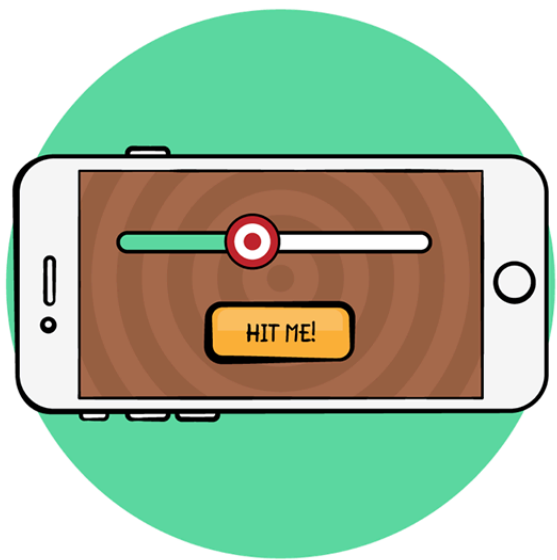


# BEGINNING iOS 10

**PART 1**

GETTING STARTED



**HANDS-ON CHALLENGES**

## Beginning iOS 10 Part 1: Getting Started

Brian Moakley

Copyright ©2016 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 #5: Accessing Controls

By Brian Moakley

In this challenge, you'll be creating a tip calculator. The tip calculator has already been created for you. Only, instead of printing the result to the console, you will have to print to the actual screen. None of the outlets have been set up so it's up to you to create all the actions and the outlets.

## Getting Started

There's a few things that need to be added. First, when working with text fields, they won't automatically dismiss when the users hits the return key. You have to add that behavior, otherwise, the players will not be able to escape them.

Open the Xcode starter project and in **ViewController**, add the following code underneath the ViewController class:

```
extension ViewController: UITextFieldDelegate {  
    func textFieldShouldReturn(_ textField: UITextField) -> Bool {  
        textField.resignFirstResponder()  
        return true  
    }  
}
```

This bit of code implements the method to return the keyboard. To do this, the text field must resign its first responder status. A first responder means that the control is receiving all the input. By resigning its status as a first responder, other controls can receive events. This also causes the keyboard to dismiss itself.

That's only part of the solution. Your text fields must become delegates. This means, they must listen to events of the control. Once you name your text fields, you need to set the delegate like so:

```
amount.delegate = self
```

The text field will now respond to the event.

The rest now is up to you! Add some outlets to the text view and the labels. To get information from a text field, you use the text property. If you have a text field called amount, you'd access the value like so:

```
let total = amount.text
```

This returns a string so you will have to cast the value. Here's the formula to calculate the tip:

```
amount * percentage / 100
```

Good luck!