

Introduction to iOS development with Swift

Lesson 3



Adrien Humilière
Trainline

adhum+dant@gmail.com



- Collections
- Loops
- Introduction to UIKit
- Auto Layout & Stack Views

Collections



Collection types



Array

Dictionary

Arrays

```
[value1, value2, value3]
```

```
var names: [String] = ["Anne", "Gary", "Keith"]
```

Arrays

```
[value1, value2, value3]
```

```
var names = ["Anne", "Gary", "Keith"]
```

```
var numbers = [1, -3, 50, 72, -95, 115]
```

Arrays

```
[value1, value2, value3]
```

```
var names = ["Anne", "Gary", "Keith"]
```

```
var numbers: [Double] = [1, -3, 50, 72, -95, 115]
```

Arrays – contains

```
let numbers = [4, 5, 6]
if numbers.contains(5) {
    print("There is a 5")
}
```

Arrays types

```
var myArray: [Int] = []
var myArray: Array<Int> = []
var myArray = [Int]()
```

Working with arrays

```
var myArray = [Int](repeating: 0, count: 100)
let count = myArray.count
if myArray.isEmpty { }
```

Working with arrays

```
var names = ["Anne", "Gary", "Keith"]
let firstName = names[0]
print(firstName) // Anne
```

```
names[1] = "Paul"
print(names) // ["Anne", "Paul", « Keith"]
```

Working with arrays

```
var names = ["Amy"]
names.append("Joe")
names += ["Keith", "Jane"]
print(names) // ["Amy", "Joe", "Keith", « Jane"]
```

Working with arrays

```
var names = ["Amy", "Brad", "Chelsea", "Dan"]
names.insert("Bob", at: 0)
print(names) // ["Bob", "Amy", "Brad", "Chelsea", « Dan"]
```

Working with arrays

```
var names = ["Amy", "Brad", "Chelsea", "Dan"]
let chelsea = names.remove(at:2)
let dan = names.removeLast()
print(names) // ["Amy", "Brad"]
```

```
names.removeAll()
print(names) // []
```

Working with arrays

```
var myNewArray = firstArray + secondArray
```

Dictionaries

```
[key1 : value1, key2: value2, key3: value3]
```

```
var scores = ["Richard": 500, "Luke": 400, "Cheryl": 800]
```

Dictionaries

```
var myDictionary = [String: Int]()
var myDictionary = Dictionary<String, Int>()
var myDictionary: [String: Int] = [:]
```

Add/remove/modify a dictionary

```
var scores = ["Richard": 500, "Luke": 400, "Cheryl": 800]  
  
scores["Oli"] = 399  
  
let oldValue = scores.updateValue(100, forKey: "Richard")
```

Add/remove/modify a dictionary

```
var scores = ["Richard": 500, "Luke": 400, "Cheryl": 800]

scores["Oli"] = 399

if let oldValue = scores.updateValue(100, forKey: "Richard") {
    print("Richard's old value was \(oldValue)")
}
```

Add/remove/modify a dictionary

```
var scores = ["Richard": 100, "Luke": 400, "Cheryl": 800]
scores["Richard"] = nil
print(scores) // ["Cheryl": 800, "Luke": 400]

if let oldValue = scores.removeValue(forKey: "Luke") {
    print("Luke's score was \(oldValue) before he stopped playing")
}
print(scores) // ["Cheryl": 800]
```

Accessing a dictionary

```
var scores = {"Richard": 500, "Luke": 400, "Cheryl": 800}

let players = Array(scores.keys) //["Richard", "Luke", "Cheryl"]
let points = Array(scores.values) //[500, 400, 800]

print(myScore)
if let myScore = scores["Luke"] {
  print(myScore)
}
```

Accessing a dictionary

```
var scores = ["Richard": 500, "Luke": 400, "Cheryl": 800]

let players = Array(scores.keys) //["Richard", "Luke", "Cheryl"]
let points = Array(scores.values) //[500, 400, 800]

print(scores["Luke"]) // Optional(400)
if let myScore = scores["Luke"] {
  print(myScore) // 400
}
```

Loops



Loops

for

while

for loops

```
for index in 1...5 {  
    print("This is number \$(index)")  
}
```

```
for _ in 1...5 {  
    print("Hello!")  
}
```

for loops

```
let names = ["Joseph", "Cathy", "Winston"]
for name in names {
    print("Hello \(name)")
}
```

```
for letter in "ABCDEFG".characters {
    print("The letter is \(letter)")
}
```

for loops

```
for (index, letter) in "ABCDEFG".characters.enumerated() {  
    print("\(index): \(letter)")  
}
```

for loops

```
let vehicles = ["unicycle" : 1, "bicycle" : 2, "tricycle" : 3]
for (vehicleName, wheelCount) in vehicles {
    print("A \vehicleName has \wheelCount wheels")
}
```

while loops

```
var numberOfLives = 3

while numberOfLives > 0 {
    playMove()
    updateLivesCount()
}
```

while loops

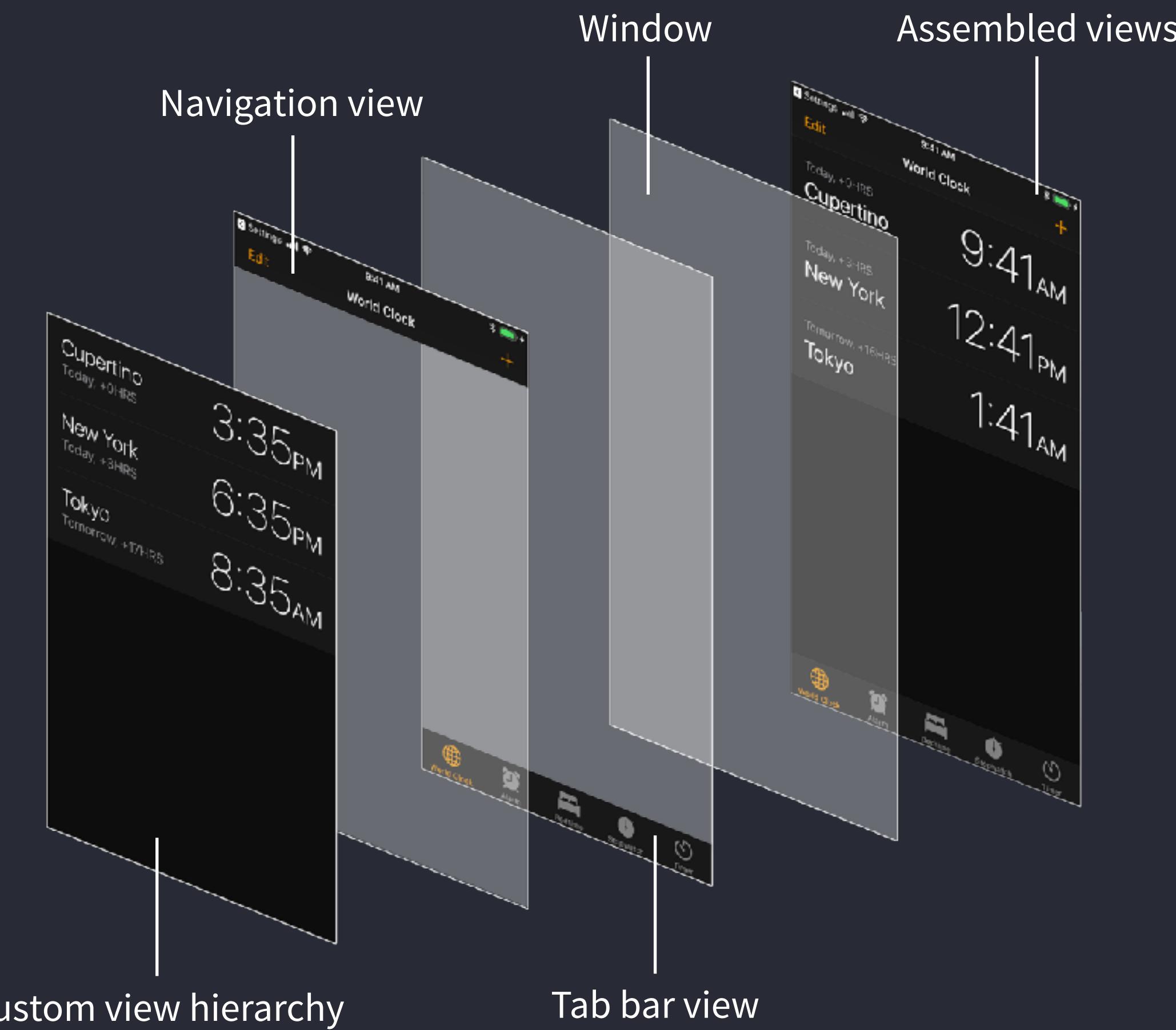
```
var numberOfLives = 3
var stillAlive = true

while stillAlive {
    print("I still have \$(numberOfLives) lives.")
    numberOfLives -= 1
    if numberOfLives == 0 {
        stillAlive = false
    }
}
```

Introduction to UIKit



Common system views



Common system views

Size

Width and height of the view

Position

Position of the view onscreen

Alpha

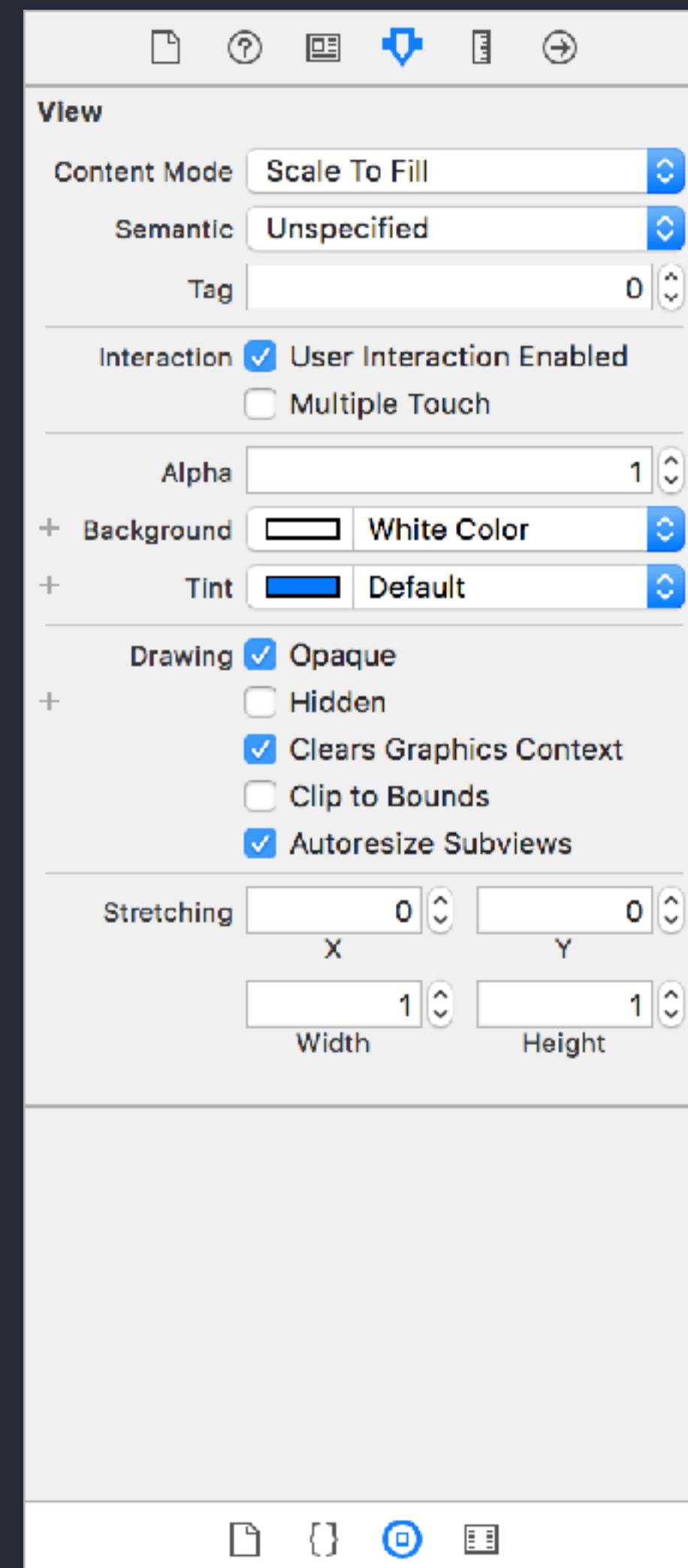
Transparency of the view

Background Color

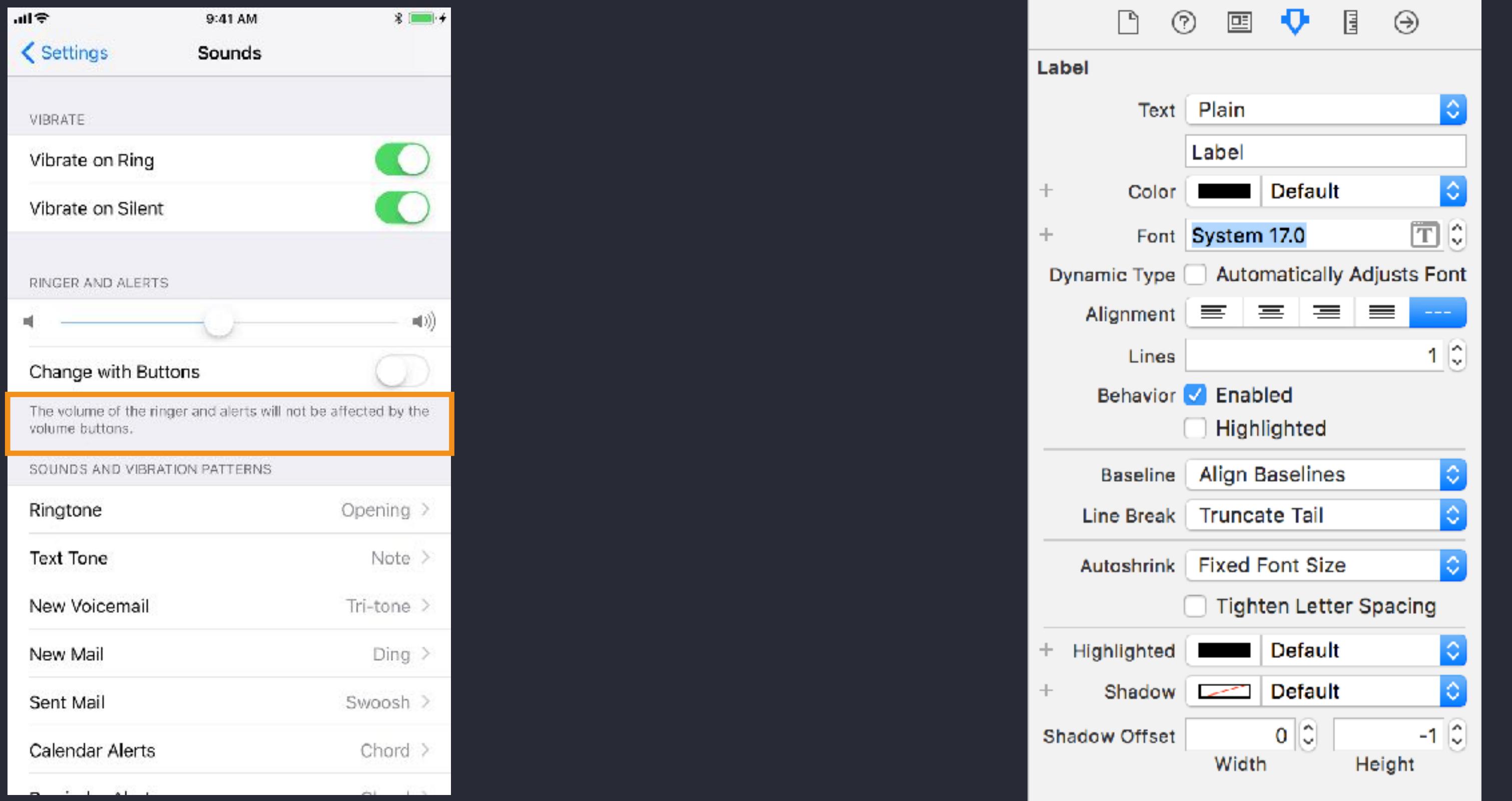
Background color that will be displayed

Tag

Integer that you can use to identify view objects



Label – UILabel



Label – UILabel



Image – UIImageView

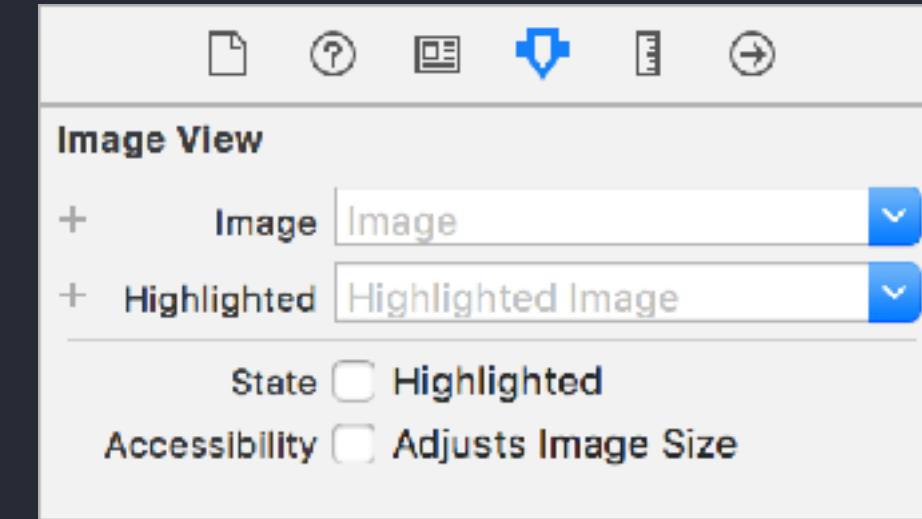
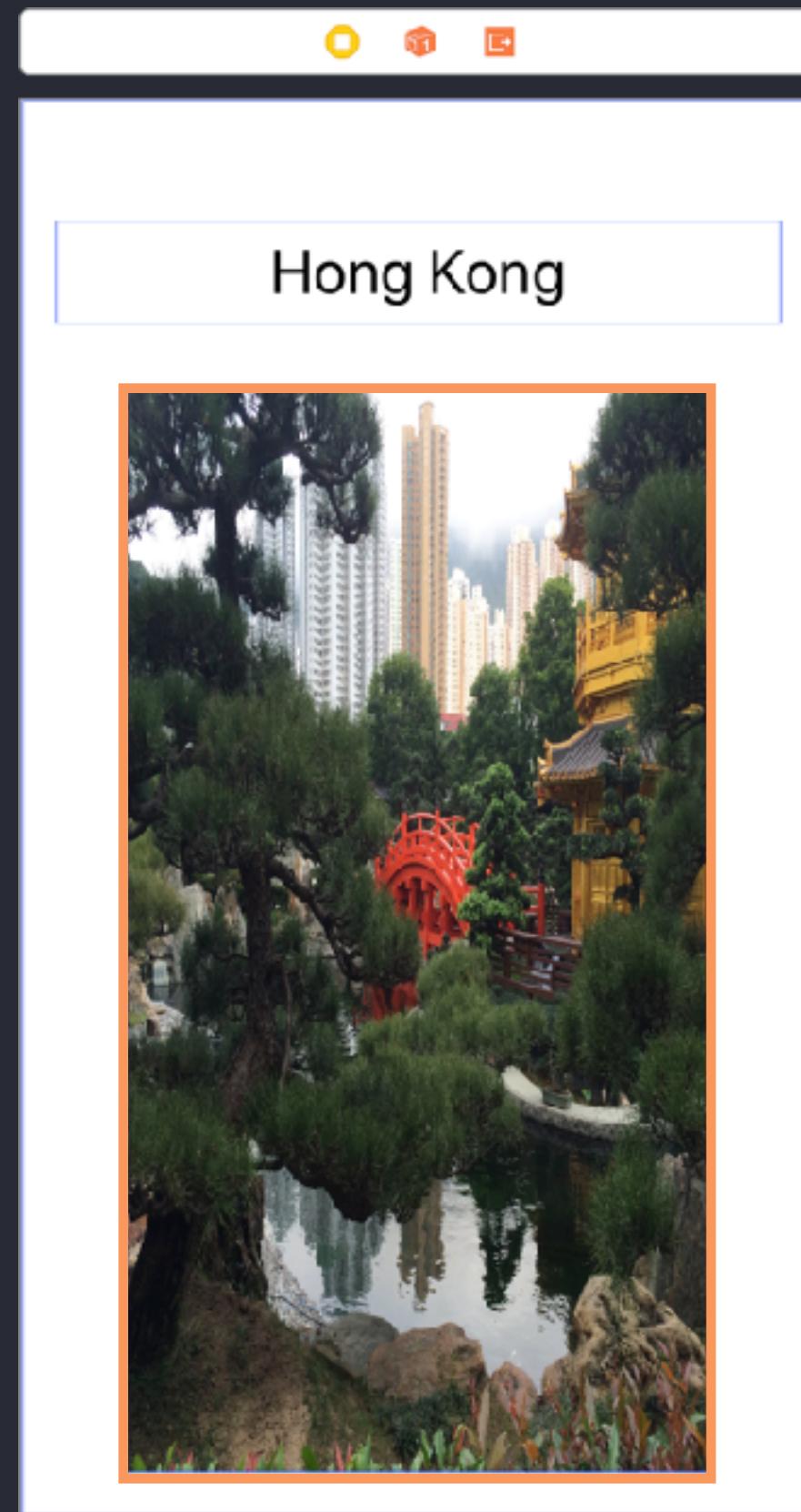
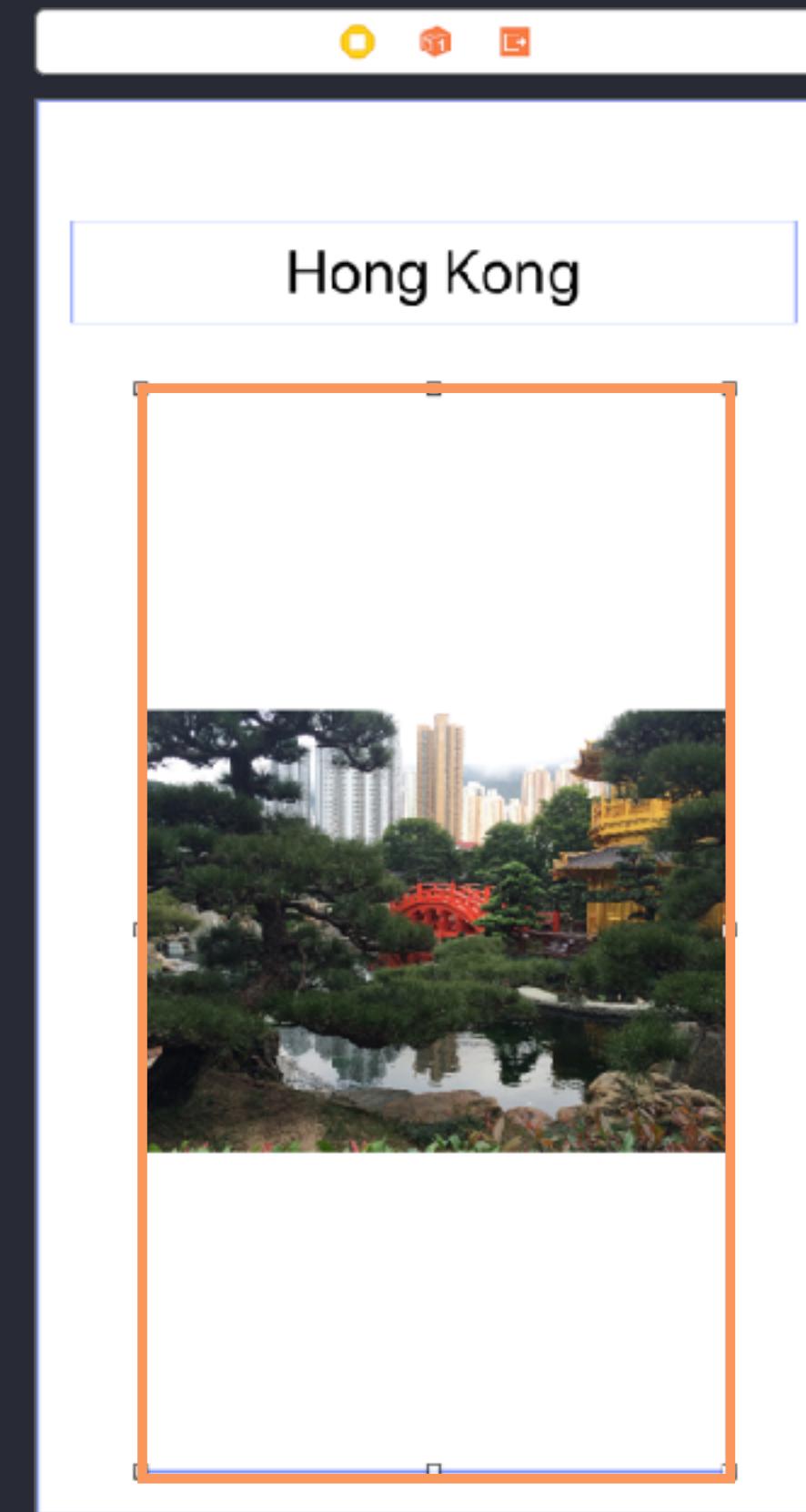


Image – UIImageView



Scale To Fill

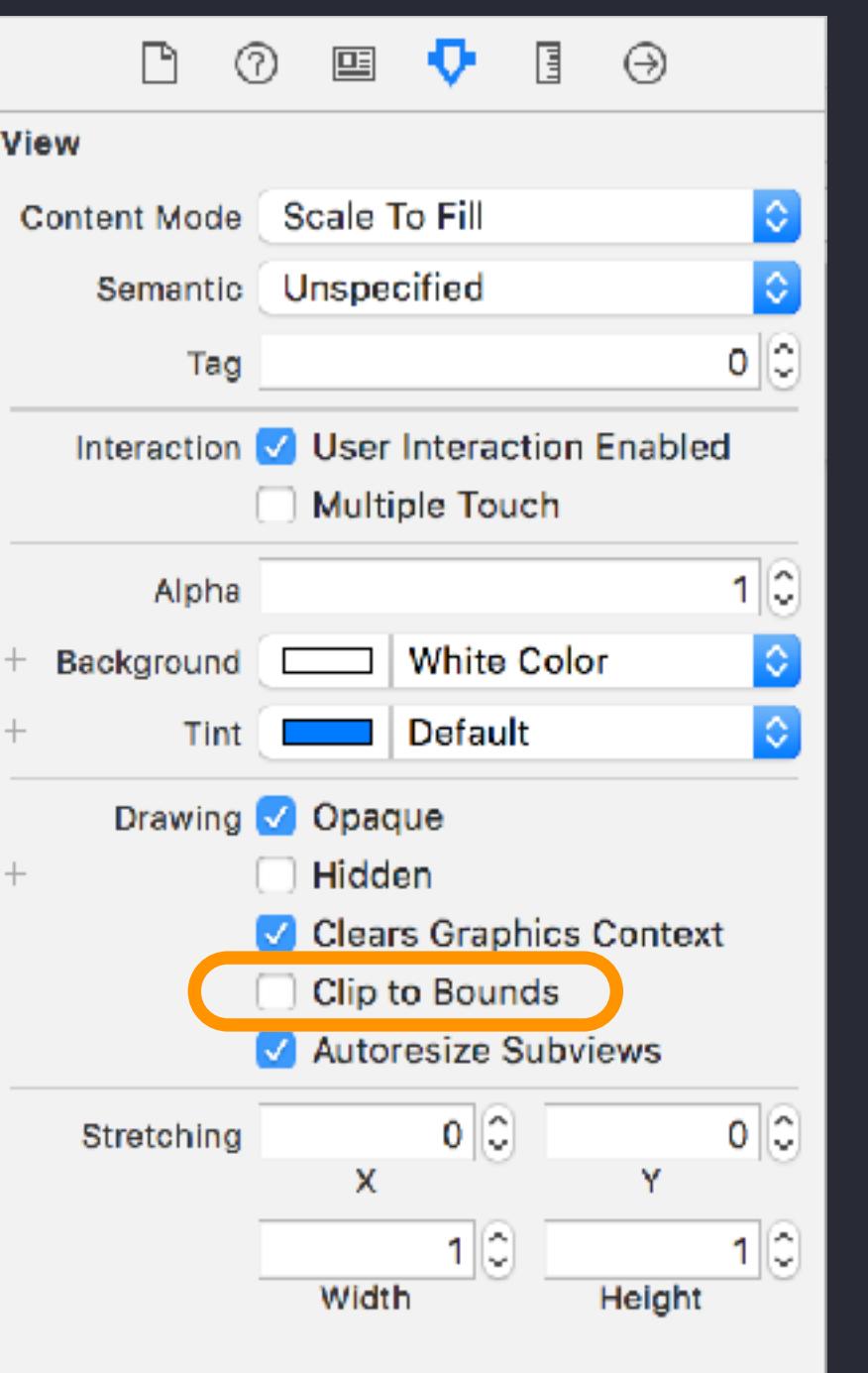


Aspect Fit

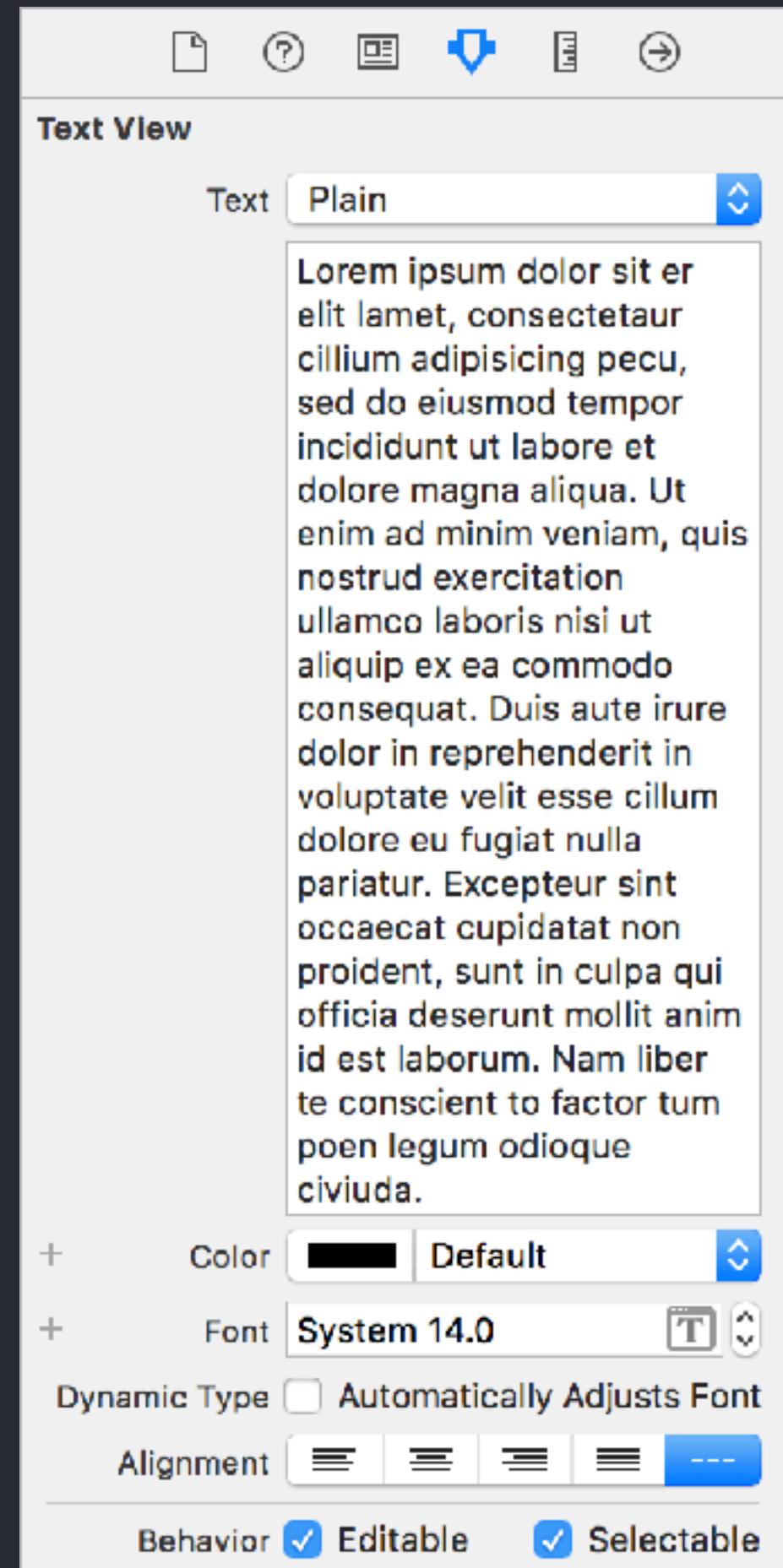
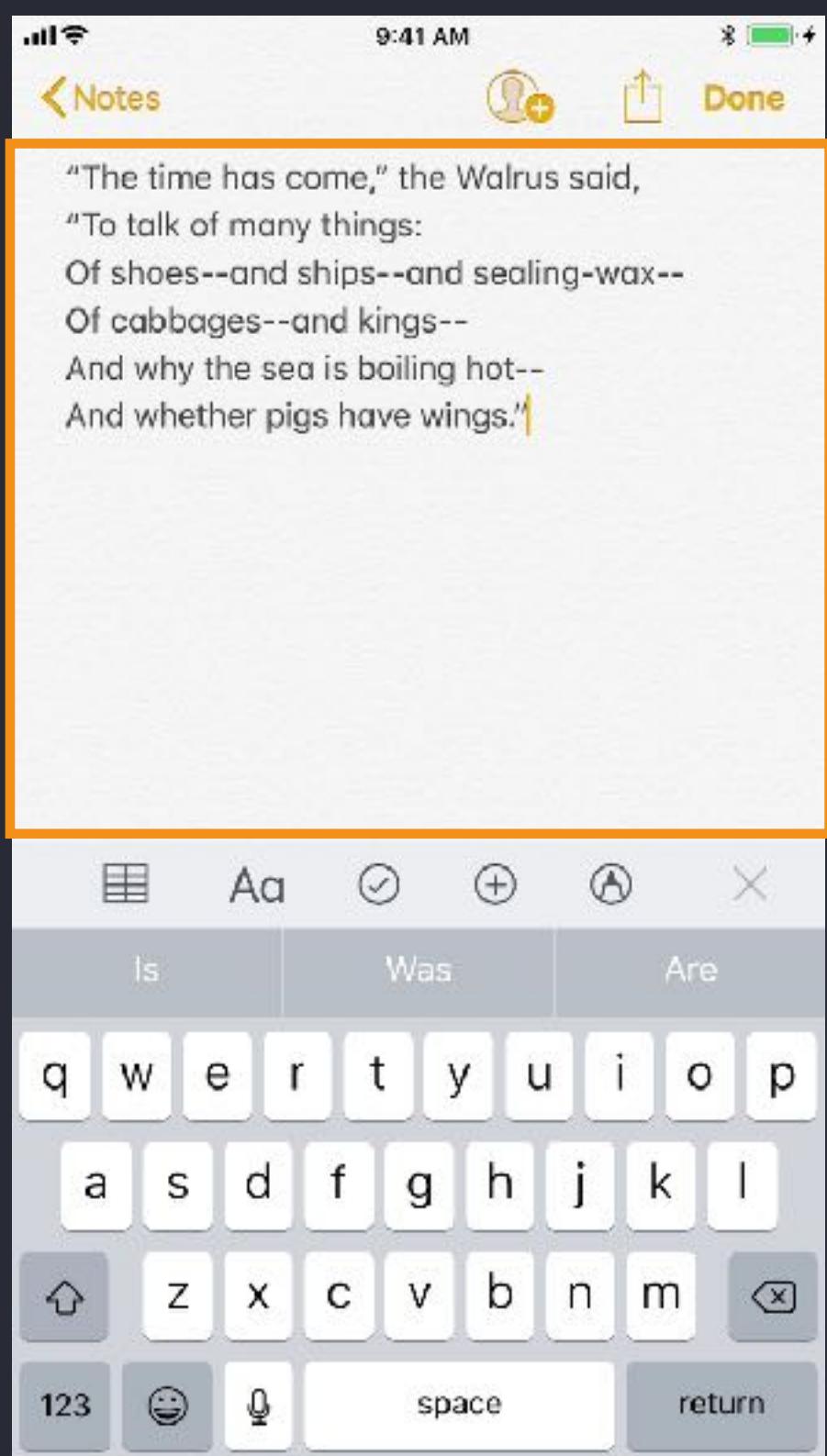


Aspect Fill

Label – UILabel



Text – UITextView



ScrollView – UIScrollView

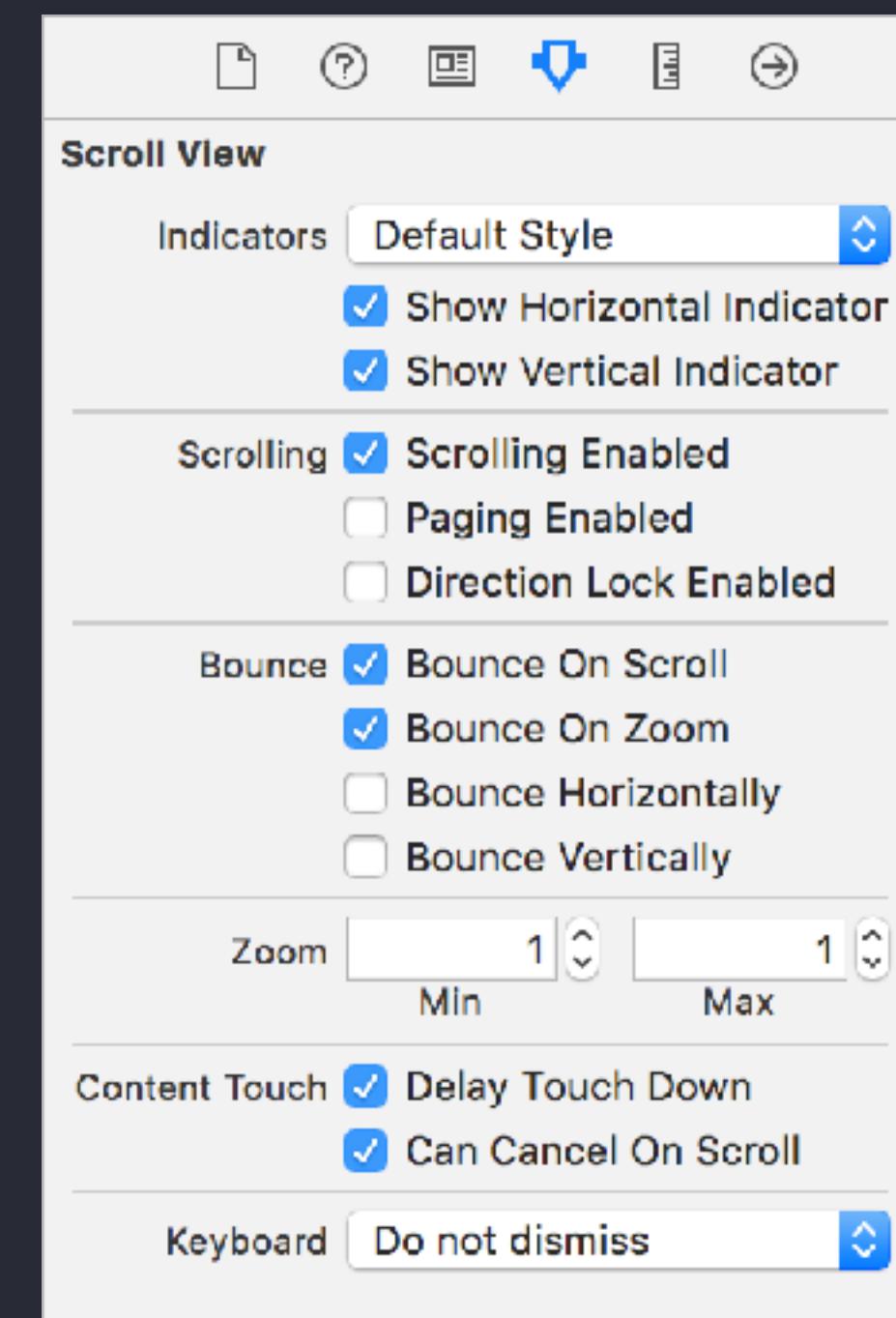
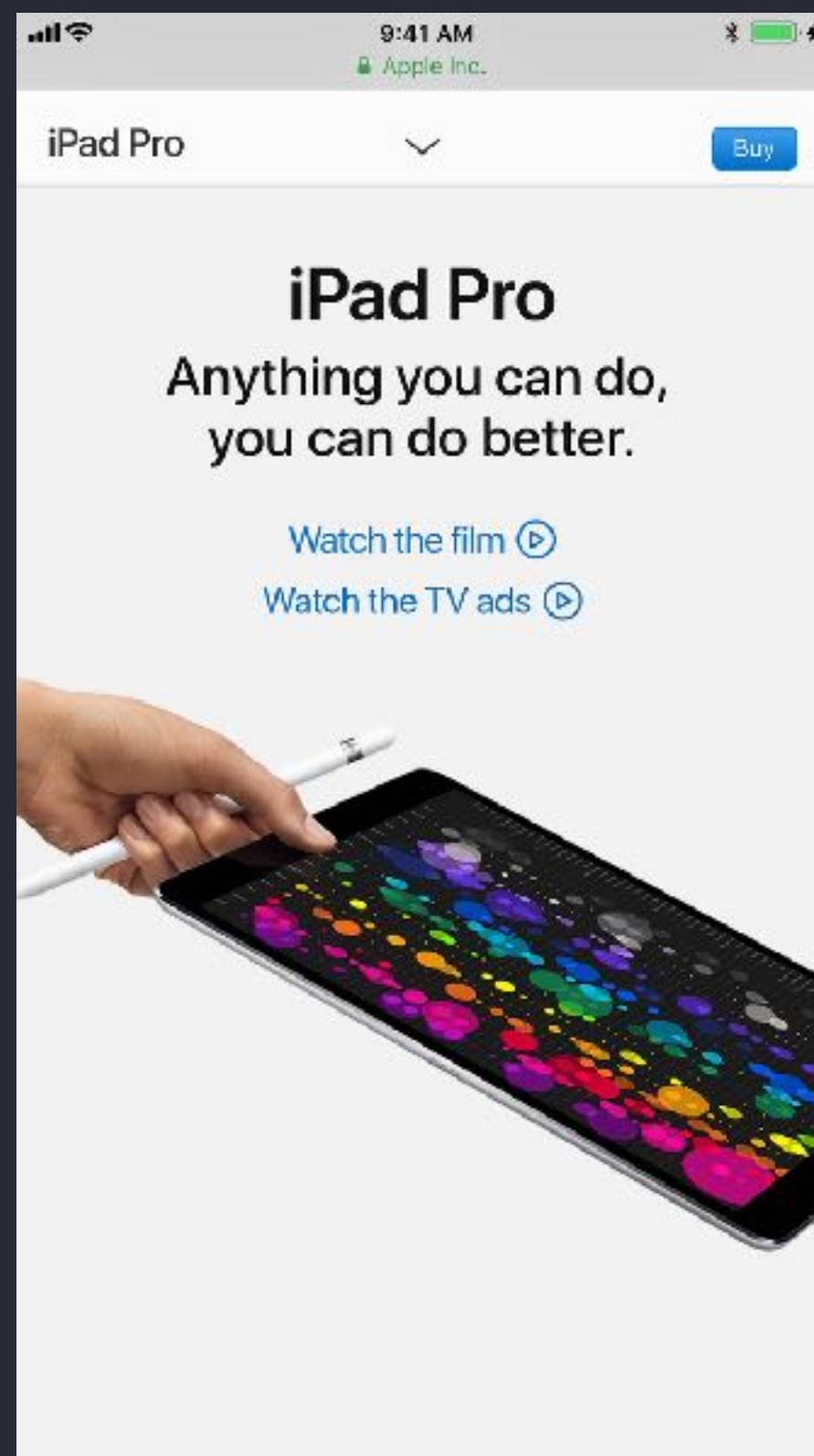
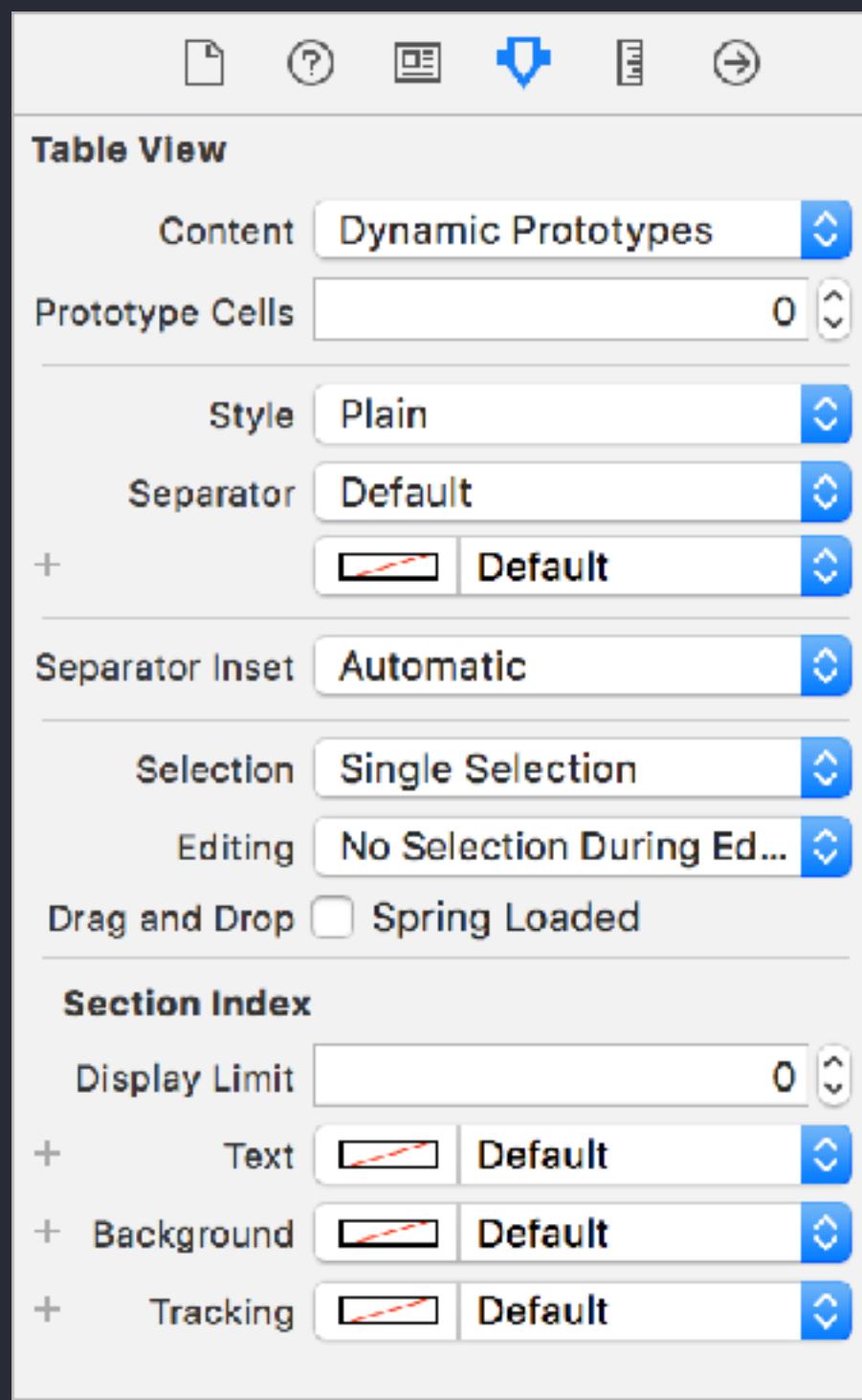
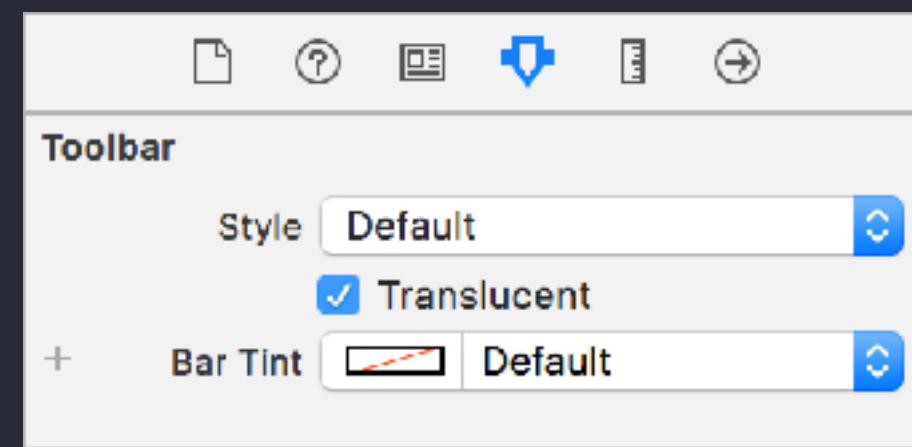
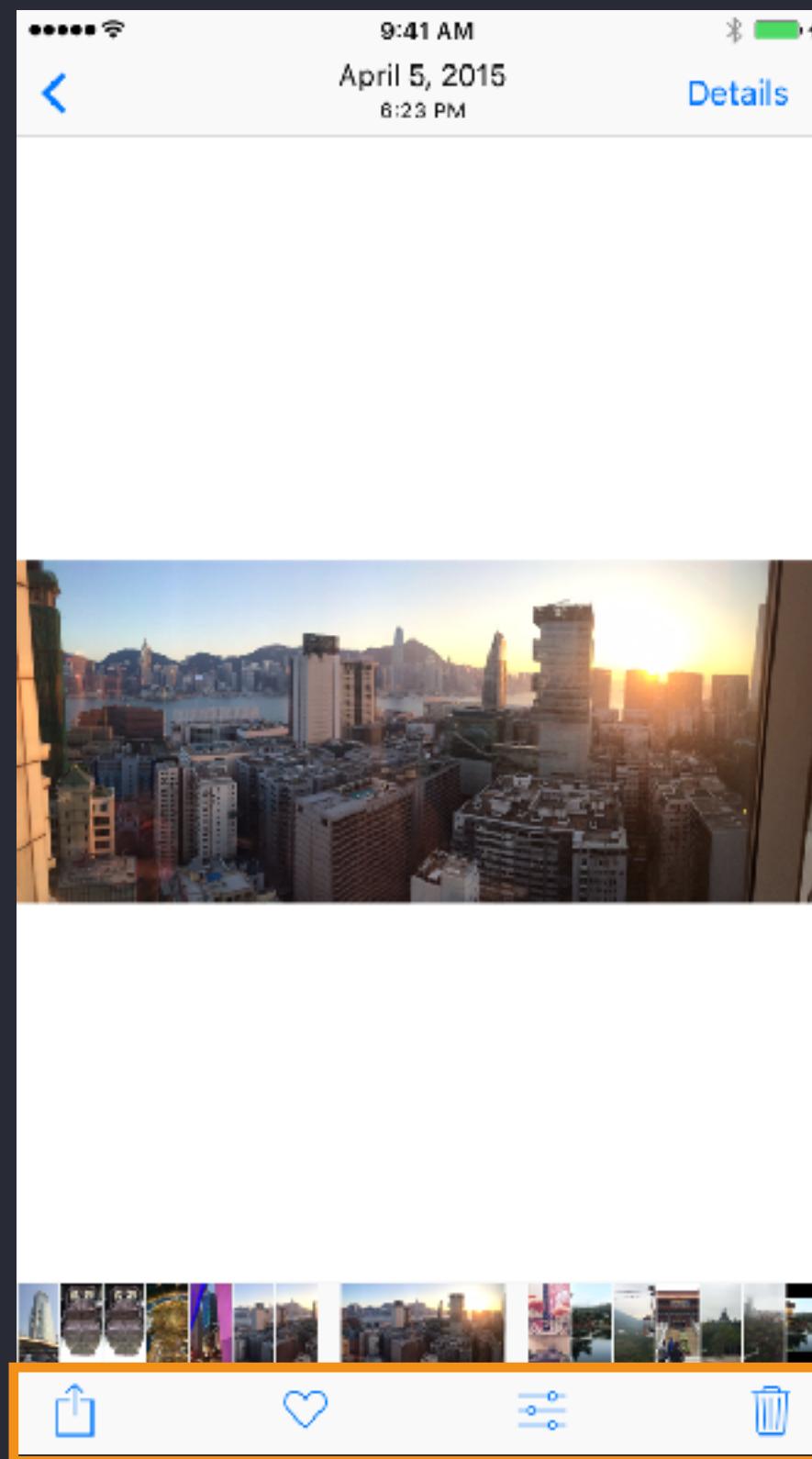


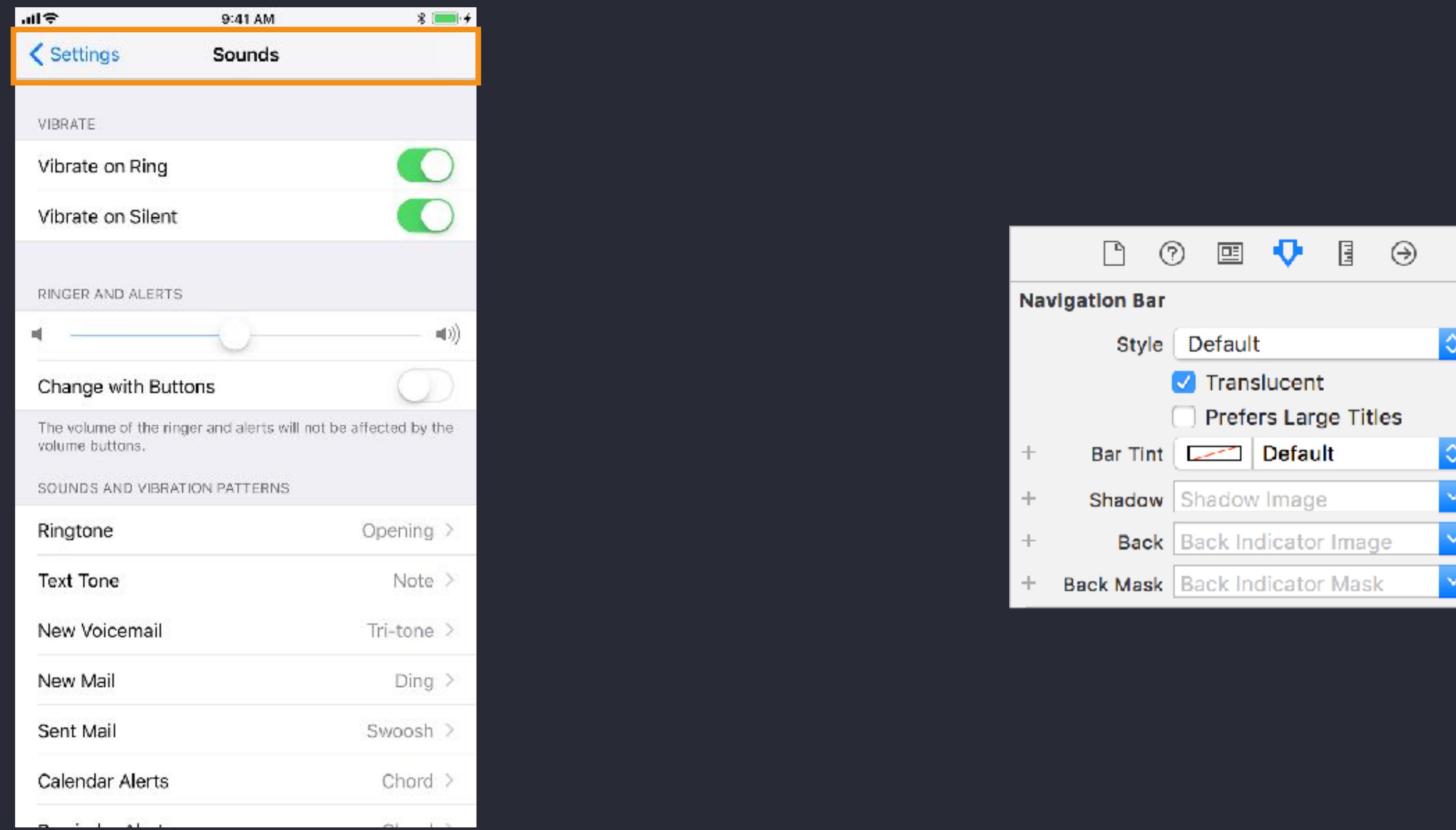
Table – UITableView



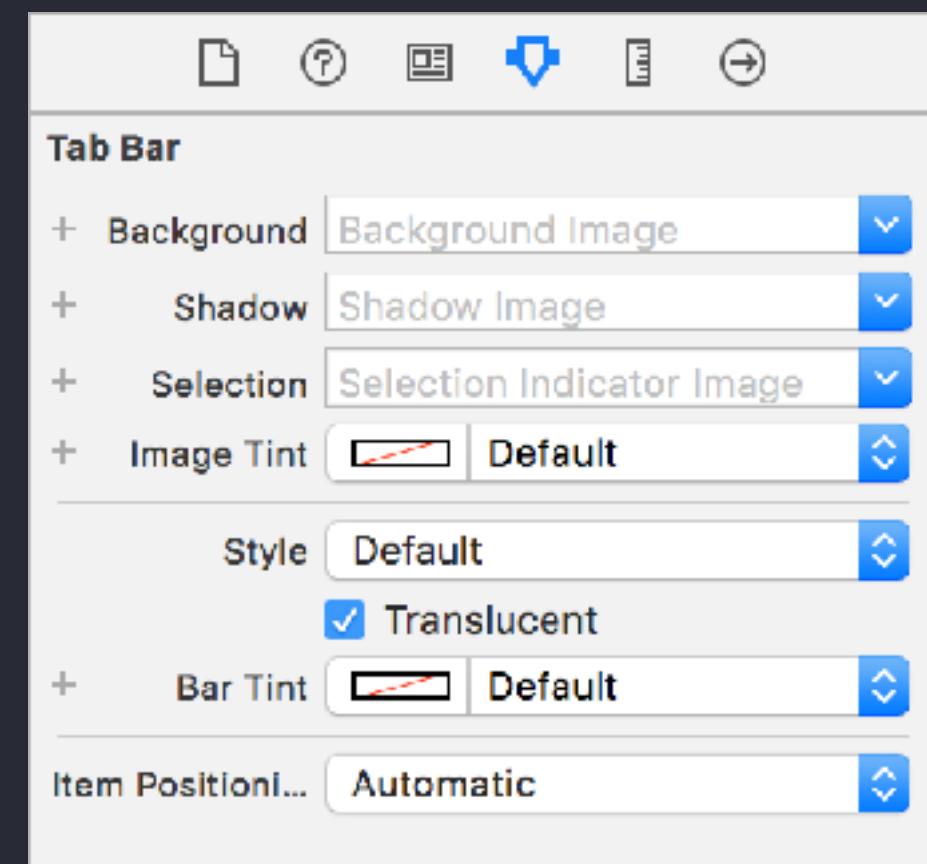
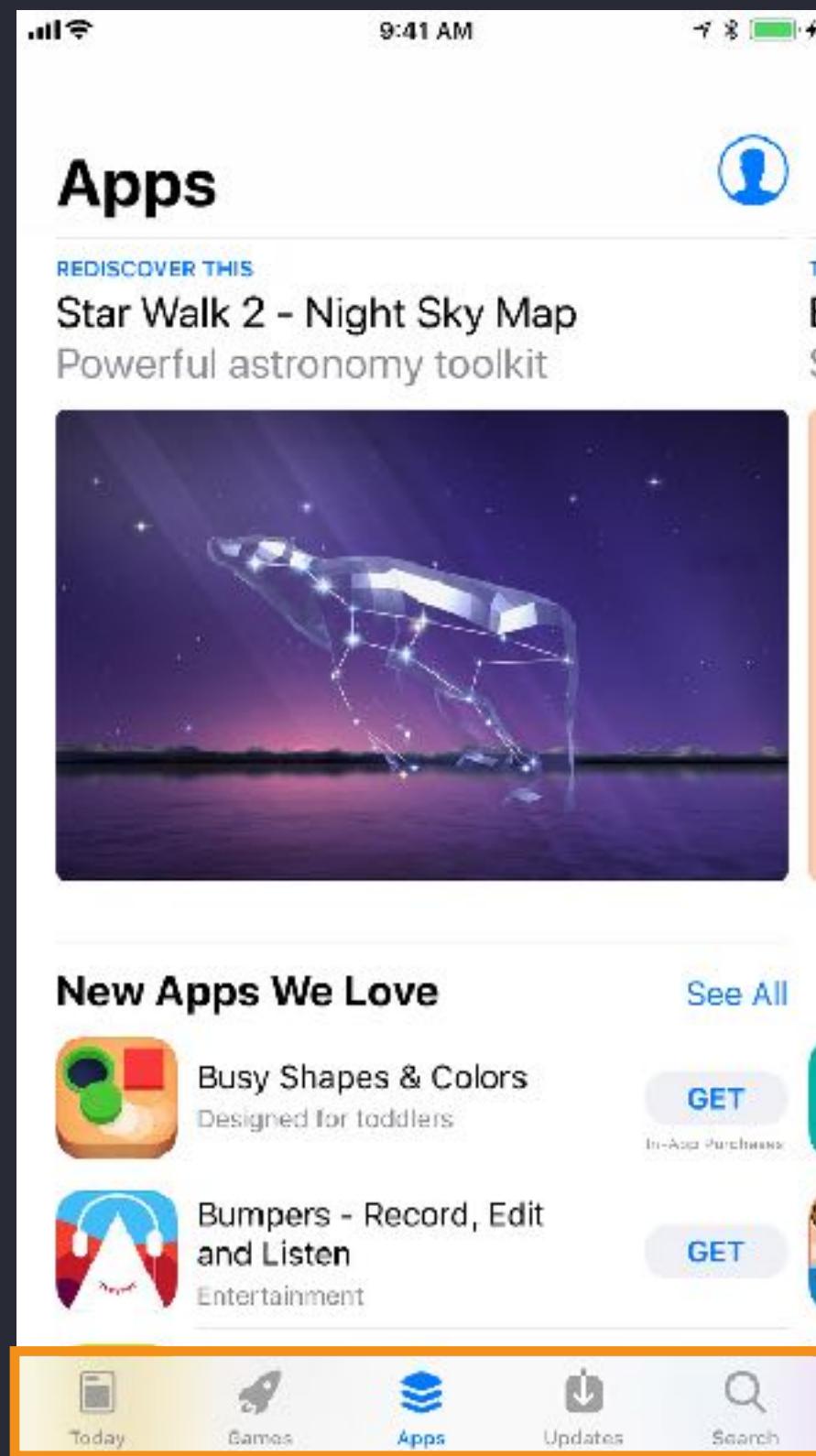
Toolbars – UIToolbar



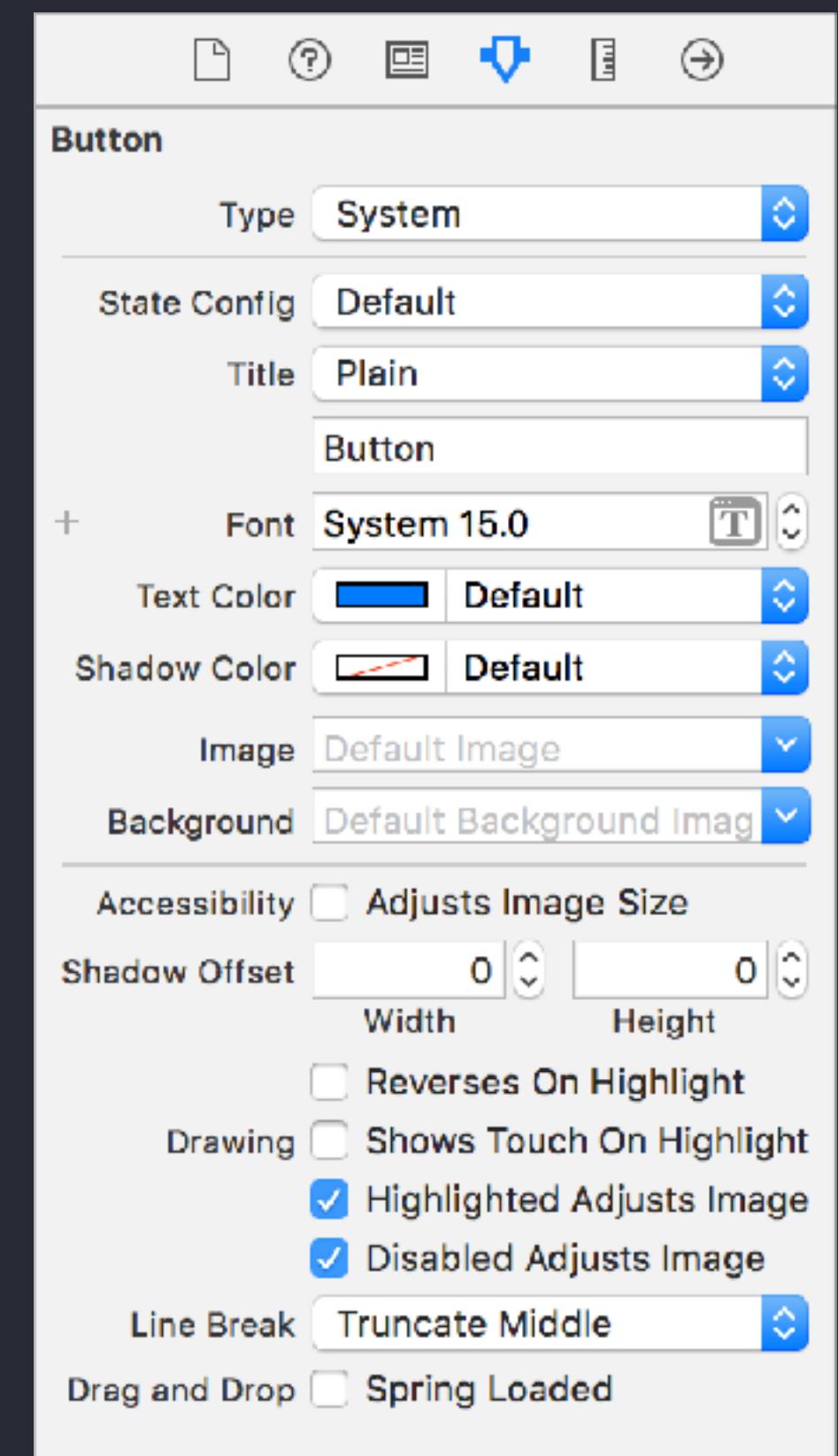
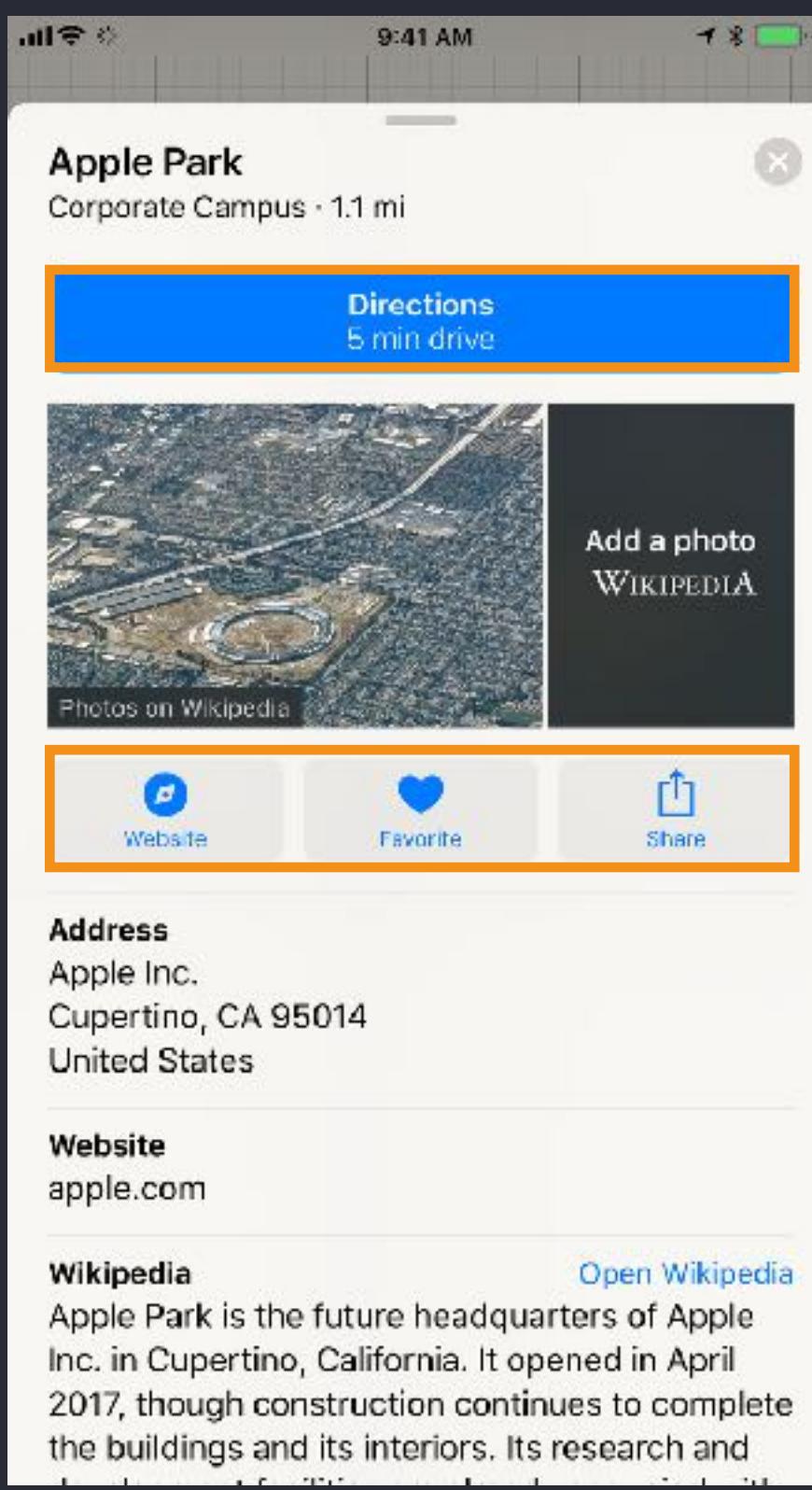
Navigation bar - UINavigationBar



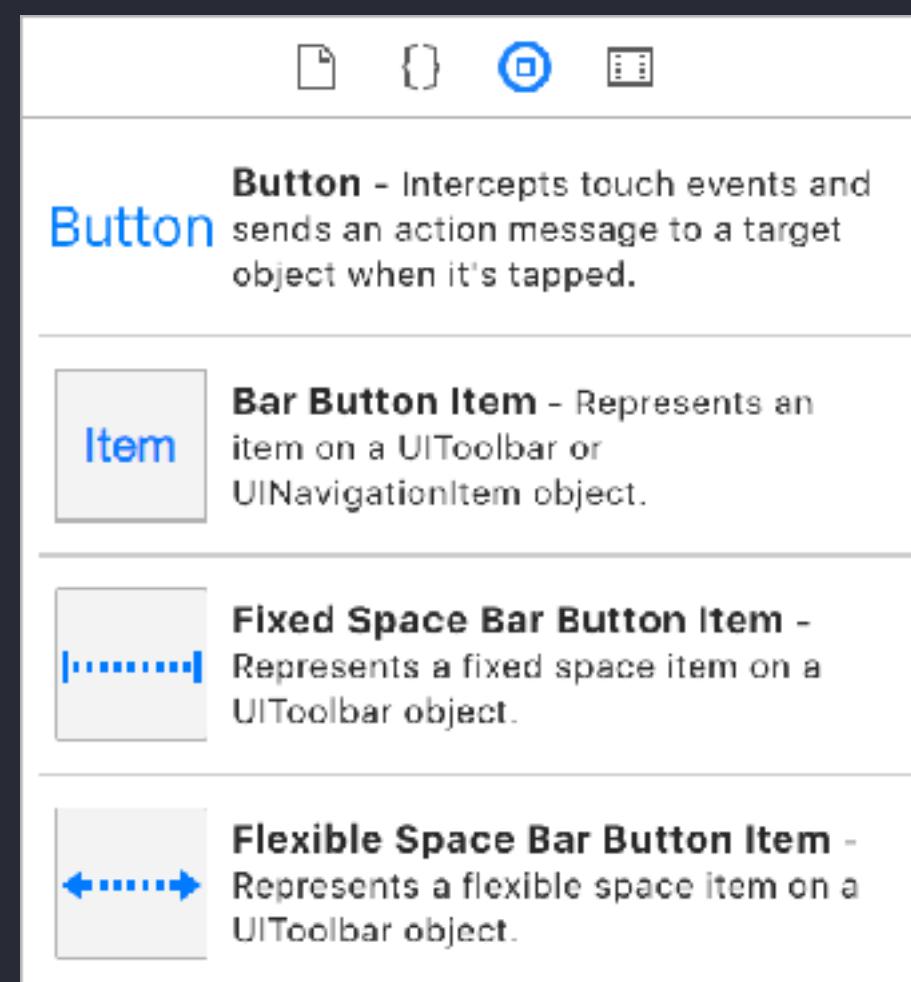
Tabs – UITabBar



Buttons – UIButton

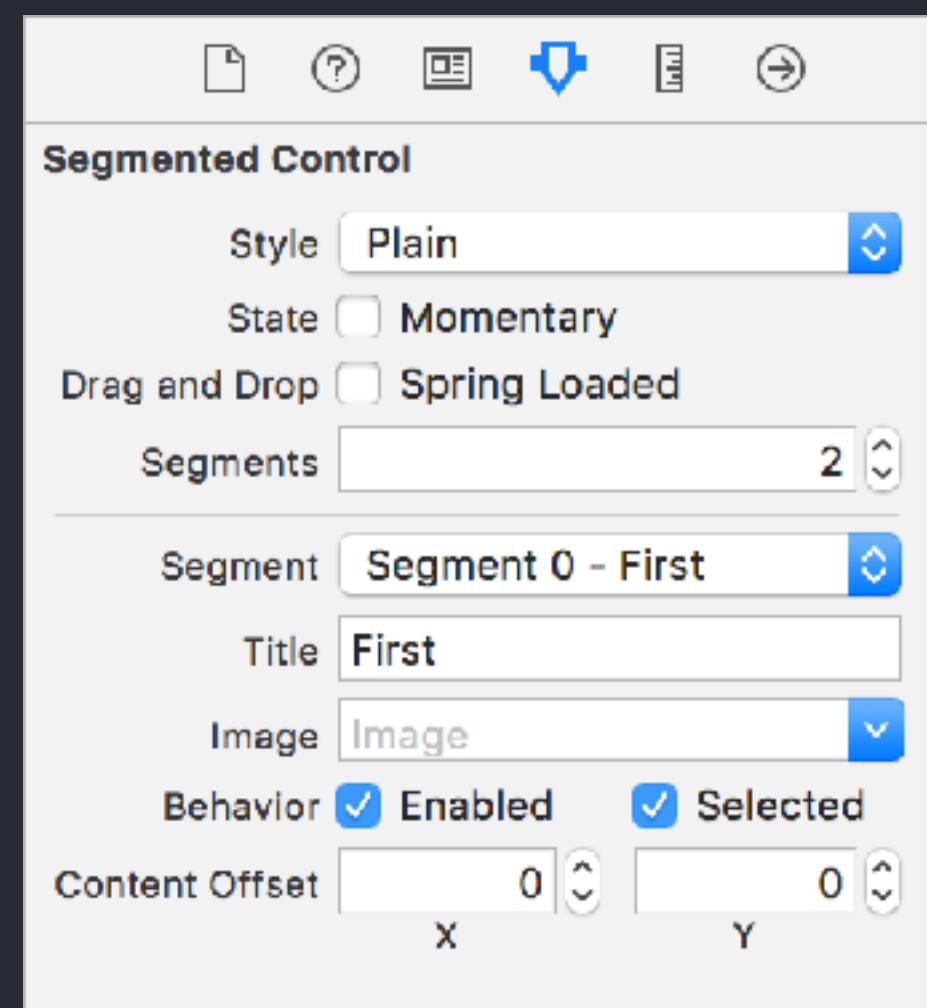
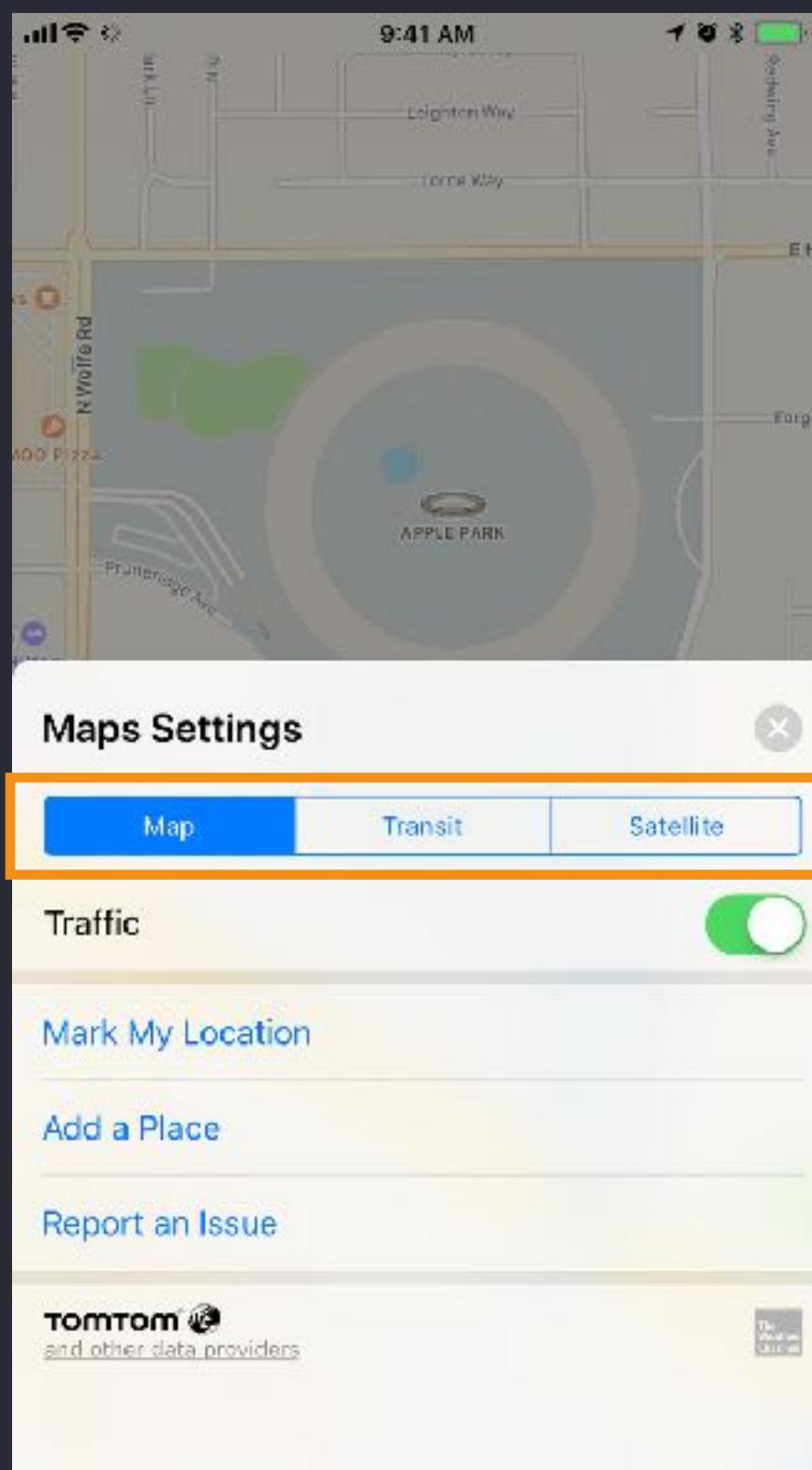


Buttons – UIButton

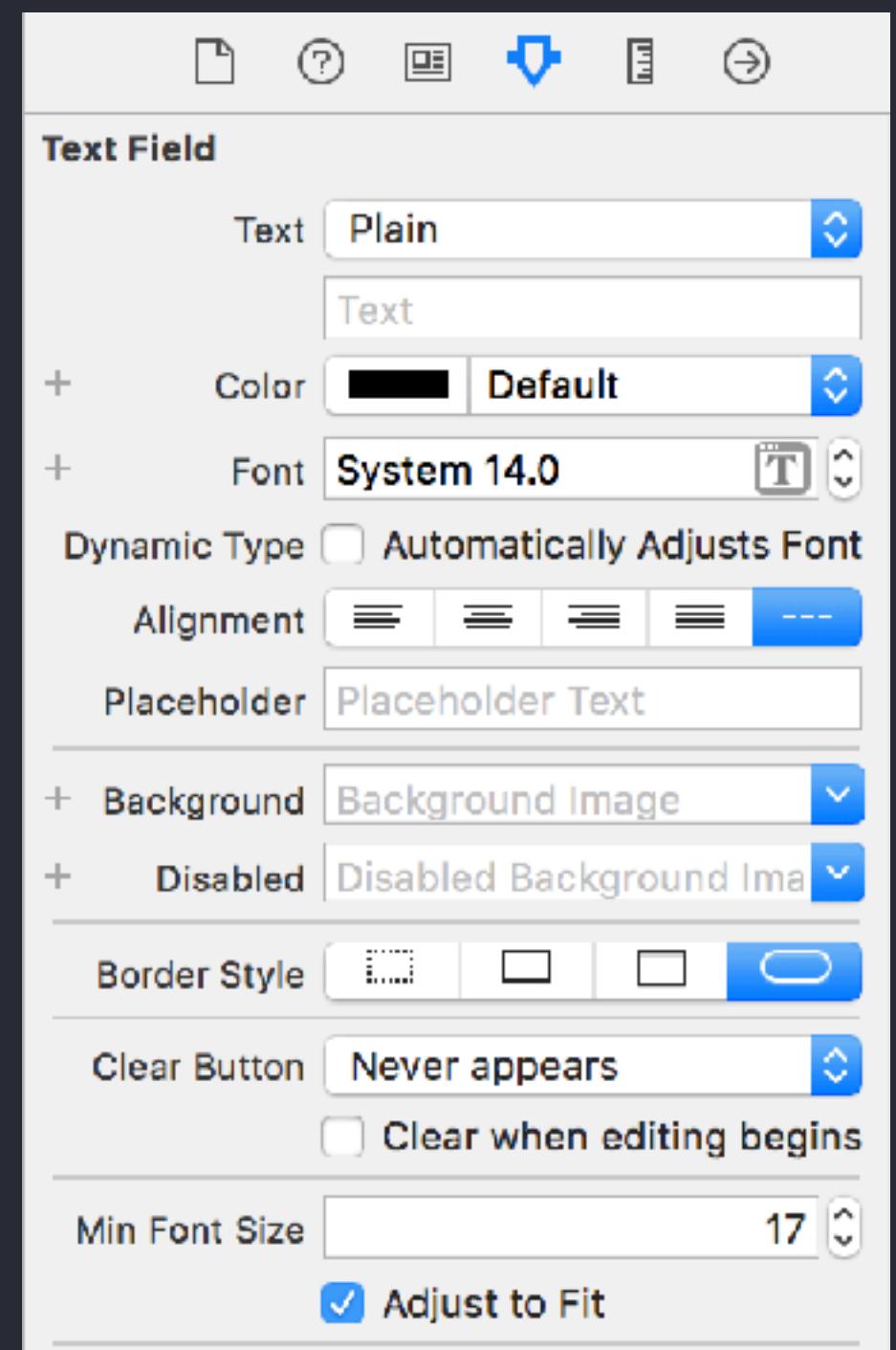
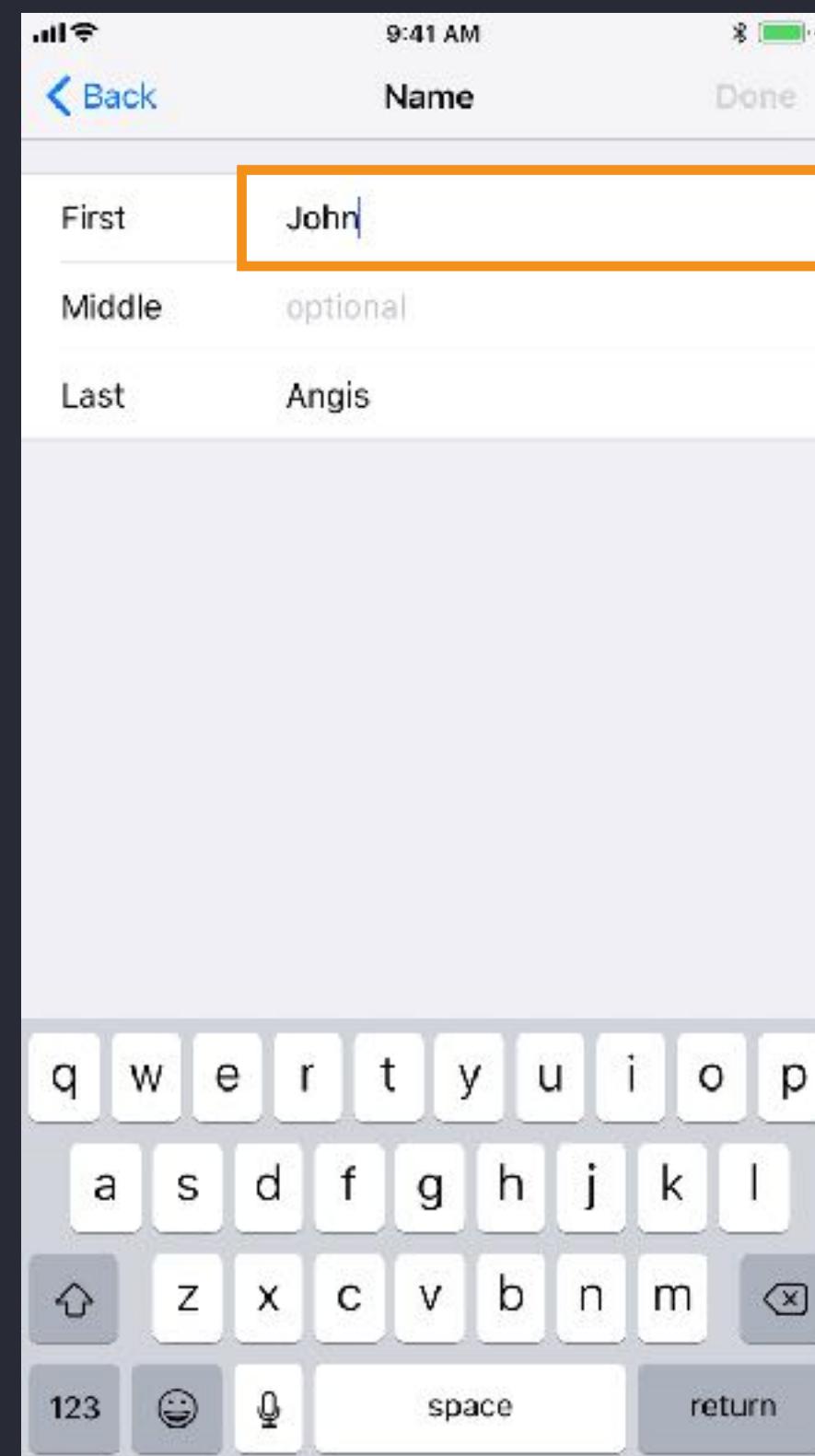


```
@IBAction func buttonTapped(_ sender: Any)
    // Code to respond to button
}
```

Segmented control – UISegmentedControl



Text fields – UITextField



Text fields – UITextField

Text

Text Field - Displays editable text and sends an action message to a target object when Return is tapped.

```
@IBAction func keyboardReturnKeyTapped(_  
sender: UITextField) {  
    if let text = sender.text {  
        print(text)  
    }  
}
```

UITextField

Placeholder Text

Text displayed when the text field is empty

Text

Text displayed by the text field

Capitalization

How the keyboard deals with capitalization

Correction

Enables or disables autocorrect

Keyboard

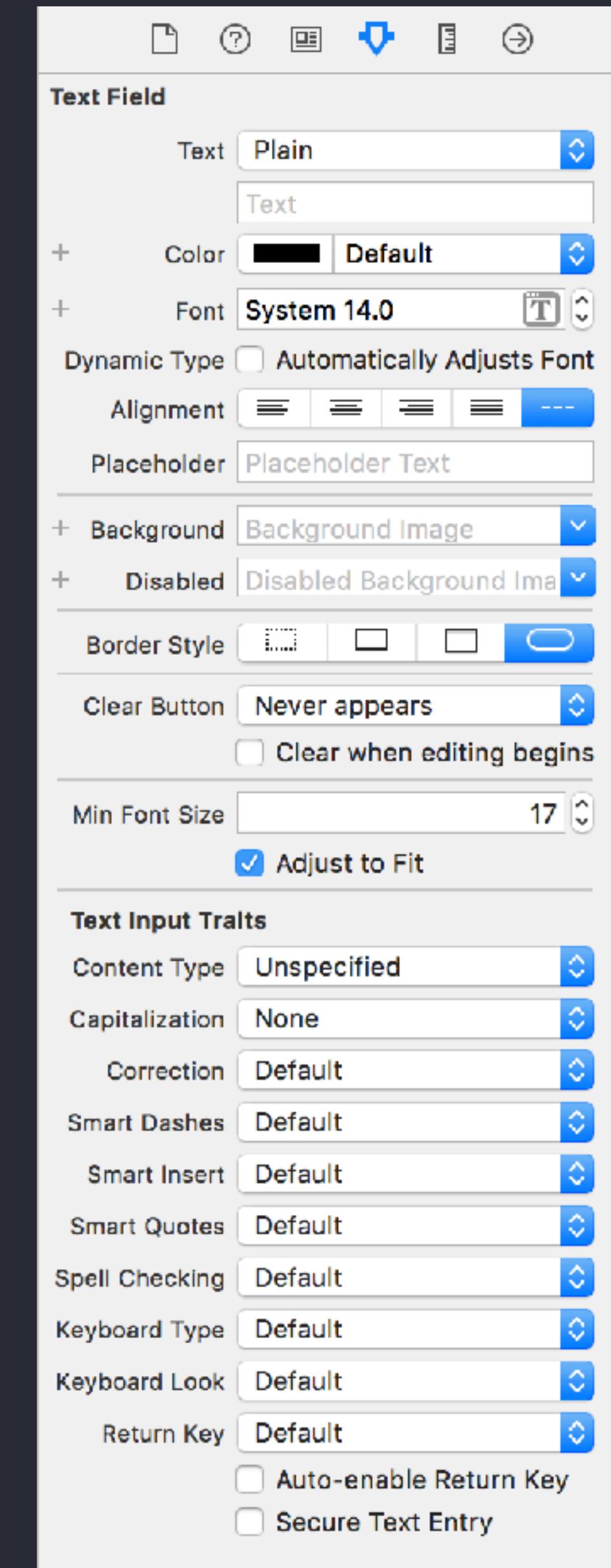
Which keyboard is displayed—for example, email, web, or default

Return Key

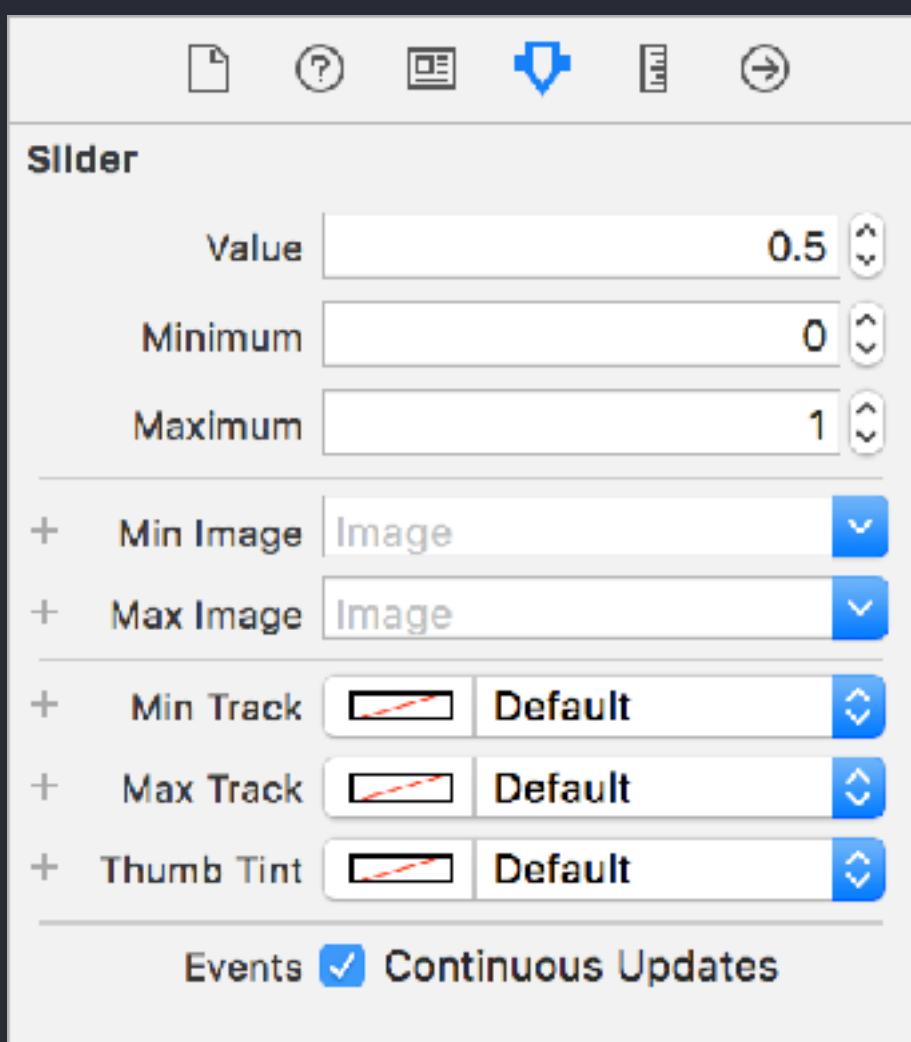
Text on the return key

Secure

Specific text fields that don't display their contents, commonly used for passwords



Sliders – UISlider



Sliders – UISlider

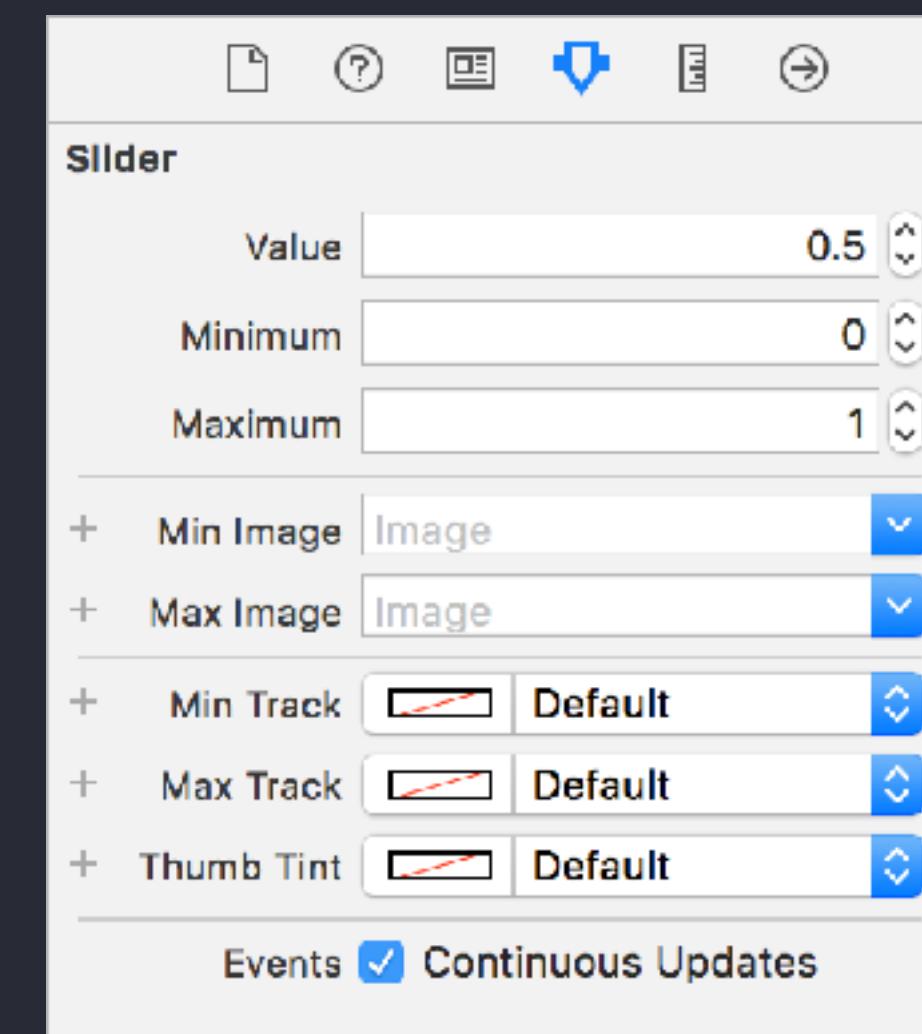


Slider - Displays a continuous range of values and allows the selection of a single value.

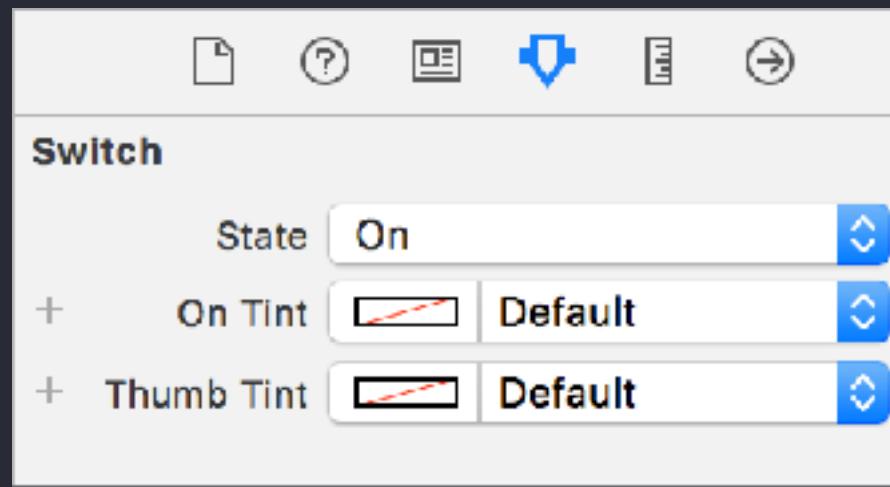
```
@IBAction func sliderValueChanged(_  
sender: UISlider) {  
    print(sender.value)  
}
```

Sliders – UISlider

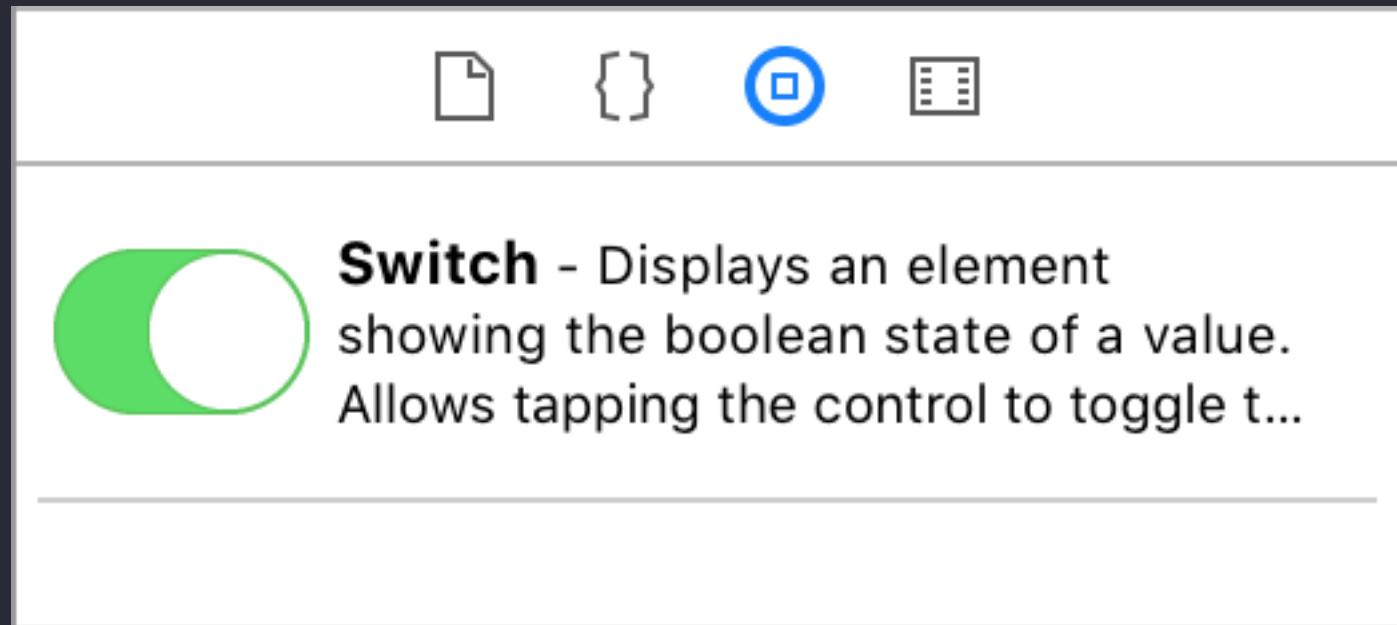
Minimum Value	Lowest number value the slider may represent
Maximum Value	Highest number value the slider may represent
Current	Starting number value the slider will represent
Min Image	Optional image on the minimum end of the slider
Max Image	Optional image on the maximum end of the slider



Switches – UISwitch

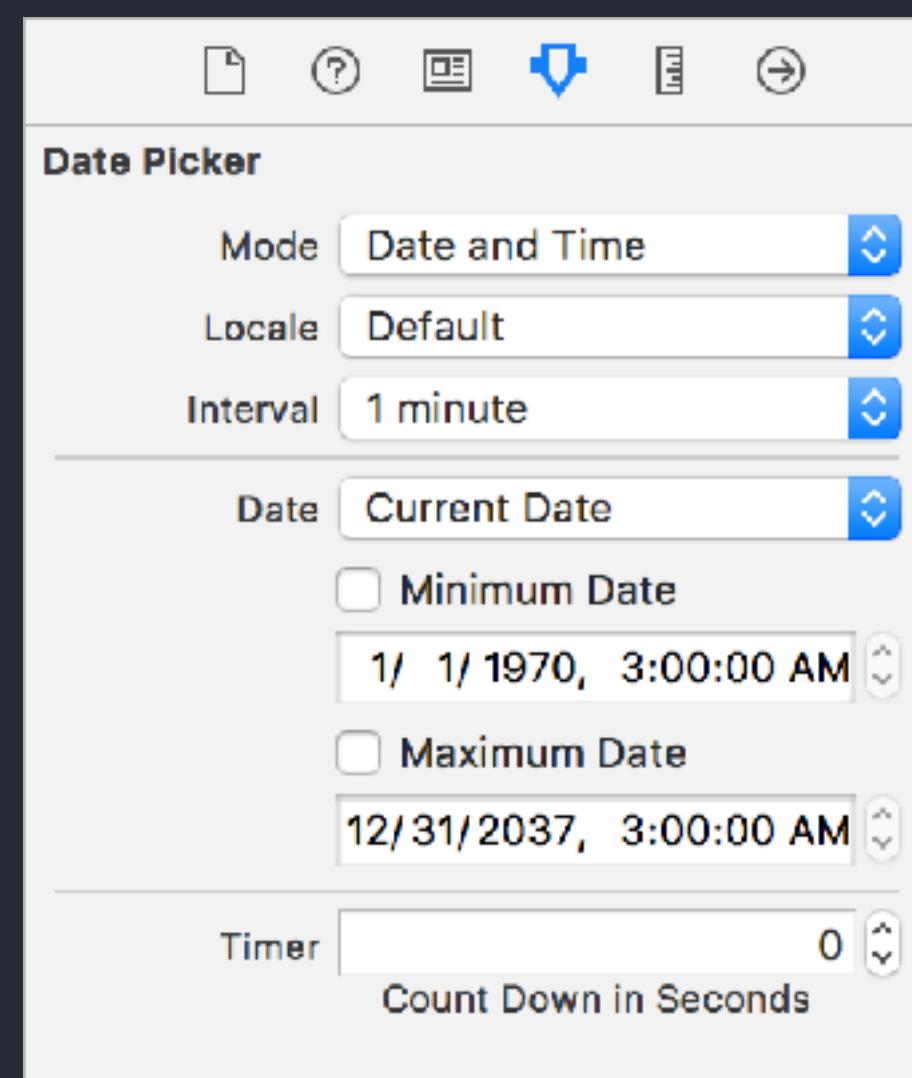
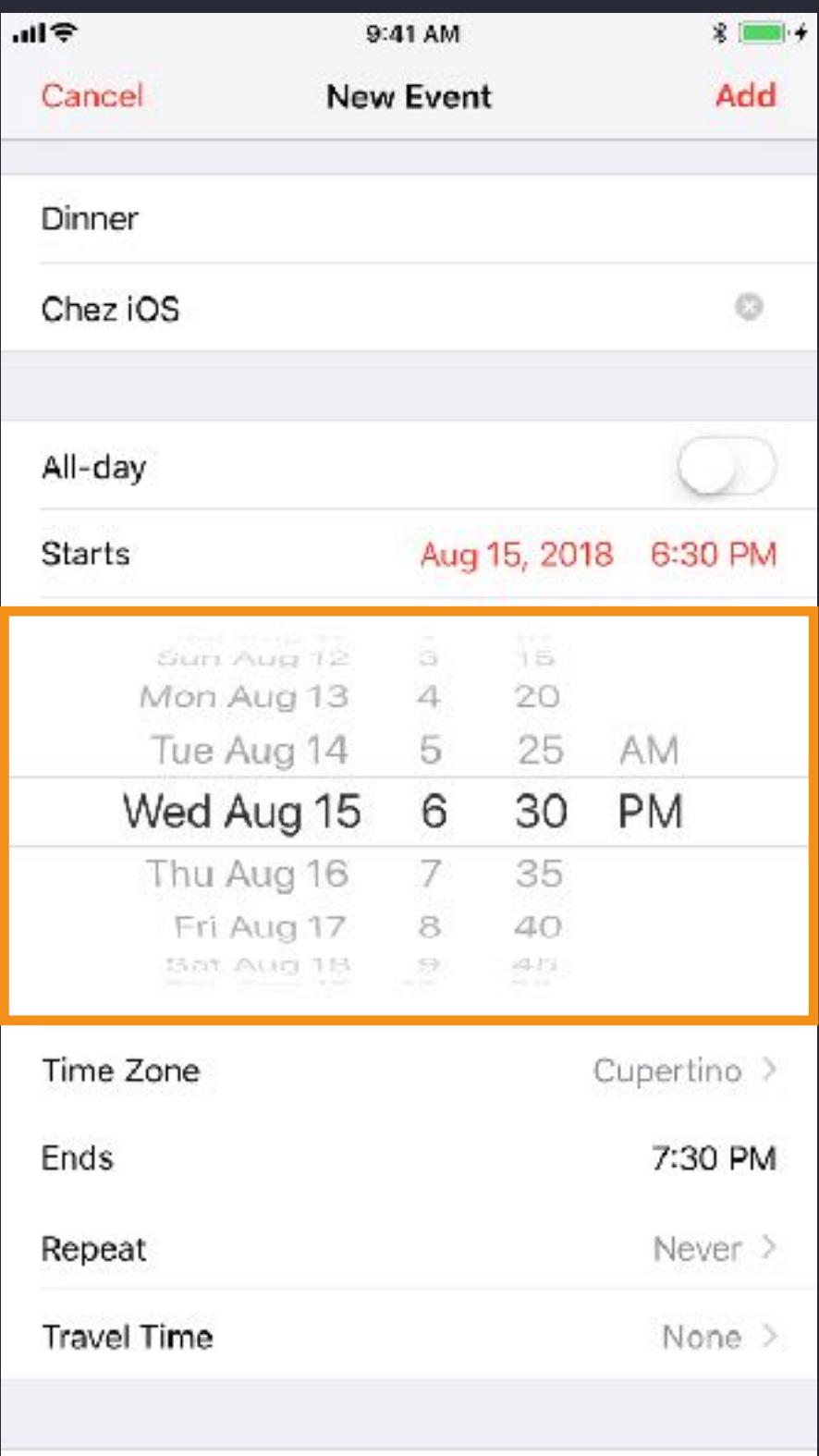


Switches – UISwitch



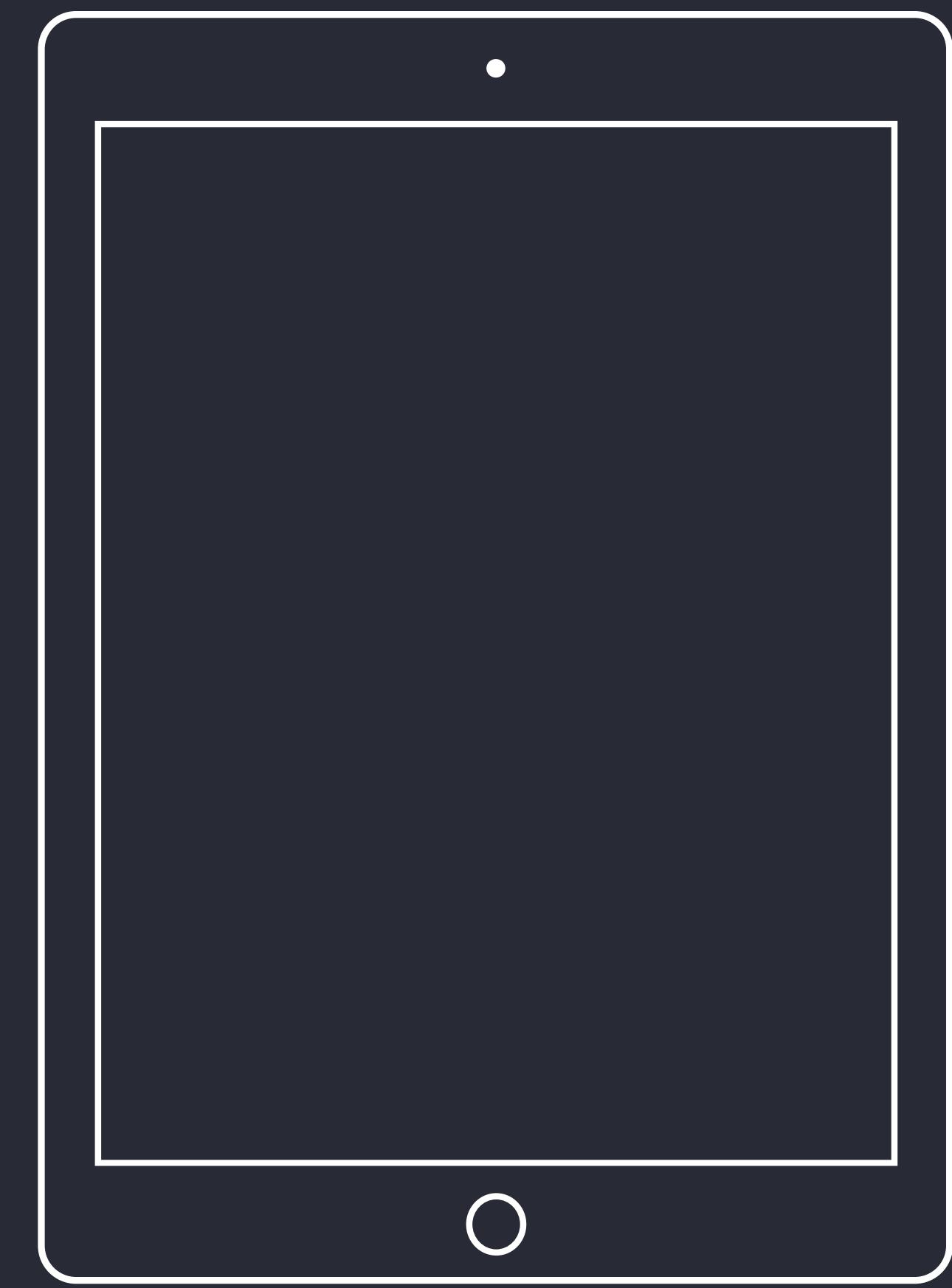
```
@IBAction func switchToggled(_ sender:  
UISwitch) {  
    if sender.isOn {  
        print("The switch is on!")  
    } else {  
        print("The switch is off.")  
    }  
}
```

Date pickers – UIDatePicker

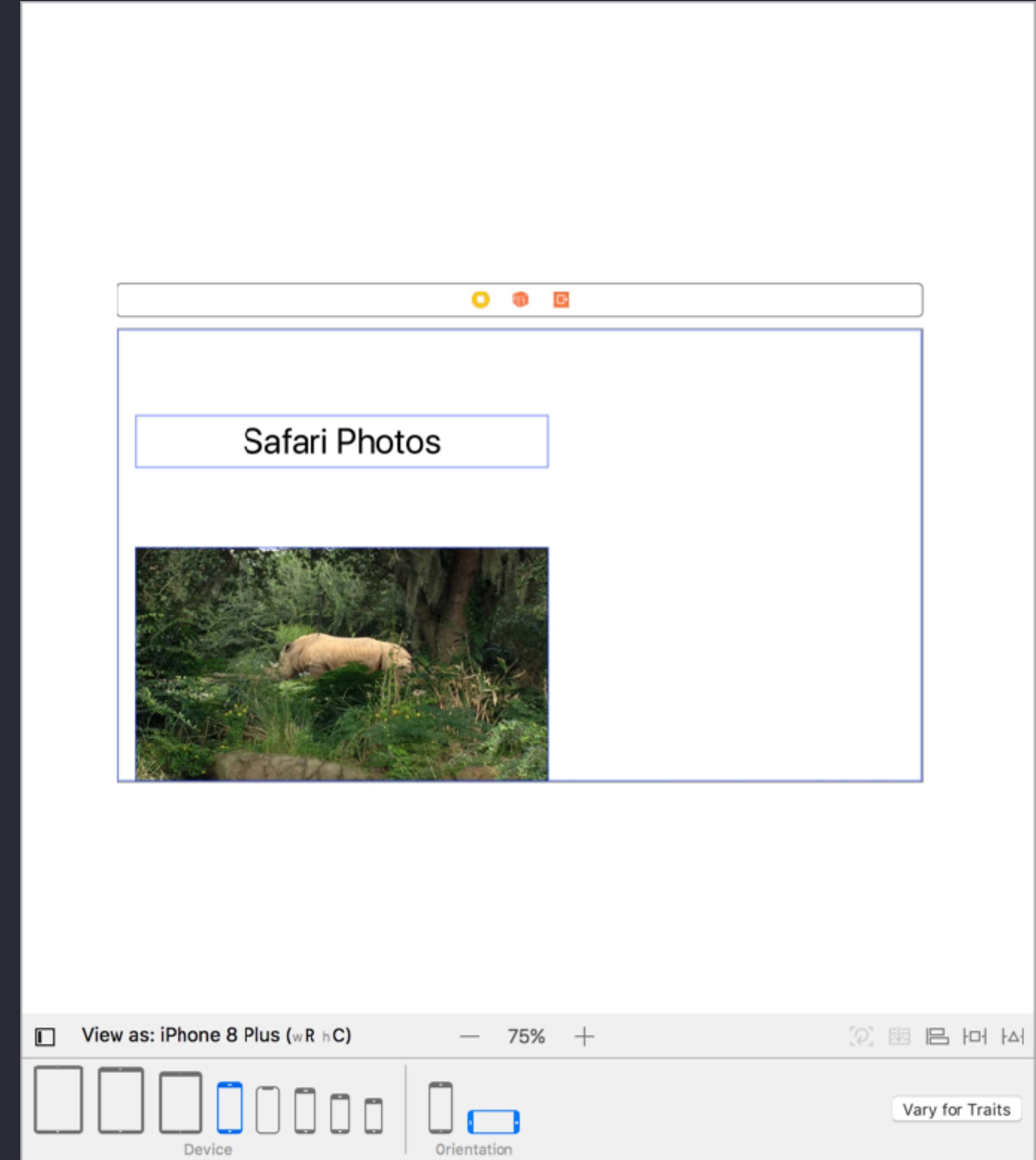


Auto Layout & Stack Views

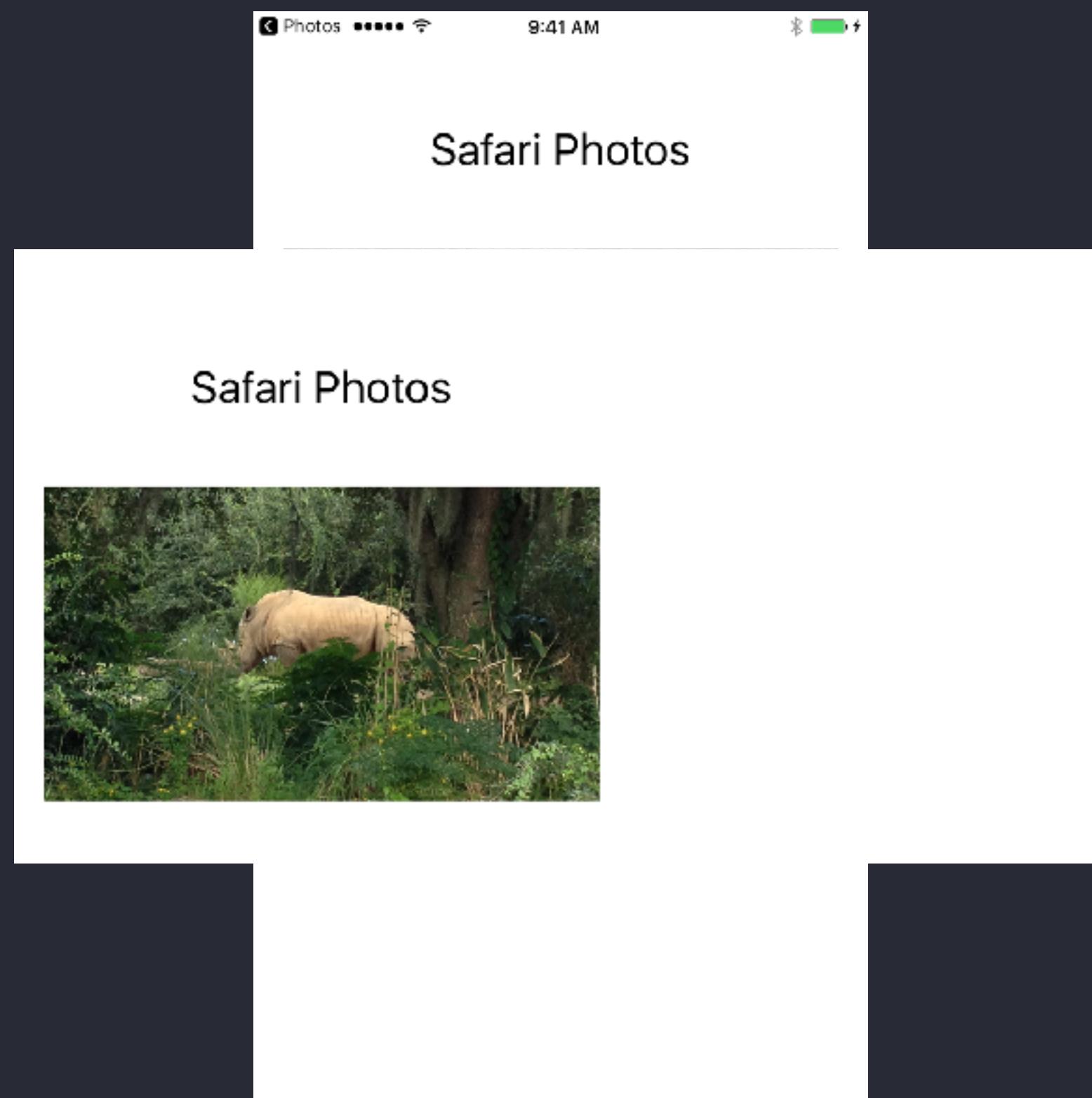




Interface Builder



Why Auto Layout?

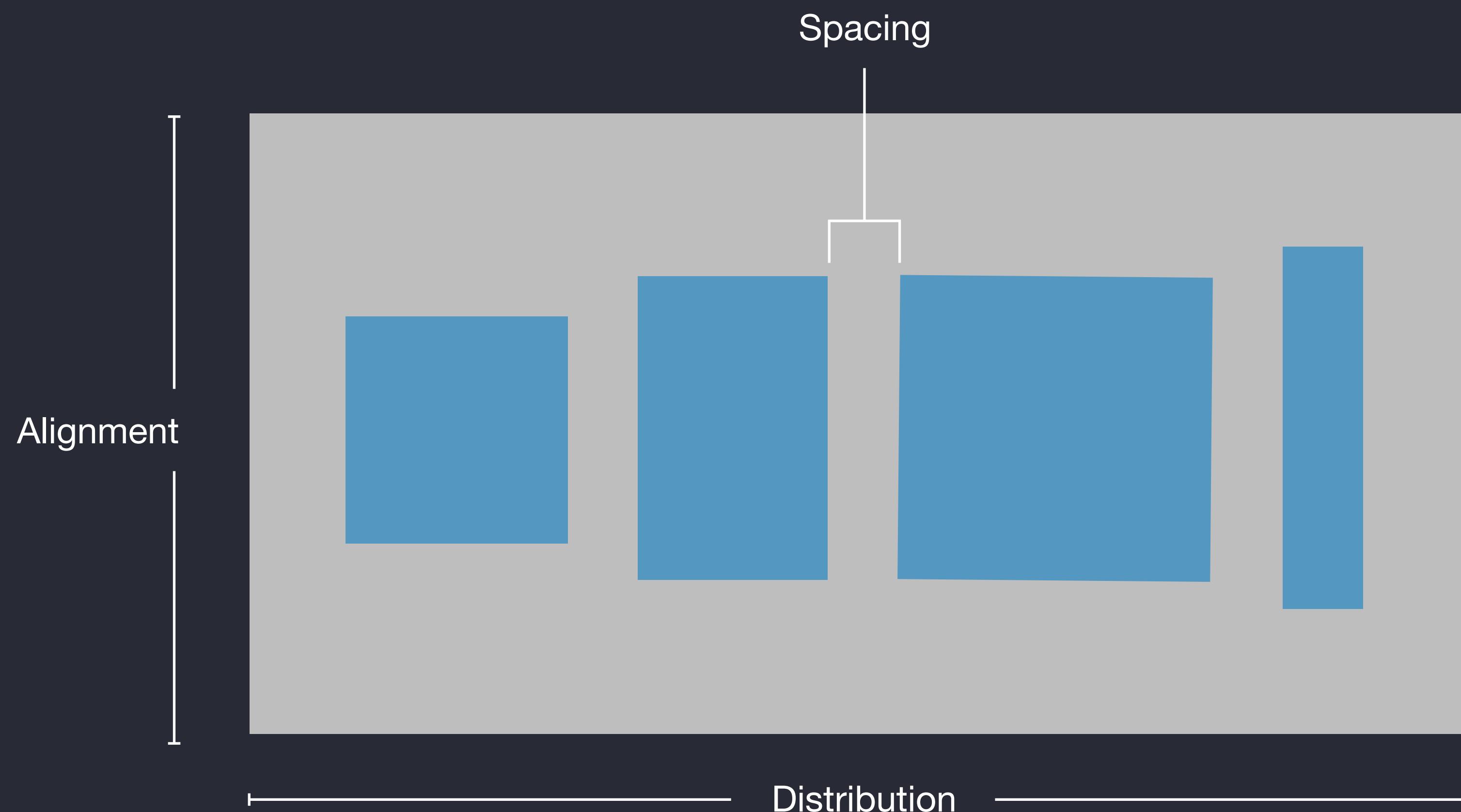


Demo

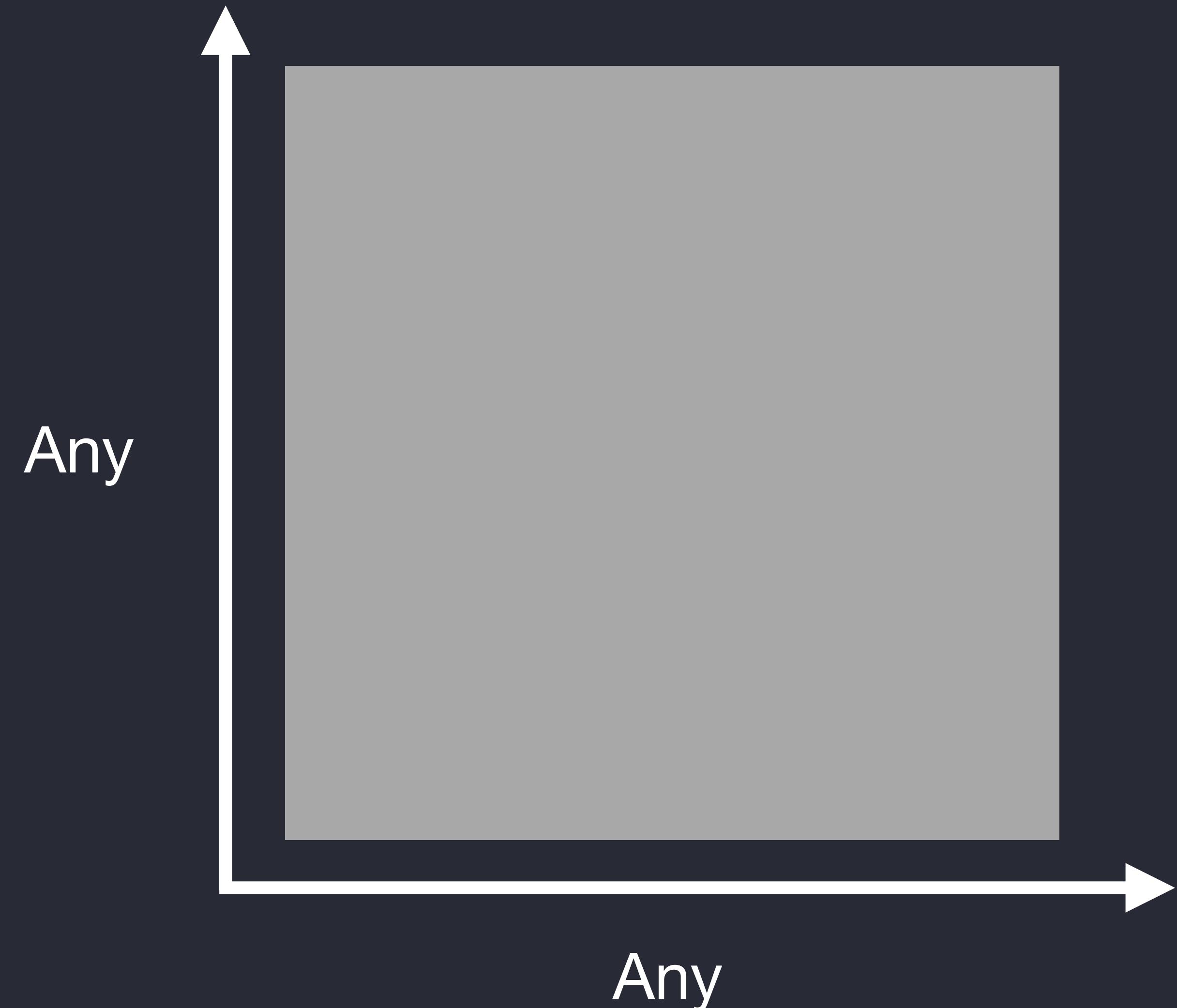
Stack views



Stack views

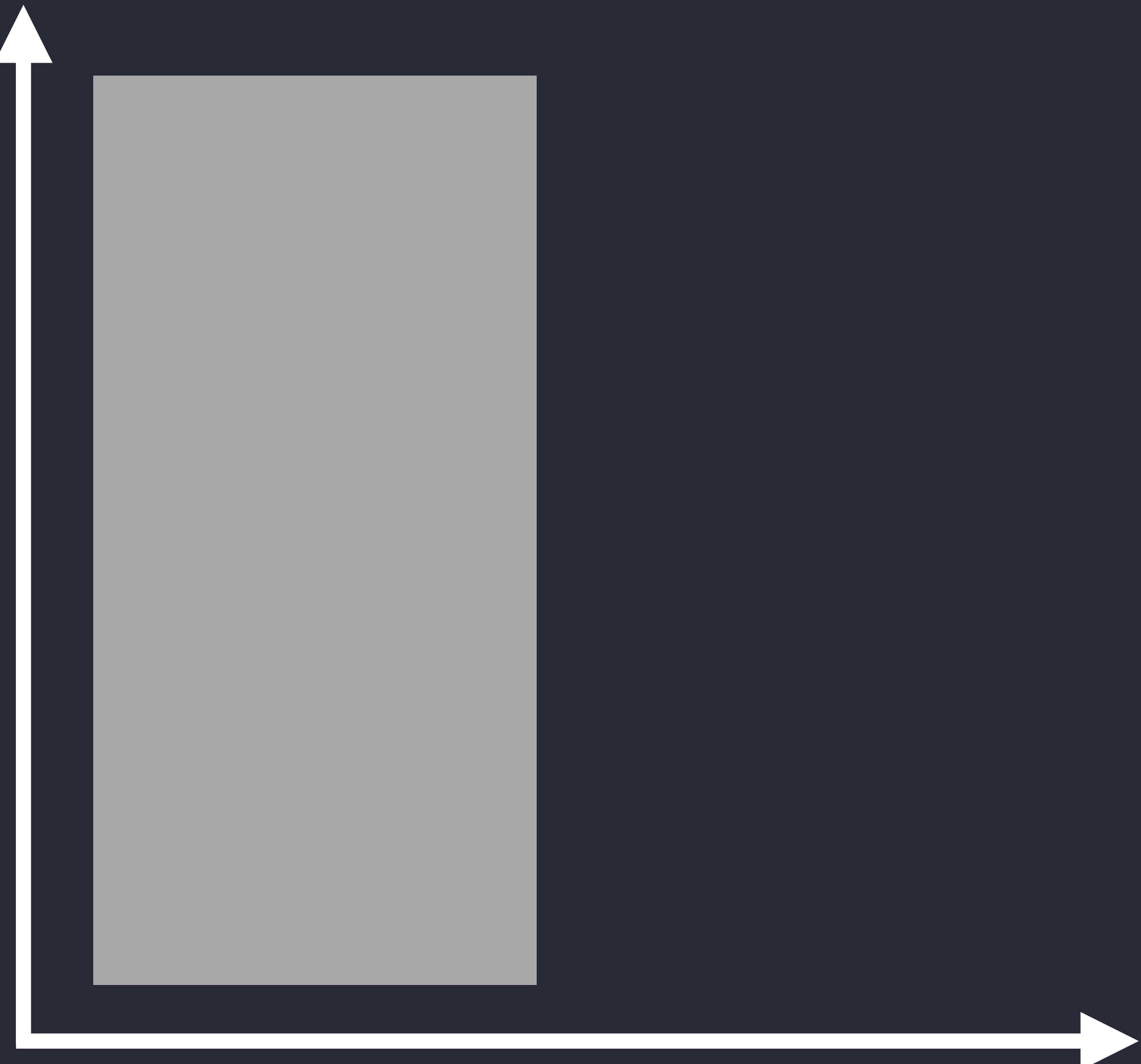


Size classes



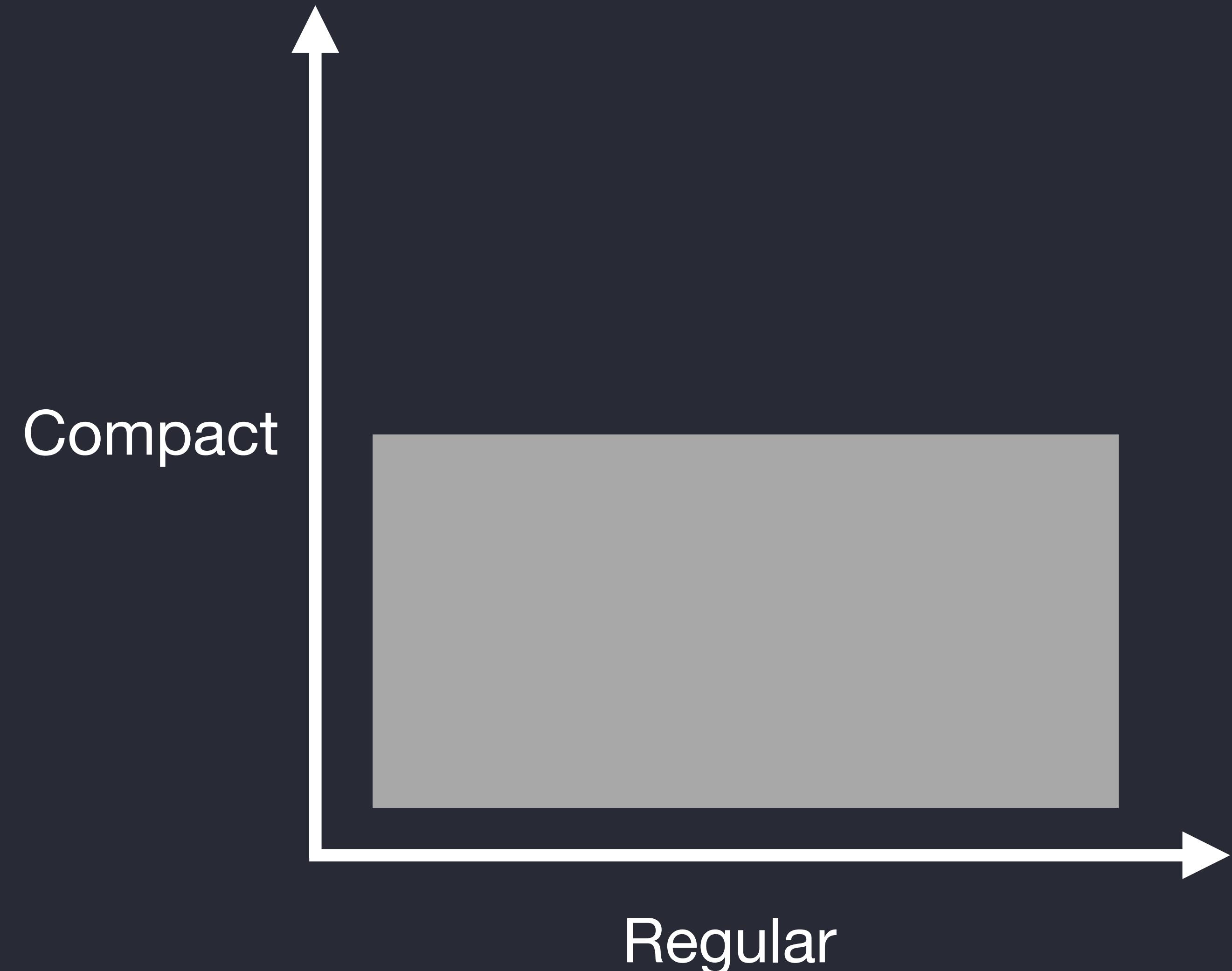
Size classes

Regular

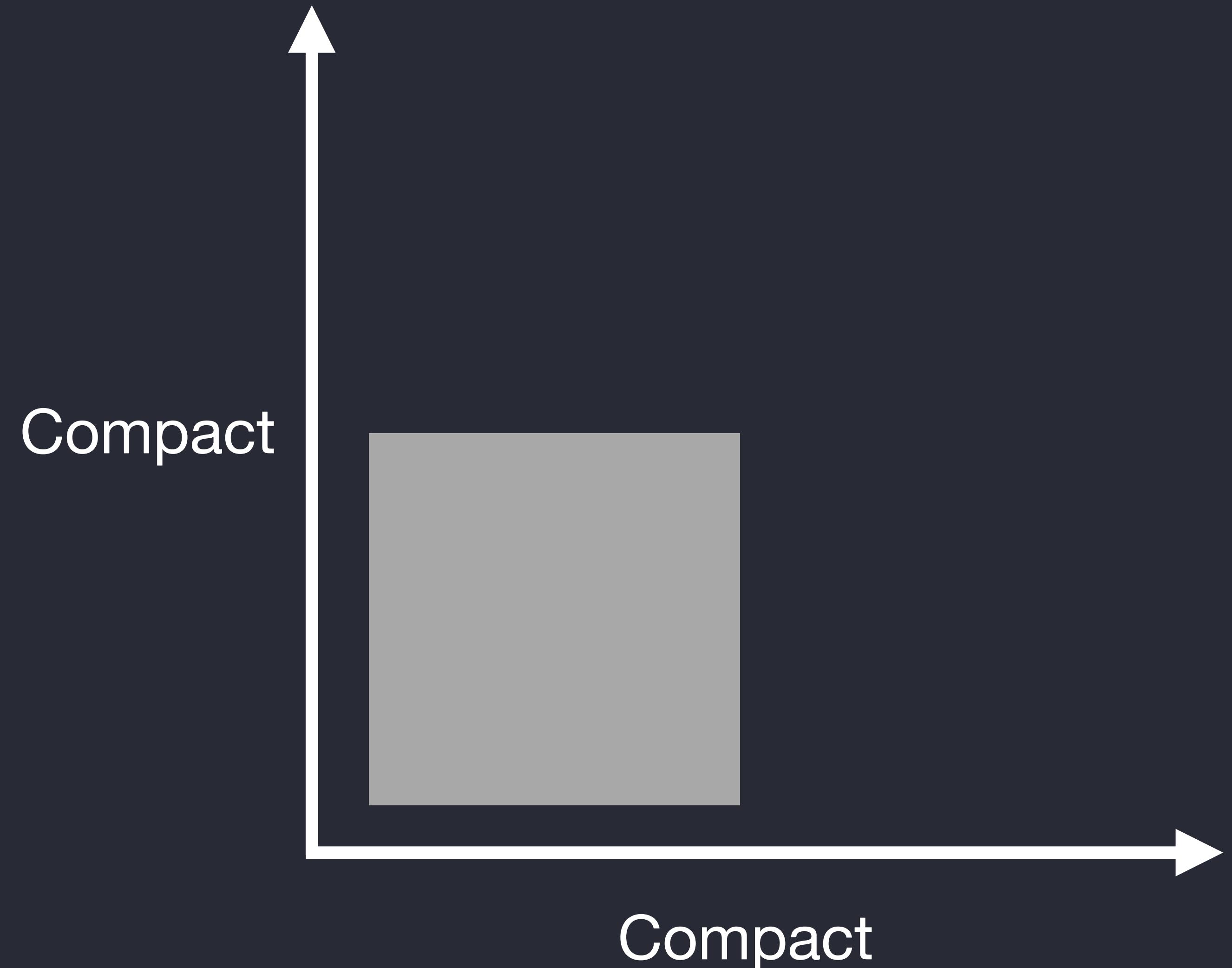


Compact

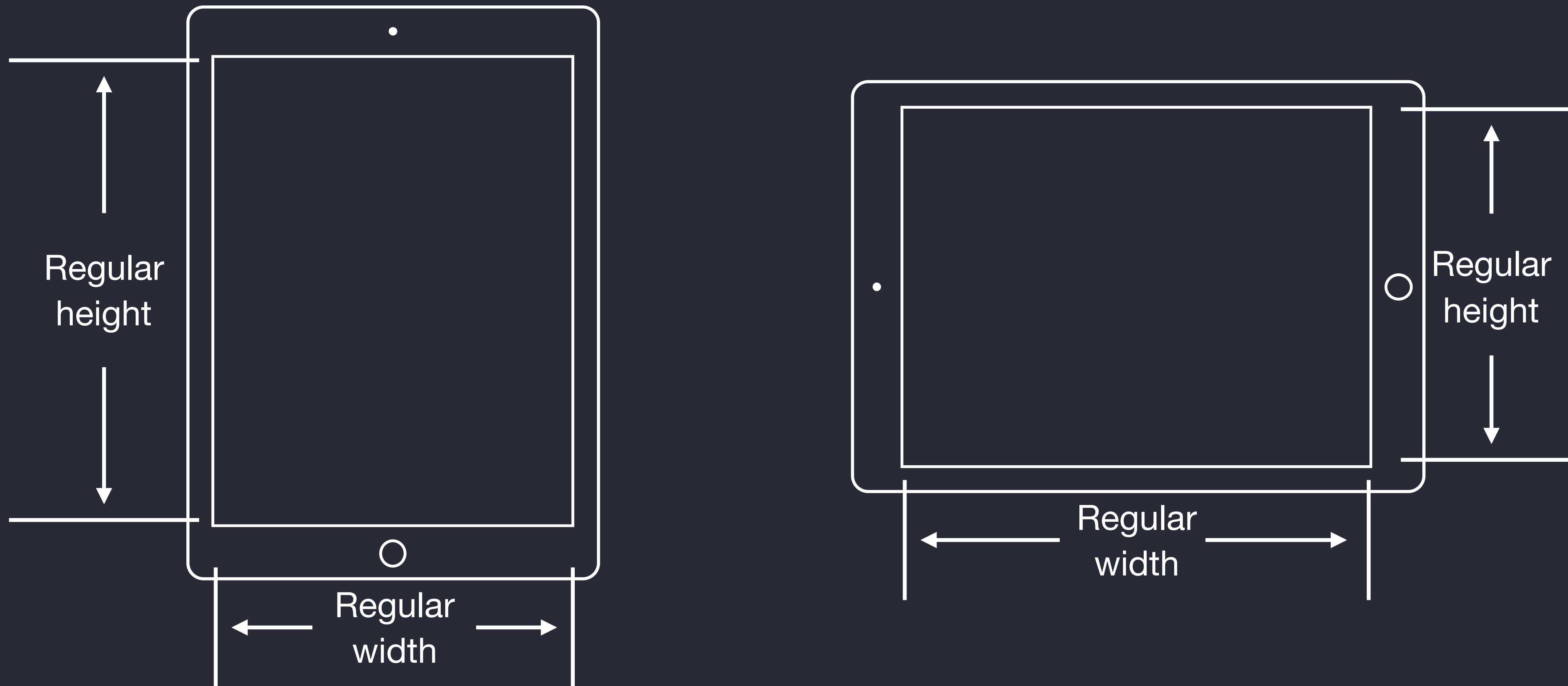
Size classes



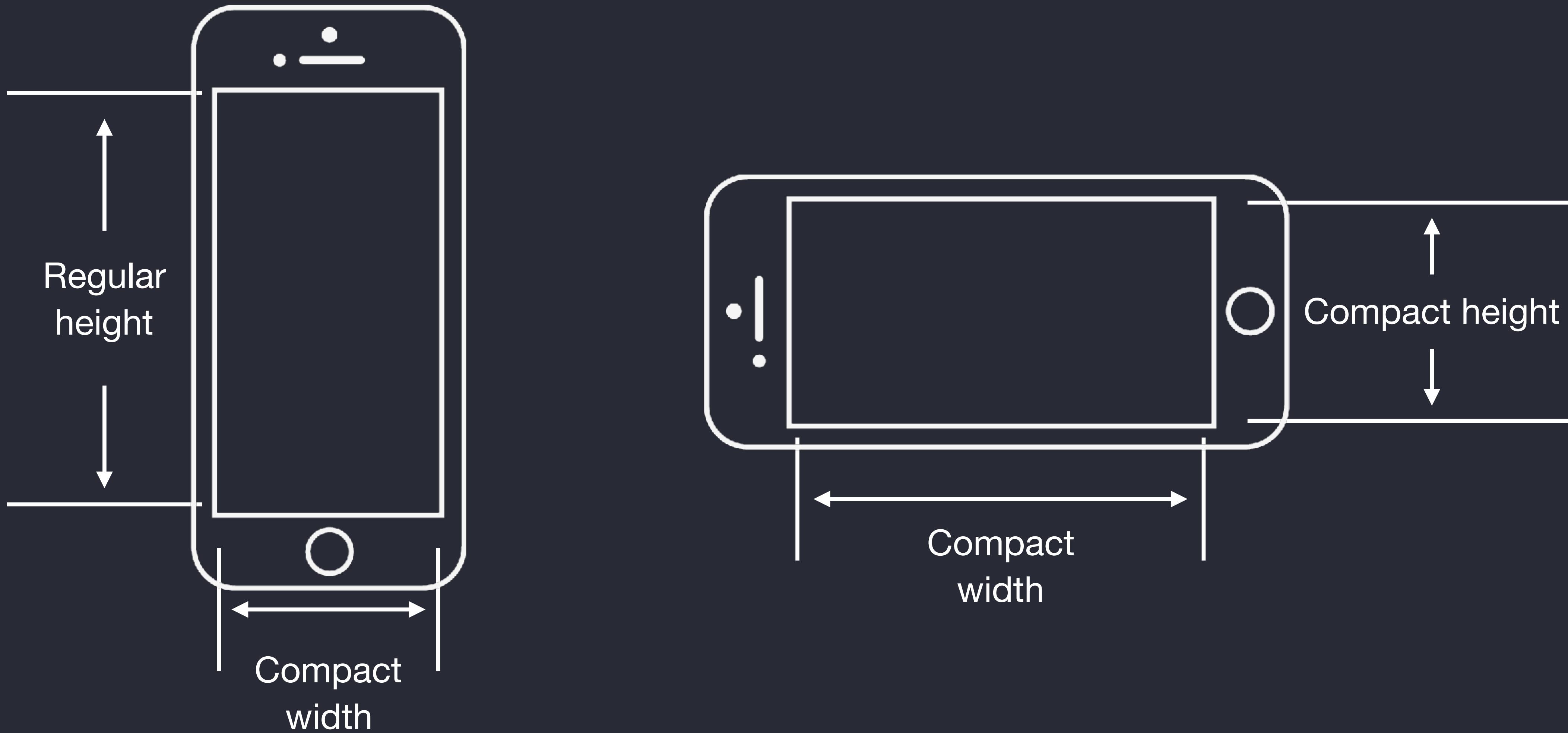
Size classes



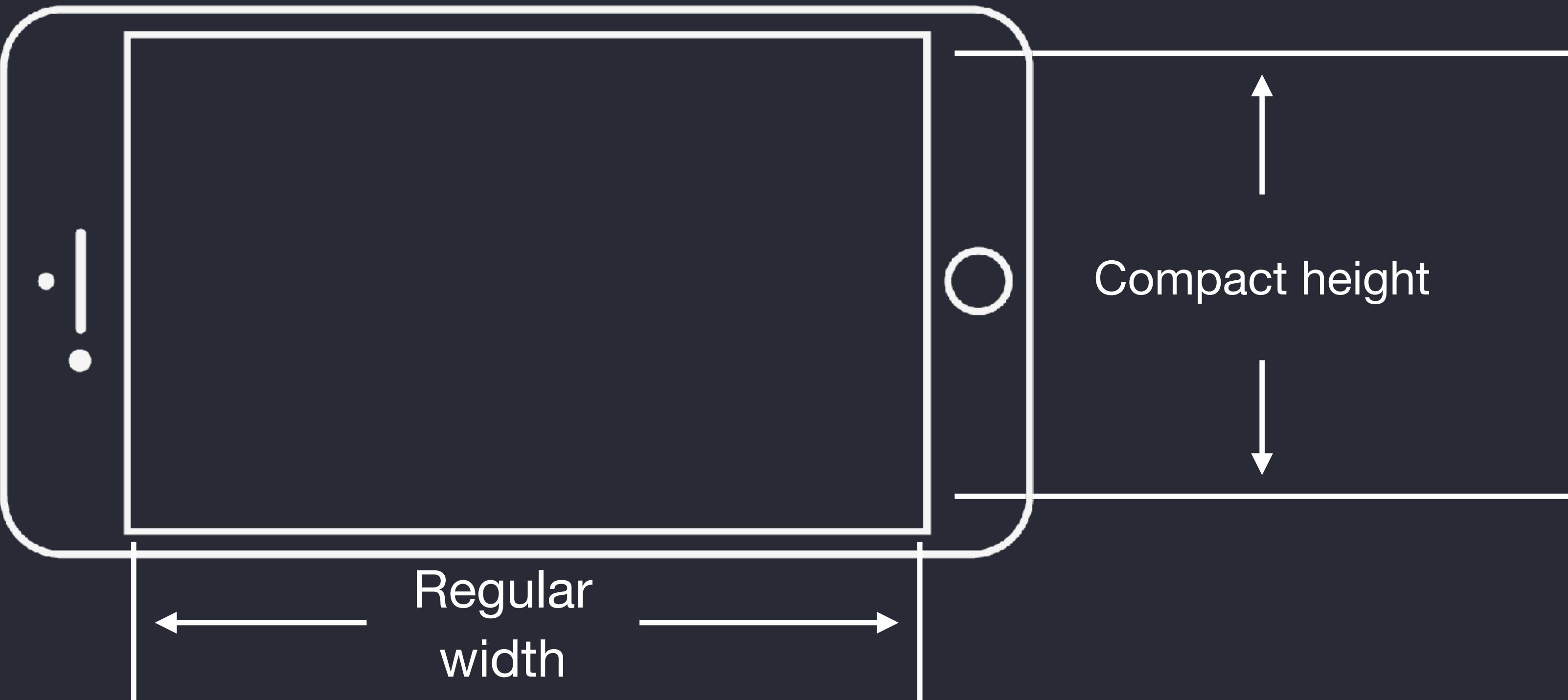
Size classes



Size classes



Size classes



The End.