-----------无法归类-------------------------

NSTimer轮询运行指定方法

该指针声明需要放在头文件中，以便其他方法（被轮方法）暂停该定时器

myTimer = [NSTimer scheduledTimerWithTimeInterval:0.1 //设定间隔时间

target:self //指定调用方法的消息发给谁

selector:@selector(downloadPro) //指定要调用的方法名

userInfo:nil //userInfo可以给消息发参数

repeats:YES];

停止定时器：[myTimer invalidate];

NSString从文件读取字符串

NSError \*error = nil;

NSString \*htmlPath = [[NSBundle mainBundle] pathForResource:@"hehe" ofType:@"html"];

NSString \*html = [[NSString alloc]

initWithContentsOfFile:htmlPath encoding:NSUTF8StringEncoding error:&error];

NSString解码NSData

NSError \*error = nil;

NSString \*htmlPath = [[NSBundle mainBundle] pathForResource:@"hehe" ofType:@"html"];

NSData \*htmlData = [[NSData alloc] initWithContentsOfFile:htmlPath];

[[NSString alloc] initWithData:htmlData encoding:NSUTF8StringEncoding]

[self.webView loadData:htmlData

MIMEType:@"text/html" textEncodingName:@"UTF-8" baseURL:bundleUrl]

KVO注册与回调

原型：

- (void)addObserver:(NSObject \*)observer

forKeyPath:(NSString \*)keyPath

options:(NSKeyValueObservingOptions)options

context:(void \*)context

options: 设定通知观察者时传递的属性值，是传改变前的呢，还是改变后的，具体的设定可以这儿：

NSKeyValueObservingOptionNew

NSKeyValueObservingOptionOld

NSKeyValueObservingOptionInitial

NSKeyValueObservingOptionPrior

可以多个（所以change是字典类型）：(NSKeyValue……OptionNew | NSKeyValueObservingOptionOld)

注意：在使用该方法并用if判断捕获需要监听的属性后，不要忘了在else调用super的该方法

使用：

[self addObserver:self.observer forKeyPath:@"appStatus"

options:NSKeyValueObservingOptionNew | NSKeyValueObservingOptionOld context:@"说点 啥好呢"];

添加回调

- (void)observeValueForKeyPath:(NSString \*)keyPath

ofObject:(id)object

change:(NSDictionary \*)change

context:(void \*)context

{

// CGFloat newCurveHeight = [(NSNumber\*)change[NSKeyValueChangeNewKey] floatValue];

CGFloat newCurveHeight = [(NSNumber\*)[change objectForKey:@"NSKeyValueChangeNewKey"] floatValue];

}

清除观察者:对象通过下面这个方法移除观察者：

- (void)removeObserver:(NSObject \*)anObserver forKeyPath:(NSString \*)keyPath

触摸事件

添加手势回调指向器：

UIPanGestureRecognizer \* pan = [[UIPanGestureRecognizer alloc]initWithTarget:self action:@selector(panHandle:)];

[self addGestureRecognizer:pan];

手势回调：

-(void)panHandle:(UIPanGestureRecognizer \*)pan{

CGPoint location = [pan locationInView:self];

if (pan.state == UIGestureRecognizerStateBegan) { …… }

}

scrollView自带的gesture：

NSLog(@"observeValueForKeyPath: \t虽然我没有为scrollView添加gesture，但是我依然能检测到： （%0.1f,%0.1f）", [myTableView.panGestureRecognizer locationInView:myTableView].x, [myTableView.panGestureRecognizer locationInView:myTableView].y);

普通View的：

[\_refreshView.gestureRecognizers[0] locationInView:\_refreshView].y

Transport Security

在info.plist中加入如下内容允许非安全请求

<key>NSAppTransportSecurity</key>

<dict> <key>NSAllowsArbitraryLoads</key> <true/> </dict>

-----------常用示范代码-------------------------------------------------

类

@interface MyException : NSObject

@property(readwrite,copy) NSString \*myReason;

@property(readwrite,strong) NSError \*offcialError;

@end

@implementation MyException

@synthesize myReason = apellation;

@synthesize offcialError;

- (id) init{

if (self = [super init]) { .... }

return (self);

}

- (void) setTire: (Tire \*) tire

atIndex: (int) index

{

}

+(Car \*) initWithTire: (Tire\*) tire

{

return [Car new];

}

@end

类别 - 非正式协议

@interface NSString (NumberConvenience)

- (NSNumber \*) lengthAsNumber;

@end

@implementation NSString (NumberConvenience)

- (NSNumber \*) lengthAsNumber

{

unsigned int length = [self length];

return ([NSNumber numberWithUnsignedInt: length]);

}

@end

使用：[dict setObject: [@"hello" lengthAsNumber]

forKey: @"hello"];

协议- 接口

@protocol NSCopying

@required

-(id) copyWithZone:(NSZone \*) zone;

@optional

@end

@interface Engine : NSObject <NSCopying，NSCoding>

//不需要特别声明协议（接口）中的方法，直接在实现类中实现就好了

-(void) someFunction:(id<NSCopying>) object;

@end

@implementation Engine

- (id) copyWithZone: (NSZone \*) zone // 》》》zone是NSZone类的对象，指向一块可供分配的内存区域

{

Engine \*engineCopy; // 》》》使用三层嵌套的复制方法，这里声明将返回的复制类的指针

// 》》》allocWithZone是一个“+”的类方法，应该被发给类，而不是对象

engineCopy = [[[self class] // 》》》使用self，是因为所有该类有子类，该方法能够自动返回子类对象

allocWithZone: zone]// 》》》使用可分配的内存 zone 来创建新对象

init]; // 》》》初始化

carCopy.name = self.name; // 》》》name在接口声明时设置为了自动复制

Engine \*engineCopy = [[engine copy] autorelease];//》》》说过的copy，alloc，new，都要注意随时释放

carCopy.engine = engineCopy;// 》》》因为在ARC计数中，或@property实现的setter中，持有engine都要+1

//》》》这里的tires实际是一个初始化后的新字典(init中字典new了，其他是置空)，在初始化时放入空对象，与源对象无关

tireCopy = [[self tireAtIndex: i] copy];

[tireCopy autorelease];

[carCopy setTire: tireCopy

atIndex: i];

return (engineCopy);

}

@end

闭包

声明与释放

int (^innerFunc)(int number) = ^(int number){ return (number \* number); };

Block\_release(innerFunc);

方法中的声明

-(void) animateBounce: (CGFloat)time complete: (int(^)(int a))handler;

别名

typedef int (^innerFunc)(int number);

innerFunc myFunc = ^(int number){ return (number \* number); };

NSLog(@"%d", myFunc());

后台运行

AnObject \*object = [[AnObject alloc]init];

[object performSelectorInBackground:@selector(backMethod:) withObject:innerObject];

脱离了主线程，需要在方法中自己添加释放池，硬性规定

C语言的指针函数

void (\*FunP)(int x);

* void MyFun(int x) //这里定义一个MyFun函数
* {  printf(“%d\n”,x); }
* FunP=&MyFun; //将MyFun函数的地址赋给FunP变量

(\*FunP)(20);

键值检索与写入 - 类似SQL

NSArray \*manufacturers;

manufacturers = [garage valueForKeyPath: @"cars.@distinctUnionOfObjects.make"];

// 》》》 分类（对象是表的一行，我取出一列）

NSLog (@"makers: %@", manufacturers);

NSArray \*keys = [NSArray arrayWithObjects: @"make", @"model", @"modelYear", nil];

//设置需要的属性组

NSDictionary \*carValues = [[garage valueForKeyPath: @"cars"] dictionaryWithValuesForKeys: keys]; //获取多列，列与列名为键值对（先行后列 —> 先列后行）

NSLog (@"Car values : %@", carValues);

//强行为对象成员赋值(不包含键路径，下一个方法可以)

[garage setValue: @"bunny" forKey: @"fluffy"];

[garage setValue: @"ooooo" forKeyPath: @"cars.model"];

//批量赋值（应该也有keyPathes的方法吧）

NSDictionary \*newValues = [NSDictionary dictionaryWithObjectsAndKeys:

@"Chevy", @"make",

@"Nova", @"model",

[NSNumber numberWithInt:1964], @"modelYear",

[NSNull null], @"mileage",

nil];

[[garage valueForKeyPath: @"cars"] setValuesForKeysWithDictionary: newValues];

NSLog (@"car with new values is %@", [garage valueForKeyPath: @"cars"]);

NSPredicate

// 使用SELF取交集

NSArray \*names1 = [NSArray arrayWithObjects: @"Herbie", @"Badger", @"Judge", @"Elvis", nil];

NSArray \*names2 = [NSArray arrayWithObjects: @"Judge", @"Paper Car", @"Badger", @"Phoenix", nil];

predicate = [NSPredicate predicateWithFormat: @"SELF IN %@", names1];

results = [names2 filteredArrayUsingPredicate: predicate];

NSLog (@"使用SELF： %@", predicate);

NSLog (@"使用SELF%@", results);

printf("\n");

NSMutableArray \*carsCopy = [cars mutableCopy];// 一个能够返回可变数组的复制方法——深拷贝

[carsCopy filterUsingPredicate: predicate];

NSLog (@"删除可变数组的不符合对象%@", carsCopy);

的