Method Hook Swift

```
import Foundation

extension DispatchQueue {
    private static var _onceTracker = [String]()
    public class func once(token: String, block: () -> ()) {
        objc_sync_enter(self)
        defer {
            objc_sync_exit(self)
        }
        if _onceTracker.contains(token) {
            return
        }
        _onceTracker.append(token)
        block()
    }

    func async(block: @escaping ()->()) {
        self.async(execute: block)
    }

    func after(time: DispatchTime, block: @escaping ()->()) {
        self.asyncAfter(deadline: time, execute: block)
    }
}


import UIKit



private let onceToken = "Method Swizzling viewWillAppear"
```

```
extension UIViewController {

    public class func initializeMethod() {
        // Make sure This isn't a subclass of UIViewController, So that It applies to all UIViewCont
oller childs

        if self != UIViewController.self {
            return
        }

        //DispatchQueue函数保证代码只被执行一次，防止又被交换回去导致得不到想要的效果
        DispatchQueue.once(token: onceToken) {

            let originalSelector = #selector(UIViewController.viewWillAppear(_:))
            let swizzledSelector = #selector(UIViewController.swizzled_viewWillAppear(animated:))

            let originalMethod = class_getInstanceMethod(self, originalSelector)
            let swizzledMethod = class_getInstanceMethod(self, swizzledSelector)
            //在进行 Swizzling 的时候,需要用 class_addMethod 先进行判断一下原有类中是否有要替
方法的实现
            let didAddMethod: Bool = class_addMethod(self, originalSelector, method_getImplem
ntation(swizzledMethod!), method_getTypeEncoding(swizzledMethod!))
            //如果 class_addMethod 返回 yes,说明当前类中没有要替换方法的实现,所以需要在父类中
找,这时候就用到 method_getImplemetation 去获取 class_getInstanceMethod 里面的方法实现,
后再进行 class_replaceMethod 来实现 Swizzing

            if didAddMethod {
                class_replaceMethod(self, swizzledSelector, method_getImplementation(originalMe
hod!), method_getTypeEncoding(originalMethod!))
            } else {
                method_exchangeImplementations(originalMethod!, swizzledMethod!)
            }

        }


    }


    @objc func swizzled_viewWillAppear(animated: Bool) {
        //需要注入的代码写在此处
        view.backgroundColor = UIColor.red
        self.swizzled_viewWillAppear(animated: animated)
    }
}



import UIKit

private let onceToken1 = "UITextField.swizzling_draw"
extension UITextField {

    @objc func swizzling_draw(_ rect: CGRect) {
        setValue(UIFont.systemFont(ofSize: 5, weight: UIFont.Weight.thin), forKeyPath: "_placeho
derLabel.font")
        swizzling_draw(rect)
    }

    public class func drawMethod() {

        if self != UITextField.self {
            return
        }
        //DispatchQueue函数保证代码只被执行一次，防止又被交换回去导致得不到想要的效果
        DispatchQueue.once(token: onceToken1) {
            let originalSelector = #selector(UITextField.draw(_:))
            let swizzledSelector = #selector(UITextField.swizzling_draw(_:))

            let originalMethod = class_getInstanceMethod(self, originalSelector)
            let swizzledMethod = class_getInstanceMethod(self, swizzledSelector)
            //在进行 Swizzling 的时候,需要用 class_addMethod 先进行判断一下原有类中是否有要替
方法的实现
            let didAddMethod: Bool = class_addMethod(self, originalSelector, method_getImplem
ntation(swizzledMethod!), method_getTypeEncoding(swizzledMethod!))
            //如果 class_addMethod 返回 yes,说明当前类中没有要替换方法的实现,所以需要在父类中
找,这时候就用到 method_getImplemetation 去获取 class_getInstanceMethod 里面的方法实现,
后再进行 class_replaceMethod 来实现 Swizzing

            if didAddMethod {
                class_replaceMethod(self, swizzledSelector, method_getImplementation(originalMe
hod!), method_getTypeEncoding(originalMethod!))
            } else {
                method_exchangeImplementations(originalMethod!, swizzledMethod!)
            }
        }
    }
}


    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptio
s: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {

        UITextField.drawMethod()
        UIViewController.initializeMethod()
        return true
    }
```

--EOF--

原文链接：Method Hook Swift