iOS①群: 275821025（2000人 满）
iOS②群: 190892815（2000人 满）
iOS③群: 335930567（1000人 满）
iOS码农群：106868842 （未满）

从自己笔记打印了一份开发经常用到的代码，供大家学习使用！

1.退回输入键盘
2.隐藏导航栏
3.屏幕变动检测
4.判断网络
5.时间转换
6.动画效果
7.图片压缩
8.正则表达式
9.图片上传
10.图库选择
11.AlertView背景
12.键盘遮挡问题
13.常用加密
14.程序结构混排加密
15.获取设备号
16.版本比较（等）

****
**********陆续更新**********
****

退回输入键盘

```
- (BOOL) textFieldShouldReturn:(id)textField{ [textField
resignFirstResponder];
}
```

CGRect

```
CGRect frame = CGRectMake (origin.x, origin.y,
size.width, size.height);矩形 NSStringFromCGRect(someCG)
把 CGRect 结构转变为格式化字符串; CGRectFromString(aString)
由字符串恢复出矩形;
CGRectInset(aRect) 创建较小或较大的矩形(中心点相同),+较小 -较大
CGRectIntersectsRect(rect1, rect2) 判断两矩形是否交叉,是否重
叠 CGRectZero 高度和宽度为零的/位于(0,0)的矩形常量
CGPoint & CGSize
CGPoint aPoint = CGPointMake(x, y); CGSize aSize =
CGSizeMake(width, height);
```

设置透明度

```
[myView setAlpha:value]; (0.0 < value < 1.0)
```

设置背景色

```
[myView setBackgroundColor:[UIColor redColor]];
(blackColor;darkGrayColor;lightGrayColor;
whiteColor;grayColor; redColor; greenColor; blueColor;
cyanColor;yellowColor;
magentaColor;orangeColor;purpleColor; brownColor;
clearColor; )
```

自定义颜色

```
UIColor *newColor = [[UIColor alloc]
                     initWithRed:(float) green:(float)
blue:(float) alpha:(float)];
0.0~1.0
```

竖屏
320X480

横屏
480X320

状态栏高（显示时间和网络状态）20 像素

导航栏、工具栏高(返回）44 像素

隐藏状态栏
```
[[UIApplication shareApplication] setStatusBarHidden: YES
animated:NO]
```

横屏
```
[[UIApplication shareApplication]
setStatusBarOrientation:UIInterfaceOrientationLandscapeRi
ght].
```

屏幕变动检测
```
orientation == UIInterfaceOrientationLandscapeLeft
```

全屏
```
window=[[UIWindow alloc] initWithFrame:[UIScreen
mainScreen] bounds];
```

自动适应父视图大小:
```
aView.autoresizingSubviews = YES;
aView.autoresizingMask = (UIViewAutoresizingFlexibleWidth
|

UIViewAutoresizingFlexibleHeight);
```

定义按钮
```
UIButton *scaleUpButton = [UIButton
buttonWithType:UIButtonTypeRoundedRect]; [scaleUpButton
setTitle:@"放 大" forState:UIControlStateNormal];
scaleUpButton.frame = CGRectMake(40, 420, 100, 40);
[scaleUpButton addTarget:self
```

```
                    action:@selector(scaleUp)
forControlEvents:UIControlEventTouchUpInside];
```

设置视图背景图片

```
UIImageView *aView;
[aView setImage:[UIImage imageNamed:@”name.png”]];
view1.backgroundColor = [UIColor colorWithPatternImage:
[UIImage imageNamed:@"image1.png"]];
```

活动表单

```
<UIActionSheetDelegate>
- (IBActive) someButtonPressed:(id) sender {
    UIActionSheet *actionSheet = [[UIActionSheet alloc]
initWithTitle:@”Are you sure?”

    delegate:self

cancelButtonTitle:@”No way!”
destructiveButtonTitle:@”Yes, I’m Sure!”
otherButtonTitles:nil];
    [actionSheet showInView:self.view];
    [actionSheet release]; }
```

警告视图

```
<UIAlertViewDelegate>
- (void) actionSheet:(UIActionSheet *)actionSheet
didDismissWithButtonIndex:(NSInteger) buttonIndex
{
    if(buttonIndex != [actionSheet cancelButtonIndex]) {
        NSString *message = [[NSString alloc]
initWithFormat:@”You can breathe easy, everything went
OK.”];
        UIAlertView *alert = [[UIAlertView alloc]
initWithTitle:@”Something was done”
                                  [alert show]; [alert
release]; [message release];
                                  }
}
```

动画效果

```
-(void)doChange:(id)sender {
  if(view2 == nil)
```

```
  {
      [self loadSec];
  }
    [UIView beginAnimations:nil context:NULL];
    [UIView setAnimationDuration:1];
    [UIView setAnimationTransition:([view1 superview]?
UIViewAnimationTransitionFlipFromLeft:UIViewAnimationTran
sitionFlipFromRigh t)forView:self.view cache:YES];
    if([view1 superview]!= nil) {
        [view1 removeFromSuperview]; [self.view
addSubview:view2];
    }else {
        [view2 removeFromSuperview]; [self.view
addSubview:view1]; }
    [UIView commitAnimations];
}
                                Table View
<UITableViewDateSource> #pragma mark -
#pragma mark Table View Data Source Methods //指定分区中的
行数,默认为 1
                                - (NSInteger)tableView:
(UITableView *)tableView numberOfRowsInSection:
(NSInteger)section
        {
            return [self.listData count]; }
                                设置每一行 cell 显示的内容
                                - (UITableViewCell
*)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath
        {
            static NSString *SimpleTableIndentifier =
@"SimpleTableIndentifier"; UITableViewCell *cell =
dequeueReusableCellWithIdentifier:SimpleTableIndentifier]
;
            if (cell == nil) {
                cell = [[[UITableViewCell alloc]
initWithStyle:UITableViewCellStyleSubtitle
reuseIdentifier:SimpleTableIndentifier]
                        autorelease];
            }
            UIImage *image = [UIImage
imageNamed:@"13.gif"]; cell.imageView.image = image;
            [tableView
```

```
            NSUInteger row = [indexPath row];
cell.textLabel.text = [listData objectAtIndex:row];
            cell.textLabel.font = [UIFont
boldSystemFontOfSize:20];
            if(row < 5)
            cell.detailTextLabel.text = @"Best friends";
else
            cell.detailTextLabel.text = @"friends";
return cell;
            }
```

判断邮箱格式是否正确的代码

利用正则表达式验证

```
-(BOOL)isValidateEmail:(NSString *)email
{
    NSString *emailRegex = @"[A-Z0-9a-z._%+-]+@[A-Za-
z0-9.-]+\\.[A-Za-z]{2,4}";
    NSPredicate *emailTest = [NSPredicate
predicateWithFormat:@"SELF MATCHES%@",emailRegex];
    return [emailTest evaluateWithObject:email];
}
```

图片压缩

用法: UIImage *yourImage= [self imageWithImageSimple:image scaledToSize:CGSizeMake(210.0, 210.0)];

压缩图片

```
- (UIImage*)imageWithImageSimple:(UIImage*)image
scaledToSize:(CGSize)newSize
{
    Create a graphics image context
    UIGraphicsBeginImageContext(newSize);
    Tell the old image to draw in this newcontext, with
the desired
    new size
    [image
drawInRect:CGRectMake(0,0,newSize.width,newSize.height)];
    Get the new image from the context
    UIImage* newImage =
UIGraphicsGetImageFromCurrentImageContext();
```

```
    End the context
    UIGraphicsEndImageContext();
    Return the new image.
    return newImage;
}

亲测可用的图片上传代码
- (IBAction)uploadButton:(id)sender {
    UIImage *image = [UIImage imageNamed:@"1.jpg"]; 图片名
    NSData *imageData = UIImageJPEGRepresentation(image,
0.5);压缩比例
    NSLog(@"字节数:%i",[imageData length]);
    post url
    NSString *urlString = @"http://192.168.1.113:8090/
text/UploadServlet";
    服务器地址
    setting up the request object now
    NSMutableURLRequest *request = [[NSMutableURLRequest
alloc] init] ;
    [request setURL:[NSURL URLWithString:urlString]];
    [request setHTTPMethod:@"POST"];

    NSString *boundary = [NSString
stringWithString:@"---------------------------14737809831
466499882746641449"];

    NSString *contentType = [NSString
stringWithFormat:@"multipart/form-data;boundary=
%@",boundary];

    [request addValue:contentType forHTTPHeaderField:
@"Content-Type"];
    NSMutableData *body = [NSMutableData data];

    [body appendData:[[NSString stringWithFormat:@"\r\n--
%@\r\n",boundary]
dataUsingEncoding:NSUTF8StringEncoding]];

    [body appendData:[[NSString
stringWithString:@"Content-Disposition:form-data; name=
\"userfile\"; filename=\"2.png\"\r\n"]
dataUsingEncoding:NSUTF8StringEncoding]]; 上传上去的图片名字
```

```
    [body appendData:[[NSString
stringWithString:@"Content-Type: application/octet-stream
\r\n\r\n"] dataUsingEncoding:NSUTF8StringEncoding]];

    [body appendData:[NSData dataWithData:imageData]];

    [body appendData:[[NSString stringWithFormat:@"\r\n--
%@--\r\n",boundary]
dataUsingEncoding:NSUTF8StringEncoding]];

    [request setHTTPBody:body];

    NSLog(@"1-body:%@",body);
    NSLog(@"2-request:%@",request);

    NSData *returnData = [NSURLConnection
sendSynchronousRequest:request returningResponse:nil
error:nil];

    NSString *returnString = [[NSString alloc]
initWithData:returnData encoding:NSUTF8StringEncoding];

    NSLog(@"3-测试输出：%@",returnString);

    给imageView加载图片

    UIImage *myImage = [UIImage imageNamed:@"1.jpg"];
    [imageView setImage:myImage];
    [self.view addSubview:imageView];

   对图库的操作
    选择相册：


UIImagePickerControllerSourceTypesourceType=UIImagePicker
ControllerSourceTypeCamera;
    if (!
[UIImagePickerControllerisSourceTypeAvailable:UIImagePick
erControllerSourceTypeCamera]) {

sourceType=UIImagePickerControllerSourceTypePhotoLibrary;
    }
    UIImagePickerController * picker =
[[UIImagePickerControlleralloc]init];
```

```
    picker.delegate = self;
    picker.allowsEditing=YES;
    picker.sourceType=sourceType;
    [self presentModalViewController:picker
animated:YES];
```

选择完毕：

```
    -(void)imagePickerController:
(UIImagePickerController*)pickerdidFinishPickingMediaWith
Info:(NSDictionary *)info
    {
        [picker dismissModalViewControllerAnimated:YES];
        UIImage * image=[info
objectForKey:UIImagePickerControllerEditedImage];
        [self performSelector:@selector(selectPic:)
withObject:imageafterDelay:0.1];
    }
    -(void)selectPic:(UIImage*)image
    {
        NSLog(@"image%@",image);
        imageView = [[UIImageView alloc]
initWithImage:image];
        imageView.frame = CGRectMake(0, 0,
image.size.width, image.size.height);
        [self.viewaddSubview:imageView];
        [self
performSelectorInBackground:@selector(detect:)
withObject:nil];
    }
```

detect为自己定义的方法，编辑选取照片后要实现的效果

取消选择：

```
    -(void)imagePickerControllerDIdCancel:
(UIImagePickerController*)picker
    {
        [picker dismissModalViewControllerAnimated:YES];
    }
```

跳到下个View

```
    nextWebView = [[WEBViewController alloc]
initWithNibName:@"WEBViewController" bundle:nil];
    [self presentModalViewController:nextWebView
animated:YES];
    创建一个UIBarButtonItem右边按钮
    UIBarButtonItem *rightButton = [[UIBarButtonItem
alloc] initWithTitle:@"右边"
style:UIBarButtonItemStyleDone target:self
action:@selector(clickRightButton)];
    [self.navigationItem
setRightBarButtonItem:rightButton];
    设置navigationBar隐藏
    self.navigationController.navigationBarHidden = YES;
    iOS开发之UIlabel多行文字自动换行 （自动折行）
    UIView *footerView = [[UIView
alloc]initWithFrame:CGRectMake(10, 100, 300, 180)];
    UILabel *label = [[UILabel
alloc]initWithFrame:CGRectMake(10, 100, 300, 150)];
    label.text = @"Hello world! Hello world!Hello world!
Hello world! Hello world! Hello world! !";
    背景颜色为红色
    label.backgroundColor = [UIColor redColor];
    设置字体颜色为白色
    label.textColor = [UIColor whiteColor];
    文字居中显示
    label.textAlignment = UITextAlignmentCenter;
    自动折行设置
    label.lineBreakMode = UILineBreakModeWordWrap;
    label.numberOfLines = 0;

   代码生成button

    CGRect frame = CGRectMake(0, 400, 72.0, 37.0);
    UIButton *button = [UIButton
buttonWithType:UIButtonTypeRoundedRect];
    button.frame = frame;
    [button setTitle:@"新添加的按钮" forState:
UIControlStateNormal];
    button.backgroundColor = [UIColor clearColor];
    button.tag = 2000;
    [button addTarget:self
action:@selector(buttonClicked:)
forControlEvents:UIControlEventTouchUpInside];
```

```
    [self.view addSubview:button];
```

让某个控件在View的中心位置显示

```
(某个控件，比如label，View) label.center =
self.view.center;
```

好看的文字处理

以tableView中cell的textLabel为例子：

```
    cell.backgroundColor =
[UIColorscrollViewTexturedBackgroundColor];
    设置文字的字体
    cell.textLabel.font = [UIFont
fontWithName:@"AmericanTypewriter" size:100.0f];
    设置文字的颜色
    cell.textLabel.textColor = [UIColor orangeColor];
    设置文字的背景颜色
    cell.textLabel.shadowColor = [UIColor whiteColor];
    设置文字的显示位置
    cell.textLabel.textAlignment = UITextAlignmentCenter;
```

隐藏Status Bar

读者可能知道一个简易的方法,那就是在程序的viewDidLoad中加入

```
    [[UIApplication
sharedApplication]setStatusBarHidden:YES animated:NO];
```

更改AlertView背景

```
     UIAlertView *theAlert = [[[UIAlertViewalloc]
initWithTitle:@"Atention"

message: @"I'm a Chinese!"

delegate:nil

cancelButtonTitle:@"Cancel"

otherButtonTitles:@"Okay",nil] autorelease];
    [theAlert show];
    UIImage *theImage =
[UIImageimageNamed:@"loveChina.png"];
```

```
    theImage = [theImage
stretchableImageWithLeftCapWidth:0topCapHeight:0];
    CGSize theSize = [theAlert frame].size;
    UIGraphicsBeginImageContext(theSize);
    [theImage drawInRect:CGRectMake(5, 5,
theSize.width-10, theSize.height-20)];这个地方的大小要自己调
    整，以适应alertview的背景颜色的大小。
    theImage =
UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    theAlert.layer.contents = (id)[theImage CGImage];

    键盘透明
    textField.keyboardAppearance =
UIKeyboardAppearanceAlert;

    状态栏的网络活动风火轮是否旋转
    [UIApplication
sharedApplication].networkActivityIndicatorVisible，默认值
是NO。

    截取屏幕图片
    创建一个基于位图的图形上下文并指定大小为CGSizeMake(200,400)

    UIGraphicsBeginImageContext(CGSizeMake(200,400));
    renderInContext 呈现接受者及其子范围到指定的上下文
    [self.view.layer
renderInContext:UIGraphicsGetCurrentContext()];
    返回一个基于当前图形上下文的图片
    UIImage *aImage =
UIGraphicsGetImageFromCurrentImageContext();
    移除栈顶的基于当前位图的图形上下文
    UIGraphicsEndImageContext();
    以png格式返回指定图片的数据
    imageData = UIImagePNGRepresentation(aImage);

    更改cell选中的背景
    UIView *myview = [[UIView alloc] init];
    myview.frame = CGRectMake(0, 0, 320, 47);
    myview.backgroundColor =
[UIColorcolorWithPatternImage:[UIImage
imageNamed:@"0006.png"]];
```

```
    cell.selectedBackgroundView = myview;
```

显示图像

```
    CGRect myImageRect = CGRectMake(0.0f, 0.0f, 320.0f,
109.0f);
    UIImageView *myImage = [[UIImageView alloc]
initWithFrame:myImageRect];
    [myImage setImage:[UIImage
imageNamed:@"myImage.png"]];
    myImage.opaque = YES; opaque是否透明
    [self.view addSubview:myImage];
```

能让图片适应框的大小（待确认）

```
    NSString*imagePath = [[NSBundle mainBundle]
pathForResource:@"XcodeCrash"ofType:@"png"];
    UIImage *image = [[UIImage
alloc]initWithContentsOfFile:imagePath];
    UIImage *newImage= [image transformWidth:80.f height:
240.f];
    UIImageView *imageView = [[UIImageView
alloc]initWithImage:newImage];
    [newImagerelease];
    [image release];
    [self.view addSubview:imageView];
```

实现点击图片进行跳转的代码：生成一个带有背景图片的button，给button绑定想要的事件！

```
    UIButton *imgButton=[[UIButton
alloc]initWithFrame:CGRectMake(0, 0, 120, 120)];
    [imgButton setBackgroundImage:(UIImage *)
[self.imgArray objectAtIndex:indexPath.row]
forState:UIControlStateNormal];
    imgButton.tag=[indexPath row];
    [imgButton addTarget:self
action:@selector(buttonClick:)
forControlEvents:UIControlEventTouchUpInside];
```

多线程和结束后的更新UI操作

```
dispatch_async(dispatch_get_global_queue(0, 0), ^{
    耗时操作
```

```
    dispatch_async(dispatch_get_main_queue(), ^{
        更新UI操作

    });

});
```

修改PlaceHolder的默认颜色

```
[username_text setValue:[UIColor colorWithRed:1 green:1
blue:1 alpha:0.5]
forKeyPath:@"_placeholderLabel.textColor"];
```

页面上移解决文本框被键盘弹出挡住的问题

textfield的函数

```
-(void)touchesBegan:(NSSet *)touches withEvent:(UIEvent
*)event{
    [username_text resignFirstResponder];
    [password_text resignFirstResponder];
    When the user presses return, take focus away from
the text field so that the keyboard is dismissed.
    NSTimeInterval animationDuration = 0.30f;
    [UIView beginAnimations:@"ResizeForKeyboard"
context:nil];
    [UIView setAnimationDuration:animationDuration];
    CGRect rect = CGRectMake(0.0f, 0.0f,
self.view.frame.size.width, self.view.frame.size.height);
    self.view.frame = rect;
    [UIView commitAnimations];
}

- (BOOL)textFieldShouldReturn:(UITextField *)textField
{
    When the user presses return, take focus away from
the text field so that the keyboard is dismissed.
    NSTimeInterval animationDuration = 0.30f;
    [UIView beginAnimations:@"ResizeForKeyboard"
context:nil];
    [UIView setAnimationDuration:animationDuration];
    CGRect rect = CGRectMake(0.0f, 0.0f,
self.view.frame.size.width, self.view.frame.size.height);
    self.view.frame = rect;
```

```
    [UIView commitAnimations];
    [textField resignFirstResponder];
    return YES;
}

- (void)textFieldDidBeginEditing:(UITextField *)textField
{
    CGRect frame = password_text.frame;
    int offset = frame.origin.y + 32 -
(self.view.frame.size.height - 216.0);//键盘高度216
    NSTimeInterval animationDuration = 0.30f;
    [UIView beginAnimations:@"ResizeForKeyBoard"
context:nil];
    [UIView setAnimationDuration:animationDuration];
    float width = self.view.frame.size.width;
    float height = self.view.frame.size.height;
    if(offset > 0)
    {
        CGRect rect = CGRectMake(0.0f, -
offset,width,height);
        self.view.frame = rect;
    }
    [UIView commitAnimations];
}
```

iOS代码加密常用加密方式，常见的iOS代码加密常用加密方式算法包括MD5加密、AES加密、BASE64加密：

MD5 iOS代码加密

创建MD5类，代码如下

```
#import <Foundation/Foundation.h>
@interface CJMD5 : NSObject

+(NSString *)md5HexDigest:(NSString *)input;

@end

#import "CJMD5.h"
#import <CommonCrypto/CommonDigest.h>

@implementation CJMD5
```

```
+(NSString *)md5HexDigest:(NSString *)input{


    const char* str = [input UTF8String];

    unsigned char result[CC_MD5_DIGEST_LENGTH];

    CC_MD5(str, strlen(str), result);

    NSMutableString *ret = [NSMutableString
stringWithCapacity:CC_MD5_DIGEST_LENGTH];



    for(int i = 0; i<CC_MD5_DIGEST_LENGTH; i++) {

        [ret appendFormat:@"%02X",result];

    }

    return ret;

}
@end
```

MD5是不可逆的只有加密没有解密，iOS代码加密使用方式如下

```
NSString *userName = @"cerastes";
NSString *password = @"hello Word";

MD5加密
NSString *md5 = [CJMD5 md5HexDigest:password];

NSLog(@"%@",md5);
END
```

AES加密iOS代码加密

AES加密iOS代码加密使用方法

```
AES加密
NSString *encryptedData = [AESCrypt encrypt:userName
password:password];加密
```

```
NSString *message = [AESCrypt decrypt:encryptedData
password:password]; 解密

NSLog(@"加密结果 = %@",encryptedData);

NSLog(@"解密结果 = %@",message);
END
```

BASE64加密iOS代码加密

BASE64加密iOS代码加密添加如下方法

```
.h

+ (NSString*)encodeBase64String:(NSString *)input;

+ (NSString*)decodeBase64String:(NSString *)input;

+ (NSString*)encodeBase64Data:(NSData *)data;

+ (NSString*)decodeBase64Data:(NSData *)data;

.m

+ (NSString*)encodeBase64String:(NSString * )input {

    NSData *data = [input
dataUsingEncoding:NSUTF8StringEncoding
allowLossyConversion:YES];

    data = [GTMBase64 encodeData:data];

    NSString *base64String = [[NSString alloc]
initWithData:data encoding:NSUTF8StringEncoding];

    return base64String;

}


+ (NSString*)decodeBase64String:(NSString * )input {
```

```
    NSData *data = [input
dataUsingEncoding:NSUTF8StringEncoding
allowLossyConversion:YES];

    data = [GTMBase64 decodeData:data];

    NSString *base64String = [[NSString alloc]
initWithData:data encoding:NSUTF8StringEncoding];

    return base64String;

}

+ (NSString*)encodeBase64Data:(NSData *)data {

    data = [GTMBase64 encodeData:data];

    NSString *base64String = [[NSString alloc]
initWithData:data encoding:NSUTF8StringEncoding];

    return base64String;

}

+ (NSString*)decodeBase64Data:(NSData *)data {

    data = [GTMBase64 decodeData:data];

    NSString *base64String = [[NSString alloc]
initWithData:data encoding:NSUTF8StringEncoding];

    return base64String;

}
```

BASE64加密iOS代码加密使用方法

BASE64加密

```
NSString *baseEncodeString = [GTMBase64
encodeBase64String:password];

NSString *baseDecodeString = [GTMBase64
decodeBase64String:baseEncodeString];

NSLog(@"baseEncodeString = %@",baseEncodeString);
```

```
NSLog(@"baseDecodeString = %@",baseDecodeString);
```

RSA加密：

RSA是目前最有影响力的公钥加密算法，它能够抵抗到目前为止已知的绝大多数密码攻击，已被ISO推荐为公钥数据加密标准。

RSA公开密钥密码体制。所谓的公开密钥密码体制就是使用不同的加密密钥与解密密钥，是一种“由已知加密密钥推导出解密密钥在计算上是不可行的”密码体制。
通常是先生成一对RSA 密钥，其中之一是保密密钥，由用户保存；另一个为公开密钥，可对外公开，甚至可在网络服务器中注册。为提高保密强度，RSA密钥至少为500位长，一般推荐使用1024位。这就使加密的计算量很大。为减少计算量，在传送信息时，常采用传统加密方法与公开密钥加密方法相结合的方式，即信息采用改进的DES或IDEA对话密钥加密，然后使用RSA密钥加密对话密钥和信息摘要。对方收到信息后，用不同的密钥解密并可核对信息摘要。
RSA算法是第一个能同时用于加密和数字签名的算法，也易于理解和操作。
RSA是被研究得最广泛的公钥算法.

RSA算法是一种非对称密码算法，所谓非对称，就是指该算法需要一对密钥，使用其中一个加密，则需要用另一个才能解密。

RSA的算法涉及三个参数，n、e1、e2。
其中，n是两个大质数p、q的积，n的二进制表示时所占用的位数，就是所谓的密钥长度。
e1和e2是一对相关的值，e1可以任意取，但要求e1与(p-1) *(q-1) 互质；再选择e2，要求 (e2*e1) mod((p-1) *(q-1)) =1。
（n，e1）,(n，e2）就是密钥对。其中(n，e1）为公钥，(n，e2）为私钥。[1]
RSA加解密的算法完全相同，设A为明文，B为密文，则：A=B^e2 mod n；B=A^e1 mod n；（公钥加密体制中，一般用公钥加密，私钥解密）
e1和e2可以互换使用，即：
A=B^e1 mod n；B=A^e2 mod n;

1）本地数据加密

对NSUserDefaults，sqlite存储文件数据加密，保护帐号和关键信息。）
URL编码加密

对程序中出现的URL进行编码加密，防止URL被静态分析

2）网络传输数据加密

对客户端传输数据提供加密方案，有效防止通过网络接口的拦截获取

3）方法体，方法名高级混淆

对应用程序的方法名和方法体进行混淆，保证源码被逆向后无法解析代码

4）程序结构混排加密

对应用程序逻辑结构进行打乱混排，保证源码可读性降到最低

返回指定范围的随机数(m-n之间)的公式

```
Math.random()*(n-m)+m
```

防止被Iframe嵌套

```
if(top != self){
    location.href = ”about:blank”;
}
```

```
/**
 *  HTTP GET 请求
 **/
+(NSData *) doHttpGet:(NSString *)url
{
    NSURL *uri = [NSURL URLWithString:url];
    NSMutableURLRequest *request = [[NSMutableURLRequest
alloc] initWithURL:uri];
    [request setHTTPMethod: @"GET" ];
    NSData *returnData = [NSURLConnection
sendSynchronousRequest: request returningResponse: nil
error: nil];
    return returnData;
}
```

```
/**
 *  HTTP POST请求
 **/
+(NSData *) doHttpPost:(NSString *)url withString:
(NSString *)param
{
    NSData *data = nil;
```

```
    if(param != nil && [param isEqualToString:@""] == NO)
{
        param = [param
stringByAddingPercentEscapesUsingEncoding:CFStringConvert
EncodingToNSStringEncoding(kCFStringEncodingGB_18030_2000
)];
        data = [param
dataUsingEncoding:CFStringConvertEncodingToNSStringEncodi
ng(kCFStringEncodingGB_18030_2000)];
    }
    //调用withParam NSData*类型的方法.
    return [self doHttpPost:url withParam:data];
}



/**
 *  HTTP POST请求
 **/
+(NSData *) doHttpPost:(NSString *)url withParam:(NSData
*)param
{
    新建请求
    NSURL *uri = [NSURL URLWithString:url];
    NSMutableURLRequest *request = [NSMutableURLRequest
requestWithURL:uri
cachePolicy:NSURLRequestReloadIgnoringLocalCacheData
timeoutInterval:40.0];
    设置请求参数
    [request setHTTPMethod:@"POST"];
    [request addValue:@"application/x-www-form-
urlencoded" forHTTPHeaderField:@"Content-Type"];
    if(param != nil)
        [request setHTTPBody:param];
    打开访问网络的状态提示
    [[UIApplication sharedApplication]
setNetworkActivityIndicatorVisible:YES];
    请求链接
    NSError *error = nil;
    NSData *retData = [NSURLConnection
sendSynchronousRequest:request returningResponse:nil
error:nil];
    NSLog(@"%d: %@", error.code, error.description);
    关闭访问网络的状态提示
```

```
    [[UIApplication sharedApplication]
setNetworkActivityIndicatorVisible:NO];
    返回结果
    return retData;
}



/**
 *  获取网络图片
 **/
+(UIImage *) getImageFromUrl:(NSString *)url
{
    if(url == nil || [url isEqualToString:@""]){
        return nil;
    }
    url = StringByTrimWhiteSpace(url);
    NSData *imageData = [[NSData
alloc]initWithContentsOfURL:[NSURL URLWithString:url]];
    UIImage *image =[[UIImage alloc]
initWithData:imageData];
    return image;
}

/**
 *  获取网络图片的内容
 **/
+(NSData *)getImageDataFromUrl:(NSString *)url
{
    if(url == nil || [url isEqualToString:@""]){
        return nil;
    }

    NSData *imageData = [[NSData
alloc]initWithContentsOfURL:[NSURL URLWithString:url]];
    return imageData;
}

#pragma mark - 字符串处理
/**
 *  利用正则表达示获取字符串的匹配结果
 **/
+(NSString *) getRegExpressResult:(NSString *)source
regExp:(NSString *)regExp
```

```
{
    NSString *temp = [NSString stringWithFormat:@"%@",
source];

    NSRegularExpression *regex = [NSRegularExpression
regularExpressionWithPattern:regExp
options:NSRegularExpressionCaseInsensitive error:nil];

    if (regex != nil) {
        NSTextCheckingResult *firstMatch = [regex
firstMatchInString:temp options:0 range:NSMakeRange(0,
[temp length])];

        if (firstMatch) {
            NSRange resultRange = [firstMatch
rangeAtIndex:0];
            截取数据
            NSString *result = [temp
substringWithRange:resultRange];
            返回结果
            return result;
        }
    }
    return @"";
}


/**
 *  匹配字符串中整个HTML标记的内容
 **/
+(NSString *) getHtmlText:(NSString *)source tagName:
(NSString *)tag
{
    NSString *regexp = [NSString stringWithFormat:@"<\\s*
%@\\s+([^>]*)\\s*>([^/%@>]*</%@>)?", tag, tag, tag];
    return [BaseFunction getRegExpressResult:source
regExp:regexp];
}


/**
 *  匹配HTML标记内容中的属性值
 **/
```

```
+(NSString *) getHtmlTagAttr:(NSString *)tagContext
attrName:(NSString *)attr
{
    NSString *regexp = [NSString stringWithFormat: @"%@\
\s*=\\s*?(['\"][^'\"]*?)['\"]", attr];
    NSString *result = [BaseFunction
getRegExpressResult:tagContext regExp:regexp];
    替换
    NSString *oldstr = [NSString stringWithFormat:@"%@=
\"", attr];
    NSString *newstr = [result
stringByReplacingOccurrencesOfString:oldstr
withString:@""];
    newstr = [newstr substringToIndex:[newstr length] -
1];
    return newstr;
}

/**
 *  获取HTML标记的文本
 **/
+(NSString *) getHTmlTagText:(NSString *)tagContext
{
    NSString *regExp = @"<\\s*\\w+\\s+([^>]*)\\s*>";
    NSRegularExpression *regex = [NSRegularExpression
regularExpressionWithPattern:regExp
options:NSRegularExpressionCaseInsensitive error:nil];

    NSTextCheckingResult *firstMatch = [regex
firstMatchInString:tagContext options:0
range:NSMakeRange(0, [tagContext length])];
    NSRange resultRange = [firstMatch rangeAtIndex:0];
    NSString *newStr = [tagContext
substringFromIndex:resultRange.length];

    regExp = @"</\\w+\\s*>";
    regex = [NSRegularExpression
regularExpressionWithPattern:regExp
options:NSRegularExpressionCaseInsensitive error:nil];
    firstMatch = [regex firstMatchInString:newStr
options:0 range:NSMakeRange(0, [newStr length])];
    resultRange = [firstMatch rangeAtIndex:0];
```

```
    return [newStr
substringToIndex:resultRange.location];
}

/**
 *  替换HTML标签
 **/
+(NSString *) replaceHtmlTag:(NSString *)source
{
    source = [BaseFunction replaceString:source
byRegexp:@"<[^>]+>"];
    return [BaseFunction replaceString:source
byRegexp:@"</[^>]+>"];
}

+(NSString *) replaceString:(NSString *)source byRegexp:
(NSString *)exp
{
    NSRegularExpression *regex = [NSRegularExpression
regularExpressionWithPattern:exp options:0 error:nil];

    if(regex == nil)
        return source;

    NSString *ret = [NSString stringWithFormat:@"%@",
source];
    NSArray *array = [regex matchesInString:ret
options:NSMatchingReportProgress range:NSMakeRange(0,
[ret length])];
    for(int i = (int)[array count] - 1; i >= 0; i--)
    {
        NSTextCheckingResult *tcr = [array
objectAtIndex:i];
        NSRange range = [tcr range];
        ret = [ret
stringByReplacingCharactersInRange:range withString:@""];
    }
    return ret;
}



/**
 *  正则验证
 **/
```

```
+(BOOL) string:(NSString *)source MatchRegex:(NSString *)
exp
{
    NSPredicate *predicate = [NSPredicate
predicateWithFormat:@"SELF MATCHES %@", exp];
    return [predicate evaluateWithObject:source];
}



/**
 *  获取正则表达式中匹配的个数
 **/
+ (NSInteger) getMatchCount:(NSString *)text inRegx:
(NSString *)exp
{
    NSRegularExpression *regex = [NSRegularExpression
regularExpressionWithPattern:exp options:0 error:nil];

    int count = 0;
    if (regex != nil) {
        NSArray *array = [regex matchesInString:text
options:NSMatchingReportProgress range:NSMakeRange(0,
[text length])];

        for(int i=0; i< [array count]; i++)
        {
            NSTextCheckingResult *tcr = [array
objectAtIndex:i];
            NSRange range = [tcr range];
            count += range.length;
        }
    }
    return count;
}



/**
 *  替换XML敏感字符
 **/
+ (NSString *) replaceXMLSensitiveLettler:(NSString
*)text
{
```

```
    NSString *tmp = [text
stringByReplacingOccurrencesOfString:@"&"
withString:@"&amp;"];
    tmp = [tmp stringByReplacingOccurrencesOfString:@"<"
withString:@"&lt;"];
    tmp = [tmp stringByReplacingOccurrencesOfString:@">"
withString:@"&gt;"];
    return tmp;
}

/**
 *  分离坐标
 **/
+(void) separateCoordinate:(NSString *)coord lat:
(NSString **)lat lng:(NSString **)lng
{
    *lng = @"", *lat = @"";
    验证数据的合法性
    if(coord == nil){ return; }
    coord = StringByTrimWhiteSpace(coord);
    if(IsStringEmpty(coord)){
        return;
    }

    将坐标分开
    NSArray *coordArray = [coord
componentsSeparatedByString:@","];
    if([coordArray count]>0)
        *lng = [coordArray objectAtIndex:0];
        if([coordArray count]>1)
            *lat = [coordArray objectAtIndex:1];
            }


/**
 *  从文件路径中分解出文件名
 **/
+ (NSString *) splitFileNameForPath:(NSString *)filePath
{
    NSArray *array = [filePath
componentsSeparatedByString:@"/"];
    return [array lastObject];
}
```

```
/**
 *  从文件路径中分解出文件的扩展名
 **/
+ (NSString *) getFileExtension:(NSString *)filePath
{
    NSString *fileName = [self
splitFileNameForPath:filePath];
    NSArray *array = [fileName
componentsSeparatedByString:@"."];
    return [NSString stringWithFormat:@".%@",[array
lastObject]];
}

/**
 *  获取设备型号
 **/
+ (NSString *) platform
{
    size_t size;
    sysctlbyname("hw.machine", NULL, &size, NULL, 0);
    char *machine = (char *)malloc(size);
    sysctlbyname("hw.machine", machine, &size, NULL, 0);
    NSString *platform = [NSString
stringWithCString:machine encoding:NSUTF8StringEncoding];
    free(machine);
    NSRange range = [platform rangeOfString:@","];
    return [platform substringToIndex:range.location];
}



/**
 *  MD5加密
 **/
+ (NSString *)md5Digest:(NSString *)str {
    const char *cStr = [str UTF8String];
    unsigned char result[CC_MD5_DIGEST_LENGTH];
    CC_MD5(cStr, (CC_LONG)strlen(cStr), result); // This
is the md5 call
    NSMutableString *md5Result = [[NSMutableString alloc]
init];
    for (int i = 0; i < CC_MD5_DIGEST_LENGTH; i++) {
```

```
        [md5Result appendFormat:@"%02x", result[i]];
    }

    return md5Result;
        return [NSString stringWithFormat:
                @"%02x%02x%02x%02x%02x%02x%02x%02x%02x
%02x%02x%02x%02x%02x%02x%02x",
                result[0], result[1], result[2],
result[3],
                result[4], result[5], result[6],
result[7],
                result[8], result[9], result[10],
result[11],
                result[12], result[13], result[14],
result[15]];
}

+ (NSString *)SHA1:(NSString *)str {
    const char *cStr = [str UTF8String];
    NSData *data = [NSData dataWithBytes:cStr
length:str.length];
    uint8_t digest[CC_SHA1_DIGEST_LENGTH];
    CC_SHA1(data.bytes, (CC_LONG)data.length, digest);
    NSMutableString *result = [[NSMutableString alloc]
init];
    for(int i = 0; i < CC_SHA1_DIGEST_LENGTH; i++) {
        [result appendFormat:@"%02x", digest[i]];
    }

    return result;
}

判断是否为整形
+ (BOOL)isPureInt:(NSString *)string{
    NSScanner* scan = [NSScanner
scannerWithString:string];
    int val;
    return [scan scanInt:&val] && [scan isAtEnd];
}

判断是否为浮点形
+ (BOOL)isPureFloat:(NSString *)string{
    NSScanner* scan = [NSScanner
scannerWithString:string];
```

```
    float val;
    return [scan scanFloat:&val] && [scan isAtEnd];
}

/**
 *  版本比较
 **/
+ (BOOL)isVersion:(NSString*)versionA biggerThanVersion:
(NSString*)versionB
{
    NSArray *arrayNow = [versionB
componentsSeparatedByString:@"."];
    NSArray *arrayNew = [versionA
componentsSeparatedByString:@"."];
    BOOL isBigger = NO;
    NSInteger i = arrayNew.count > arrayNow.count?
arrayNow.count : arrayNew.count;
    NSInteger j = 0;
    BOOL hasResult = NO;
    for (j = 0; j < i; j ++) {
        NSString* strNew = [arrayNew objectAtIndex:j];
        NSString* strNow = [arrayNow objectAtIndex:j];
        if ([strNew integerValue] > [strNow
integerValue]) {
            hasResult = YES;
            isBigger = YES;
            break;
        }
        if ([strNew integerValue] < [strNow
integerValue]) {
            hasResult = YES;
            isBigger = NO;
            break;
        }
    }
    if (!hasResult) {
        if (arrayNew.count > arrayNow.count) {
            NSInteger nTmp = 0;
            NSInteger k = 0;
            for (k = arrayNow.count; k < arrayNew.count;
k++) {
                nTmp += [[arrayNew
objectAtIndex:k]integerValue];
            }
            if (nTmp > 0) {
```

```
                isBigger = YES;
            }
        }
    }
    return isBigger;
}
```

Reveal使用

1. Build Settings 搜索Other
将Other Linker Flags设置为 -ObjC

2. Reveal.framework拖到项目中即可
~/资源库/Caches/
找到com.ittybittyapps.Reveal文件夹删除

~/资源库/Preferences/

找到com.ittybittyapps.Reveal.plist删除

又可以使用30天

UILabel设置多种字体、颜色

```
NSMutableAttributedString *str =
[[NSMutableAttributedString alloc] initWithString:@"Using
NSAttributed String, try your best to test attributed
string text"];

[str addAttribute:NSForegroundColorAttributeName value:
[UIColor blueColor] range:NSMakeRange(0,5)];

[str addAttribute:NSForegroundColorAttributeName value:
[UIColor redColor] range:NSMakeRange(6,12)];

[str addAttribute:NSForegroundColorAttributeName value:
[UIColor greenColor] range:NSMakeRange(19,6)];

[str addAttribute:NSFontAttributeName value:[UIFont
fontWithName:@"Arial" size:30.0] range:NSMakeRange(0,
5)];
```

```
[str addAttribute:NSFontAttributeName value:[UIFont
fontWithName:@"Arial" size:30.0] range:NSMakeRange(6,
12)];

[str addAttribute:NSFontAttributeName value:[UIFont
fontWithName:@"Arial" size:30.0] range:NSMakeRange(19,
6)];

UILabel *attrLabel = [[UILabel alloc]
initWithFrame:CGRectMake(20, 150, 320 - 40, 90)];

attrLabel.attributedText = str;

attrLabel.numberOfLines = 0;

[self.view addSubview:attrLabel];
```

iOS 调试方法（利用lldb）

po _image 可以看到_imge 的信息

模拟器安装app

xcrun simctl install booted +APP的路径

文本框键盘遮挡问题

```
UINavigationControllerDelegate
_niChenText.delegate=self;

[[NSNotificationCenterdefaultCenter]addObserver:selfselec
tor:@selector(keyboardWillShow:)name:UIKeyboardWillShowNo
tificationobject:nil];

[[NSNotificationCenterdefaultCenter]addObserver:selfselec
tor:@selector(keyboardWillHide:)name:UIKeyboardWillHideNo
tificationobject:nil];

@property (strong,nonatomic)UITextField *SelectTextField;
```

键盘显示事件

```
- (void) keyboardWillShow:(NSNotification *)notification
{



    获取键盘高度，在不同设备上，以及中英文下是不同的

    CGFloat kbHeight = [[notification.userInfo
objectForKey:UIKeyboardFrameEndUserInfoKey]
CGRectValue].size.height;



    计算出键盘顶端到inputTextView panel底端的距离(加上自定义的
缓冲距离INTERVAL_KEYBOARD)

    CGFloat offset = (_SelectTextField.frame.origin.y
+_SelectTextField.frame.size.height+50) -
(self.view.frame.size.height - kbHeight);



    取得键盘的动画时间，这样可以在视图上移的时候更连贯

    double duration = [[notification.userInfo
objectForKey:UIKeyboardAnimationDurationUserInfoKey]
doubleValue];



    将视图上移计算好的偏移

    if(offset > 0) {

        [UIView animateWithDuration:duration
animations:^{

            self.view.frame = CGRectMake(0.0f, -offset,
self.view.frame.size.width, self.view.frame.size.height);

        }];

    }
```

```
}


键盘消失事件
- (void) keyboardWillHide:(NSNotification *)notify {

    键盘动画时间
    double duration = [[notify.userInfo
objectForKey:UIKeyboardAnimationDurationUserInfoKey]
doubleValue];



    视图下沉恢复原状
    [UIView animateWithDuration:duration animations:^{

        self.view.frame = CGRectMake(0, 0,
self.view.frame.size.width, self.view.frame.size.height);

    }];

}

- (void)textFieldDidBeginEditing:(UITextField
*)textField{

    _SelectTextField=textField;

}


-(BOOL)textFieldShouldReturn:(UITextField *)textField

{

    [textField resignFirstResponder];

    return YES;

}


图片拉神
```

```
UIImage * resizeImage=[normal
resizableImageWithCapInsets:UIEdgeInsetsMake(h, w, h,
w)];
```

背景单击事件

```
@interface LCTianJianDiZhiController
()<HZAreaPickerDelegate,UIGestureRecognizerDelegate>

UITapGestureRecognizer *tapGestureRecognize =
[[UITapGestureRecognizer alloc] initWithTarget:self
action:@selector(backGroundOnclick)];

tapGestureRecognize.delegate = self;

[self.view  addGestureRecognizer:tapGestureRecognize];

-(void)backGroundOnclick

{

    [_selectDiQu setTitle:@"选择"
forState:UIControlStateNormal];

    [self.view endEditing:YES];

    [self cancelLocatePicker];



}
```

以下情况不能交互

```
alpha<0.01

hidden=yes

userInteracion=no
```

父视图不允许交换，子视图也不能

在父图可见范围内可以交换，范围之外不能交互

UiImageView 默认不允许用户交互

uitableviewcell 点击没有效果
```
cell.selectionStyle = UITableViewCellSelectionStyleNone;
```

隐藏导航栏
```
[self.navigationController setNavigationBarHidden:YES
animated:NO];
```

隐藏标签栏
```
self.tabBarController.tabBar.hidden =YES;
```

时间戳转换为时间
```
NSString
*str=[_mydata[indexPath.row]valueForKey:@"appr_comment_ti
me"];时间戳

NSTimeInterval time=[str doubleValue]/1000+28800;因为时差
问题要加8小时 == 28800 sec

NSDate *detaildate=[NSDate
dateWithTimeIntervalSince1970:time];

NSLog(@"date:%@",[detaildate description]);

实例化一个NSDateFormatter对象
NSDateFormatter *dateFormatter = [[NSDateFormatter alloc]
init];

设定时间格式,这里可以设置成自己需要的格式
[dateFormatter setDateFormat:@"yyyy-MM-dd HH:mm:ss"];

NSString *currentDateStr = [dateFormatter stringFromDate:
detaildate];

NSLog(@"%@",currentDateStr);
```

密码文本框secureTextEntry

把数据存储到本地

存储

```
获取userDefault单例
NSUserDefaults *userDefaults = [NSUserDefaults
standardUserDefaults];

登陆成功后把用户名和密码存储到UserDefault
[userDefaults setObject:_userName.text forKey:@"name"];

[userDefaults setObject:_password.text
forKey:@"password"];

[userDefaults synchronize];
```

调用

```
获取UserDefault
NSUserDefaults *userDefault = [NSUserDefaults
standardUserDefaults];

_userName.text= [userDefault objectForKey:@"name"];

_password.text= [userDefault objectForKey:@"password"];
```

页面切换的方式

从一个ViewController切换到另一个ViewController有下面几种方法:

（1）addsubview方法切换视图

```
self.view addSubview:(加载的新页面);
```

相应的 [self.view removeFromSuperview];移除添加的view

（2）self.view insertSubview:(加载的新页面) atIndex:n;

对n的解释：页面都是层次叠加的，n表示加载到那一层上面

（3）presentModalViewController方法

```
photoNacController.modalTransitionStyle =
UIModalTransitionStyleCrossDissolve;

photoNacController.modalPresentationStyle =
UIModalPresentationFullScreen;

self presentModalViewController:(加载的新页面) animated:
```

modalTransitionStyle用于设置页面切换的动画

modalPresentationStyle用于设置视图显示的方式

两种方法试试就知道用途了！

（4）pushViewController导航

```
[self.navigationController pushViewController:(加载的新页
面) animated:YES];
```

对应的

```
[self.navigationController
popViewControllerAnimated:YES];
```

总结：系统提供了我们视图切换的方法以及视图切换的默认动画，我们可以选择这几种方法中的去使用，也可以自定义切换的动画animation

```
/** 隐藏状态栏 */
- (BOOL)prefersStatusBarHidden

{

    return YES;

}
```

调整边距，可以让表格视图让开状态栏

```
self.tableView.contentInset = UIEdgeInsetsMake(20, 0, 0, 0);
```

代删除线的UILabel

```
- (void)drawRect:(CGRect)rect {

    Drawing code

    [super drawRect:rect];

    CGContextRef context=UIGraphicsGetCurrentContext();

    CGContextMoveToPoint(context, 0, 8);

    CGContextAddLineToPoint(context, rect.size.width, rect.size.height-5);

    CGContextStrokePath(context);

}
```

从xib加载时如果不显示或出错就可能是伸缩的问题

```
dropdown.autoresizingMask=UIViewAutoResizingNone;
```

unbutton 设置圆角边框

```
[huoQuYanZhenMa.layer setMasksToBounds:YES];

[huoQuYanZhenMa.layer setCornerRadius:10.0]; 设置矩形四个圆角半径

[huoQuYanZhenMa.layer setBorderWidth:1.0]; 边框宽度

CGColorSpaceRef colorSpace = CGColorSpaceCreateDeviceRGB();
```

```
CGColorRef colorref = CGColorCreate(colorSpace,
(CGFloat[]){ 1, 0, 0, 1 });

[huoQuYanZhenMa.layer setBorderColor:colorref];边框颜色

[self.view addSubview:huoQuYanZhenMa];



界面跳转
- (void)pushAction{

    PushViewController *pushVC = [[PushViewController
alloc] init];

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

    RootViewController *rootVC = (RootViewController
*)self.tabBarController;

    [rootVC showTabBar:NO];

    [self.navigationController
showViewController:<#(UIViewController *)#>
sender:<#(id)#>]

}

- (void)presentAction{

    ModalViewController *modalVC = [[ModalViewController
alloc] init];

    模态视图

    [self presentViewController:modalVC animated:YES
completion:nil];

}

- (void)dismissAction{
```

```
    [self dismissViewControllerAnimated:YES
completion:nil];

}

- (void)popAction{

    [self.navigationController
popViewControllerAnimated:YES];

}

UIStoryboard *stryBoard=[UIStoryboard
storyboardWithName:@"Main" bundle:nil];

[self presentViewController:[stryBoard
instantiateViewControllerWithIdentifier:@"mytabBarContol"
] animated:NO completion:nil];
```

语法约定
方法首字母大小，单词切换用大写
类名要大写
初始化应用要initW～～，w一定要大些，并且前面一定是init才行；不然self= [super init] 会出错

打印结构体

```
NSLog(@“%@”,nSStringFromRange(range));
```

代理的作用
监听哪些不能用addTarget监听的事件
主要用来负责两个对象之间的消息传递

代理实现的步骤
（1）成为（子）控件的代理，父亲（控制器）成为儿子（文本框）的代理
（2）遵守协议，利用智能提示具体实现

代理的id应该用weak 弱引用，不然会照成循环引用

判断是否实现某个协议方法

```
if ([self.delegate
respondsToSelector:@selector(tgFooterViewDidDowenLoadButt
on:)]) {

    [self.delegate tgFooterViewDidDowenLoadButton:self];

}
```

代理模式：是父控件（视图控制器）监听子控件的事件，当子控件发生某些事情时通知父控件工作

footerView->controller 去工作，用代理

conterView->footerView去工作，直接调用用footView的方法

不要分割线

```
tabQQChat.separatorStyle=UITableViewCellSeparatorStyleNon
e;
```

内边距

```
_textView.contentEdgeInsets=UIEdgeInsetsMake(20, 20, 20,
20);
```

获取文本的宽高

正文

```
CGFloat textX;

CGFloat textY=iconY;

CGSize textmaxSize= CGSizeMake(150, MAXFLOAT);

CGSize textRealSizw=[message.text
boundingRectWithSize:textmaxSize
options:NSStringDrawingUsesLineFragmentOrigin
attributes:@{NSFontAttributeName:[UIFont
systemFontOfSize:15.0f]} context:nil].size;
```

模型中数据中文乱码

对象描述方法，类似toStrong

```
-(NSString *)description

{

    return [NSString stringWithFormat:@"=<%@:
%p>{answer: %@,icon:%@,title:%@,option:
%@}",self.class,self ,self.answer,self.icon,self.title,se
lf.options];



}
```

然后在viewDidLoad中

```
NSLog(@"%@",self.question);
```

然后导入NSArray+Log.h

改变状态栏的颜色

```
-(UIStatusBarStyle)preferredStatusBarStyle

{

    return UIStatusBarStyleLightContent;

}
```

隐藏返回按钮文字

```
[[UIBarButtonItem appearance]
setBackButtonTitlePositionAdjustment:UIOffsetMake(0, -60)


forBarMetrics:UIBarMetricsDefault];
```

uitableVIew的背景图片

```
UIImageView *imageView = [[UIImageView alloc]
initWithImage:[UIImage imageNamed:@"leftMenu.jpg"]];
```

```
self.tableView.backgroundView = imageView;

分割线
self.tableView.separatorStyle=UITableViewCellSeparatorSty
leLSingleLine;

删除多余的分割线
self.tableView.tableFooterView = [[UIView alloc]
initWithFrame:CGRectZero];//tableView删除多余的分割线

- (void)setExtraCellLineHidden: (UITableView *)tableView

{


    UIView *view = [UIView new];


    view.backgroundColor = [UIColor clearColor];


    [tableView setTableFooterView:view];

}

Cell中的代理方法
/**

 初始化方法



 使用代码创建Cell的时候会被调用，如果使用XIB或者Storyboard，此方
法不会被调用

 */

- (id)initWithStyle:(UITableViewCellStyle)style
reuseIdentifier:(NSString *)reuseIdentifier
```

```
/**

 从XIB被加载之后，会自动被调用，如果使用纯代码，不会被执行

 */
- (void)awakeFromNib
```

Cell 被选中或者取消选中是都会被调用
如果是自定义Cell控件，所有的子控件都应该添加到contentView中

```
- (void)setSelected:(BOOL)selected animated:
(BOOL)animated
```

UIView的常用方法：
- （void）addSubview:(UIVIew *)view;添加一个子控件
- （void）removeFromSuperview; 从父控件中移除

- （UIView *）viewWithTag:(NSInteger)tag;根据tag表示寻找到对应的控件（一般是用于寻找子控件）

加载xib文件：

```
NSArray *array=[[NSBundle mainBundle]
loadNibName:@“HMAppView”owne:nil  options:nil];
UIView *view=[array firstObject];
```

计算给定文本字符串
UILabel要换行就要给设置行为0；

```
_textView.numberOfLines=0;
```

boundingRectWithSize计算给定文本字符串所占的区域
返回值是一个x,y = 0的CGRect,w,h是计算好的宽高

如果要计算多行的准确高度，需要传入
NSStringDrawingUsesLineFragmentOrigin选项

dict用于指定字体的相关属性的字典，UIKit框架中的第一个头文件

```
context: nil
NSDictionary *nameDict = @{NSFontAttributeName:
kNameFont};

CGRect nameFrame = [self.status.name
boundingRectWithSize:CGSizeMake(MAXFLOAT, MAXFLOAT)
options:NSStringDrawingUsesLineFragmentOrigin
attributes:nameDict context:nil];

nameFrame.origin.x = CGRectGetMaxX(self.iconView.frame) +
padding;

nameFrame.origin.y = padding +
(self.iconView.bounds.size.height -
nameFrame.size.height) * 0.5;

self.nameView.frame = nameFrame;
```

代码块存放路径：
/Users/a625/Library/Developer/Xcode/UserData/CodeSnippets

uitableview继承自uiscrollview
需要知道共有多少行，每一行有多高才能计算出UIscrollview的高度
知道每一行的高度，就可以计算出每一个屏幕显示多数航，才计算出表格明细
方法的执行次数
tableview .rowhight 的效率比代理更高，如果行高一样就用属性，不一样用代理

修改控件大小：

```
- (IBAction) top:(UIButton *)sender{

    CGRect btnFrame=self.HeadBtn.Fram;

    btnFrame.origin-=10;
```

```
    self.headBtn.fram=btnFrame;
}
```

下面代码错误的，oc规定不允许直接修改对象的结构体属性的成员

```
self。headBtn.frmae.origin.y-=10;
```

代码创建按钮：

1.创建一个自定义的按钮

```
UIButton *btn=[UIButton buttonWithTypeCustom];
```

2.添加按钮

```
[self.view addSubview:btn];
```

3.设置按钮的位置和尺寸

```
btn.frame=CGRectMake(100,100,100,100);
```

4.监听按钮的点击事件（点击按钮后就会调用self的btnClick方法）

```
[btn addTarget:self action:@selector(btnClick)
forControlEvents:UIControlEventTouchUpInside];
```

5.设置按钮在在默认状态下的属性

5.1默认状态下的背景

```
[btn setBackgroundImage:[UIImage imageName@“btn_01”]
forState:UIControlStateNormal];
```

5.2.设置默认状态下的文字：千万不要用

```
btn.titleLabel.text=@“sdsd”;
```

```
[btn setTitle:@“点我啊”forState:UIControlStateNormal];
```

5.3 默认状态的文字颜色

```
[btn setTitleColor:[UIColor
redColor]forState:UIControlStateNormal];
```

6.设置按钮在高亮状态下的属性

6.1 高亮状态的背景

```
[btn setBackgroupImage:[UIImage imageName:@“btn_01”]
forState:UIControlStateHighlighted];
```

6.2 高亮状态下的文字颜色

```
[btn setTitle:@“摸我干
啥”forState:UIControlStateHighlighted];
```

6.3 高亮状态下文字颜色

```
[btn setTitleColor:[UIColor blueColor]
forState:UIControlStateHightlighted];
```

修改按键的字体(titleLabel 是只读的 readonly 表示不允许修改titleLabel的指针)

```
btn.titleLabel.font=[UIFont sysTemFontOfsize 13];
```

sender.currentTitle 取出当前按钮的标题文字

```
CGRectGetMaxY (lable.frame) ;
```

frame属性，一般不要修改，通常用于实例化控件，指定初始位置
如果需要改变控件大小，使用bounds
如果需要改变控件位置，使用center

@property
1.生成getter () 方法
2.生成setter () 方法
3生成带下划线的成员变量（纪录属性内容）

readonly的属性不会生成带下划线的成员变量
@synthesize 可以合成出来 @synthesize image=_image;

代理的相关
1.遵守相关的协议，预先定义好方法，具体的实现工作有代理负责
<控件名称+DataSource> 定义的数据有关的方法
<控件名称+Delegate> 定义的与事件有关的方法，通常用来监听控件事件的

2.代理方法
1> 方面名以控件名称开口（没有前缀） ->方便程序员编写的时候快速找到需要的方法
2> 第一个参数是自己 ->意味着在协议方法中可以直接访问对象的属性，或者调用方法
3>代理方法的返回值 ->控制器向控件（委托）发送数据

内存管理:
控件:
如果是托线，用Weak
如果是代码，用Strong
NString 用copy
数字型的int 使用Assig

图片:
JPG:压缩比较高，通常用于照片，网页，有损压缩，解压缩时 对cpu消耗大，意味慢，费电
PNG: 压缩比高，无损压缩，

UIScrollView

```
self.scrollView.contentInset=UIEdgeInsetsMake(20, 20, 20, 20);
self.scrollView.showsHorizontalScrollIndicator=NO;
self.scrollView.showsVerticalScrollIndicator=NO;
```

偏移位置
```
self.scrollView.contentOffset=CGPointMake(100, 100);
self.scrollView.bounces=NO;取消弹簧效果
```

contentSize　　会根据边距调整offset
contentInset　　不会调整offset

动画的两种方式:
1.头尾式
```
[UIView beginAnimations:nil context:nil];

/**需要执行的动画**/
[UIViewcommitAnimations];
```

2.Block式
```
[UIView animateWithDuration:0.5 animations:^{

    /**需要执行动画的代码**/

}];
```

修改控件的位置和尺寸:
位置
frame.origin 原点
center　　中心

尺寸:
frame.size
bounds.sizes

查看是否调用该函数
```
NSLog(“%@”, __func__);
```

在get方法中
，如果跟自己相关的用下划线，不相干的用self

字典转模型
```
-(NSArray)appList

{
```

```
    if(appList==nil)

    {

        NSArray *array=[NSArray arrayWithContentsOfFile:
[[NSBundle mainBundle] PathForResouce:@“app.plist”
ofType:nil]];

        创建一个临时数组

        NSMutableArray *arrayM=[MSMutableArray array];

        for(NSDictionary *dict in array)

        {

            HMAppInfo *appInfo=[[HMAppInfo alloc]init];

            appInfo.name=dict[@“name”];

            appInfo.icon   =dict[@“icon”];

            [arrayM oddObject:appInfo];


        }

        将临时数组复制给属性

        _appList=arrayM;

    }

    return _appList;

}
```

使用时:

先实例化: `HMAppInfo *appInfo=self.appList[i];`

再使用: `icon.image=[UIImage imageName:appInfo.icon];`

UITextView 光标不再最开始位置

```
self.automaticallyAdjustsScrollViewInsets = NO;
```

设置标签栏属性

```
UITabBarItem*item =self.tabBarController.tabBar.items[0];

UIImage*imageNomal = [[UIImage
imageNamed:@"table_zhuye_off"]imageWithRenderingMode:UIIm
ageRenderingModeAlwaysOriginal];

item.image= imageNomal;

UIImage*imageSelected = [[UIImage
imageNamed:@"table_zhuye_on"]imageWithRenderingMode:UIIma
geRenderingModeAlwaysOriginal];

item.selectedImage= imageSelected;

self.tabBarController.tabBar.barTintColor = [UIColor
whiteColor];

self.tabBarController.tabBar.translucent = false;  关闭透
明

self.tabBarController.tabBar.tintColor = [UIColor
colorWithRed:0.359 green:0.902 blue:0.296 alpha:1.000];
```

退出键盘

方法一： [self.textfield resigFirstResponder] ;

方法二：[self.view endEditing:YES];Yes是否性关闭键盘

transform修改控件的位移（位置），缩放，旋转

创建一个transform属性相对初始

```
CGAffineTransForm
CGAffineTransformMakeTransLation(CGFloat tx,CGFloat ty);
```

左右移动

```
CGAffineTransForm CGAffineTransformMakeScale(CGFloat
sx,CGFloat sy);
```

放大缩小

```
CGAffineTransForm CGAffineTransformMakeRotaton(CGFloat
angle);
```

angle是弧度制，，排M_PI_4

在某个transform的基础上进行叠加

```
CGAffienTransForm
CGAffienTransformTransSlate(CGAffienTransForm t,CGFloat
tx.CGFloat ty);
CGAffienTransForm
CGAffienTransFormScale(CGAffienTransform t , CGFloat
sx,CGFloat sy);
CGAffienTransForm
CgAffeinTransformRotate(CGAffienTransform t,CGFloat
angle);
```

清空之前设置的transForm属性

```
view.transfrom=CGAffineTransformIdentity;
```

例：

```
self.butotn.transform=CGAffineTransForMakeTranslation(0,-
100);向上平移100;
NSLog(@“%@”,NSStringFromCGAffineTRansfor(self.button.tran
sform));
self.button.transform=CGAffineTransFormRotate(self.button
.transform,-M_PI_4);//逆时针旋转45度
```

UIImage加载图片：

一个UIImage对象代表了一张图片，一般通过Image：@”图片名“加载图片（png格式的图片可以省略拓展名）

```
UIImage *image= [UI image imageName：@“btn_01”] ;
```

imageNamed :图像是实例化之后由系统 负责，

```
String *path=[[NSBundle mainBundle]
pathForResource:imageName];
```

可以使用：

```
UIImae *image= [UIImage
imageWithContentsOfFile:path] ; //但是不能放在
images.xcassets
```

解析Plist文件

1.获取Plist文件路径

```
NSBundle *bundle=[NSBundle MianBundle];
NSString *path=[bundle pathForResource:@“imageData”
ofType:@“plist”];
```

2.懒加载Plist文件

```
_image=[NSArray arrayWithContentOfFile:path];

-(NSArray *)image

{

    if(_image==nil){

        NSBundle *bundle=[NSBundle MianBundle];

        NSString *path=[bundle
pathForResource:@“imageData” ofType:@“plist”];

        _image=[NSArray arrayWithContentOfFile:path];

    }

    return _image;

}
```

get（）懒加载

在get（）方法中不要在调用get（）方法了，self.btn是属于get（）方法,_btn 是变量名，不属于get（）方法

在懒加载中属于get（）方法，所有懒加载中不能出现self.btn~ 不然会出现死循环

时钟

scheduledTimerWithTimeInterval 方法本质上就是创建一个时钟，添加到运行循环的模式是DefaultRunLoopMode

```
self.timer = [NSTimer scheduledTimerWithTimeInterval:1.0
target:self selector:@selector(updateTimer:)
userInfo:@"hello timer" repeats:YES];
```

与1等价

```
self.timer = [NSTimer timerWithTimeInterval:1.0
target:self selector:@selector(updateTimer:) userInfo:nil
repeats:YES];
```

将timer添加到运行循环
模式：默认的运行循环模式

```
[[NSRunLoop currentRunLoop] addTimer:self.timer
forMode:NSDefaultRunLoopMode];
```

```
self.timer = [NSTimer timerWithTimeInterval:1.0
target:self selector:@selector(updateTimer:) userInfo:nil
repeats:YES];
```

将timer添加到运行循环
模式：NSRunLoopCommonModes的运行循环模式（监听滚动模式）

```
[[NSRunLoop currentRunLoop] addTimer:self.timer
forMode:NSRunLoopCommonModes];
```

停止时钟

```
[self.timer invalidate];
```

枚举
枚举类型本质上是整数，定义的时候，如果指定了第一个整数值，然后后面的就会递增
枚举时解决魔法数字的很好工具

```
typedef enum {
KmovingDirTop=10;
KmovingDirBottom;
KmovingDirLeft;
KmovingDirRigth
} KmoingDir;
```

通知中心传值：

发送通知

```
[[NSNotificationCenter
defaultCenter]postNotificationName:@"categoryDidChanged"
object:nil userInfo:@{@"categoryModel":_seletedModel}];
```

接受：

```
[[NSNotificationCenter defaultCenter]addObserver:self
selector:@selector(categoryChange:)
name:@"categoryDidChanged" object:nil];

- (void)categoryChange:(NSNotification*)noti{

    CategoriyModel *md =
(CategoriyModel*)noti.userInfo[@"categoryModel"];

    NSString *str = noti.userInfo[@"subCategoryName"];

    NSLog(@"左表：%@",md.name);

    NSLog(@"右表：%@",str);



}

- (void)dealloc{

    [[NSNotificationCenter
defaultCenter]removeObserver:self];

}
```

UITableviewCell 更新

一个section刷新

```
NSIndexSet *indexSet=[[NSIndexSet alloc]initWithIndex:2];

[tableview reloadSections:indexSet
withRowAnimation:UITableViewRowAnimationAutomatic];
```

一个cell刷新

```
NSIndexPath *indexPath=[NSIndexPath indexPathForRow:3
inSection:0];

[tableView reloadRowsAtIndexPaths:[NSArray
arrayWithObjects:indexPath,nil]
withRowAnimation:UITableViewRowAnimationNone];
```

获取屏幕高度

```
#define  kScreenWidth   [UIScreen
mainScreen].bounds.size.width
#define kScreenHeight   [UIScreen
mainScreen].bounds.size.height
```

生成随机数

arc4random_uniform(10) => 0~9之间的随机数

ARC 编译:

-fno-objc-arc

AFNetworking 的使用:

1.创建

```
AFHTTPRequestOperationManager *manager =
[AFHTTPRequestOperationManager manager];
```

2.指定解析器

manager 默认情况下是解析 json plist

xml 手动设置

```
manager.responseSerializer
manager.requestSerializer
aplication/json  text/json
```

3.get/post 请求

接口地址 urlString

请求参数

网络请求成功后的回调 blocks

```
[manager GET:<#(NSString *)#> parameters:<#(id)#>
success:^(AFHTTPRequestOperation *operation, id
responseObject) {

    成功后的操作

    operation.resposeData

} failure:^(AFHTTPRequestOperation *operation, NSError
*error) {
```

```
    请求失败后的操作
    NSLog(@"%@",error);

}];

post
[manager POST:url parameters:nil
success:^(AFHTTPRequestOperation *operation, id
responseObject) {

    发送后的回调

} failure:^(AFHTTPRequestOperation *operation, NSError
*error) {

    发送失败后的回调

}];

1. 网络下载数据然后把数据转为图片 再加载
AFHTTPRequestOperationManager *manager =
[AFHTTPRequestOperationManager manager];
manager.responseSerializer.acceptableContentTypes =
[NSSet setWithObject:@"image/jpeg"];
[manager GET:imageURL parameters:nil
success:^(AFHTTPRequestOperation *operation, id
responseObject) {

    NSLog(@"下载图片成功! ");

    iv.image = [UIImage
imageWithData:operation.responseData];

} failure:^(AFHTTPRequestOperation *operation, NSError
*error) {

    NSLog(@"%@",error);

}];


-(void)download{
```

```
    &tag=1001011&telephone=18398850943&password=111111

    AFHTTPRequestOperationManager *manager =
[AFHTTPRequestOperationManager manager];

    manager.responseSerializer =
[AFHTTPResponseSerializer serializer];

    manager.requestSerializer = [AFHTTPRequestSerializer
serializer];

    [manager GET:@"http://120.25.160.35:8080/
LocalGoodBrand/UserServlet" parameters:

     @{@"tag"           : @"100111"}

         success:^(AFHTTPRequestOperation *operation, id
responseObject) {

             NSString *string=[[NSString alloc]
initWithData:responseObject
encoding:NSUTF8StringEncoding];

             NSData* jsonData=[string
dataUsingEncoding:NSUTF8StringEncoding];

             _dataArray=   [NSJSONSerialization

                            JSONObjectWithData:jsonData


options:NSJSONReadingMutableContainers

                           error:nil];


         } failure:^(AFHTTPRequestOperation *operation,
NSError *error) {

             NSLog(@"Error: %@", error);

         }];
```

```
}
```

SDWebImage的使用：

```
[iv sd_setImageWithURL:[NSURL URLWithString:imageURL]];

UITapGestureRecognizer *tapGestureRecognize =
[[UITapGestureRecognizer alloc] initWithTarget:self
action:@selector(backGroundOnclick)];

tapGestureRecognize.delegate = self;
[self.view  addGestureRecognizer:tapGestureRecognize];
```

删除storyboard：

1.删除storyboard

2.general 中删除“main”

2.info 中删除 storyboard选项

3.屏幕初始化：

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary
*)launchOptions {


    self.window = [[UIWindow alloc]initWithFrame:
[UIScreen mainScreen].bounds];

    self.window.backgroundColor = [UIColor whiteColor];



    创建导航栏 需要传入一个UICollectionViewController

    FirstViewController *fvc = [[FirstViewController
alloc]init];

    MyNavController *nav = [[MyNavController
alloc]initWithRootViewController:fvc];

    self.window.rootViewController = nav;//

    [self.window makeKeyAndVisible];
```

```
    return YES;

}
```

按钮监听事件：

```
- (void)addtarget:(id)target action:(SEL)action{

    [self.button addTarget:target action:action
forControlEvents:UIControlEventTouchUpInside];

}

[first addtarget:self action:@selector(firstClick)];

#pragma mark - 点击事件
- (void)firstClick{

    [self createPopver];

}
```

获取plist文件数据

```
获取plist文件地址
NSString *file = [[NSBundle
mainBundle]pathForResource:@"categories.plist"
ofType:nil];

加载plist为数组
NSArray *plistArray = [NSArray
arrayWithContentsOfFile:file];
```

关闭popview 的自动适应屏幕属性：

```
pop.autoresizingMask = UIViewAutoresizingNone;
```

屏幕切换动态效果：

```
nav.modalPresentationStyle =
UIModalPresentationFormSheet;
```

隐藏导航栏；

```
[self.navigationController setNavigationBarHidden:YES
animated:YES];
```

设置tabbar的初始位置：

```
self.selectedIndex=button.1;
```

判断网络连接：

```
OOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary
*)launchOptions {

    开启网络指示器
    [[AFNetworkActivityIndicatorManager sharedManager]
setEnabled:YES];

    基准网站
    NSURL *url = [NSURL URLWithString:@"http://
baidu.com"];

    监听结果回调
    AFHTTPRequestOperationManager *manager =
[[AFHTTPRequestOperationManager alloc]
initWithBaseURL:url];

    NSOperationQueue *operationQueue       =
manager.operationQueue;

    [manager.reachabilityManager
setReachabilityStatusChangeBlock:^(AFNetworkReachabilityS
tatus status) {

        switch (status) {

            case
AFNetworkReachabilityStatusReachableViaWWAN:

            case
AFNetworkReachabilityStatusReachableViaWiFi:



                [operationQueue setSuspended:NO];
```

```
                NSLog(@"有网络");

                break;

            case AFNetworkReachabilityStatusNotReachable:

            default:

                [operationQueue setSuspended:YES];

                NSLog(@"无网络");

                break;

        }

    }];

    开始监听
    [manager.reachabilityManager startMonitoring];
```

UILabel 自动换行和自适应

要想UI Label自动换行，line设置为0

1.N行完全自适应：

```
    UILabel *testLabel = [[UILabel alloc]
initWithFrame:CGRectMake(10, 30, 100, 21)];

    NSString *txt =
@"dffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffff";

    testLabel.numberOfLines = 0; 相当于不限制行数
    testLabel.text = txt;
```

这样不行，还需要调用 [testLabel sizeToFit];

2.限制在N行内自适应：

```
    UILabel *testLabel = [[UILabel alloc]
initWithFrame:CGRectMake(10, 30, 100, 21)];

    NSString *txt =
@"dffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffff";

    testLabel.numberOfLines = 3;  限制在3行内自适应

    testLabel.text = txt;

    [testLabel sizeToFit];
```

结果不起作用，全部在一行显示了。

3.为了实现2的需求，需要这么做：

```
    CGSize maxSize = CGSizeMake(100, 21*3);

    UILabel *testLabel = [[UILabel alloc]
initWithFrame:CGRectMake(10, 30, 100, 21)];

    NSString *txt =
@"dffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffff";

    CGSize labelSize = [txt sizeWithFont:testLabel.font
constrainedToSize:maxSize lineBreakMode:
UILineBreakModeTailTruncation];


    testLabel.frame =
CGRectMake(testLabel.frame.origin.x,
testLabel.frame.origin.y, labelSize.width,
labelSize.height);

    testLabel.text = txt;
```

NSString:

创建一个字符串常量

```
    NSString *string1 = @"hello";
```

```
    string1 = @"hello world";

    NSLog(@"%@",string1);



    创建字符串
    NSString *string2 = [[NSString alloc]
initWithString:@"hello"];

    initWithFormat: 多个字符串拼接
    NSString *string3 = [[NSString alloc]
initWithFormat:@"hello %@",string2];

    NSLog(@"string2 = %@",string2);

    NSLog(@"string3 = %@",string3);

    NSString *ss1 = [[NSString alloc]
initWithFormat:@"ZHANGsan"];

    NSString *ss2 = [[NSString alloc]
initWithFormat:@"zhangsan"];

    NSLog(@"[ss1 caseInsensitiveCompare:ss2]:%ld",[ss1
caseInsensitiveCompare:ss2]);



    创建空的字符串
    NSString *string4 = [[NSString alloc] init]; //等价于
string4 = @"";

    stringWithFormat: 使用类方法创建字符串对象

    NSString *string5 = [NSString
stringWithString:@"hello"];  //等价于string5 = @"hello";
    NSString *string6 = [NSString
stringWithFormat:@"hello %@",@"world"];



    使用格式化符拼接数值
    int number = 101;
```

```
    NSString *string7 = [NSString
stringWithFormat:@"class:%d",number];

    NSLog(@"string7=%@",string7);



    字符串的比较
    NSString *s0 = @"Ediosn";

    NSString *s11 = @"Edison";   //s0与s11的指针地址是一样
的，指向的都是常量区同一个字符串对象


    NSString *s1 = [NSString stringWithFormat:@"Ediosn"];

    NSString *s2 = [[NSString alloc]
initWithFormat:@“Ediosn"];


    判断s1与s2的指针地址是否相等
    if (s0 == s2) {   不相等，因为是两个对象，指针地址不一样

        NSLog(@"s0 == s2");

    }


    isEqualToString: 是比较两个字符串内容是否相同

    if ([s0 isEqualToString:s2]) {

        NSLog(@"s0与s2的字符串内容相同");

    }

    NSObject *obj1;

    NSObject *obj2;


    NSString *string8 = @"a";
```

```
    NSString *string9 = @"A";


    compare: 比较字符串的大小

    NSComparisonResult result = [string8
compare:string9];

    if (result == NSOrderedAscending) { 结果为升序

        NSLog(@"string8 < string9");

    } else if(result == NSOrderedSame) {

        NSLog(@"string8 string9 内容一样");

    } else if(result == NSOrderedDescending) {

        NSLog(@"string8 > string9");

    }


    length: 获取字符串的长度
    NSString *string10 = @"abcdef";

    NSInteger len = [string10 length];

    NSLog(@"len = %ld",len);


    NSString *string11 = @"hELlo";

    uppercaseString:将字符串中的字母转成大写
    NSLog(@"upper:%@",[string11 uppercaseString]);

    uppercaseString
    NSLog(@"lower:%@",[string11 lowercaseString]);

    capitalizedString:首字母大写,其他字母小写
    NSLog(@"capitalized:%@",[string11
capitalizedString]);
```

```
    将字符串转成基本数据类型
    NSString *string12 = @"3.14";

    float f = (float)string12;  错误

    floatValue:字符串中是一个浮点数值转成float

    float f = [string12 floatValue];

    NSLog(@"floatValue:%f",f);

    NSString *string13 = @"1";

    BOOL bo = [string13 boolValue];  //true


    ---------------(4)字符串截取--------------------

    NSString *string14 = @"abcdef";

    ---------------(4)字符串截取--------------------

    NSString *substring1 = [string14 substringToIndex:3];

    NSLog(@"substringToIndex:%@",substring1);


    substringFromIndex:从指定索引位置开始截取到末尾，包含指定的索引f

    NSString *substring2 = [string14 substringFromIndex:1];

    NSLog(@"substringFromIndex:%@",substring2);


    NSRange rang = {2,3};  2:指定位置  3:需要截取的长度

    substringWithRange:截取指定范围的字符串
```

```
    NSString *substring3 = [string14
substringWithRange:rang];

    NSLog(@"substringWithRange:%@",substring3);



    NSArray *array = [string
componentsSeparatedByString:@"A"]; 从字符A中分隔成2个元素的数
组


    ---------------(5)拼接字符串--------------------

    NSString *str1 = @"Hello";

    NSString *str2 = @"World";

    NSString *str3 = @"OC!";

    NSString *string15 = [NSString stringWithFormat:@"%@-
%@-%@",str1,str2,str3];

    NSLog(@"string15:%@",string15);



    字符串追加
    NSString *string16 = [string15
stringByAppendingString:@"-iOS"];

    NSString *string17 = [string15
stringByAppendingFormat:@"%@,%@",@"iOS",@"iPhone"];

    NSLog(@"string16:%@",string16);

    NSLog(@"string17:%@",string17);



    字符串追加
    NSString *link = @"www.iphonetrain.com/.html";

    rangeOfString:查找字符串所在的位置
```

```
    NSRange linkRang = [link rangeOfString:@"html"];

    if (linkRang.location != NSNotFound) {

        NSLog(@"location:%ld,length:
%ld",linkRang.location,linkRang.length);

    }



    例如：能查找到@163.com，说明此邮箱是网易邮箱

    NSString *email = @"12345@163.com";



    /*_____________________________NSMutableString(不可变字
符串)_____________________________*/

    NSMutableString *mutableString1 = @"string";  错误
    NSMutableString *mutableString1 = [[NSMutableString
alloc] initWithFormat:@"字符串"];

    insertString: 在原有的字符串基础上插入字符串

    [mutableString1 insertString:@"可变" atIndex:0];

    NSLog(@"mutableString1:%@",mutableString1);

    NSMutableString *mutableString2 = [NSMutableString
stringWithFormat:@"字符符符串"];

    rangeOfString:查找指定字符串所在的范围

    NSRange rang3 = [mutableString2 rangeOfString:@"符
符"];

    deleteCharactersInRange:根据范围删除指定的字符串

    [mutableString2 deleteCharactersInRange:rang3];
```

```
    NSLog(@"mutableString2:%@",mutableString2);


    NSMutableString *mutableString3 = [NSMutableString
stringWithFormat:@"字符串"];

    NSRange rang4 = [mutableString3 rangeOfString:@"字
符"];

    replaceCharactersInRange:withString:  字符串替换

    [mutableString3 replaceCharactersInRange:rang4

                                  withString:@"羊肉"];

    NSLog(@"mutableString3:%@",mutableString3);


    NSArray:

    /*_____________________________不可变数组
(NSArray)_____________________________________*/
     ----------------------1.数组的创
建--------------------------

    NSString *s1 = @"zhangsan";

    NSString *s2 = @"lisi";

    NSString *s3 = @"wangwu";


    NSArray *array1 = [[NSArray alloc]
initWithObjects:s1,s2,s3, nil];

    NSLog(@"%@",array1);


    类方法创建，注意：最后以nil结尾。
```

```
    NSArray *array2 = [NSArray arrayWithObjects:s1,s2,s3,
nil];

    初始一个元素对象
    NSArray *array3 = [NSArray arrayWithObject:s1];



    创建一个数组，此数组中的元素来自array1
    NSArray *array4 = [NSArray arrayWithArray:array1];



    ----------------------2.通过下标取元
素--------------------------

    NSString *str1 = [array4 objectAtIndex:0];

    NSLog(@"str1 = %@",str1);



    ----------------------3.数组的元素个
数--------------------------

    NSUInteger count = [array4 count];

    NSUInteger count2 = array4.count;  点语法等价于->
[array4 count];注意count是不能带参数

    NSLog(@"count2 = %ld",count2);



    ----------------------4.判断是否包含某个对
象--------------------------

    BOOL isContains = [array4
containsObject:@"zhangsan"];

    NSLog(@"isContains: %d",isContains);
```

```
    ----------------------5.对象在数组中的索引位
置-------------------------

    NSUInteger index = [array4 indexOfObject:@"wangwu"];

    NSLog(@"index = %ld",index);



    ----------------------6.链接数组中的字符
串-------------------------

    注意：数组中的元素必须都是字符串，才可以使用此方法

    NSString *joinString = [array4
componentsJoinedByString:@","];

    NSLog(@"join:%@",joinString); //zhangsan,lisi,wangwu



    ----------------------7.访问最后一个元
素-------------------------

    NSString *last = [array4 lastObject]; //等价于点语法：
array4.lastObject;

    NSLog(@"last:%@",last);



    ----------------------8.在原来的数组上追加对
象-------------------------

    追加之后，创建了一个新的数组

    NSArray *array5 = [array4
arrayByAddingObject:@"zhaoliu"];

    NSLog(@"array5:%@",array5);
```

```
    /*
     注意：1.数组中不能存放基本数据类型，只能存放对象
     2.数组越界
     */
     NSArray *array6 = [NSArray arrayWithObject:100];  //错误，基本数据类型不能存放到数组中

    int idx = 4;
    if (idx < array5.count) {  严谨的写法，只有下标小于元素个数时，才可以使用下标取元素
        [array5 objectAtIndex:idx];
    }
    ---------------xcode4.4以后对数组的创建和访问，语法上做了优化---------------
    1.创建一个数组
    NSArray *array7 = @[s1,s2,s3];
    等价于NSArray *array2 = [NSArray arrayWithObjects:s1,s2,s3, nil];
    NSLog(@"array7=%@",array7);
    NSString *str = array7[0];
    NSLog(@"array7[0] = %@",str);

    /*_____________________________可变数组(NSMutableArray)_____________________________*
```

```
    NSString *t1 = @"zhangsan";

    NSString *t2 = @"lisi";

    NSString *t3 = @"wangwu";


    ----------------------1.创建可变数
组-------------------------

    NSMutableArray *marray1 = [[NSMutableArray alloc]
initWithObjects:t1,t2,t3, nil];


    创建数组时，开辟3个空间来存储元素，当存储的元素超过3个时，系统
会自动增大此数组的空间

    NSMutableArray *marray2 = [[NSMutableArray alloc]
initWithCapacity:3];

    NSMutableArray *marray3 = [NSMutableArray
arrayWithCapacity:3];



    新语法创建的是不可变数组
    NSMutableArray *marray4 = @[s1,s2,s3];  //错误

    ----------------------2.添加元
素-------------------------

    [marray2 addObject:s1];

    [marray2 addObject:s2];

    [marray2 addObject:s3];


    将marray2中所有的元素全都添加到marray3中

    [marray3 addObjectsFromArray:marray2];
```

```
    这是marray2添加到marray3中，marray3则是个二维数组

    [marray3 addObject:marray2];

    ----------------------3.插入元
素--------------------------

    [marray2 insertObject:@"赵六" atIndex:0];

    NSLog(@"marray2 = %@",marray2);

    [marray2 insertObject:@"zhaoliu" atIndex:0];  错误，数
组越界



    ----------------------4.替换元
素--------------------------

    [marray2 replaceObjectAtIndex:1
withObject:@"zhangfei"];

    NSLog(@"marray2 = %@",marray2);



    ----------------------5.互换两个元素的位
置--------------------------

    [marray2 exchangeObjectAtIndex:3 withObjectAtIndex:
2];

    NSLog(@"marray2 = %@",marray2);



    ----------------------6.将另外一个数组的所有元素添加到当前数
组--------------------------

    [marray3 addObjectsFromArray:marray2];
```

```
    ----------------------7.删除元
素--------------------------
    /*
     7.1根据下标删除
     [marray2 removeObjectAtIndex:0];
     NSLog(@"marray2 = %@",marray2);

     7.2根据对象删除
     [marray2 removeObject:@"zhangfei"];
     NSLog(@"marray2 = %@",marray2);

     7.3删除最后一个元素
     [marray2 removeLastObject];
     NSLog(@"marray2 = %@",marray2);

     7.4删除所有元素
     [marray2 removeAllObjects];
     NSLog(@"marray2 = %@",marray2);
     */

    ----------------------遍历数
组--------------------------
```

1.普通遍历

```
/*
 for (int i=0; i<marray2.count; i++) {
 NSString *str = [marray2 objectAtIndex:i];
 // NSString *str2 = marray2[i];
 NSLog(@"%@",str);
 }
 */
```

2.快速遍历

```
for (NSString *s in marray2) {
    NSLog(@"%@",s);
}
```

3.

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    块代码
    NSArray *array = @[@(1), @(2), @(3), @(4), @(5)];

    排序
```

```
        array = [array
sortedArrayUsingComparator:^NSComparisonResult(NSNumber
*num1, NSNumber *num2) {


            乱序=>一会升序，一会降序
            随机
            arc4random_uniform(10) => 0~9之间的随机数

            int seed = arc4random_uniform(2);

            if (seed) {

                return [num1 compare:num2];

            } else {

                return [num2 compare:num1];

            }

        }];


        NSLog(@"%@", array);

    }


    - (void)sortWith:(NSArray *)array

    {

        排序

        array = [array
sortedArrayUsingComparator:^NSComparisonResult(NSNumber
*num1, NSNumber *num2) {


            /**
```

```
            1 4 5 2

            4 1 5 2

            4 1 5 2

            5 4 1 2

            5 4 1 2

            5 4 2 1

            */

            NSLog(@"%@ %@", num1, num2);


            升序
            return [num1 compare:num2];
            降序
            return [num2 compare:num1];

    }];

    NSLog(@"%@", array);

}

- (void)arrayWith:(NSArray *)array

{

    int i = 0;

    for (NSNumber *num in array) {

        NSLog(@"%@", num);


        if (i == 1) {
```

```
                break;
            }
            i++;
        }

        参数：对象，索引，是否中断
        数组的块方法遍历的效率比for in高
        [array enumerateObjectsUsingBlock:^(NSNumber
*obj, NSUInteger idx, BOOL *stop) {
            NSLog(@"%@", obj);

            // idx == 1 退出循环
            if (idx == 1) {
                *stop = YES;
            }
        }];
    }

    NSDictonary：

    /*_____________________________不可变字典
(NSDictonary)_____________________________*/
```

```
    1.字典的创建
    NSArray *array1 = [NSArray
arrayWithObjects:@"zhangsan",@"zhangfei", nil];

    NSArray *array2 = [NSArray
arrayWithObjects:@"lisi",@"liping", nil];



    第一个元素：key:@“zhang”  value:array1

    第二个元素：key:@"li"     value:array2
    NSDictionary *dic1 = [[NSDictionary alloc]
initWithObjectsAndKeys:array1,@"zhang",array2,@"li",
nil];

    NSUInteger count = [dic1 count];

    NSLog(@"count:%ld",count);



    NSDictionary *dic2 = [NSDictionary
dictionaryWithObjectsAndKeys:array1,@"zhang",array2,@"lis
i", nil];



    创建字典时，初始化了一个元素
    NSDictionary *dic3 = [NSDictionary
dictionaryWithObject:array1 forKey:@"zhangsan"];



    2.获取字典中元素的个数
    NSUInteger count2 = [dic1 count];



    3.获取字典中所有的key
    NSArray *allkeys = [dic1 allKeys];

    NSLog(@"allkeys:%@",allkeys);
```

```
4.获取字典中所有的value
NSArray *allvalues = [dic1 allValues];

NSLog(@"allvalues:%@",allvalues);



5.通过key取得value
NSArray *array3 = [dic1 objectForKey:@"zhang"];

NSLog(@"array3:%@",array3);



/*_____________________________字典的新语法
_____________________________*/



创建的语法：{"key1":"value1","key2":"value2"};
新语法创建字典
NSDictionary *dic4 = @{@"zhang":array1,@"li":array2};

NSLog(@"dic4:%@",dic4);



取value的语法：字典["key"]
NSArray *array4 = dic4[@"zhang"];

NSLog(@"array4:%@",array4);

使用字典存储一个工人的信息

/*

 {

 "name": "zhangsan", 工人的姓名

 "age" : "22"         工人的年龄
```

```
    ...

    }

    */

    NSDictionary *worker = [NSDictionary
dictionaryWithObjectsAndKeys:@"zhangsan",@"name",@"23",@"
age", nil];




    /*_____________________________可变字典
(NSMutableDictionary)_____________________________*/


    1.创建一个可变的数组
    NSMutableDictionary *mdic1 = [[NSMutableDictionary
alloc] initWithCapacity:3];

    NSMutableDictionary *mdic2 = [NSMutableDictionary
dictionaryWithCapacity:3];



    2.添加元素
    [mdic1 setObject:array1 forKey:@"zhang"];

    [mdic1 setObject:array2 forKey:@"li"];



    将字典dic1中的元素添加到此字典中
    [mdic1 addEntriesFromDictionary:dic1];  注意：相同的key
是不能重复添加到字典中

    NSLog(@"mdic1:%@",mdic1);
```

3.删除
根据key删除元素

```
[mdic1 removeObjectForKey:@"zhang"];
```

删除多个元素

```
[mdic1 removeObjectsForKeys:@[@"zhang",@"li"]];

NSLog(@"mdic1:%@",mdic1);
```

删除所有的元素

```
[mdic1 removeAllObjects];
```

1.第一种方式

```
/*
 for (NSString *key in mdic1) {
 NSArray *names = [mdic1 objectForKey:key];
 NSLog(@"names:%@",names);
 }
 */
```

2.第二种方式

```
/*
 NSArray *keys = [mdic1 allKeys];
 for (int i=0; i<keys.count; i++) {
 NSString *key = [keys objectAtIndex:i];
 NSArray *names = [mdic1 objectForKey:key];
 NSLog(@"names:%@",names);
```

```
 }

 */
```

3.第三种方式

获取枚举对象，枚举对象中存储的是字典里所有的key

```
NSEnumerator *enumer = [mdic1 keyEnumerator];

让枚举对象的游标指向下一个对象
id key = [enumer nextObject];

while (key != nil) {

    NSArray *names = [mdic1 objectForKey:key];

    NSLog(@"names:%@",names);

    key = [enumer nextObject];

}

数组也可以使用枚举对象遍历

/*

 NSArray *array = [NSArray array];

 NSEnumerator *enumer2 = [array objectEnumerator];

 */

/*_____________________________字典排序
_____________________________*/

NSDictionary *sortDic = @{
```

```
                                @"zhangsan":@"50",

                                @"lisi":@"90",

                                @"wangwu":@"80",

                                @"zhao6":@"60"

                                };
```

对字典中的value进行排序，参数obj1,obj2是字典中的value
返回值是排好序的key

```
    NSArray *sortedKeys = [sortDic
keysSortedByValueUsingComparator:^NSComparisonResult(id
obj1, id obj2) {


        int v1 = [obj1 intValue];

        int v2 = [obj2 intValue];


        if (v1 > v2) {

            return NSOrderedDescending;

        } else if(v1 < v2) {

            return NSOrderedAscending;

        }


        return NSOrderedSame;


    }];
```

```
    for (NSString *name in sortedKeys) {
        NSString *score = [sortDic objectForKey:name];
        NSLog(@"name:%@,score:%@",name,score);
    }
```

NSSet

1.NSSet的创建

```
    NSString *s1 = @"zhangsan";
    NSString *s2 = @"lisi";
    NSSet *set1 = [[NSSet alloc] initWithObjects:s1,s2,
nil];
    NSSet *set2 = [NSSet setWithObjects:s1,s2, nil];
    把数组array中的所有元素，存储到set3中
    NSSet *set3 = [NSSet setWithArray:array];
```

2.NSSet转成数组

```
    NSArray *array1 = [set1 allObjects];
```

3.返回元素的个数

```
    NSUInteger count = [set1 count];
```

4.从容器中随机取出一个元素

```
    NSString *string1 = [set1 anyObject];
```

5.判断某一个对象是否在NSSet中

```
    BOOL isContains = [set1 containsObject:@"lisi"];


    6.NSSet中不能重复存同一个对象
    数组中是可以存取重复的对象

    NSString *str = @"jack";

    NSArray *array2 = [NSArray arrayWithObjects:str,str,
nil];

    NSLog(@"%@",array2);



    NSSet中不能重复存储相同的对象
    NSSet *set4 = [NSSet setWithObjects:str,str, nil];

    NSLog(@"%@",set4);



    /*

     NSSet与NSArray的区别

     1. 数组是有下标，NSSet是没有下标的

     2. 数组是有序的，NSSet是无序的

     3. 数组是可以重复存储同一个对象，NSSet反之，不能重复存储对象

     */


   NSMutableSet
   NSNumber:
    /*_____________________________________NSNumber的使用
_____________________________________*/


    1.创建NSNumber(包装基本数据类型)
```

```
    NSNumber *intNumber = [NSNumber numberWithInt:100];

    NSNumber *floatNumber = [NSNumber numberWithFloat:
9.8f];

    NSNumber *longNumber = [NSNumber numberWithLong:
145677766666];

    NSNumber *boolNumber = [NSNumber numberWithBool:YES];



    NSArray *array =
@[intNumber,floatNumber,longNumber,boolNumber];

    NSLog(@"array=%@",array);



    2.解包
    int intValue = [intNumber intValue];

    float floatValue = [floatNumber floatValue];

    long longValue = [longNumber longValue];

    BOOL boolValue = [boolNumber boolValue];



    3.新语法创建Number对象

    NSNumber *intNumber1 = @12;  //@"123";

    NSNumber *floatNumber1 = @12.0f;

    NSNumber *longValue1 = @19929292992;

    NSNumber *boolValue1 = @YES;

    NSNumber *charValue = @'a';
```

```
    /*_____________________________________NSValue的使用
_____________________________________*/
    struct WXPoint {

        float x;

        float y;

    };


    1.创建NSValue(包装结构体), NSValue是NSNumber的父类

    注意: 结构体不是对象

    NSRange rang = {100,6};

    NSRange封包

    NSValue *rangValue = [NSValue valueWithRange:rang];

    NSPoint封包

   NSValue *pointValue = [NSValue
valueWithPoint:<#(NSPoint)#>];
    将自定义的结构体包装成NSValue对象
    struct WXPoint p = {50,100};
    NSValue *pointValue = [NSValue value:&p
withObjCType:@encode(struct WXPoint)];


    2.解包结构体
    struct WXPoint p2;

    [pointValue getValue:&p2];

    NSLog(@"x=%f,y=%f",p2.x,p2.y);

    3.NSNull对象
    NSNull *n1 = [NSNull null];

    NSNull *n2 = [NSNull null];
```

```
    NSArray *arrayNull = @[n1,n2];
    NSLog(@"%@",arrayNull);

    for (id item in arrayNull) {
        判断数组中的对象是否为一个NSNull对象，如果是，则过滤掉
        if (item == [NSNull null]) {
            continue;
        }

    }
NSDate:
    /*_____________________________NSDate的使用
_____________________________*/

    1.创建日期
    NSDate *date1 = [NSDate date];  创建了一个当前的日期对象
    NSDate *date2 = [[NSDate alloc] init];
    NSLog(@"date2:%@",date2);

    在当前日期的基础上累加一个数值,单位是秒
    明天
    NSDate *date3 = [NSDate dateWithTimeIntervalSinceNow:
24*60*60];
    NSLog(@"date3:%@",date3);
```

```
    昨天
    NSDate *date4 = [NSDate
dateWithTimeIntervalSinceNow:-24*60*60];

    NSLog(@"date4:%@",date4);

    在1970年上加一个数值,该数值是一个时间戳数值
    NSDate *date1970 = [NSDate
dateWithTimeIntervalSince1970:0];

    NSLog(@"date1970:%@",date1970);


    2.获取日期的时间戳
    NSTimeInterval time1970 = [date1
timeIntervalSince1970];

    NSLog(@"time1970:%f",time1970);



    取得日期对象date3到当前日期时间的数值差
    NSTimeInterval timeNow = [date3
timeIntervalSinceNow];

    NSLog(@"timeNow:%f",timeNow);



    3.日期的比较
    (1)通过日期对象的compare方法进行比较

    NSComparisonResult result = [date3 compare:date1];

    if (result == NSOrderedDescending) {

        NSLog(@"date3 > date1");

    }

    (2)通过比较时间戳
```

```
    if ([date3 timeIntervalSince1970] > [date1
timeIntervalSince1970]) {

        NSLog(@"date3 > date1");

    }




    /*_____________________________NSDateFormatter格式化日期
_____________________________*/



    1.日期对象格式化为字符串：  2013-07-29 15:20:59  2013年07
月29日

     日期对象 --> 字符串
    NSDate *nowDate = [NSDate date];

    NSDateFormatter *dateFormatter = [[NSDateFormatter
alloc] init];

    设置日期的格式
    [dateFormatter setDateFormat:@"yyyy年MM月dd日
HH:mm:ss"];

    设置时区
    NSTimeZone *timezone = [NSTimeZone
timeZoneWithName:@"America/New_York"];

    [dateFormatter setTimeZone:timezone];

    stringFromDate：将日期对象格式化为字符串
    NSString *datestring = [dateFormatter
stringFromDate:nowDate];

    NSLog(@"格式化之后：%@",datestring);

    2.将字符串格式化成日期对象
```

```
    字符串 —>日期对象
    NSString *string = @"2013年07月29日 16:56:05";

    NSDateFormatter *dateFormatter2 = [[NSDateFormatter
alloc] init];

    [dateFormatter2 setDateFormat:@"yyyy年MM月dd日
HH:mm:ss"];

    dateFromString: 将字符串格式化成日期对象
    NSDate *formatDate = [dateFormatter2
dateFromString:string];

    NSLog(@"%@",formatDate);


    获取到所有时区的名称
    NSArray *zoneNames = [NSTimeZone knownTimeZoneNames];

    for (NSString *name in zoneNames) {

        NSLog(@"%@",name);

    }


    /*_____________________________捕捉异常
_____________________________*/

    创建一个空数组
    NSArray *arr = [NSArray array];



    @try {  有可能出异常的代码块

            数组越界异常
        [arr objectAtIndex:5];

    }

    @catch (NSException *exception) {
```

```
        如果捕捉到错误，则会执行此处的代码
        NSLog(@"错误：%@",exception);

    }

    @finally {  @finally是可选的

        不管有没有捕捉到异常，此处代码都会执行

        NSLog(@"@finally");

    }

    判断wearNeat方法是否在Student类中定义,如果定义了，才调用

    if ([stu respondsToSelector:@selector(wearNeat)]) {

        [stu wearNeat];

    }


    归档：数据存储
    /*_____________________________________第一种形式
_____________________________________*/

    /*****************对象归档**********************/

    对象------>文件

    /*

     NSArray *array = [NSArray
arrayWithObjects:@"zhangsan",@"lisi",@"中文", nil];

     归档保存的文件路径
     NSString *filePath = [NSHomeDirectory()
stringByAppendingPathComponent:@"array.arc"];
```

```
    归档对象
    BOOL success = [NSKeyedArchiver
archiveRootObject:array toFile:filePath];



    if (success) {

    NSLog(@"归档成功");

    }

    */




   /*****************解归档*********************/
   /*

    文件----->对象
    归档保存的文件路径
    NSString *filePath = [NSHomeDirectory()
stringByAppendingPathComponent:@"array.arc"];

    解归档
    NSArray *array = [NSKeyedUnarchiver
unarchiveObjectWithFile:filePath];

    for (NSString *s in array) {

    NSLog(@"%@",s);

    }

    */


   /*_____________________________________第二种形式
_____________________________________*/
```

```
    /*
     NSArray *array = [NSArray
arrayWithObjects:@"zhangsan",@"lisi",@"中文", nil];
     此NSMutableData用于存储归档对象中的数据
     NSMutableData *data = [NSMutableData data];

     创建归档对象
     NSKeyedArchiver *archiver = [[NSKeyedArchiver alloc]
initForWritingWithMutableData:data];

     编码数据和对象
     [archiver encodeObject:array forKey:@"array"];

     [archiver encodeInt:100 forKey:@"scope"];

     [archiver encodeObject:@"jack" forKey:@"name"];

     完成归档，将归档数据填充至data中，此时data中已经存储了归档对
象的数据

     [archiver finishEncoding];

     [archiver release];



     NSString *filePath = [NSHomeDirectory()
stringByAppendingPathComponent:@"ar.text"];

     将归档数据写入文件
     BOOL success = [data writeToFile:filePath
atomically:YES];

     if (success) {

     NSLog(@"arichiver success");

     }

     */
```

```
    NSString *filePath = [NSHomeDirectory()
stringByAppendingPathComponent:@"ar.text"];

    读取归档数据
    NSData *data = [[NSData alloc]
initWithContentsOfFile:filePath];

    创建解归档对象，对data中的数据进行解归档
    NSKeyedUnarchiver *unarchiver = [[NSKeyedUnarchiver
alloc] initForReadingWithData:data];

    解归档，还原数据
    NSArray *array = [unarchiver
decodeObjectForKey:@"array"];

    int scope = [unarchiver decodeIntForKey:@"scope"];

    NSString *name = [unarchiver
decodeObjectForKey:@"name"];

    NSLog(@"array=%@",array);

    NSLog(@"scope=%d",scope);

    NSLog(@"name=%@",name);

}
```

单例设计模式

```
获取单例对象的类方法
+ (AdressBook *)shareInstance {



    if (instacne == nil) {

        instacne = [[AdressBook alloc] init];

    }

    return instacne;
}
```

限制方法，限制这个类只能创建一个对象

```
+ (id)allocWithZone:(NSZone *)zone {
    if (instacne == nil) {
        instacne = [super allocWithZone:zone];
    }
    return instacne;
}
- (id)copyWithZone:(NSZone *)zone {
    return self;
}
- (id)retain {
    return instacne;
}
- (oneway void)release {
}
- (id)autorelease {
    return self;
}
- (NSUInteger)retainCount {
    return UINT_MAX;
}
```

沙盒路径：

1、Documents目录：您应该将所有de应用程序数据文件写入到这个目录下。这个目录用于存储用户数据或其它应该定期备份的信息。
2、AppName.app目录：这是应用程序的程序包目录，包含应用程序的本身。由于应用程序必须经过签名，所以您在运行时不能对这个目录中的内容进行修改，否则可能会使应用程序无法启动。
3、Library 目录：这个目录下有两个子目录：Caches 和 Preferences
Preferences 目录：包含应用程序的偏好设置文件。您不应该直接创建偏好设置文件，而是应该使用NSUserDefaults类来取得和设置应用程序的偏好.
Caches 目录：用于存放应用程序专用的支持文件，保存应用程序再次启动过程中需要的信息。
4、tmp目录：这个目录用于存放临时文件，保存应用程序再次启动过程中不需要的信息。

获取这些目录路径的方法：
1，获取家目录路径的函数：

```
NSString *homeDir = NSHomeDirectory();
```

2，获取Documents目录路径的方法：

```
NSArray *paths =
NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES);
NSString *docDir = [paths objectAtIndex:0];
```

3，获取Caches目录路径的方法：

```
NSArray *paths =
NSSearchPathForDirectoriesInDomains(NSCachesDirectory,
NSUserDomainMask, YES);
NSString *cachesDir = [paths objectAtIndex:0];
```

4，获取tmp目录路径的方法：

```
NSString *tmpDir = NSTemporaryDirectory();
```

5，获取应用程序程序包中资源文件路径的方法：
例如获取程序包中一个图片资源（apple.png）路径的方法：

```
NSString *imagePath = [[NSBundle mainBundle]
pathForResource:@”apple” ofType:@”png”];
UIImage *appleImage = [[UIImage alloc]
initWithContentsOfFile:imagePath];
```

做推送：
https://itunes.apple.com/tw/app/apn-tester-free/id626590577?l=zh&mt=12

文件路径处理：

```
演示路径
NSString *path = @"/Users/apple/file.text";
NSLog(@"演示路径：%@",path);

1.返回路径的组成部分
NSArray *array = [path pathComponents];
NSLog(@"pathComponents:%@",array);

2.路径的最后组成部分
NSString *lastComponent = [path lastPathComponent];
NSLog(@"lastComponent:%@",lastComponent);

3.追加子路径
NSString *newPath1 = [path stringByAppendingString:@"/appFile.text"];
NSLog(@"newPath1=%@",newPath1);

NSString *newPath2 = [path stringByAppendingPathComponent:@"appFile.text"];
NSLog(@"newPath2=%@",newPath2);

4.删除最后的组成部分
NSString *deleteLast = [path stringByDeletingLastPathComponent];
NSLog(@"deleteLast:%@",deleteLast);

5.删除扩展名
NSString *deleteExtension = [path stringByDeletingPathExtension];
NSLog(@"deleteExtension:%@",deleteExtension);

6.获取路径最后部分的扩展名
NSString *extension = [path pathExtension];
NSLog(@"extension:%@",extension);

7.追加扩展名
```

```
NSString *appendExt = [path
stringByAppendingPathExtension:@"jpg"];
NSLog(@"appendExt:%@",appendExt);

NSString ---> NSData
NSString *s = @"tsdfsdfsdfsdf";
NSData *data = [s
dataUsingEncoding:NSUTF8StringEncoding];

NSData ----> NSString
NSString *str = [[NSString alloc] initWithData:data
encoding:NSUTF8StringEncoding];
NSLog(@"str = %@",str);

NSMutableData 可变的Data对象，可以追加数据
文件操作：

NSString *homePath = NSHomeDirectory();

源文件路径
NSString *srcPath = [homePath
stringByAppendingPathComponent:@"06 第六课 文件管理.pdf"];
目标文件路径
NSString *targetPath = [homePath
stringByAppendingPathComponent:@"Documents/06第六课文件管
理.pdf"];

/*

 注意：使用NSFileHandle只能读写已经存在的文件，不能创建文件

 使用NSFileManager创建文件

 */

NSFileManager *fileManager = [NSFileManager
defaultManager];

创建目标文件
BOOL success = [fileManager createFileAtPath:targetPath
contents:nil attributes:nil];
```

```
if (success) {

    NSLog(@"目标文件创建成功!");

}

创建用于读取文件的NSFileHandle对象
NSFileHandle *readHandle = [NSFileHandle
fileHandleForReadingAtPath:srcPath];

创建用于写入的NSFileHandle对象
NSFileHandle *wrteHandle = [NSFileHandle
fileHandleForWritingAtPath:targetPath];

从当前偏移量读到文件的末尾，偏移量默认是起始位置
NSData *data = [readHandle readDataToEndOfFile];
NSData *data = [readHandle availableData];

将数据写入目标文件
[wrteHandle writeData:data];

关闭文件
[readHandle closeFile];
[wrteHandle closeFile];

/*_____________________________1.创建文件
_____________________________*/

/*

 获取当前app的沙盒根目录
 NSString *homePath = NSHomeDirectory();

 追加子路径
 NSString *filePath = [homePath
stringByAppendingPathComponent:@"Documents/file.text"];
 NSString *string = @"无限互联";

 将NSString转成NSData对象
```

```
 NSData *data = [string
dataUsingEncoding:NSUTF8StringEncoding];
 NSFileManager 不能使用alloc创建，这个类设计为单实例
 NSFileManager *fileM = [[NSFileManager alloc] init];
 NSFileManager 只能通过类方法defaultManager 创建
 NSFileManager *fileManager = [NSFileManager
defaultManager];


 根据路径filePath创建对应的文件，注意：只能创建文件，不能创建目录
(文件夹)
 BOOL success = [fileManager createFileAtPath:filePath
 contents:data
 attributes:nil];

 if (success) {

 NSLog(@"文件创建成功");

 } else {

 NSLog(@"文件创建失败");

 }


 创建文件夹
 NSString *filePath2 = [homePath
stringByAppendingPathComponent:@"Documents/demo"];

 NSError *error;

 BOOL suucces2 = [fileManager
createDirectoryAtPath:filePath2

 withIntermediateDirectories:YES

 attributes:nil

 error:&error];

 if (!suucces2) {
```

```
 NSLog(@"创建失败:%@",error);

 }

 */

/*_____________________________2.读取文件
_____________________________*/
获取当前app的沙盒根目录
NSString *homePath = NSHomeDirectory();

追加子路径
NSString *filePath = [homePath
stringByAppendingPathComponent:@"Documents/file.text"];

NSFileManager *fileManager = [NSFileManager
defaultManager];

根据路径读取文件中的数据
NSData *data = [fileManager contentsAtPath:filePath];

NSData 转 NSString
NSString *string = [[NSString alloc] initWithData:data
encoding:NSUTF8StringEncoding];
NSLog(@"%@",string);

/*_____________________________3.移动(剪切)文件
_____________________________*/
/*

 获取当前app的沙盒根目录
 NSString *homePath = NSHomeDirectory();

 源路径
 NSString *filePath = [homePath
stringByAppendingPathComponent:@"Documents/file.text"];


 目标路径
 NSString *tagetPath = [homePath
stringByAppendingPathComponent:@"Documents/demo/
file2.text"];
```

```
 NSFileManager *fileManager = [NSFileManager
defaultManager];


 moveItemAtPath: 移动文件
 BOOL success = [fileManager moveItemAtPath:filePath
toPath:tagetPath error:nil];

 if (!success) {

 NSLog(@"移动失败！！");

 }

 */

/*_____________________________4.复制文件
_____________________________*/
/*

 获取当前app的沙盒根目录
 NSString *homePath = NSHomeDirectory();

 源路径
 NSString *filePath = [homePath
stringByAppendingPathComponent:@"Documents/demo/
file3.text"];



 目标路径
 NSString *tagetPath = [homePath
stringByAppendingPathComponent:@"Documents/file.text"];
 NSFileManager *fileManager = [NSFileManager
defaultManager];


 copyItemAtPath: 将源文件复制到目标路径
 BOOL success = [fileManager copyItemAtPath:filePath
toPath:tagetPath error:nil];

 if (!success) {
```

```
 NSLog(@"复制失败！！");

 }

 */

/*_____________________________5.删除文件
_____________________________*/

/*

 获取当前app的沙盒根目录
 NSString *homePath = NSHomeDirectory();

 源路径
 NSString *filePath = [homePath
stringByAppendingPathComponent:@"Documents/demo/
file3.text"];
 NSFileManager *fileManager = [NSFileManager
defaultManager];

 判断文件是否存在
 BOOL fileExist = [fileManager
fileExistsAtPath:filePath];

 if (fileExist) {

 removeItemAtPath: 删除文件

 BOOL success = [fileManager removeItemAtPath:filePath
error:nil];

 if (success) {

 NSLog(@"删除成功!!");

 }

 }

 */
```

```
/*_____________________________6.获取文件的属性
_____________________________*/

/*

 NSFileManager *fileManager = [NSFileManager
defaultManager];

 NSString *homePath = NSHomeDirectory();

 目标路径
 NSString *filePath = [homePath
stringByAppendingPathComponent:@"Documents/file.text"];


 获取到文件的属性信息，文件的属性信息存储fileAttr字典中
 NSDictionary *fileAttr = [fileManager
attributesOfItemAtPath:filePath error:nil];
 NSLog(@"%@",fileAttr);

 从字典中通过key:NSFileSize获取到文件大小
 NSNumber *filesize = [fileAttr objectForKey:NSFileSize];
 long sizeValue = [filesize longValue];
 NSLog(@"文件大小:%ld",sizeValue);

 如下读取文件的大小，不可取，因为将文件中的数据全都读到内存中，文件
大时，太占用内存了
 NSData *data = [fileManager contentsAtPath:filePath];
 NSInteger len = data.length;

 return UIApplicationMain(argc, argv, nil,
NSStringFromClass([AppDelegate class]));

 */


UITableView的一些使用方法:
_tableView.dataSource = self;
_tableView.delegate = self;
_tableView = [[UITableView alloc]
initWithFrame:self.view.bounds
style:UITableViewStylePlain];
```

```
/**

 UITableViewStylePlain,      平板的格式

 UITableViewStyleGrouped     分组的格式

 */

设置行高
self.tableView.rowHeight = 120;

分隔线
self.tableView.separatorStyle =
UITableViewCellSeparatorStyleSingleLine;

headView，放在tableView最顶部的视图，通常用来放图片轮播器
UIView *head = [[UIView alloc]
initWithFrame:CGRectMake(0, 0, 320, 130)];
head.backgroundColor = [UIColor blueColor];
self.tableView.tableHeaderView = head;

footerView，通常做上拉刷新
UIView *foot = [[UIView alloc]
initWithFrame:CGRectMake(0, 0, 320, 44)];
foot.backgroundColor = [UIColor redColor];
self.tableView.tableFooterView = foot;
```