c动画key:

cornerRadius--🡪设置圆角动画

path --🡪 CAShapeLayer来说的

transform

position

(\_\_bridge type)host ----表示把host转化成type类型

s1.Labs()取绝对值

2.NSNumberFormatter

NSNumberFormatter \*numberFormatter = [[NSNumberFormatter alloc] init];

[numberFormatter setPositiveFormat:@"###,##0.00;"];

NSString \*formattedNumberString = [numberFormatter stringFromNumber:[NSNumber numberWithDouble:value]];

输出的格式：1，000.00

#代表占位，但是不能用0占位，因为用0的时候，位数不够的时候，会用0填充，而#不会

3.NSMutableAttributedString

[attributedString addAttribute:NSParagraphStyleAttributeName value:paragraphStyle range:range];

4.NSMutableParagraphStyle

5.NSScanner

6.NSPredicate谓词

判断是否含有数字

用NSPredicate在NSString中查询数字：

NSString \*test = @"[0-9]";

NSString \*testString = @"ab9";

NSPredicate \*predicate = [NSPredicate predicateWithFormat:@"(%@ CONTAINS[c] %@)", test,testString];

BOOL bResult = [predicate evaluateWithObject:testString]

但是只搜索到到了开始的数字，我也试过@"[0-9]+"和@"[0-9]\*"这样，但是都得不到正确结果

7.一般一个项目的框架：可以通过创建一个公共根TabBarViewController，然后设置改tabBarViewController的self.viewControllers = @[\_firstPageVC, \_buyVC, \_sellVC, \_profileVC];设置子试图控制器，在通过重写- (void)setSelectedIndex:(NSUInteger)selectedIndex方法，来改变子视图控制器的切换视图控制器的导航条样式，他们公用一个tabBarViewController的navigationBar

8.通过代码约束来控制视图的现实，在调用其frame的时候，为{0，0，0，0}

9.模拟器手势调出快捷键：

option + 左键 两指方向相反

option + shift + 左键 两指方向相同

10.用类方法，初始化Nib对象

+ (instancetype)headViewWith:(WNXHomeModel \*)headModel

{

WNXRmndHeadView \*headView = [[NSBundle mainBundle]loadNibNamed:NSStringFromClass(self) owner:nil options:nil].lastObject;

headView.headModel = headModel;

return headView;

}

11. 用CADisplayLink可以实现不停重绘。（CADisplayLink是与屏幕刷新的频率是一致的）

例子：

CADisplayLink\* gameTimer;

gameTimer = [CADisplayLink displayLinkWithTarget:self

selector:@selector(updateDisplay:)];

[gameTimer addToRunLoop:[NSRunLoop currentRunLoop forMode:NSDefaultRunLoopMode];

[displayLink setFrameInterval: self.interval];//设置帧率

取消：[displayLink invalidate];

displayLink = nil;

12.定时器取消

[self.timer invalidate]

self.timer = nil

13.动画组，与非动画多个动画

动画组：动画组中的动画是同步执行的，若单个动画不设置时间，则动画一直从假如动画组开始，一直到结束，默认的周期是和动画组的周期一致的；若有动画时间，则先在该动画的周期内执行完该动画，然后在重复执行该动画，直至该动画结束

13.键盘弹起，遮挡问题

[[NSNotificationCenter defaultCenter]addObserver:self selector:@selector(keyboardChange:) name:UIKeyboardWillShowNotification object:nil];

[[NSNotificationCenter defaultCenter]addObserver:self selector:@selector(keyboardChange:) name:UIKeyboardWillHideNotification object:nil];

-(void)keyboardChange:(NSNotification \*)notification

{

NSDictionary \*userInfo = [notification userInfo];

NSTimeInterval animationDuration;

UIViewAnimationCurve animationCurve;

CGRect keyboardEndFrame;

[[userInfo objectForKey:UIKeyboardAnimationCurveUserInfoKey] getValue:&animationCurve];//获得键盘弹起的动画

[[userInfo objectForKey:UIKeyboardAnimationDurationUserInfoKey] getValue:&animationDuration];//获取键盘弹起动画的周期

[[userInfo objectForKey:UIKeyboardFrameEndUserInfoKey] getValue:&keyboardEndFrame];//获取键盘弹起动画结束的frame

[UIView beginAnimations:nil context:nil];

[UIView setAnimationDuration:animationDuration];

[UIView setAnimationCurve:animationCurve];

//adjust ChatTableView's height

if (notification.name == UIKeyboardWillShowNotification) {

self.bottomConstraint.constant = keyboardEndFrame.size.height+40;

}else{

self.bottomConstraint.constant = 40;

}

[self.view layoutIfNeeded];

//adjust UUInputFunctionView's originPoint

CGRect newFrame = IFView.frame;

newFrame.origin.y = keyboardEndFrame.origin.y - newFrame.size.height;

IFView.frame = newFrame;

[UIView commitAnimations];}

14.定时器创建于取消

创建：playTimer = [NSTimer scheduledTimerWithTimeInterval:1 target:self selector:@selector(countVoiceTime) userInfo:nil repeats:YES];

取消：

[playTimer invalidate];

playTimer = nil;

15. floor(12 \* kScreenRate)这里的floor表示向下取整

floor(x),有时候也写做Floor(x)，其功能是“向下取整”，或者说“向下舍入”，即取不大于x的最大整数

16.xib,或者storyboard中的Label，会自己默认根据文字的多少来自适应高度

两个Label实现，cell的textlabel与detailTextLabel在subTitle和非subTitle样式下，textlabel的位置居中与居上的关系，titlelabel添加上约束与居中显示，detailLabel添加下约束与居中显示，最后把titleLabel与detailLabel两个底部与顶部关联起来，最后设置detailLabel的优先级大于titlelabel----content Hugging Priorty这是两个Label之间的优先级

点击约束编辑Edit，可以设置Priorty对单个对象Label内自身的优先级设置

17. 程序挂起

[[UIApplication sharedApplication]performSelector:@selector(suspend)];

18.MBProgressHUD 可以自定义加载动画

progressHUD.customView = customView

19.跳转到appstore上，评分-------

[[UIApplication sharedApplication] openURL:[NSURL URLWithString: @"https://itunes.apple.com/us/app/hao-wu-zhong-guo/id921449753?mt=8&uo=4"]];

20.查看cocoapods下载进度

cd ~/

cd ~/.cocoapods/

du –sh

更新最新cocosPods

gem install cocoapods –pr

下载库

pod search ‘AFNetworking’

**升级、更新**

sudo gem update --system

通过淘宝的镜像来访问cocoapods

gem sources –-remove <https://rubygems.org/>

gem sources –a [https://ruby.taobao.org/](http://ruby.taobao.org/)

gem sources –l

sudo gem install cocoapods

pod setup

查看版本

pod --version

建立链接

pod install

更新pod库

pod update

搜索相关的库

pod search + 相关搜索索引（如下载相关的：network）

添加库：

在Podfile里面添加信息，然后再pod update更新

pod cache clean –all清除所有的缓存

教程：http://www.tuicool.com/articles/7vUnay

pod init 创建Podfile文件

21. 现在app审核不允许，开发者自己去加入软件更新代码，appStore会为应用自动提醒更新（一般是通过后台来设置）

22. MKNetworkEngine它的hostName是根据什么来的

23. 视频下载的地址跟视频播放的地址有什么不同

24. 视频边放边下载实现：是先把网络资源下载到本地，然后下载一点就播放一点，但是要通过httpServer来启动本地服务

25. 用storyboard千万要注意，被关联的类对象，需要代码获取时，千万不要用init方法，而是用let storyboard = UIStoryboard(name: "Main", bundle: NSBundle.mainBundle())

\_tableVC = storyboard.instantiateViewControllerWithIdentifier(NSString(string: "TableViewControllerId")) as TableViewController获取

26.storyboard界面之间如何传值

通过方法：

* (void)prepareForSegue:(UIStoryboardSegue \*)segue sender:(id)sender

但是要注意：在storyboard点击关联两者之间的线，然后点击从左向右第四个选项设置storyboard Segue的identifer

27.stroyboard中的tableView

在重用cell的时候，直接从缓存池里取出即可，而不需要任何注册或者加载Nib文件的代码

28.同一个控件，貌似不能同时存在代码约束NSLayoutConstraints与xib或者stroyboard约束。有时候也是可以的

在添加约束前，需要先禁止自动约束webView.translatesAutoresizingMaskIntoConstraints = false

在对控件进行代码约束时，参考对象，和父视图，可以不是一个；如：

[self addConstraint:[NSLayoutConstraint constraintWithItem:iconStar attribute:NSLayoutAttributeCenterX relatedBy:NSLayoutRelationEqual toItem:iconImageView attribute:NSLayoutAttributeCenterX multiplier:1 constant:36]];

[self addConstraint:[NSLayoutConstraint constraintWithItem:iconStar attribute:NSLayoutAttributeCenterY relatedBy:NSLayoutRelationEqual toItem:iconImageView attribute:NSLayoutAttributeCenterY multiplier:1 constant:12]];

[self addConstraint:[NSLayoutConstraint constraintWithItem:iconStar attribute:NSLayoutAttributeWidth relatedBy:NSLayoutRelationEqual toItem:nil attribute:0 multiplier:1 constant:25]];

[self addConstraint:[NSLayoutConstraint constraintWithItem:iconStar attribute:NSLayoutAttributeHeight relatedBy:NSLayoutRelationEqual toItem:nil attribute:0 multiplier:1 constant:25]];

29.先调用该方法

- (UIView \*)tableView:(UITableView \*)tableView viewForHeaderInSection:(NSInteger)section

后调用

* (CGFloat)tableView:(UITableView \*)tableView heightForRowAtIndexPath:(NSIndexPath \*)indexPath

30.label

用约束constrainst的时候，不是从Label顶部开始文字的，而是距离顶部一定距离开始抒写文字的，但是当取消掉约束的时候又从顶部开始（注意：是在Label需要显示的文字，完全可以填满Label文字显示的范围）

31. [imageView convertRect:imageView.frame toView:nil];这里toView为nil表示把imageView的frame转换成在整个屏幕的frame

32.在做头视图跟着tableview向上滚动而滚动的时候，如果是作为tableview的headView显示的时候，要把tableView的样式设为group样式，否则，它到达顶部的时候，不会随着tableview的滚动而滚动。（另一种实现方法是单独作为一个cell）

33.tableview的group样式下，要想设置group之间的高度距离，可以受每组的头和脚视图的高度影响的。而且，系统默认也会添加默认的头与脚高度，因此，要想自己设置高度，可以通过下面两个代理方法去设置：

- (CGFloat)tableView:(UITableView \*)tableView heightForHeaderInSection:(NSInteger)section

{

if (section == 0) {

return 200;

}

return 0.00000000000001;

}

- (CGFloat)tableView:(UITableView \*)tableView heightForFooterInSection:(NSInteger)section

{

return 10;

}

34. [CATransaction begin];

[CATransaction setAnimationDuration:0.2];

[CATransaction setAnimationTimingFunction:[CAMediaTimingFunction functionWithName:kCAMediaTimingFunctionEaseIn]];

[CATransaction setCompletionBlock:^{

[self removeFromSuperview];

// [\_scoreView removeFromSuperview];

}];

[self.layer addAnimation:self.lightingAnimation forKey:@"lighting"];

[\_scoreView.layer addAnimation:self.dismissMenuAnimation forKey:@"dismissMenu"];

[CATransaction commit];

35.

UIWindow \*window = [[UIApplication sharedApplication] keyWindow];

[window addSubview:self];

36.

CGRectContainsPoint(\_scoreView.frame, p)判断点p在不在\_scrollView的区域内

37.

UIImageView的drawRect方法系统不调用

因此，我们只能通过UIView去创建调用

38.

enumerateObjectsUsingBlock如何退出循环

在block里面写上\*stop = YES；

39.iOS8新加的类：毛玻璃效果

 UIBlurEffect \*blur = [UIBlurEffect effectWithStyle:UIBlurEffectStyleLight];

（blurView.alpha 的值 来修改模糊度）

    UIVisualEffectView \*effectview = [[UIVisualEffectView alloc] initWithEffect:blur];

    effectview.frame = CGRectMake(0, 0, imageview.size.width/2, 300);

[imageview addSubview:effectview];

40. CAReplicatorLayer要想动画开始是隐藏的（包括其创建的副本）可以通过很短的时间去改变动画视图的opacity:

CABasicAnimation \*opacityAnimation = [CABasicAnimation animationWithKeyPath:@"opacity"];

opacityAnimation.duration = 0.1;

opacityAnimation.toValue = @(1.0);

opacityAnimation.fillMode = kCAFillModeForwards;

41.在绘图结束的时候，必须调用以下的方法一种

CGContextDrawPath

CGContextSetTextDrawingMode

CGContextStrokePath

42.swift----for循环不能带（）括号，且for var index = 0; index < 3; ++index（注意这里的index的++）

43.swift里面的基本类型强制转换---

for i in 0...5

{number = CGFloat(i)}

44.swift字典和数组的初始化都是

chartView.points = [["x":1.5,"y":2],["x":2.2,"y":4.5],["x":3.2,"y":3.4],["x":4.3,"y":3.7]];

45. swift中没有了#Define这种宏定义了，可以用let来声明常量来取代

46. 路径性的动画，可以通过shapeLayer的strokeStart. strokeEnd对其进行动画设置CABasicAnimation(keyPath: "strokeEnd")

另外，折线动画，可以通过一个var path = CGPathCreateMutable()路径然后分别依次设置起始点CGPathMoveToPoint(path, nil, startPoint.x, startPoint.y)

CGPathAddLineToPoint(path, nil, endPoint.x, endPoint.y)

47.终端里取消control+c

48.CoreImage框架、coreplot框架

49.CIDetector进行检测---（可以是人脸与二维码等）

CIFaceFeature---（微笑，闭眼等）

50.swfit里面判断对象存不存在，使用nil或者Optional.None

51.貌似如何通过tableview.tableHeaderView设置头视图的时候，它的高度是tableHeaderView自己决定的，而且一个tableview只有一个，而代理方法获得的，是每一组一个头视图，而且高度和头视图都是通过代理方法来设置的。当同时设置tableHeaderView和执行代理方法设置的时候，会两者并存

52. ReactiveCocoa—为UIKit上控件添加相应事件--http://benbeng.leanote.com/post/ReactiveCocoaTutorial-part1

53.React-native框架、MVVM模式

54．即时通讯----ComChat

55.mac终端清屏ctrl+l-----清楚命令历史history –c

移动mv ------删除rm

获得权限：sudo

删除文件夹：sudo rm –r –f +文件夹名称

open .打开文件夹

56.react-native (第一次要通过brew link node链接到bin目录下,按终端提示一步一步改可读可写权限)---从gitHub 上下载的（如果报错—可以尝试npm install解决）

终端下：npm start-----要在该工程目录下调用

57. - (BOOL)application:(UIApplication \*)application didFinishLaunchingWithOptions:(NSDictionary \*)launchOptions {

static dispatch\_once\_t onceToken;

dispatch\_once(&onceToken, ^{

\_player = [[MPMoviePlayerController alloc]init];

\_playList = [NSMutableArray array];

[self loadArchiver];

//[application setMinimumBackgroundFetchInterval:UIApplicationBackgroundFetchIntervalMinimum];

[self initChannelsData];

//后台播放

AVAudioSession \*session = [AVAudioSession sharedInstance];

[session setCategory:AVAudioSessionCategoryPlayback error:nil];

[session setActive:YES error:nil];

});

return YES;

}

58.GCD—dispatch\_after是异步的---异步线程，如果放在global队列里，则异步---放在main队列里则同步---可用[NSThread current]打印出来查看

GCD中有三种队列类型：

1. The main queue: 与主线程功能相同。实际上，提交至main queue的任务会在主线程中执行。main queue可以调用dispatch\_get\_main\_queue()来获得。因为main queue是与主线程相关的，所以这是一个串行队列。
2. Global queues: 全局队列是并发队列，并由整个进程共享。进程中存在三个全局队列：高、中（默认）、低三个优先级队列。可以调用dispatch\_get\_global\_queue函数传入优先级来访问队列。

用户队列: 用户队列 (GCD并不这样称呼这种队列, 但是没有一个特定的名字来形容这种队列，所以我们称其为用户队列) 是用函数 dispatch\_queue\_create 创建的队列. 这些队列是串行的。正因为如此，它们可以用来完成同步机制, 有点像传统线程中的mutex。

59.

每一个线程都有其对应的RunLoop，但是**默认非主线程的RunLoop是没有运行的，需要为RunLoop添加至少一个事件源**，然后去run它。一般情况下我们是没有必要去启用线程。

**60.枚举类型**

typedef enum {

Sign,

NoSign,

}SignType;

@property (nonatomic, assign)SignType type;默认是0，也就是Sign

61. enumerateObjectsUsingBlock:^(id obj, NSUInteger idx, BOOL **stop){在里面设置**\*stop=YES则循环停止}

62.多个storyboard切换

//1.获取Main.storyboard的第一个控制器

id vc = [[UIStoryboard storyboardWithName:@"Main" bundle:nil]instantiateInitialViewController];//切换到进入箭头所指的根视图控制器

//2.切换window的根控制器

[UIApplication sharedApplication].keyWindow.rootViewController = vc;\

63.block的循环引用解决

\_\_weak typeof (newGame)weakGame = newGame;

\_\_weak typeof (self)selfVC = self;

\_\_weak WCLoginViewController \*weakSelf = self;

\_\_unsafe\_unretained WCLoginViewController \*weakSelf = self;

另一种方式，是在：block函数调用结束以后，置为空：

//回调resultBlock

if(\_resultBlock){

\_resultBlock(XMPPResultTypeLoginSucess);

\_resultBlock = nil;

}

64.想要一个工程创建的文件名都带有前缀的话，点击工程-》最右边show the File inspector -> Project Document->Class Prefix设置即可。

注意：该方法，也可以直接修改storyboard和xib上的名字，不需要再对storyboard和xib上的控件名字关联进行修改了

65.注册与登录的时候，不管你输入的jid存在与否，都会与服务器建立起连接。

67.修改某个类属性的名字->选中改名字->双指按压出菜单->Refactor->rename

68.修改某个类.m里面所有的名字->查找command+f->点击最左边find,菜单选中replace->all->done

69. CAKeyframeAnimation🡪keyTimes,values相对应

70. \_puppleView.transform = CGAffineTransformMakeScale(0.5f \* height, 0.5f \* height);进行形变

71.storyboard视图控制器间跳转

--->首先视图控制器左上角（类名字），连线下一个视图控制器

---->点击连线，设置identifier

---->在需要跳转的事件里，添加代码：

[self performSegueWithIdentifier:@"toEditSegue" sender:nil];

---->这里传得sender可以在下面的方法，直接使用

---->另外还可以在方法- (void)prepareForSegue:(UIStoryboardSegue \*)segue sender:(id)sender {segue.destinationViewController;实现功能}里面去实现，跳转之前的一些操作

---->

----->第二种：笨方法

FeedbackViewController \*feedbackVC = [story instantiateViewControllerWithIdentifier:@"FeedbackViewControllerId"];

[self.navigationController pushViewController:feedbackVC animated:YES];

72.跟视图控制器(或者最开始的)，要想有导航栏，必须通过navigationController->然后再作为其根视图控制器

如果，后面的视图控制器，只需连线，，，就会自带导航栏，但是如果需要添加标题啥的，navigationItem,--返回按扭啥的通过BarButtonItem

72.readonly（总的来说，就是外面不可以通过setter方法去对该属性设值，仅仅只能在该类内部通过赋值的方法来设值）

@property (nonatomic, copy,readonly)NSString \*domain;

---> static NSString \*domain = @"liqiangqiang";

---> - (NSString \*)domain{

return domain;

}

@property (nonatomic, strong, readonly)XMPPvCardTempModule \*vCard;

---> \_vCard = [[XMPPvCardTempModule alloc]initWithvCardStorage:\_vCardStorage];

73.static 与 dynamic cell

与 static cell 对应的是 dynamic cell， 二者的差异是什么？ 使用场景又是怎样的呢 ？  这就是 静态与动态之分。

二者的共同点： 支持多个section， 支持的cell 可以多于一屏，多于一屏时，可上下滑动。

不同点：static tableView 用不着 datasource  和 delegate。  动态tableView 是一定要重用cell的， 对于静态tableView， 似乎用不着 cell 的重用。

这就是说，静态cell的功能完全可以通过动态cell 来实现。当cell 数量超过一屏幕时， 建议还是使用 dynamic tableView 为好。

74.图片转成二进制NSData数据

myCard.photo = UIImageJPEGRepresentation([self.avatarImageView.image], 1.00);

//第二个参数是：比例

75.自定义输出日志：（\_\_VA\_ARGS\_\_与...貌似都是占位符）

//自定义日志

#ifdef DEBUG

//#define WClog(...) NSLog(\_\_VA\_ARGS\_\_)->WClog(@"daadaadaaad");\_\_VA\_ARGS\_\_标识可变参数

#define WCLog(...) NSLog(@"\n%s\n %@ \n\n",\_\_func\_\_,[NSString stringWithFormat:\_\_VA\_ARGS\_\_]);

#else

76.对数据库内容进行监听---数据库查询，如果数据很多会放在子线程查询，移动客户端的数据库里数据不会很多，所以很多数据库的查询是在主线程进行的

NSManagedObjectContext \*rosterContext = [WCXMPPTool sharedWCXMPPTool].rosterStorage.mainThreadManagedObjectContext;

NSFetchRequest \*request = [NSFetchRequest fetchRequestWithEntityName:@"XMPPUserCoreDataStorageObject"];

NSSortDescriptor \*sort = [NSSortDescriptor sortDescriptorWithKey:@"displayName" ascending:YES ];

request.sortDescriptors = @[sort];

\_resultContrl = [[NSFetchedResultsController alloc]initWithFetchRequest:request managedObjectContext:rosterContext sectionNameKeyPath:nil cacheName:nil];

\_resultContrl.delegate = self;

NSError \*err = nil;

[\_resultContrl performFetch:&err]返回值是BOOL值

代理方法

//数据库内容改变

* (void)controllerDidChangeContent:(NSFetchedResultsController \*)controller

77.一般UI的显示都是在主线程进行的操作，否则：界面会延迟刷新或者显示，因此如果当前的操作是在子线程里进行的，则需要回到主线程去刷新UI，

dispatch\_async(dispatch\_get\_main\_queue(), ^{//UI刷新操作}

另外可以通过[NSThread currentThread];确认当前是不是在主线程。

78.串行与并行队列

并行队列：

并行队列是不允许自己创建的，系统中存在三个不同优先级的并行队列。并行队列依旧按照任务添加的顺序启动任务的，但是，后一个任务无需等待前一个任务执行完毕，而是启动第一个任务后，立马启动下一个任务。至于同一时刻允许同时运行多少个任务由系统决定。任务各自运行在并行队列为他们提供的独立线程上，并行队列中同时运行多少个任务，就必须维护多少个线程。

串行队列：dispatch\_get\_global\_queue(DISPATCH\_QUEUE\_PRIORITY\_DEFAULT, 0)

你可以创建任意个数的串行队列，每个队列依次执行添加的任务，一个队列同一时刻只能执行一个任务（串行），但是各个队列之间不影响，可以并发执行。每个队列中的任务运行在一个由各自串行队列维护的对立线程，一个队列中只有一个线程。

主调度队列：

主调度队列中的任务运行在应用程序的主线程上，所以，如果你要修改应用程序的界面，他是唯一的选择dispatch\_async(dispatch\_get\_main\_queue(), ^{

        .....//跟新界面的操作

    });

79.enum、typedef enum

enum AlertTableSections(枚举名称){};

typedef enum {} AlertTableSections(枚举名称);

typedef NS\_ENUM(NSInteger, Test1(枚举名称)) {}；

80. [iOS中定时器NSTimer的使用](http://www.cnblogs.com/pengyingh/articles/2350843.html)

1、初始化

+ (NSTimer \*)timerWithTimeInterval:(NSTimeInterval)ti target:(id)aTarget selector:(SEL)aSelector userInfo:(id)userInfo repeats:(BOOL)yesOrNo;

+ (NSTimer \*)scheduledTimerWithTimeInterval:(NSTimeInterval)ti target:(id)aTarget selector:(SEL)aSelector userInfo:(id)userInfo repeats:(BOOL)yesOrNo;

注：不用scheduled方式初始化的，需要手动addTimer:forMode: 将timer添加到一个runloop中。

而scheduled的初始化方法将以默认mode直接添加到当前的runloop中.

注：

NSTimer可以精确到50-100毫秒

NSTimer不是绝对准确的，而且中间耗时或阻塞错过下一个点，那么下一个点就pass过去了

/延时函数：

[NSThread sleepForTimeInterval:5.0]； //暂停5s.

//Timer的使用：

NSTimer \*connectionTimer; //timer对象

//实例化timer

self.connectionTimer=[NSTimer scheduledTimerWithTimeInterval:1.5 target:self selector:@selector(timerFired:) userInfo:nil repeats:NO];

[[NSRunLoop currentRunLoop]addTimer:self.connectionTimer forMode:NSDefaultRunLoopMode];

//用timer作为延时的一种方法

do{

[[NSRunLoopcurrentRunLoop]runUntilDate:[NSDatedateWithTimeIntervalSinceNow:1.0]];

}while(!done);

//timer调用函数

-(void)timerFired:(NSTimer \*)timer{

done =YES;

}

90.CADisplayLink

CADisplayLink是一个能让我们以和屏幕刷新率同步的频率将特定的内容画到屏幕上的定时器类。CADisplayLink以特定模式注册到runloop后，每当屏幕显示内容刷新结束的时候，runloop就会向CADisplayLink指定的target发送一次指定的selector消息， CADisplayLink类对应的selector就会被调用一次。

NSTimer以指定的模式注册到runloop后，每当设定的周期时间到达后，runloop会向指定的target发送一次指定的selector消息。

iOS设备的屏幕刷新频率(FPS)是60Hz，因此CADisplayLink的selector默认调用周期是每秒60次，这个周期可以通过frameInterval属性设置，CADisplayLink的selector每秒调用次数=60/frameInterval。比如当frameInterval设为2，每秒调用就变成30次。因此，CADisplayLink周期的设置方式略显不便。

NSTimer的selector调用周期可以在初始化时直接设定，相对就灵活的多。

91.UILocalNotification(本地通知)-----通知是独立于app程序以外的，不会因为程序的关闭而关闭，因此，需要注意，添加通知之前先取消掉之前添加的通知

注册有：

alertAction

alertBody

applicationIconBadgeNumber

alertLaunchImage

fireDate

repeatInterval

userInfo（通过通知打开应用时，传过去的信息）

--------》如果通知打开应用之前程序仍是启动的时候，会调用方法：

-(void)application:(UIApplication)application didReceiveLocalNotification:(UILocationNotification \*)notification

---------》如果通知方式打开之前程序未启动，则通过方法

didFinishLaunchingWithOptions:(NSDictionary \*)launchOptions

该launchOptions字典里存着的有通知的一些信息，包括userInfo设置的用户信息

scheduleLocalNotification添加通知（UIApplication对象调用）

cancelAllLocalNotification取消所有通知

92.远程推送《APNs服务器》

所有的苹果设备，在联网的状态下，都会与苹果的服务器建立长连接。

保持长连接的作用：--------》

时间校准

系统升级

查找我的iPhone

长连接的好处：-----------》

数据传输速度快

数据保持最新状态

远程推送原理：

项目第一次运行的时候：

UUID+AppID(项目的Bundle identifier)—貌似是申请的证书实现

1:app客户端<-----------------------------------🡪苹果服务器APNs

服务器返回deviceToken

deviceToken

2.app客户端---------------------- ---》公司的服务器（存储deviceTokern）

发送消息时

body:”吃饭没”

1：app客户端----------------》公司服务器

deviceToken+body:”吃饭没”

2：公司服务器--------------------------》APNs

DeviceToken+body

3：APNs----------------------------🡪苹果手机

AppID(项目的Bundle identifier)

3：苹果手机解析deviceToken------------------------------🡪相应地app

提送必备条件：

1. 告诉苹果：哪个应用可以做推送
2. 告诉苹果：在哪台电脑上调试推送服务
3. 告诉苹果：在哪台设备上调试推送服务

真机调试：(一个开发者账号只能配置两个开发者证书，当证书配置时appID为\*，表示所有的bundleId-app都可以使用)

真机调试的主要步骤

1. 登录开发者主页
2. 生成cer证书：cer是一个跟电脑相关的证书文件，让电脑具备真机调试的功能
3. 添加AppID:调试哪些app?
4. 注册真机设备：哪台设备需要做真机调试
5. 生成MobileProvision文件：结合2.3.4生成一个手机规定文件
6. 导入cer(需要安装在电脑上—双击（mac的key钥匙串上面可以查看）)、MobileProvision（双击安装到xcode上再通过内部生成ipa文件安装到手机上）文件

最终会得到2个文件

Cer文件：让电脑具备真机调试的功能

MobileProvision文件：哪台设备、哪些app、哪台电脑需要做真机调试？

远程推送：

最终需要三个文件：

两个cer（开发与生产）

一个MobileProvision

　UUID是指在一台机器上生成的数字，它保证对在同一时空中的所有机器都是唯一的。通常平台会提供生成的API。

92.图片拉伸问题，仅仅只拉伸中间部分，而不拉伸两端！(http://blog.csdn.net/q199109106q/article/details/8615661)

iOS5以前------

* (UIImage \*)stretchableImageWithLeftCapWidth:(NSInteger)leftCapWidth topCapHeight:(NSInteger)topCapHeight;

iOS5----------

UIEdgeInsets insets = UIEdgeInsetsMake(top, left, bottom, right);

// 伸缩后重新赋值

image = [image resizableImageWithCapInsets:insets];

iOS6----------

 [image resizableImageWithCapInsets:insets resizingMode:UIImageResizingModeStretch];

93.swift与OC枚举enum类型关系

OC:

typedef enum {

Sign,

NoSign,

}SignType;

swift:

let type:SignType!=self.calendarView.type

if(type.value==NoSign.value)

{

self.calendarView.type=Sign

}else

{

self.calendarView.type=NoSign

}

94.获取项目里某个文件的路径

NSString \*plistPath = [[NSBundle mainBundle] pathForResource:@"fillCheckRule" ofType:@"plist"];

95.drawRect方法里面，最后绘制图片

CGContextDrawPath(context, kCGPathFillStroke);

CGContextFillPath(Context)

96.block函数

函数的作为参数：

-(void)paySalaryForMoney:(void(^)(int salary))amount；

调用：

[self paySalaryForMoney:^(int salary) {

}]

返回值是对象的block函数

- (Person \*(^)(NSString \*))study

{

return ^(NSString \*name){

NSLog(@"study----%@", name);

return self;

};

}

函数作为变量：

\_\_block **int** local = 500;

**void**(^block)(**void**) = ^(**void**)

{

    local++;

    NSLog(@"local:%d", local);

};

block函数的妙用：

调用部分

int main(int argc, const char \* argv[])

{

@autoreleasepool {

Person \*p = [[Person alloc] init];

p.study(@"xx宝典").run().study(@"xx剪发");

}

return 0;

}

实现部分

.h

@interface Person : NSObject

- (Person \*(^)(NSString \*name))study;

- (Person \*(^)())run;

@end

.m

- (Person \*(^)(NSString \*))study

{

return ^(NSString \*name){

return self;

};

}

- (Person \*(^)())run

{

return ^{

return self;

};

}

97.把任意对象存进数据库（通过把对象归档成二进制的方式存入，在取出解档）

- (void)setup

{

// 初始化

NSString \*path = @"/Users/lizhongqiang/Desktop/shops.data";

self.db = [FMDatabase databaseWithPath:path];

[self.db open];

// 2.创表

[self.db executeUpdate:@"CREATE TABLE IF NOT EXISTS t\_shop (id integer PRIMARY KEY, shop blob NOT NULL);"];//shop blob NOT NULL表示存进的二进制shop不能为空

}

- (void)readShops

{

FMResultSet \*set = [self.db executeQuery:@"SELECT \* FROM t\_shop LIMIT 0,10;"];//LIMIT 0,10表示一条限制语句：取id从0开始到后面的总共10条数据

while (set.next) {

NSData \*data = [set objectForColumnName:@"shop"];

HMShop \*shop = [NSKeyedUnarchiver unarchiveObjectWithData:data];

NSLog(@"%@", shop);

}

}

- (void)addShops

{

for (int i = 0; i<100; i++) {

HMShop \*shop = [[HMShop alloc] init];

shop.name = [NSString stringWithFormat:@"商品--%d", i+100];

shop.price = arc4random() % 10000;

NSData \*data = [NSKeyedArchiver archivedDataWithRootObject:shop];

[self.db executeUpdateWithFormat:@"INSERT INTO t\_shop(shop) VALUES (%@);", data];

}

}

98.extern与const关键字：

.h文件----声明

/\*

extern可置于变量或者函数前，以表示变量或者函数的定义在别的文件中，提示编译器遇到此变量或函数时，在其它模块中寻找其定义

\*/

extern NSString \* const HWRedirectURI;

/\*

从NSString \*到NSString \*const的区别:(const也可以从字面上理解，常数，常量与变量之间的区别)

NSString \*值的指针可以改变指向

NSString \*const的就是无论内容，指向都不能改了

\*/

.m文件-----定义

NSString \* const HWRedirectURI = @"http://www.baidu.com";

99.错误信息提示:

\_\_LINE\_\_代码行号，表示错误发生在第几行

\_\_FILE\_\_被处理成字符串，表示错误发生的路径(NSString \*file = [NSString stringWithUTF8String:\_\_FILE\_\_];)

\_\_FUNCTION\_\_表示方法

#include  < stdio.h >

  int  main( int  ,  char \*\* )

 {

printf( " This fake error is in %s on line %d\n " , \_\_FILE\_\_, \_\_LINE\_\_);

return   0 ;

 }

输出结果：

This fake error is in c:\temp\test.cpp on line 5

100.给结构体取别名

typedef struct objc\_ivar \*Ivar;

struct objc\_ivar {

char \*ivar\_name

char \*ivar\_type

int ivar\_offset

#ifdef \_\_LP64\_\_

int space

#endif

}

101

Objective-C运行时定义了几种重要的类型。

class：定义Objective-C类

Ivar：定义对象的实例变量，包括类型和名字。

Protocol：定义正式协议。

objc\_property\_t：定义属性。叫这个名字可能是为了防止和Objective-C 1.0中的用户类型冲突，那时候还没有属性。

Method：定义对象方法或类方法。这个类型提供了方法的名字（就是\*\*选择器\*\*）、参数数量和类型，以及返回值（这些信息合起来称为方法的\*\*签名\*\*），还有一个指向代码的函数指针（也就是方法的\*\*实现\*\*）。

SEL：定义选择器。选择器是方法名的唯一标识符。

IMP：定义方法实现。这只是一个指向某个函数的指针，该函数接受一个对象、一个选择器和一个可变长参数列表（varargs），返回一个对象

102:# 52c332的RGB颜色

deformationBtn.contentColor = [self getColor:@"52c332"];

- (UIColor \*)getColor:(NSString \*)hexColor

{

unsigned int red,green,blue;

NSRange range;

range.length = 2;

range.location = 0;

[[NSScanner scannerWithString:[hexColor substringWithRange:range]] scanHexInt:&red];

range.location = 2;

[[NSScanner scannerWithString:[hexColor substringWithRange:range]] scanHexInt:&green];

range.location = 4;

[[NSScanner scannerWithString:[hexColor substringWithRange:range]] scanHexInt:&blue];

return [UIColor colorWithRed:(float)(red/255.0f) green:(float)(green / 255.0f) blue:(float)(blue / 255.0f) alpha:1.0f];

}

103.评分

NSString \*appid= @”725296055”;

NSString \*str=[NSString stringWithFormat:@”items-apps://itunes.apple.com/cn/app/id%@?mt=8”,appid];

[[UIApplication sharedApplication] openURL:url];

104.cell里面获取系统内部子组件

NSLog(@"+++++++%@++++++",NSStringFromCGRect(self.textLabel.frame));经测试，打印出来的是**{{0, 0}, {0, 0}}，因此推测，cell内部的子组件位置是通过约束实现的**

**105.添加约束报错解决**

UITableView坑

带有UITableView的界面如果到遇到以下警告

Warning once only: Detected a case where constraints ambiguously suggest a height of zero for a tableview cell's content view. We're considering the collapse unintentional and using standard height instead.

添加以下代码可解决

self.tableView.rowHeight = 44.0f;

、

106.Quartz2D

语句最后的渲染到View上的语句有：

CGContextStrokePath(<#CGContextRef c#>)

CGContextDrawPath(<#CGContextRef c#>, <#CGPathDrawingMode mode#>)

………目测共同点都是最后加Path路径

107：self

在类方法中使用，表示的是当前类

在对象方法中使用，表示的是当前对象

对象方法----可以通过：

[类名 alloc]创建对象---去调用

108：CALayer-------需要调用drawInContext方法，必须调用[\_backgroundLayer setNeedsDisplay];方法

使用部分：

\_backgroundLayer = [BackgroundLayer layer];

\_backgroundLayer.frame = CGRectMake(0, 0, CGRectGetWidth([UIScreen mainScreen].bounds), CGRectGetHeight(self.frame));

[\_backgroundLayer setNeedsDisplay];

\_backgroundLayer.backgroundColor = [UIColor clearColor].CGColor;

[self.contentView.layer addSublayer:\_backgroundLayer];

声明部分：

- (void)drawInContext:(CGContextRef)ctx

{

CGContextMoveToPoint(ctx, 15, 0);

if(\_isFirst)

{

CGContextMoveToPoint(ctx, 0, 0);

}

CGContextSetStrokeColorWithColor(ctx, TITLE\_COLOR\_99.CGColor);//

CGContextSetLineWidth(ctx, 0.45);

CGContextAddLineToPoint(ctx, CGRectGetWidth(self.frame), 0);

if(\_isLast)

{

CGContextMoveToPoint(ctx, 0, CGRectGetHeight(self.frame));

CGContextAddLineToPoint(ctx, CGRectGetWidth(self.frame), CGRectGetHeight(self.frame));

}

CGContextStrokePath(ctx);

}

109:字符串操作：

NSScanner、正则表达式

110.

类别和类扩展的区别。

答：category和extensions的不同在于 后者可以添加属性。另外后者添加的方法是必须要实现的。

extensions可以认为是一个私有的Category。

111.CAShapeLayer与CALayer的区别

普通CALayer在被初始化时是需要给一个frame值的,这个frame值一般都与给定view的bounds值一致,它本身是有形状的,而且是矩形.

CAShapeLayer在初始化时也需要给一个frame值,但是,它本身没有形状,它的形状来源于你给定的一个path,然后它去取CGPath值,它与CALayer有着很大的区别

CAShapeLayer有着几点很重要:

1. 它依附于一个给定的path,必须给与path,而且,即使path不完整也会自动首尾相接

2. strokeStart以及strokeEnd代表着在这个path中所占用的百分比

3. CAShapeLayer动画仅仅限于沿着边缘的动画效果,它实现不了填充效果

需要注意的是：

CAShapeLayer不需要设置bounds与frame，否则CAShapelayer就会有自己的范围了，可以通过CAShapeLayer的backgroundColor进行测试。

112. CFTimeInterval与NSTimeInterval

没有区别，除了名字不同。

在NS和CF之间存在很多相通和类似

114.

CTFramesetter是CTFrame的创建工厂，NSAttributedString需要通过CTFrame绘制到界面，得到CTFramesetter后，创建path(绘制路径),然后得到CTFrame，最后直接以CTFrame为参数使用CTFrameDraw绘制或者从CTFrame中获取CTLine进行微调后使用CTLineDraw进行绘制。。

CTFramesetterRef framesetter = CTFramesetterCreateWithAttributedString((CFAttributedStringRef)attriString);  //注意这里的attriString创建，需要通过[attriString addAttribute:(**NSString** \*)kCTForegroundColorAttributeName  value:(**id**)[UIColor redColor].CGColor

      range:NSMakeRange(0, 4)];方式设值

CGMutablePathRef path = CGPathCreateMutable();

CGPathAddRect(path, **NULL**, rect);

CTFrameRef frame = CTFramesetterCreateFrame(framesetter, CFRangeMake(0, 0), path, **NULL**);

CFRelease(path);

CFRelease(framesetter);

一个CTFrame是由一行一行的CLine组成，每个CTLine又会包含若干个CTRun(既字形绘制的最小单元)，通过相应的方法可以获取到不同位置的CTRun和CTLine，以实现对不同位置touch事件的响应。

115.共用体和结构体的区别

当一个[共用体](http://baike.baidu.com/view/1698544.htm)被声明时, [编译程序](http://baike.baidu.com/view/454895.htm)自动地产生一个变量, 其长度为联合中最大的变量长度的整数倍

一. 共用体和[结构体](http://baike.baidu.com/view/204974.htm)都是由多个不同的数据类型成员组成, 但在任何同一时刻, 共用体只存放了一个被选中的成员, 而结构体的所有成员都存在。

二. 对于共用体的不同成员赋值, 将会对其它成员重写, 原来成员的值就不存在了, 而对于结构体的不同成员赋值是互不影响的。

联合体中包含结构体：

union \_GLKVector3

{

struct { float x, y, z; };

struct { float r, g, b; };

struct { float s, t, p; };

float v[3];

};

typedef union \_GLKVector3 GLKVector3;

116.NSTimeInterval以秒为单位。

117.是View的类方法—---该方法返回coreAnimation使用该视图的时候，返回CALayer子类

每一个UIView实例都有一个相关联的被Cocoa Touch按需自动创建的Core Animation层。Cocoa Touch会调用“+layerClass”方法来确定要创建什么类型的层。

+ (Class)layerClass

{

return [CAEAGLLayer class];

}

118.Cocoa Touch会自动调用下面的方法，这是反归档先前归档入一个文件的对象的过程的一部分。

- (id)initWithCoder:(NSCoder\*)coder

119.任何在接收到视图重新调整大小的消息时，Cocoa Touch都会调用下面的layout Subviews方法。

120.offsetof宏

在stddef.h头文件中

该宏用于求结构体中一个成员在该结构体中的偏移量。

在msdn上，该宏被写作：

size\_t offsetoff(structName, memberName);

第一个参数：结构体的名字

第二个参数：结构体成员的名字。

该宏返回结构体structName中成员memberName的偏移量。偏移量是size\_t类型的。

121.取绝对值：abs fabs fabsf用法

int abs(int i); //处理int类型的取绝对值

double fabs(double i); // 处理double类型的取绝对值

float fabsf(float i); // 处理float类型的取绝对值

122.

cosf求弧度值的余弦值。

extern float cosf(float);

extern double cos(double);

extern long double cosl(long double);

extern float sinf(float);

extern double sin(double);

extern long double sinl(long double);

123.iOS8开始实现毛玻璃效果的方法：

//UIVisualEffect该对象是为了初始化UIVisualEffectView对象，设置毛玻璃效果样式

UIVisualEffect \*effect = [UIBlurEffect effectWithStyle:UIBlurEffectStyleDark];

//--UIVisualEffectView->实现模糊效果

UIVisualEffectView \*effectView = [[UIVisualEffectView alloc]initWithEffect:effect];

effectView.frame = self.collectionView.bounds;

effectView.alpha = 1.0;

124.判断两个矩形是否有交叉：返回值是一个bool值

CG\_EXTERN bool CGRectIntersectsRect(CGRect rect1, CGRect rect2)

125. CATransform3D矩阵

CATransform3D t = CATransform3DIdentity;

t.m34 = -1.0/500;//透视效果，要操作的这个对象要有旋转的角度，且必须要放在旋转之前，否则没有效果。正直/负值都有意义

//负值表示旋转缩小

//正值表示旋转放大

/\*

m34矩阵值得作用：

m34负责z轴方向的translation（移动），m34= -1/D, 默认值是0，也就是说D无穷大，这意味layer in projection plane（投射面）和layer in world coordinate重合了。

上面最重要的是m34这个属性，CATransform3DRotate获取的旋转如果之前联合的transform不支持透视，那在x、y轴上做旋转是只有frame放大缩小的变化，我们需要的是在旋转的时候要使得离视角近的地方放大，离视角远的地方缩小，就是所谓的视差来形成3D的效果。

M34=-1/D-🡪其中D表示光源距离投影面的距离

\*/

CGFloat radianFraction = fraction \* self.maxAngle;

t = CATransform3DRotate(t, radianFraction, 0, 0, 1);

126.anchorPoint与center

center是确定视图的anchorPoint的位置的，默认anchorPoint是视图的中心点

视图可以通过设置anchorPoint的位置来设置翻转的中心点

anchorpoint时图片动画旋转等效果时的支撑点

另外需要注意的是：layer.anchorpoint需要设置在frame的前面

position、anchorPoint、origin三者之间分关系

frame.origin.x = position.x - anchorPoint.x \* bounds.size.width/2.0;

frame.origin.y=position.y-anchorPoint.x\*bounds.size.height/2.0;

127. animation.repeatCount = INFINITY;//无限大

128. 注意下面设置的菜单栏，要想实现显示，必须实现item的方法

UIMenuItem \*menuItem = [[UIMenuItem alloc]initWithTitle:@"分享到新浪微博" action:@selector(changeColor:)];

UIMenuController \*menu = [UIMenuController sharedMenuController];

[menu setMenuVisible:YES animated:YES];

[menu setMenuItems:@[menuItem]];

- (void)changeColor:(UIMenuItem \*)item

{}

129.类扩展（一般情况下只能扩展方法，而不能是属性）

因此我们可以通过关联去实现：

OBJC\_EXPORT void objc\_setAssociatedObject(id object, const void \*key, id value, objc\_AssociationPolicy policy)

第一个参数是：关联的对象

第二个参数是：关联的key

第三个参数使：被关联的属性值（如果该参数为nil,则表示断开关联）

第四个参数是：属性的策略（类似retain、copy、strong、assign等）

OBJC\_EXPORT id objc\_getAssociatedObject(id object, const void \*key)获取关联属性

关联的key:

1.声明 static char kAssociatedObjectKey; ，使用 &kAssociatedObjectKey 作为 key 值;

2.声明 static void \*kAssociatedObjectKey = &kAssociatedObjectKey; ，使用 kAssociatedObjectKey 作为 key 值；

3.用 selector ，使用 getter 方法的名称作为 key 值。因为SEL生成的时候就是一个唯一的常量

\_cmd在Objective-C的方法中表示当前方法的selector,正如同self表示当前方法调用的对象实例。

130.屏幕截图

iOS7开始：

- (UIView \*)snapshotViewAfterScreenUpdates:(BOOL)afterUpdates NS\_AVAILABLE\_IOS(7\_0);

- (UIView \*)resizableSnapshotViewFromRect:(CGRect)rect afterScreenUpdates:(BOOL)afterUpdates withCapInsets:(UIEdgeInsets)capInsets NS\_AVAILABLE\_IOS(7\_0);

可以快速截取视图的图像，注意返回值是一个视图view

+(UIImage \*)cutFromView:(UIView \*)view{

//开启图形上下文

UIGraphicsBeginImageContextWithOptions(view.frame.size, NO, 0.0f);

//获取上下文

CGContextRef context = UIGraphicsGetCurrentContext();

//在新建的图形上下文中渲染view的layer

[view.layer renderInContext:context];

[[UIColor clearColor] setFill];

//获取图片

UIImage \*image = UIGraphicsGetImageFromCurrentImageContext();

//关闭图形上下文

UIGraphicsEndImageContext();

return image;

}

131.图片切成两半的方法

- (NSArray\*)generateImagesFromImage:(UIImage\*)image;

{

NSMutableArray \*images = [NSMutableArray array];

for (int i=0; i<2; i++) {

CGSize size = CGSizeMake(image.size.width, image.size.height/2);

CGFloat yPoint = (i==0) ? 0 : -size.height;

// draw half of the image in a new image

UIGraphicsBeginImageContextWithOptions(size, NO, [UIScreen mainScreen].scale);

[image drawAtPoint:CGPointMake(0, yPoint)];

UIImage \*image = UIGraphicsGetImageFromCurrentImageContext();

UIGraphicsEndImageContext();

// save image

[images addObject:image];

}

return images;

}

133.

#pragma mark --因为我加了这一层view在tableview的上面，所以要把触摸时间给tableview

- (UIView \*)hitTest:(CGPoint)point withEvent:(UIEvent \*)event {

if (![self pointInside:point withEvent:event]) {

return nil;

}

else {

for (UIView \*view in self.superview.subviews) {

if ([view isKindOfClass:[UITableView class]]) {

return view;

}

}

}

return nil;

}

134.

//stringByPaddingToLength:withString:startingAtIndex:创建一个新字符串,长度为length,旧字符串从索引字符开始拷贝到旧字符串后面

        NSString \* str14 = [str13 stringByPaddingToLength:20 withString:str13 startingAtIndex:1];

135.调试技巧

watchpoint set v string\_weak\_设置观察点，观察string\_weak\_变量的值得变化

136.关键字：\_cmd与宏\_\_func\_\_的区别：

\_\_func\_\_宏确实是该方法的名称

而

\_cmd关键字则是方法实现对应的@selector（）

在Objective-C中调用一个方法，其实是向一个对象发送消息，查找消息的唯一依据是selector的名字。利用Objective-C的动态特性，可以实现在运行时偷换selector对应的方法实现，达到给方法挂钩的目的。

每个类都有一个方法列表，存放着selector的名字和方法实现的映射关系。IMP有点类似函数指针，指向具体的Method实现。

137

Objective-C运行时定义了几种重要的类型。

Class：定义Objective-C类

Ivar：定义对象的实例变量，包括类型和名字。

Protocol：定义正式协议。

objc\_property\_t：定义属性。叫这个名字可能是为了防止和Objective-C 1.0中的用户类型冲突，那时候还没有属性。

Method：定义对象方法或类方法。这个类型提供了方法的名字（就是\*\*选择器\*\*）、参数数量和类型，以及返回值（这些信息合起来称为方法的\*\*签名\*\*），还有一个指向代码的函数指针（也就是方法的\*\*实现\*\*）。

SEL：定义选择器。选择器是方法名的唯一标识符。

IMP：定义方法实现。这只是一个指向某个函数的指针，该函数接受一个对象、一个选择器和一个可变长参数列表（varargs），返回一个对象

(其中SEL与IMP是对应的)

138.

ios 启动程序时隐藏状态栏，启动后显示状态栏

1、在info.plist里面  Status bar is initially hidden 设置为 YES

---该操作仅仅只是加载app的时候，是否隐藏状态栏

View controller-based status bar appearance 设置为NO---（这个是设置View controller与status bar的优先级设置，例如设置为YES时View controller对status bar的设置优先级高于application的设置，为NO则以application的设置为准，view controller的prefersStatusBarHidden方法无效，是根本不会被调用的）

2、在appDelagate里面 设置

iOS9之前

[application setStatusBarHidden:NO withAnimation:UIStatusBarAnimationFade]; 即可。

iOS9

[UIApplication sharedApplication].statusBarHidden = NO;

137.

//判断是否是启动程序,

[[NSUserDefaults standardUserDefaults] setObject:@"1" forKey:@"launchApp"];

[[NSUserDefaults standardUserDefaults] synchronize];这是同步保存数据，以防后面的某些操作影响数据保存

139.cell代理方法返回的高度，cell会重新布局所有子视图，此时注意：

例如：cell画底线，底线的frame可以写在cell的layoutSubviews方法里

也可以通过约束来动态的改变底线相对于cell高度的位置不变

140.

swift里面

var与let

需要注意的是：真正的var是指数据初始化以后，该数据的指向发生改变，转而指向另一个数据的时候，需要设置为var,反之设置成let

注意：如果一个数据设置成var型，而初始化之后，该数据并没有被改变，则会提示警告

注意：如果一个对象是let型，并不意味着该数据完全的不能够被修改，其内部的属性依然可以被修改的，内部的属性能否被修改根据其属性自身的var或者let设置有关

141.swift里面，如果一个数据一致往里面存，而不往外取，就会有警告

142.swift里面的结构体

struct locationStruct {

var log:String?

var lat:String?

init(loca:CLLocationCoordinate2D){

log = "\(loca.longitude)"

lat = "\(loca.latitude)"

}

}

143.swift使用函数类型

在swift中，使用函数类型就像使用其他类型一样。例如，你可以定义一个类型为函数的常量或变量，并将函数赋值给它

var mathFunction: (Int,Int)->Int = addTwoInts

func addTwoInts(a:Int,b:Int)->Int{

return a+b

}//这个函数的类型是：（Int,Int）->Int,可以读作“这个函数类型，它有两个Int型的参数，并返回一个Int型的值”

144. UIPickerViews

默认是有frame的，其默认值为：{0，0，320，216}

因此，设置frame的时候需要注意这一点；

另外，需要注意的是，最好一开始的时候，就设置pickView的farme

否则，pickview的分割线，不会顶到frame的两端

155.

-(void)willMoveToSuperview:(UIView \*)newSuperview//通知接收者他的父视图将会改变到指定的父视图（也有可能是nil）,这一句，是在添加父视图，或者修改父视图（superview addSubView:的时候调用的）

156.

MVC是系统框架

MVP（Model-View-Presenter）是用在某个特定页面上的，也就是说MVP的灵活性要远远大于MVC，实现起来也极为复杂

MVVM（Model-View-ViewModel）

三者依次进化而成MVC->MVP->MVVM

157.

获取项目中文件的路径

[NSBundle mainBundle]pathForResource:ofType:;

158.

获取当前时间

[NSTimer timer];

CACurrentMediaTime();

159.

void \*info

---void \*叫做无确切类型指针，这个指针指向一块内存，却没有告诉程序该用何种方式来解释这片内存，所以这种类型的指针不能直接进行取内容的操作，必须先转成别的类型的指针才能把内容解释出来

如：

RLSampleViewController \*obj = (\_\_bridge RLSampleViewController \*)info;

160.UIImage缓存取舍

imgaeNamed初始化，默认加载图片成功后会缓存中缓存图片

imageWithContentsOfFile初始化，仅只加载图片，不缓存。

当大量使用imageNamed方式会在不需要缓存的地方额外增加开销CPU的时间来做着一件事。当应用程序需要加载一张比较大的退片并且一次性使用，那么其实没有必要去缓存这一张图片，用imageWithContentsOfFile是最为经济的方式，这样不会因为UIImage元素较多情况下，CPU会被逐个分散在不必要缓存上浪费过多时间

161.

CALayer不是UIResponder的子类，这说明CALayer无法响应事件，这也是UIView和CALayer的重要区别之一

171.

\_certificationBtn.contentHorizontalAlignment = UIControlContentHorizontalAlignmentLeft;

UIButton的titleLabel文字左对齐

178.

//CGContextStrokePath(ctx);用于画线的线条绘制//CGContextFillPath不能够用于画线的线条绘制，而是仅仅是针对于一定的线条围城的区域

画线不清晰的话，需要设置反锯齿

CGContextSetAllowsAntialiasing(ctx, true)

179.

NSObject

里面有一个方法mutableCopy

可以把NSString，NSArray等转换成相信的mutableString、mutableArray等可变的对象

180.

NSAssert()函数和NSLog()函数一样简单易用

例如：

NSAssert(x!=0,@”x must not be zero”);

表示：

在表达式“x!=0”不成立时，程序就会抛出异常，并显示自定义的消息“x must not be zero”,并同时显示出错的文件、代码和调用函数等信息，是一个程序追踪的很好手段。

假设x!=0不成立，就提示reason-x must not be zero

190.

问题：

需要根据图片，来自适应cell高度，(autoLayout—不需要把约束设置完全，而留一些>=的约束)

本地图片是可以的。

但是网络图片，貌似难以实现——貌似现在想到的好办法，是先获取当前界面的所有cell,然后判断是否所有界面显示的cell都已经加载完图片了，然后再统一刷新tableView(或者——直接后台直接在获取tableview列表的时候，就把每个图片的长宽返回给我们了)

UITableViewCell的prepareForReuse方法

当前已经被分配的cell如果被重用了(通常是滚动出屏幕外了),会调用cell的prepareForReuse通知cell，

注意：

当重写prepareForReuse方法的时候，一定要在方法内部调用父类方法[super prepareForReuse].

* (void)prepareForReuse{

[super prepareForReuse];

}

这个在使用cell作为网络访问的代理容器时尤其要注意,需要在这里通知取消掉前一次网络请求，不要再给这个cell发送数据了.

自定义UITableViewCell的方法有很多，发现一些人都会遇到自定义的cell里面图片错落的问题，这个问题解决方法：就是在UITableViewCell的子类里实现prepareForReuse方法，把内容清空掉。

191.

SEL类型：

SEL类型的变量，可以通过@selector(方法名)来取得，

可以把它理解为：“方法名的某种映射结果”

IMP类型：

这个才是函数指针，IMP可以从对象&SEL的方法得到：

IMP imp = [self methodForSelector:selector];

这是IMP的定义：

typedef id (\*IMP)(id,SEL,…);

另外注意NSObject里面的这两个方法：

* (IMP)methodForSelector: (SEL)aSelector;

+ (IMP)instanceMethodForSelector: (SEL)aSelector;

instanceMethodForSelector是NSObject的一个类方法，会得到一个指向这个实例方法的一个函数指针。

例子：

* BOOL overrided = [cell.class instanceMethodForSelector:selector] != [UITableViewCell instanceMethodForSelector:selector];

这个方法是判断是否是继承父类的方法

192.

titleLabel.sizeToFit()

该方法是使该控件的frame.size与内容相匹配，根据内容的增大，而增大

193.银行卡号输入，并且做字数限制

- (BOOL)textField:(UITextField \*)textField shouldChangeCharactersInRange:(NSRange)range replacementString:(NSString \*)string{

int count = 0;

for (int i = 0; i < textField.text.length; i++) {

NSString \*s = [textField.text substringWithRange:NSMakeRange(i, 1)];

if ([s isEqualToString:@" "]) {

count ++;

}

}

// count = count/2;

int a = count/2 + 1;

if (range.location != 0 &&

range.location % (4 \* a + count) == 0 &&

textField.text.length <= range.location &&

textField.text.length <= 27)

{

textField.text = [textField.text stringByAppendingString:@" "];

}

if ([string isEqualToString:@""])

{

return YES;

}

if (textField.text.length > 26)

{

return NO;

}

return YES;

}

193.技巧：在开发中，可以适当的使用C语言函数

194. CGContextFillPath，CGContextClip 使用的是非零绕组规则

CGContextEOClip 使用的是奇偶规则

195.技巧：

- (void)layoutSubviews

{

[super layoutSubviews];

// 设置y值(当自己的高度发生改变了，肯定要重新调整Y值，所以放到placeSubviews方法中设置y值)

self.mj\_y = - self.mj\_h - self.ignoredScrollViewContentInsetTop;

}

196.超出UIButton的范围，添加子视图，貌似，不能响应其事件

所以一般，弹窗类的是添加在[UIApplication sharedApplication].keyWindow上

这是因为：

注意当子视图超过父视图的frame之外时，是无法响应事件的

197. Block的循环引用问题，由于我们很多行为会导致Block的copy，而当Block被copy时，会对block中用到的对象产生强引用(ARC下)或者引用计数加一(non-ARC下)。----这个copy需要参考mutablcopy与copy之间的区别

----这个理解：block会对block里面的内容进行copy一次

198.

设计模式中的 “ 单一功能原则 ”(Single responsibility principle)，我们应该把功能拆分，把不同的功能都放到各自不同的类里面。