Innotech
MohammadHosein
March 16, 2018

Swift Document

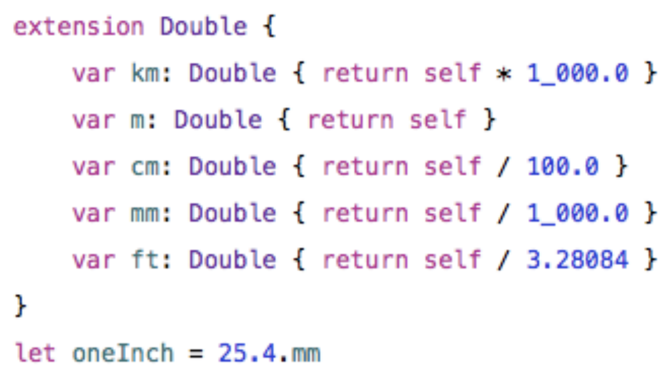
Udemy Ios10(EggTimer App)

In this document I cover some of the points I got after watching Udemy videos and also steps to build an application in Swift. At first in the following of previous document there's a little about extensions and then we dive into graphics together.

Extensions

Extensions add new functionality to an existing class, structure, enumeration, or protocol type.

To extend the Double type in swift and use it:

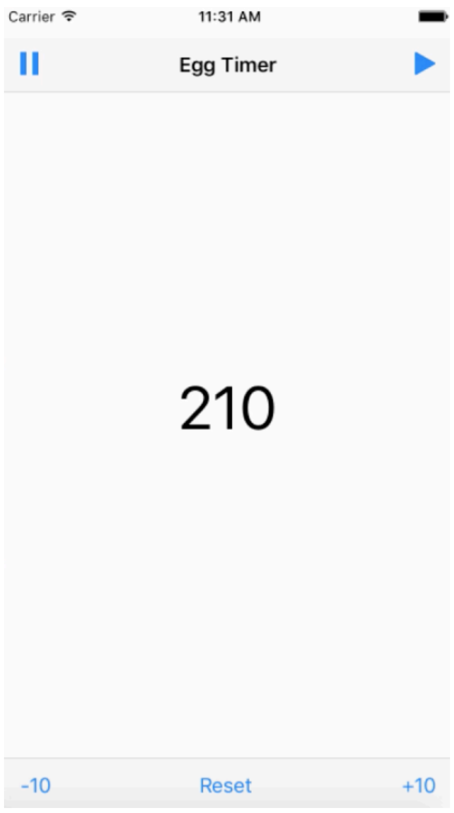


```
extension Double {
    var km: Double { return self * 1_000.0 }
    var m: Double { return self }
    var cm: Double { return self / 100.0 }
    var mm: Double { return self / 1_000.0 }
    var ft: Double { return self / 3.28084 }
}
let oneInch = 25.4.mm
```

In this document we use single mode app; open your Xcode and select this mode. Lets start:D

EggTimer App

Here's our timer to build this together first of all we need to be familiar with some definitions and objects in swift.



To reduce the size of this document I explain as short as possible so for more information you can use their quick helps as explained before.

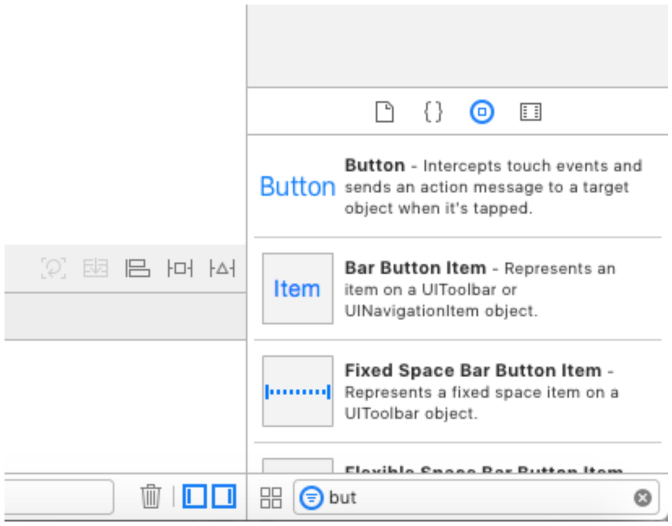
Timer() : the scheduledTimer function in this class helps us to make a timer.

- **timeInterval**: the number of seconds between firings of the timer
- **aSelector**: function(message) which called when the timer fires.
- **target**: the object to which to send the message specified by aSelector
- **userInfo**: the user info for the timer which is commonly nil
- **repeats**: If true, the timer will repeatedly reschedule itself until invalidated. If false, the timer will be invalidated after it fires.

In swift our code have two parts; ViewController and Main.storyboard. We do graphics in storyboard and link it to our code in ViewController.

For the bar at the top I used Navigation Bar and with the help of Bar Button Item, pause and play items are built.

to use these items you can search at the right-bottom corner of the display indicated bellow and drag it into your Main.storyboard.

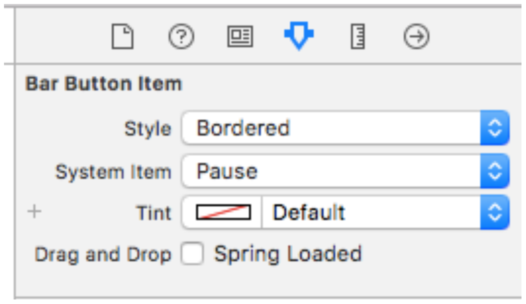


Button - Intercepts touch events and sends an action message to a target object when it's tapped.

Bar Button Item - Represents an item on a UIToolbar or UINavigationItem object.

Fixed Space Bar Button Item - Represents a fixed space item on a UIToolbar object.

some shapes like pause, play , stop are available after you click on your Bar Button



Bar Button Item

Style: Bordered

System Item: Pause

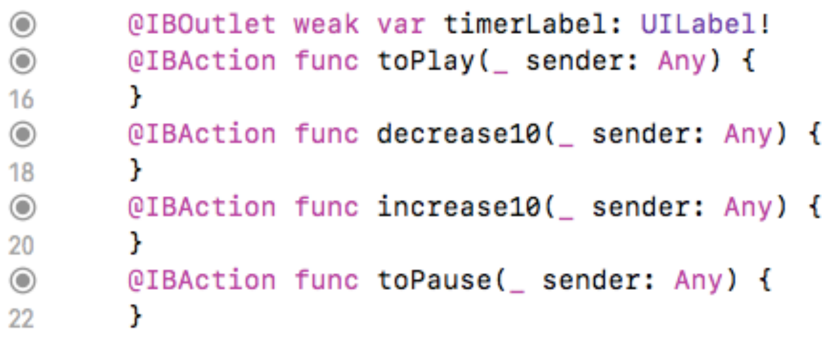
Tint: Default

Drag and Drop: Spring Loaded

Use label for the number at the center.

To make the bar at the bottom use Toolbar, Bar Button and Flexible space. I'm sure you've got that intelligence to figure it out.

Now its the time to join your graphic with your code. click on the double bobble at the top of the screen to appear the assistant editor. one of them must be Main.storyboard and the other, ViewController. ctrl + drag your buttons to your code in the class and above all of the other functions as an ActionListener and label as an outlet.



```
    @IBOutlet weak var timerLabel: UILabel!
    @IBAction func toPlay(_ sender: Any) {
    }
    @IBAction func decrease10(_ sender: Any) {
    }
    @IBAction func increase10(_ sender: Any) {
    }
    @IBAction func toPause(_ sender: Any) {
    }
```

Now we have what we want from UI. Define timer globally in code.

var timer = Timer()

this class have two functions which help us the most first scheduledTimer function which have been explained before and invalidate function that stops the timer(we use invalidate function in the toPause function and the scheduledTimer in toPlay function to fire the timer.)

there must be a function that called when the timer fires (in front of aSelector in scheduledTimer function) which I called it decrease time. one important thing that Swift4 will aware you if you don't make attention is to use @objc:

Before Swift 4, the compiler made some **Swift declarations automatically available to Objective-C**. For example, if one subclassed from NSObject, the compiler created Objective-C entry points for all methods in such classes. The mechanism is called @objc inference. but now **In Swift 4, such automatic @objc inference is deprecated** because it is costly to generate all those Objective-C entry points. to expose our function to objective-c @objc is needed.

to increase and decrease timer we have to change the label of our timerLabel. First there is a global variable called time which is an Integer and is our current time, so:

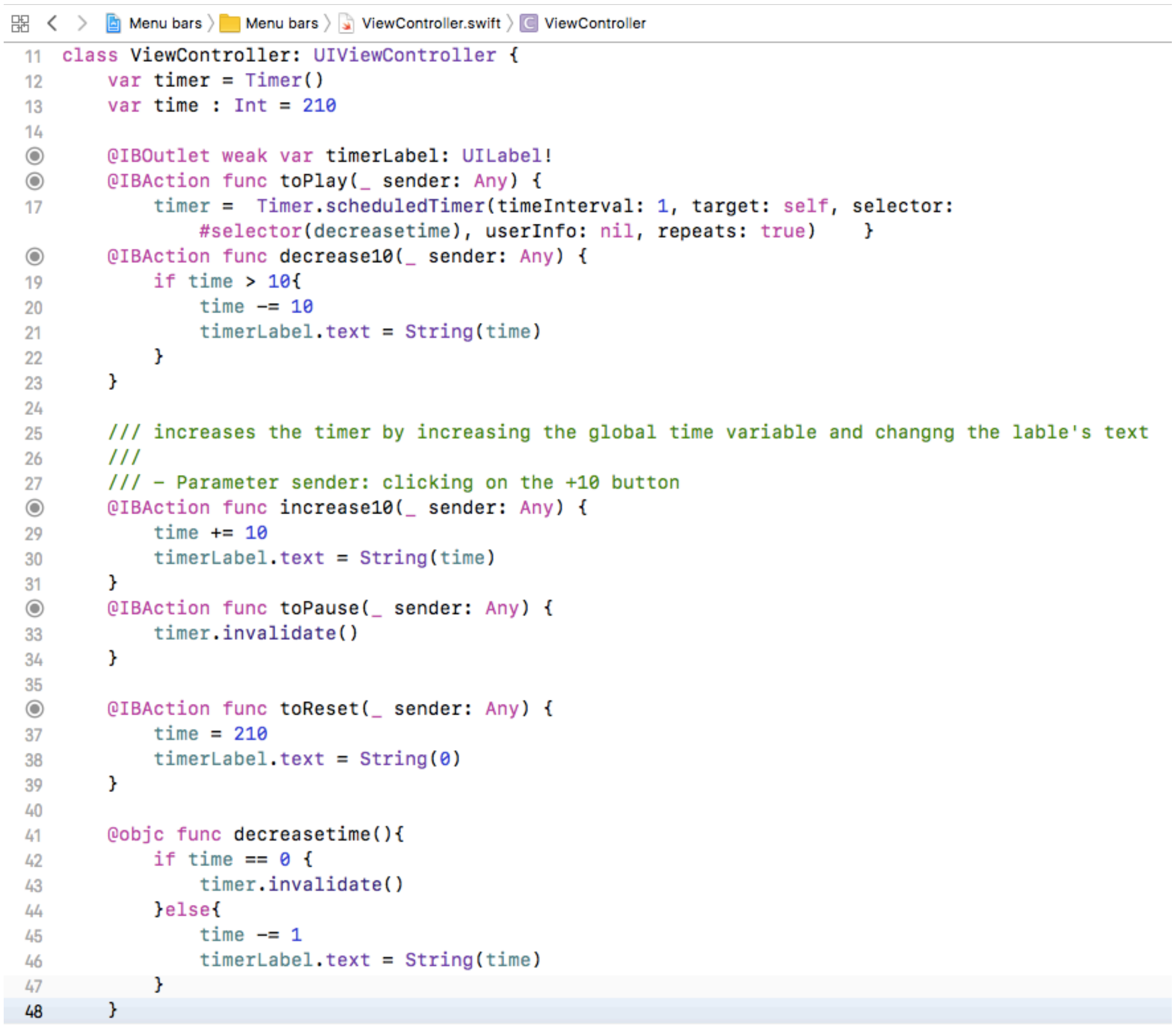
time += 10   to increase

time -= 10   to decrease

And then we assign time to our label:

timerLabel.text = String(time)

* notice that if our time is less than 10 timer can't decrease the time 10 seconds.



```swift
class ViewController: UIViewController {
    var timer = Timer()
    var time : Int = 210

    @IBOutlet weak var timerLabel: UILabel!
    @IBAction func toPlay(_ sender: Any) {
        timer =  Timer.scheduledTimer(timeInterval: 1, target: self, selector:
            #selector(decreasetime), userInfo: nil, repeats: true)    }
    @IBAction func decrease10(_ sender: Any) {
        if time > 10{
            time -= 10
            timerLabel.text = String(time)
        }
    }

    /// increases the timer by increasing the global time variable and changng the lable's text
    ///
    /// - Parameter sender: clicking on the +10 button
    @IBAction func increase10(_ sender: Any) {
        time += 10
        timerLabel.text = String(time)
    }
    @IBAction func toPause(_ sender: Any) {
        timer.invalidate()
    }

    @IBAction func toReset(_ sender: Any) {
        time = 210
        timerLabel.text = String(0)
    }

    @objc func decreasetime(){
        if time == 0 {
            timer.invalidate()
        }else{
            time -= 1
            timerLabel.text = String(time)
        }
    }
```