SweetTutos
iOS and Mobile Tutorials

Tutorials Forums Submit Your App

SweetTutos
iOS and Mobile Tutorials
Keep visiting sweettutos.com for new coming tutorials, and don't forget to subscribe to be notified by email :)
Let's Start

How to make a horizontal paging UIScrollView with Auto Layout in Storyboards [Swift 3]

April 13, 2015 iOS 10 72 Comments

82 SHARES 7 60 4 7 4

Update August 2016: Fully updated for **Xcode 8** and **Swift 3**

Since Auto Layout came to life, the task of making adaptive user interfaces that support all screen sizes, has become a piece of cake. Although that's a bit of a challenge in some of the situations, this technology has helped get things done easier and faster than before with more enhanced and optimised UI.

In this tutorial, you will learn how to make a cool starting slideshow for your app, something you would show to the user the first time he launches the app, so that you guide him over the main features of your app. These kind of slideshows are famous and are implemented in many apps with different fancy layouts and animations.

Without further ado, let's slide in 🙂

Open up Xcode, select "File\New\Project" from the menu, choose the "Single View Application" template and make sure the default language is *Swift*.

The slideshow you will implement in this tutorial consists of four slides with fixed logo but moving background images, you will also put some text in there and change it dynamically while you scroll. Also, in the last slide, a button will get shown with a nice fade-in animation to allow the user to exit the slideshow and explore the app effectively.

Select 'Main.storyboard' from the 'Project navigator' view, then choose one of the device sizes from the "View as" panel. Let's choose the iPhone 6s model to work with together.

To search type and hit enter

Stay in the loop, subscribe to sweettutos.com

Enter your email address to subscribe to sweettutos.com and receive notifications of new posts by email. Get the latest posts + upcoming discount coupons on our products delivered to your mailbox.

Email*

Subscribe

Top Tutorials

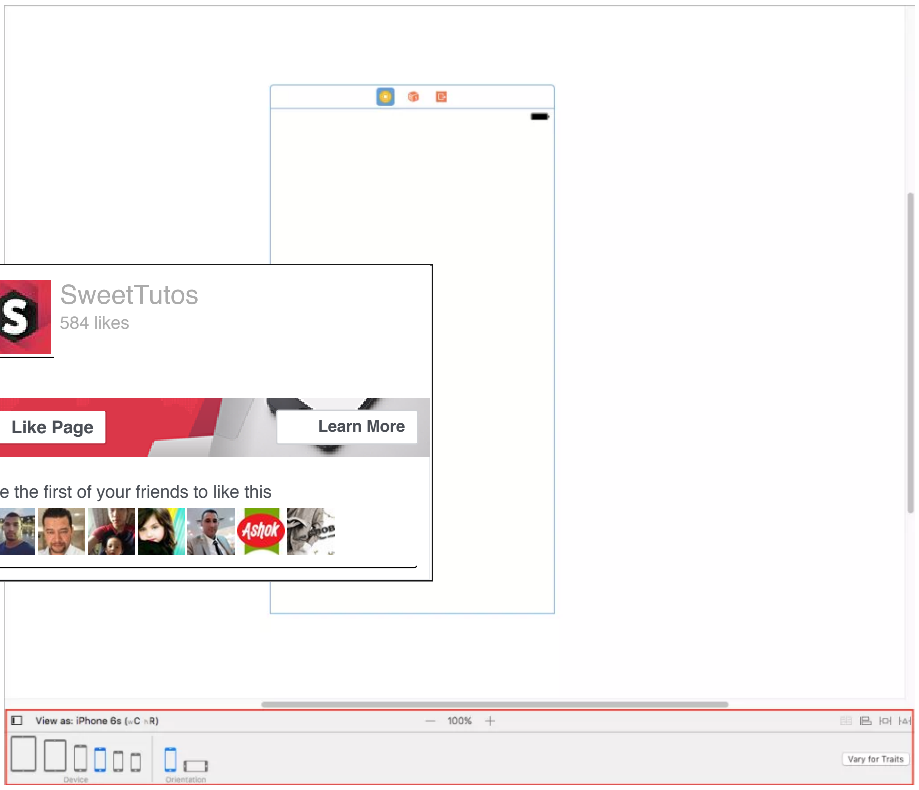
How to make a horizontal paging UIScrollView with Auto Layout in Storyboards [Swift 3]

[Swift] How to Asynchronously Download and Cache Images without Relying on Third-Party Libraries

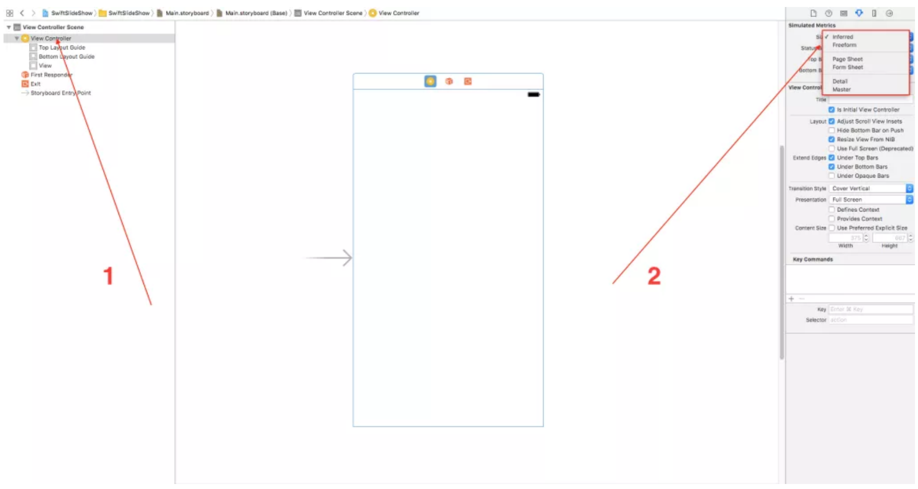
How to Programmatically Save and Load UIImage Files in The Document Directory with Swift

How To Completely Customise Your Map Annotations Callout Views

[Swift MapKit Tutorial Series] How to search a place, address or POI on the map



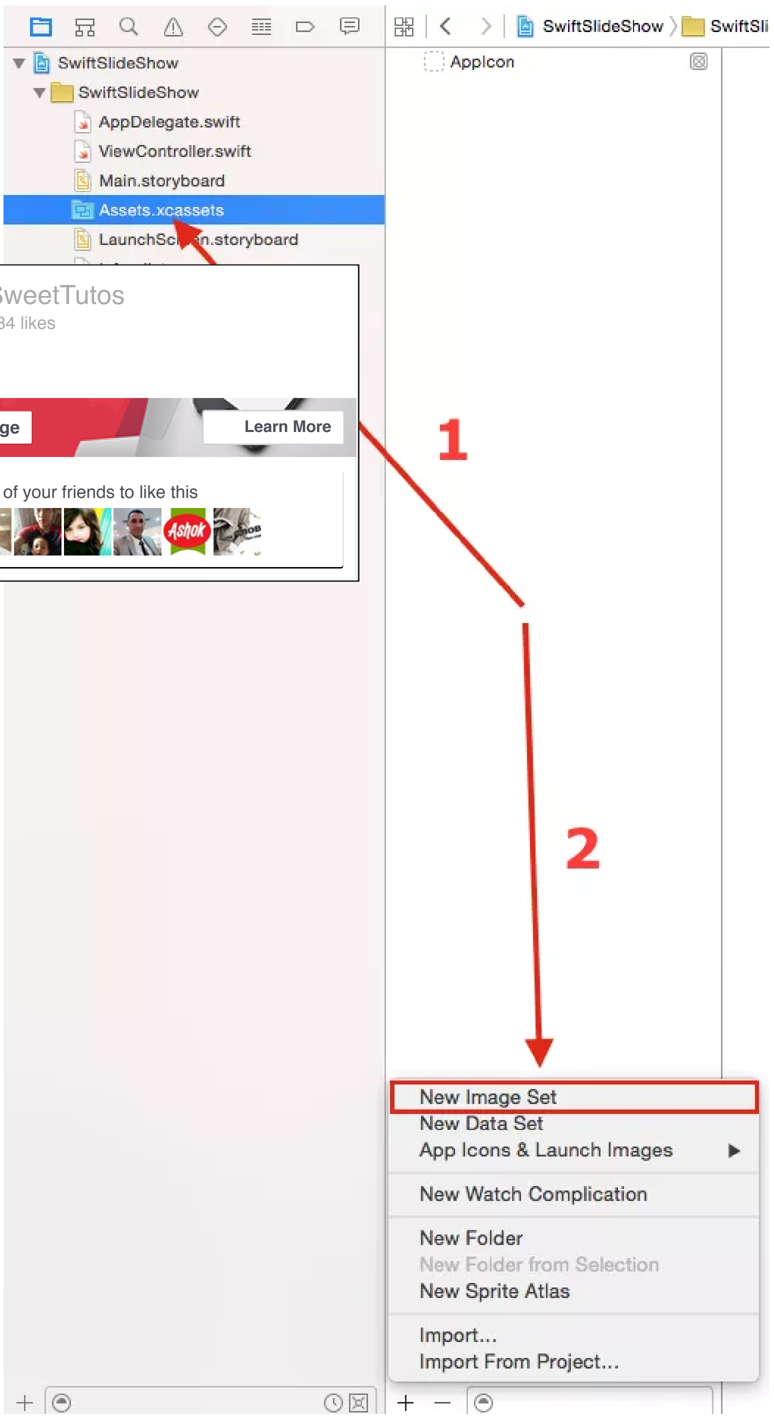
Note: After you choose the device model, Xcode inferred a size of 375*667 (4.7″inch) for the screen, if you want to setup your UI elements in a different dimension view, you can change the size from the 'Attributes inspector' view, like shown below.



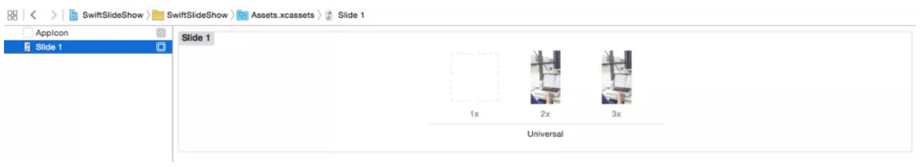
Let's keep the 4.7″ inch screen for the moment so we can follow each other along the way 🙂

Time to import the slides images and icons you will use in the project, download the zip file here.

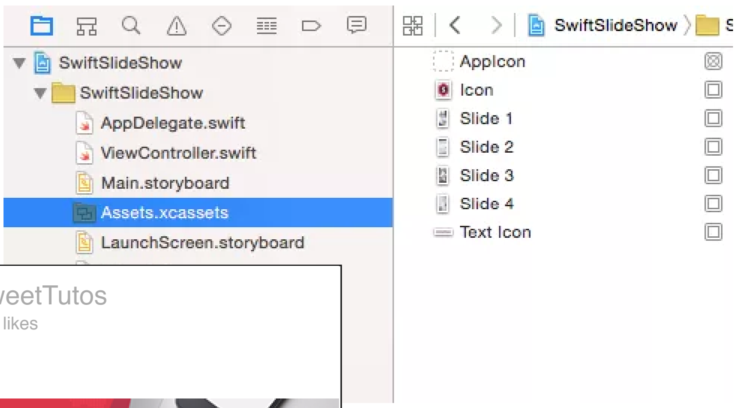
From the 'Project navigator' view, select Assets.xcassets, then click the '+' icon at the bottom and select 'New Image Set' from the list.

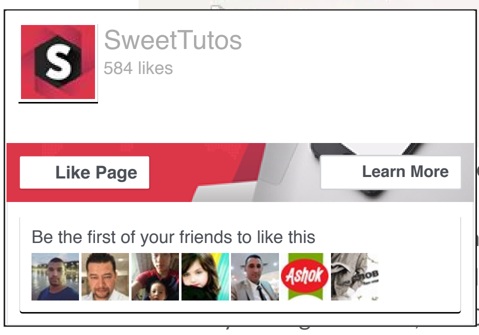


Name it 'Slide 1', you may notice the new asset is provided with 3 resolutions placeholders (1x, 2x and 3x), here you will take care of the Retina and Retina HD (2x and 3x respectively). So drag the two images named slide1@2x.png and slide1@3x.png from the folder you dowloaded to the 2x and 3x placeholders in Xcode.



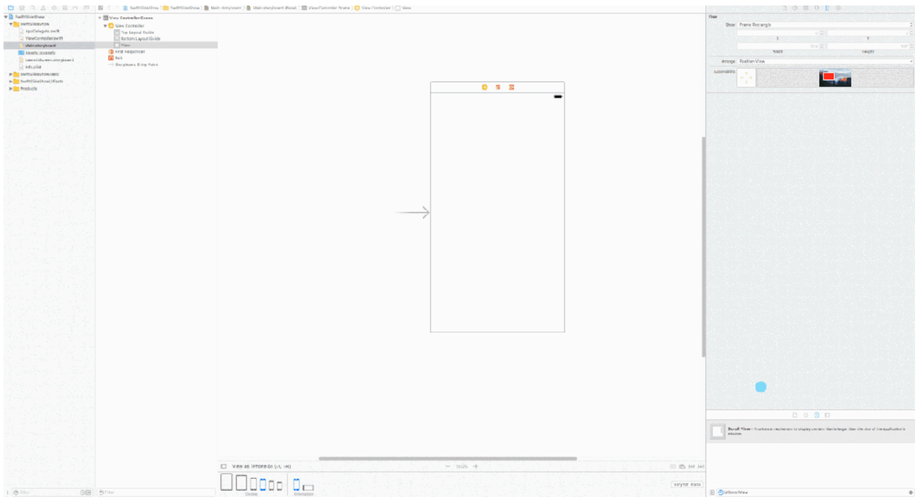
Repeat the same steps to make new set of images for the rest of the downloaded assets, and drag their images to the relevant placeholders, call the images set 'Slide 2', 'Slide 3', 'Slide 4', 'Icon' and 'Text Icon'.





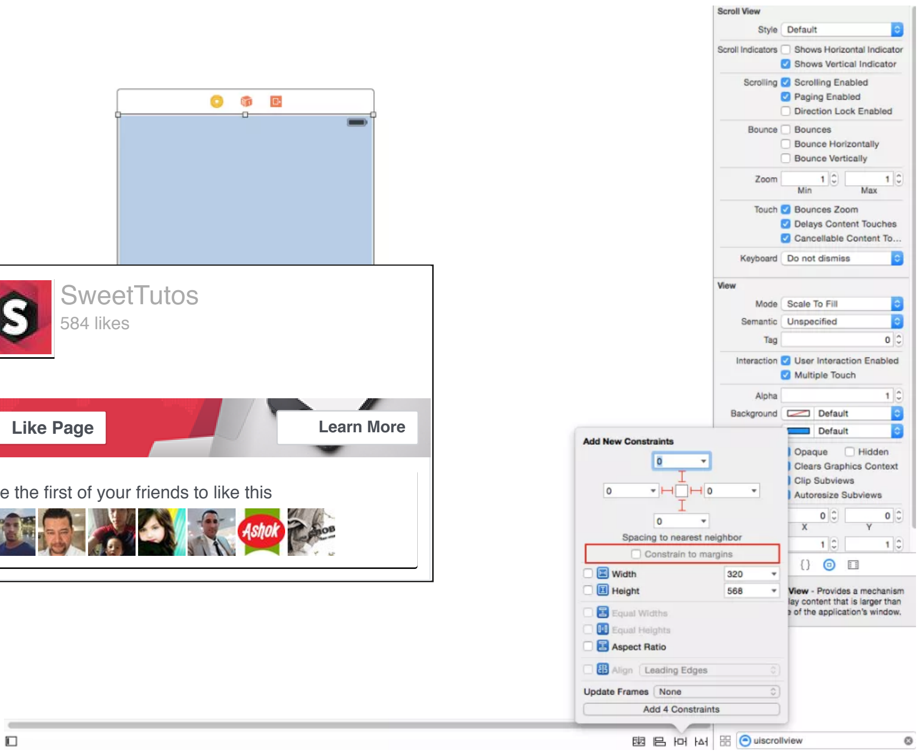
e 8 after you added all assets

d enable paging on it. From the 'Object library', make sure it's filling the whole view area, you), **Width** to 375 and **Height** to 667 from the 'Size inspector'. Switch to the 'Attributes inspector' view, ensure the 'Paging Enabled' property is checked, the 'Bounces' and 'Shows Horizontal Indicator' properties are unchecked.



Let's add some basic constraints so that the scroll view will fill the whole screen on different screen sizes. Select the Scroll View object from the "Document Outline" view, then select the 'Pin' menu and activate the leading, trailing, top and bottom constraints for the selected view.

Note: Make sure the checkbox 'Constrain to margins' shown in the Pin menu is UNchecked.



Scroll View
Style: Default
Scroll Indicators: ☐ Shows Horizontal Indicator, ✓ Shows Vertical Indicator
Scrolling: ✓ Scrolling Enabled, ✓ Paging Enabled, ☐ Direction Lock Enabled
Bounce: ☐ Bounces, ☐ Bounce Horizontally, ☐ Bounce Vertically
Zoom: Min 1, Max 1
Touch: ✓ Bounces Zoom, ✓ Delays Content Touches, ✓ Cancellable Content To...
Keyboard: Do not dismiss

View
Mode: Scale To Fill
Semantic: Unspecified
Tag: 0
Interaction: ✓ User Interaction Enabled, ✓ Multiple Touch
Alpha: 1
Background: Default
Default
Opaque, ☐ Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
X 0, Y 0
1, 1

View - Provides a mechanism ...lay content that is larger than ...of the application's window.

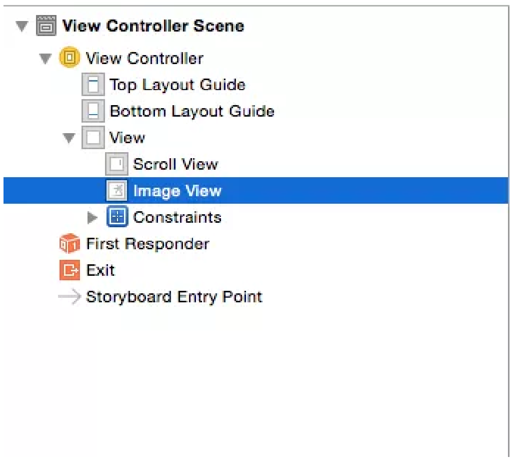
Add New Constraints
Top 0, Leading 0, Trailing 0, Bottom 0
Spacing to nearest neighbor
☐ Constrain to margins
☐ Width 320
☐ Height 568
☐ Equal Widths
☐ Equal Heights
☐ Aspect Ratio
☐ Align Leading Edges
Update Frames: None
Add 4 Constraints

uiscrollview

SweetTutos
584 likes
Like Page
Learn More
Be the first of your friends to like this

Activate leading, trailing, top and bottom constraints

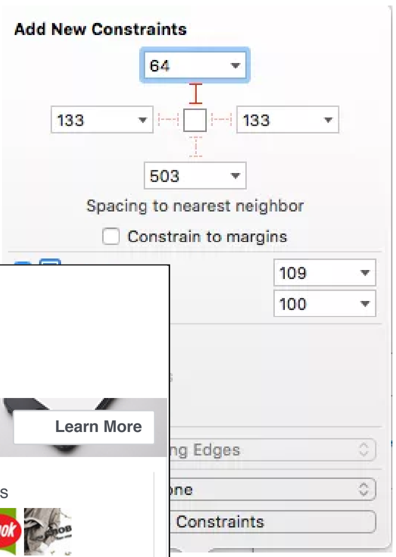
Drag a UIImageView from the 'Object library' to the view, make sure it's a subview of the root view and not the UIScrollView because it will not be dependant on the scrollable content and need to be fixed while the other content will scroll horizontally 🙂

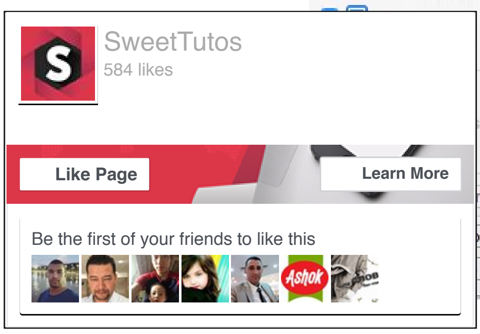


View Controller Scene
- View Controller
  - Top Layout Guide
  - Bottom Layout Guide
  - View
    - Scroll View
    - Image View
    - Constraints
  - First Responder
  - Exit
  - Storyboard Entry Point

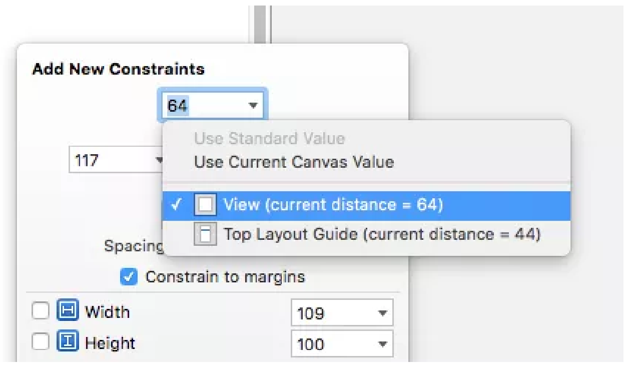
Ensure the image view is selected, switch to the 'Attributes inspector' view and set the 'Image' property to 'Icon' and 'Mode' to 'Aspect Fit', then switch to the 'Size inspector' view and set **X**=133, **Y**=64, **Width**=109 and **Height**=100.

To finish with the icon, select the 'Align' menu and check the 'Horizontally in Container' constraint then click the 'Add constraint' button at the bottom of the menu to apply the constraint. Also select the 'Pin' menu and activate the top spacing, width, height and the aspect ratio constraints, then click the 'Add Constraint' button to apply the changes.





Note: You may need to make sure the Top spacing is set against the top of the view and not the 'Top Layout Guide'. To do so, select the small arrow at the right of the Top spacing value and select 'View', like shown below:

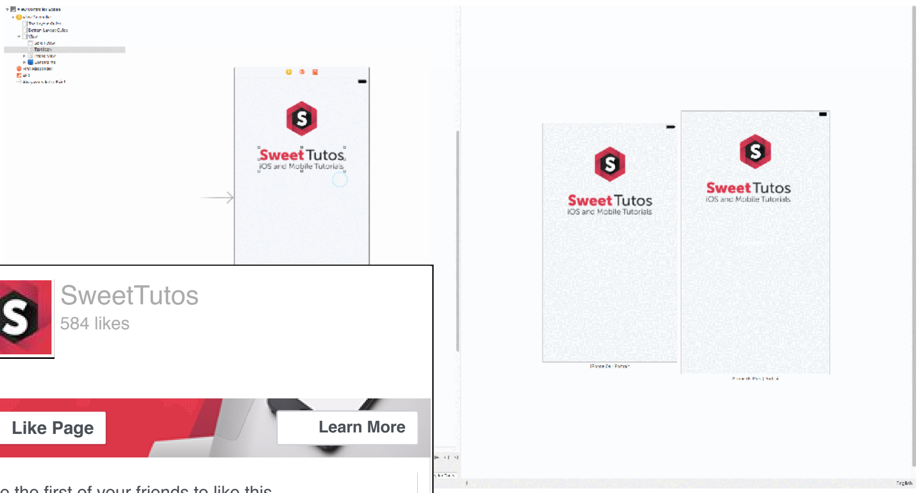


Go on and add another UIImageView to hold the text icon below the main icon. From the 'Object library', drag a 'UIImageView' object to the view and make sure it's a subview of the root view and not the UIScrollView. Set its image to 'Text Icon' and its mode to 'Aspect Fit' from the 'Attributes inspector' view. Next, change its position and frame to **X**=68, **Y**=194, **Width**=239 and **Height**=66 from the 'Size inspector' view.

Finally, Set the following constraints to the text icon:

1/ Activate the 'Horizontally in Container' constraint from the 'Align' menu.
2/ Activate the 'Top' spacing, 'Width' and 'Height' constraints from the 'Pin' menu.
3/ Maintain a Control click and drag a line from the text icon to the scroll view and select the 'Aspect Ratio' constraint from the list.

Here is a quick animation demonstrating the steps above:

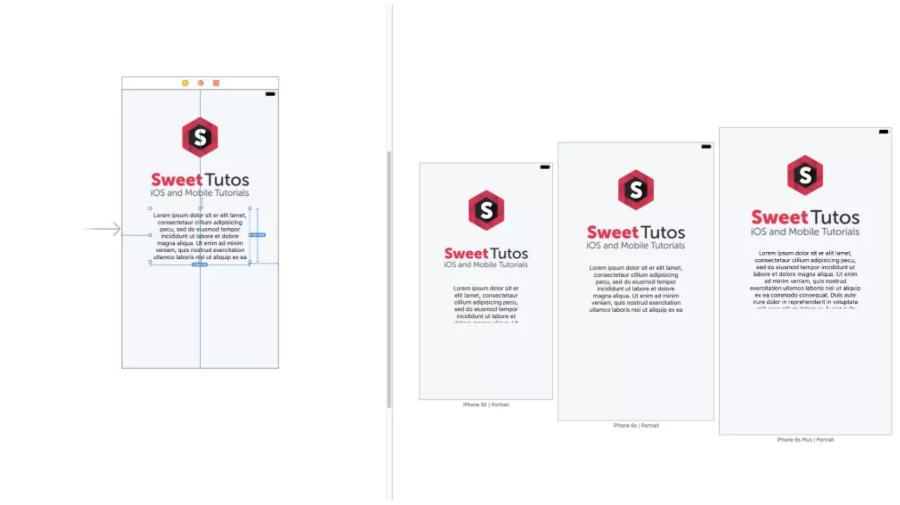


er full screen view

So far so good, let's add a text view to hold some text for the slideshow, the text view will change its text as the scroll view is moving. Drag a UITextView object to the view and ensure it's a subview of the root view as you did for the two previous image views. Next, retrace the following steps for a quick setup of the text view:

1/ From the 'Size inspector' view, set the **X** to 68, **Y** to 284, **Width** to 239 and **Height** to 128.
2/ From the 'Attributes inspector' view, set the 'Alignment' to the center and the 'Background' color to 'Clear Color'. Also, UNcheck the 'User Interaction Enabled' property.
3/ Activate the 'Horizontally in Container' constraint from the 'Align' menu.
4/ From the 'Pin' menu, activate the 'Top', 'Leading', 'Trailing' and 'Aspect Ratio' constraints, and make sure that the "Constrain to margins" checkbox is UNchecked.

Here is how the text view should look like on different screen sizes:



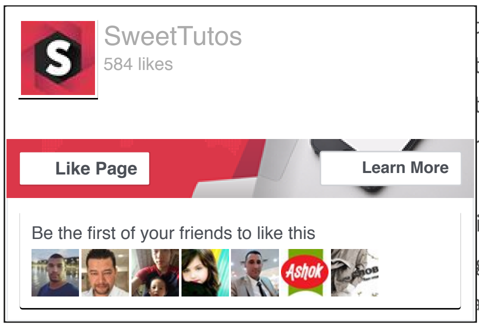
You are almost done. Quickly add a UIPageControl from the 'Object library' to the view (again, it should be a subview of the main view and not the scroll view).

Let's customise the page control color, reposition it and set some constraints. Retrace the following steps:

1/ Select 'Size inspector', set **X** to 154, **Y** to 421 and **Width** to 67.
2/ Select 'Attributes inspector' and set the number of pages to 4 and Hex Tint Color to #de354b.
3/ Select the 'Align' menu and activate the 'Horizontally in Container' constraint.
4/ Select the 'Pin' menu and activate the Top spacing and Width constraints.

Good job, now let's finish up and put a button in the bottom that will remain hidden till the last slide is reached. Drag a UIButton object from the 'Object library' to the view, as usual, it should be a subview of the main view and not the scroll view since you need this button to remain fixed while scrolling.

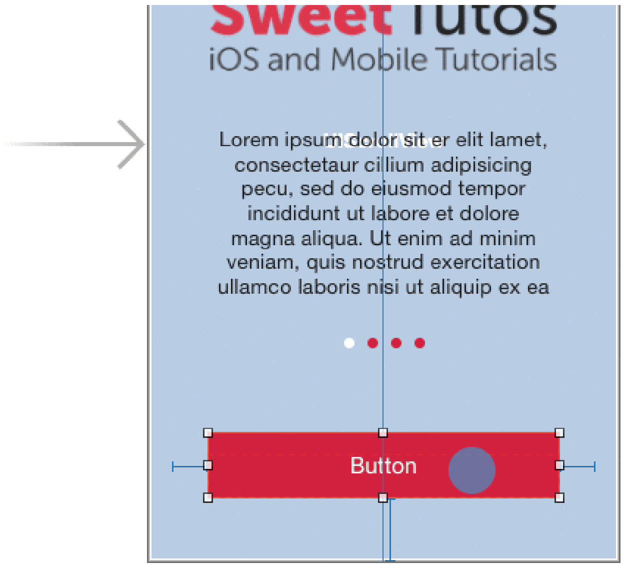
Follow the steps below to set up the button:



o 538, **Width** to 239 and **Height** to 45.
the text to say 'Let's Start', set the color of the
the button to the Hex code #de354b. Also, set
tant for the fade in animation you will

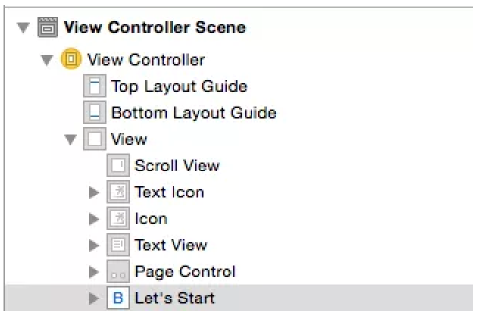
izontally in Container' constraint.
g, trailing and bottom constraints.
rt of the button to the top part of the same
button, select 'Aspect Ratio' from the list.

The following animation illustrates step 5:



Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea

The view hierarchy of the final screen will look like this:



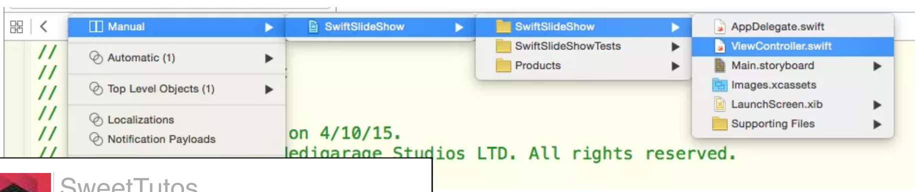
- View Controller Scene
  - View Controller
    - Top Layout Guide
    - Bottom Layout Guide
    - View
      - Scroll View
      - Text Icon
      - Icon
      - Text View
      - Page Control
      - Let's Start

Note: The only required order in the view hierarchy is that the scrollview should be placed on top of all other views, like shown above.

You are done with the UI setup, everything is on place, now we need to hook them up to the code to make them alive 🙂

To do so, switch to the 'Assistant editor' view and follow the steps below:

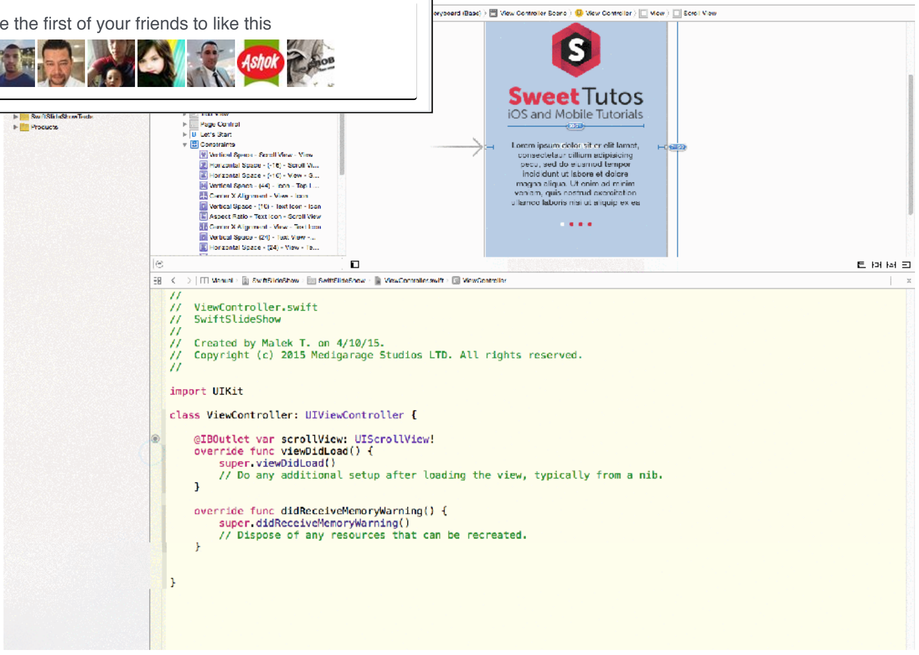
1/ Make sure the ViewController.swift file is opened in the split editor along with the storyboard, if it's not already, you can open it from the top menu in the editor like below.



```
//
//
//
//
//              on 4/10/15.
//          edigarage Studios LTD. All rights reserved.
```

from the scroll view in the storyboard to the

e the outlet 'scrollView' and click 'Connect'

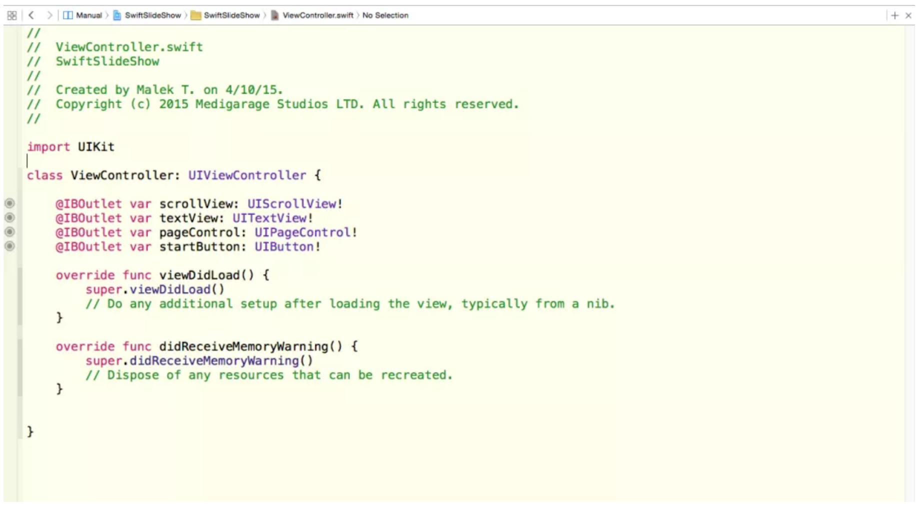


```
//
//  ViewController.swift
//  SwiftSlideShow
//
//  Created by Malek T. on 4/10/15.
//  Copyright (c) 2015 Medigarage Studios LTD. All rights reserved.
//

import UIKit

class ViewController: UIViewController {

    @IBOutlet var scrollView: UIScrollView!
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

}
```

Click for better view resolution

3/ Repeat the step 2 on the text view, the page control and the button. Name them 'textView', 'pageControl' and 'startButton' respectively.

Here is what the ViewController.swift file should look like so far:


Manual > SwiftSlideShow > SwiftSlideShow > ViewController.swift > No Selection

```
//
//  ViewController.swift
//  SwiftSlideShow
//
//  Created by Malek T. on 4/10/15.
//  Copyright (c) 2015 Medigarage Studios LTD. All rights reserved.
//

import UIKit

class ViewController: UIViewController {

    @IBOutlet var scrollView: UIScrollView!
    @IBOutlet var textView: UITextView!
    @IBOutlet var pageControl: UIPageControl!
    @IBOutlet var startButton: UIButton!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }


}
```

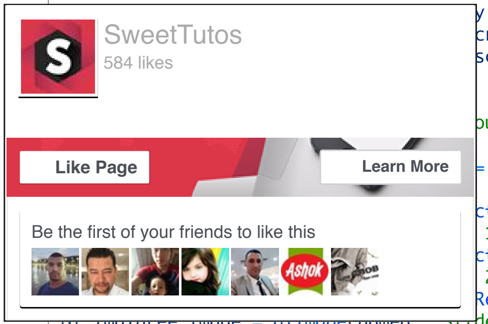
What you did is very important because you will need to interact with all connected outlets from the code.

Now time to write some code to finish up. Switch back to the 'Standard editor' with the ViewController.swift file opened. First you need to make the class adopt the

UIScrollViewDelegate protocol, change the class definition to the following:

```
class ViewController: UIViewController, UIScrollViewDelegate {
```

Cool, now locate the viewDidLoad function and place the following code inside (right after the super.viewDidLoad call):



```
//1
                                           y:0, width:self.view.frame.width, height:self.
                                           crollView.frame.width
                                           scrollView.frame.height


                                           our blog of choice for Mobile tutorials"

                                           = 4.0

                                           ct(x:0, y:0,width:scrollViewWidth, height:scro
                                            1")
                                           ct(x:scrollViewWidth, y:0,width:scrollViewWidt
                                            2")
                                           Rect(x:scrollViewWidth*2, y:0,width:scrollView
imgThree.image = UIImage(named: "Slide 3")
let imgFour = UIImageView(frame: CGRect(x:scrollViewWidth*3, y:0,width:scrollViewW
imgFour.image = UIImage(named: "Slide 4")

self.scrollView.addSubview(imgOne)
self.scrollView.addSubview(imgTwo)
self.scrollView.addSubview(imgThree)
self.scrollView.addSubview(imgFour)
//4
self.scrollView.contentSize = CGSize(width:self.scrollView.frame.width * 4, height
self.scrollView.delegate = self
self.pageControl.currentPage = 0
```

Let's explain some parts of your code:

//1 You set the frame of the scroll view to be equal to the frame of the container view, this is very important since you cannot ensure this to be equal to all screen sizes right from Interface builder, so you need to programmatically retrieve the root view frame and assign it to the scroll view to guarantee it will fill all the screen.

//2 That should be self explanatory, you just loaded a text for the first slide and set the text color of the text view. Also, you changed the corner radius of the button to look nice 🙂

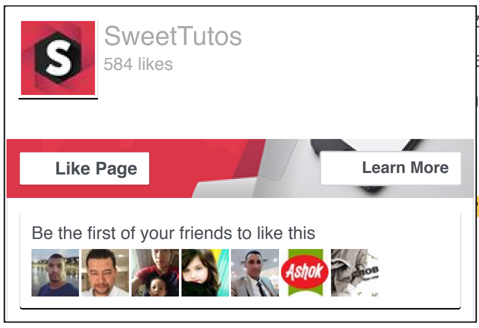
//3 Here you added some background images for the four slides, set their frames accordingly to appear the one next to the other and embed them to the scroll view. Now with the paging enabled, the slides will be bouncing smoothly.

//4 Finally, and most importantly, you changed the content size of the scroll view to reveal the new size of the slides being added horizontally.

Now you will finish up your work by implementing a delegate function related to the scroll view behavior, you need to be notified each time the scroll view has finished scrolling and which is the current page being shown at that moment. This is important in order to: change the current page indicator dot, change the text in the text view and also show the button if it's the last slide being shown.

Place the following code somewhere in your ViewController.swift file before the closing bracket (I put comments to explain the steps):

```
func scrollViewDidEndDecelerating(_ scrollView: UIScrollView){
// Test the offset and calculate the current page after scrolling ends
let pageWidth:CGFloat = scrollView.frame.width
let currentPage:CGFloat = floor((scrollView.contentOffset.x-pageWidth/2)/pageWidth
// Change the indicator
self.pageControl.currentPage = Int(currentPage);
// Change the text accordingly
if Int(currentPage) == 0{
      textView.text = "Sweettutos.com is your blog of choice for Mobile tutorials"
}else if Int(currentPage) == 1{
        textView.text = "I write mobile tutorials mainly targeting iOS"
}else if Int(currentPage) == 2{
        textView.text = "And sometimes I write games tutorials about Unity"
}else{
        textView.text = "Keep visiting sweettutos.com for new coming tutorials, and
// Show the "Let's Start" button in the last slide (with a fade in animation)
        self.startButton.alpha = 1.0
       })
  }
}
```



le, your code above will be running each time
eck for the current page to change the dots
sweet fade in animation for the button on the

Eunice Obugyei, one of our sweet readers, asks me how to make this scrolling app slides automatically without the need to manually scroll it left and right. So here I will explain you how to do so easily 🙂

First, you need to schedule a timer in order to fire a function call every *n* seconds. The called function will do the sliding movement automatically for you.

Locate the *viewDidLoad* method and place the following NSTimer set before the closing bracket of the method:

```
Timer.scheduledTimer(timeInterval: 2, target: self, selector: #selector(moveToNextP
```

As you can see, this timer will call the *moveToNextPage* function every 2 seconds. The repeats argument is very important and should be set to true in order for the *moveToNextPage* function to run continuously every 2 seconds. Otherwise the call to the function will occur only one time.

Next, you need to implement the *moveToNextPage* function, where the auto sliding will be implemented. Place the following code before the closing bracket of the class:

```
func moveToNextPage (){

let pageWidth:CGFloat = self.scrollView.frame.width
let maxWidth:CGFloat = pageWidth * 4
let contentOffset:CGFloat = self.scrollView.contentOffset.x

var slideToX = contentOffset + pageWidth

if  contentOffset + pageWidth == maxWidth
{
      slideToX = 0
}
self.scrollView.scrollRectToVisible(CGRect(x:slideToX, y:0, width:pageWidth, heigh
}
```

The code above will calculate the contentOffset of the scrollview and set its new X position accordingly. If the contentOffset reached the last page, then the X position will be reset to 0.

Last thing before you run the new auto sliding scroll view, you need a way to tell the scroll view delegate about any scrolling changes so that it will update the page indicator and the text accordingly. One way to do so is to rename the delegate method *scrollViewDidEndDecelerating* you already implemented to *scrollViewDidEndScrollingAnimation*. The reason why is that *scrollViewDidEndDecelerating* is not called when we scroll programmatically using *scrollRectToVisible*, unlike the *scrollViewDidEndScrollingAnimation* protocol method.

Run your app and enjoy your auto scrolling slideshow!

That's it folks 🙂

As usual, you can download the completed project for this tutorial. Download the default scrolling slideshow project here, and download the auto scrolling slideshow project here.

SweetTutos
584 likes

Like Page Learn More

Be the first of your friends to like this

slideshow? what fancy animations will you use?

form below to get notified of all our upcoming

Subscribe

82 SHARES 7 60 4 7 4

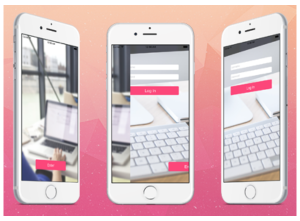
About Malek

Malek is a passionate iOS Engineer and Founder of Medigarage Studios, a small mobile games startup. I started my iOS adventures in 2011 and since then I fell in love with it. You can hire me for your project, get in touch by Email to discuss further details. Also, feel free to reach out on Twitter and Google+.

Related Posts

Working with UIWebView in Swift

In a previous tutorial, I wrote about UIWebView and UIActivityIndicator and how to use them…



[UIKit Dynamics Series] How to apply a force animation to your dynamic items with UIPushBehavior class

In the previous tutorial, you saw how to work with UISnapBehavior and UIDynamicItemBehavior to extend…

Develop RSS feed parser app in Swift

RSS is a well known web feed format, usually used to publish different informations frequently…

72 Comments Sweettutos 1 Login

Recommend 8 Share Sort by Best

Join the discussion…

Rui Damasio • a year ago
One more thing. I was having a problem after the tutorial: My scrollview also was having vertical scrolling. I solved the issue with:

self.scrollView.contentSize = CGSizeMake(self.scrollView.frame.width * 4, 1.0)

Got this solution here: http://stackoverflow.com/quest...

oblem - I applied the tutorial on a view that
some 3rd librarys. Anyway, if someone else have

SweetTutos
584 likes

Like Page Learn More

Be the first of your friends to like this

a year ago
he problem: "self.view.frame.height" was not
me reason, maybe because my scrollview are
frame.

Sascha Melcher • 8 months ago
Thanks for this tutorial! What about having an additional option to slide manually between the automatic slides? My ScrollView slides automatically every 8 seconds, but if I slide manually before time limit is reached, the page control doesn't get informed about the slide. Will work on this extension.
1 ^ | v • Reply • Share ›

Malek_T Mod ➦ Sascha Melcher • 8 months ago
Yello Sascha :)

If you want to further detect manual besides the automatic scrolling, you need to implement the scrollViewDidScroll protocol method. For this case, implement the following:

```
func scrollViewDidScroll(scrollView: UIScrollView)
{
scrollViewDidEndScrollingAnimation(scrollView)
}
```

Here, the scrollViewDidScroll will call the scrollViewDidEndScrollingAnimation protocol method which you already implemented to update the page control dots. Hope this helps!
^ | v • Reply • Share ›

Benny Chew • 8 months ago
This is a great tutorial, thanks a lot! I've been wondering for a long time how to do a paging scroll view in Swift. I didn't think it was so easy. It took me about 2 hours and I finally understand how this is done.
1 ^ | v • Reply • Share ›

Malek_T Mod ➦ Benny Chew • 8 months ago
Thanks Benny :)
Feel free to subscribe for more tutorials!
^ | v • Reply • Share ›

Matt Wyeth • 9 months ago
great tutorial, i love you <3
1 ^ | v • Reply • Share ›

Malek_T Mod ➦ Matt Wyeth • 8 months ago
You welcome, we love you too!
^ | v • Reply • Share ›

Fabrice • 10 months ago
Sorry for my last post, but i found the solution
Again thank you for this tutorial
1 ^ | v • Reply • Share ›

Sublervious • a month ago

Awesome tutorial, thank you very much! I'm having an issue when turning the device from portrait to landscape, the images are not displaying in the correct size, I'm seeing two images on one screen when rotated, If I go back to the previous screen of my app, rotate to landscape then proceed to the screen that has the paging backgrounds, it works fine, but if I rotate the device back to portrait the background image shows a large gap at the bottom. How could I correct this to either: 1) Just

ce it is a texture anyway) or 2) scale correctly

ance you can provide.

SweetTutos
584 likes
Like Page
Learn More
Be the first of your friends to like this

itional button in my case, I want to you "Login"

last page. In other words have two buttons

oes not show up. Any suggestion from you?

John Ken • 2 months ago

Hi malek,

Just wondering , I have an almost complete app I built using swift2. shouldI change it all to swift 3? is there an automated tool? and can you mix swift 2 and 3 code?

Thanks! love your tutorials!

Reply • Share ›

Malek_T Mod ➜ John Ken • 2 months ago

Hi John :)
Once you open your App project in Xcode 8, you should be prompted with the migration tool to Swift 3, follow the steps and your project source code will be upgraded to Swift 3 automatically. There might be some manual changes to be done though.
Hope this helps.

Reply • Share ›

dubi • 3 months ago

Thank you!! amazing tutorial...
i have one problem.
in iPhone 6 plus its work great but in iPhone 6 the scroll stop in the middle of the image.
why its happening?

Reply • Share ›

Kirk Hooten • 3 months ago

Great tutorial. I'd love to see the same in Objective-c as well.

Reply • Share ›

Odusina Timilehin • 4 months ago

Great Tutorial, thank you.
But my button is showing from the second page

Reply • Share ›

PrettyITGirl.com • 5 months ago

WHATTT??? YOU MADE IT! Whoever made this tutorial (probably Malek??) is awesome! I just found a blog before this one and posted this comment:

"This is cool. alextarrago, can you make a tutorial (I've been seeking for this for few days now) how to make a responsive UI using IB and ScrollView? I mean, for example, making a login screen, wherein you have SignIn Button, SignUp Button, Username TextField, and Password TextFields, and logo of the app as well. The challenge is that those fields should be positioned correctly even if you rotate your device and regardless what type of device you are running the app on."

I was going to paste this comment on this blog too but I realised that you made what I've been seeking for :(Thank you.

^ | v • Reply • Share ›

PrettyITGirl.com ➦ PrettyITGirl.com • 5 months ago

Uhmmm, maybe I was wrong, but this tutorial will surely help me. But if you have time, can you make my request above? Thanks.

^ | v • Reply • Share ›

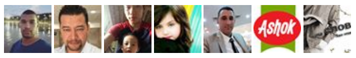
Muddana Mallikarjun • 6 months ago

n of app. when i use this solution its not having

SweetTutos
584 likes

Like Page Learn More

Be the first of your friends to like this



zed images being used as slides. How would one when that is not a constant? So that the image height will determine the height of the scroll view of each page?

^ | v • Reply • Share ›

Malek_T Mod ➦ Carolyn • 6 months ago

Hi Carolyn :)
You can dynamically calculate the sum of the height of your subviews and then assign it as part of the scroll view "contentSize" property.
Let me know if you need further help, I'll be glad to assist!

^ | v • Reply • Share ›

Carolyn ➦ Malek_T • 6 months ago

Thanks so much for the response. I'm having trouble figuring out how to assign that.

Let's say for those four slides that they are scrollview height, 1536, 800, scroll view height high. How can that be reflected in the contentSize property?

^ | v • Reply • Share ›

Malek_T Mod ➦ Carolyn • 6 months ago

Hi Carolyn, the content size is what counts in order for the scrollview to do paging correctly. So if you have n slide, the content size height should be (n*scrollView.frame.height).
This is regardless of the images' sizes.
Note: I assume you want to do scroll paging vertically. If you want to it horizontally, then the content size width would be (n*scrollView.frame.width) with keeping the content size height as it is (scrollView.frame.height).

^ | v • Reply • Share ›

Carolyn ➦ Malek_T • 6 months ago

I guess I'm not explaining myself well enough. I don't want to waste your time. If you do have any to spare, what I am hoping to accomplish is to have 4 pages, which can be scrolled through horizontally, as you've done here. But those each page is taller than the screen size/scrollView size. I'd like each page itself to scroll vertically in order to see all the information on that page.

So, for instance, page one. My image is 1024 x 1536. I want it to be for iPad, landscape, so the frame width is the same as the width of the image. If I just leave the code as is, well, it squishes down to the size of the screen, even if I have Aspect Fit, which it's being told to by the "let imgOne = UIImageView(frame: CGRectMake(scrollViewWidth, 0,scrollViewWidth, scrollViewHeight))"

When I change this to "let imgOne = UIImageView(frame:

CGRectMake(scrollViewWidth, 0,scrollViewWidth, 1536))",

see more

Reply • Share ›

Филипп Хлюпин • 8 months ago

Hello! great tutorial thank you! I saved much time with large part of my application.
I have a question, how protect scrollview to stop between images? I tried to put in
rating

idth

(CGRectMake(slideToX, 0, pageWidth,
ame)), animated: true)

page will started only after scroll animation is

SweetTutos
584 likes

Like Page
Learn More

Be the first of your friends to like this

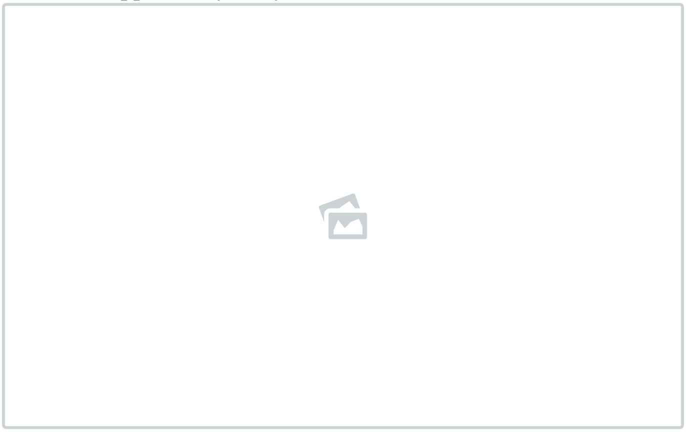
Malek_T Mod ➦ Филипп Хлюпин • 8 months ago

Hi :)
What do you try to do? By "protect scrollview to stop between images", do you mean setting a delay between pages transition ?

Reply • Share ›

Филипп Хлюпин ➦ Malek_T • 8 months ago

Hello Malek!
this what happened if you try to scroll:



So scrollview stops between images. And I want scrollview to continue scrolling till the next page. Thank you.

Reply • Share ›

Malek_T Mod ➦ Филипп Хлюпин • 8 months ago

I believe the content size is somehow incorrectly set or calculated. If you have 'n' slides to show (for example 4), how much is the value of the scroll view content size?
Also, did you set the content size width to be as much as the scroll view frame width (multiplied by the number of slides like below)?

```
self.scrollView.contentSize =
CGSizeMake(self.scrollView.frame.width * 4,
self.scrollView.frame.height)
print(self.scrollView.contentSize)
```

Reply • Share ›

Филипп Хлюпин ➦ Malek_T • 8 months ago

Well you see problem is different. Slide width calculates properly. But
if you try to slide manualy, scrollView doesn't continue scrolling till
the end of new slide.
I tried to put slide scrolling code into

I tried to put slide scrolling code into
scrollViewDidEndDecelerating but in that case only after
animation is
over (or you should touch screen again) scroll view will scroll
properly.
What event I should handle to catch when I take my finger out
of the screen after my finger initiate scrolling?

^ | ∨ • Reply • Share ›

SweetTutos
584 likes

Like Page Learn More

Be the first of your friends to like this

ин ➜ Филипп Хлюпин • 8 months ago

issue adding scrolling: paging enabled

y • Share ›

п Хлюпин • 6 months ago

you mean when you said you added scrolling:
ed parameter. I would like to do the same thing
g trouble finding the code for this on the internet

^ | ∨ • Reply • Share ›

Malek_T Mod ➜ Tyler • 6 months ago

Hi Tyler, you just need to enable paging for the scrollview:
self.scrollView.pagingEnabled = true
or directly from the Attributes inspector, check the "Paging
Enabled" property.

^ | ∨ • Reply • Share ›

James Gobert • 10 months ago

Awesome tutorial! **@Malek_T**

I modified your code a bit but having an issue.

Instead of having a set background image, my app dynamically chooses an image based on a Switch case that is run after an API call. My thought is to somehow get the result of my case (String) and place it in the:

imgOne.image = UIImage(named: "dynamicBackground")

Is there a better way to do this? My API is called in viewDidLoad() so the relevant data should be available on load. It was working fine before I added the ScrollView. I just want it to show the same dynamic background on all pages.

(I made sure the background ImageView isn't nested inside the ScrollView FYI)

An even better solution would be how can I get remove your background implementation altogether and just keep my same background solution working, even when I swipe to show more content.

Thanks for any help with this!

James

^ | ∨ • Reply • Share ›

Malek_T Mod ➜ James Gobert • 10 months ago

Hi James,
Can you show some of your relevant code and the UI hierarchy ? Please open a forum thread here http://sweettutos.com/forums/, I will make sure to follow up with you.

^ | ∨ • Reply • Share ›

James Gobert ➜ Malek_T • 10 months ago

I actually figured out another way to get the outcome I wanted.
Thanks for responding!

^ | ∨ • Reply • Share ›

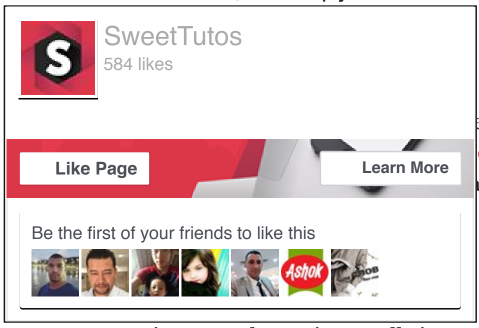
Malek_T Mod ➜ James Gobert • 10 months ago

Cool!

^ | ∨ • Reply • Share ›

Raj • 10 months ago
Hi Malek, thank you very much for the tutorial. Highly appreciate it.
Reply • Share ›

Raj → Raj • 10 months ago
Do you have a Youtube channel?
Reply • Share ›


SweetTutos
584 likes
Like Page
Learn More
Be the first of your friends to like this

0 months ago

ee to subscribe
e.com/channe...
re ›

Nice example. Paging scroll views aren't used often enough in iOS apps - maybe because they have historically been difficult to implement.

I took a stab at implementing your example using MarkupKit, an open-source project I developed. Just thought I'd share the results in case you were interested:

http://gkbrown.org/2016/01/09/...

If you have a minute to check it out, I'd be interested to hear what you think. Thanks!
Reply • Share ›

Malek_T Mod → Greg Brown • 10 months ago
Hi Greg, just download your project code, it's just amazing. Great work you put on the library. Keep it up :)
Reply • Share ›

Greg Brown → Malek_T • 10 months ago
Thanks. :-) Hope you find it useful. Let me know if you have any questions!
Reply • Share ›

Konsy • 10 months ago
Hey! Great tutorial, really helpful :) I was wondering if it would be possible to gain some help though, I was looking to use this feature to display different buttons rather than images, this would be so the user can scroll through and select a character which loads a different view controller (hopefully that makes sense). I'm still new to coding and it would be great if you could assist me in anyway on doing this? Thank you!
Reply • Share ›

Malek_T Mod → Konsy • 10 months ago
Hi Konsey,

Sure, you can simply replace UIImageView with UIButton, and that should work correctly. Also, you can keep your character objects as UIImageView and attach a UITapGestureRecognizer object to detect click events so that behave like UIButtons.

Let me know how it goes, if you already tried something and got troubles, then feel free to ask on the Forum section, I will be there to assist you further :)
Reply • Share ›

doctor • a year ago
Great tuto! Please make in the future, a tutorial teaching how to merge 2 xcode projects in one. Will be very usefull! thanks a lot
Reply • Share ›

Malek_T Mod → doctor • 10 months ago

Malek_T Mod ↪ doctor • 10 months ago

Hi, can you elaborate on the idea. Do you mean how to mix Objective-C project with Swift one ?

^ | ˅ • Reply • Share ›

Rui Damasio • a year ago

Hey Malek_T, thanks for the great and illustrated tutorial. I have a question: I could not understand the difference of use between the pure UIScrollView vs UIPageViewController. For example, in my project I dont know which one I should ove my tableview, and the user should be able to mage with some information, like the ss from the any applications contain the same layout and I'm you think I should go? Thanks again! Regards!

SweetTutos
584 likes

Like Page Learn More

Be the first of your friends to like this

^ | ˅ • Reply • Share ›

M Reddy • a year ago

awesome...

^ | ˅ • Reply • Share ›

Malek_T Mod ↪ M Reddy • a year ago

Thanks :)

^ | ˅ • Reply • Share ›

Suat Ozgur • a year ago

Extremely helpful, thank you for the great tutorial, Malek!
I need to ask a question by hoping you could lead me to the right way. I will already use the paging scrollview that you presented. However, as a separate view, I need a scrollview that I can put some images to click to open the corresponding views, consider images are the section buttons, plus they need to be scrolled vertically (not horizontally) and work in different devices as in your sample even my sample only needs to work portrait to keep it simple for now. I attached an image to show how it looks like. Can I still use scrollview for this purpose? Should I put a view in the scrollview that holds the images?

^ | ˅ • Reply • Share ›

Load more comments

ALSO ON **SWEETTUTOS**

Networking in Swift: How to download a file with NSURLSession
5 comments • a year ago•
Avat deepak vaishnav — only one PDF file download i need to more download by selected table view cells

How To Make Expandable UITableViewCells For Dynamic Text …
3 comments • 3 months ago•
Avat CK13 — Sorry that you misunderstood. I mean I have to change some code in my Xcode 8 like override func prepare(for …

How to make a custom info window for your marker with Google Maps …
3 comments • a year ago•
Avat Guy Kahlon — Hey Malek. Thanks for very helpful tutorial.A question: How can I load an asynchronies Image to my …

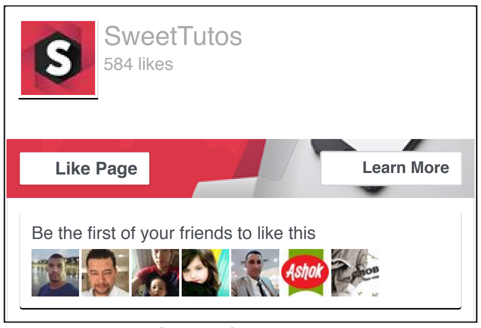
[Swift] How to read and write into plist files
12 comments • a year ago•
Avat Luis Lopes — Hello.I'm trying to use it on my app but I'm getting this error:Could not cast value of type …

Subscribe   Add Disqus to your site Add Disqus Add   Privacy

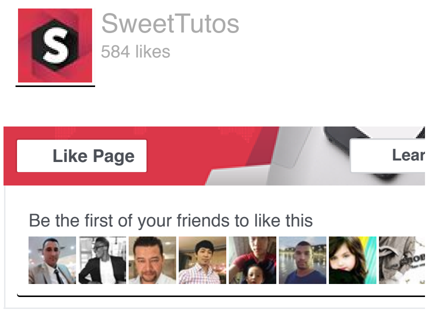
[Swift MapKit Tutorial Series] How to search a place, address or POI on the map

Make games with Unity [Part 2: Customising scenes and deploying to your device]

Top Posts & Pages



UIImage Files in The Document Directory with Swift



Recent Posts

- How To Implement a Sticky View While Table View Scrolls; Like App Store app [Swift 3]
- How To Make Expandable UITableViewCells For Dynamic Text Height With Auto Layout
- How To Hugely Simplify Your iOS Development With MarkupKit
- Need For Something Different From Map Markers? Try Google Maps Ground Overlays

Recent Comments

- Daniel on [Swift MapKit Tutorial Series] How to make a map-based overlays
- Marko Teodorovic on How to use the Google Places Autocomplete API with Google Maps SDK on iOS
- org on How to make a simple 2D drawing app in Swift
- Malek_T on How To Play Sounds Files And Manage Duration Progress – AVAudioPlayer Tutorial
- Islombek Hasanov on How To Play Sounds Files And Manage Duration Progress – AVAudioPlayer Tutorial