Handling Rotation
Custom Views
Custom Drawing
Animation

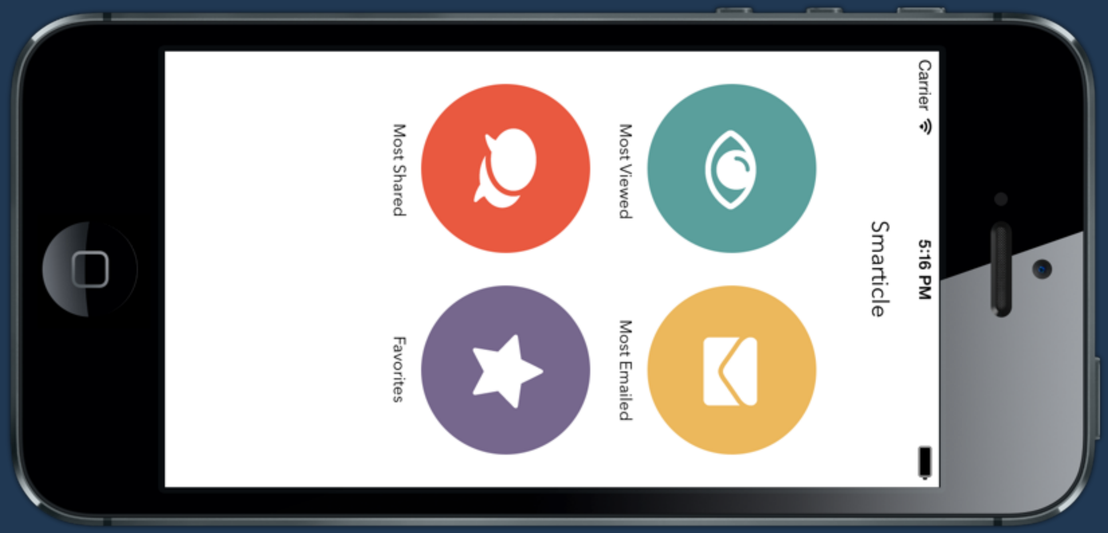
Rotation

Rotation

- Device vs Interface orientation
- Declaring support for rotation
- Autorotation

Device Orientation

- UIDeviceOrientationDidChangeNotification
- [UIDevice currentDevice].orientation

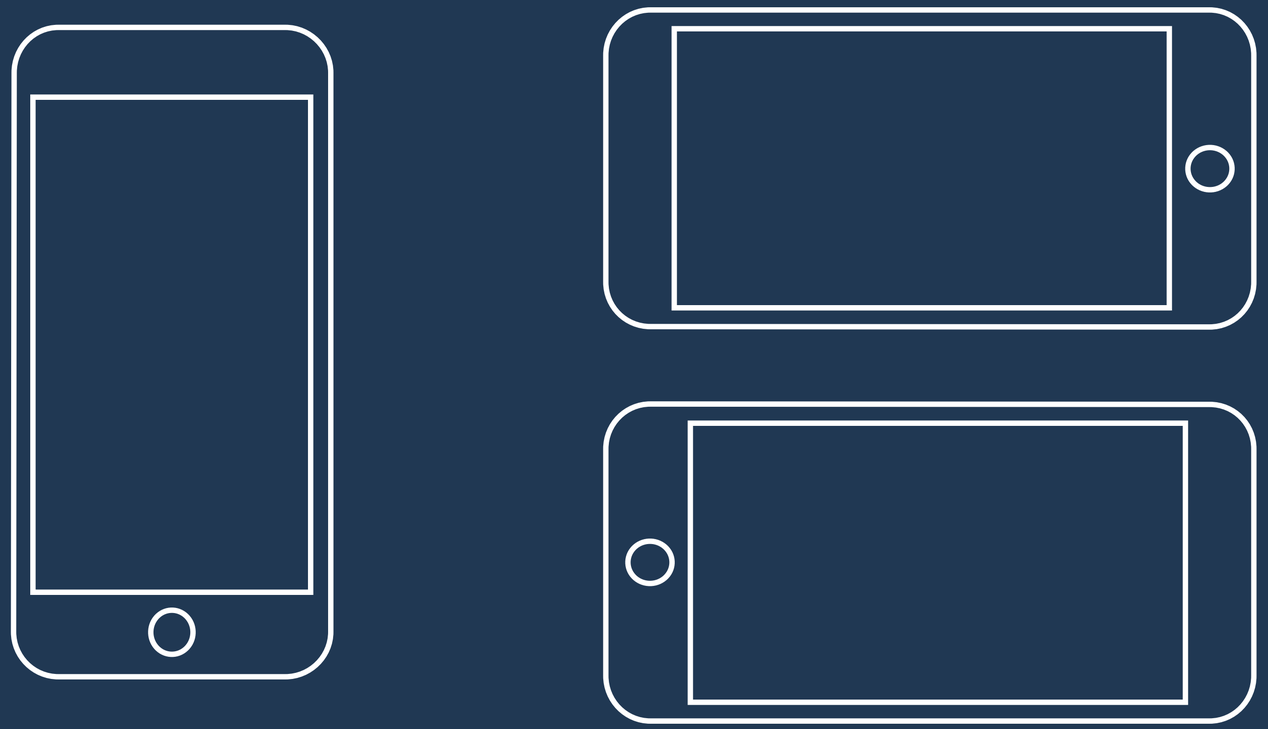


Generally, don’t rely on this to update your interface!

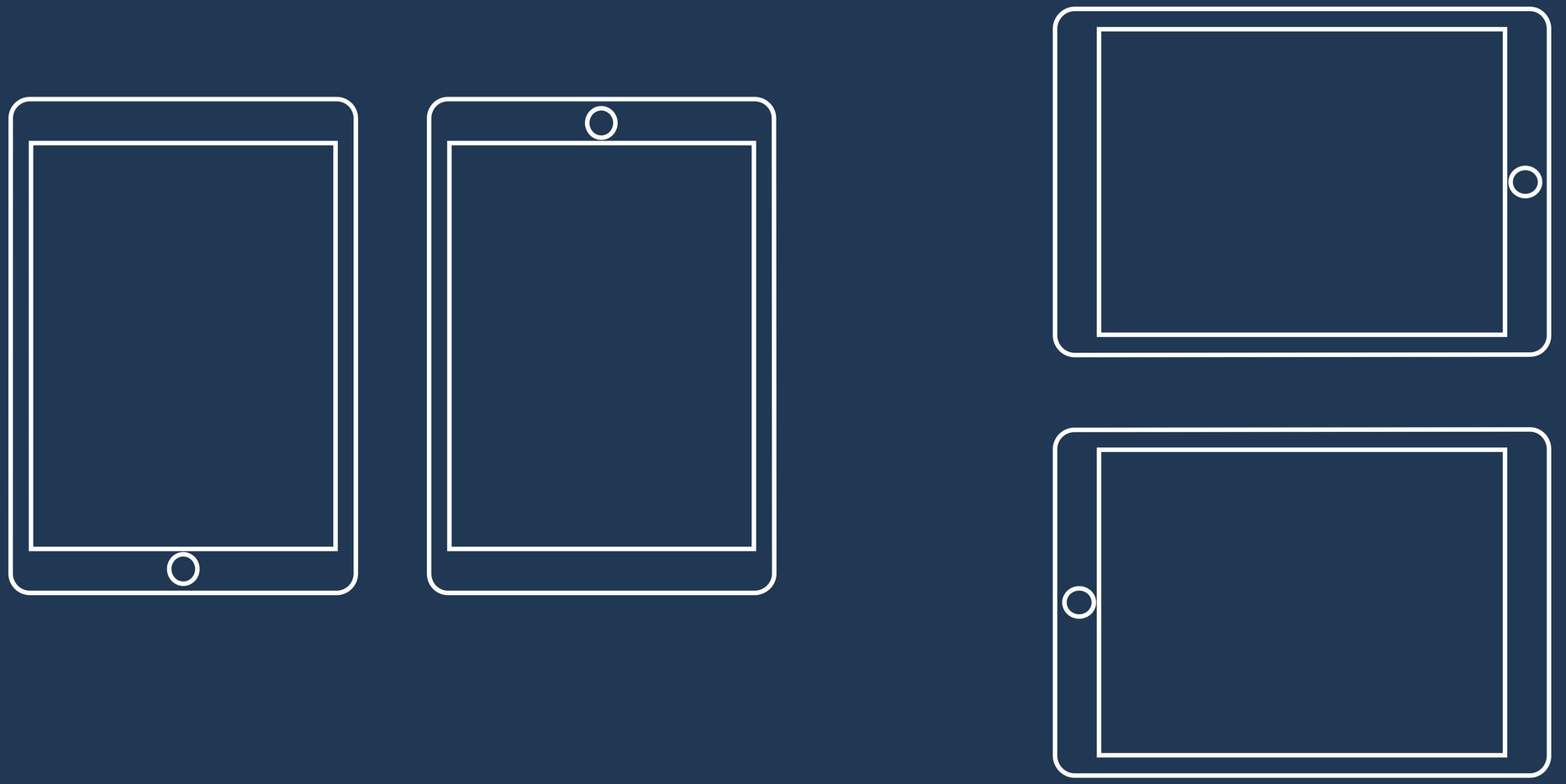
Interface Orientation

- UIViewController overrides (willRotate…, willAnimate…, didRotate…)
- [UIApplication sharedApplication].statusBarOrientation

iPhone



iPad

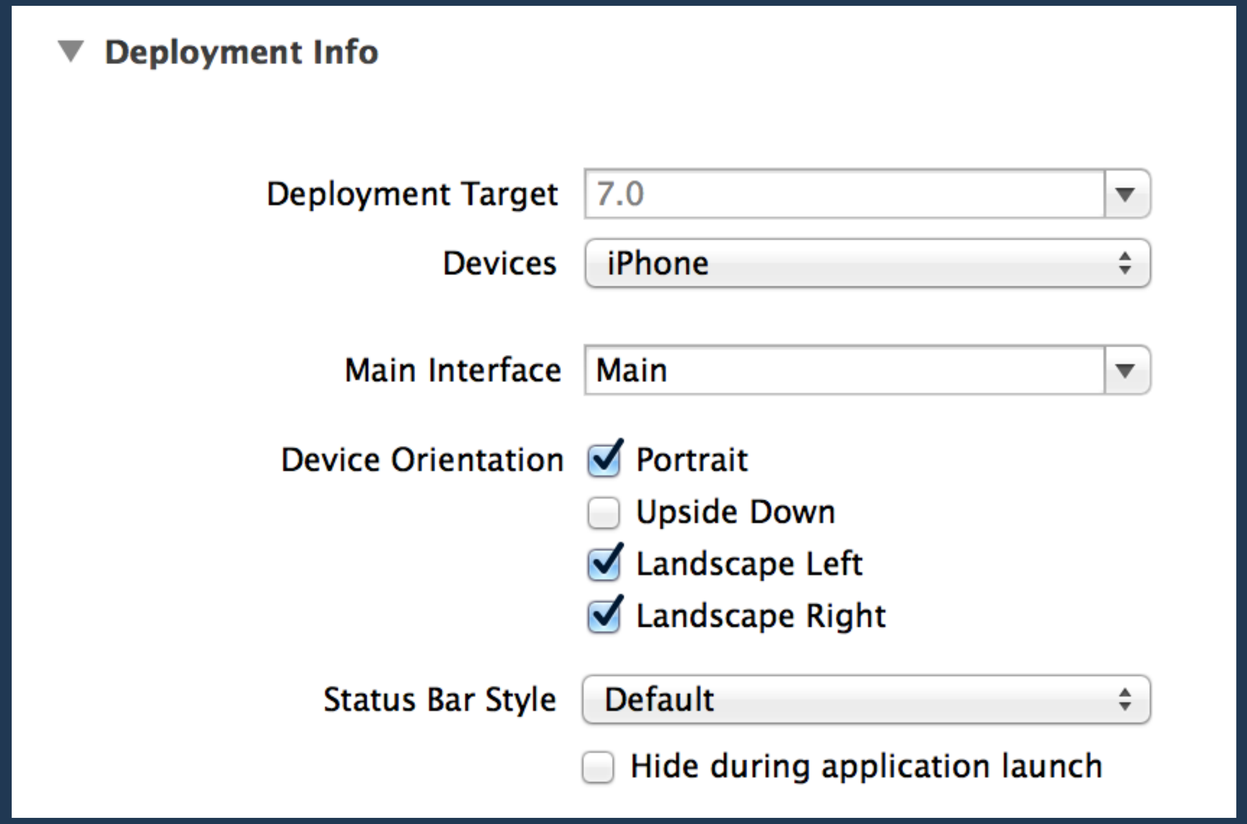


Declare your support

- Application-wide
- View controller

Declare your support

`Info.plist`



Declare your support

```
@implementation MainViewController

- (NSUInteger)supportedInterfaceOrientations
{
    return UIInterfaceOrientationMaskLandscapeRight | UIInterfaceOrientationMaskLandscapeLeft;
}

@end
```

Declare your support

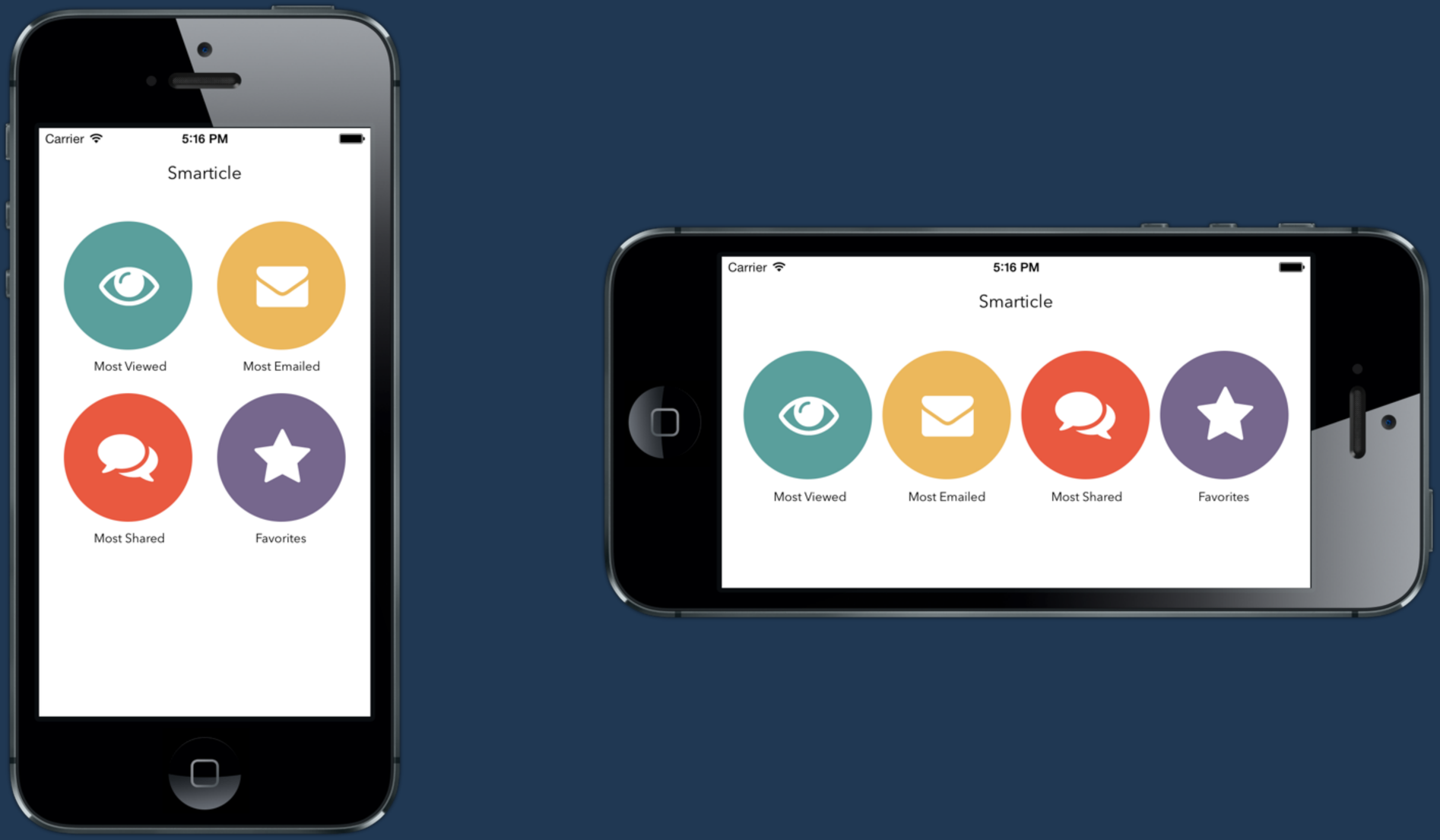
	Info.plist	supportedInterfaceOrientations	Supported
Portrait	**YES**	**YES**	**YES**
Portrait Upside Down		**YES**	
Landscape Left	**YES**		
Landscape Right	**YES**		

Declare your support

- Decide at the app-level (`Info.plist`) which orientations to support
- Design view controllers to be reusable (orientation-agnostic)
- In general, maintain the same orientation support throughout your app
- Special orientation support for full screen modals

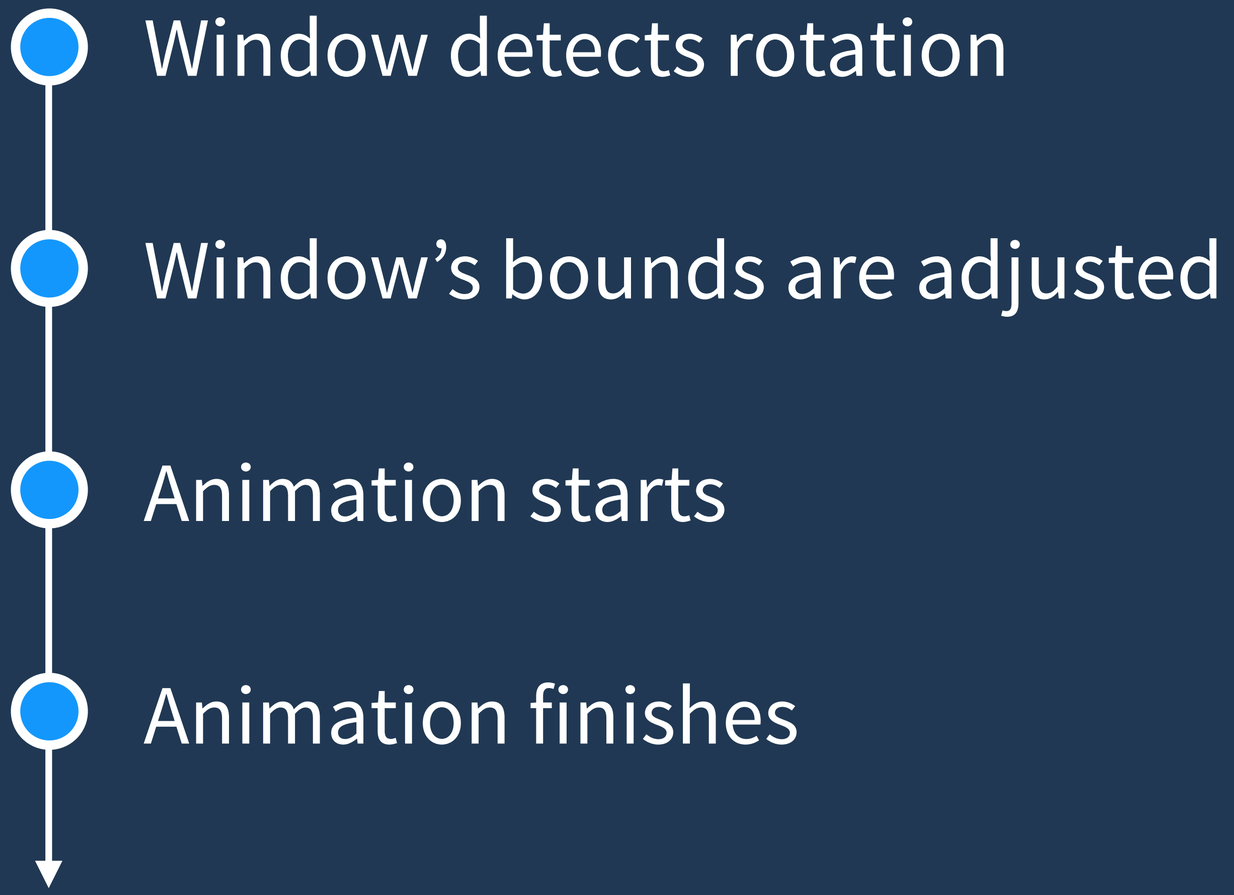
```
presentViewController:animated:completion:
preferredInterfaceForPresentation
```

Autorotation

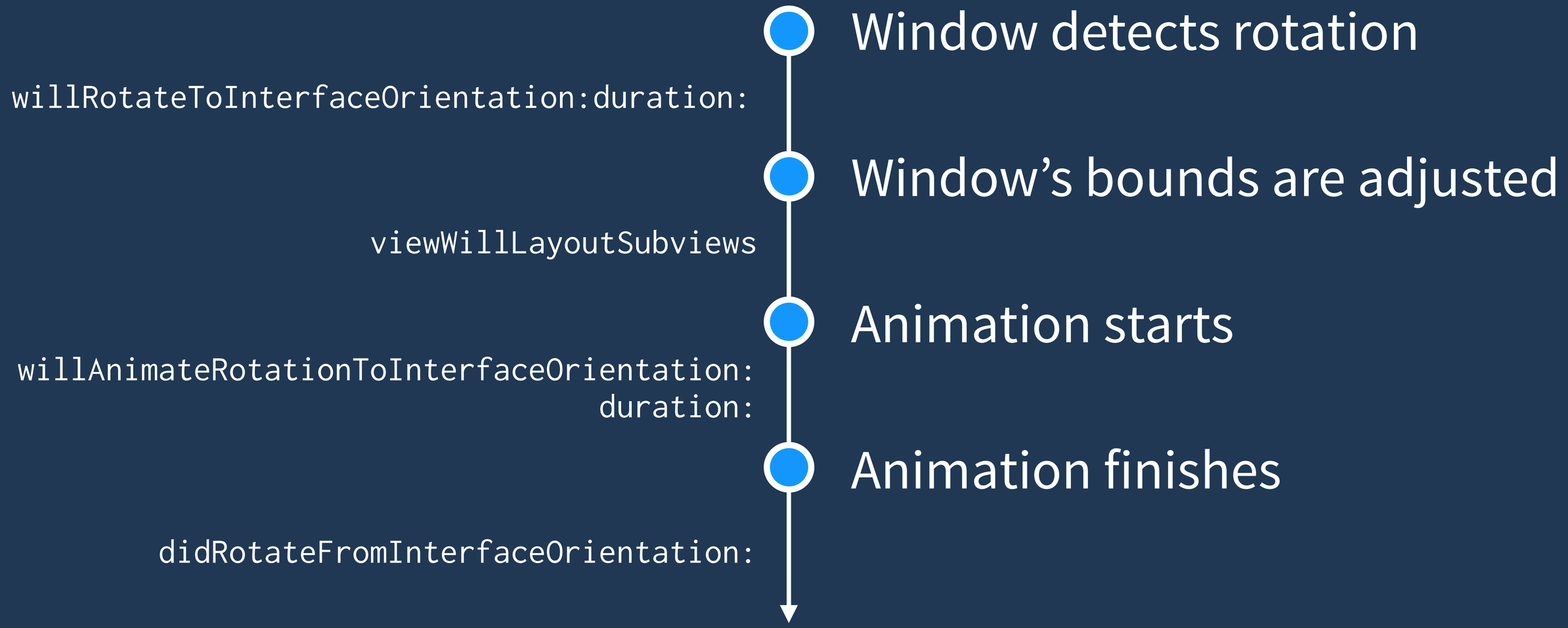

Carrier
5:16 PM
Smarticle
Most Viewed
Most Emailed
Most Shared
Favorites
Carrier
5:16 PM
Smarticle
Most Viewed
Most Emailed
Most Shared
Favorites

Autorotation

Autorotation



Autorotation



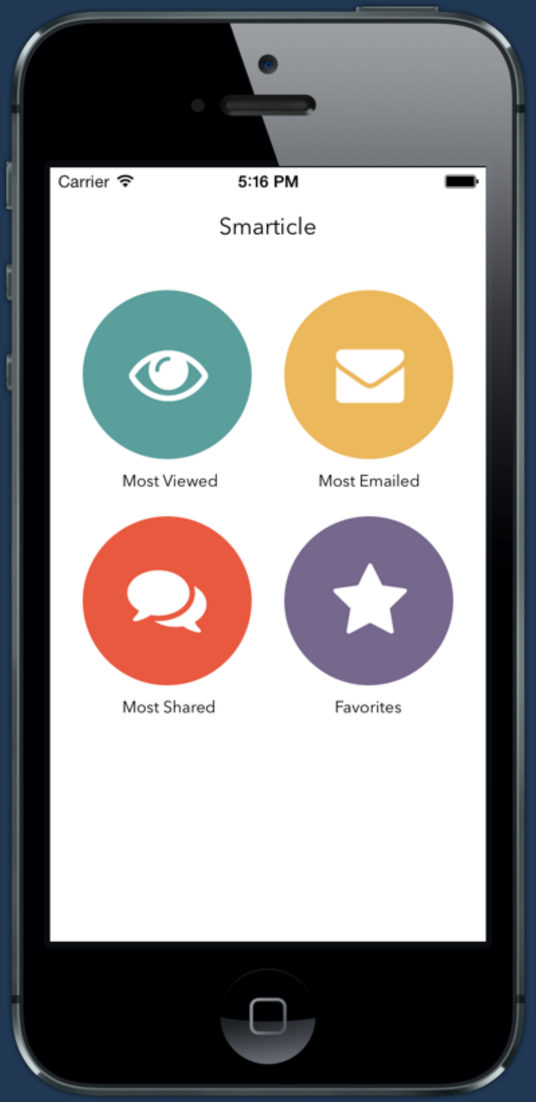
Autorotation

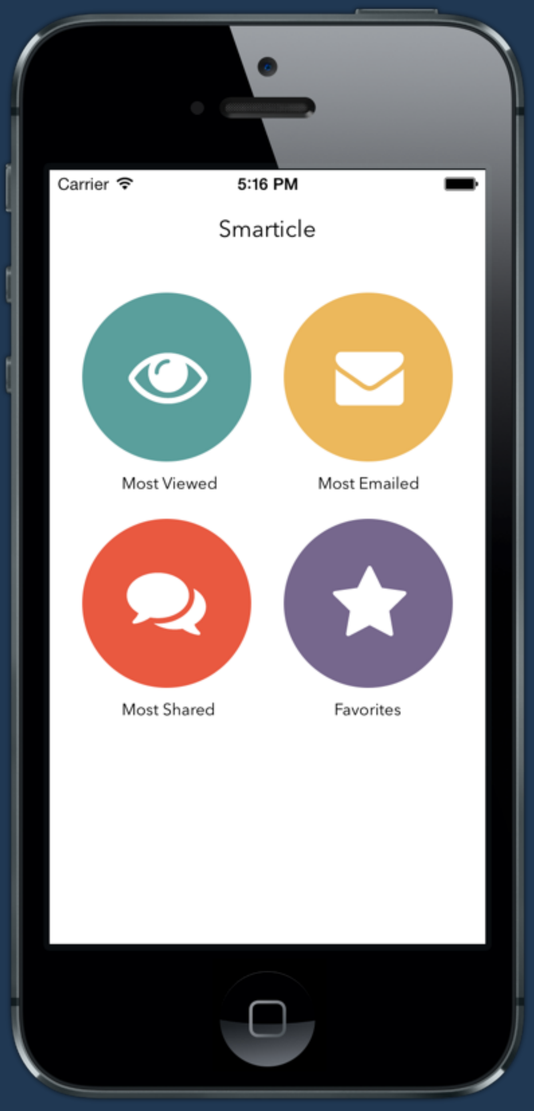
willRotateToInterfaceOrientation:duration:

viewWillLayoutSubviews

willAnimateRotationToInterfaceOrientation:
duration:

didRotateFromInterfaceOrientation:


Carrier
5:16 PM
Smarticle
Most Viewed
Most Emailed
Most Shared
Favorites


Carrier
5:16 PM
Smarticle
Most Viewed
Most Emailed
Most Shared
Favorites

Window detects rotation

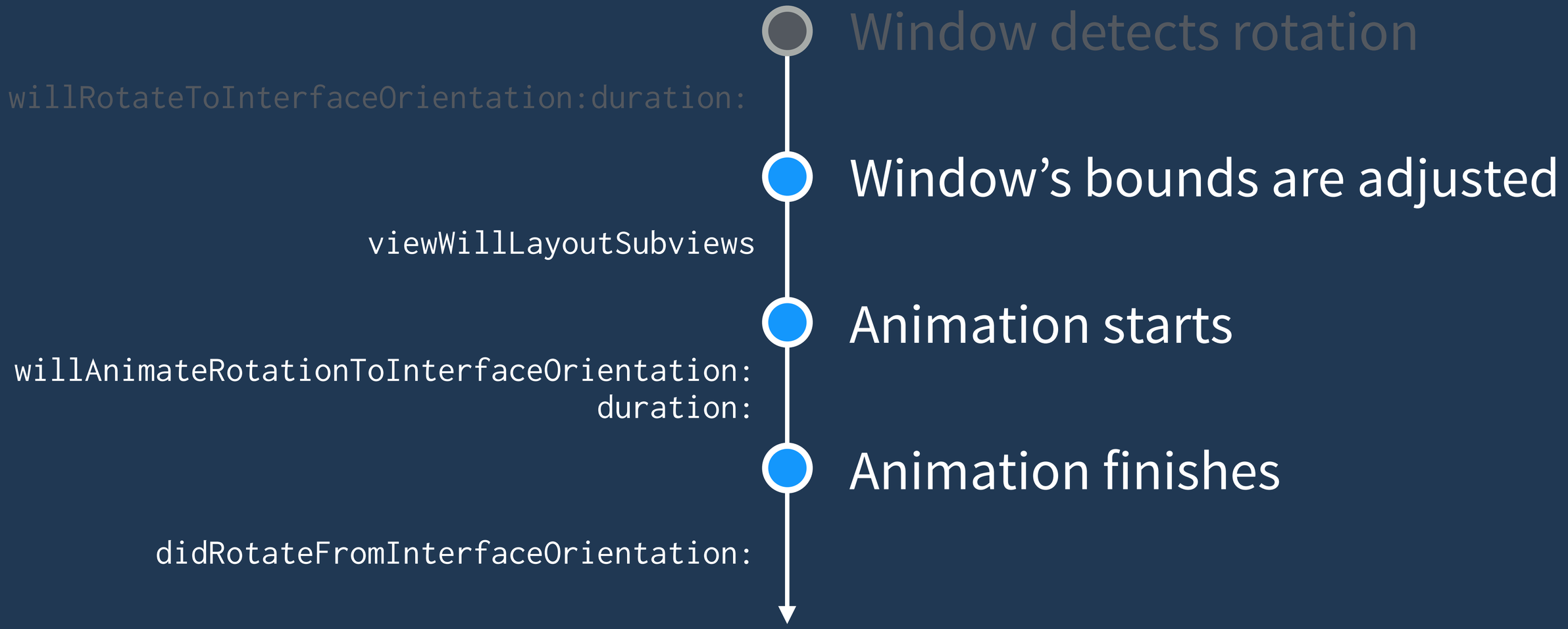

```
- (void)willRotateToInterfaceOrientation:(UIInterfaceOrientation)toInterfaceOrientation
                                duration:(NSTimeInterval)duration
{
    [super willRotateToInterfaceOrientation:toInterfaceOrientation duration:duration];

    // Hide expensive views, stop media playback, etc.
}
```

Window detects rotation

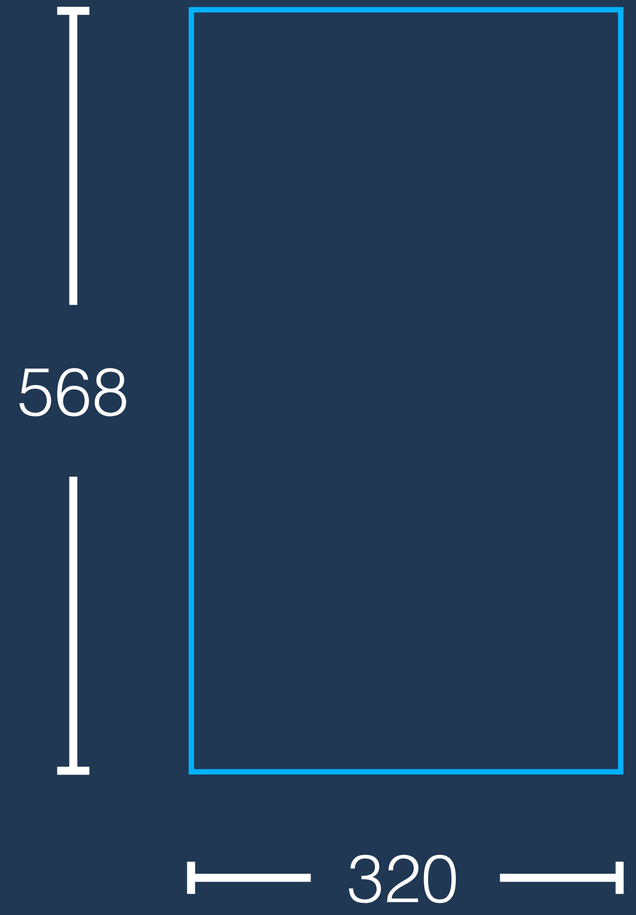
```
- (void)willRotateToInterfaceOrientation:(UIInterfaceOrientation)toInterfaceOrientation
                                duration:(NSTimeInterval)duration
```

- Orientation has not changed yet
- Root View Controller’s view bounds haven’t changed yet
- Avoid adjusting your layout

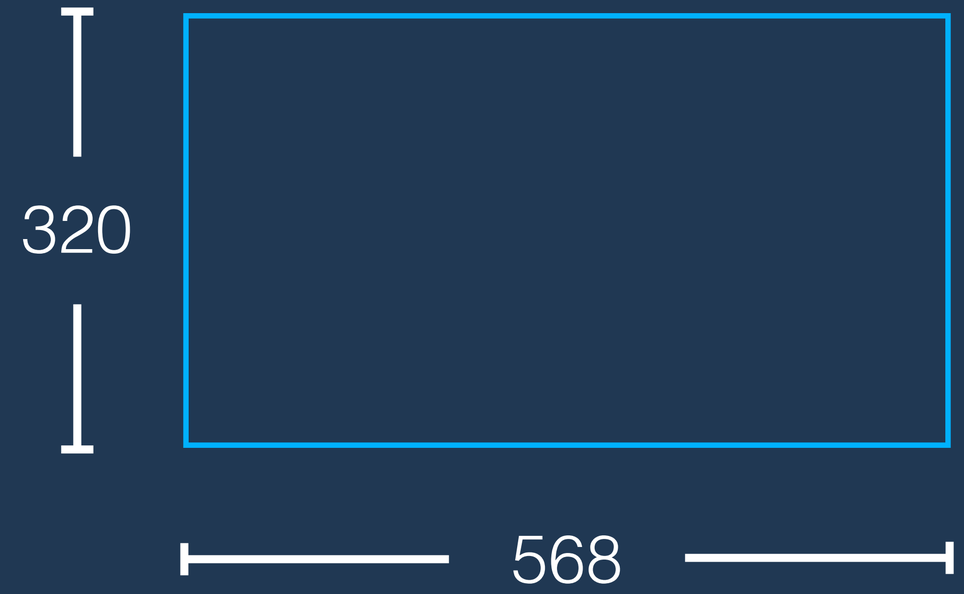
Autorotation



Windows bounds are adjusted



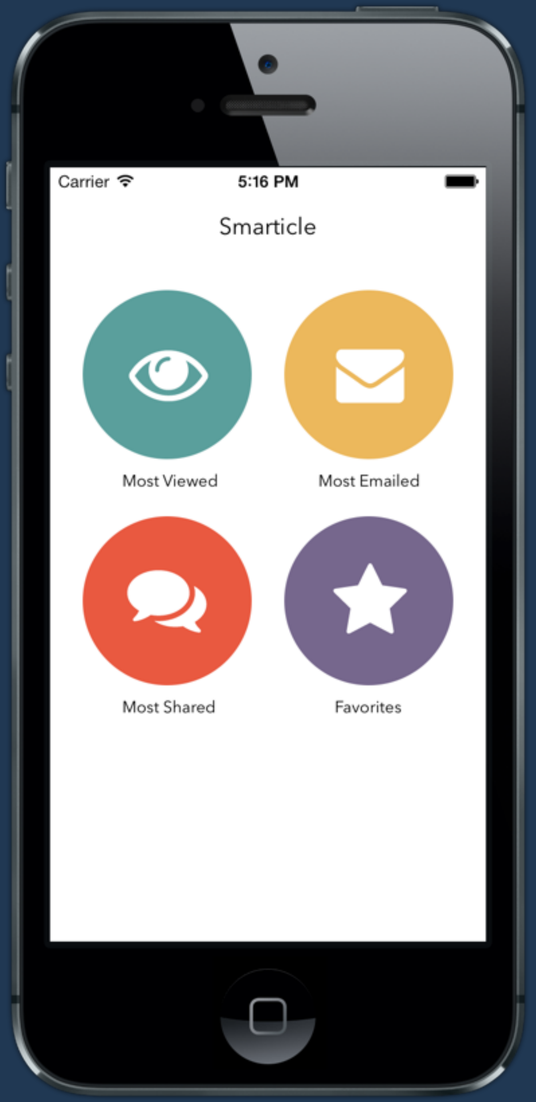
Windows bounds are adjusted

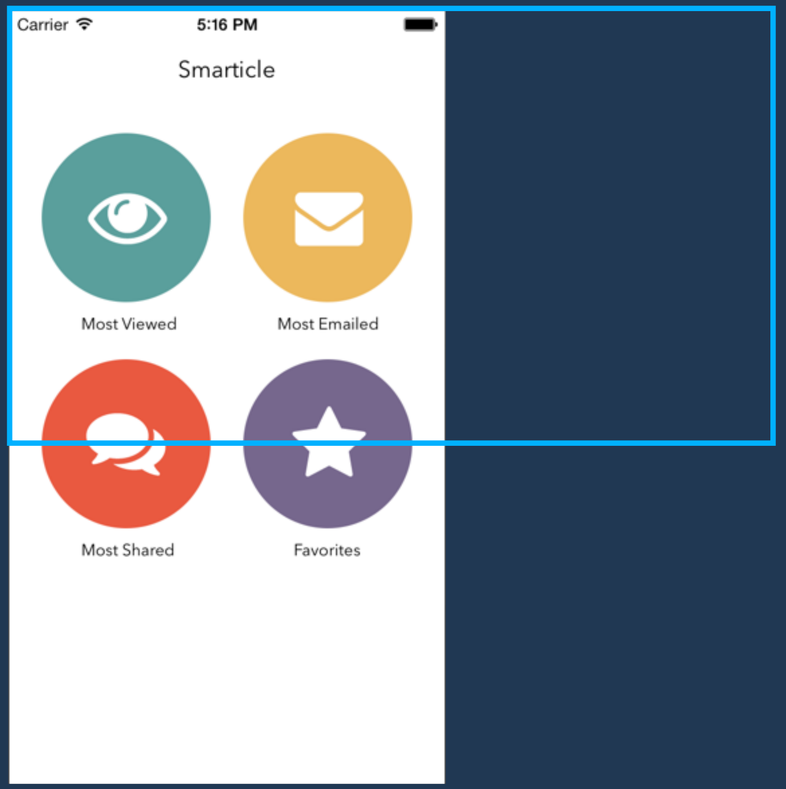


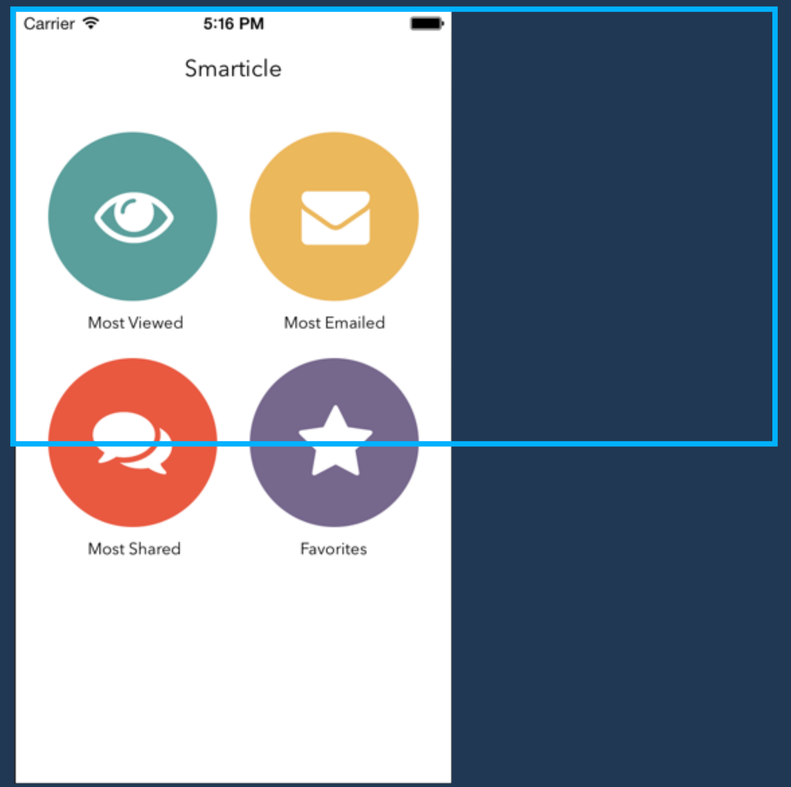
Adjust layout

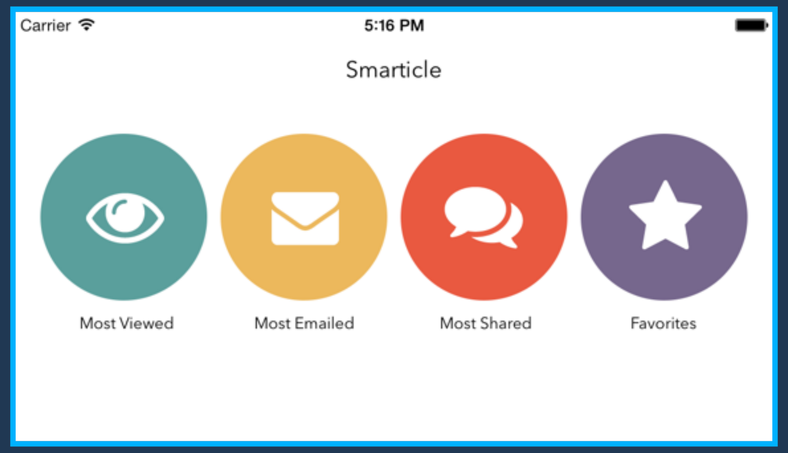
Root view controller receives:

```
- (void)viewWillLayoutSubviews
- (void)updateViewConstraints
- (void)viewDidLayoutSubviews
```


Carrier
5:16 PM
Smarticle
Most Viewed
Most Emailed
Most Shared
Favorites


Carrier
5:16 PM
Smarticle
Most Viewed
Most Emailed
Most Shared
Favorites


Carrier
5:16 PM
Smarticle
Most Viewed
Most Emailed
Most Shared
Favorites

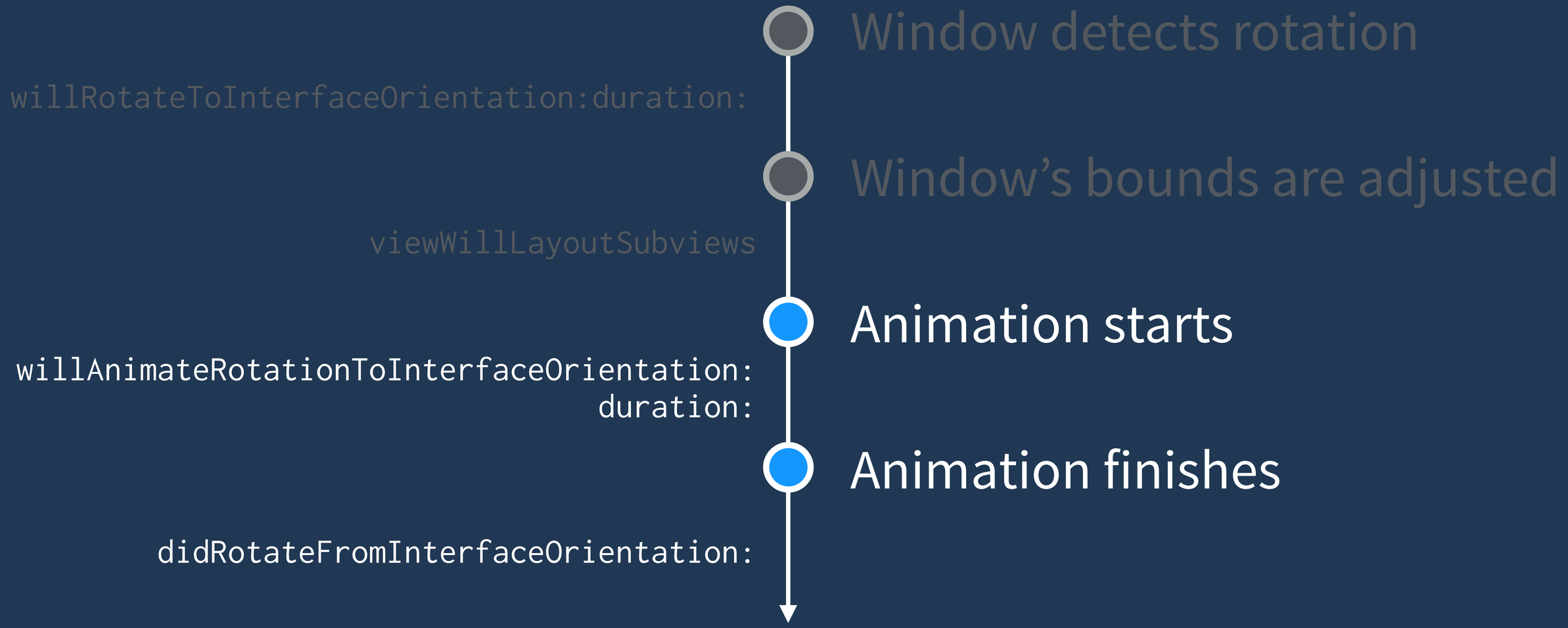

Carrier
5:16 PM
Smarticle
Most Viewed
Most Emailed
Most Shared
Favorites


Carrier
5:16 PM
Smarticle
Most Viewed
Most Emailed
Most Shared
Favorites

Providing different layouts

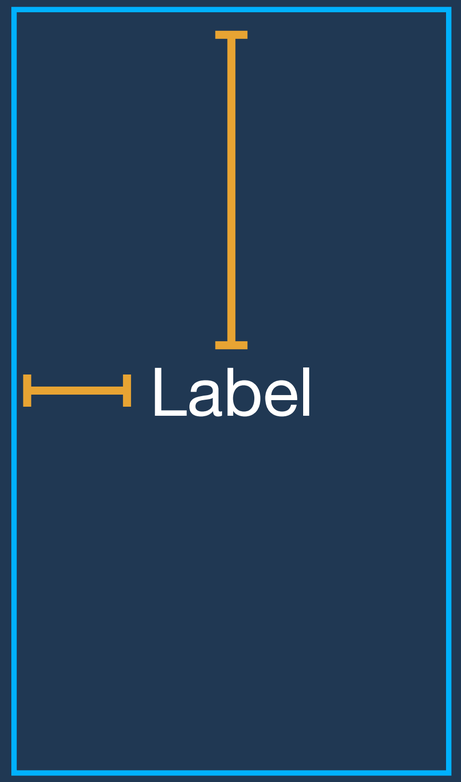
- If not using autolayout, adjust frames manually
- Create separate XIBs for portrait and landscape
- Create separate view controllers in your storyboard and use `performSegueWithIdentifier:`
- Make changes to your auto layout constraints

Autorotation

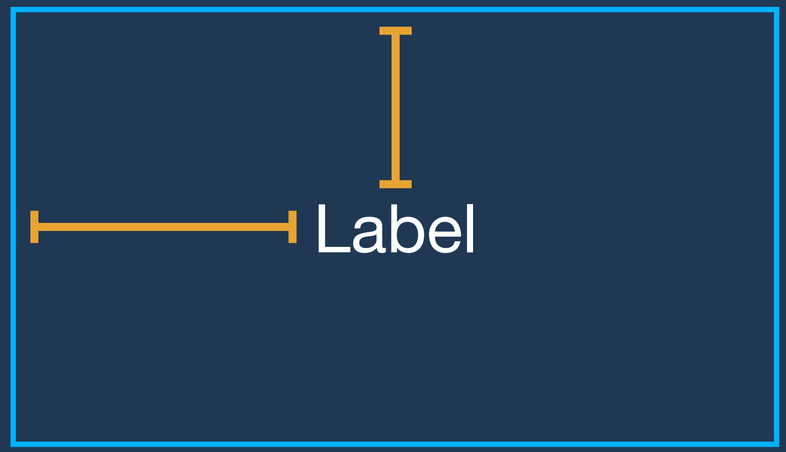


Animation starts

Existing layout



New layout

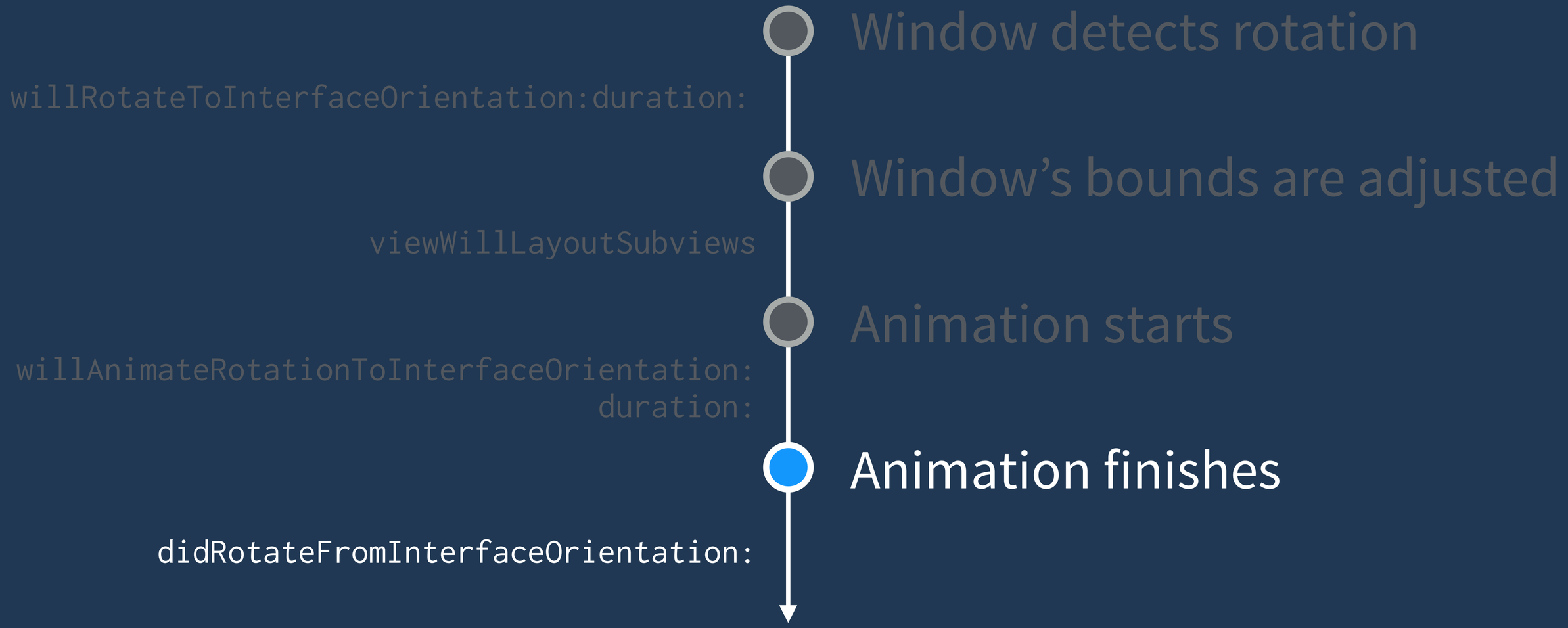


Animation starts

Animation starts


```
- (void)willAnimateRotationToInterfaceOrientation:(UIInterfaceOrientation)toInterfaceOrientation
                                         duration:(NSTimeInterval)duration
{
    // Any (animatable) view property changes here will be automatically animated
}
```

Autorotation



Animation finishes

Animation finishes


```
- (void)didRotateFromInterfaceOrientation:(UIInterfaceOrientation)fromInterfaceOrientation
{
    [super didRotateFromInterfaceOrientation:fromInterfaceOrientation]

    // Undo what was done in willRotateToInterfaceOrientation
}
```

Deprecation

- Apple is somewhat schizophrenic when it comes to UIViewController rotation-handling
- Don’t get too attached to any particular UIViewController rotation methods

An API graveyard

- didAnimateFirstHalfOfRotationToInterfaceOrientation (iOS 5.0)
- willAnimateFirstHalfOfRotationToInterfaceOrientation:duration (iOS 5.0)
- willAnimateSecondHalfOfRotationFromInterfaceOrientation:duration (iOS 5.0)
- automaticallyForwardAppearanceAndRotationMethodsToChildViewControllers (iOS 6.0)
- shouldAutorotateToInterfaceOrientation: (iOS 6.0)
- willRotateToInterfaceOrientation:duration: (iOS 8.0)
- willAnimateRotationToInterfaceOrientation:duration: (iOS 8.0)
- didRotateFromInterfaceOrientation: (iOS 8.0)

Demo: MainViewController

Custom Views

Custom Views

Don’t bother.

Customize existing UIKit views wherever you can.

Custom Views

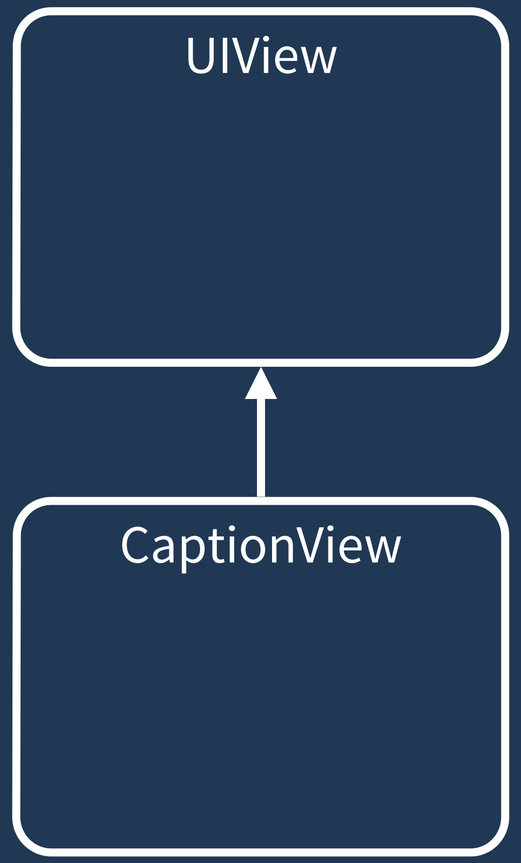
- Composite views
- Custom-drawn views
- Combination of the two

Custom Views

Composite Views  Custom-drawn Views

Combination of the two

Custom Views



Composite Views

Composite Views

Composite Views

- Still taking advantage of built in views
- Larger reusable units
- Thins out your view controllers

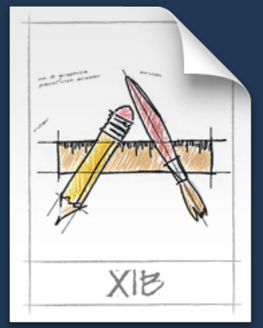
Composite Views

- Create subviews during your custom view’s initialization
- Layout those views in `layoutSubviews`
- Logic to handle touch events or other interactions
- Avoid unnecessarily implementing - `drawRect:`

Initialization

From code:

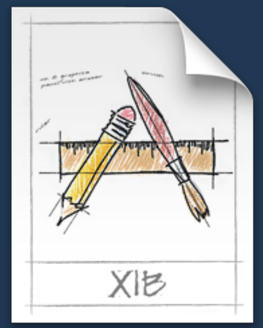
```
- (id)initWithFrame:(CGRect)frame
```



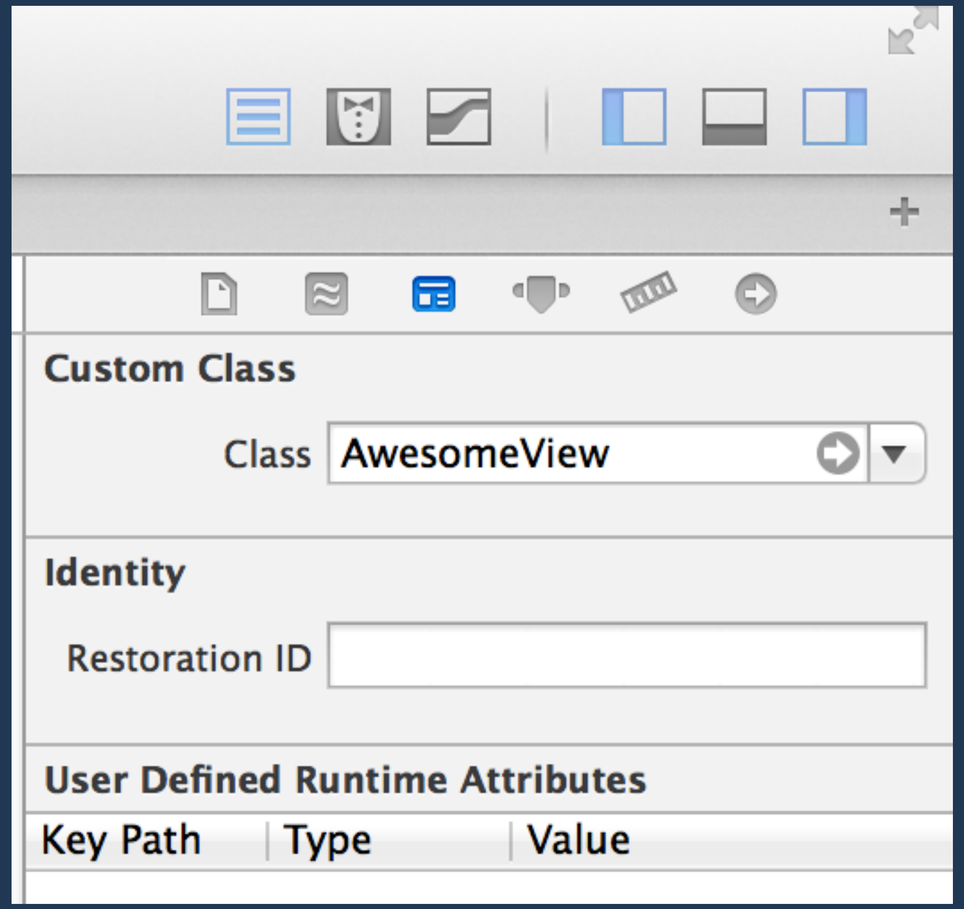
From xib or storyboard:

```
- (id)initWithCoder:(NSCoder *)aDecoder;
```

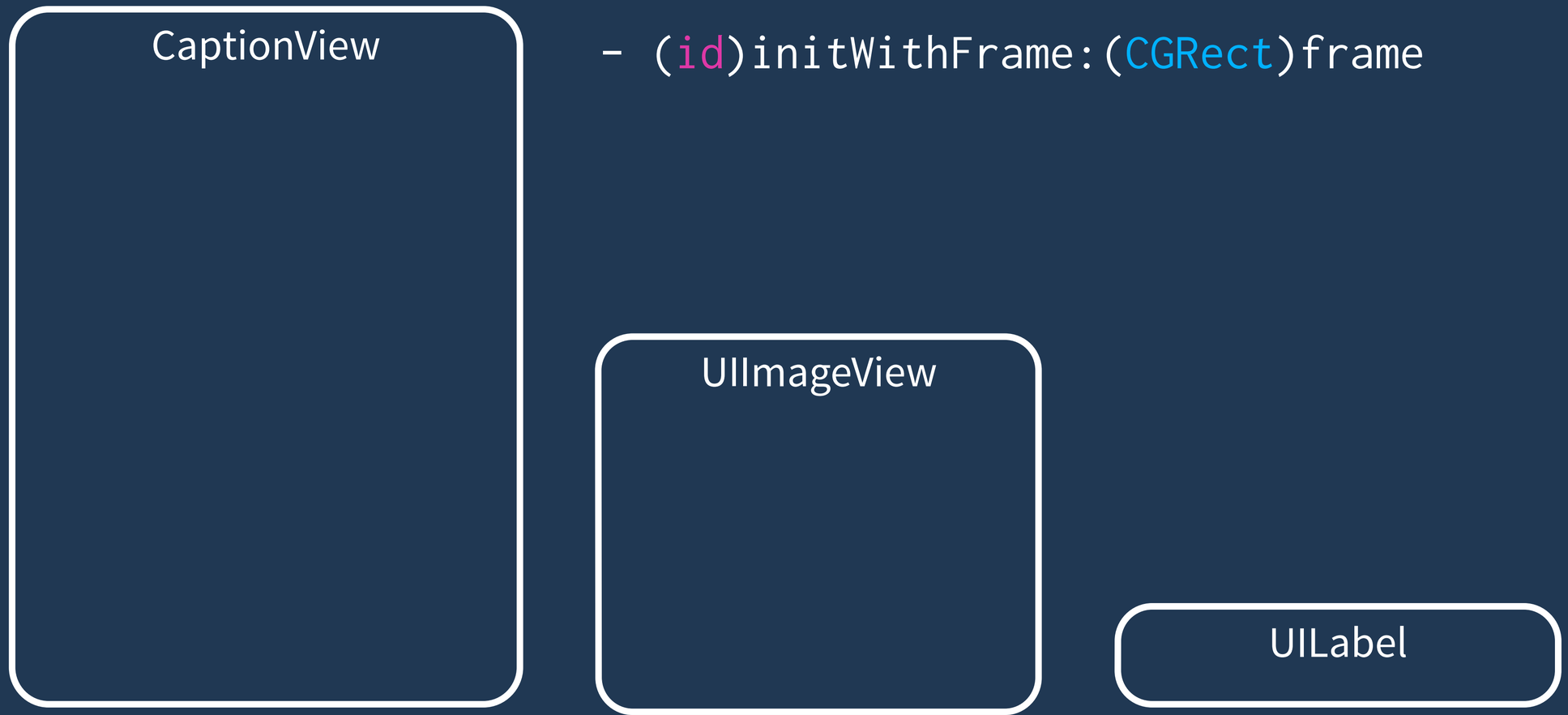
Initialization



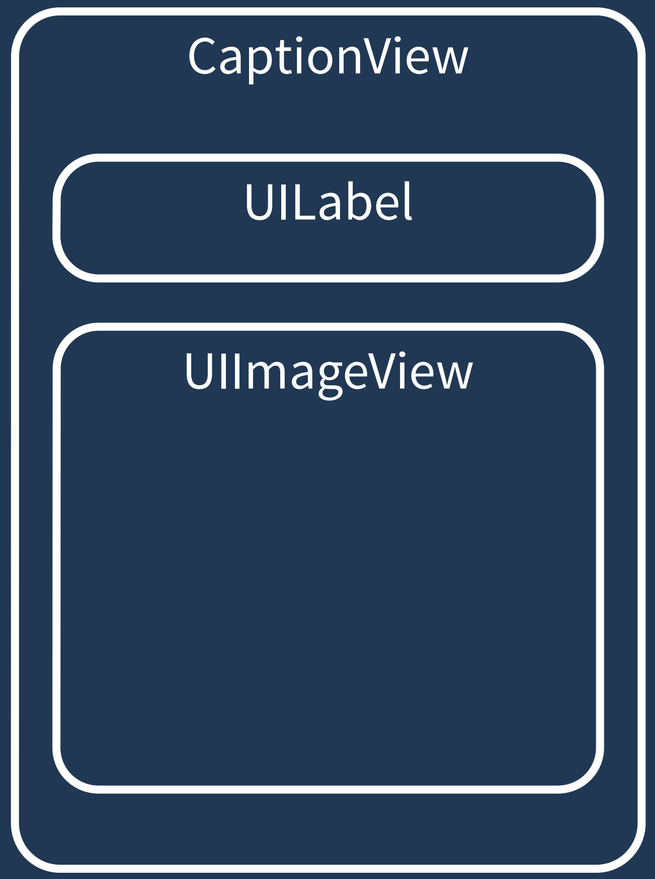

```
- (id)initWithCoder:(NSCoder *)aDecoder;
```



Initialization



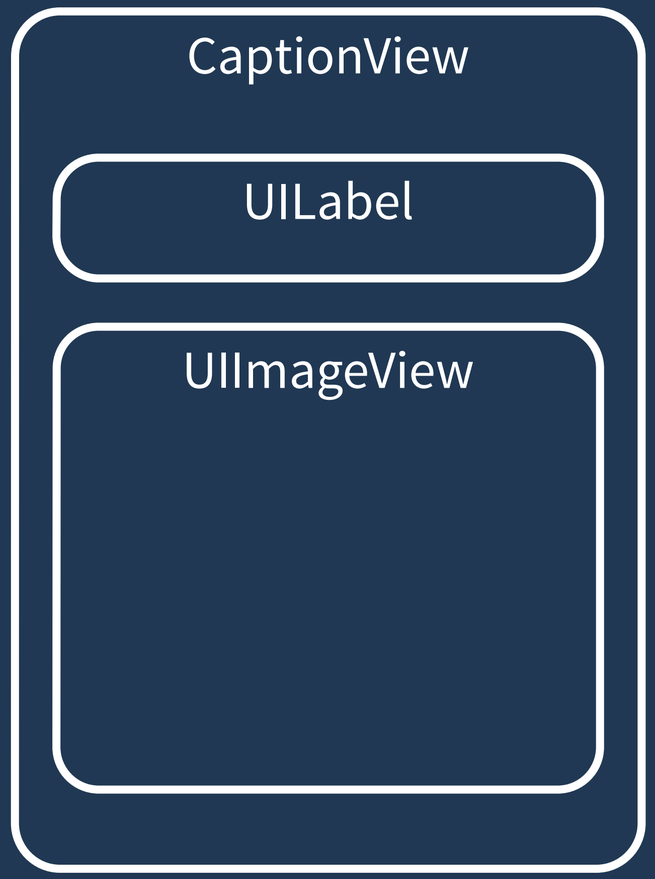
Initialization




```
- (id)initWithFrame:(CGRect)frame
```

- Configure default view properties
- Initialize and configure subviews
- Setup the view hierarchy

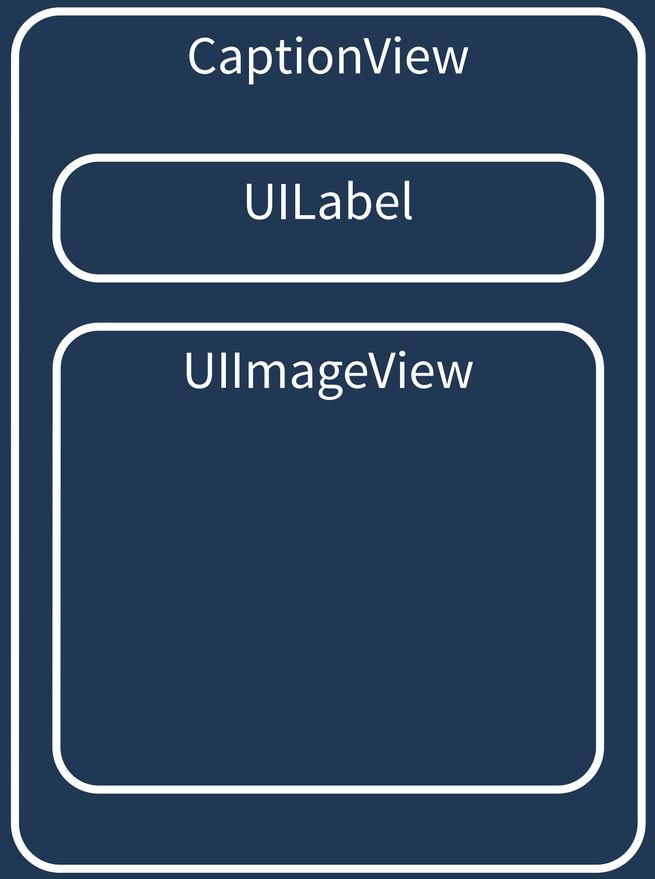
Layout




```
- (void)layoutSubviews
```

- Adjust the frames of all subviews
- Called whenever layout is required by a size change or when setNeedsLayout is called
- Generally, do not call layoutSubviews yourself.

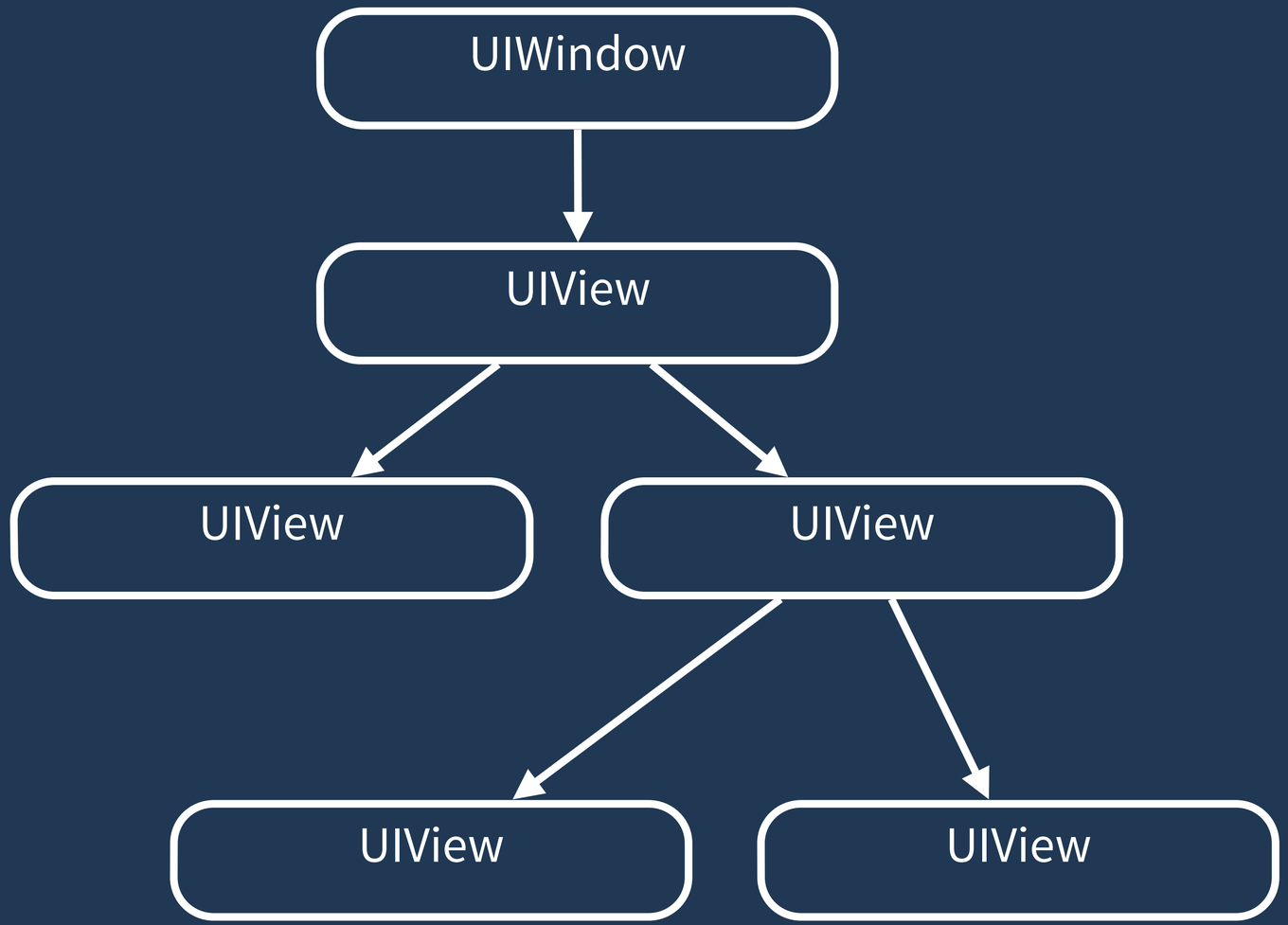
Layout




```
- (void)updateConstraints
```

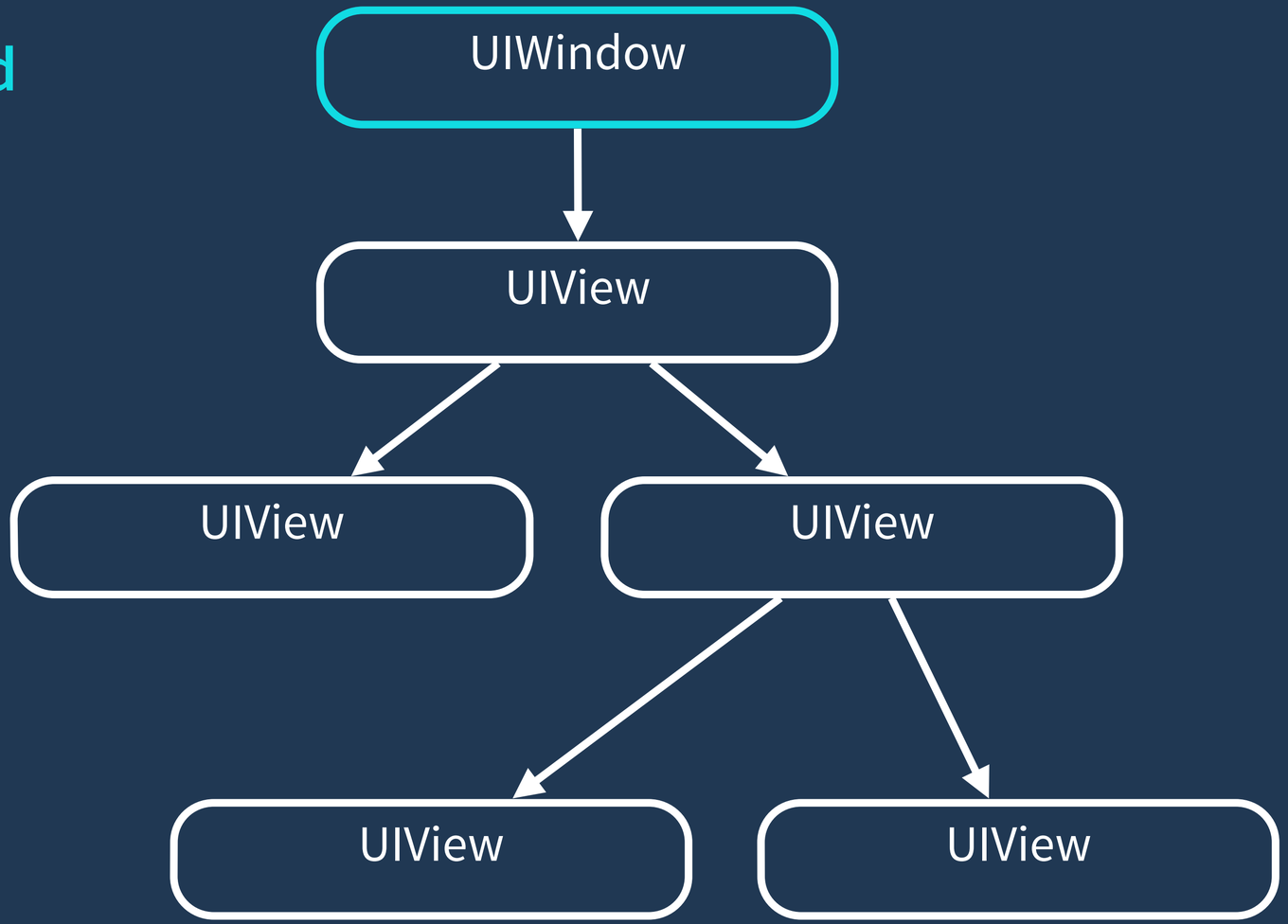
- Setup the constraints of all subviews
- Called whenever layout is required by a size change or when setNeedsUpdateConstraints is called
- Generally, do not call updateConstraints yourself.

Layout



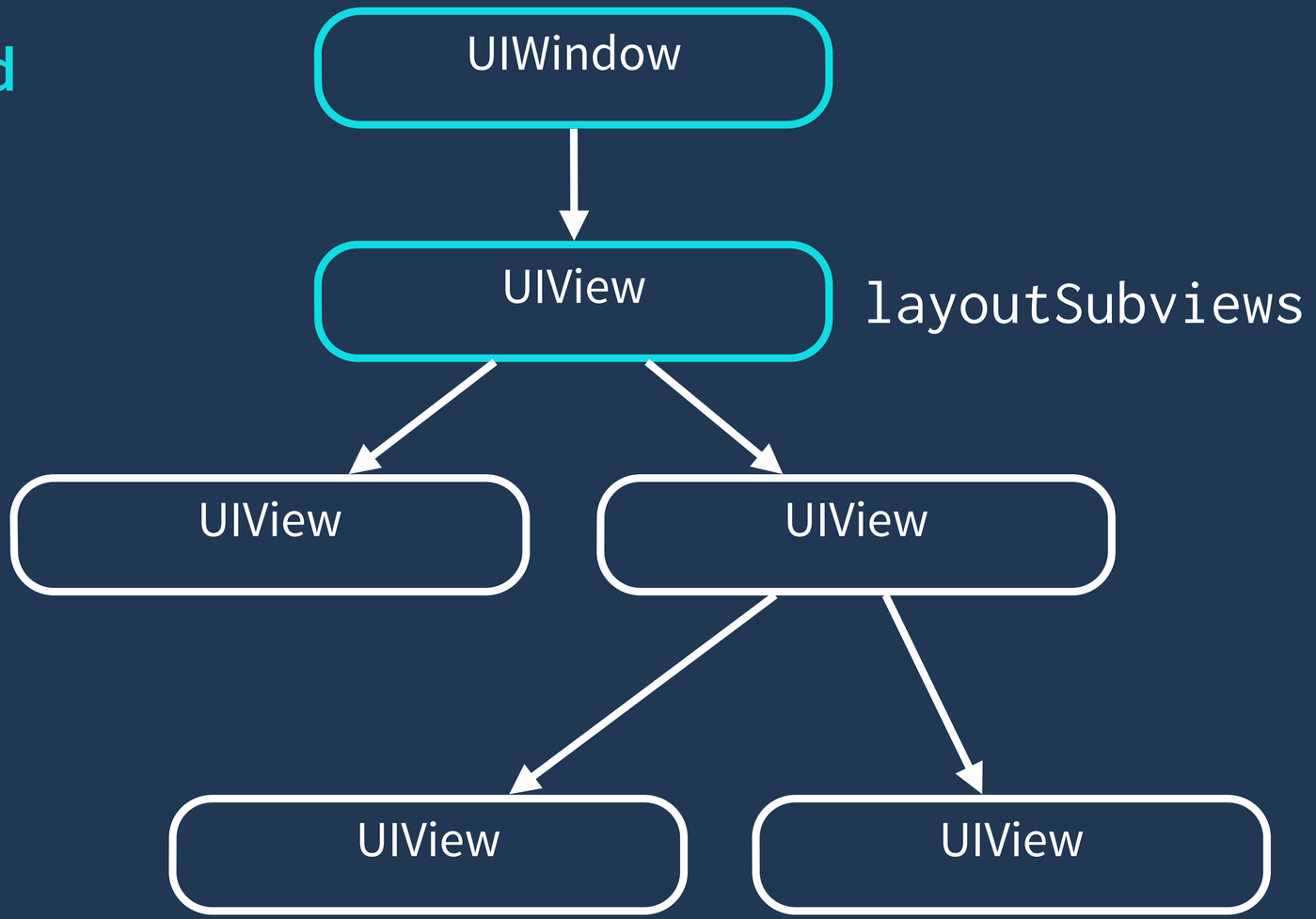
Layout

Bounds changed



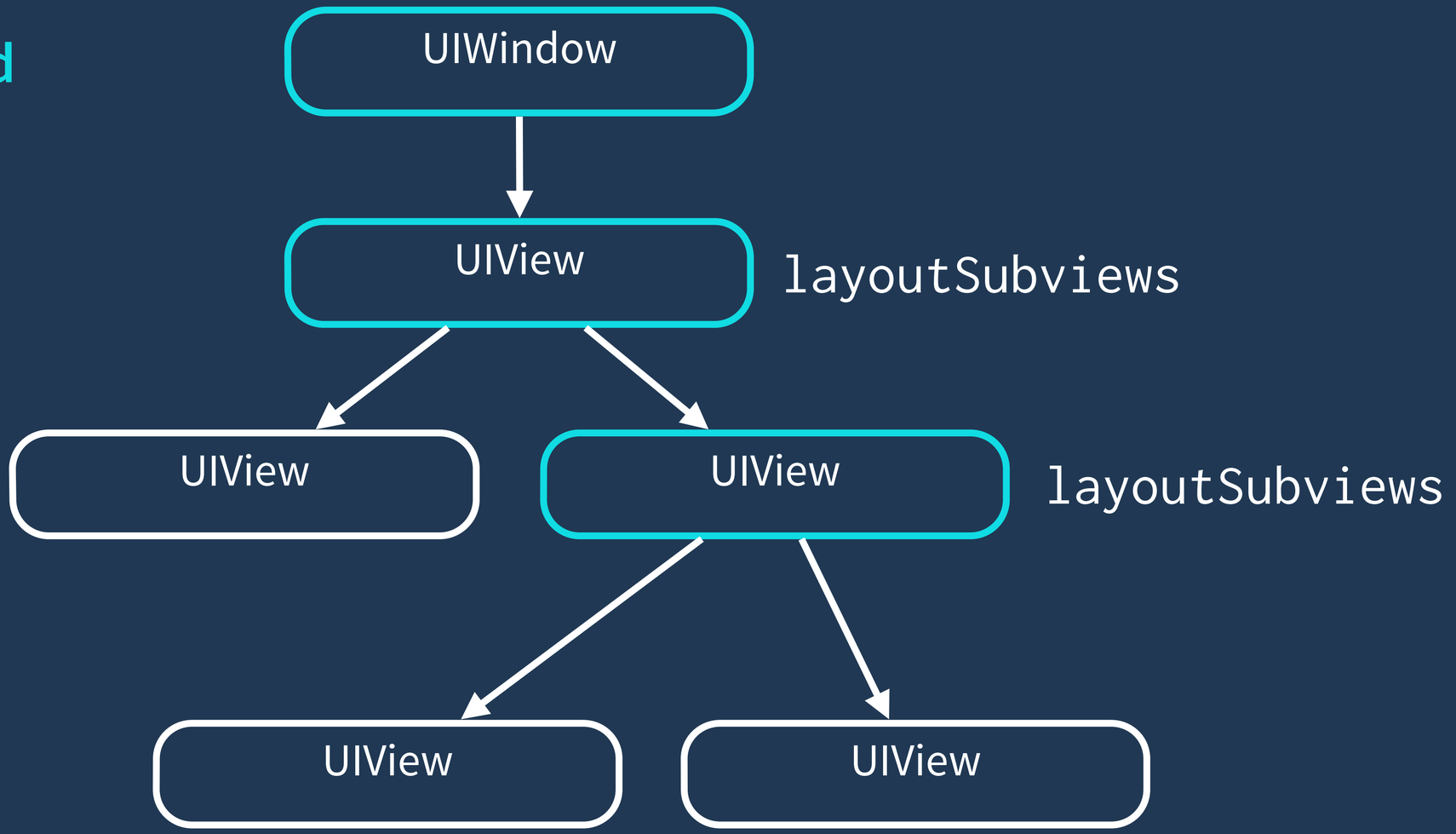
Layout

Bounds changed



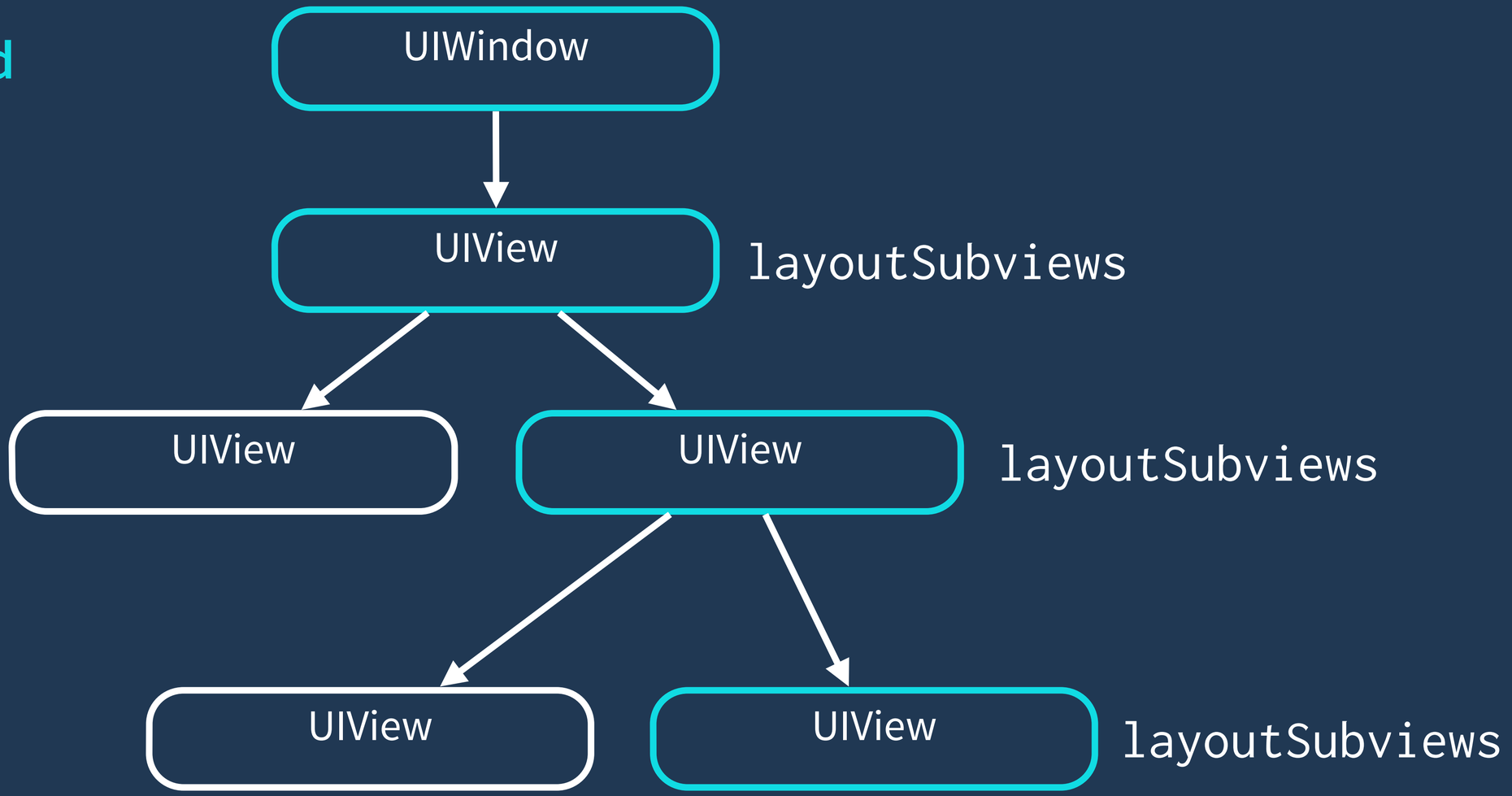
Layout

Bounds changed



Layout

Bounds changed



Custom-drawn Views

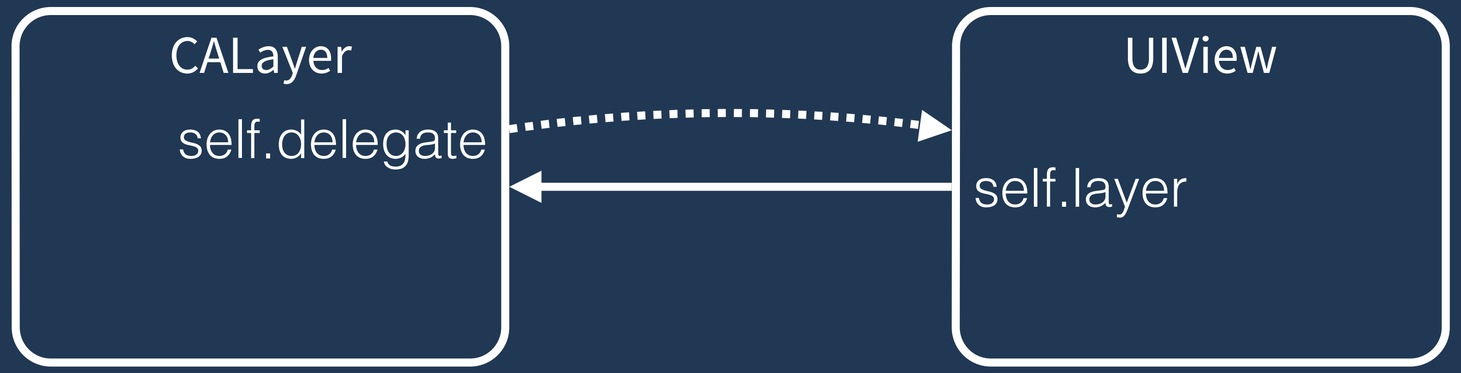
Composite Views

- Override - `drawRect:`
- Logic to handle touch events or other interactions
- Decide when your view should be redrawn

Custom-drawn views

```
- (void)drawRect:(CGRect)rect
```

Layers



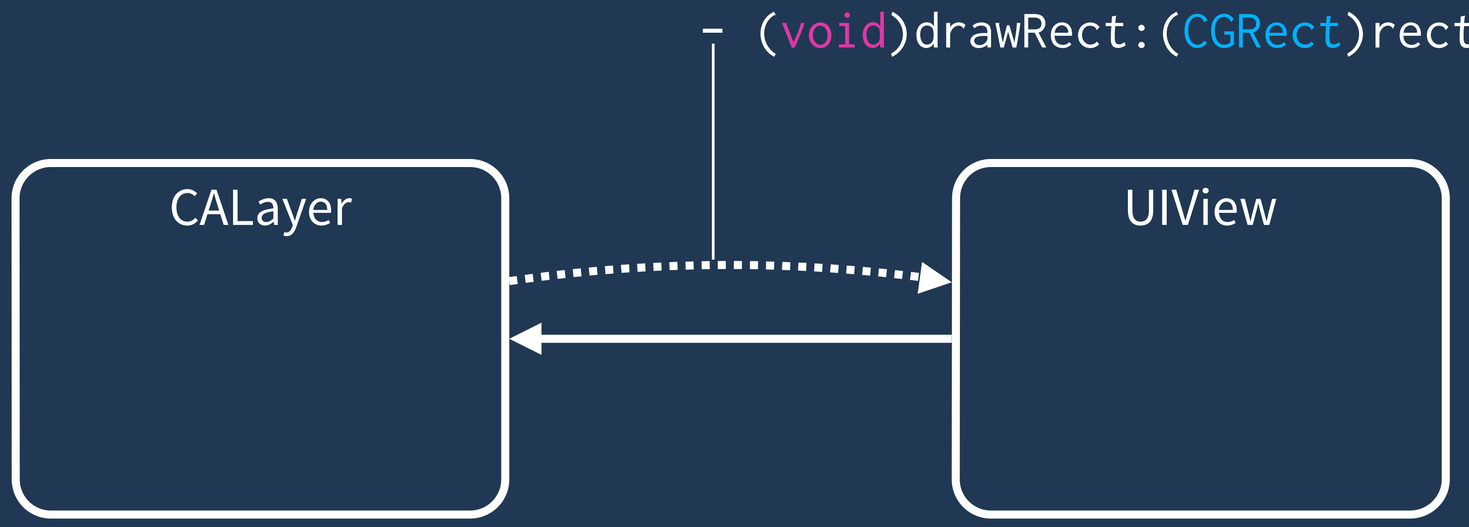
Layers

The canvas

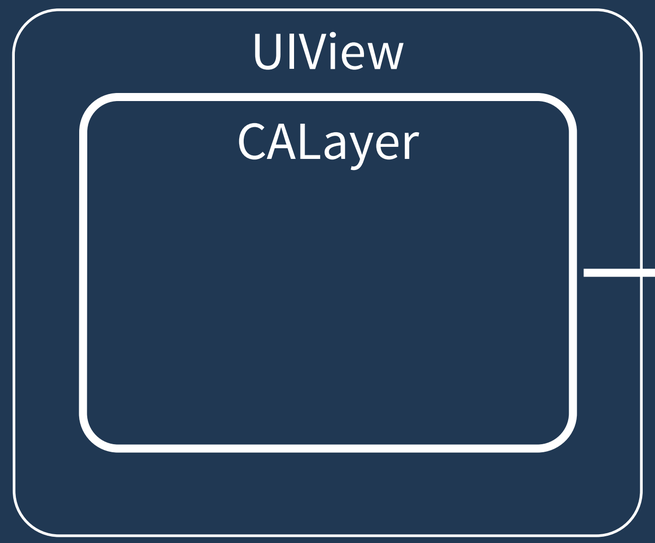
Layers

The painter

Layers



Layers



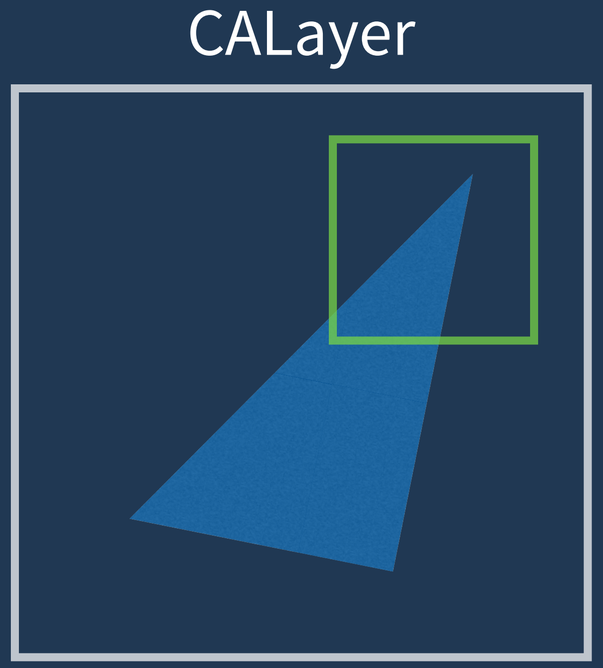
backgroundColor
borders
shadows
rounded corners
masking
animation
creme filling

Drawing

```
- (void)drawRect:(CGRect)rect
```

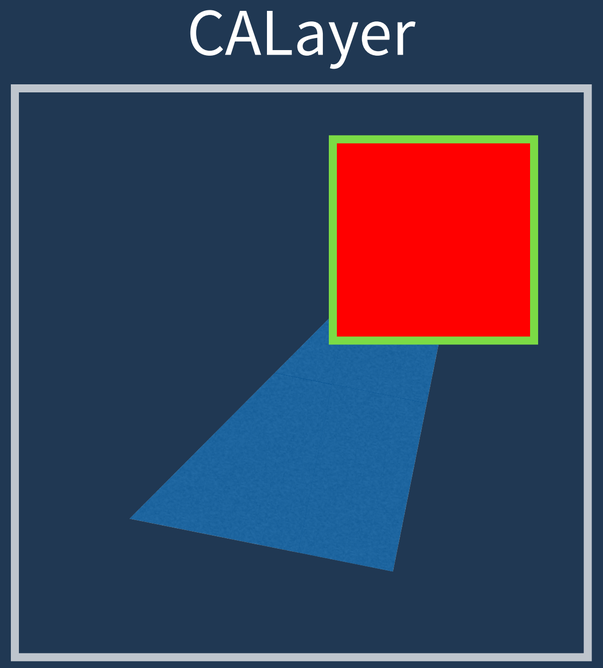
```
- (void)drawRect:(CGRect)rect
{
    CGContextRef context = UIGraphicsGetCurrentContext();

    CGColorRef redCGColor = [UIColor redColor].CGColor;
    CGContextSetFillColorWithColor(context, redCGColor);

    CGContextFillRect(context, rect);
}
```

drawRect




```
- (void)drawRect:(CGRect)rect
```

drawRect




```
- (void)drawRect:(CGRect)rect
```

Core Graphics

- Fill rects
- Draw lines
- Paths
- Transforms
- Patterns
- Gradients

```
#import <QuartzCore/QuartzCore.h>
```

- Bitmaps
- Shadows
- Masks
- PDFs
- Text
- Transforms

Redrawing

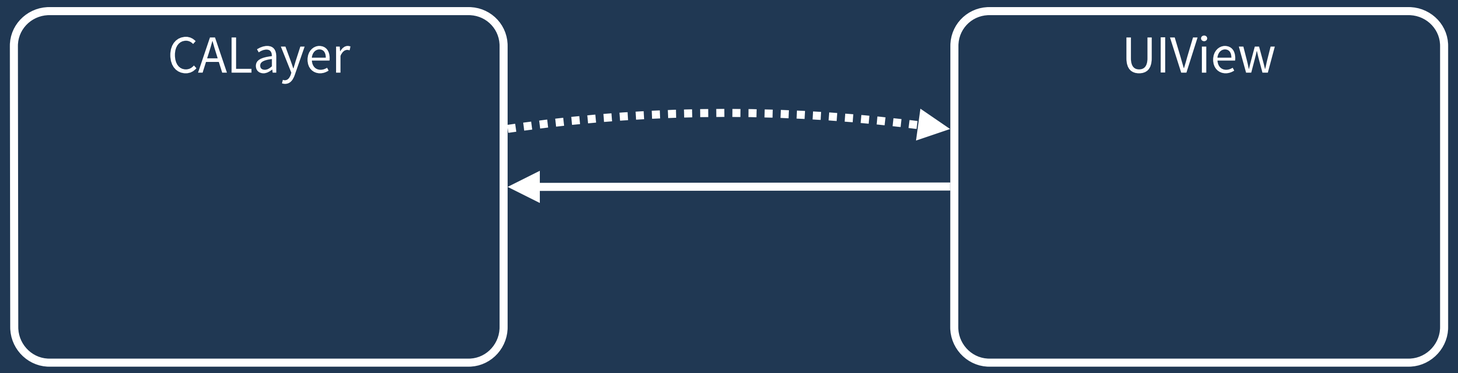
- Drawing is expensive, avoid if possible
- Never call drawRect: directly, call -setNeedsDisplay
- Set the contentMode to handle resizing

Demo: LightningBoltView

Animation

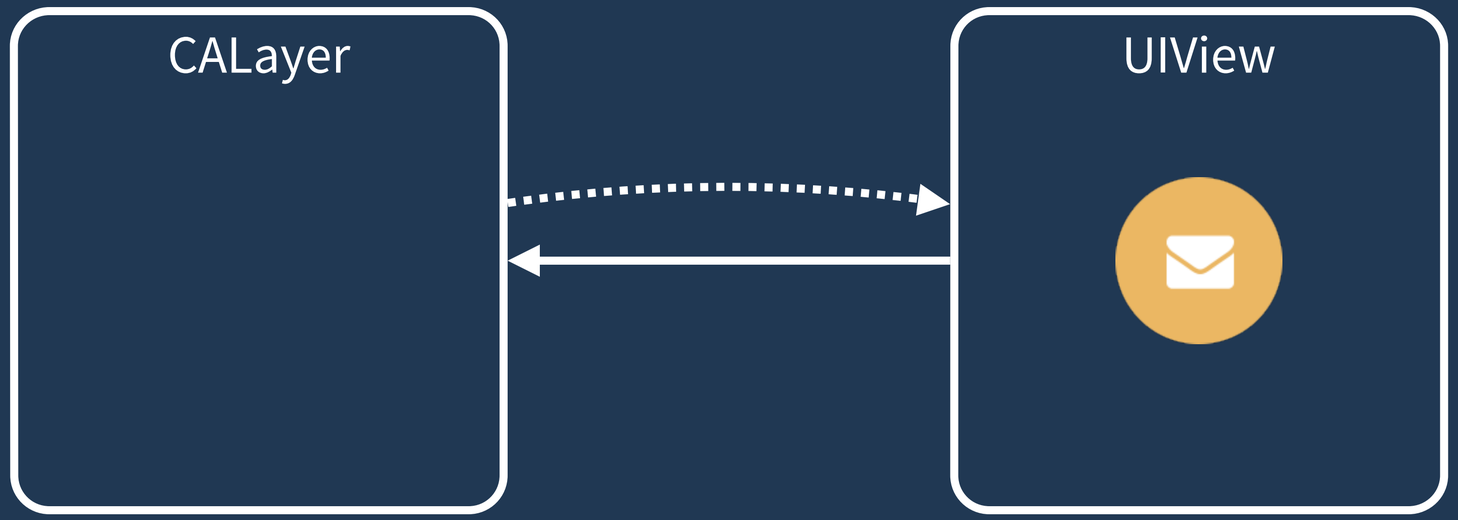
CoreAnimation

CoreAnimation

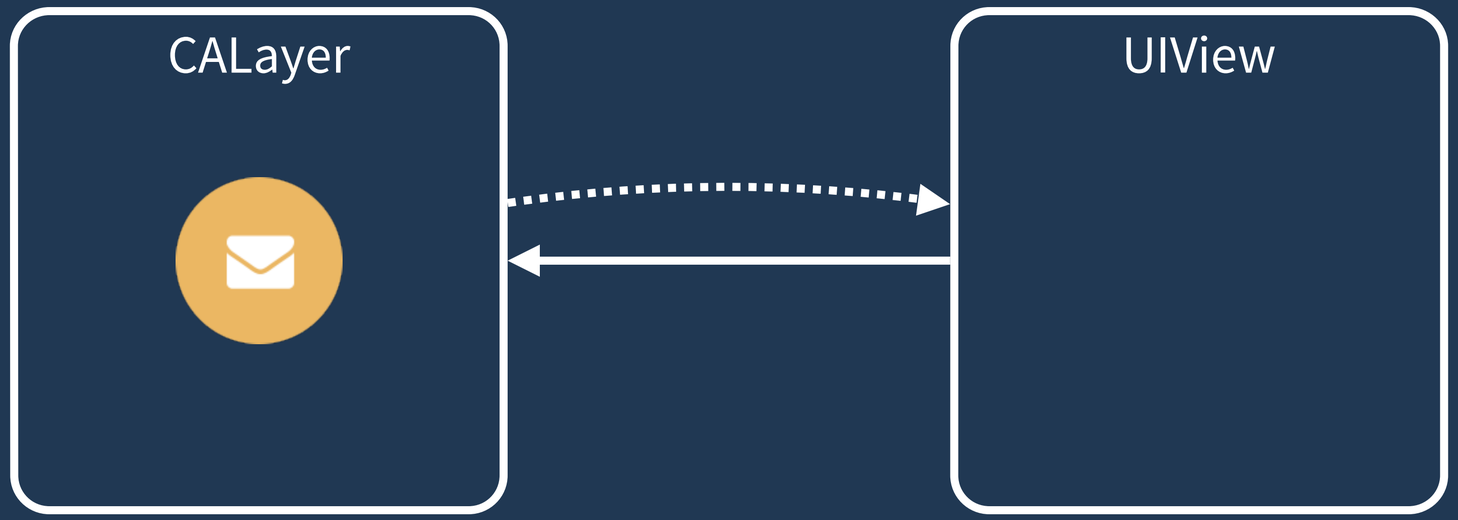


Hardware	Software
Animation Compositing	Drawing

CoreAnimation

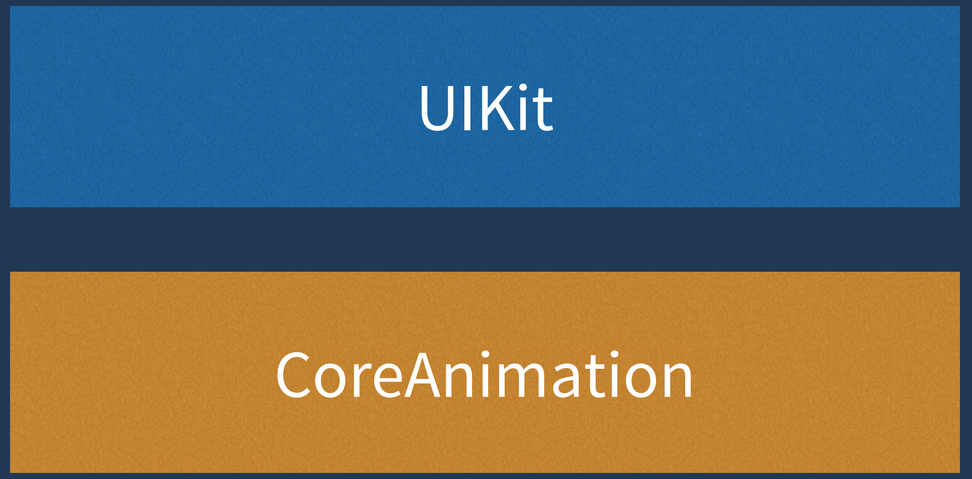


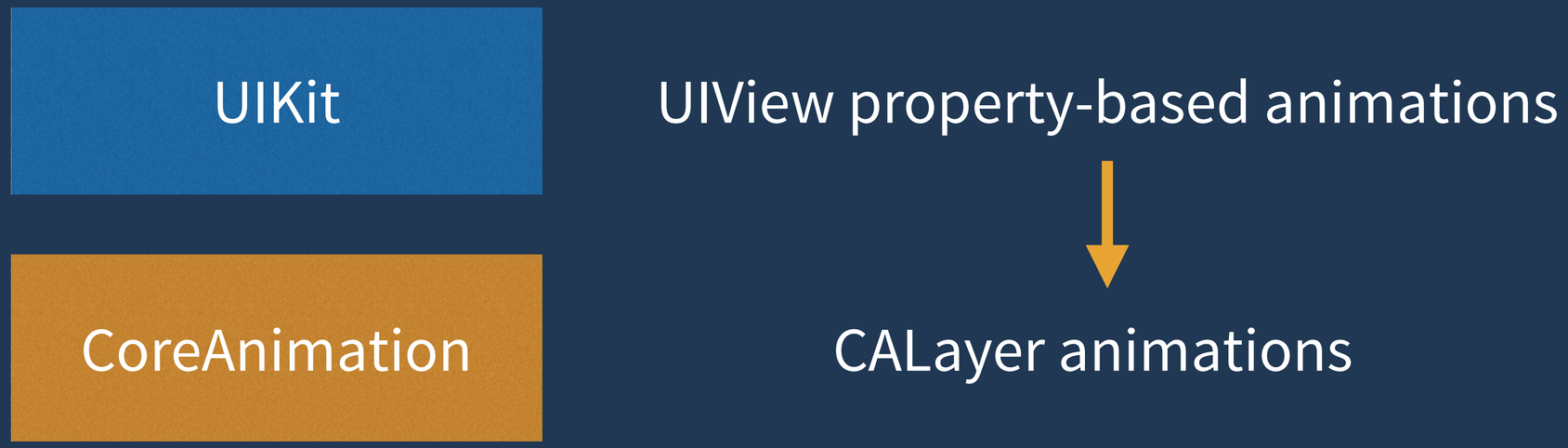
CoreAnimation



CoreAnimation

CoreAnimation


UIKit
CoreAnimation


UIKit
CoreAnimation
UIView property-based animations
CALayer animations

Animatable View Properties

frame

bounds

center

transform

alpha

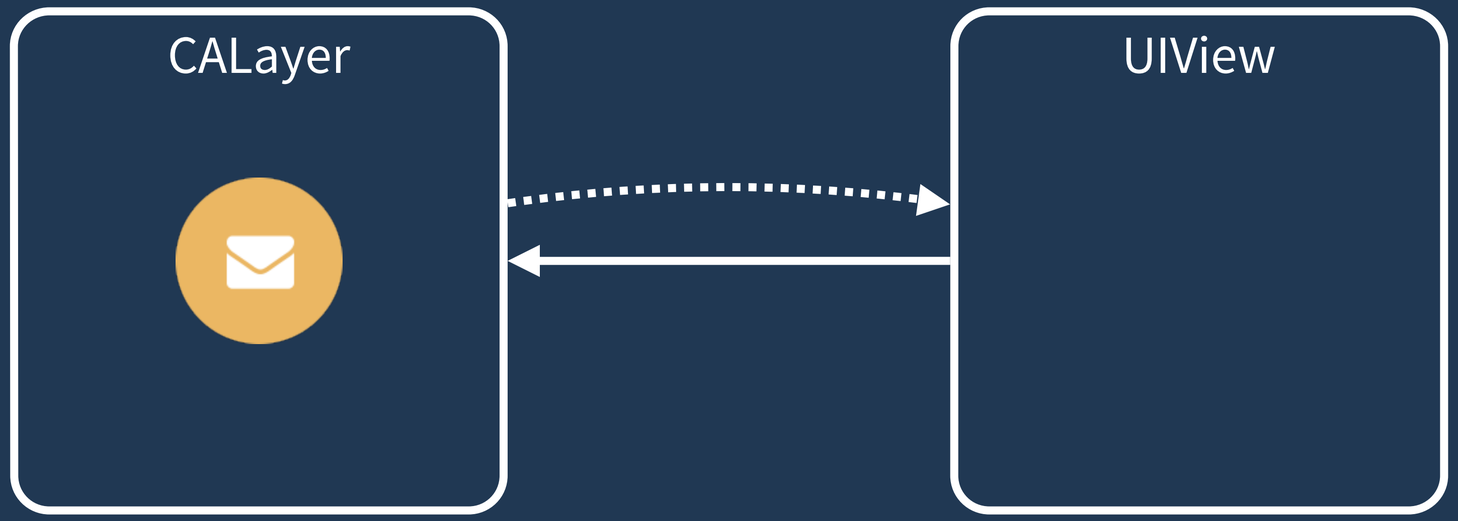
backgroundColor

contentStretch

Property-based view animation

```
self.view.alpha = 1.0;

[UIView animateWithDuration:1.0 animations:^{
    self.view.alpha = 0.0;
}]
```

```
self.view.alpha = 1.0;

[UIView animateWithDuration:1.0 animations:^{
    self.view.alpha = 0.0;
}]
```



Animatable Layer Properties

anchorPoint
backgroundColor
backgroundFilters
borderColor
borderWidth
bounds
compositingFilter
contents
contentsRect
cornerRadius
doubleSided
filters
hidden
mask
masksToBounds
opacity
position
shadowColor
shadowOffset
shadowOpacity
shadowPath
shadowRadius
sublayers
sublayerTransform
transform
zPosition

CABasicAnimation

```
CABasicAnimation *layerAnimation = [CABasicAnimation  animationWithKeyPath:@“position.x”];

layerAnimation.duration = 1.0;
layerAnimation.fromValue = @0.0;
layerAnimation.toValue = @100.0;

[self.view.layer addAnimation:layerAnimation forKey:@“moveRight”]
```

LoadingSpinnerView

1. Start with bolts off screen

1. Start with bolts off screen
2. Animate them together

1. Start with bolts off screen
2. Animate them together
3. Pause briefly

1. Start with bolts off screen
2. Animate them together
3. Pause briefly
4. Perform 1 full rotation
5. Repeat 3- 4 until stopped

1. Start with bolts off screen
2. Animate them together
3. Pause briefly
4. Perform 1 full rotation
5. Repeat 3- 4 until stopped
6. Fade out

Lab 3.2