

# Intermediate iOS Animation

Hands-on Challenges

# Intermediate iOS Animation Hands-On Challenges

Copyright © 2016 Razeware LLC.

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.

This book 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 or other dealings in the software.

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



# Challenge A: Re-using animations

In this short challenge you are going to exercise re-using and modifying animations. Do you remember that `CABasicAnimation` is just a data model that you send to the Core Animation server to render? Well, nothing really stops you to modify the used animation and add it to yet another layer.

Let's give that a try.

Five seconds after you tint the button background you will animate the background color back to its original color.

The starter project includes a handy `delay` function, which you will use. At the bottom of `actionLogin()` add:

```
delay(seconds: 5, completion: {
    self.loginButton.layer.backgroundColor = startColor

    tint.fromValue = tintColor
    tint.toValue = startColor
    self.loginButton.layer.addAnimation(tint, forKey: nil)
})
```

At first you change directly the button background to the final value of the animation. Then you exchange the `fromValue` and `toValue` of the animation you used previous and then add it again to the button.

It's worth noting again that this is possible because `addAnimation(_, forKey:_)` copies the supplied animation object. That's why when you modify the object afterwards it does not affect the first animation that is already running.

