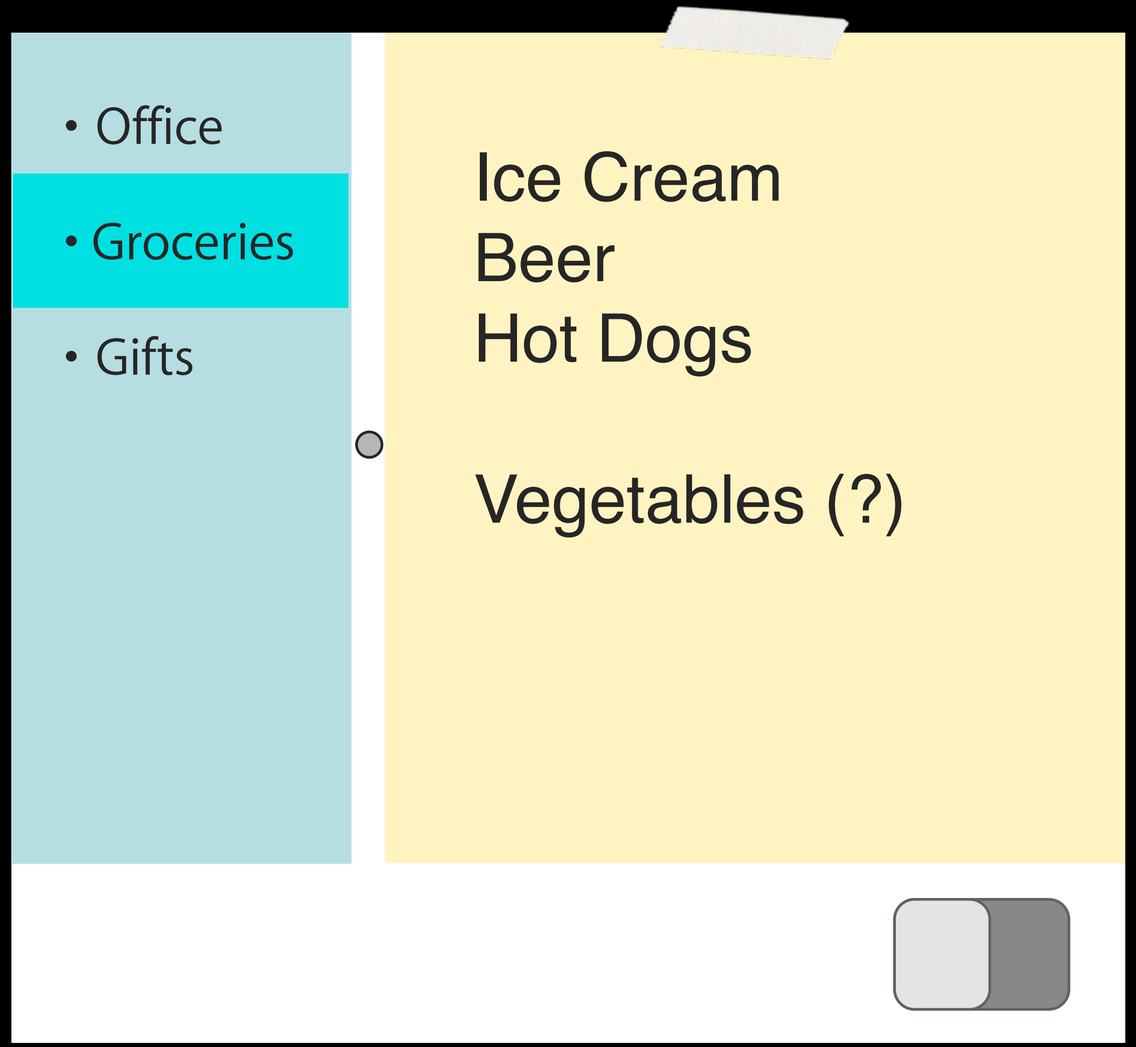
Animate Constraints Directly

Animate Window Size Changes

Master-Detail Window



Master-Detail Window



Animate Window Size Changes

- The window should grow
- Content should reflow in a particular way
- Different from window resizing

Master-Detail Window

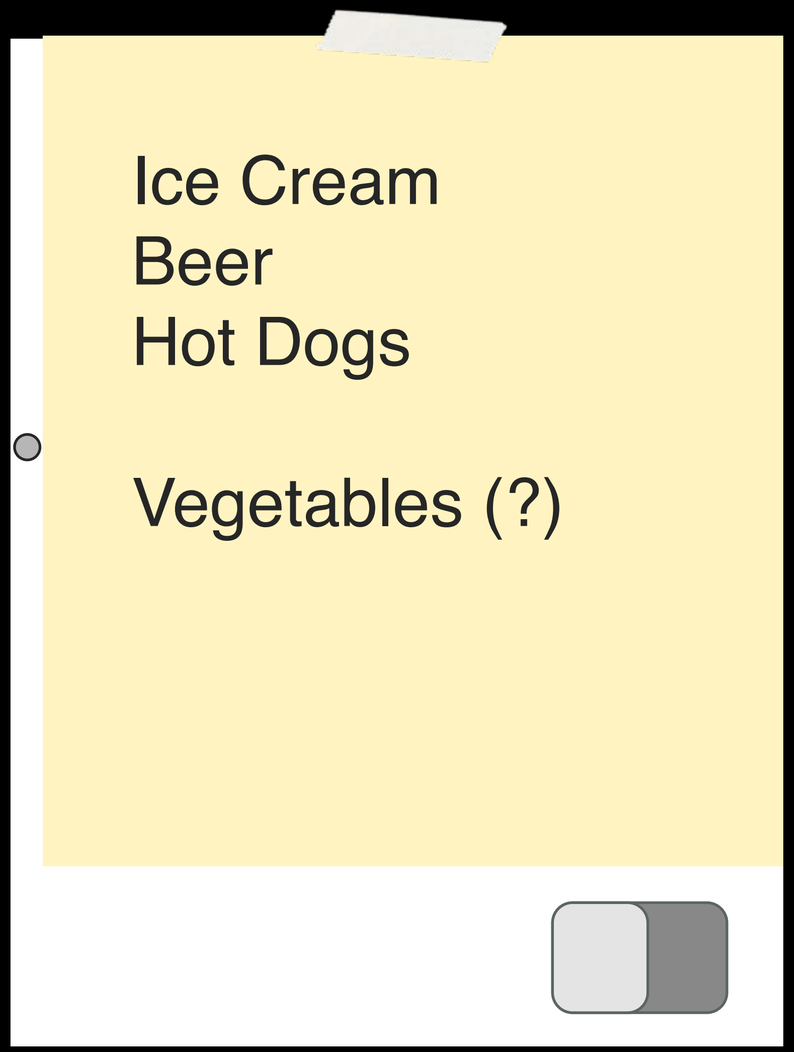
Normal window resizing

Master-Detail Window

Normal window resizing

Master-Detail Window

Our animation



Master-Detail Window

Our animation

Master-Detail Window

Master-Detail Window

- Solution:
  - Set temporary constraints to control how the panes resize
  - Grow the window
  - Remove those constraints

Master-Detail Window

- Solution:
  - Set temporary constraints to control how the panes resize
  - Grow the window
  - Remove those constraints
- NSSplitView will create these constraints for us

Master-Detail Window

- Solution:
  - Set temporary constraints to control how the panes resize
  - Grow the window
  - Remove those constraints
- NSSplitView will create these constraints for us
- Adjust holding priorities

Master-Detail Window

- Solution:
  - Set temporary constraints to control how the panes resize
  - Grow the window
  - Remove those constraints

```
NSRect windowFrame = [window frame];
windowFrame.origin.x -= 120; windowFrame.size.width += 120;
[splitView setHoldingPriority:1 forSubviewAtIndex:0];
[window setFrame:windowFrame display:YES animate:YES];
[splitView setHoldingPriority:NSLayoutPriorityDefaultLow forSubviewAtIndex:0];
```

Master-Detail Window

- Solution:
  - Set temporary constraints to control how the panes resize
  - Grow the window
  - Remove those constraints

```
NSRect windowFrame = [window frame];
windowFrame.origin.x -= 120; windowFrame.size.width += 120;
[splitView setHoldingPriority:1 forSubviewAtIndex:0];
[window setFrame:windowFrame display:YES animate:YES];
[splitView setHoldingPriority:NSLayoutPriorityDefaultLow forSubviewAtIndex:0];
```

Master-Detail Window

- Solution:
  - Set temporary constraints to control how the panes resize
  - Grow the window
  - Remove those constraints

```
NSRect windowFrame = [window frame];
windowFrame.origin.x -= 120; windowFrame.size.width += 120;
[splitView setHoldingPriority:1 forSubviewAtIndex:0];
[window setFrame:windowFrame display:YES animate:YES];
[splitView setHoldingPriority:NSLayoutPriorityDefaultLow forSubviewAtIndex:0];
```

Master-Detail Window

- Solution:
  - Set temporary constraints to control how the panes resize
  - Grow the window
  - Remove those constraints

```
NSRect windowFrame = [window frame];
windowFrame.origin.x -= 120; windowFrame.size.width += 120;
[splitView setHoldingPriority:1 forSubviewAtIndex:0];
[window setFrame:windowFrame display:YES animate:YES];
[splitView setHoldingPriority:NSLayoutPriorityDefaultLow forSubviewAtIndex:0];
```

Master-Detail Window

- Solution:
  - Set temporary constraints to control how the panes resize
  - Grow the window
  - Remove those constraints

```
NSRect windowFrame = [window frame];
windowFrame.origin.x -= 120; windowFrame.size.width += 120;
[splitView setHoldingPriority:1 forSubviewAtIndex:0];
[window setFrame:windowFrame display:YES animate:YES];
[splitView setHoldingPriority:NSLayoutPriorityDefaultLow forSubviewAtIndex:0];
```

Master-Detail Window

- Solution:
  - Set temporary constraints to control how the panes resize
  - Grow the window
  - Remove those constraints

```
NSRect windowFrame = [window frame];
windowFrame.origin.x -= 120; windowFrame.size.width += 120;
[splitView setHoldingPriority:1 forSubviewAtIndex:0];
[window setFrame:windowFrame display:YES animate:YES];
[splitView setHoldingPriority:NSLayoutPriorityDefaultLow forSubviewAtIndex:0];
```

One Remaining Wrinkle

One Remaining Wrinkle

- We want the view **and the divider** to disappear entirely

One Remaining Wrinkle

- We want the view **and the divider** to disappear entirely
- NSSplitView calls this “collapsing” the pane

One Remaining Wrinkle

- We want the view **and the divider** to disappear entirely
- NSSplitView calls this “collapsing” the pane
- Collapsing is a deliberate user action

One Remaining Wrinkle

- We want the view **and the divider** to disappear entirely
- NSSplitView calls this “collapsing” the pane
- Collapsing is a deliberate user action
- Constraints cannot collapse or uncollapse NSSplitView panes

One Remaining Wrinkle

One Remaining Wrinkle

Ice Cream
Beer
Hot Dogs

Vegetables (?)

One Remaining Wrinkle

One Remaining Wrinkle

- Solution:
  - Shrink the pane as small as we can get it with auto layout
  - Collapse it with `[splitView setPosition:0 ofDividerAtIndex:0]`
  - Uncollapse it with `[splitView setPosition:1 ofDividerAtIndex:0]`
  - Grow the pane with auto layout

One Remaining Wrinkle

- Solution:
  - Shrink the pane as small as we can get it with auto layout
  - Collapse it with `[splitView setPosition:0 ofDividerAtIndex:0]`
  - Uncollapse it with `[splitView setPosition:1 ofDividerAtIndex:0]`
  - Grow the pane with auto layout
- Don’t forget to enable collapsing!

```
- (BOOL)splitView:(NSSplitView *)view canCollapseSubview:(NSView *)subview
{
    return subview == splitView.subviews[0];
}
```

Demo

More Information

Jake Behrens
App Frameworks Evangelist
behrens@apple.com

Documentation
Core Animation Programming Guide
Auto Layout Programming Guide

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

Related Sessions

Interface Builder Core Concepts	Nob Hill Wednesday 9:00AM	
Optimizing Drawing and Scrolling on OS X	Marina Wednesday 3:15PM	

Labs

Auto Layout Lab	Tools Lab A Wednesday 2:00 PM
Full Screen and Cocoa Lab	Frameworks Lab A Thursday 9:00 AM
Interface Builder Lab	Tools Lab B Thursday 9:00AM
NSTableView, NSView, and Cocoa Lab	Frameworks Lab A Thursday 10:15AM
Cocoa Animations, Drawing, and Cocoa Lab	Frameworks Lab A Friday 9:00AM

Summary

- NSStackView lays out views in a list
  - Power through auto layout
- Animate view positions by adjusting constraints and triggering layout
  - Layers can produce very smooth animations
- Animate constraints directly
  - Window resizing is possible
- Animate window size changes directly
- Layers + implicit animations + window resize = jitter / drifting
  - Don’t cross the streams

WWDC2013