UIView Hide/Show with animation

Asked 12 years, 10 months ago Modified 1 year, 11 months ago Viewed 222k times

Part of Mobile Development Collective

186

My simple goal is to fade animate hiding and showing functions.

```
Button.hidden = YES;
```

Simple enough. However, is it possible to have it fade out rather than just disappearing? It looks rather unprofessional that way.

MD ios objective-c uiviewanimation

Share
Improve this question
Follow

edited Apr 7, 2017 at 10:29
IPS Brar
369 ● 1 ● 14

asked Feb 2, 2012 at 16:21
Jtaylorapps
5,760 ● 8 ● 41 ● 56

22 Answers

Sorted by: Highest score (default)

310

In iOS 4 and later, there's a way to do this just using the UIView transition method without needing to import QuartzCore. You can just say:

Objective C

```
[UIView transitionWithView:button
                  duration:0.4
                   options:UIViewAnimationOptionTransitionCrossDissolve
                animations:^{
                     button.hidden = YES;
                }
                completion:NULL];
```

Swift

```
UIView.transition(with: button, duration: 0.4,
                  options: .transitionCrossDissolve,
                  animations: {
                 self.button.isHidden = false
              })
```

Previous Solution

Michail's solution will work, but it's not actually the best approach.

The problem with alpha fading is that sometimes the different overlapping view layers look weird as they fade out. There are some other alternatives using Core Animation. First include the QuartzCore framework in your app and add `#import <QuartzCore/QuartzCore.h>` to your header. Now you can do one of the following:

1. set `button.layer.shouldRasterize = YES;` and then use the alpha animation code that Michail provided in his answer. This will prevent the layers from blending weirdly, but has a slight performance penalty, and can make the button look blurry if it's not aligned exactly on a pixel boundary.

Alternatively:

2. Use the following code to animate the fade instead:

   CATransition *animation = [CATransition animation]; animation.type = kCATransitionFade; animation.duration = 0.4; [button.layer addAnimation:animation forKey:nil];

   button.hidden = YES;

The nice thing about this approach is you can crossfade any property of the button even if they aren't animatable (e.g. the text or image of the button), just set up the transition and then set your properties immediately afterwards.

Share

Improve this answer

Follow

edited Jan 23, 2023 at 10:37
Amr
290 ● 1 ● 2 ● 12

answered Feb 2, 2012 at 16:49
Nick Lockwood
41k ● 11 ● 113 ● 105

9 @robmathers , I just test your code , above two code just work when button.hidden = NO, for fade in situation; have no animation effect for fade out when button.hidden = YES; – Jason Jul 9, 2017 at 5:45

Seems to be broken in iOS 12.0 – user3532505 Dec 15, 2018 at 15:15

16 You should use the superview of the thing you are animating as the `transitionWithView` parameter to ensure a successful fade in and out. – allenh Jan 29, 2019 at 19:10

176

UIView animated properties are:

```
- frame
- bounds
- center
- transform
- alpha
- backgroundColor
- contentStretch
```

> Describe in: Animations

`isHidden` is not one of them, so as I see it the best way is:

Swift 4:

```
func setView(view: UIView, hidden: Bool) {
    UIView.transition(with: view, duration: 0.5, options:
.transitionCrossDissolve, animations: {
        view.isHidden = hidden
    })
}
```

Objective C:

```
- (void)setView:(UIView*)view hidden:(BOOL)hidden {
    [UIView transitionWithView:view duration:0.5
options:UIViewAnimationOptionTransitionCrossDissolve animations:^(void){
        [view setHidden:hidden];
    } completion:nil];
}
```

Share

Improve this answer

Follow

edited May 28, 2018 at 5:17

Mahendra
8,884 ● 4 ● 35 ● 59

answered Mar 16, 2015 at 15:37

evya
3,647 ● 1 ● 28 ● 30

10 Actually this is the simple and best answer – Mohamed Irshad Aug 12, 2015 at 13:21

3 While this animates correctly, the UISearchBar I'm trying to display appears in the wrong location until the animation completes and then instantly jumps to the correct position. Any idea? I'm using Storyboards with Interface Builder and Constraints. – Greg Hilston Jun 13, 2017 at 21:52

8 This code doesn't work ... it directly change the state without animating – Mihir Mehta Jun 27, 2017 at 6:49

7 @evya Only work for fade in when hidden = NO ， Not work for fade out, hidden = YES – Jason Jul 9, 2017 at 5:51

Seems to be broken in iOS 12.0 – user3532505 Dec 15, 2018 at 15:15

To fade out:

Objective-C

144

```
[UIView animateWithDuration:0.3 animations:^{
    button.alpha = 0;
} completion: ^(BOOL finished) {//creates a variable (BOOL) called "finished" that is set to *YES* when animation IS completed.
    button.hidden = finished;//if animation is finished ("finished" == *YES*), then hidden = "finished" ... (aka hidden = *YES*)
}];
```

Swift 2

```
UIView.animateWithDuration(0.3, animations: {
    button.alpha = 0
}) { (finished) in
    button.hidden = finished
}
```

Swift 3, 4, 5

```
UIView.animate(withDuration: 0.3, animations: {
    button.alpha = 0
}) { (finished) in
    button.isHidden = finished
}
```

To fade in:

Objective-C

```
button.alpha = 0;
button.hidden = NO;
[UIView animateWithDuration:0.3 animations:^{
    button.alpha = 1;
}];
```

Swift 2

```
button.alpha = 0
button.hidden = false
UIView.animateWithDuration(0.3) {
    button.alpha = 1
}
```

Swift 3, 4, 5

```
button.alpha = 0
button.isHidden = false
UIView.animate(withDuration: 0.3) {
    button.alpha = 1
}
```

Share

Improve this answer

Follow

edited Feb 19, 2020 at 12:59

Karen Hovhannisyan
1,238 ● 2 ● 23 ● 33

answered Feb 2, 2012 at 16:28

Mikhail Grebionkin
3,836 ● 2 ● 19 ● 19

using the fade in/out in conjunction with the hidden state solved my problem – aclima Jun 1, 2016 at 15:58

For some reason, animating to hidden=YES worked fine for me, but animating to hidden=NO did nothing, so this combination of animating the alpha and setting the hidden property was helpful. – arlomedia Feb 10, 2017 at 1:30

I just write a demo , but only hidden = NO , fade in works, strange – Jason Jul 9, 2017 at 5:48

13

Use this solution for a smooth fadeOut and fadeIn effects

```
extension UIView {
    func fadeIn(duration: TimeInterval = 0.5, delay: TimeInterval = 0.0,
completion: @escaping ((Bool) -> Void) = {(finished: Bool) -> Void in }) {
        self.alpha = 0.0

        UIView.animate(withDuration: duration, delay: delay, options:
UIView.AnimationOptions.curveEaseIn, animations: {
            self.isHidden = false
            self.alpha = 1.0
        }, completion: completion)
    }

    func fadeOut(duration: TimeInterval = 0.5, delay: TimeInterval = 0.0,
completion: @escaping (Bool) -> Void = {(finished: Bool) -> Void in }) {
        self.alpha = 1.0

        UIView.animate(withDuration: duration, delay: delay, options:
UIView.AnimationOptions.curveEaseOut, animations: {
            self.isHidden = true
            self.alpha = 0.0
        }, completion: completion)
    }
}
```

usage is as like

```
uielement.fadeIn()
uielement.fadeOut()
```

Thanks

Share Improve this answer Follow

answered Jul 23, 2019 at 10:20

Dhanu K
12.4k ● 6 ● 26 ● 42

1 `fadeOut` works on iOS 13 only if I remove the lines that set `self.isHidden`.
– Mike Taverne Dec 31, 2019 at 22:57

12

I use this little **Swift 3** extension:

```
extension UIView {

  func fadeIn(duration: TimeInterval = 0.5,
              delay: TimeInterval = 0.0,
              completion: @escaping ((Bool) -> Void) = {(finished: Bool) ->
Void in }) {
    UIView.animate(withDuration: duration,
                   delay: delay,
                   options: UIViewAnimationOptions.curveEaseIn,
                   animations: {
      self.alpha = 1.0
    }, completion: completion)
  }

  func fadeOut(duration: TimeInterval = 0.5,
               delay: TimeInterval = 0.0,
               completion: @escaping (Bool) -> Void = {(finished: Bool) -> Void
in }) {
    UIView.animate(withDuration: duration,
                   delay: delay,
                   options: UIViewAnimationOptions.curveEaseIn,
                   animations: {
      self.alpha = 0.0
    }, completion: completion)
  }
}
```

Share
Improve this answer
Follow

edited Feb 28, 2017 at 3:49

answered Feb 27, 2017 at 16:45
Mark Mckelvie
343 ● 4 ● 13

1 This one actually worked. None of the other solutions seemed to actually make it fade in/out. And the extension is super nice. Well done. :) – Iskeraet Nov 6, 2020 at 4:00

10

Swift 3

```
func appearView() {
     self.myView.alpha = 0
     self.myView.isHidden = false

     UIView.animate(withDuration: 0.9, animations: {
         self.myView.alpha = 1
     }, completion: {
         finished in
```

```
            self.myView.isHidden = false
        })
}
```

Share Improve this answer Follow

answered Dec 15, 2016 at 3:06

Scaraux
4,152 ● 6 ● 45 ● 84

9

swift 4.2

with extension :

```
extension UIView {
func hideWithAnimation(hidden: Bool) {
        UIView.transition(with: self, duration: 0.5, options:
.transitionCrossDissolve, animations: {
            self.isHidden = hidden
        })
    }
}
```

simple method:

```
func setView(view: UIView, hidden: Bool) {
    UIView.transition(with: view, duration: 0.5, options:
.transitionCrossDissolve, animations: {
        view.isHidden = hidden
    })
}
```

Share Improve this answer Follow

answered Dec 9, 2018 at 8:54

Mohsen mokhtari
3,032 ● 1 ● 32 ● 41

How can I add delay for this one? – cvdogan Mar 30, 2019 at 20:06

7

the code of @Umair Afzal working fine in swift 5 after some changes

```
 extension UIView {

func fadeIn(duration: TimeInterval = 0.5, delay: TimeInterval = 0.0,
completion: @escaping ((Bool) -> Void) = {(finished: Bool) -> Void in }) {
    self.alpha = 0.0

    UIView.animate(withDuration: duration, delay: delay, options:
UIView.AnimationOptions.curveEaseIn, animations: {
        self.isHidden = false
```

```
            self.alpha = 1.0
        }, completion: completion)
    }

    func fadeOut(duration: TimeInterval = 0.5, delay: TimeInterval = 0.0,
    completion: @escaping (Bool) -> Void = {(finished: Bool) -> Void in }) {
        self.alpha = 1.0

        UIView.animate(withDuration: duration, delay: delay, options:
    UIView.AnimationOptions.curveEaseIn, animations: {
            self.alpha = 0.0
        }) { (completed) in
            self.isHidden = true
            completion(true)
        }
      }
    }
```

for use

```
yourView.fadeOut()
yourView.fadeIn()
```

Share Improve this answer Follow

answered May 10, 2019 at 12:42

Sanjay Mishra

692 ● 8 ● 19

giving some hard effect while fadeout, added some better solution here – Dhanu K Jul 23, 2019 at 10:22

6

I created category for `UIView` for this purpose and implemented a special little bit different concept: `visibility`. The main difference of my solution is that you can call `[view setVisible:NO animated:YES]` and right after that synchronously check `[view visible]` and get correct result. This is pretty simple but extremely useful.

Besides, it is allowed to avoid using "negative boolean logic" (see Code Complete, page 269, Use positive boolean variable names for more information).

Swift

`UIView+Visibility.swift`

```
import UIKit


private let UIViewVisibilityShowAnimationKey =
"UIViewVisibilityShowAnimationKey"
private let UIViewVisibilityHideAnimationKey =
"UIViewVisibilityHideAnimationKey"


private class UIViewAnimationDelegate: NSObject {
    weak var view: UIView?

    dynamic override func animationDidStop(animation: CAAnimation, finished: 
Bool) {
        guard let view = self.view where finished else {
            return
        }

        view.hidden = !view.visible
        view.removeVisibilityAnimations()
    }
}


extension UIView {

    private func removeVisibilityAnimations() {
        self.layer.removeAnimationForKey(UIViewVisibilityShowAnimationKey)
        self.layer.removeAnimationForKey(UIViewVisibilityHideAnimationKey)
    }

    var visible: Bool {
        get {
            return !self.hidden && 
self.layer.animationForKey(UIViewVisibilityHideAnimationKey) == nil
        }

        set {
            let visible = newValue

            guard self.visible != visible else {
                return
            }

            let animated = UIView.areAnimationsEnabled()

            self.removeVisibilityAnimations()

            guard animated else {
                self.hidden = !visible
                return
            }

            self.hidden = false

            let delegate = UIViewAnimationDelegate()
            delegate.view = self

            let animation = CABasicAnimation(keyPath: "opacity")
            animation.fromValue = visible ? 0.0 : 1.0
            animation.toValue = visible ? 1.0 : 0.0
            animation.fillMode = kCAFillModeForwards
            animation.removedOnCompletion = false
            animation.delegate = delegate

            self.layer.addAnimation(animation, forKey: visible ? 
UIViewVisibilityShowAnimationKey : UIViewVisibilityHideAnimationKey)
        }
    }

    func setVisible(visible: Bool, animated: Bool) {
        let wereAnimationsEnabled = UIView.areAnimationsEnabled()

        if wereAnimationsEnabled != animated {
            UIView.setAnimationsEnabled(animated)
            defer { UIView.setAnimationsEnabled(!animated) }
        }

        self.visible = visible
    }

}
```

Objective-C

`UIView+Visibility.h`

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

@interface UIView (Visibility)

- (BOOL)visible;
- (void)setVisible:(BOOL)visible;
- (void)setVisible:(BOOL)visible animated:(BOOL)animated;

@end
```

`UIView+Visibility.m`

```
#import "UIView+Visibility.h"

NSString *const UIViewVisibilityAnimationKeyShow =
@"UIViewVisibilityAnimationKeyShow";
NSString *const UIViewVisibilityAnimationKeyHide =
@"UIViewVisibilityAnimationKeyHide";

@implementation UIView (Visibility)

- (BOOL)visible
{
    if (self.hidden || [self.layer
animationForKey:UIViewVisibilityAnimationKeyHide]) {
        return NO;
    }

    return YES;
}

- (void)setVisible:(BOOL)visible
{
    [self setVisible:visible animated:NO];
}
```

```
- (void)setVisible:(BOOL)visible animated:(BOOL)animated
{
    if (self.visible == visible) {
        return;
    }

    [self.layer removeAnimationForKey:UIViewVisibilityAnimationKeyShow];
    [self.layer removeAnimationForKey:UIViewVisibilityAnimationKeyHide];

    if (!animated) {
        self.alpha = 1.f;
        self.hidden = !visible;
        return;
    }

    self.hidden = NO;

    CGFloat fromAlpha = visible ? 0.f : 1.f;
    CGFloat toAlpha = visible ? 1.f : 0.f;
    NSString *animationKey = visible ? UIViewVisibilityAnimationKeyShow :
UIViewVisibilityAnimationKeyHide;

    CABasicAnimation *animation = [CABasicAnimation
animationWithKeyPath:@"opacity"];
    animation.duration = 0.25;
    animation.fromValue = @(fromAlpha);
    animation.toValue = @(toAlpha);
    animation.delegate = self;
    animation.removedOnCompletion = NO;
    animation.fillMode = kCAFillModeForwards;
    [self.layer addAnimation:animation forKey:animationKey];
}

#pragma mark - CAAnimationDelegate

- (void)animationDidStop:(CAAnimation *)animation finished:(BOOL)finished
{
    if ([[self.layer animationForKey:UIViewVisibilityAnimationKeyHide]
isEqual:animation]) {
        self.hidden = YES;
    }
}

@end
```

Share

Improve this answer

Follow

edited Apr 27, 2016 at 0:09

answered Dec 25, 2014 at 12:14

Valentin Shergin

7,334 ● 3 ● 54 ● 54

@valentin shergin Swift version is coming? – Juan Boero Mar 15, 2016 at 17:35

Sure! I have added Swift version. – Valentin Shergin Apr 27, 2016 at 0:09

Defer block is executed before leaving if scope. I believe it is not how you expect it to behave – Vladimir Borodko Jan 12, 2021 at 21:42

5

Swift 5.0, with generics:

```
func hideViewWithAnimation<T: UIView>(shouldHidden: Bool, objView: T) {
    if shouldHidden == true {
        UIView.animate(withDuration: 0.3, animations: {
            objView.alpha = 0
        }) { (finished) in
            objView.isHidden = shouldHidden
        }
    } else {
        objView.alpha = 0
        objView.isHidden = shouldHidden
        UIView.animate(withDuration: 0.3) {
            objView.alpha = 1
        }
    }
}
```

Use:

```
hideViewWithAnimation(shouldHidden: shouldHidden, objView:
itemCountLabelBGView)
hideViewWithAnimation(shouldHidden: shouldHidden, objView: itemCountLabel)
hideViewWithAnimation(shouldHidden: shouldHidden, objView: itemCountButton)
```

Here `itemCountLabelBGView` is a `UIView`, `itemCountLabel` is a `UILabel` & `itemCountButton` is a `UIButton`, So it will work for every view object whose parent class is `UIView`.

Share Improve this answer Follow

answered Aug 13, 2020 at 19:09

Tulon
4,128 ● 6 ● 38 ● 63

4

Swift 4

```
extension UIView {

func fadeIn(duration: TimeInterval = 0.5, delay: TimeInterval = 0.0,
completion: @escaping ((Bool) -> Void) = {(finished: Bool) -> Void in }) {
    self.alpha = 0.0

    UIView.animate(withDuration: duration, delay: delay, options:
UIViewAnimationOptions.curveEaseIn, animations: {
        self.isHidden = false
        self.alpha = 1.0
    }, completion: completion)
}

func fadeOut(duration: TimeInterval = 0.5, delay: TimeInterval = 0.0,
completion: @escaping (Bool) -> Void = {(finished: Bool) -> Void in }) {
    self.alpha = 1.0
    UIView.animate(withDuration: duration, delay: delay, options:
UIViewAnimationOptions.curveEaseIn, animations: {
        self.alpha = 0.0
    }) { (completed) in
        self.isHidden = true
        completion(true)
    }
}
}
```

And to use use it, simple call these functions like:

```
yourView.fadeOut() // this will hide your view with animation
yourView.fadeIn() /// this will show your view with animation
```

Share
Improve this answer
Follow

edited Jul 19, 2018 at 11:13

answered Jul 19, 2018 at 11:06
Umair Afzal
5,037 ●5 ●26 ●51

You've just copied @MarkMckelvie`s answer – Ashley Mills Jul 19, 2018 at 11:09

There is a difference, he was not hiding the view. And I needed to hide the view as well. So did that and share it. – Umair Afzal Jul 19, 2018 at 11:12

3 Why not just comment on the other answer instead of copying it and passing off as your own? – Ashley Mills Jul 19, 2018 at 11:13

4

`isHidden` is an immediate value and you cannot affect an animation on it, instead of this you can use Alpha for hide your view

```
UIView.transition(with: view, duration: 0.5, options: .transitionCrossDissolve,
animations: {
        view.alpha = 0
    })
```

And for showing:

```
UIView.transition(with: view, duration: 0.5, options: .transitionCrossDissolve,
animations: {
      view.alpha = 1
})
```

Share Improve this answer Follow

answered Sep 30, 2019 at 12:56
mohsen
5,008 ●1 ●36 ●58

1

```
func flipViews(fromView: UIView, toView: UIView) {

    toView.frame.origin.y = 0

    self.view.isUserInteractionEnabled = false

    UIView.transition(from: fromView, to: toView, duration: 0.5, options:
.transitionFlipFromLeft, completion: { finished in

        fromView.frame.origin.y = -900

        self.view.isUserInteractionEnabled = true

    })


}
```

Share Improve this answer Follow

answered Apr 3, 2017 at 20:00

Vimal Saifudin

1,845 ● 1 ● 22 ● 28

1

You can try this.

```
 func showView(objView:UIView){

    objView.alpha = 0.0
    UIView.animate(withDuration: 0.5, animations: {
        objView.alpha = 0.0
    }, completion: { (completeFadein: Bool) -> Void in
        objView.alpha = 1.0
        let transition = CATransition()
        transition.duration = 0.5
        transition.timingFunction = CAMediaTimingFunction(name:
kCAMediaTimingFunctionEaseInEaseOut)
        transition.type = kCATransitionFade
        objView.layer.add(transition, forKey: nil)
    })
}

func HideView(objView:UIView){

    UIView.animate(withDuration: 0.5, animations: {
        objView.alpha = 1.0
    }, completion: { (completeFadein: Bool) -> Void in
        objView.alpha = 0.0
        let transition = CATransition()
        transition.duration = 0.5
        transition.timingFunction = CAMediaTimingFunction(name:
kCAMediaTimingFunctionEaseInEaseOut)
        transition.type = kCATransitionFade
        objView.layer.add(transition, forKey: nil)
    })
}
```

And pass your view name

```
        showView(objView: self.viewSaveCard)
        HideView(objView: self.viewSaveCard)
```

Share Improve this answer Follow

answered Apr 17, 2018 at 9:50

kalpesh
1,287 ● 1 ● 17 ● 30

1

If your view is set to hidden by default or you change the Hidden state which I think you should in many cases, then none of the approaches in this page will give you both FadeIn/FadeOut animation, it will only animate one of these states, the reason is you are setting the Hidden state to false before calling **UIView.animate** method which will cause a sudden visibility and if you only animate the alpha then the object space is still there but it's not visible which will result to some UI issues.

So the best approach is to check first if the view is hidden then set the alpha to 0.0, like this when you set the Hidden state to false you won't see a sudden visibility.

```
func hideViewWithFade(_ view: UIView) {
    if view.isHidden {
        view.alpha = 0.0
    }

    view.isHidden = false

    UIView.animate(withDuration: 0.3, delay: 0.0, options:
.transitionCrossDissolve, animations: {
        view.alpha = view.alpha == 1.0 ? 0.0 : 1.0
    }, completion: { _ in
        view.isHidden = !Bool(truncating: view.alpha as NSNumber)
    })
}
```

Share Improve this answer Follow

answered Oct 23, 2019 at 21:12

Shahriyar
678 ● 9 ● 21

This solves the issue others have asked about where fadeins weren't working. Clever approach. – BooTooMany Nov 11, 2019 at 22:23

1

UIView.transition(with:) function is nice and neat.

Many have posted it but none has noticed there lies a fault will show up only when you run it.

You can transition hidden property to true perfectly, whereas when you attempt to transition it to false, the view will simple disappear suddenly without any animation.

That's because this api only works **within** a view, which means when you transition a view to show, in fact itself shows immediately, only its content animated out gradually.

When you try to hide this view, itself hide right away, makes the animation to its content meaningless.

To solve this, when hiding a view, the transition target should be its parent view instead of the view you want to hide.

```
func transitionView(_ view: UIView?, show: Bool, completion: BoolFunc? = nil) {
    guard let view = view, view.isHidden == show, let parent = view.superview
else { return }

    let target: UIView = show ? view : parent
    UIView.transition(with: target, duration: 0.4, options:
[.transitionCrossDissolve], animations: {
        view.isHidden = !show
    }, completion: completion)
}
```

Share
Improve this answer
Follow

edited Jul 17, 2020 at 6:55

answered Mar 27, 2020 at 7:01
LiLi Kazine
213 ● 3 ● 10

1

```
UIView.transition(with: title3Label, duration: 0.4,
                          options: .transitionCrossDissolve,
                          animations: {
                            self.title3Label.isHidden =
!self.title3Label.isHidden
                      })
```

Applying transition on View with some delay gives hide and show effect

Share
Improve this answer
Follow

edited Oct 23, 2020 at 17:17

answered Oct 22, 2020 at 22:34
Wasim
1,150 ● 15 ● 16

4 Please don't post only code as answer, but also provide an explanation what your code does and how it solves the problem of the question. Answers with an explanation are usually more helpful and of better quality, and are more likely to attract upvotes. – Mark Rotteveel Oct 23, 2020 at 7:47

You can do it VERY easily using Animatics library:

0

```
//To hide button:
AlphaAnimator(0) ~> button

//to show button
AlphaAnimator(1) ~> button
```

Share Improve this answer Follow

answered Oct 25, 2015 at 17:53

Nikita Arkhipov
1,248 ● 4 ● 11 ● 28

0

My solution for **Swift 3**. So, I created the function, that hide/unhide view in the right order(when hiding - set alpha to 0 and then isHidden to true; unhiding - first reveal the view and then set it's alpha to 1):

```
func hide(_ hide: Bool) {
    let animations = hide ? { self.alpha = 0 } :
                            { self.isHidden = false }
    let completion: (Bool) -> Void = hide ? { _ in self.isHidden = true } :
                                            { _ in UIView.animate(withDuration:
duration, animations: { self.alpha = 1 }) }
    UIView.animate(withDuration: duration, animations: animations, completion:
completion)
}
```

Share Improve this answer Follow

answered Feb 20, 2017 at 17:52

Nazariy Vlizlo
967 ● 9 ● 16

Why in `completion` block there's another animation when `hide` is false? – Giorgio Jun 19, 2018 at 10:58

0

Swift 4 Transition

```
    UIView.transition(with: view, duration: 3, options: .transitionCurlDown,
                      animations: {
                        // Animations
                        view.isHidden = hidden
    },
                      completion: { finished in
                        // Compeleted
    })
```

If you use the approach for older swift versions you'll get an error :

```
Cannot convert value of type '(_) -> ()' to expected argument type '(() ->
Void)?'
```

Useful reference.

Share Improve this answer Follow

answered Sep 24, 2017 at 5:21
nanospeck
3,538 ● 3 ● 39 ● 45

does this work with autolayout? similar code isn't animating. the `isHidden` value is rendered instantly (i.e., instantly hiding/showing view). – Crashalot Nov 1, 2017 at 9:04

0

> This code give an animation like pushing viewController in uinavigation controller...

```
CATransition *animation = [CATransition animation];
 animation.type = kCATransitionPush;
 animation.subtype = kCATransitionFromRight;
 animation.duration = 0.3;
 [_viewAccountName.layer addAnimation:animation forKey:nil];

 _viewAccountName.hidden = true;
```

> Used this for pop animation...

```
 CATransition *animation = [CATransition animation];
 animation.type = kCATransitionPush;
 animation.subtype = kCATransitionFromLeft;
 animation.duration = 0.3;
 [_viewAccountName.layer addAnimation:animation forKey:nil];

 _viewAccountName.hidden = false;
```

Share Improve this answer Follow

answered Oct 3, 2018 at 11:17
Prasanna
955 ● 10 ● 24

0

Tried some of the exited answers, some only work for one situation, some of them need to add two functions.

Option 1

Nothing to do with `view.isHidden`.

```
extension UIView {
    func animate(fadeIn: Bool, withDuration: TimeInterval = 1.0) {
        UIView.animate(withDuration: withDuration, delay: 0.0, options:
.curveEaseInOut, animations: {
            self.alpha = fadeIn ? 1.0 : 0.0
        })
    }
}
```

Then pass `isFadeIn` (`true` or `false`)

```
view.animate(fadeIn: isFadeIn)
```

Option 2

Don't pass any parameter. It fades in or out according to `isUserInteractionEnabled`. This also suits the situation animate back and forth very well.

```
func animateFadeInOut(withDuration: TimeInterval = 1.0) {
    self.isUserInteractionEnabled = !self.isUserInteractionEnabled
    UIView.animate(withDuration: withDuration, delay: 0.0, options:
.curveEaseInOut, animations: {
        self.alpha = self.isUserInteractionEnabled ? 1.0 : 0.0
    })
}
```

Then you call

```
yourView.animateFadeInOut()
```

> Why `self.isUserInteractionEnabled` ?
>
> Tried to replace `self.isUserInteractionEnabled` by `self.isHidden`, no luck at all.

That's it. Cost me sometime, hope it helps someone.

Share

Improve this answer

Follow

edited Mar 20, 2019 at 6:23

answered Mar 19, 2019 at 10:03

William Hu

16.1k ● 12 ● 108 ● 139