//

//  GlobalDefines.swift

//  ProductTest

//

//  Created by 赖霄冰 on 2016/12/4.

//  Copyright © 2016年 helloworld.com. All rights reserved.

//

import Foundation

let SCALE = UIScreen.main.scale

let SCREEN\_WIDTH = UIScreen.main.bounds.width

let SCREEN\_HEIGHT = UIScreen.main.bounds.height

let STATUS\_BAR\_HEIGHT = UIApplication.shared.statusBarFrame.height

let UIRate = SCREEN\_WIDTH / 375

let UIRateH = SCREEN\_HEIGHT / 667

let SYS\_LANGUAGE\_CHINESE = getCurLanguage() == "zh-Hans-CN"

let XB\_DARK\_TEXT = UIColorHex("333333", 1)

///产品类型枚举

public enum XBProductType:Int {

    case bed

    case pillow

    case ring

}

///错误类型

func errorMsg(\_ url:String, code:Int) -> String {

    var errorMsg = ""

    if url == LOGIN {

        switch code {

        case 1001:

            errorMsg = NSLocalizedString("Email does not exist", comment: "")

        case 1000:

            errorMsg = NSLocalizedString("Invalid Email or Password", comment: "")

        default:break

        }

    } else if url == REGIST {

        switch code {

        case 1002:

            errorMsg = NSLocalizedString("Email already exists", comment: "")

        case 0, 1005:

            errorMsg = NSLocalizedString("Server Error", comment: "")

        default:break

        }

    } else if url == MODIFY {

        switch code {

        case 3005:

            errorMsg = NSLocalizedString("Server Error", comment: "")

        default:break

        }

    } else if url == FORGET {

        if code == 1001 {

            errorMsg = NSLocalizedString("Email does not exist", comment: "")

        }

    } else if url == VERIFY\_CODE {

        switch code {

        case 1006:

            errorMsg = NSLocalizedString("Verification failed", comment: "")

        case 1007:

            errorMsg = NSLocalizedString("Too many verification errors", comment: "")

        case 1008:

            errorMsg = NSLocalizedString("Verification expired", comment: "")

        default:break

        }

    } else if url == MODIFY\_PWD {

        if code == 3005 {

            errorMsg = NSLocalizedString("Server Error", comment: "")

        }

    } else if url == CHECK\_TOKEN {

        if code == 1006 {

            errorMsg = NSLocalizedString("invalid token", comment: "")

        }

    } else if url == QUERY {

        switch code {

        case 2:

            errorMsg = NSLocalizedString("search failed", comment: "")

        case 1001:

            errorMsg = NSLocalizedString("User does not exist", comment: "")

        default:break

        }

    } else if url == APPLY {

        switch code {

        case 1001:

            errorMsg = NSLocalizedString("User does not exist", comment: "")

        case 1002:

            errorMsg = NSLocalizedString("Already applied", comment: "")

        case 1003:

            errorMsg = NSLocalizedString("You can't concern yourself", comment: "")

        default:break

        }

    } else if url == HANDLE {

        if code == 0  {

            errorMsg = NSLocalizedString("operation failed", comment: "")

        }

    } else if url == CANCEL {

        if code == 0 {

            errorMsg = NSLocalizedString("operation failed", comment: "")

        }

    } else if url == DEVICE\_ADD {

        switch code {

        case 1001:

            errorMsg = NSLocalizedString("SN does not exist", comment: "")

        case 1009:

            errorMsg = NSLocalizedString("you can't bind the same type device", comment: "")

        case 1010:

            errorMsg = NSLocalizedString("Has been bound by the others", comment: "")

        case 0:

            errorMsg = NSLocalizedString("bind failed", comment: "")

        default:break

        }

    } else if url == DEVICE\_DELETE {

        if code == 0 {

            errorMsg = NSLocalizedString("unbind failed", comment: "")

        }

    }

    return errorMsg

}

///根据RGBA获取颜色

func RGBA (r:CGFloat, g:CGFloat, b:CGFloat, a:CGFloat) -> UIColor{

    return UIColor (red: r/255.0, green: g/255.0, blue: b/255.0, alpha: a)

}

///获取随机颜色

func SMRandomColor() -> UIColor {

    return RGBA(r: CGFloat(arc4random\_uniform(255)), g: CGFloat(arc4random\_uniform(255)), b: CGFloat(arc4random\_uniform(255)), a: 1)

}

///根据hex字符串获取颜色

func UIColorHex(\_ hex:String, \_ alpha:CGFloat) -> UIColor {

    var cString = hex.trimmingCharacters(in:CharacterSet.whitespacesAndNewlines).uppercased()

    if (cString.hasPrefix("#")) {

        let index = cString.index(cString.startIndex, offsetBy:1)

        cString = cString.substring(from: index)

    }

    if (cString.characters.count != 6) {

        return UIColor.red

    }

    let rIndex = cString.index(cString.startIndex, offsetBy: 2)

    let rString = cString.substring(to: rIndex)

    let otherString = cString.substring(from: rIndex)

    let gIndex = otherString.index(otherString.startIndex, offsetBy: 2)

    let gString = otherString.substring(to: gIndex)

    let bIndex = cString.index(cString.endIndex, offsetBy: -2)

    let bString = cString.substring(from: bIndex)

    var r:CUnsignedInt = 0, g:CUnsignedInt = 0, b:CUnsignedInt = 0;

    Scanner(string: rString).scanHexInt32(&r)

    Scanner(string: gString).scanHexInt32(&g)

    Scanner(string: bString).scanHexInt32(&b)

    return UIColor(red: CGFloat(r) / 255.0, green: CGFloat(g) / 255.0, blue: CGFloat(b) / 255.0, alpha: CGFloat(1))

}

///输入尺寸快速获取常规字体

func UIFontSize(\_ size:CGFloat) -> UIFont {

    return UIFont.systemFont(ofSize: size)

}

///获取当前语言

func getCurLanguage() -> String {

    let languages = UserDefaults.standard.object(forKey: "AppleLanguages") as! [String]

    return languages[0]

}

//MARK: - Notifications

let XBSearchSleepCareHistoryNotification = "kSearchSleepCareHistoryNotification"

let XBDrawFrequecyDidChanged = "kDrawFrequecyDidChanged"

let XBDateSelectViewDidSelectDate = "kDateSelectViewDidSelectDate"

//

//  XBPhotoPickerManager.swift

//  ProductTest

//

//  Created by 赖霄冰 on 2017/1/15.

//  Copyright © 2017年 helloworld.com. All rights reserved.

//

import UIKit

import Photos

import AVFoundation

import MobileCoreServices

import LGAlertView

import SVProgressHUD

public protocol XBPhotoPickerManagerDelegate:NSObjectProtocol {

    func imagePickerDidFinishPickImage(image:UIImage)

}

class XBPhotoPickerManager: NSObject, UIImagePickerControllerDelegate, UINavigationControllerDelegate, LGAlertViewDelegate {

    static let shared = XBPhotoPickerManager()

    var sourceType:UIImagePickerControllerSourceType = .photoLibrary

    var chooser:UIViewController?

    weak var delegate:XBPhotoPickerManagerDelegate?

    func pickIn(vc:UIViewController) {

        chooser = vc

        let alert = LGAlertView(title: NSLocalizedString("Choose a way to select your avatar", comment: ""), message: nil, style: .actionSheet, buttonTitles: [NSLocalizedString("PhotoLibrary", comment: ""),NSLocalizedString("Camera", comment: "")], cancelButtonTitle: NSLocalizedString("Cancel", comment: ""), destructiveButtonTitle: nil, delegate: self)

        alert.showAnimated()

    }

    //MARK: - Auths

    func checkCameraAuth(\_ request:@escaping (Bool)->()) {

        let authStatus = AVCaptureDevice.authorizationStatus(forMediaType: AVMediaTypeVideo)

        if(authStatus == .denied || authStatus == .restricted) {

            UIAlertView(title: NSLocalizedString("This app has no permission to camera" , comment: ""), message: NSLocalizedString("Please open the permission in Privacy Settings, otherwise the function cannot be used.", comment: ""), delegate: nil, cancelButtonTitle: NSLocalizedString("DONE", comment: "")).show()

            return

        }else {

            if authStatus == .authorized {

                request(true)

            } else {

                AVCaptureDevice.requestAccess(forMediaType: AVMediaTypeVideo, completionHandler: { (grand) in

                    if grand {

                        DispatchQueue.main.async(execute: {

                            request(grand)

                        })

                    }

                })

            }

        }

    }

    //MARK: - Camera Access

    func hasAccessTo(\_ sourceType:UIImagePickerControllerSourceType) -> Bool {

        if UIImagePickerController.isSourceTypeAvailable(sourceType) {

            return true

        } else {

            let alert = LGAlertView(title: NSLocalizedString("your device have no access to camera", comment: ""), message: nil, style: .alert, buttonTitles: [NSLocalizedString("DONE", comment: "")], cancelButtonTitle: nil, destructiveButtonTitle: nil)

            alert.showAnimated()

            return false

        }

    }

    //MARK: - Private

    private func open() {

        let imagePicker = UIImagePickerController()

        imagePicker.delegate = self

        imagePicker.allowsEditing = true

        imagePicker.sourceType = sourceType

        imagePicker.mediaTypes = [kUTTypeImage as String]

        chooser?.present(imagePicker, animated: true, completion: nil)

    }

    private func photoLibraryAuth() -> Bool {

        let authStatus = PHPhotoLibrary.authorizationStatus()

        return authStatus != .denied && authStatus != .restricted

    }

    //MARK: - UIImagePickerControllerDelegate

    func imagePickerControllerDidCancel(\_ picker: UIImagePickerController) {

        picker.dismiss(animated: true, completion: nil)

    }

    func imagePickerController(\_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [String : Any]) {

        let mediaType = info[UIImagePickerControllerMediaType] as? String

        if mediaType == kUTTypeImage as String {

            delegate?.imagePickerDidFinishPickImage(image: info[UIImagePickerControllerEditedImage] as! UIImage)

        }

        picker.dismiss(animated: true, completion: nil)

    }

    //MARK: - LGAlertViewDelegate

    func alertView(\_ alertView: LGAlertView, clickedButtonAt index: UInt, title: String?) {

        if index == 0  {

            self.sourceType = .photoLibrary

            guard self.hasAccessTo(.photoLibrary) else { return }

            self.open()

        } else {

            self.sourceType = .camera

            guard self.hasAccessTo(.camera) else {return }

            checkCameraAuth({ (grand) in

                if grand {

                    self.open()

                }

            })

        }

    }

}

//

//  XBOperateUtils.swift

//  ProductTest

//

//  Created by 赖霄冰 on 2017/1/15.

//  Copyright © 2017年 helloworld.com. All rights reserved.

//

import UIKit

import SwiftyJSON

import RealmSwift

import SVProgressHUD

class XBOperateUtils: NSObject {

    private var loginUser:XBUser! {

        if let user = XBUserManager.shared.loginUser() {

            return user

        }

        return nil

    }

    static let shared = XBOperateUtils()

    func login(email:String, pwd:String, success:@escaping (\_ result:Any)->(), failure:@escaping (\_ error:Error)->()) {

        if !XBOperateUtils.validateEmail(email) {

            SVProgressHUD.showError(withStatus: NSLocalizedString("Email format error", comment: ""))

            return

        }

        let params = [

            "email":email,

            "password":pwd

        ]

        SVProgressHUD.show()

        XBNetworking.share.postWithPath(path: LOGIN, paras: params,

                                        success: { (json, message) in

                                            let tk = json[XBData]["token"].stringValue

                                            if json[Code].intValue == 1 {  //登录成功

                                                //登录信息本地缓存

                                                let loginData = LoginData(account: email, password:pwd, token: tk)

                                                Bugly.setUserIdentifier(email)

                                                XBLoginManager.shared.currentLoginData = loginData

                                                //用户信息本地缓存

                                                XBUserManager.shared.addUser(userJson: json[XBData]["userInfo"])

                                                success(message)

                                            } else {   //服务器返回失败原因

                                                failure(NSError(domain: SMErrorDomain, code: json[Code].intValue, userInfo: [kCFErrorLocalizedDescriptionKey as AnyHashable :message]))

                                            }

        }, failure: { error in

            failure(error)

        })

    }

    class func timeComps(\_ timeGap:Double) -> (hour:Int, minute:Int) {

        let h = Int(timeGap / 3600)

        let leftGap = timeGap - Double(h \* 3600)

        let m = Int(leftGap / 60)

        return (h, m)

    }

    //MARK: - 获取某年某月的天数

    func howManyDaysInThisYear(\_ year:Int, \_ month:Int) -> Int{

        if (month == 1) || (month == 3) || (month == 5) || (month == 7) || (month == 8) || (month == 10) || (month == 12) {

            return 31 ;

        }

        if (month == 4) || (month == 6) || (month == 9) || (month == 11) {

            return 30;

        }

        if(year % 4 == 1) || (year % 4 == 2) || (year % 4 == 3)

        {

            return 28;

        }

        if(year % 400 == 0) {

            return 29;

        }

        if(year % 100 == 0) {

            return 28;

        }

        return 29;

    }

    class func validateEmail(\_ email: String) -> Bool {

        let emailString = "[A-Z0-9a-z.\_%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,4}"

        let emailPredicate = NSPredicate(format: "SELF MATCHES %@", emailString)

        return emailPredicate.evaluate(with: email)

    }

    class func validatePassword(\_ pwd:String, confirmPwd:String) -> Bool {

        let pwdRegex = "^[0-9a-zA-Z]{6,16}$"

        let predicate = NSPredicate(format: "SELF MATCHES %@", pwdRegex)

        let isPwd = predicate.evaluate(with: pwd)

        if !isPwd {

            SVProgressHUD.showError(withStatus:

                NSLocalizedString("Password can only be the combination of numbers and letters. It may not be less than 6 or more than 16 characters", comment: ""))

            return false

        }

        if pwd != confirmPwd {

            SVProgressHUD.showError(withStatus: NSLocalizedString("Two passwords are not the same!", comment: ""))

            return false

        }

        return true

    }

    //MARK: - Private

    private func setDefaultRealmForUser(username: String) {

        var config = Realm.Configuration()

        // 使用默认的目录，但是使用用户名来替换默认的文件名

        config.fileURL = config.fileURL!.deletingLastPathComponent()

            .appendingPathComponent("\(username).realm")

        // 将这个配置应用到默认的 Realm 数据库当中

        Realm.Configuration.defaultConfiguration = config

    }

}

//

//  Math.swift

//  ProductTest

//

//  Created by 赖霄冰 on 2017/3/23.

//  Copyright © 2017年 helloworld.com. All rights reserved.

//

import Foundation

func averageOf(numbers:[Double]) -> Double {

    var sum = 0.0

    for n in numbers {

        sum += n

    }

    if numbers.count != 0 {

        return sum / Double(numbers.count)

    } else {

        return 0

    }

}

func maxOf(numbers:[Double]) -> Double {

    if numbers.count == 0 {

        return 0

    }

    var max = numbers[0]

    for n in numbers {

        if n > max {

            max = n

        }

    }

    return max

}

func minOf(numbers:[Double]) -> Double {

    if numbers.count == 0 {

        return 0

    }

    var min = numbers[0]

    for n in numbers {

        if n < min {

            min = n

        }

    }

    return min

}

//

//  XBLog.swift

//  ProductTest

//

//  Created by 赖霄冰 on 2017/4/6.

//  Copyright © 2017年 helloworld.com. All rights reserved.

//

import UIKit

import CocoaLumberjack

class XBLog: NSObject {

    /\*\*

     配置DDLog相关参数

     \*/

    class func configDDLog() {

        //让日志只在debug时输出

        #if DEBUG

            defaultDebugLevel = .verbose

        #else

            defaultDebugLevel = .off

        #endif

        //添加发送日志语句到苹果的日志系统

        DDLog.add(DDASLLogger.sharedInstance)

        //添加发送日志语句到Xcode控制台

        DDLog.add(DDTTYLogger.sharedInstance)

        //允许控制台带颜色

        DDTTYLogger.sharedInstance.colorsEnabled = true

        //设置Info下为蓝色

        DDTTYLogger.sharedInstance.setForegroundColor(UIColor.blue, backgroundColor: UIColor.white, for: .info)

    }

    /\*\*

     得到输出的字符串的格式

     - parameter message:  日志消息的主题

     - parameter file:     日志消息所在的文件，方便调试定位用

     - parameter function: 日志消息所在的方法，方便调试定位用

     - parameter line:     日志消息所在的方法中的行数，方便调试定位用

     - returns: 返回输出的日志字符串

     \*/

    class func getMessage(\_ message: String, file: String , function: String , line: UInt ) -> String {

        //初始化需要返回的字符串

        var returnMessage:String = ""

        //通过file获取文件的名称

        if let className = file.components(separatedBy: "/").last {

            //拼接字符串

            returnMessage = "\n" +

                "className:\(className)\n" +

                " function:\(function)\n" +

                "      ine:\(line)\n" +

            "  message:\(message)"

        }else {

            //拼接字符串

            returnMessage = "\n" +

                " function:\(function)\n" +

                "      ine:\(line)\n" +

            "  message:\(message)"

        }

        return returnMessage

    }

}

/\*\*

 输出Info等级的日志消息

 \*/

func LogInfo(message: String, file: String = #file, function: String = #function, line: UInt = #line) {

    DDLogInfo(XBLog.getMessage(message, file: file, function: function, line: line))

}

/\*\*

 输出Error等级的日志消息

 \*/

func LogError(message: String, file: String = #file, function: String = #function, line: UInt = #line) {

    DDLogError(XBLog.getMessage(message, file: file, function: function, line: line))

}

/\*\*

 输出Debug等级的日志消息

 \*/

func LogDebug(message: String, file: String = #file, function: String = #function, line: UInt = #line) {

    DDLogDebug(XBLog.getMessage(message, file: file, function: function, line: line))

}

/\*\*

 输出Warn等级的日志消息

 \*/

func LogWarn(message: String, file: String = #file, function: String = #function, line: UInt = #line) {

    DDLogWarn(XBLog.getMessage(message, file: file, function: function, line: line))

}

//

//  UIButton.swift

//  ProductTest

//

//  Created by 赖霄冰 on 2016/12/4.

//  Copyright © 2016年 helloworld.com. All rights reserved.

//

import Foundation

extension UIButton  {

    convenience init(image:UIImage?, backImage:UIImage?, color:UIColor?, target:Any, sel:Selector, title:String?) {

        self.init()

        self.setTitle(title, for: .normal)

        self.setImage(image, for: .normal)

        self.setBackgroundImage(backImage, for: .normal)

        self.addTarget(target, action: sel, for: .touchUpInside)

        if (color != nil) {

            self.backgroundColor = color

        }

    }

}

//

//  UIView+Extension.swift

//  ProductTest

//

//  Created by 赖霄冰 on 2016/12/4.

//  Copyright © 2016年 helloworld.com. All rights reserved.

//

import UIKit

extension UIView {

    // x

    var x : CGFloat {

        get {

            return frame.origin.x

        }

        set(newVal) {

            var tmpFrame : CGRect = frame

            tmpFrame.origin.x     = newVal

            frame                 = tmpFrame

        }

    }

    // y

    var y : CGFloat {

        get {

            return frame.origin.y

        }

        set(newVal) {

            var tmpFrame : CGRect = frame

            tmpFrame.origin.y     = newVal

            frame                 = tmpFrame

        }

    }

    // height

    var height : CGFloat {

        get {

            return frame.size.height

        }

        set(newVal) {

            var tmpFrame : CGRect = frame

            tmpFrame.size.height  = newVal

            frame                 = tmpFrame

        }

    }

    // width

    var width : CGFloat {

        get {

            return frame.size.width

        }

        set(newVal) {

            var tmpFrame : CGRect = frame

            tmpFrame.size.width   = newVal

            frame                 = tmpFrame

        }

    }

    // left

    var left : CGFloat {

        get {

            return x

        }

        set(newVal) {

            x = newVal

        }

    }

    // right

    var right : CGFloat {

        get {

            return x + width

        }

        set(newVal) {

            x = newVal - width

        }

    }

    // top

    var top : CGFloat {

        get {

            return y

        }

        set(newVal) {

            y = newVal

        }

    }

    // bottom

    var bottom : CGFloat {

        get {

            return y + height

        }

        set(newVal) {

            y = newVal - height

        }

    }

    var centerX : CGFloat {

        get {

            return center.x

        }

        set(newVal) {

            center = CGPoint(x: newVal, y: center.y)

        }

    }

    var centerY : CGFloat {

        get {

            return center.y

        }

        set(newVal) {

            center = CGPoint(x: center.x, y: newVal)

        }

    }

    var middleX : CGFloat {

        get {

            return width / 2

        }

    }

    var middleY : CGFloat {

        get {

            return height / 2

        }

    }

    var middlePoint : CGPoint {

        get {

            return CGPoint(x: middleX, y: middleY)

        }

    }

    func isShowingOnKeywindow() -> Bool {

        let keyWindow = UIApplication.shared.keyWindow!

        // 把这个view在它的父控件中的frame(即默认的frame)转换成在window的frame

        let convertFrame  = self.superview?.convert(self.frame, to: keyWindow)

        let windowBounds = keyWindow.bounds

        // 判断这个控件是否在主窗口上（即该控件和keyWindow有没有交叉）

        let isOnWindow = convertFrame?.intersects(windowBounds)

        let isShowOnWindow = (self.window == keyWindow) && !self.isHidden && (self.alpha > 0.01) && isOnWindow == true

        return isShowOnWindow

    }

}

//

//  UIImage+Extension.swift

//  ProductTest

//

//  Created by 赖霄冰 on 2017/1/15.

//  Copyright © 2017年 helloworld.com. All rights reserved.

//

import UIKit

extension UIImage {

    func image(with borderWidth:CGFloat, borderColor:UIColor) -> UIImage {

        let ctxSize = CGSize(width: self.size.width + 2 \* borderWidth, height: self.size.height + 2 \* borderWidth)

        UIGraphicsBeginImageContextWithOptions(ctxSize, false, 0)

        var path = UIBezierPath(ovalIn: CGRect(origin: CGPoint.zero, size: ctxSize))

        borderColor.setFill()

        let drawRect = CGRect(x: borderWidth, y: borderWidth, width: self.size.width, height: self.size.height)

        path = UIBezierPath(ovalIn: drawRect)

        path.addClip()

        self.draw(in: drawRect)

        let newImage = UIGraphicsGetImageFromCurrentImageContext()

        UIGraphicsEndImageContext()

        return newImage!

    }

    //获取圆形图片

    func circleImage() -> UIImage? {

        return self.circleImage(to: self.size)

    }

    func circleImage(to size:CGSize) -> UIImage? {

        UIGraphicsBeginImageContext(size)

        let ctx = UIGraphicsGetCurrentContext()

        let rect = CGRect(x: 0, y: 0, width: size.width, height: size.height)

        ctx?.addEllipse(in: rect)

        ctx?.clip()

        self.draw(in: rect)

        let newImage = UIGraphicsGetImageFromCurrentImageContext()

        UIGraphicsEndImageContext()

        return newImage

    }

    ///对指定图片进行拉伸

    func resizableImage(name: String) -> UIImage {

        var normal = UIImage(named: name)!

        let imageWidth = normal.size.width \* 0.5

        let imageHeight = normal.size.height \* 0.5

        normal = resizableImage(withCapInsets: (UIEdgeInsetsMake(imageHeight, imageWidth, imageHeight, imageWidth)))

        return normal

    }

    /\*\*

     \*  压缩上传图片到指定字节

     \*

     \*  image     压缩的图片

     \*  maxLength 压缩后最大字节大小

     \*

     \*  return 压缩后图片的二进制

     \*/

    func compressImage(maxLength: Int) -> Data? {

        let newSize = self.imageSize(with: CGFloat(maxLength))

        let newImage = self.resizeImage(newSize: newSize) ?? self

        var compress:CGFloat = 0.9

        var data = UIImageJPEGRepresentation(newImage, compress)

        let maxSize = maxLength \* 1024

        while data!.count > maxSize && compress > 0.01 {

            compress -= 0.1

            data = UIImageJPEGRepresentation(newImage, compress)

        }

        return data

    }

    /\*\*

     \*  通过指定图片最长边，获得等比例的图片size

     \*

     \*  image       原始图片

     \*  imageLength 图片允许的最长宽度（高度）

     \*

     \*  return 获得等比例的size

     \*/

    func imageSize(with imageLength: CGFloat) -> CGSize {

        var newWidth:CGFloat = 0.0

        var newHeight:CGFloat = 0.0

        let width = self.size.width

        let height = self.size.height

        if (width > imageLength || height > imageLength){

            if (width > height) {

                newWidth = imageLength;

                newHeight = newWidth \* height / width;

            }else if(height > width){

                newHeight = imageLength;

                newWidth = newHeight \* width / height;

            }else{

                newWidth = imageLength;

                newHeight = imageLength;

            }

        }

        return CGSize(width: newWidth, height: newHeight)

    }

    /\*\*

     \*  获得指定size的图片

     \*

     \*  image   原始图片

     \*  newSize 指定的size

     \*

     \*  return 调整后的图片

     \*/

    func resizeImage(newSize: CGSize) -> UIImage? {

        UIGraphicsBeginImageContext(newSize)

        self.draw(in: CGRect(x: 0, y: 0, width: newSize.width, height: newSize.height))

        let newImage = UIGraphicsGetImageFromCurrentImageContext()

        UIGraphicsEndImageContext()

        return newImage

    }

    /// 获取颜色填充而成的图片

    ///

    /// - Parameter color: 颜色

    /// - Returns: 图片

    class func imageWith(\_ color:UIColor) -> UIImage? {

        let rect =  CGRect(x:0, y:0, width:1, height:1)

        UIGraphicsBeginImageContextWithOptions(rect.size, false, 1)

        let ctx = UIGraphicsGetCurrentContext()

        ctx?.setFillColor(color.cgColor)

        ctx?.fill(rect)

        let image = UIGraphicsGetImageFromCurrentImageContext()

        UIGraphicsEndImageContext()

        return image

    }

}

//

//  UIImageView+Extension.swift

//  ProductTest

//

//  Created by 赖霄冰 on 2017/3/1.

//  Copyright © 2017年 helloworld.com. All rights reserved.

//

import UIKit

extension UIImageView {

    func setHeader(url:String?, uid:String) {

        let placeholder = XBUserManager.shared.placeholderForUser(uid: uid)

        if let imageUrl = url {

            self.sd\_setImage(with: URL(string: imageUrl),

                             placeholderImage:placeholder,

                             options: .retryFailed,

                             completed: { (image, err, \_, \_) in

                                if (image != nil && err == nil) {

                                    DispatchQueue.global().async(execute: {

                                        let headerImage = image?.circleImage()

                                        DispatchQueue.main.async(execute: {

                                            self.image = headerImage

                                        })

                                    })

                                }

            })

        } else {

            self.image = placeholder

        }

    }

}

//

//  UITextField+Extension.swift

//  ProductTest

//

//  Created by 赖霄冰 on 2017/4/1.

//  Copyright © 2017年 helloworld.com. All rights reserved.

//

import Foundation

extension UITextField {

    func isBlank() -> Bool {

        if let text = self.text {

           return (text as NSString).length == 0

        }

        return true

    }

}

//

//  UIDevice+Extension.swift

//  ProductTest

//

//  Created by 赖霄冰 on 2017/4/3.

//  Copyright © 2017年 helloworld.com. All rights reserved.

//

import Foundation

public extension UIDevice {

    var modelName: String {

        var systemInfo = utsname()

        uname(&systemInfo)

        let machineMirror = Mirror(reflecting: systemInfo.machine)

        let identifier = machineMirror.children.reduce("") { identifier, element in

            guard let value = element.value as? Int8, value != 0 else { return identifier }

            return identifier + String(UnicodeScalar(UInt8(value)))

        }

        switch identifier {

        case "iPod5,1":                                 return "iPod Touch 5"

        case "iPod7,1":                                 return "iPod Touch 6"

        case "iPhone3,1", "iPhone3,2", "iPhone3,3":     return "iPhone 4"

        case "iPhone4,1":                               return "iPhone 4s"

        case "iPhone5,1", "iPhone5,2":                  return "iPhone 5"

        case "iPhone5,3", "iPhone5,4":                  return "iPhone 5c"

        case "iPhone6,1", "iPhone6,2":                  return "iPhone 5s"

        case "iPhone7,2":                               return "iPhone 6"

        case "iPhone7,1":                               return "iPhone 6 Plus"

        case "iPhone8,1":                               return "iPhone 6s"

        case "iPhone8,2":                               return "iPhone 6s Plus"

        case "iPhone8,4":                               return "iPhone SE"

        case "iPhone9,1":                               return "iPhone 7"

        case "iPhone9,2":                               return "iPhone 7 Plus"

        case "iPad2,1", "iPad2,2", "iPad2,3", "iPad2,4":return "iPad 2"

        case "iPad3,1", "iPad3,2", "iPad3,3":           return "iPad 3"

        case "iPad3,4", "iPad3,5", "iPad3,6":           return "iPad 4"

        case "iPad4,1", "iPad4,2", "iPad4,3":           return "iPad Air"

        case "iPad5,3", "iPad5,4":                      return "iPad Air 2"

        case "iPad2,5", "iPad2,6", "iPad2,7":           return "iPad Mini"

        case "iPad4,4", "iPad4,5", "iPad4,6":           return "iPad Mini 2"

        case "iPad4,7", "iPad4,8", "iPad4,9":           return "iPad Mini 3"

        case "iPad5,1", "iPad5,2":                      return "iPad Mini 4"

        case "iPad6,7", "iPad6,8":                      return "iPad Pro"

        case "AppleTV5,3":                              return "Apple TV"

        case "i386", "x86\_64":                          return "Simulator"

        default:                                        return identifier

        }

    }

}

//

//  NSString+Extension.swift

//  ProductTest

//

//  Created by 赖霄冰 on 2017/4/4.

//  Copyright © 2017年 helloworld.com. All rights reserved.

//

import Foundation

extension String {

    func base64Encoding() -> String? {

        return self.data(using: .utf8)?.base64EncodedString()

    }

    func isChinese() -> Bool {

        let match = "^[\\u4e00-\\u9fa5]"

        let predicate = NSPredicate(format: "SELF matches %@", match)

        return predicate.evaluate(with:self)

    }

}

//

//  NSObject+Extension.swift

//  ProductTest

//

//  Created by 赖霄冰 on 2017/4/6.

//  Copyright © 2017年 helloworld.com. All rights reserved.

//

import Foundation

extension NSObject {

    class func properties\_name() -> [String] {

        var count:UInt32 = 0

        let ivars = class\_copyIvarList(self, &count)

        var properties\_name = [String]()

        for i in 0..<count {

            let ivar = ivars?[Int(i)]

            let ivarName = ivar\_getName(ivar!)

            let nName = String(cString: ivarName!)

            properties\_name.append(nName)

        }

        return properties\_name

    }

}

//

//  XBQRCodeScanViewController.swift

//  ProductTest

//

//  Created by 赖霄冰 on 2017/1/15.

//  Copyright © 2017年 helloworld.com. All rights reserved.

//

import UIKit

import AVFoundation

import SnapKit

class XBQRCodeScanViewController: UIViewController,AVCaptureMetadataOutputObjectsDelegate,

UIAlertViewDelegate {

    var device:AVCaptureDevice!

    var input:AVCaptureDeviceInput!

    var output:AVCaptureMetadataOutput!

    var imageOutput: AVCaptureStillImageOutput!

    var session:AVCaptureSession!

    var preview:AVCaptureVideoPreviewLayer!

    var returnScan:((String) -> ())?

    override func viewDidLoad() {

        super.viewDidLoad()

        setupUI()

        fromCamera()

        //        capture()

    }

    override func didReceiveMemoryWarning() {

        super.didReceiveMemoryWarning()

    }

    override func viewDidAppear(\_ animated: Bool) {

    }

    func setupUI(){

        self.title = "QR Code"

        self.view.backgroundColor = UIColor.clear

        self.navigationController?.navigationBar.titleTextAttributes = [NSForegroundColorAttributeName:UIColor.white]

        self.view.addSubview(centerView)

        self.view.addSubview(scanRectView)

        self.view.addSubview(upView)

        self.view.addSubview(leftView)

        self.view.addSubview(rightView)

        self.view.addSubview(bottomView)

        self.view.addSubview(kuangView)

        self.view.addSubview(textLabel1)

        self.view.addSubview(textLabel2)

        centerView.snp.makeConstraints { (make) in

            make.width.equalTo(200\*UIRate)

            make.height.equalTo(200\*UIRate)

            make.centerX.equalTo(self.view)

            make.centerY.equalTo(self.view)

        }

        scanRectView.snp.makeConstraints { (make) in

            make.width.height.equalTo(centerView)

            make.center.equalTo(centerView)

        }

        upView.snp.makeConstraints { (make) in

            make.width.equalTo(self.view)

            make.height.equalTo(200\*UIRate)

            make.bottom.equalTo(centerView.snp.top)

        }

        leftView.snp.makeConstraints { (make) in

            make.width.equalTo((SCREEN\_WIDTH - 200\*UIRate)/2.0)

            make.height.equalTo(centerView)

            make.left.equalTo(0)

            make.centerY.equalTo(centerView)

        }

        rightView.snp.makeConstraints { (make) in

            make.width.equalTo((SCREEN\_WIDTH - 200\*UIRate)/2.0)

            make.height.equalTo(centerView)

            make.right.equalTo(self.view)

            make.centerY.equalTo(centerView)

        }

        bottomView.snp.makeConstraints { (make) in

            make.width.equalTo(self.view)

            make.height.equalTo(SCREEN\_HEIGHT - 380\*UIRate)

            make.left.equalTo(0)

            make.top.equalTo(centerView.snp.bottom)

        }

        kuangView.snp.makeConstraints { (make) in

            make.width.equalTo(200\*UIRate)

            make.height.equalTo(200\*UIRate)

            make.center.equalTo(centerView)

        }

        textLabel1.snp.makeConstraints { (make) in

            make.top.equalTo(centerView.snp.bottom).offset(15\*UIRate)

            make.centerX.equalTo(self.view)

        }

        textLabel2.snp.makeConstraints { (make) in

            make.top.equalTo(textLabel1.snp.bottom).offset(5\*UIRate)

            make.centerX.equalTo(self.view)

        }

    }

    private lazy var centerView: UIView = {

        let mView = UIView()

        mView.backgroundColor = UIColor.clear

        return mView

    }()

    private lazy var upView: UIView = {

        let mView = UIView()

        mView.backgroundColor = UIColor(white: 0, alpha: 0.4)

        return mView

    }()

    private lazy var leftView: UIView = {

        let mView = UIView()

        mView.backgroundColor = UIColor(white: 0, alpha: 0.4)

        return mView

    }()

    private lazy var rightView: UIView = {

        let mView = UIView()

        mView.backgroundColor = UIColor(white: 0, alpha: 0.4)

        return mView

    }()

    private lazy var bottomView: UIView = {

        let mView = UIView()

        mView.backgroundColor = UIColor(white: 0, alpha: 0.4)

        return mView

    }()

    private lazy var kuangView: UIImageView = {

        let mView = UIImageView()

        mView.image = UIImage(named: "scan\_rect\_316x196")

        return mView

    }()

    //添加中间的探测区域绿框

    private lazy var scanRectView: UIView = {

        let mView = UIView()

        mView.layer.borderColor = UIColor.green.cgColor

        mView.layer.borderWidth = 1;

        return mView

    }()

    private lazy var textLabel1: UILabel = {

        let label = UILabel()

        label.textColor = UIColor.white

        label.font = UIFontSize(15\*UIRate)

        label.text = NSLocalizedString("Put the QR code in the scan area", comment: "")

        return label

    }()

    private lazy var textLabel2: UILabel = {

        let label = UILabel()

        label.textColor = UIColor.white

        label.font = UIFontSize(15\*UIRate)

        return label

    }()

    func fromCamera() {

        do{

            self.session = AVCaptureSession()

            self.device = AVCaptureDevice.defaultDevice(withMediaType: AVMediaTypeVideo)

            try! device.lockForConfiguration()

            device.focusMode = .continuousAutoFocus

            device.unlockForConfiguration()

            self.input = try AVCaptureDeviceInput(device: device)

            self.output = AVCaptureMetadataOutput()

            output.setMetadataObjectsDelegate(self, queue: DispatchQueue.main)

            if UIScreen.main.bounds.size.height<500 {

                self.session.sessionPreset = AVCaptureSessionPreset640x480

            }else{

                self.session.sessionPreset = AVCaptureSessionPresetHigh

            }

            self.session.addInput(self.input)

            self.session.addOutput(self.output)

            self.output.metadataObjectTypes = [AVMetadataObjectTypeQRCode]

            let scanSize:CGSize = CGSize(width: 200\*UIRate, height: 200\*UIRate)

            var scanRect:CGRect = CGRect(x:(SCREEN\_WIDTH-scanSize.width)/2,y:(self.view.height-scanSize.height)/2,width: scanSize.width, height:scanSize.height)

            //计算rectOfInterest 注意x,y交换位置

            scanRect = CGRect(x:scanRect.origin.y/SCREEN\_HEIGHT,

                              y:scanRect.origin.x/SCREEN\_WIDTH,

                              width:scanRect.size.height/SCREEN\_HEIGHT,

                              height:scanRect.size.width/SCREEN\_WIDTH)

            //设置可探测区域

            self.output.rectOfInterest = scanRect

            self.preview = AVCaptureVideoPreviewLayer(session:self.session)

            self.preview.videoGravity = AVLayerVideoGravityResizeAspectFill

            self.preview.frame = UIScreen.main.bounds

            self.view.layer.insertSublayer(self.preview, at: 0)

            //开始捕获

            self.session.startRunning()

        }catch \_ as NSError{

            //打印错误消息

            let errorAlert = UIAlertController(title: "提醒", message: "请在iPhone的\"设置-隐私-相机\"选项中,允许本程序访问您的相机", preferredStyle: UIAlertControllerStyle.alert)

            self.present(errorAlert, animated: true, completion: nil)

            errorAlert.addAction(UIAlertAction.init(title: "好的", style: UIAlertActionStyle.default, handler: { (UIAlertAction) in

                self.dismiss(animated: true, completion: nil)

            }))

        }

    }

    func capture(){

        var imageConnect = AVCaptureConnection()

        for connection in imageOutput.connections{

            for port in (connection as! AVCaptureConnection).inputPorts {

                if (port as! AVCaptureInputPort).mediaType == AVMediaTypeVideo {

                    imageConnect = (connection as? AVCaptureConnection)!

                    break

                }

            }

        }

        imageOutput.captureStillImageAsynchronously(from: imageConnect) { (buffer, error) in

        }

    }

    //摄像头捕获

    func captureOutput(\_ captureOutput: AVCaptureOutput!, didOutputMetadataObjects metadataObjects: [Any]!, from connection: AVCaptureConnection!) {

        var stringValue:String?

        print("扫描结果：\(metadataObjects)")

        if self.returnScan != nil {

            if let metadata = metadataObjects.first {

                let metaObj = metadata as! AVMetadataMachineReadableCodeObject

                self.returnScan!(metaObj.stringValue)

            }

        }

        if metadataObjects.count > 0 {

            let metadataObject = metadataObjects[0] as! AVMetadataMachineReadableCodeObject

            stringValue = metadataObject.stringValue

            if stringValue != nil{

                self.session.stopRunning()

            }

        }

        self.session.stopRunning()

        self.navigationController!.popViewController(animated: true)

    }

}

//

//  XBAlertController.swift

//  ProductTest

//

//  Created by 赖霄冰 on 2017/2/26.

//  Copyright © 2017年 helloworld.com. All rights reserved.

//

import UIKit

class XBAlertController: UIViewController, UIGestureRecognizerDelegate {

    private var alertTitle: String?

    private var alertMessage:String?

    private var alertView:XBAlertView?

    var clickAction:((Int)->())?

    required init?(coder aDecoder: NSCoder) {

        super.init(coder: aDecoder)

    }

    override init(nibName nibNameOrNil: String?, bundle nibBundleOrNil: Bundle?) {

        super.init(nibName: nibNameOrNil, bundle: nibBundleOrNil)

        modalPresentationStyle = .custom

        transitioningDelegate = self

    }

    convenience init(title:String?, message:String?) {

        self.init()

        self.alertTitle = title

        self.alertMessage = message

    }

    override func viewDidLoad() {

        super.viewDidLoad()

        let tap = UITapGestureRecognizer(target: self, action: #selector(close))

        tap.delegate = self

        tap.cancelsTouchesInView = false

        self.view.addGestureRecognizer(tap)

        setup()

    }

    override func viewWillLayoutSubviews() {

        super.viewWillLayoutSubviews()

        alertView?.width = view.width - 42 \* UIRate

        alertView?.centerX = view.centerX

        alertView?.centerY = view.centerY

        alertView?.clickBtnBlock = { [weak self] tag in

            if self?.clickAction != nil {

                self?.close()

                self?.clickAction!(tag)

            }

        }

    }

    //MARK: - Setup

    private func setup() {

        alertView = XBAlertView(title: alertTitle, message: alertMessage)

        view.addSubview(alertView!)

    }

    //MARK: - Gesture

    @objc private func close() {

        self.dismiss(animated: true, completion: nil)

    }

    func gestureRecognizer(\_ gestureRecognizer: UIGestureRecognizer, shouldReceive touch: UITouch) -> Bool {

        return touch.view == self.view

    }

}

extension XBAlertController: UIViewControllerTransitioningDelegate {

    func presentationController(

        forPresented presented: UIViewController,

        presenting: UIViewController?,

        source: UIViewController)

        -> UIPresentationController? {

            return XBDimmingPresentationController( presentedViewController: presented,

                                                    presenting: presenting)

    }

}

//

//  XBDimmingPresentationController.swift

//  ProductTest

//

//  Created by 赖霄冰 on 2017/2/26.

//  Copyright © 2017年 helloworld.com. All rights reserved.

//

import UIKit

class XBDimmingPresentationController: UIPresentationController {

    lazy var dimmingView:UIView! = {

        let dimming = UIView(frame: self.containerView!.bounds)

        dimming.backgroundColor =  UIColor(white: 1, alpha: 0.6)

        return dimming

    }()

    override func presentationTransitionWillBegin() {

        super.presentationTransitionWillBegin()

        self.dimmingView?.alpha = 0

        self.containerView?.insertSubview(dimmingView, at: 0)

        let transition = self.presentedViewController.transitionCoordinator

        transition?.animate(alongsideTransition: { [weak self] (context) in

            self?.dimmingView.alpha = 1

        }, completion: nil)

    }

    override func dismissalTransitionWillBegin() {

        self.presentedViewController.transitionCoordinator?.animate(

            alongsideTransition: { [weak self] (context) in

            self?.dimmingView.alpha = 0

            }, completion: nil)

    }

}

//

//  XBNavigationController.swift

//  ProductTest

//

//  Created by 赖霄冰 on 2016/11/23.

//  Copyright © 2016年 helloworld.com. All rights reserved.

//

import UIKit

class XBNavigationController: UINavigationController, UIGestureRecognizerDelegate {

    var shouldPopBlock:(()->())?

    override func viewDidLoad() {

        super.viewDidLoad()

        navigationBar.setBackgroundImage(UIImage(), for: .default)

        navigationBar.shadowImage = UIImage()

        interactivePopGestureRecognizer?.delegate = self

    }

    override func pushViewController(\_ viewController: UIViewController, animated: Bool) {

        if childViewControllers.count > 0 && !viewController.isKind(of: XBLoginViewController.self) {

            viewController.navigationItem.leftBarButtonItem = UIBarButtonItem.init(image:UIImage(named:"backButton") , style: .plain, target: self, action:#selector(onGoBack))

        }

        self.shouldPopBlock = nil

        super.pushViewController(viewController, animated: animated)

    }

    @objc func onGoBack() {

        if shouldPopBlock != nil {

            shouldPopBlock!()

            shouldPopBlock = nil

        } else {

            popViewController(animated: true)

        }

    }

    //MARK: - UIGestureRecognizerDelegate

    func gestureRecognizerShouldBegin(\_ gestureRecognizer: UIGestureRecognizer) -> Bool {

        return childViewControllers.count > 1 && shouldPopBlock == nil

    }

}

//

//  XBBaseViewController.swift

//  ProductTest

//

//  Created by 赖霄冰 on 2017/2/25.

//  Copyright © 2017年 helloworld.com. All rights reserved.

//

import UIKit

class XBBaseViewController: UIViewController {

    let token = XBLoginManager.shared.currentLoginData!.token

    var naviBackgroundImage:UIImage? {

        return nil

    }

    var naviTitle:String? {

        return nil

    }

    var loginUser:XBUser? {

        if let user = XBUserManager.shared.loginUser() {

            return user

        }

        return nil

    }

    deinit {

        XBNetworking.share.cancel()

        print("deinit")

    }

    private var naviBackground:UIImageView!

    private var titleLabel:UILabel!

    override func viewDidLoad() {

        super.viewDidLoad()

        setupNavigation()

        NotificationCenter.default.addObserver(self, selector: #selector(onStatusFrameChanged(\_:)), name:Notification.Name.UIApplicationWillChangeStatusBarFrame , object: nil)

    }

    private func setupNavigation() {

        naviBackground = UIImageView(frame: CGRect(x: 0, y: 0, width: view.width, height:naviBackgroundImage!.size.height\*UIRate))

        naviBackground.image = naviBackgroundImage

        titleLabel = UILabel()

        titleLabel.text = naviTitle

        titleLabel.textColor = UIColor.black

        titleLabel.numberOfLines = 0

        titleLabel.textAlignment = .center

        titleLabel.font = UIFontSize(27\*UIRate)

        titleLabel.sizeToFit()

        titleLabel.centerX = naviBackground.centerX

        titleLabel.top = UIRate \* 24

        naviBackground.addSubview(titleLabel)

        view.addSubview(naviBackground)

    }

    //MARK: - Notification

    func onStatusFrameChanged(\_ aNote:Notification) {

    }

}

//

//  MBProgressHUD.h

//  Version 1.0.0

//  Created by Matej Bukovinski on 2.4.09.

//

// This code is distributed under the terms and conditions of the MIT license.

// Copyright © 2009-2016 Matej Bukovinski

//

// Permission is hereby granted, free of charge, to any person obtaining a copy

// of this software and associated documentation files (the "Software"), to deal

// in the Software without restriction, including without limitation the rights

// to use, copy, modify, merge, publish, distribute, sublicense, and/or sell

// copies of the Software, and to permit persons to whom the Software is

// furnished to do so, subject to the following conditions:

//

// The above copyright notice and this permission notice shall be included in

// all copies or substantial portions of the Software.

//

// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR

// IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,

// FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE

// AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER

// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,

// OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN

// THE SOFTWARE.

#import <Foundation/Foundation.h>

#import <UIKit/UIKit.h>

#import <CoreGraphics/CoreGraphics.h>

@class MBBackgroundView;

@protocol MBProgressHUDDelegate;

extern CGFloat const MBProgressMaxOffset;

typedef NS\_ENUM(NSInteger, MBProgressHUDMode) {

    /// UIActivityIndicatorView.

    MBProgressHUDModeIndeterminate,

    /// A round, pie-chart like, progress view.

    MBProgressHUDModeDeterminate,

    /// Horizontal progress bar.

    MBProgressHUDModeDeterminateHorizontalBar,

    /// Ring-shaped progress view.

    MBProgressHUDModeAnnularDeterminate,

    /// Shows a custom view.

    MBProgressHUDModeCustomView,

    /// Shows only labels.

    MBProgressHUDModeText

};

typedef NS\_ENUM(NSInteger, MBProgressHUDAnimation) {

    /// Opacity animation

    MBProgressHUDAnimationFade,

    /// Opacity + scale animation (zoom in when appearing zoom out when disappearing)

    MBProgressHUDAnimationZoom,

    /// Opacity + scale animation (zoom out style)

    MBProgressHUDAnimationZoomOut,

    /// Opacity + scale animation (zoom in style)

    MBProgressHUDAnimationZoomIn

};

typedef NS\_ENUM(NSInteger, MBProgressHUDBackgroundStyle) {

    /// Solid color background

    MBProgressHUDBackgroundStyleSolidColor,

    /// UIVisualEffectView or UIToolbar.layer background view

    MBProgressHUDBackgroundStyleBlur

};

typedef void (^MBProgressHUDCompletionBlock)();

NS\_ASSUME\_NONNULL\_BEGIN

/\*\*

 \* Displays a simple HUD window containing a progress indicator and two optional labels for short messages.

 \*

 \* This is a simple drop-in class for displaying a progress HUD view similar to Apple's private UIProgressHUD class.

 \* The MBProgressHUD window spans over the entire space given to it by the initWithFrame: constructor and catches all

 \* user input on this region, thereby preventing the user operations on components below the view.

 \*

 \* @note To still allow touches to pass through the HUD, you can set hud.userInteractionEnabled = NO.

 \* @attention MBProgressHUD is a UI class and should therefore only be accessed on the main thread.

 \*/

@interface MBProgressHUD : UIView

/\*\*

 \* Creates a new HUD, adds it to provided view and shows it. The counterpart to this method is hideHUDForView:animated:.

 \*

 \* @note This method sets removeFromSuperViewOnHide. The HUD will automatically be removed from the view hierarchy when hidden.

 \*

 \* @param view The view that the HUD will be added to

 \* @param animated If set to YES the HUD will appear using the current animationType. If set to NO the HUD will not use

 \* animations while appearing.

 \* @return A reference to the created HUD.

 \*

 \* @see hideHUDForView:animated:

 \* @see animationType

 \*/

+ (instancetype)showHUDAddedTo:(UIView \*)view animated:(BOOL)animated;

/// @name Showing and hiding

/\*\*

 \* Finds the top-most HUD subview and hides it. The counterpart to this method is showHUDAddedTo:animated:.

 \*

 \* @note This method sets removeFromSuperViewOnHide. The HUD will automatically be removed from the view hierarchy when hidden.

 \*

 \* @param view The view that is going to be searched for a HUD subview.

 \* @param animated If set to YES the HUD will disappear using the current animationType. If set to NO the HUD will not use

 \* animations while disappearing.

 \* @return YES if a HUD was found and removed, NO otherwise.

 \*

 \* @see showHUDAddedTo:animated:

 \* @see animationType

 \*/

+ (BOOL)hideHUDForView:(UIView \*)view animated:(BOOL)animated;

/\*\*

 \* Finds the top-most HUD subview and returns it.

 \*

 \* @param view The view that is going to be searched.

 \* @return A reference to the last HUD subview discovered.

 \*/

+ (nullable MBProgressHUD \*)HUDForView:(UIView \*)view;

/\*\*

 \* A convenience constructor that initializes the HUD with the view's bounds. Calls the designated constructor with

 \* view.bounds as the parameter.

 \*

 \* @param view The view instance that will provide the bounds for the HUD. Should be the same instance as

 \* the HUD's superview (i.e., the view that the HUD will be added to).

 \*/

- (instancetype)initWithView:(UIView \*)view;

/\*\*

 \* Displays the HUD.

 \*

 \* @note You need to make sure that the main thread completes its run loop soon after this method call so that

 \* the user interface can be updated. Call this method when your task is already set up to be executed in a new thread

 \* (e.g., when using something like NSOperation or making an asynchronous call like NSURLRequest).

 \*

 \* @param animated If set to YES the HUD will appear using the current animationType. If set to NO the HUD will not use

 \* animations while appearing.

 \*

 \* @see animationType

 \*/

- (void)showAnimated:(BOOL)animated;

/\*\*

 \* Hides the HUD. This still calls the hudWasHidden: delegate. This is the counterpart of the show: method. Use it to

 \* hide the HUD when your task completes.

 \*

 \* @param animated If set to YES the HUD will disappear using the current animationType. If set to NO the HUD will not use

 \* animations while disappearing.

 \*

 \* @see animationType

 \*/

- (void)hideAnimated:(BOOL)animated;

/\*\*

 \* Hides the HUD after a delay. This still calls the hudWasHidden: delegate. This is the counterpart of the show: method. Use it to

 \* hide the HUD when your task completes.

 \*

 \* @param animated If set to YES the HUD will disappear using the current animationType. If set to NO the HUD will not use

 \* animations while disappearing.

 \* @param delay Delay in seconds until the HUD is hidden.

 \*

 \* @see animationType

 \*/

- (void)hideAnimated:(BOOL)animated afterDelay:(NSTimeInterval)delay;

/\*\*

 \* The HUD delegate object. Receives HUD state notifications.

 \*/

@property (weak, nonatomic) id<MBProgressHUDDelegate> delegate;

/\*\*

 \* Called after the HUD is hiden.

 \*/

@property (copy, nullable) MBProgressHUDCompletionBlock completionBlock;

/\*

 \* Grace period is the time (in seconds) that the invoked method may be run without

 \* showing the HUD. If the task finishes before the grace time runs out, the HUD will

 \* not be shown at all.

 \* This may be used to prevent HUD display for very short tasks.

 \* Defaults to 0 (no grace time).

 \*/

@property (assign, nonatomic) NSTimeInterval graceTime;

/\*\*

 \* The minimum time (in seconds) that the HUD is shown.

 \* This avoids the problem of the HUD being shown and than instantly hidden.

 \* Defaults to 0 (no minimum show time).

 \*/

@property (assign, nonatomic) NSTimeInterval minShowTime;

/\*\*

 \* Removes the HUD from its parent view when hidden.

 \* Defaults to NO.

 \*/

@property (assign, nonatomic) BOOL removeFromSuperViewOnHide;

/// @name Appearance

/\*\*

 \* MBProgressHUD operation mode. The default is MBProgressHUDModeIndeterminate.

 \*/

@property (assign, nonatomic) MBProgressHUDMode mode;

/\*\*

 \* A color that gets forwarded to all labels and supported indicators. Also sets the tintColor

 \* for custom views on iOS 7+. Set to nil to manage color individually.

 \* Defaults to semi-translucent black on iOS 7 and later and white on earlier iOS versions.

 \*/

@property (strong, nonatomic, nullable) UIColor \*contentColor UI\_APPEARANCE\_SELECTOR;

/\*\*

 \* The animation type that should be used when the HUD is shown and hidden.

 \*/

@property (assign, nonatomic) MBProgressHUDAnimation animationType UI\_APPEARANCE\_SELECTOR;

/\*\*

 \* The bezel offset relative to the center of the view. You can use MBProgressMaxOffset

 \* and -MBProgressMaxOffset to move the HUD all the way to the screen edge in each direction.

 \* E.g., CGPointMake(0.f, MBProgressMaxOffset) would position the HUD centered on the bottom edge.

 \*/

@property (assign, nonatomic) CGPoint offset UI\_APPEARANCE\_SELECTOR;

/\*\*

 \* The amount of space between the HUD edge and the HUD elements (labels, indicators or custom views).

 \* This also represents the minimum bezel distance to the edge of the HUD view.

 \* Defaults to 20.f

 \*/

@property (assign, nonatomic) CGFloat margin UI\_APPEARANCE\_SELECTOR;

/\*\*

 \* The minimum size of the HUD bezel. Defaults to CGSizeZero (no minimum size).

 \*/

@property (assign, nonatomic) CGSize minSize UI\_APPEARANCE\_SELECTOR;

/\*\*

 \* Force the HUD dimensions to be equal if possible.

 \*/

@property (assign, nonatomic, getter = isSquare) BOOL square UI\_APPEARANCE\_SELECTOR;

/\*\*

 \* When enabled, the bezel center gets slightly affected by the device accelerometer data.

 \* Has no effect on iOS < 7.0. Defaults to YES.

 \*/

@property (assign, nonatomic, getter=areDefaultMotionEffectsEnabled) BOOL defaultMotionEffectsEnabled UI\_APPEARANCE\_SELECTOR;

/// @name Progress

/\*\*

 \* The progress of the progress indicator, from 0.0 to 1.0. Defaults to 0.0.

 \*/

@property (assign, nonatomic) float progress;

/// @name ProgressObject

/\*\*

 \* The NSProgress object feeding the progress information to the progress indicator.

 \*/

@property (strong, nonatomic, nullable) NSProgress \*progressObject;

/// @name Views

/\*\*

 \* The view containing the labels and indicator (or customView).

 \*/

@property (strong, nonatomic, readonly) MBBackgroundView \*bezelView;

/\*\*

 \* View covering the entire HUD area, placed behind bezelView.

 \*/

@property (strong, nonatomic, readonly) MBBackgroundView \*backgroundView;

/\*\*

 \* The UIView (e.g., a UIImageView) to be shown when the HUD is in MBProgressHUDModeCustomView.

 \* The view should implement intrinsicContentSize for proper sizing. For best results use approximately 37 by 37 pixels.

 \*/

@property (strong, nonatomic, nullable) UIView \*customView;

/\*\*

 \* A label that holds an optional short message to be displayed below the activity indicator. The HUD is automatically resized to fit

 \* the entire text.

 \*/

@property (strong, nonatomic, readonly) UILabel \*label;

/\*\*

 \* A label that holds an optional details message displayed below the labelText message. The details text can span multiple lines.

 \*/

@property (strong, nonatomic, readonly) UILabel \*detailsLabel;

/\*\*

 \* A button that is placed below the labels. Visible only if a target / action is added.

 \*/

@property (strong, nonatomic, readonly) UIButton \*button;

@end

@protocol MBProgressHUDDelegate <NSObject>

@optional

/\*\*

 \* Called after the HUD was fully hidden from the screen.

 \*/

- (void)hudWasHidden:(MBProgressHUD \*)hud;

@end

/\*\*

 \* A progress view for showing definite progress by filling up a circle (pie chart).

 \*/

@interface MBRoundProgressView : UIView

/\*\*

 \* Progress (0.0 to 1.0)

 \*/

@property (nonatomic, assign) float progress;

/\*\*

 \* Indicator progress color.

 \* Defaults to white [UIColor whiteColor].

 \*/

@property (nonatomic, strong) UIColor \*progressTintColor;

/\*\*

 \* Indicator background (non-progress) color.

 \* Only applicable on iOS versions older than iOS 7.

 \* Defaults to translucent white (alpha 0.1).

 \*/

@property (nonatomic, strong) UIColor \*backgroundTintColor;

/\*

 \* Display mode - NO = round or YES = annular. Defaults to round.

 \*/

@property (nonatomic, assign, getter = isAnnular) BOOL annular;

@end

/\*\*

 \* A flat bar progress view.

 \*/

@interface MBBarProgressView : UIView

/\*\*

 \* Progress (0.0 to 1.0)

 \*/

@property (nonatomic, assign) float progress;

/\*\*

 \* Bar border line color.

 \* Defaults to white [UIColor whiteColor].

 \*/

@property (nonatomic, strong) UIColor \*lineColor;

/\*\*

 \* Bar background color.

 \* Defaults to clear [UIColor clearColor];

 \*/

@property (nonatomic, strong) UIColor \*progressRemainingColor;

/\*\*

 \* Bar progress color.

 \* Defaults to white [UIColor whiteColor].

 \*/

@property (nonatomic, strong) UIColor \*progressColor;

@end

@interface MBBackgroundView : UIView

/\*\*

 \* The background style.

 \* Defaults to MBProgressHUDBackgroundStyleBlur on iOS 7 or later and MBProgressHUDBackgroundStyleSolidColor otherwise.

 \* @note Due to iOS 7 not supporting UIVisualEffectView, the blur effect differs slightly between iOS 7 and later versions.

 \*/

@property (nonatomic) MBProgressHUDBackgroundStyle style;

/\*\*

 \* The background color or the blur tint color.

 \* @note Due to iOS 7 not supporting UIVisualEffectView, the blur effect differs slightly between iOS 7 and later versions.

 \*/

@property (nonatomic, strong) UIColor \*color;

@end

@interface MBProgressHUD (Deprecated)

+ (NSArray \*)allHUDsForView:(UIView \*)view \_\_attribute\_\_((deprecated("Store references when using more than one HUD per view.")));

+ (NSUInteger)hideAllHUDsForView:(UIView \*)view animated:(BOOL)animated \_\_attribute\_\_((deprecated("Store references when using more than one HUD per view.")));

- (id)initWithWindow:(UIWindow \*)window \_\_attribute\_\_((deprecated("Use initWithView: instead.")));

- (void)show:(BOOL)animated \_\_attribute\_\_((deprecated("Use showAnimated: instead.")));

- (void)hide:(BOOL)animated \_\_attribute\_\_((deprecated("Use hideAnimated: instead.")));

- (void)hide:(BOOL)animated afterDelay:(NSTimeInterval)delay \_\_attribute\_\_((deprecated("Use hideAnimated:afterDelay: instead.")));

- (void)showWhileExecuting:(SEL)method onTarget:(id)target withObject:(id)object animated:(BOOL)animated \_\_attribute\_\_((deprecated("Use GCD directly.")));

- (void)showAnimated:(BOOL)animated whileExecutingBlock:(dispatch\_block\_t)block \_\_attribute\_\_((deprecated("Use GCD directly.")));

- (void)showAnimated:(BOOL)animated whileExecutingBlock:(dispatch\_block\_t)block completionBlock:(nullable MBProgressHUDCompletionBlock)completion \_\_attribute\_\_((deprecated("Use GCD directly.")));

- (void)showAnimated:(BOOL)animated whileExecutingBlock:(dispatch\_block\_t)block onQueue:(dispatch\_queue\_t)queue \_\_attribute\_\_((deprecated("Use GCD directly.")));

- (void)showAnimated:(BOOL)animated whileExecutingBlock:(dispatch\_block\_t)block onQueue:(dispatch\_queue\_t)queue

     completionBlock:(nullable MBProgressHUDCompletionBlock)completion \_\_attribute\_\_((deprecated("Use GCD directly.")));

@property (assign) BOOL taskInProgress \_\_attribute\_\_((deprecated("No longer needed.")));

@property (nonatomic, copy) NSString \*labelText \_\_attribute\_\_((deprecated("Use label.text instead.")));

@property (nonatomic, strong) UIFont \*labelFont \_\_attribute\_\_((deprecated("Use label.font instead.")));

@property (nonatomic, strong) UIColor \*labelColor \_\_attribute\_\_((deprecated("Use label.textColor instead.")));

@property (nonatomic, copy) NSString \*detailsLabelText \_\_attribute\_\_((deprecated("Use detailsLabel.text instead.")));

@property (nonatomic, strong) UIFont \*detailsLabelFont \_\_attribute\_\_((deprecated("Use detailsLabel.font instead.")));

@property (nonatomic, strong) UIColor \*detailsLabelColor \_\_attribute\_\_((deprecated("Use detailsLabel.textColor instead.")));

@property (assign, nonatomic) CGFloat opacity \_\_attribute\_\_((deprecated("Customize bezelView properties instead.")));

@property (strong, nonatomic) UIColor \*color \_\_attribute\_\_((deprecated("Customize the bezelView color instead.")));

@property (assign, nonatomic) CGFloat xOffset \_\_attribute\_\_((deprecated("Set offset.x instead.")));

@property (assign, nonatomic) CGFloat yOffset \_\_attribute\_\_((deprecated("Set offset.y instead.")));

@property (assign, nonatomic) CGFloat cornerRadius \_\_attribute\_\_((deprecated("Set bezelView.layer.cornerRadius instead.")));

@property (assign, nonatomic) BOOL dimBackground \_\_attribute\_\_((deprecated("Customize HUD background properties instead.")));

@property (strong, nonatomic) UIColor \*activityIndicatorColor \_\_attribute\_\_((deprecated("Use UIAppearance to customize UIActivityIndicatorView. E.g.: [UIActivityIndicatorView appearanceWhenContainedIn:[MBProgressHUD class], nil].color = [UIColor redColor];")));

@property (atomic, assign, readonly) CGSize size \_\_attribute\_\_((deprecated("Get the bezelView.frame.size instead.")));

@end

NS\_ASSUME\_NONNULL\_END

//

//  Toast.swift

//  Toast-Swift

//

//  Copyright (c) 2015 Charles Scalesse.

//

//  Permission is hereby granted, free of charge, to any person obtaining a

//  copy of this software and associated documentation files (the

//  "Software"), to deal in the Software without restriction, including

//  without limitation the rights to use, copy, modify, merge, publish,

//  distribute, sublicense, and/or sell copies of the Software, and to

//  permit persons to whom the Software is furnished to do so, subject to

//  the following conditions:

//

//  The above copyright notice and this permission notice shall be included

//  in all copies or substantial portions of the Software.

//

//  THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS

//  OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF

//  MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.

//  IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY

//  CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,

//  TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE

//  SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

import UIKit

import ObjectiveC

public enum ToastPosition {

    case top

    case center

    case bottom

}

/\*\*

 Toast is a Swift extension that adds toast notifications to the `UIView` object class.

 It is intended to be simple, lightweight, and easy to use. Most toast notifications

 can be triggered with a single line of code.

 The `makeToast` methods create a new view and then display it as toast.

 The `showToast` methods display any view as toast.

 \*/

public extension UIView {

    /\*\*

     Keys used for associated objects.

     \*/

    private struct ToastKeys {

        static var Timer        = "CSToastTimerKey"

        static var Duration     = "CSToastDurationKey"

        static var Position     = "CSToastPositionKey"

        static var Completion   = "CSToastCompletionKey"

        static var ActiveToast  = "CSToastActiveToastKey"

        static var ActivityView = "CSToastActivityViewKey"

        static var Queue        = "CSToastQueueKey"

    }

    /\*\*

     Swift closures can't be directly associated with objects via the

     Objective-C runtime, so the (ugly) solution is to wrap them in a

     class that can be used with associated objects.

     \*/

    private class ToastCompletionWrapper {

        var completion: ((Bool) -> Void)?

        init(\_ completion: ((Bool) -> Void)?) {

            self.completion = completion

        }

    }

    private enum ToastError: Error {

        case insufficientData

    }

    private var queue: NSMutableArray {

        get {

            if let queue = objc\_getAssociatedObject(self, &ToastKeys.Queue) as? NSMutableArray {

                return queue

            } else {

                let queue = NSMutableArray()

                objc\_setAssociatedObject(self, &ToastKeys.Queue, queue, .OBJC\_ASSOCIATION\_RETAIN\_NONATOMIC)

                return queue

            }

        }

    }

    // MARK: - Make Toast Methods

    /\*\*

     Creates and presents a new toast view with a message and displays it with the

     default duration and position. Styled using the shared style.

     @param message The message to be displayed

    \*/

    public func makeToast(\_ message: String) {

        self.makeToast(message, duration: ToastManager.shared.duration, position: ToastManager.shared.position)

    }

    /\*\*

     Creates and presents a new toast view with a message. Duration and position

     can be set explicitly. Styled using the shared style.

     @param message The message to be displayed

     @param duration The toast duration

     @param position The toast's position

     \*/

    public func makeToast(\_ message: String, duration: TimeInterval, position: ToastPosition) {

        self.makeToast(message, duration: duration, position: position, style: nil)

    }

    /\*\*

     Creates and presents a new toast view with a message. Duration and position

     can be set explicitly. Styled using the shared style.

     @param message The message to be displayed

     @param duration The toast duration

     @param position The toast's center point

     \*/

    public func makeToast(\_ message: String, duration: TimeInterval, position: CGPoint) {

        self.makeToast(message, duration: duration, position: position, style: nil)

    }

    /\*\*

     Creates and presents a new toast view with a message. Duration, position, and

     style can be set explicitly.

     @param message The message to be displayed

     @param duration The toast duration

     @param position The toast's position

     @param style The style. The shared style will be used when nil

     \*/

    public func makeToast(\_ message: String, duration: TimeInterval, position: ToastPosition, style: ToastStyle?) {

        self.makeToast(message, duration: duration, position: position, title: nil, image: nil, style: style, completion: nil)

    }

    /\*\*

     Creates and presents a new toast view with a message. Duration, position, and

     style can be set explicitly.

     @param message The message to be displayed

     @param duration The toast duration

     @param position The toast's center point

     @param style The style. The shared style will be used when nil

     \*/

    public func makeToast(\_ message: String, duration: TimeInterval, position: CGPoint, style: ToastStyle?) {

        self.makeToast(message, duration: duration, position: position, title: nil, image: nil, style: style, completion: nil)

    }

    /\*\*

     Creates and presents a new toast view with a message, title, and image. Duration,

     position, and style can be set explicitly. The completion closure executes when the

     toast completes presentation. `didTap` will be `true` if the toast view was dismissed

     from a tap.

     @param message The message to be displayed

     @param duration The toast duration

     @param position The toast's position

     @param title The title

     @param image The image

     @param style The style. The shared style will be used when nil

     @param completion The completion closure, executed after the toast view disappears.

            didTap will be `true` if the toast view was dismissed from a tap.

     \*/

    public func makeToast(\_ message: String?, duration: TimeInterval, position: ToastPosition, title: String?, image: UIImage?, style: ToastStyle?, completion: ((\_ didTap: Bool) -> Void)?) {

        var toastStyle = ToastManager.shared.style

        if let style = style {

           toastStyle = style

        }

        do {

            let toast = try self.toastViewForMessage(message, title: title, image: image, style: toastStyle)

            self.showToast(toast, duration: duration, position: position, completion: completion)

        } catch ToastError.insufficientData {

            print("Error: message, title, and image are all nil")

        } catch {}

    }

    /\*\*

     Creates and presents a new toast view with a message, title, and image. Duration,

     position, and style can be set explicitly. The completion closure executes when the

     toast completes presentation. `didTap` will be `true` if the toast view was dismissed

     from a tap.

     @param message The message to be displayed

     @param duration The toast duration

     @param position The toast's center point

     @param title The title

     @param image The image

     @param style The style. The shared style will be used when nil

     @param completion The completion closure, executed after the toast view disappears.

            didTap will be `true` if the toast view was dismissed from a tap.

     \*/

    public func makeToast(\_ message: String?, duration: TimeInterval, position: CGPoint, title: String?, image: UIImage?, style: ToastStyle?, completion: ((\_ didTap: Bool) -> Void)?) {

        var toastStyle = ToastManager.shared.style

        if let style = style {

            toastStyle = style

        }

        do {

            let toast = try self.toastViewForMessage(message, title: title, image: image, style: toastStyle)

            self.showToast(toast, duration: duration, position: position, completion: completion)

        } catch ToastError.insufficientData {

            print("Error: message, title, and image cannot all be nil")

        } catch {}

    }

    // MARK: - Show Toast Methods

    /\*\*

    Displays any view as toast using the default duration and position.

    @param toast The view to be displayed as toast

    \*/

    public func showToast(\_ toast: UIView) {

        self.showToast(toast, duration: ToastManager.shared.duration, position: ToastManager.shared.position, completion: nil)

    }

    /\*\*

     Displays any view as toast at a provided position and duration. The completion closure

     executes when the toast view completes. `didTap` will be `true` if the toast view was

     dismissed from a tap.

     @param toast The view to be displayed as toast

     @param duration The notification duration

     @param position The toast's position

     @param completion The completion block, executed after the toast view disappears.

     didTap will be `true` if the toast view was dismissed from a tap.

     \*/

    public func showToast(\_ toast: UIView, duration: TimeInterval, position: ToastPosition, completion: ((\_ didTap: Bool) -> Void)?) {

        let point = self.centerPointForPosition(position, toast: toast)

        self.showToast(toast, duration: duration, position: point, completion: completion)

    }

    /\*\*

     Displays any view as toast at a provided position and duration. The completion closure

     executes when the toast view completes. `didTap` will be `true` if the toast view was

     dismissed from a tap.

     @param toast The view to be displayed as toast

     @param duration The notification duration

     @param position The toast's center point

     @param completion The completion block, executed after the toast view disappears.

     didTap will be `true` if the toast view was dismissed from a tap.

     \*/

    public func showToast(\_ toast: UIView, duration: TimeInterval, position: CGPoint, completion: ((\_ didTap: Bool) -> Void)?) {

        objc\_setAssociatedObject(toast, &ToastKeys.Completion, ToastCompletionWrapper(completion), .OBJC\_ASSOCIATION\_RETAIN\_NONATOMIC);

        if let \_ = objc\_getAssociatedObject(self, &ToastKeys.ActiveToast) as? UIView, ToastManager.shared.queueEnabled {

            objc\_setAssociatedObject(toast, &ToastKeys.Duration, NSNumber(value: duration), .OBJC\_ASSOCIATION\_RETAIN\_NONATOMIC);

            objc\_setAssociatedObject(toast, &ToastKeys.Position, NSValue(cgPoint: position), .OBJC\_ASSOCIATION\_RETAIN\_NONATOMIC);

            self.queue.add(toast)

        } else {