Suyati℠

dream. dare. do

WHY YOU SHOULD USE SWIFT?

The new programming language from Apple is likely to be a huge hit with developers for the ***safety***, ***stability***, ***ease*** and ***flexibility*** it offers.

Index

Introduction

Apple introduced their new programming language "**Swift**" during the 2014 World Wide Developers Conference (WWDC). Swift was declared the native language for future iOS and Mac OSX applications, which will eventually replace the very old Objective-C programming language of the 1980s.

According to Apple,
"*Swift is a new programming language for iOS and OS X apps that builds on the best of C and Objective-C without the constraints of C compatibility.*"

In fact, it is called Objective-C without C. Swift is designed to work with Apple's existing Core technologies like ***Cocoa, Cocoa Touch, Core Animation and Quartz.***

There is a lot going on as Apple decides to go for a new language by bidding its current stable and yet successful objective-C goodbye. The primary aim, as we know, is to make its platform reachable to more developers. At the same time, apps written on Swift are more stable and safe when compared to the older version.

Swift, an apt successor to **Objective-C**

Apple believes that Swift is an innovative programming language for Cocoa and Cocoa Touch. Swift code was built to be fast, and yet work alongside Objective-C. Writing code in Swift is fun and interactive, and although the syntax is concise, it is expressive. In fact, the syntax and standard library have been refined, making code-writing incredibly easy and highly functional. The high performance LLVM compiler transforms Swift code into optimized native code, which ensures matchless results with Mac, iPhone and iPad hardware.

Swift has low-level primitives such as types, flow control, and operators. Cocoa and Cocoa Touch developers will get just the kind of performance and power they desire with Swift’s object-oriented features such as classes, protocols, and generics.

While Objective-C is unsafe, leading to unwanted debugging time, Swift is considered “safe by default”. Also, the range checks on array access and overflow operators help developers to make apps more stable. Unlike the complexity of Objective C, which does not have shades of any other programming language, Swift has functional elements that reduce app development time drastically. For a developer, the best thing about Swift is it feels a lot like a scripting language but even more powerful with OO integration. Readability is enhanced by using simplified syntax.

The Swift code will look familiar to all developers, especially those who are familiar with C#, Javascript, Python, Ruby, etc. because Swift mixes language ideas from these languages.

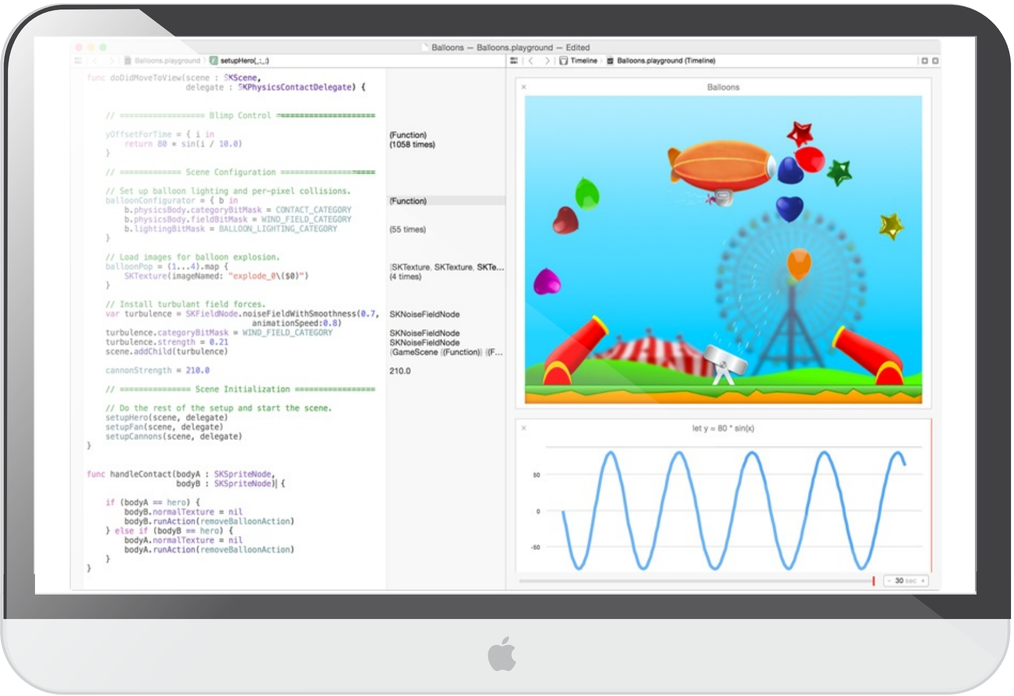
As compared to Objective-C, which is very dynamic in nature, Swift is strongly typed and static language. It simplifies the memory management by using Automatic Reference counting. Code written in Swift can work side by side with Objective-C.

The main concern was how the existing apps would respond to Swift. Will the apps written on Objective-C have to be rewritten? What about all the libraries and frameworks written on Objective-C?

Thankfully, Objective-C is not going away, at least not immediately. Xcode 6.0 is more powerful now that it can handle both Swift and Objective-C in parallel. This inter-operability will help developers reuse their old frameworks and libraries, and at the same time, develop new apps using Swift. Thus, the overhead time will be minimal.

Apple says Swift can be nearly twice as fast as Objective-C in complex tasks like sorting and encryption. The development time will be much lesser than before. Stability and reliability of apps has gone up. With the support of iOS 8 and OSX Yosemite apps will work flawlessly. There is a remarkable boost in app performance when Swift comes into play. This also means that tasks, which had been difficult to implement can be accommodated easily and apps will be faster than ever before. Setting up a main function, or import references can be avoided. It is an approach that is designed to make coding more accessible and readable.

Apple added an innovative feature called playgrounds to Swift that helps programmers to experiment with Swift code and see the results immediately. Plus, there is no need of building and running the app.



```
func doDidMoveToView(scene : SKScene,
                     delegate : SKPhysicsContactDelegate) {

    // ================== Blimp Control =====================

    yOffsetForTime = { i in
        return 80 * sin(i / 10.0)
    }

    // ============= Scene Configuration ====================

    // Set up balloon lighting and per-pixel collisions.
    balloonConfigurator = { b in
        b.physicsBody.categoryBitMask = CONTACT_CATEGORY
        b.physicsBody.fieldBitMask = WIND_FIELD_CATEGORY
        b.lightingBitMask = BALLOON_LIGHTING_CATEGORY
    }

    // Load images for balloon explosion.
    balloonPop = (1...4).map {
        SKTexture(imageNamed: "explode_0\($0)")
    }

    // Install turbulant field forces.
    var turbulence = SKFieldNode.noiseFieldWithSmoothness(0.7,
                                   animationSpeed:0.8)
    turbulence.categoryBitMask = WIND_FIELD_CATEGORY
    turbulence.strength = 0.21
    scene.addChild(turbulence)

    cannonStrength = 210.0

    // =============== Scene Initialization =================

    // Do the rest of the setup and start the scene.
    setupHero(scene, delegate)
    setupFan(scene, delegate)
    setupCannons(scene, delegate)
}


func handleContact(bodyA : SKSpriteNode,
                   bodyB : SKSpriteNode) {

    if (bodyA == hero) {
        bodyB.normalTexture = nil
        bodyB.runAction(removeBalloonAction)
    } else if (bodyB == hero) {
        bodyA.normalTexture = nil
        bodyA.runAction(removeBalloonAction)
    }
}
```

```
(Function)
(1058 times)

(Function)

(55 times)

[SKTexture, SKTexture, SKTe...
(4 times)

SKNoiseFieldNode

SKNoiseFieldNode
SKNoiseFieldNode
{GameScene {(Function)} {(F...

210.0
```

Source: Apple.com

Playground makes the coding in Swift incredibly fast and fun. It can be considered as an interactive document where Swift code can be compiled and run live as we type. Playground allows results of operations to be presented step by step when they execute. Playgrounds can be created within an XCode project or as a separate one. These kind of interactive developments with live results help in prototyping complex algorithms before deployment in a full application.

Swift is going to be an inevitable part of app development. In years to come we can see Swift's evolution. Objective-C might become entirely redundant in a few years. Swift, with its performance and reliability, will stay steady in the long run.

The apps created on Swift, together with the help of iOS8 and Yosemite, will outrun Android and Windows easily.

Since the effort of the developer is much less, the cost of the product will also come down. Also code written on Swift is easily maintainable. So any change requests or new feature can be accommodated without much headache. We could see more stable and lightning-fast apps in the coming days.

Suyati *technologies*

Suyati is a young, upwardly mobile company focused on delivering niche IT services to support myriad Digital Engagement strategies. Our expertise includes integration and delivery of CRM, CMS, and e-commerce solutions, in addition to mobile app development for iOS, Android and Window.

To Know more about how we can help you, get in touch with use: services@suyati.com

www.suyati.com

Referrence:
https://developer.apple.com/swift/
http://www.macrumors.com/2014/09/10/swift-gm-ios-os-x-with-yosemite/