Часть 2. Связь интерфейса и кода, переменные и константы, типы данных.

Теперь мы наконец-то начнём (ш)кодить! У нашего первого приложения шикарный внешний вид, но хотелось бы добавить интерактивности. Сначала немного теории (честное слово, немного!)

Чтобы выполнять осмысленный код, наше приложение должно передавать данные. Для этого в языке Swift существуют константы ***(let)*** и переменные ***(var)***. Их главное отличие в том, что значение константы нельзя менять. Чтобы создать (разрабы говорят “объявить“) константу, откроем файл *ViewController.swift* и напишем на 11 строке (под словом class):

```
let a: String = "My first constant"
```

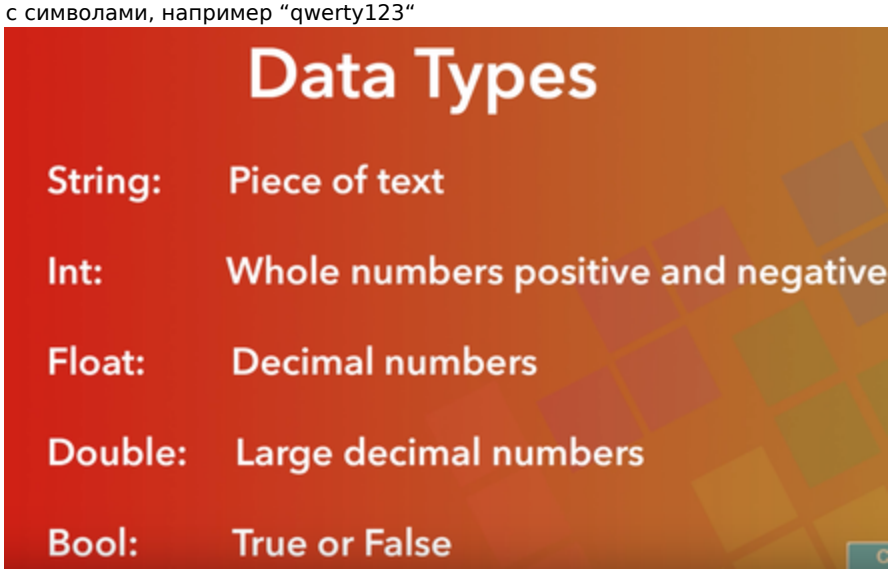
Посмотрим, как объявляется константа. Сначала пишем ключевое слово *let* (оно обозначает константу), потом придумываем название константы. После двоеточия указываем тип, и наконец, устанавливаем значение

```
let <   >: <  > = <  >
```

Объявление переменной отличается только ключевым словом вначале (*var* вместо *let*)

У нашей константы *a* указан тип *String* (строка). Это значит, что в качестве значения у неё может быть любая строка с набором символов (они записываются в кавычках, чтобы Xcode не перепутал нашу строку с программным кодом). На нашем курсе мы будем использовать 3 основных типа данных:

1. Int - целое число, например 124
2. Float / Double - дробное число с точкой, например 42.439
3. String - строка с символами, например “qwerty123“



А теперь немного яблочной магии) Если в объявлении нашей константы убрать тип данных

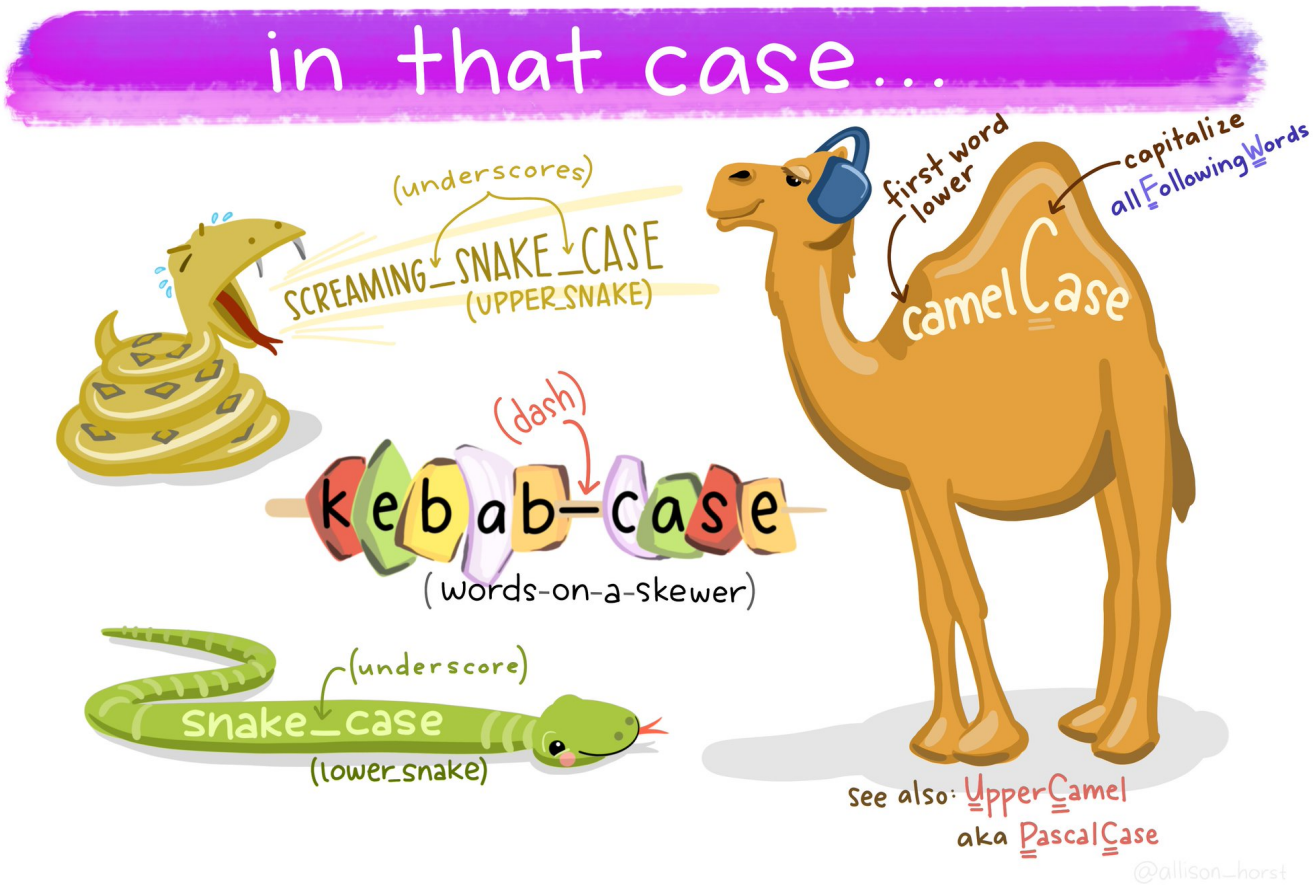
```
let a = "My first constant"
```

Xcode всё равно поймёт, что это строка и присвоит ей тип String! Это работает и с другими типами, так что не стесняйтесь экспериментировать)

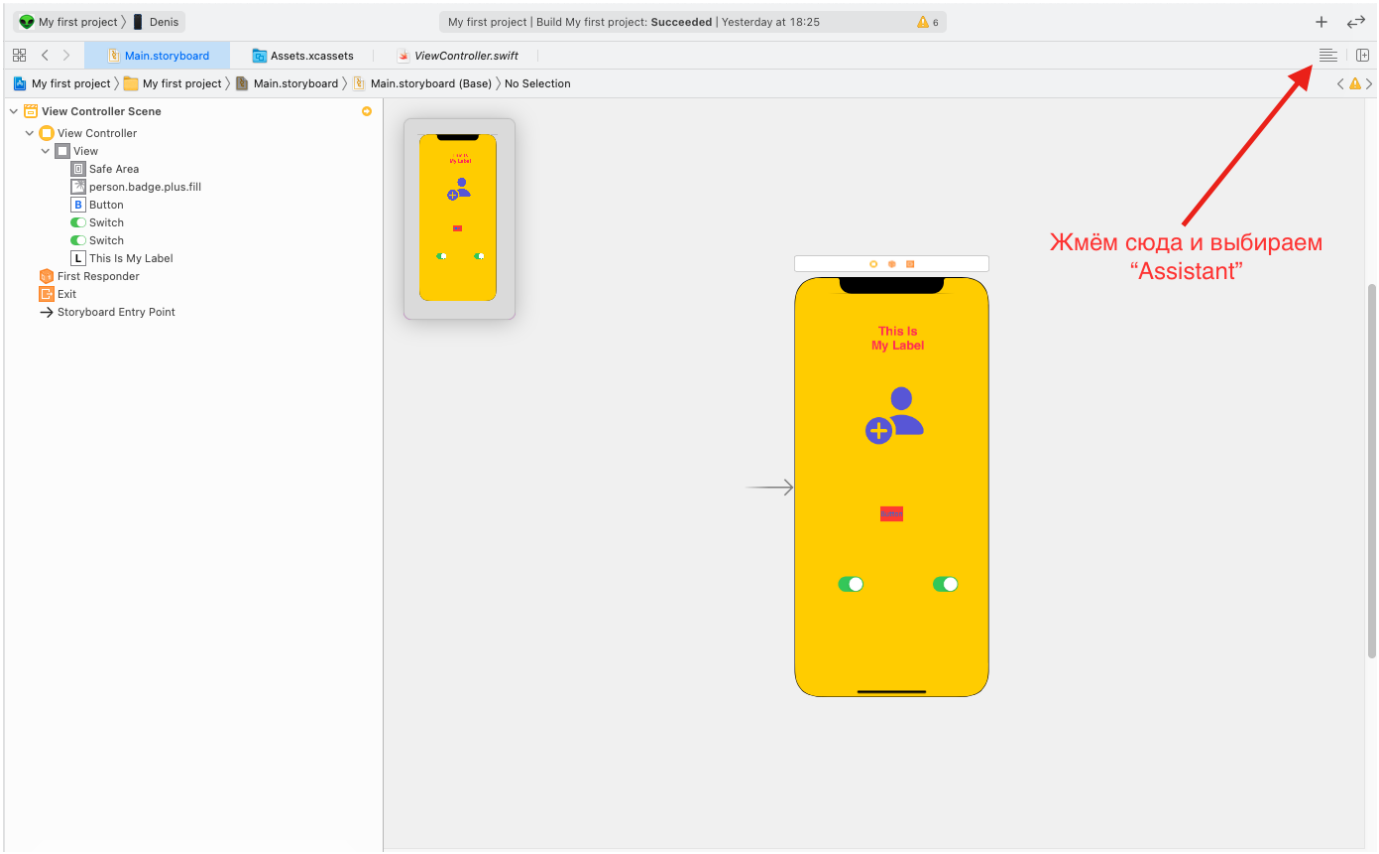
Ещё один важный момент - это именование переменных и констант. Чтобы не запутаться в собственном коде, лучше всего называть их осмысленными словами, например

```
let simpleConstant = "My first constant"
```

и записывать названия в “Верблюжьем стиле“ (он же Camel Case) - с маленькой буквы, но каждое следующее слово выделять с большой. получаетсяПримерноВотТак)

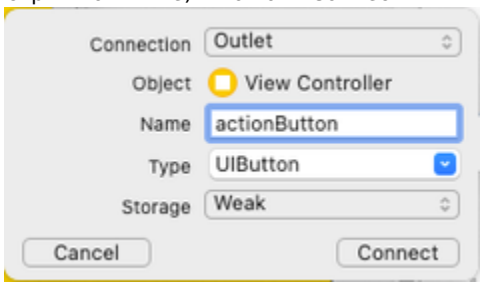


А теперь возвращаемся к нашему приложению! Открываем Main.storyboard и пробуем новый режим - Assistant View



В результате рядом с нашим сторибордом появился файл с кодом. Эти двое действительно работают как напарники (или один из них - ассистент другого). И сейчас мы соединим кнопку с макета с реальным кодом - для этого нужно сделать 3 шага

1. выделить кнопку на макете
2. зажать клавишу *ctrl* и мышью протянуть связующую линию из сториборда во ViewController
3. указать нужные параметры (см. скриншот ниже) и нажать Connect



В результате Xcode автоматически создал переменную! *@IBOutlet* в начале означает, что эта переменная отсылает нас к интерфейсу (IBOutlet = Interface Builder Outlet). А её тип - *UIButton* - позволит нам прямо в коде настраивать все свойства кнопки.

Попробуем поменять цвет фона нашей кнопки в коде. Напишем под строкой **super**.viewDidLoad()

```
actionButton.backgroundColor = UIColor.blue
```

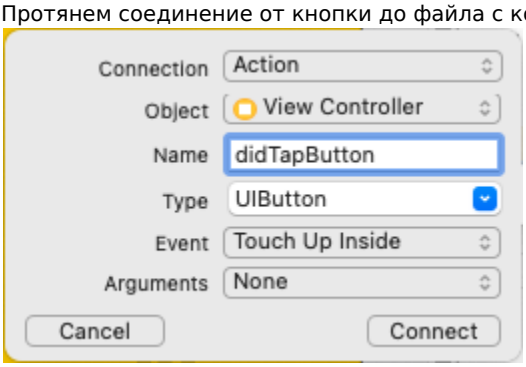
Свойства объектов в коде отделяются точкой. В данном примере мы взяли свойство `backgroundColor` у объекта `actionButton`. А затем присвоим ему свойство `blue` объекта `UIColor`. А значит цвет бэкграунда нашей кнопки должен стать синим! (Не забудем запустить приложение и проверить, что это сработало))

Усложним задачу и добавим ещё один IBOutlet - на этот раз от нашего Label (переменную назовем infoLabel). Получим возможность менять текст на нашем экране

```
infoLabel.text = "Testing testing"
```

Такими соединениями сториборда и контроллера можно создавать не только переменные, но и целые функции (блоки кода, которые будут выполняться при выполнении действия с интерфейсом)

Протянем соединение от кнопки до файла с кодом, но выберем другие настройки



Результат - наш IBAction! Внутри фигурных скобок мы можем писать код, и он будет выполняться каждый раз, когда юзер тапает на кнопку. Начнём с простого

```
print("Button was tapped!")
```

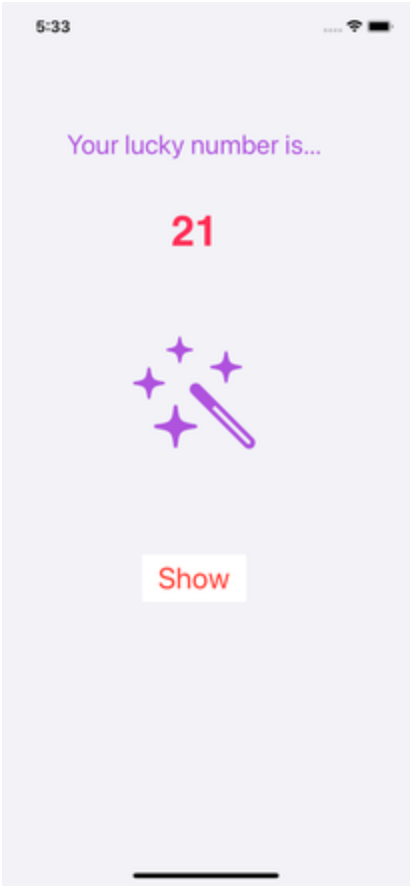
Запустим и проверим, что это сообщение появляется при каждом нажатии на кнопку.

А теперь используем небольшой трюк и заставим наш infoLabel показывать рандомное число от 0 до 100 при каждом тапе

```
let number = Int.random(in: 1...100)
infoLabel.text = String(number)
```

Каждый раз, когда юзер тапает по кнопке, мы создаем переменную number со случайным числом. Затем вписываем его в наш infoLabel (предварительно превращая в строку)

Наше второе приложение - генератор счастливых чисел!)




```
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var actionButton: UIButton!
    @IBOutlet weak var infoLabel: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()

    }

    @IBAction func didTapButton() {
        infoLabel.text = String(Int.random(in: 1...100))
    }
}
```