IBDesignable IBInsepectable

<https://medium.com/@nimjea/ibdesignable-and-ibinspectable-in-swift-c12ea557b82b>

<https://www.freecodecamp.org/news/how-to-create-a-beautiful-reusable-gradient-view-in-swift-with-ibdesignable-981aebb43d30/>

<https://www.iosdev.recipes/uiview/apis-you-forgot-layerclass-maskView-and-uitintadjustmentmode/>

<https://stackoverflow.com/questions/41680946/how-to-apply-gradient-to-uiview-in-ios-10-on-a-real-device>

InitialUpdateCorners

<https://gist.github.com/stevencurtis/9c31b5c7943213188bf5c8d105e8c1e8>

UpdateCornersViewcornerRadius

<https://gist.github.com/stevencurtis/a778daa7af6fc79c4eba7b5f2589f351>

updatedUpdateCornersView

https://gist.github.com/stevencurtis/a859156444daddb6e7af320c0ff6ec36

### Using IBDesignable and IBInspectable in Swift

#### See your stuff in the Storyboard

Photo by [Molly Belle](https://unsplash.com/@mollybelle?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com/?utm_source=medium&utm_medium=referral)

Difficulty: Beginner | **Easy**| Normal | Challenging

If you are using Storyboards to create your App, it is rather lovely to see your work (layout and design) all come together. You can also see your new subclass do exactly the same, and this is what IBDesignable and IBInspectable are for!

Let us get started!

### Prerequisites:

* You’ll need to [create a single view project](https://medium.com/swlh/your-first-ios-application-using-xcode-9983cf6efb71) using storyboards to be able to see your changes

### Terminology

IBDesignable: An attribute that allows the Storyboard file to see and compile a custom UI element

IBInspectable: An attribute that allows the Storyboard file to allow properties of a custom UI element to be altered in the Storyboard

Storyboard: A visual representation of the User Interface of an Application

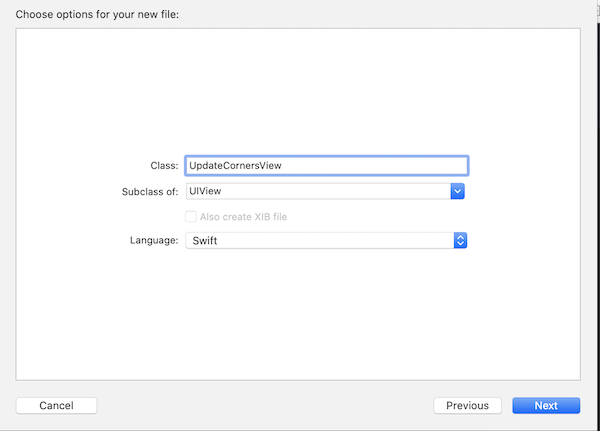
### Motivation

The Storyboard is a fantastic feature of the iOS SDK, and it allows us to see the user interface right in the Storyboard — without needing to fully compile and run a project.

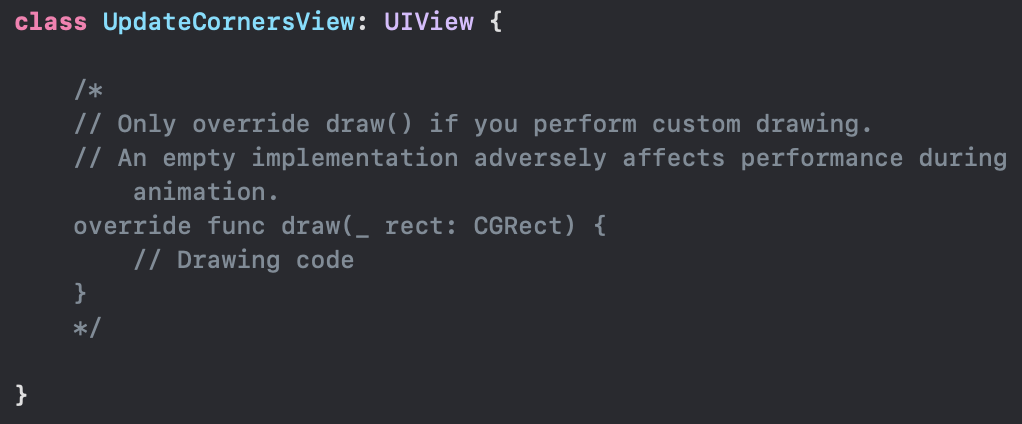
### A simple UIView subclass

Subclassing in Swift is actually pretty simple — as we actually get a little bit of help when we do so.

When we create through File>New>File we can choose to make our new file a subclass:

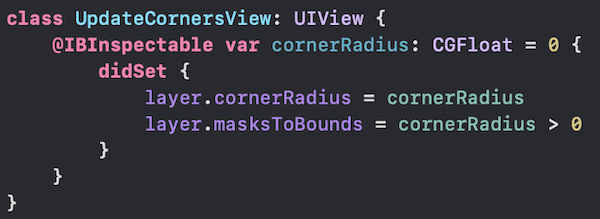


So in this case UpdateCornersView

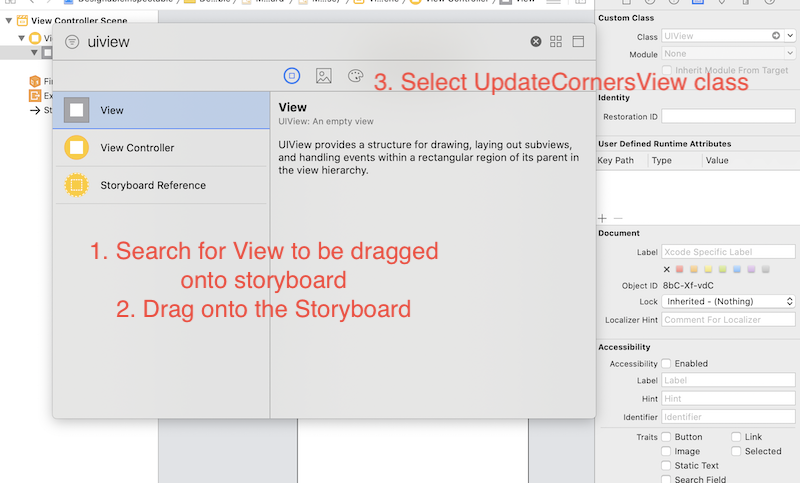
[Click for Gist](https://gist.github.com/stevencurtis/9c31b5c7943213188bf5c8d105e8c1e8)

In this particular tutorial we don’t need to perform custom drawing, so thanks Swift but we won’t override draw(\_ rect: CGRect).

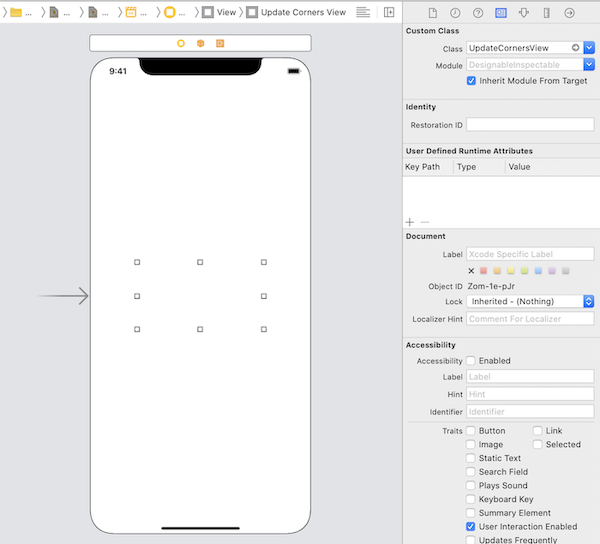
Instead we will use the attribute @IBInspectable to se the corner radius using the didSet [property setter](https://medium.com/swlh/property-getters-and-setters-in-swift-8157d5d027c7)

[Click for Gist](https://gist.github.com/stevencurtis/a778daa7af6fc79c4eba7b5f2589f351)

this gives us the ability in the storyboard to see the UpdateCornersView. This is a multi-step process — we need to drag and drop a UIView onto the storyboard, set the class as UpdateCornersView.



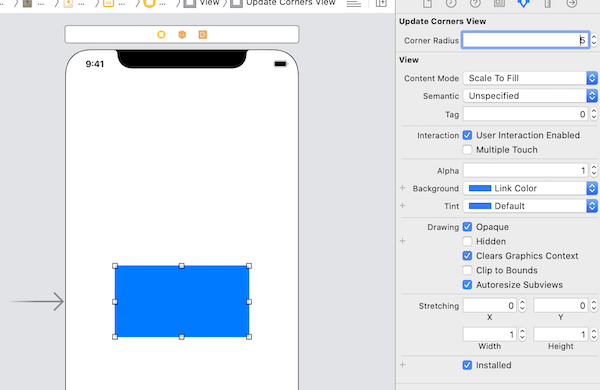
Giving the following result:



Now by selecting the following icon:

https://cdn-images-1.medium.com/max/1600/1*gsTFHkFlwDsFUEb-7Z6Aug.png

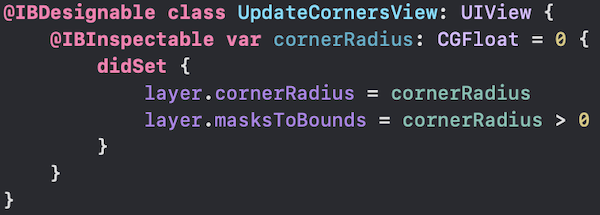
And then adjusting the corner radius (and having selected a background color so we can see what is going on) as follow — which appears to be nothing.



However this isn’t quite true. Things **ARE** happening. Running on a device reveals that the UIView does have rounded corners!



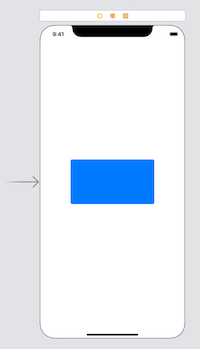
So what is going on? The UIView is updated in the build but not in the storyboard? Well — the truth is that by default the storyboard does not update.

[Click for Gist](https://gist.github.com/stevencurtis/63a30077269bae01834ce3b6a847c32f)

It is only using the attribute @IBDesignable that gets the storyboard to update.

You will notice that once you add @IBDesignable and then switch to the storyboard the project will build and then the storyboard will change.

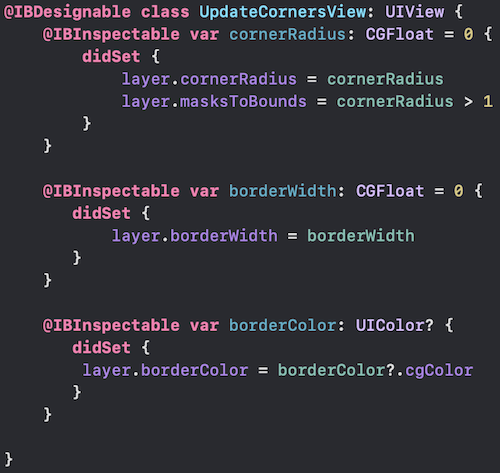
Fantastic!



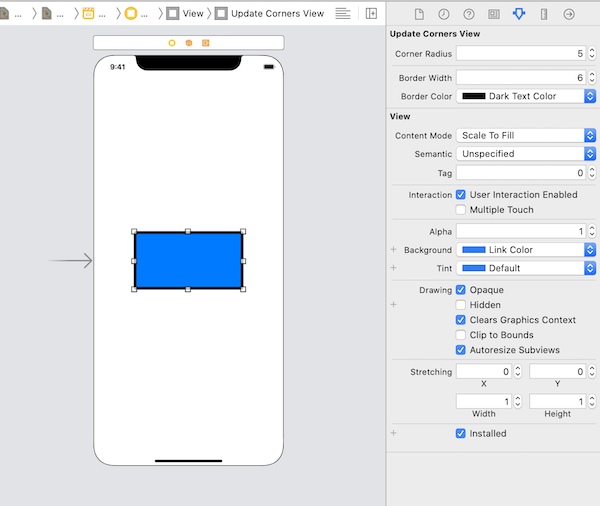
#### Updating the simple UpdateCornersView class

The class created above isn’t all that useful. What about adding on useful functionality.

What about a nice — looking border?

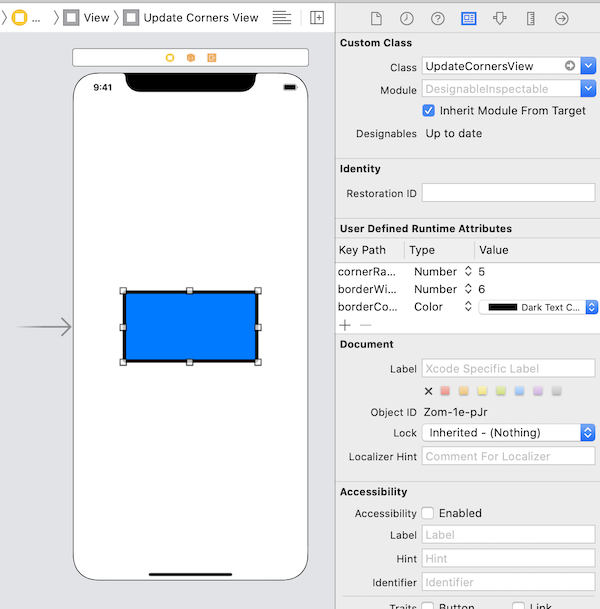
[Click for Gist](https://gist.github.com/stevencurtis/a859156444daddb6e7af320c0ff6ec36)

We can now update the border width and border color, using the property setters in a similar style to the above.

[Click for Gist](https://gist.github.com/stevencurtis/a859156444daddb6e7af320c0ff6ec36)

Of course this also updates any compilation in the Simulator. Nice!

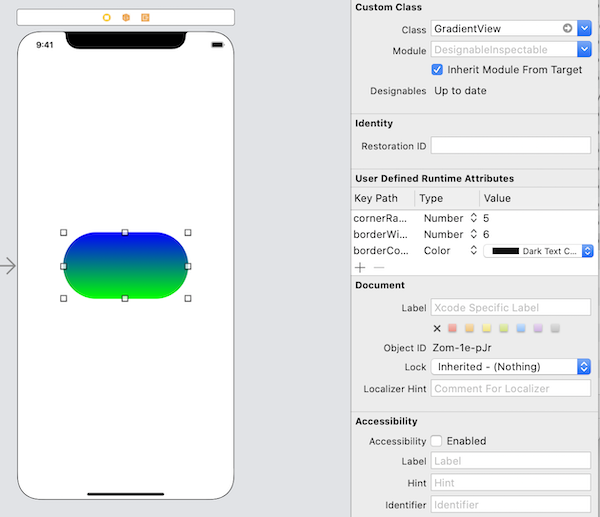
But did you notice something else? There is the **User Defined Runtime Attributes** under the class — that means that your attributes can be altered there too!



See it? That’s a nice update place (I think)

### A UIView subclass with a gradient

Subclassing with a gradient is fun!



The coding?

Take a look here:

[Click for Gist](https://gist.github.com/stevencurtis/ea55f04a87b1e952f01ae09f353d2b7d)

### Conclusion:

Yes, you can look at the completed repo to [see how this can be implemented in a real project](https://github.com/stevencurtis/IBDesignableIBInspectable). This is an extremely important area, and creating subclasses of views is a great way of seeing how your changes will alter your project in real time. That is, with UIKit.

Want to get stuck in? Certainly, I do too.

### Extend your knowledge

* [Read Apple’s documentation for UIView](https://developer.apple.com/documentation/uikit/uiview)

### The Twitter contact:

Any questions? [Get in touch with me on Twitter](https://twitter.com/stevenpcurtis)