

CUSTOM CONTROLS IN iOS



HANDS-ON CHALLENGES

Custom Controls in iOS

Catie & Jessy Catterwaul

Copyright ©2017 Razeware LLC.

Notice of Rights

All rights reserved. No part of this book or corresponding materials (such as text, images, or source code) may be reproduced or distributed by any means without prior written permission of the copyright owner.

Notice of Liability

This challenge and all corresponding materials (such as source code) are provided on an "as is" basis, without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose, and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in action of contract, tort or otherwise, arising from, out of or in connection with the software or the use of other dealing in the software.

Trademarks

All trademarks and registered trademarks appearing in this book are the property of their own respective owners.

Challenge #2: Your First Custom Control

By Catie & Jessy Catterwaul

Your challenge is to create a property to adjust the padding for the image and make the stack view's size interact beautifully with the border width.

Start by creating a computed `imagePadding` property, below the `borderWidth`:

```
var imagePadding: CGFloat {  
}
```

You're going to use the image's `alignmentRectInsets` property to create even padding around the image.

Add the following to `imagePadding`:

```
var imagePadding: CGFloat {  
  get {  
    return image?.alignmentRectInsets.top ?? 0  
  }  
  set {  
    image = image?.withAlignmentRectInsets(  
      UIEdgeInsets(  
        top: -newValue,  
        left: -newValue,  
        bottom: -newValue,  
        right: -newValue  
      )  
    )  
  }  
}
```

To see the effect of this property, set the spacing to something easily visible, near where you set all of the other API properties:

```
deluxeButton.imagePadding = 15
```



In the live view, you can see the image padding working wonderfully!

But remember what we saw in the demo. What happens if we change `borderWidth` to something much smaller, like 2?

```
deluxeButton.borderWidth = 2
```



or larger, like 20?

```
deluxeButton.borderWidth = 20
```



Ah! If the border is too small, we can see an unwanted space around the label. But if the border is too large, our text, and eventually the image, will be covered up by the border! We are laying `stackView` out based on the layout margins, but they're currently left at their default values. Luckily, you can explicitly set layout margins!

In the `set` for `borderWidth`, set the `layoutMargins` as well:

```
set {  
    layoutMargins = UIEdgeInsets(  
        top: newValue,  
        left: newValue,  
        bottom: newValue / 2,  
        right: newValue  
    )  
    layer.borderWidth = newValue  
}
```



The top, left, and right margins are set to be the same as the `borderWidth`, and the bottom is set to 1/2 the border width so the text will remain more centered within the block of color on the bottom.

What if we want to go for an artsy, cropped effect with the image?

Set `borderWidth` to 10, and `imagePadding` to a significantly negative number, say -20:

```
deluxeButton.borderWidth = 10
...
deluxeButton.imagePadding = -20
```



Now the image is expanding past the edges of the view, as we'd like, but it's not being cropped!

Back in `initPhase2`:

```
clipsToBounds = true
```



Tadaa! Mission accomplished. Your image and text are now adjusting beautifully along with the border! :]