张靖元 Jingyuan “Knight” Zhang

Mobile Designer & Developer

CocoaHeads Shanghai 2017-02-23

Profile

Portfolio

Who am I ?

- 张靖元 Jingyuan “Knight” Zhang
- Full Sail University - Mobile Development
- Fudan University Shanghai Institute of Visual Art - Visual Communication Design
- Apple Retail - Creative trainer
- Upwork Platform-approved Freelance Mobile Developer

“Simple techniques that make Xcode Storyboard more intuitive.”

What to share tonight ?

- one expectation for intuitive Storyboard
- Strategies for intuitive Storyboard
- How to get started
- Inspiration for the future

Thanks for your support !

Thanks for your support !

Guanshan Liu

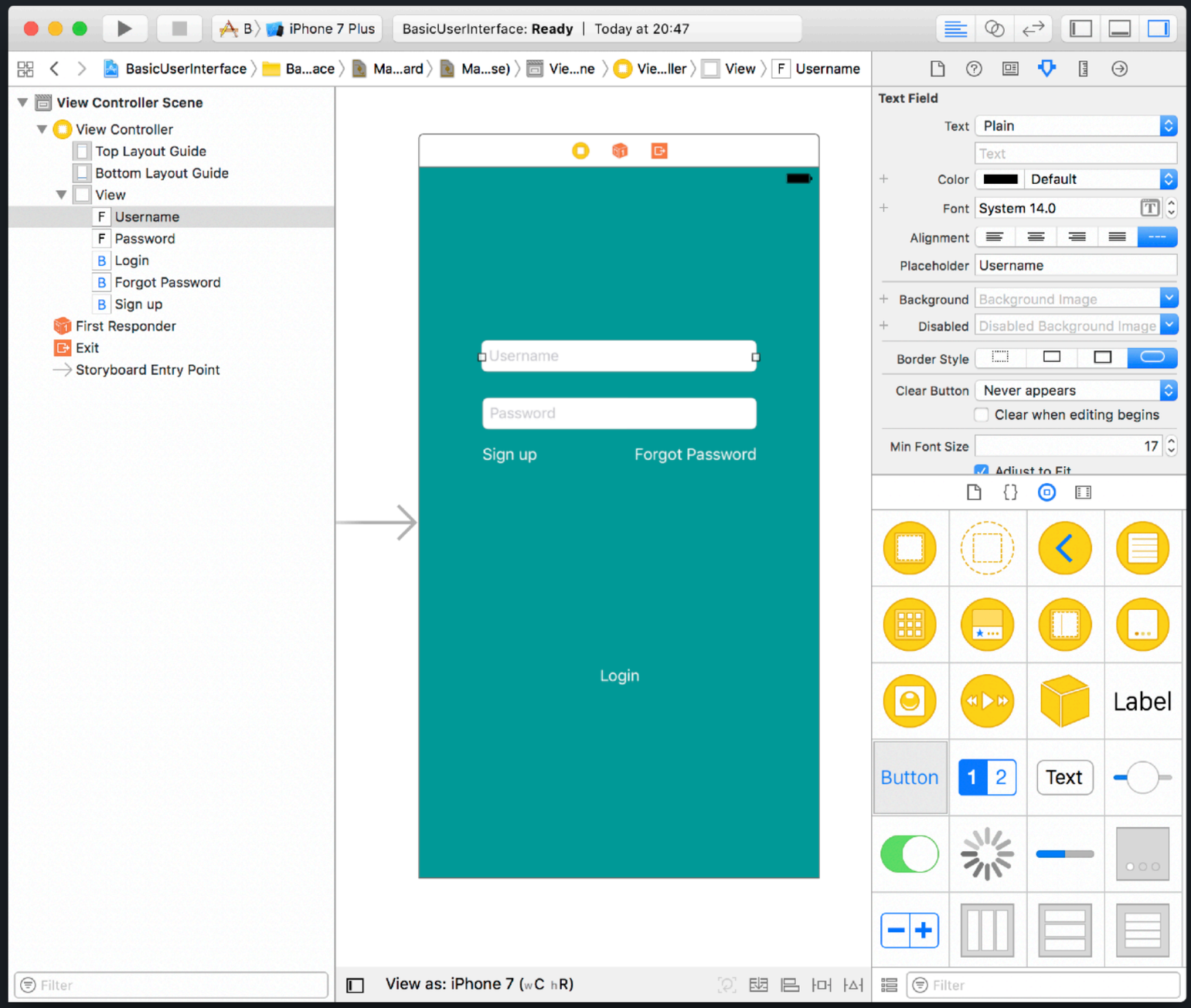
“What do we expect for an intuitive Storyboard ?”

What do we expect for an intuitive Storyboard ?

- “WYSIWYG”
- “What You See Is What You Get”

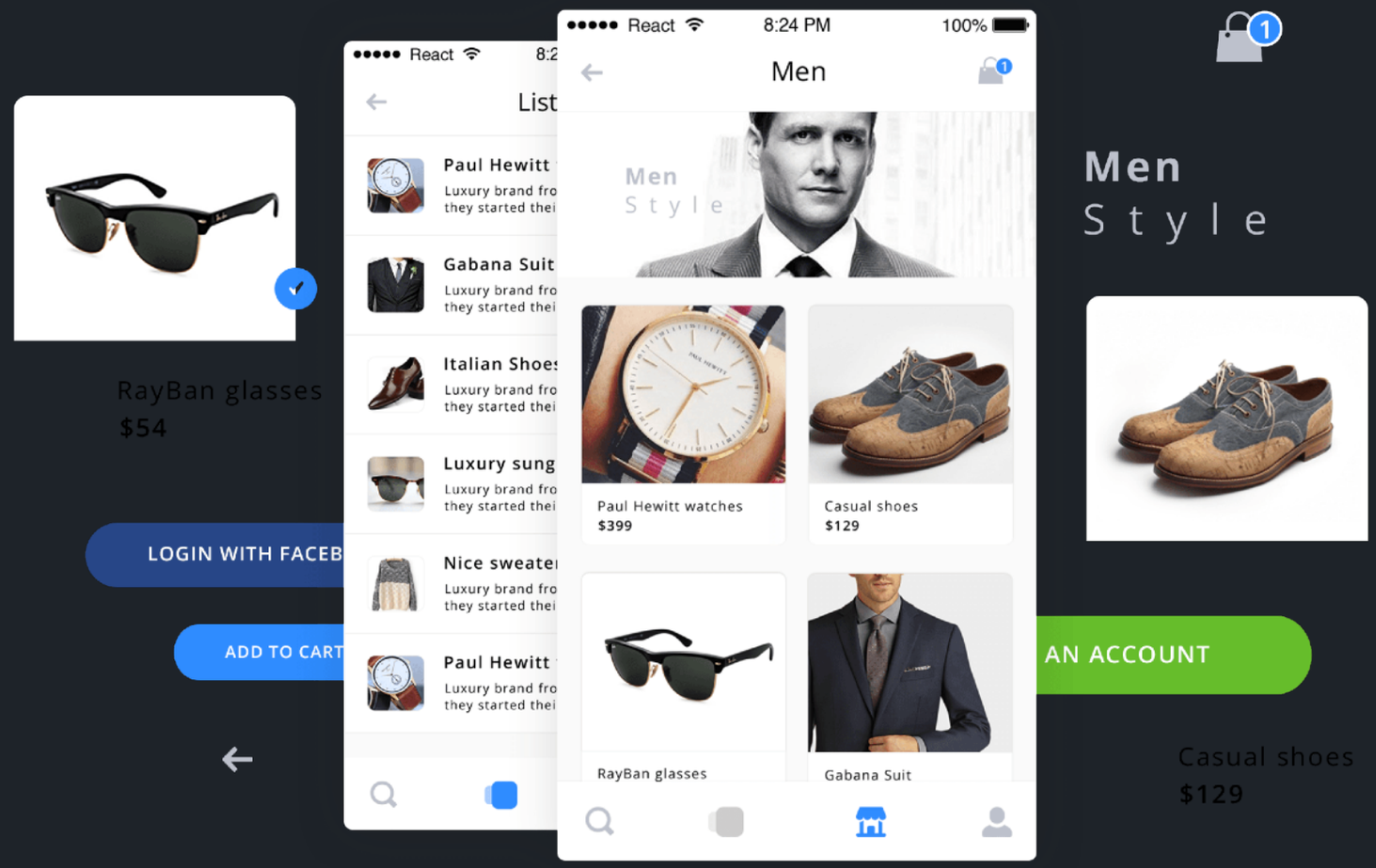
What do we expect for an intuitive Storyboard ?

- Apple iOS built-in UIKit Framework



What do we expect for an intuitive Storyboard ?

- Your own custom user interface elements



http://ui-cloud.com/guacamole-ui-kit/

Extend Xcode project behaviors

like image creation tools

Extend Xcode behaviors

- Use built-in Apple Swift and Xcode features
- Scale limitlessly
- How to get started

What do we use ?

@IBDesignable

@IBInspectable

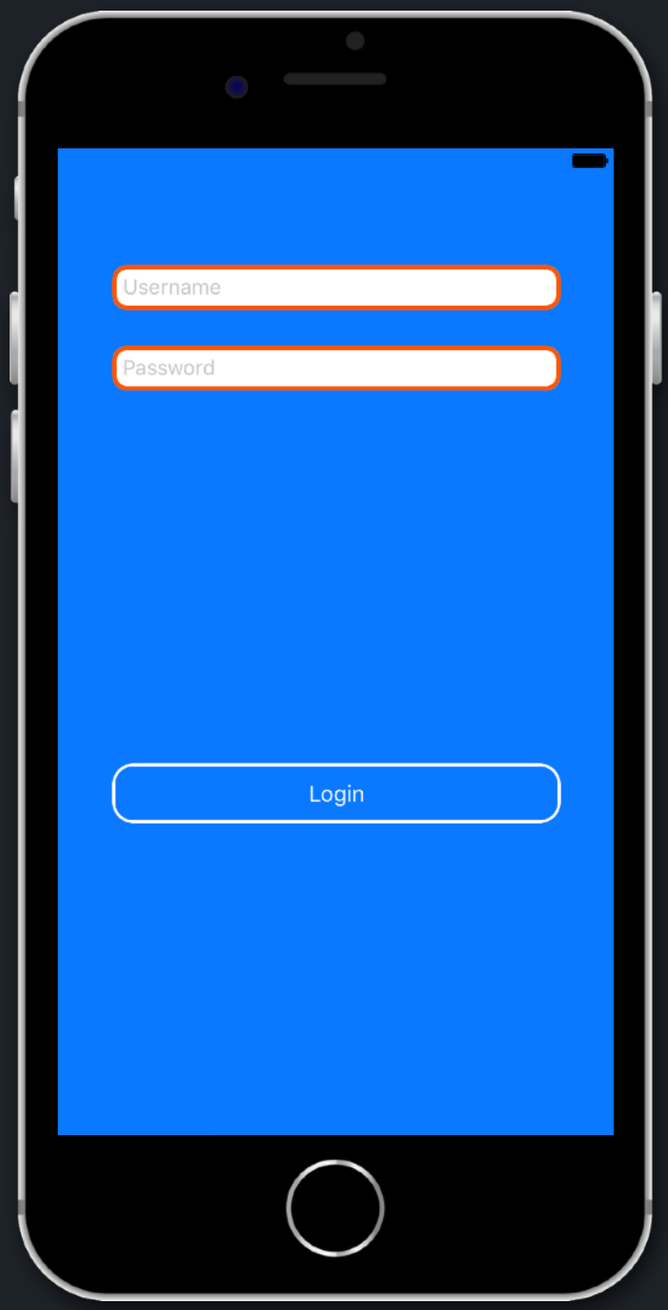
Swift Extensions

“you can use two different attributes, @IBDesignable and @IBInspectable, to enable live, interactive custom view design in Interface Builder.” – Apple

“Extensions add new functionality to an existing class, structure, enumeration, or protocol type…Extensions are similar to categories in Objective-C.” – Apple

A simple challenge

- Generic login view
- 2 text fields
- 1 login button
- rounded corners
- border stroke



A simple challenge of Generic Login View

- “User-Defined Runtime Attributes”
- “Key-Value Coding”
- @IBInspetable - Adjust properties in Xcode Inspector
- @IBDesignable - Live render in Xcode Storyboard

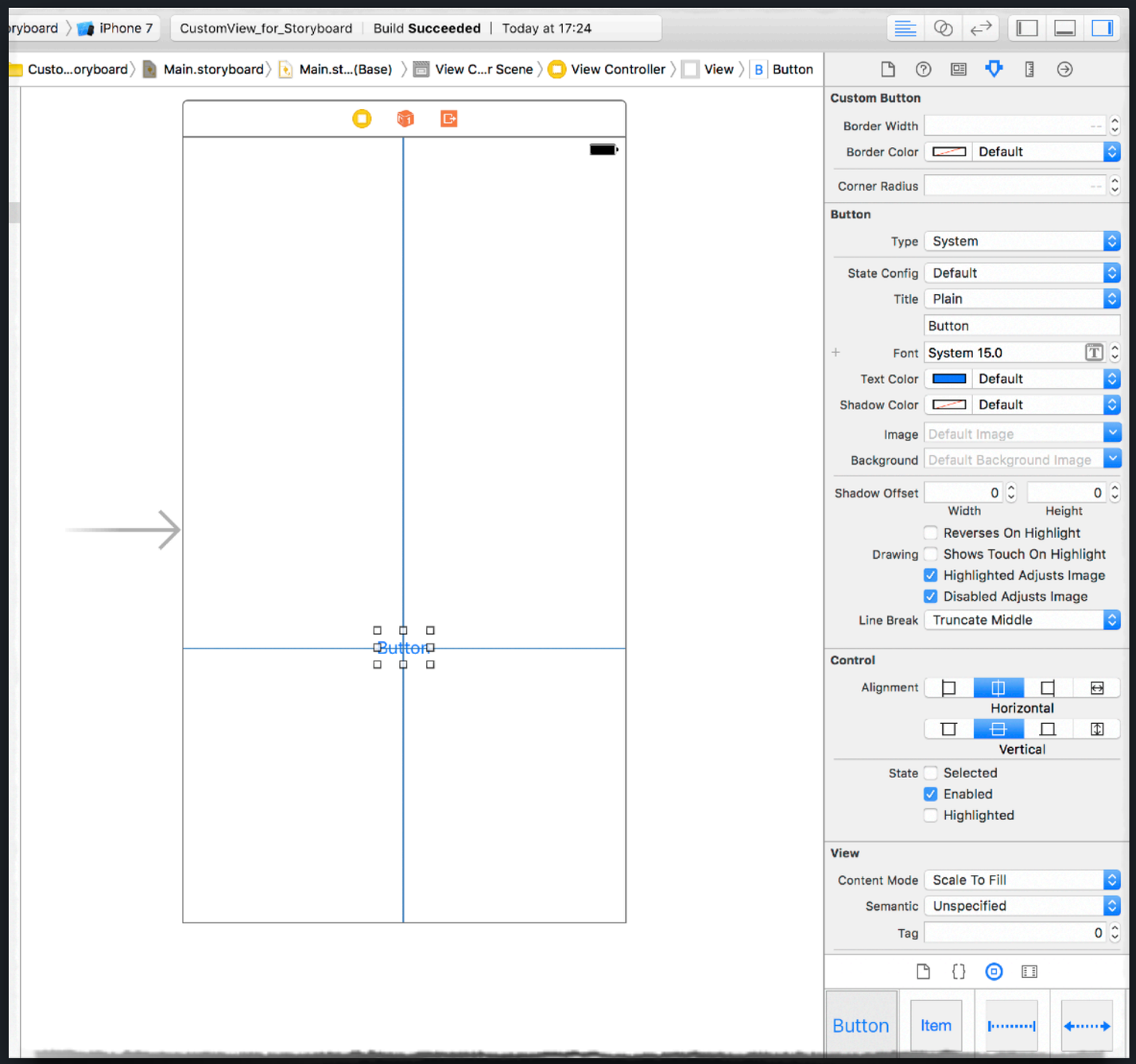
@IBDesignable, @IBInspectable

- Custom object types - @IBDesignable
- Custom properties - @IBInspectable

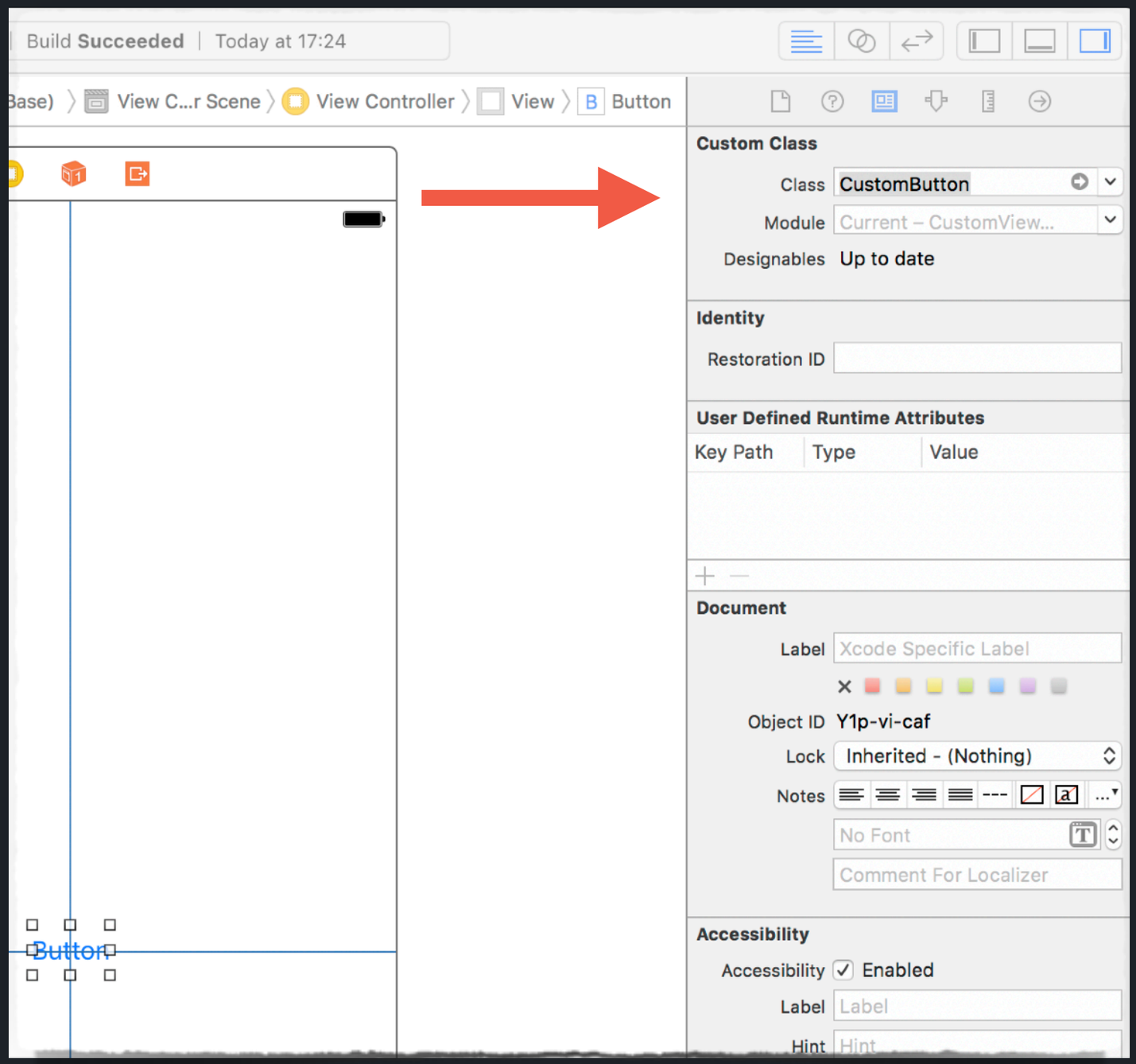
Start with the custom button

```
8
9   import UIKit
10
11  @IBDesignable
12  class CustomButton: UIButton {
13
14      @IBInspectable
15      var borderWidth: CGFloat = 0 {
16          didSet {
17              layer.borderWidth = borderWidth
18          }
19      }
20
21      @IBInspectable
22      var borderColor: UIColor? {
23          didSet {
24              layer.borderColor = borderColor?.cgColor
25          }
26      }
27
28      @IBInspectable
29      var cornerRadius: CGFloat = 0 {
30          didSet {
31              layer.cornerRadius = cornerRadius
32              layer.masksToBounds = cornerRadius > 0
33          }
34      }
35
36  }
37
```

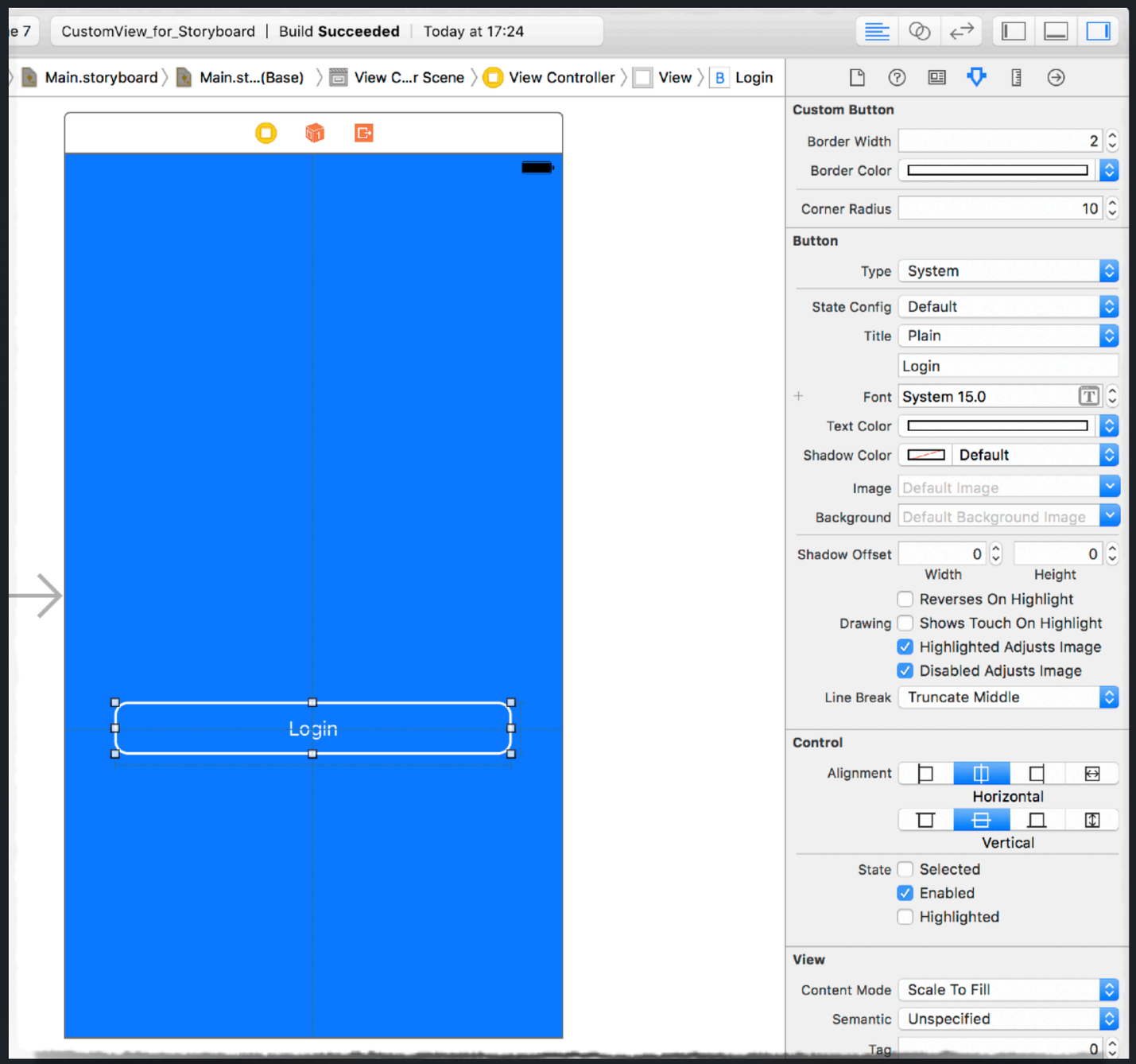
Place a UIButton to Storyboard



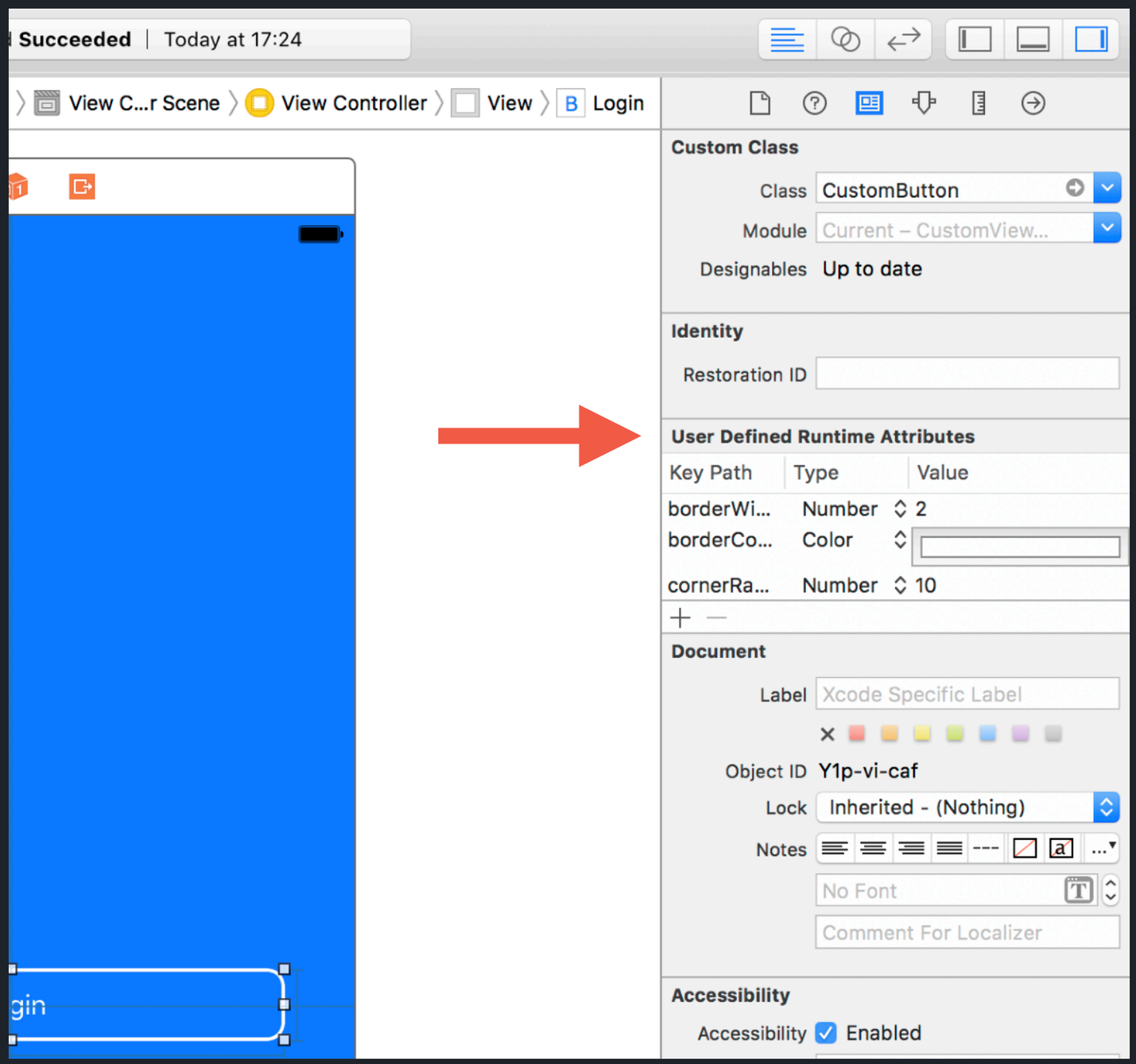
Set the Custom Class



Play in “Attribute” Inspector



“User-Defined Runtime Attributes”



What about the text fields ?

- Swift Extensions
- UIView
- Generic approach

The generic approach

- Extend UIView via Swift Extensions
- Custom properties - @IBInspectable
- Inherit custom object types from built-in counterparts
- Custom object types - @IBDesignable

Extend UIView

```swift

import UIKit

extension UIView {

    @IBInspectable
    var borderWidth: CGFloat {
        get {
            return layer.borderWidth
        }
        set {
            layer.borderWidth = newValue
        }
    }

    @IBInspectable
    var borderColor: UIColor? {
        get {

            if layer.borderColor != nil {
                return UIColor(cgColor: layer.borderColor!)
            } else {
                return nil
            }

        }
        set {
            layer.borderColor = newValue?.cgColor
        }
    }

    @IBInspectable
    var cornerRadius: CGFloat {
        get {
            return layer.cornerRadius
        }
        set {
            layer.cornerRadius = newValue
            layer.masksToBounds = newValue > 0
        }
    }

}

```

```swift

import UIKit

@IBDesignable
class CustomButton: UIButton {

    @IBInspectable
    var borderWidth: CGFloat = 0 {
        didSet {
            layer.borderWidth = borderWidth
        }
    }

    @IBInspectable
    var borderColor: UIColor? {
        didSet {
            layer.borderColor = borderColor?.cgColor
        }
    }

    @IBInspectable
    var cornerRadius: CGFloat = 0 {
        didSet {
            layer.cornerRadius = cornerRadius
            layer.masksToBounds = cornerRadius > 0
        }
    }

}

```

Compare to
what we had done previously
for CustomButton Class

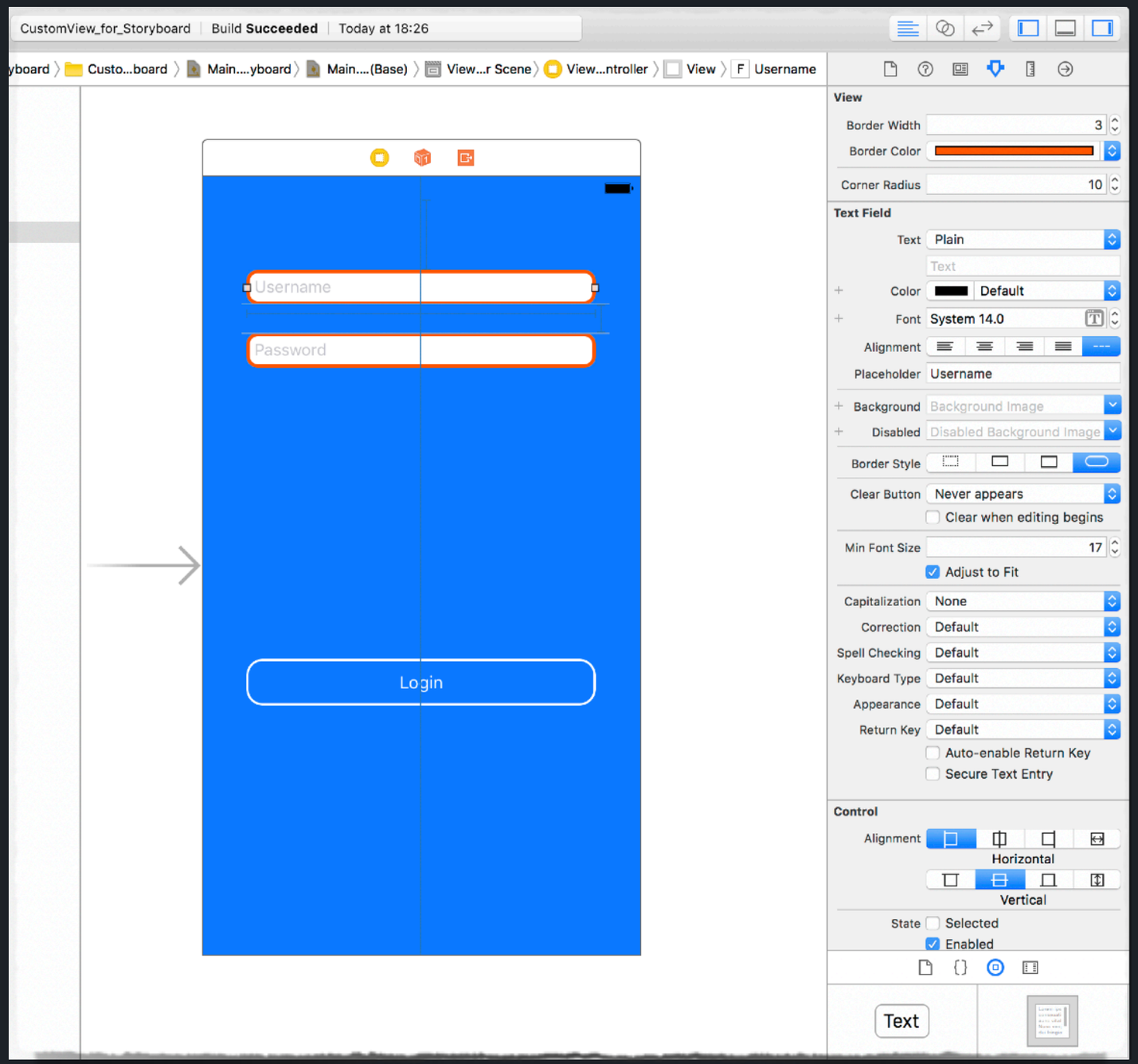
Inherit from built-in counterparts

```
8
9   import UIKit
10
11  @IBDesignable
12  class CustomTextField: UITextField {
13
14  }
15
```

Clean CustomButton

```
8
9  import UIKit
10
11 @IBDesignable
12 class CustomButton: UIButton {
13
14 }
15
```

Play with CustomTextFiled as before



Summary

- @IBInspectable – Expose Custom Properties to Inspector
- @IBDesignable – Live render Custom Objects in Storyboard
- Swift Extensions – Enhance efficiency creatively

“Simple techniques that make Xcode Storyboard more intuitive.”

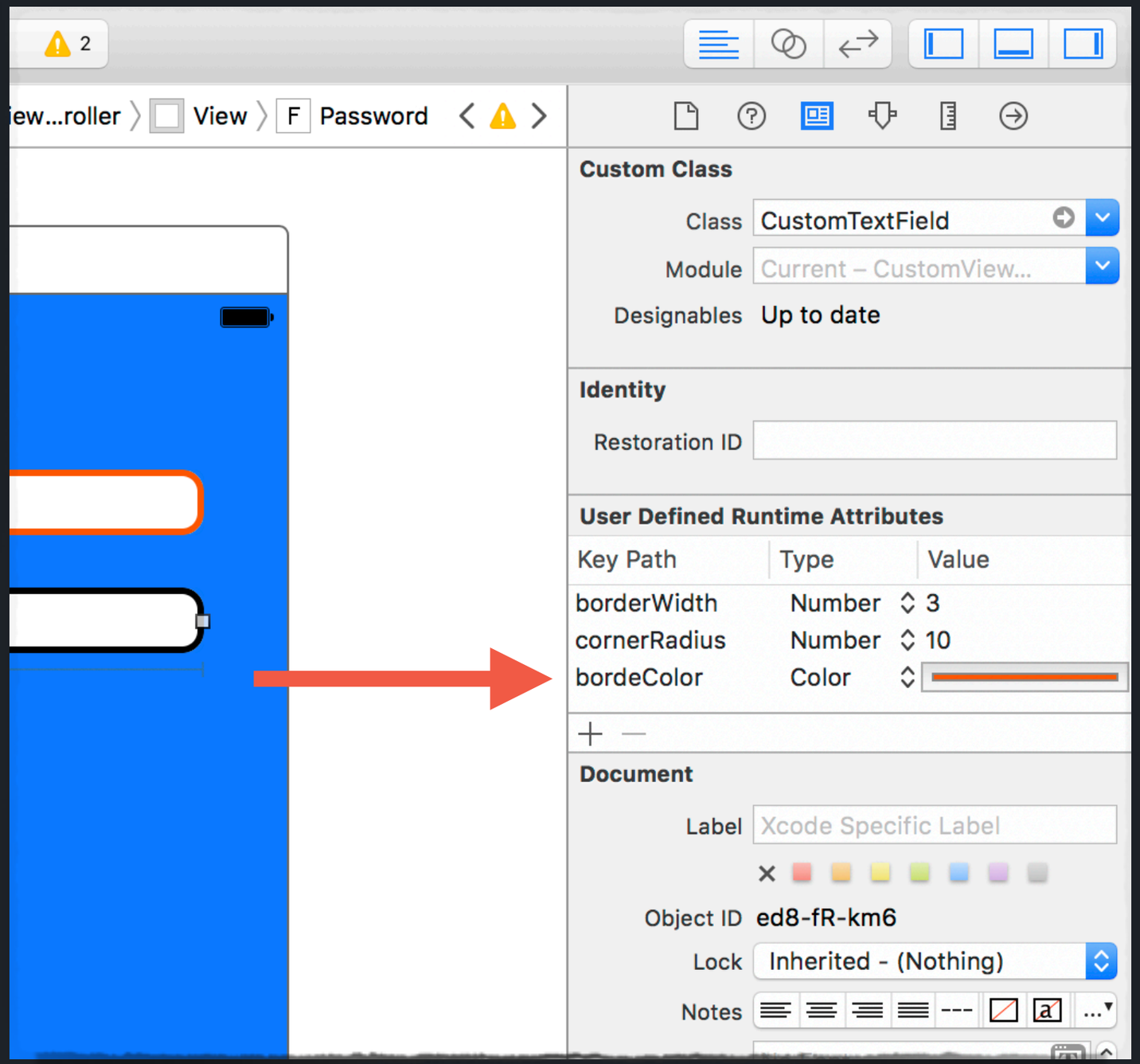
Important notes !

- Crashes
- Unexpected exceptions
- Possible issues
- Quick fix tips

Check “User-Defined Runtime Attributes”

```
2017-02-20 21:29:48.639
CustomView_for_Storyboard[41063:2306462]
Failed to set (bordeColor) user defined
inspected property on
(CustomView_for_Storyboard.CustomTextField):
[<CustomView_for_Storyboard.CustomTextField
0x7fbe3ec147a0> setValue:forUndefinedKey:]:
this class is not key value coding-compliant
for the key bordeColor.
Message from debugger: Terminated due to
signal 15
```

Error message
that contains “Key-Value Coding”
in the console



Check types of properties

```swift
import UIKit

@IBDesignable
class CustomButton: UIButton {

    @IBInspectable
    var borderWidth: CGFloat = 0 {
        didSet {
            layer.borderWidth = borderWidth
        }
    }

    @IBInspectable
    var borderColor: UIColor? {
        didSet {
            layer.borderColor = borderColor?.cgColor
        }
    }

    @IBInspectable
    var cornerRadius: CGFloat = 0 {
        didSet {
            layer.cornerRadius = cornerRadius
            layer.masksToBounds = cornerRadius > 0
        }
    }

}
```

- Custom Class
- Stored property
- Set default value
- Observer “didSet”

```swift
import UIKit

extension UIView {

    @IBInspectable
    var borderWidth: CGFloat {
        get {
            return layer.borderWidth
        }
        set {
            layer.borderWidth = newValue
        }
    }

    @IBInspectable
    var borderColor: UIColor? {
        get {

            if layer.borderColor != nil {
                return UIColor(cgColor: layer.borderColor!)
            } else {
                return nil
            }

        }
        set {
            layer.borderColor = newValue?.cgColor
        }
    }
```

- Swift Extensions
- Computed property
- Default value not needed
- Setter and Getter

Customize Your Intuitive Storyboard with Creativity

Thank You

Useful Resources

Profile

张靖元 Jingyuan “Knight” Zhang

Mobile Designer & Developer

CocoaHeads Shanghai 2017-02-23

Portfolio