iOS bringSubviewToFront和exchangeSubviewAtIndex方法应用

```
[self.viewbringSubviewToFront:self.changeCityBtn];//将子视图在前面
/用第二层子视图换第一层子视图的位置,self.view是第0层
//[self.view exchangeSubviewAtIndex:2 withSubviewAtIndex:1];
//回到顶部
[self.tableView setContentOffset:CGPointZero];
```

```
*NSRange表示一个范围*/
void test1(){
   NSRange rg={3,5};//第一参数是起始位置第二个参数是长度
   //NSRange rg;
   //rg.location=3;
   //rg.length=5;
   //NSRange rg={.location=3,.length=5};
   //常用下面的方式定义 NSRange rg2=NSMakeRange(3,5);//使用NSMakeRange
定义一个NSRange
   //打印NSRange可以使用Foundation中方法 NSLog(@"rg2 is %@",
NSStringFromRange(rg2));//注意不能直接NSLog(@"rg2 is %@", rg2)，因为rg2不
是对象（准确的说%@是指针）而是结构体
}
/*NSPoint表示一个点*/
void test2(){
   NSPoint p=NSMakePoint(10, 15);//NSPoint其实就是CGPoint
   //这种方式比较常见 NSPoint p2=CGPointMake(10, 15);
   NSLog(NSStringFromPoint(p2));
}
/*NSSize表示大小*/
void test3(){
   NSSize s=NSMakeSize(10, 15);//NSSize其实就是CGSize
   //这种方式比较常见 CGSize s2=CGSizeMake(10, 15);
   NSLog(NSStringFromSize(s2));
}
/*NSRect表示一个矩形*/
void test4(){
   NSRect r=NSMakeRect(10, 5, 100, 200);//NSRect其实就是CGRect
   //这种方式比较常见 NSRect r2=CGRectMake(10, 5, 100, 200);
   NSLog(NSStringFromRect(r2));
}
```

===============================日期==================

```
 NSDate *date1=[NSDate date];//获得当前日期
   NSLog(@"%@",date1); //结果：2014-07-16 07:25:28 +0000
   NSDate *date2=[NSDate dateWithTimeIntervalSinceNow:100];//在当前日期的
基础上加上100秒，注意在ObjC中多数时间单位都是秒
   NSLog(@"%@",date2); //结果：2014-07-16 07:27:08 +0000

   NSDate *date3=[NSDate distantFuture];//随机获取一个将来的日期
   NSLog(@"%@",date3); //结果：4001-01-01 00:00:00 +0000

   NSTimeInterval time=[date2 timeIntervalSinceDate:date1];//日期之差,返回单
位为秒
   NSLog(@"%f",time); //结果：100.008833

   NSDate *date5=[date1 earlierDate:date3];//返回比较早的日期
   NSLog(@"%@",date5); //结果：2014-07-16 07:25:28 +0000

   //日期格式化
   NSDateFormatter *formater1=[[NSDateFormatter alloc]init];
   formater1.dateFormat=@"yy-MM-dd HH:mm:ss";
   NSString *datestr1=[formater1 stringFromDate:date1];
   NSLog(@"%@",datestr1); //结果：14-07-16 15:25:28
   //字符串转化为日期
   NSDate *date6=[formater1 dateFromString:@"14-02-14 11:07:16"];
   NSLog(@"%@",date6); //结果：2014-02-14 03:07:16 +0000
```

==============================**不可变字符串**==================

```
NSLog(@"has prefix ab? %i",[@"abcdef" hasPrefix:@"ab"]);
   //结果：has prefix ab? 1
   NSLog(@"has suffix ab? %i",[@"abcdef" hasSuffix:@"ef"]);
   //结果：has suffix ab? 1
   NSRange range=[@"abcdefabcdef" rangeOfString:@"cde"];//注意如果遇到
cde则不再往后面搜索,如果从后面搜索或其他搜索方式可以设置第二个options参数
   if(range.location==NSNotFound){
      NSLog(@"not found.");
   }else{
      NSLog(@"range is %@",NSStringFromRange(range));
   }
   //结果：range is {2, 3}
}
//字符串分割
void test4(){
   NSLog(@"%@",[@"abcdef" substringFromIndex:3]);//从第三个索引开始（包括
第三个索引对应的字符）截取到最后一位
   //结果：def
   NSLog(@"%@",[@"abcdef" substringToIndex:3]);////从0开始截取到第三个索引
（不包括第三个索引对应的字符）
   //结果：abc
```

```
    NSLog(@"%@",[@"abcdef" substringWithRange:NSMakeRange(2, 3)]);
    //结果：cde
    NSString *str1=@"12.abcd.3a";
    NSArray *array1=[str1 componentsSeparatedByString:@"."];//字符串分割
    NSLog(@"%@",array1);
     /*结果：
      (
         12,
         abcd,
         3a
      )
      */

}
```

=================**扩展--文件操作**=====================

```
 //读取文件内容
    NSString *path=@"/Users/kenshincui/Desktop/test.txt";
    NSString *str1=[NSString stringWithContentsOfFile:path
encoding:NSUTF8StringEncoding error:nil];
    //注意上面也可以使用gb2312 gbk等,例如
kCFStringEncodingGB_18030_2000，但是需要用
CFStringConvertEncodingToNSStringEncoding转换
    NSLog(@"str1 is %@",str1);
    //结果：str1 is hello world,世界你好！




    //上面我们看到了读取文件，但并没有处理错误,当然在ObjC中可以@try @catch
@finnally但通常我们并不那么做
    //由于我们的test.txt中有中文，所以使用下面的编码读取会报错，下面的代码演
示了错误获取的过程
    NSError *error;
    NSString *str2=[NSString stringWithContentsOfFile:path
encoding:kCFStringEncodingGB_18030_2000 error:&error];//注意这句话中的
error变量是**error，就是指针的指针那就是指针的地址，由于error就是一个指针此
处也就是error的地址&error，具体原因见下面补充
    if(error){
        NSLog(@"read error ,the error is %@",error);
    }else{
        NSLog(@"read success,the file content is %@",str2);
    }
    //结果：read error ,the error is Error Domain=NSCocoaErrorDomain
Code=261 "The file couldn’t be opened using the specified text encoding."
UserInfo=0x100109620 {NSFilePath=/Users/kenshincui/Desktop/test.txt,
NSStringEncoding=1586}



    //读取文件内容还有一种方式就是利用URl，它除了可以读取本地文件还可以读取
网络文件
    //NSURL *url=[NSURL URLWithString:@"file:///Users/kenshincui/Desktop/
test.txt"];
    NSURL *url=[NSURL URLWithString:@"http://www.apple.com"];
    NSString *str3=[NSString stringWithContentsOfURL:url
encoding:NSUTF8StringEncoding error:nil];
    NSLog(@"str3 is %@",str3);
}
void test2(){
    //下面是文件写入
    NSString *path1=@"/Users/kenshincui/Desktop/test2.txt";
    NSError *error1;
    NSString *str11=@"hello world,世界你好！";
    [str11 writeToFile:path1 atomically:YES encoding:NSUTF8StringEncoding
error:&error1];//automically代表一次性写入，如果写到中间出错了最后就全部不写
入
    if(error1){
        NSLog(@"write fail,the error is %@",[error1 localizedDescription]);//调用
localizedDescription是只打印关键错误信息
    }else{
        NSLog(@"write success!");
    }
    //结果：write success!
}
//路径操作
void test3(){
    NSMutableArray *marray=[NSMutableArray array];//可变数组
    [marray addObject:@"Users"];
    [marray addObject:@"KenshinCui"];
    [marray addObject:@"Desktop"];

    NSString *path=[NSString pathWithComponents:marray];
    NSLog(@"%@",path);//字符串拼接成路径
    //结果：Users/KenshinCui/Desktop

    NSLog(@"%@",[path pathComponents]);//路径分割成数组
    /*结果：
     (
        Users,
        KenshinCui,
        Desktop
    )
    */

    NSLog(@"%i",[path isAbsolutePath]);//是否绝对路径（其实就是看字符串是否以“/”开头）
    //结果：0
    NSLog(@"%@",[path lastPathComponent]);//取得最后一个目录
    //结果：Desktop
    NSLog(@"%@",[path stringByDeletingLastPathComponent]);//删除最后一个目录，注意path本身是常量不会被修改,只是返回一个新字符串
    //结果：Users/KenshinCui
    NSLog(@"%@",[path stringByAppendingPathComponent:@"Documents"]);//路径拼接
    //结果：Users/KenshinCui/Desktop/Documents
}
 //扩展名操作
void test4(){
    NSString *path=@"Users/KenshinCui/Desktop/test.txt";
    NSLog(@"%@",[path pathExtension]);//取得扩展名，注意ObjC中扩展名不包括"."
    //结果：txt
    NSLog(@"%@",[path stringByDeletingPathExtension]);//删除扩展名，注意包含"."
    //结果：Users/KenshinCui/Desktop/test
    NSLog(@"%@",[@"Users/KenshinCui/Desktop/test" stringByAppendingPathExtension:@"mp3"]);//添加扩展名
    //结果：Users/KenshinCui/Desktop/test.mp3
}

=================数组=================
/数组排序
void test4(){
    //方法1,使用自带的比较器
    NSArray *array=[NSArray arrayWithObjects:@"3",@"1",@"2", nil];
    NSArray *array2= [array sortedArrayUsingSelector:@selector(compare:)];
    NSLog(@"%@",array2);
    /*结果：
     (
        1,
        2,
        3
     )
     */
    //方法2,自己定义比较器
    Person *person1=[Person personWithName:@"Kenshin"];
    Person *person2=[Person personWithName:@"Kaoru"];
    Person *person3=[Person personWithName:@"Rosa"];
    NSArray *array3=[NSArray arrayWithObjects:person1,person2,person3, nil];
    NSArray *array4=[array3 
sortedArrayUsingSelector:@selector(comparePerson:)];
    NSLog(@"%@",array4);
    /*结果：
     (
         "name=Kaoru",
         "name=Kenshin",
         "name=Rosa"
     )
     */


    //方法3使用代码块
    NSArray *array5=[array3 
sortedArrayUsingComparator:^NSComparisonResult(Person *obj1, Person 
*obj2) {
        return [obj2.name compare:obj1.name];//降序
    }];
    NSLog(@"%@",array5);
    /*结果：
     (
         "name=Rosa",
         "name=Kenshin",
         "name=Kaoru"
     )
     */


    //方法4 通过描述器定义排序规则
    Person *person4=[Person personWithName:@"Jack"];
    Person *person5=[Person personWithName:@"Jerry"];
    Person *person6=[Person personWithName:@"Tom"];
    Person *person7=[Person personWithName:@"Terry"];
    NSArray *array6=[NSArray 
arrayWithObjects:person4,person5,person6,person7, nil];
    //定义一个排序描述
    NSSortDescriptor *personName=[NSSortDescriptor 
sortDescriptorWithKey:@"name" ascending:YES];
    NSSortDescriptor *accountBalance=[NSSortDescriptor 
sortDescriptorWithKey:@"account.balance" ascending:YES];
    NSArray *des=[NSArray arrayWithObjects:personName,accountBalance,
nil];//先按照person的name排序再按照account的balance排序
    NSArray *array7=[array6 sortedArrayUsingDescriptors:des];
    NSLog(@"%@",array7);
    /*结果：
     (
        "name=Jack",
        "name=Jerry",
        "name=Terry",
        "name=Tom"
     )
     */
}

==============================
/*常用方法*/
    Person *person1=[Person personWithName:@"Kenshin"];
    NSLog(@"%i",[person1 isKindOfClass:[NSObject class]]); //判断一个对象是否
为某种类型（如果是父类也返回YES），结果：1
    NSLog(@"%i",[person1 isMemberOfClass:[NSObject class]]); //判断一个对象
是否是某个类的实例化对象，结果：0
    NSLog(@"%i",[person1 isMemberOfClass:[Person class]]); //结果：1
    NSLog(@"%i",[person1 conformsToProtocol:@protocol(NSCopying)]);//是否实
现了某个协议，结果：0
    NSLog(@"%i",[person1 respondsToSelector:@selector(showMessage:)]);//是
否存在某个方法，结果：1

    [person1 showMessage:@"Hello,world!"];//直接调用一个方法
    [person1 performSelector:@selector(showMessage:)
withObject:@"Hello,world!"];
    //动态调用一个方法，注意如果有参数那么参数类型只能为ObjC对象，并且最多
只能有两个参数


    /*反射*/
    //动态生成一个类
    NSString *className=@"Person";
    Class myClass=NSClassFromString(className);//根据类名生成类
    Person *person2=[[myClass alloc]init]; //实例化
    person2.name=@"Kaoru";
    NSLog(@"%@",person2);//结果：name=Kaoru

    //类转化为字符串
    NSLog(@"%@,
%@",NSStringFromClass(myClass),NSStringFromClass([Person class])); //结
果：Person,Person
```

```
    //调用方法
    NSString *methodName=@"showMessage:";
    SEL mySelector=NSSelectorFromString(methodName);
    Person *person3=[[myClass alloc]init];
    person3.name=@"Rosa";
    [person3 performSelector:mySelector withObject:@"Hello,world!"]; //结果：
My name is Rosa,the infomation is "Hello,world!".

    //方法转化为字符串
    NSLog(@"%@",NSStringFromSelector(mySelector)); //结果：showMessage:



/*目录操作*/
void test1(){
    //文件管理器是专门用于文件管理的类
    NSFileManager *manager=[NSFileManager defaultManager];

    //获得当前程序所在目录(当然可以改变)
    NSString *currentPath=[manager currentDirectoryPath];
    NSLog(@"current path is :%@",currentPath);
    //结果：/Users/kenshincui/Library/Developer/Xcode/DerivedData/
FoundationFramework-awxjohcpgsqcpsanqofqogwbqgbx/Build/Products/
Debug

    //创建目录
    NSString *myPath=@"/Users/kenshincui/Desktop/myDocument";
    BOOL result= [manager createDirectoryAtPath:myPath
withIntermediateDirectories:YES attributes:nil error:nil];
    if(result==NO){
        NSLog(@"Couldn't create directory!");
    }

    //目录重命名，如果需要删除目录只要调用removeItemAtPath:<#(NSString *)#>
error:<#(NSError **)#>
    NSError *error;
    NSString *newPath=@"/Users/kenshincui/Desktop/myNewDocument";
    if([manager moveItemAtPath:myPath toPath:newPath error:&error]==NO){
        NSLog(@"Rename directory failed!Error infomation is:%@",error);
    }

    //改变当前目录
    if([manager changeCurrentDirectoryPath:newPath]==NO){
        NSLog(@"Change current directory failed!");
    }
    NSLog(@"current path is :%@",[manager currentDirectoryPath]);
    //结果：current path is :/Users/kenshincui/Desktop/myNewDocument

    //遍历整个目录
    NSString *path;
    NSDirectoryEnumerator *directoryEnumerator= [manager
enumeratorAtPath:newPath];
    while (path=[directoryEnumerator nextObject]) {
        NSLog(@"%@",path);
    }
    /*结果：
     documents
     est.txt
    */

    //或者这样遍历
    NSArray *paths= [manager contentsOfDirectoryAtPath:newPath error:nil];
    NSObject *p;
    for (p in paths) {
        NSLog(@"%@",p);
    }
    /*结果：
     documents
     est.txt
     */
}

/*文件操作*/
void test2(){
    NSFileManager *manager=[NSFileManager defaultManager];
    NSString *filePath=@"/Users/kenshincui/Desktop/myNewDocument/test.txt";
    NSString *filePath2=@"/Users/kenshincui/Desktop/test.txt";
    NSString *newPath=@"/Users/kenshincui/Desktop/myNewDocument/
test2.txt";

    //判断文件是否存在，这个方法也可以判断目录是否存在，这要后面的参数设置位
YES
    if ([manager fileExistsAtPath:filePath isDirectory:NO]) {
        NSLog(@"File exists！");
    }

    //文件是否可读
    if([manager isReadableFileAtPath:filePath]){
        NSLog(@"File is readable!");
    }

    //判断两个文件内容是否相等
```

```
if ([manager contentsEqualAtPath:filePath andPath:filePath2]) {
    NSLog(@"file1 equals file2");
}

//文件重命名，方法类似于目录重命名
if (![manager moveItemAtPath:filePath toPath:newPath error:nil]) {
    NSLog(@"Rename file1 failed!");
}

//文件拷贝
NSString *filePath3=@"/Users/kenshincui/Desktop/test3.txt";
if(![manager copyItemAtPath:newPath toPath:filePath3 error:nil]){
    NSLog(@"Copy failed!");
}

//读取文件属性
NSDictionary *attributes;
if ((attributes=[manager attributesOfItemAtPath:newPath error:nil])==nil) {
    NSLog(@"Read attributes failed!");
}
for (NSObject *key in attributes) {
    NSLog(@"%@=%@",key,attributes[key]);
}
/*结果:
     NSFileOwnerAccountID=501
     NSFileHFSTypeCode=0
     NSFileSystemFileNumber=1781953
     NSFileExtensionHidden=0
     NSFileSystemNumber=16777218
     NSFileSize=27
     NSFileGroupOwnerAccountID=20
     NSFileOwnerAccountName=kenshincui
     NSFileCreationDate=2014-07-28 11:47:58 +0000
     NSFilePosixPermissions=420
     NSFileHFSCreatorCode=0
     NSFileType=NSFileTypeRegular
     NSFileExtendedAttributes={
     "com.apple.TextEncoding" = <7574662d 383b3133 34323137 393834>;
     }
     NSFileGroupOwnerAccountName=staff
     NSFileReferenceCount=1
     NSFileModificationDate=2014-07-28 11:47:58 +0000
 */

//删除文件
[manager removeItemAtPath:newPath error:nil];
```

```
}
//文件操作--文件内容操作（NSData，非结构化字节流对象，有缓冲区管理机制，
可用于网络传输）
void test3(){
    NSFileManager *manager=[NSFileManager defaultManager];
    NSString *filePath=@"/Users/kenshincui/Desktop/myNewDocument/
test2.txt";
    NSData *data=[manager contentsAtPath:filePath];
    NSLog(@"%@",data);//存储的是二进制字节流
    //结果:<68656c6c 6f20776f 726c642c e4b896e7 958ce4bd a0e5a5bd
efbc81>

    //NSData转化成字符串
    NSString *str1=[[NSString alloc]initWithData:data
encoding:NSUTF8StringEncoding];
    NSLog(@"%@",str1);
    //结果：hello world,世界你好！

    //字符串转化成NSData
    NSString *str2=@"Kenshin";
    NSData *data2=[str2 dataUsingEncoding:NSUTF8StringEncoding];
    NSLog(@"%@",data2);

    //当然一般如果仅仅是简单读取文件内容，直接用户NSString方法即可
    NSString *content=[NSString stringWithContentsOfFile:filePath
encoding:NSUTF8StringEncoding error:nil];
    NSLog(@"%@",content);
    //结果：hello world,世界你好！

}
//文件操作--细粒度控制文件,文件操作柄
void test4(){
    NSFileManager *manager=[NSFileManager defaultManager];
    NSString *filePath=@"/Users/kenshincui/Desktop/myNewDocument/
test2.txt";

    //以只读方式打开文件
    NSFileHandle *fileHandle=[NSFileHandle
fileHandleForReadingAtPath:filePath];//注意这个方法返回类型为instancetype，
也就是说对于上面的NSFileHandle它的返回类型也是NSFileHandle
    NSData *data= [fileHandle readDataToEndOfFile];//完整读取文件
    NSString *newPath=@"/Users/kenshincui/Desktop/test4.txt";
    [manager createFileAtPath:newPath contents:nil attributes:nil];
    NSFileHandle *fileHandle2=[NSFileHandle
fileHandleForWritingAtPath:newPath];//以可写方式打开文件
```

```
    [fileHandle2 writeData:data];//写入文件内容

    [fileHandle2 closeFile];//关闭文件


    //定位到指定位置,默认在文件开头
    [fileHandle seekToFileOffset:12];
    NSData *data2= [fileHandle readDataToEndOfFile];
    NSLog(@"data2=%@",[[NSString alloc]initWithData:data2
encoding:NSUTF8StringEncoding]);
    //结果：data2=世界你好！

    [fileHandle seekToFileOffset:6];
    NSData *data3=[fileHandle readDataOfLength:5];
    NSLog(@"data3=%@",[[NSString alloc]initWithData:data3
encoding:NSUTF8StringEncoding]);
    //结果：data3=world

    [fileHandle closeFile];

}

//文件路径
void test5(){
    NSString *filePath=@"/Users/kenshincui/Desktop/myDocument";
    NSString *filePath2=@"/Users/kenshincui/Desktop/test.txt";

    //临时文件所在目录
    NSString *path=NSTemporaryDirectory();
    NSLog(@"temporary directory is :%@",path);
    //结果：/var/folders/h6/lss6gncs509c2pgzgty3wd_40000gn/T/

    NSString *lastComponent= [filePath lastPathComponent];
    NSLog(@"%@",lastComponent); //结果：myDocument

    NSLog(@"%@",[filePath stringByDeletingLastPathComponent]);
    //结果：/Users/kenshincui/Desktop
    NSLog(@"%@",[filePath stringByAppendingPathComponent:@"Pictrues"]);
    //结果：/Users/kenshincui/Desktop/myDocument/Pictrues
    NSLog(@"%@",[filePath2 pathExtension]);
    //结果：txt

    [[filePath pathComponents] enumerateObjectsUsingBlock:^(id obj,
NSUInteger idx, BOOL *stop) {
        NSLog(@"%i=%@",idx,obj);
    }];
    /*结果：
     0=/
     1=Users
     2=kenshincui
     3=Desktop
     4=myDocument
     */


}

//文件操作--NSURL
void test6(){
    NSURL *url=[NSURL URLWithString:@"http://developer.apple.com"];
    NSString *str1=[NSString stringWithContentsOfURL:url
encoding:NSUTF8StringEncoding error:nil];
    NSLog(@"%@",str1);
}

//文件操作--NSBundle，程序包，一般用于获取Resource中的资源（当然由于当前
并非IOS应用没有程序包，只是表示当前程序运行路径）
//在ios中经常用于读取应用程序中的资源文件，如图片、声音、视频等
void test7(){
    //在程序包所在目录创建一个文件
    NSFileManager *manager=[NSFileManager defaultManager];
    NSString *currentPath=[manager currentDirectoryPath];
    NSLog(@"current path is :%@",currentPath);
    //结果：current path is :/Users/kenshincui/Library/Developer/Xcode/
DerivedData/FoundationFramework-awxjohcpgsqcpsanqofqogwbqgbx/Build/
Products/Debug
    NSString *filePath=[currentPath
stringByAppendingPathComponent:@"test.txt"];
    [manager createFileAtPath:filePath contents:[@"Hello,world!"
dataUsingEncoding:NSUTF8StringEncoding] attributes:nil];


    //利用NSBundle在程序包所在目录查找对应的文件
    NSBundle *bundle=[NSBundle mainBundle];//主要操作程序包所在目录
    //如果有test.txt则返回路径，否则返回nil
    NSString *path=[bundle pathForResource:@"test" ofType:@"txt"];//也可以写
成：[bundle pathForResource:@"instructions.txt" ofType:nil];
    NSLog(@"%@",path);
    //结果：/Users/kenshincui/Library/Developer/Xcode/DerivedData/
FoundationFramework-awxjohcpgsqcpsanqofqogwbqgbx/Build/Products/
Debug/test.txt
    NSLog(@"%@",[NSString stringWithContentsOfFile:path
encoding:NSUTF8StringEncoding error:nil]);
    //结果：Hello,world!

    //假设我们在程序运行创建一个Resources目录，并且其中新建pic.jpg，那么用下
面的方法获得这个文件完整路径
    NSString *path1= [bundle pathForResource:@"pic" ofType:@"jpg"
inDirectory:@"Resources"];
    NSLog(@"%@",path1);
    //结果：/Users/kenshincui/Library/Developer/Xcode/DerivedData/
FoundationFramework-awxjohcpgsqcpsanqofqogwbqgbx/Build/Products/
Debug/Resources/pic.jpg
}


#pragma mark - NSString字符串

// 判断字符串为空
+ (BOOL)isEmptyOrNull:(NSString *)string
{
    if (string == nil)
    {
        return YES;
    }
    if (string == NULL)
    {
        return YES;
    }
    if ([string isKindOfClass:[NSNull class]])
    {
        return YES;
    }
    if ([string isEqualToString:@""])
    {
        return YES;
    }
    if ([[string stringByTrimmingCharactersInSet:[NSCharacterSet
whitespaceAndNewlineCharacterSet]] length] == 0)
    {
        return YES;
    }
    return NO;
}

// 检查字符串是否是纯数字
+ (BOOL)checkStringIsOnlyDigital:(NSString *)str
{
  NSString *string = [str stringByTrimmingCharactersInSet:[NSCharacterSet
decimalDigitCharacterSet]];
    if(string.length >0)
    {
        return NO;
    }else return YES;
}

//检查字符串是否为nil 转为@""
+ (NSString *)checkStringValue:(id)str
{
    if ([str isKindOfClass:[NSNull class]]) {
        return @"";
    }
    return str;
}

//判断字符串中包含汉字
+ (BOOL)checkStringIsContainerChineseCharacter:(NSString *)string
{
    for (int i = 0; i < string.length; i++)
    {
        int a = [string characterAtIndex:i];
        if (a >= 0x4e00 && a <= 0x9fff) {
            return YES;
        }
    }
    return NO;
}

//过滤特殊字符串
+ (NSString *)filterSpecialString:(NSString *)string
{
    NSCharacterSet *dontWant = [NSCharacterSet
characterSetWithCharactersInString:@"[]{}（#%-*+=_）\\|~(＜＞$%^&*)_+,.;':|/
@!? "];
    //stringByTrimmingCharactersInSet只能去掉首尾的特殊字符串
    return [[[string componentsSeparatedByCharactersInSet:dontWant]
componentsJoinedByString:@""] stringByReplacingOccurrencesOfString:@"\n"
withString:@""];
}


#pragma mark - 字符串size

/**
```

```
 *  计算字符串尺寸
 *
 *  @param string
 *  @param font   字体
 *  @param size
 *
 *  @return
 */
+ (CGSize)sizeWithString:(NSString *)string font:(UIFont *)font size:
(CGSize)size
{
   NSDictionary *dic = @{NSFontAttributeName:font};
   return [string boundingRectWithSize:size options:
(NSStringDrawingUsesLineFragmentOrigin) attributes:dic context:nil].size;
}

#pragma mark - UIColor

+ (UIColor *)colorFromHexCode:(NSString *)hexString
{
   if (!hexString) {
      return [UIColor clearColor];
   }
   NSString *cleanString = [hexString
stringByReplacingOccurrencesOfString:@"#" withString:@""];
   if([cleanString length] == 3) {
      cleanString = [NSString stringWithFormat:@"%@%@%@%@%@%@",
                 [cleanString substringWithRange:NSMakeRange(0, 1)],
[cleanString substringWithRange:NSMakeRange(0, 1)],
                 [cleanString substringWithRange:NSMakeRange(1, 1)],
[cleanString substringWithRange:NSMakeRange(1, 1)],
                 [cleanString substringWithRange:NSMakeRange(2, 1)],
[cleanString substringWithRange:NSMakeRange(2, 1)]];
   }
   if([cleanString length] == 6) {
      cleanString = [cleanString stringByAppendingString:@"ff"];
   }

   unsigned int baseValue;
   [[NSScanner scannerWithString:cleanString] scanHexInt:&baseValue];

   float red = ((baseValue >> 24) & 0xFF)/255.0f;
   float green = ((baseValue >> 16) & 0xFF)/255.0f;
   float blue = ((baseValue >> 8) & 0xFF)/255.0f;
   float alpha = ((baseValue >> 0) & 0xFF)/255.0f;
```

```
    return [UIColor colorWithRed:red green:green blue:blue alpha:alpha];
}

//从图片转到颜色
+ (UIColor *)colorFromImage:(UIImage *)image
{
    if (image == nil)
    {
        return [UIColor clearColor];
    } else {
        return [UIColor colorWithPatternImage:image];
    }
}


#pragma mark - UIImage
//从颜色生成图片
+ (UIImage *)imageFromUIColor:(UIColor *)color {
    if (!color) {
        color = [UIColor clearColor];
    }
    CGRect rect = CGRectMake(0, 0, 1, 1);
    // create a 1 by 1 pixel context
    UIGraphicsBeginImageContextWithOptions(rect.size, NO, 0);
    [color setFill];
    UIRectFill(rect);
    UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    return image;
}

#pragma mark - 压缩图片
// 压缩图片按照大小
+ (UIImage *)image:(UIImage *)image scaleToSize:(CGSize)size
{
    CGImageRef imgRef = image.CGImage;
    CGSize originSize = CGSizeMake(CGImageGetWidth(imgRef),
CGImageGetHeight(imgRef)); // 原始大小
    if (CGSizeEqualToSize(originSize, size)) {
        return image;
    }
    UIGraphicsBeginImageContextWithOptions(size, NO, 0);            //[UIScreen
mainScreen].scale
    CGContextRef context = UIGraphicsGetCurrentContext();
    /**
     *  设置CGContext集插值质量
     *  kCGInterpolationHigh 插值质量高
     */
    CGContextSetInterpolationQuality(context, kCGInterpolationHigh);
    [image drawInRect:CGRectMake(0, 0, size.width, size.height)];
    UIImage *scaledImage = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    return scaledImage;
}

// 压缩图片按照比例
+ (UIImage *)image:(UIImage *)image scaleWithRatio:(CGFloat)ratio
{
    CGImageRef imgRef = image.CGImage;
    if (ratio > 1 || ratio <= 0) {
        return image;
    }
    CGSize size = CGSizeMake(CGImageGetWidth(imgRef) * ratio,
CGImageGetHeight(imgRef) * ratio); // 缩放后大小
    return [self image:image scaleToSize:size];
}

#pragma mark - 添加水印
+ (UIImage *)image:(UIImage *)img addLogo:(UIImage *)logo
{
    if (logo == nil ) {
        return img;
    }
    if (img == nil) {
        return nil;
    }
    //get image width and height
    int w = img.size.width;
    int h = img.size.height;
    int logoWidth = logo.size.width;
    int logoHeight = logo.size.height;

    CGColorSpaceRef colorSpace = CGColorSpaceCreateDeviceRGB();
    //create a graphic context with CGBitmapContextCreate
    CGContextRef context = CGBitmapContextCreate(NULL, w, h, 8, 44 * w,
colorSpace, kCGImageAlphaPremultipliedFirst);
    CGContextDrawImage(context, CGRectMake(0, 0, w, h), img.CGImage);
    CGContextDrawImage(context, CGRectMake(w-logoWidth-15, 10, logoWidth,
logoHeight), [logo CGImage]);
    CGImageRef imageMasked = CGBitmapContextCreateImage(context);
    UIImage *returnImage = [UIImage imageWithCGImage:imageMasked];
    CGContextRelease(context);
    CGImageRelease(imageMasked);
    CGColorSpaceRelease(colorSpace);
    return returnImage;
}

#pragma mark - NSUserDefault

// 取值
id UserDefaultGetObj(NSString *key)
{
    NSUserDefaults *ud = [NSUserDefaults standardUserDefaults];
    return [ud objectForKey:key];
}

// 存入
void UserDefaultSetObjForKey(id object,NSString *key)
{
    NSUserDefaults *ud = [NSUserDefaults standardUserDefaults];
    [ud setValue:object forKey:key];
    [ud synchronize];
}

// 移除
void UserDefaultRemoveObjForKey(NSString *key)
{
    NSUserDefaults *ud = [NSUserDefaults standardUserDefaults];
    [ud removeObjectForKey:key];
    [ud synchronize];
}

// 清空
void UserDefaultClean()
{
    NSUserDefaults *ud = [NSUserDefaults standardUserDefaults];
    [ud removePersistentDomainForName:[[NSBundle mainBundle]
bundleIdentifier]];
    [ud synchronize];
}


#pragma mark - SandBox 沙盒相关

NSString *pathDocuments()
{
    return NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES)[0];
```

```
}

NSString *pathCaches()
{
    return NSSearchPathForDirectoriesInDomains(NSCachesDirectory,
NSUserDomainMask, YES)[0];
}

/**
 *  Documents/name path
 *
 *  @param name
 *
 *  @return
 */
NSString *pathDocumentsWithFileName(NSString *name)
{
    return [pathDocuments() stringByAppendingString:name];
}

/**
 *  Caches/name path
 *
 *  @param name
 *
 *  @return
 */
NSString *pathCachesWithFileName(NSString *name)
{
    return [pathCaches() stringByAppendingString:name];
}


#pragma mark - 检查手机号码是否规范
/**
 *  检查是否为正确手机号码
 *
 *  @param phoneNumber 手机号
 *
 *  @return
 */
+ (BOOL)checkPhoneNumber:(NSString *)phoneNumber
{
    if (phoneNumber.length != 11)
    {
        return NO;
    }
    /**
     * 手机号码:
     * 13[0-9], 14[5,7], 15[0, 1, 2, 3, 5, 6, 7, 8, 9], 17[0, 1, 6, 7, 8], 18[0-9]
     * 移动号段:
134,135,136,137,138,139,147,150,151,152,157,158,159,170,178,182,183,184,187
,188
     * 联通号段: 130,131,132,145,155,156,170,171,175,176,185,186
     * 电信号段: 133,149,153,170,173,177,180,181,189
     */
    NSString *MOBILE = @"^1(3[0-9]|4[57]|5[0-35-9]|7[0135678]|8[0-9])\
\d{8}$";
    /**
     * 中国移动：China Mobile
     *
134,135,136,137,138,139,147,150,151,152,157,158,159,170,178,182,183,184,187
,188
     */
    NSString *CM = @"^1(3[4-9]|4[7]|5[0-27-9]|7[08]|8[2-478])\\d{8}$";
    /**
     * 中国联通：China Unicom
     * 130,131,132,145,155,156,170,171,175,176,185,186
     */
    NSString *CU = @"^1(3[0-2]|4[5]|5[56]|7[0156]|8[56])\\d{8}$";
    /**
     * 中国电信：China Telecom
     * 133,149,153,170,173,177,180,181,189
     */
    NSString *CT = @"^1(3[3]|4[9]|53|7[037]|8[019])\\d{8}$";


    NSPredicate *regextestmobile = [NSPredicate predicateWithFormat:@"SELF
MATCHES %@", MOBILE];
    NSPredicate *regextestcm = [NSPredicate predicateWithFormat:@"SELF
MATCHES %@", CM];
    NSPredicate *regextestcu = [NSPredicate predicateWithFormat:@"SELF
MATCHES %@", CU];
    NSPredicate *regextestct = [NSPredicate predicateWithFormat:@"SELF
MATCHES %@", CT];

    if (([regextestmobile evaluateWithObject:phoneNumber] == YES)
        || ([regextestcm evaluateWithObject:phoneNumber] == YES)
        || ([regextestct evaluateWithObject:phoneNumber] == YES)
        || ([regextestcu evaluateWithObject:phoneNumber] == YES))
    {
        return YES;
    }
    else
    {
        return NO;
    }
}
#pragma mark - 检查邮箱地址格式
/**
 *  检查邮箱地址格式
 *
 *  @param EmailAddress 邮箱地址
 *
 *  @return
 */
+ (BOOL)checkEmailAddress:(NSString *)EmailAddress
{
    NSString *emailRegEx =
    @"(?:[a-z0-9!#$%\\&'*+/=?\\^_`{|}~-]+(?:\\.[a-z0-9!#$%\\&'*+/=?\\^_`{|}"
    @"~-]+)*|\"(?:[\\x01-\\x08\\x0b\\x0c\\x0e-\\x1f\\x21\\x23-\\x5b\\x5d-\\"
    @"x7f]|\\\\[\\x01-\\x09\\x0b\\x0c\\x0e-\\x7f])*\")@(?:(?:[a-z0-9](?:[a-"
    @"z0-9-]*[a-z0-9])?\\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?|\\[(?:(?:25[0-5"
    @"]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-"
    @"9][0-9]?|[a-z0-9-]*[a-z0-9]:(?:[\\x01-\\x08\\x0b\\x0c\\x0e-\\x1f\\x21"
    @"-\\x5a\\x53-\\x7f]|\\\\[\\x01-\\x09\\x0b\\x0c\\x0e-\\x7f])+)\\])";

    NSPredicate *regExPredicate = [NSPredicate predicateWithFormat:@"SELF
MATCHES %@", emailRegEx];

    //先把NSString转换为小写
    NSString *lowerString       = EmailAddress.lowercaseString;

    return [regExPredicate evaluateWithObject:lowerString] ;
}
#pragma mark - 身份证相关
/**
 *  判断身份证是否合法
 *
 *  @param number 身份证号码
 *
 *  @return
 */
+ (BOOL)checkIdentityNumber:(NSString *)number
{
    {
        //必须满足以下规则
        //1. 长度必须是18位或者15位，前17位必须是数字，第十八位可以是数字或X
    //2. 前两位必须是以下情形中的一种：
11,12,13,14,15,21,22,23,31,32,33,34,35,36,37,41,42,43,44,45,46,50,51,52,53,5
4,61,62,63,64,65,71,81,82,91
    //3. 第7到第14位出生年月日。第7到第10位为出生年份；11到12位表示月份，
范围为01-12；13到14位为合法的日期
    //4. 第17位表示性别，双数表示女，单数表示男
    //5. 第18位为前17位的校验位
    //算法如下：
    //（1）校验和 = (n1 + n11) * 7 + (n2 + n12) * 9 + (n3 + n13) * 10 + (n4 +
n14) * 5 + (n5 + n15) * 8 + (n6 + n16) * 4 + (n7 + n17) * 2 + n8 + n9 * 6 + n10 *
3，其中n数值，表示第几位的数字
    //（2）余数 ＝ 校验和 % 11
    //（3）如果余数为0，校验位应为1，余数为1到10校验位应为字符
串“0X98765432”(不包括分号)的第余数位的值（比如余数等于3，校验位应为9）
    //6. 出生年份的前两位必须是19或20
    number = [number stringByTrimmingCharactersInSet:[NSCharacterSet
whitespaceAndNewlineCharacterSet]];
    number = [self filterSpecialString:number];
    //1️⃣判断位数
    if (number.length != 15 && number.length != 18) {
        return NO;
    }
    //2️⃣将15位身份证转为18位
    NSMutableString *mString = [NSMutableString stringWithString:number];
    if (number.length == 15) {
        //出生日期加上年的开头
        [mString insertString:@"19" atIndex:6];
        //最后一位加上校验码
        [mString insertString:[self
getLastIdentifyNumberForIdentifyNumber:mString] atIndex:[mString length]];
        number = mString;
    }
    //3️⃣开始判断
    NSString *mmdd = @"(((0[13578]|1[02])(0[1-9]|[12][0-9]|3[01]))|((0[469]|
11)(0[1-9]|[12][0-9]|30))|(02(0[1-9]|[1][0-9]|2[0-8])))";
    NSString *leapMmdd = @"0229";
    NSString *year = @"(19|20)[0-9]{2}";
    NSString *leapYear = @"(19|20)(0[48]|[2468][048]|[13579][26])";
    NSString *yearMmdd = [NSString stringWithFormat:@"%@%@", year,
mmdd];
    NSString *leapyearMmdd = [NSString stringWithFormat:@"%@%@",
leapYear, leapMmdd];
    NSString *yyyyMmdd = [NSString stringWithFormat:@"((%@)|(%@)|
(%@))", yearMmdd, leapyearMmdd, @"20000229"];
    //区域
    NSString *area = @"(1[1-5]|2[1-3]|3[1-7]|4[1-6]|5[0-4]|6[1-5]|82|[7-9]1)
```

```
[0-9]{4}";
        NSString *regex = [NSString stringWithFormat:@"%@%@%@", area,
yyyyMmdd  , @"[0-9]{3}[0-9Xx]"];
        NSPredicate *regexTest = [NSPredicate predicateWithFormat:@"SELF
MATCHES %@", regex];

        if (![regexTest evaluateWithObject:number]) {
            return NO;
        }
        //4️⃣验证校验码
        return [[self getLastIdentifyNumberForIdentifyNumber:number]
isEqualToString:[number substringWithRange:NSMakeRange(17, 1)]];
    }
}
/**
 *  从身份证里面获取性别man 或者 woman 不正确的身份证返回nil
 *
 *  @param number 身份证
 *
 *  @return
 */
+ (NSString *)getGenderFromIdentityNumber:(NSString *)number
{
    if ([self checkIdentityNumber:number]) {
        number = [self filterSpecialString:number];
        NSInteger i = [[number substringWithRange:NSMakeRange(number.length
- 2, 1)] integerValue];
        if (i % 2 == 1) {
            return @"man";
        } else {
            return @"woman";
        }
    } else {
        return nil;
    }
}
/**
 *  从身份证获取生日,身份证格式不正确返回nil,正确返回:1990年01月01日
 *
 *  @param number 身份证
 *
 *  @return
 */
+ (NSString *)getBirthdayFromIdentityNumber:(NSString *)number
{
    if ([self checkIdentityNumber:number]) {
        number = [self filterSpecialString:number];
        if (number.length == 18) {
            return [NSString stringWithFormat:@"%@年%@月%@日",[number
substringWithRange:NSMakeRange(6,4)], [number
substringWithRange:NSMakeRange(10,2)], [number
substringWithRange:NSMakeRange(12,2)]];
        }
        if (number.length == 15) {
            return [NSString stringWithFormat:@"19%@年%@月%@日",[number
substringWithRange:NSMakeRange(6,2)], [number
substringWithRange:NSMakeRange(8,2)], [number
substringWithRange:NSMakeRange(10,2)]];
        };
        return nil;
    } else {
        return nil;
    }
}

+ (NSString *)getLastIdentifyNumberForIdentifyNumber:(NSString *)number {
    //位数不小于17
    if (number.length < 17) {
        return nil;
    }
    //加权因子
    int R[] = {7,9,10,5,8,4,2,1,6,3,7,9,10,5,8,4,2};
    //校验码
    unsigned char sChecker[11] = {'1','0','X','9','8','7','6','5','4','3','2'};
    long p =0;
    for (int i =0; i<=16; i++){
        NSString * s = [number substringWithRange:NSMakeRange(i, 1)];
        p += [s intValue]*R[i];
    }
    //校验位
    int o = p%11;
    NSString *string_content = [NSString stringWithFormat:@"%c",sChecker[o]];
    return string_content;
}

#pragma mark - JSON和字典、数组
/**
 *  JSON字符串转字典或者数组
 *
 *  @param string JSON字符串
 *
 *  @return 返回字典或者数组
 */
id JSONTransformToDictionaryOrArray(NSString *string)
{
   NSError *error;
   id object = [NSJSONSerialization JSONObjectWithData:[string
dataUsingEncoding:NSUTF8StringEncoding] options:
(NSJSONReadingMutableContainers) error:&error];
   if (error != nil) {
#ifdef DEBUG
      NSLog(@"fail to get dictioanry or array from JSON: %@, error: %@", string,
error);
#endif
   }
   return object;
}

/**
 *  字典或者数组转为JSON字符串
 *
 *  @param object 字典或者数组
 *
 *  @return 返回字符串
 */
NSString *dictionaryOrArrayTransformToString(id object)
{
   if (![object isKindOfClass:[NSArray class]] && ![object isKindOfClass:
[NSDictionary class]])
   {
      return nil;
   }
   NSError *error;
   //options为0 则不会有换行符和空格   NSJSONWritingPrettyPrinted有空格和换
行符方便阅读
   NSData *data = [NSJSONSerialization dataWithJSONObject:object options:
(0) error:&error];
   if (error != nil) {
#ifdef DEBUG
      NSLog(@"fail to get JSON from object: %@, error: %@", object, error);
#endif
   }
   return [[NSString alloc] initWithData:data encoding:NSUTF8StringEncoding];
}


#pragma mark - 屏幕截图的几种方式
```

```
/**
 *  屏幕截图有状态栏
 *
 *  @param type 图片保存位置
 *
 *  @return
 */
+ (UIImage *)imageWithScreenshotWithCaptureType:(CaptureType)type
{
    CGSize imageSize = [UIScreen mainScreen].bounds.size;
    UIGraphicsBeginImageContextWithOptions(imageSize, YES, 0);

    for (UIWindow *window in [UIApplication sharedApplication].windows) {
        if (window.screen == [UIScreen mainScreen]) {
            [window drawViewHierarchyInRect:[UIScreen mainScreen].bounds
afterScreenUpdates:NO];
        }
    }
    UIView *statusBar = [[UIApplication sharedApplication]
valueForKey:@"statusBar"];
    [statusBar drawViewHierarchyInRect:statusBar.bounds
afterScreenUpdates:NO];
    UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();

    [self saveImage:image WithCaptureType:type];

    return image;
}


/**
 *  屏幕截图没有状态栏
 *
 *  @param type 图片保存位置
 *
 *  @return
 */
+ (UIImage *)imageWithScreenshotNoStatusBarWithCaptureType:
(CaptureType)type
{
    CGSize size = [UIScreen mainScreen].bounds.size;

    UIGraphicsBeginImageContextWithOptions(size, NO, 0);

    CGContextRef context = UIGraphicsGetCurrentContext();
    for (UIWindow *window in [UIApplication sharedApplication].windows)
    {
        if (window.screen == [UIScreen mainScreen]) {
            CGContextSaveGState(context);
            CGContextTranslateCTM(context, window.center.x, window.center.y);
            CGContextConcatCTM(context, window.transform);
            CGContextTranslateCTM(context, -window.bounds.size.width
*window.layer.anchorPoint.x, -window.bounds.size.height
*window.layer.anchorPoint.y);
            [window.layer renderInContext:context];
            CGContextRestoreGState(context);
        }

    }
    UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();

    [self saveImage:image WithCaptureType:type];
    return image;

}

/**
 *  给一个view截图
 *
 *  @param type 图片保存位置
 *
 *  @return
 */
+ (UIImage *)imageForView:( UIView * _Nonnull )view withCaptureType:
(CaptureType)type
{
    CGSize size = view.bounds.size;

    UIGraphicsBeginImageContextWithOptions(size, NO, 0);
    CGContextRef context = UIGraphicsGetCurrentContext();
    [view.layer renderInContext:context];
    UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();

    [self saveImage:image WithCaptureType:type];
    return image;

}

/**
```

```
 *  保存image到指定的位置
 *
 *  @param image image
 *  @param type  类型
 */
+ (void)saveImage:(UIImage *)image WithCaptureType:(CaptureType)type
{

    NSData *data = UIImagePNGRepresentation(image);
    /**
     *  时间戳
     */
    NSString *time =[NSString stringWithFormat:@"%.0f", [[NSDate date]
timeIntervalSince1970]];
    switch (type) {
        case CaptureTypeSandbox:
        {
            [data writeToFile:pathCachesWithFileName([NSString
stringWithFormat:@"%@_mainScrren_status.png",time]) atomically:YES];
        }
            break;
        case CaptureTypePhotes:
        {
            [data writeToFile:pathCachesWithFileName([NSString
stringWithFormat:@"%@_mainScrren_status.png",time]) atomically:YES];

        }
            break;
        case CaptureTypeBoth:
        {
            [data writeToFile:pathCachesWithFileName([NSString
stringWithFormat:@"%@_mainScrren_status.png",time]) atomically:YES];
            [data writeToFile:pathCachesWithFileName([NSString
stringWithFormat:@"%@_mainScrren_status.png",time]) atomically:YES];
        }
            break;
        default:
            break;
    }
}

// 指定回调方法
+ (void)image:(UIImage *)image didFinishSavingWithError:(NSError *)error
contextInfo:(void *)contextInfo {
    NSString *msg = nil ;
#ifdef DEBUG
```

```
    if(error != NULL)
    {
        msg = @"保存图片失败" ;
    } else {
        msg = @"保存图片成功" ;
    }
#endif
}

#pragma mark - 获取当前Controller
//获取当前屏幕显示的viewcontroller
+ (UIViewController *)getCurrentVC
{
    UIViewController *result = nil;

    UIWindow * window = [[UIApplication sharedApplication] keyWindow];
    if (window.windowLevel != UIWindowLevelNormal)
    {
        NSArray *windows = [[UIApplication sharedApplication] windows];
        for(UIWindow * tmpWin in windows)
        {
            if (tmpWin.windowLevel == UIWindowLevelNormal)
            {
                window = tmpWin;
                break;
            }
        }
    }
    UIView *frontView = [[window subviews] objectAtIndex:0];
    id nextResponder = [frontView nextResponder];

    if ([nextResponder isKindOfClass:[UIViewController class]])
        result = nextResponder;
    else
        result = window.rootViewController;

    return result;
}

+ (UIViewController *)findViewController:(UIView *)sourceView
{
    id target = sourceView;
    while (target) {
        target = ((UIResponder *)target).nextResponder;
        if ([target isKindOfClass:[UIViewController class]]) {
            break;
        }
    }
    return target;
}


/////////////#pragma mark -- 本地视频文件播放//////////////
#import <AVFoundation/AVFoundation.h> //需要导入框架

#define EYScreenWidth [[UIScreen mainScreen] bounds].size.width
#define EYScreenHeight [[UIScreen mainScreen] bounds].size.height

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];

    //1.从mainBundle获取test.mp4的具体路径
    NSString * path = [[NSBundle mainBundle] pathForResource:@"test"
ofType:@"mp4"];
    //2.文件的url
    NSURL * url = [NSURL fileURLWithPath:path];

    //3.根据url创建播放器(player本身不能显示视频)
    AVPlayer * player = [AVPlayer playerWithURL:url];

    //4.根据播放器创建一个视图播放的图层
    AVPlayerLayer * layer = [AVPlayerLayer playerLayerWithPlayer:player];

    //5.设置图层的大小
    layer.frame = CGRectMake(0, 0, EYScreenWidth, EYScreenHeight);

    //6.添加到控制器的view的图层上面
    [self.view.layer addSublayer:layer];

    //7.开始播放
    [player play];
}
```

1、禁止手机睡眠

```
[UIApplication sharedApplication].idleTimerDisabled = YES;
```

2、隐藏某行cell

```
- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:
(NSIndexPath *)indexPath
{
// 如果是你需要隐藏的那一行，返回高度为0
    if(indexPath.row == YouWantToHideRow)
        return 0;
    return 44;
}
// 然后再你需要隐藏cell的时候调用
[self.tableView beginUpdates];
[self.tableView endUpdates];
```

3、禁用button高亮

```
button.adjustsImageWhenHighlighted = NO;
```

或者在创建的时候

```
 UIButton *button = [UIButton buttonWithType:UIButtonTypeCustom];
```

7、动画切换window的根控制器

```
// options是动画选项
[UIView transitionWithView:[UIApplication sharedApplication].keyWindow
duration:0.5f options:UIViewAnimationOptionTransitionCrossDissolve
animations:^{
        BOOL oldState = [UIView areAnimationsEnabled];
        [UIView setAnimationsEnabled:NO];
        [UIApplication sharedApplication].keyWindow.rootViewController =
[RootViewController new];
        [UIView setAnimationsEnabled:oldState];
    } completion:^(BOOL finished) {

    }];
```

8、去除数组中重复的对象

```
NSArray *newArr = [oldArr valueForKeyPath:@“@distinctUnionOfObjects.self"];
```

15、跳进app权限设置

```
// 跳进app设置
            if (UIApplicationOpenSettingsURLString != NULL) {
                NSURL *url = [NSURL
URLWithString:UIApplicationOpenSettingsURLString];
                [[UIApplication sharedApplication] openURL:url];
            }
        }
```

16、给一个view截图

```
UIGraphicsBeginImageContextWithOptions(view.bounds.size, YES, 0.0);
    [view.layer renderInContext:UIGraphicsGetCurrentContext()];
    UIImage *img = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
```

19、collectionView的内容小于其宽高的时候是不能滚动的，设置可以滚动：

```
collectionView.alwaysBounceHorizontal = YES;
collectionView.alwaysBounceVertical = YES;
```

20、设置navigationBar上的title颜色和大小

```
  [self.navigationController.navigationBar
setTitleTextAttributes:@{NSForegroundColorAttributeName : [UIColor
youColor], NSFontAttributeName : [UIFont systemFontOfSize:15]}]
```

22、view设置圆角

```
#define ViewBorderRadius(View, Radius, Width, Color)\
\
[View.layer setCornerRadius:(Radius)];\
[View.layer setMasksToBounds:YES];\
[View.layer setBorderWidth:(Width)];\
[View.layer setBorderColor:[Color CGColor]] // view圆角
```

28、获取沙盒 Document

```
#define PathDocument
[NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES) firstObject]
```

27、获取temp

```
#define PathTemp NSTemporaryDirectory()
```

29、获取沙盒 Cache

```
#define PathCache
[NSSearchPathForDirectoriesInDomains(NSCachesDirectory,
NSUserDomainMask, YES) firstObject]
```

38、获取window

```
+(UIWindow*)getWindow {
    UIWindow* win = nil; //[UIApplication sharedApplication].keyWindow;
    for (id item in [UIApplication sharedApplication].windows) {
        if ([item class] == [UIWindow class]) {
            if (!((UIWindow*)item).hidden) {
                win = item;
                break;
            }
        }
    }
    return win;
}
```

39、修改textField的placeholder的字体颜色、大小

```
[textField setValue:[UIColor redColor]
forKeyPath:@"_placeholderLabel.textColor"];
[textField setValue:[UIFont boldSystemFontOfSize:16]
forKeyPath:@"_placeholderLabel.font"];
```

40、统一收起键盘

```
[[[UIApplication sharedApplication] keyWindow] endEditing:YES];
```

42、获取app缓存大小

```
- (CGFloat)getCachSize {

   NSUInteger imageCacheSize = [[SDImageCache sharedImageCache]
getSize];
   //获取自定义缓存大小
   //用枚举器遍历 一个文件夹的内容
   //1.获取 文件夹枚举器
   NSString *myCachePath = [NSHomeDirectory()
stringByAppendingPathComponent:@"Library/Caches"];
   NSDirectoryEnumerator *enumerator = [[NSFileManager defaultManager]
enumeratorAtPath:myCachePath];
   __block NSUInteger count = 0;
   //2.遍历
   for (NSString *fileName in enumerator) {
      NSString *path = [myCachePath
stringByAppendingPathComponent:fileName];
      NSDictionary *fileDict = [[NSFileManager defaultManager]
attributesOfItemAtPath:path error:nil];
      count += fileDict.fileSize;//自定义所有缓存大小
   }
   // 得到是字节  转化为M
   CGFloat totalSize = ((CGFloat)imageCacheSize+count)/1024/1024;
   return totalSize;
}
```

43、清理app缓存

```
- (void)handleClearView {
   //删除两部分
   //1.删除 sd 图片缓存
   //先清除内存中的图片缓存
   [[SDImageCache sharedImageCache] clearMemory];
   //清除磁盘的缓存
   [[SDImageCache sharedImageCache] clearDisk];
   //2.删除自己缓存
   NSString *myCachePath = [NSHomeDirectory()
stringByAppendingPathComponent:@"Library/Caches"];
   [[NSFileManager defaultManager] removeItemAtPath:myCachePath error:nil];
}
```

46、打印百分号和引号

```
   NSLog(@"%%");
   NSLog(@"\"");
```

49、长按复制功能

```
- (void)viewDidLoad
{
   [self.view addGestureRecognizer:[[UILongPressGestureRecognizer alloc]
initWithTarget:self action:@selector(pasteBoard:)]];
}
- (void)pasteBoard:(UILongPressGestureRecognizer *)longPress {
   if (longPress.state == UIGestureRecognizerStateBegan) {
      UIPasteboard *pasteboard = [UIPasteboard generalPasteboard];
      pasteboard.string = @"需要复制的文本";
   }
}
```

52、判断图片类型

```
//通过图片Data数据第一个字节 来获取图片扩展名
- (NSString *)contentTypeForImageData:(NSData *)data
{
   uint8_t c;
   [data getBytes:&c length:1];
   switch (c)
   {
      case 0xFF:
         return @"jpeg";

      case 0x89:
         return @"png";

      case 0x47:
         return @"gif";

      case 0x49:
      case 0x4D:
         return @"tiff";

      case 0x52:
      if ([data length] < 12) {
         return nil;
      }

      NSString *testString = [[NSString alloc] initWithData:[data
subdataWithRange:NSMakeRange(0, 12)] encoding:NSASCIIStringEncoding];
      if ([testString hasPrefix:@"RIFF"]
         && [testString hasSuffix:@"WEBP"])
      {
         return @"webp";
      }
```

```
        return nil;
    }

    return nil;
}
```

53、获取手机和app信息

```
NSDictionary *infoDictionary = [[NSBundle mainBundle] infoDictionary];
 CFShow(infoDictionary);
// app名称
 NSString *app_Name = [infoDictionary
objectForKey:@"CFBundleDisplayName"];
 // app版本
 NSString *app_Version = [infoDictionary
objectForKey:@"CFBundleShortVersionString"];
 // app build版本
 NSString *app_build = [infoDictionary objectForKey:@"CFBundleVersion"];



    //手机序列号
    NSString* identifierNumber = [[UIDevice currentDevice] uniqueIdentifier];
    NSLog(@"手机序列号: %@",identifierNumber);
    //手机别名： 用户定义的名称
    NSString* userPhoneName = [[UIDevice currentDevice] name];
    NSLog(@"手机别名: %@", userPhoneName);
    //设备名称
    NSString* deviceName = [[UIDevice currentDevice] systemName];
    NSLog(@"设备名称: %@",deviceName );
    //手机系统版本
    NSString* phoneVersion = [[UIDevice currentDevice] systemVersion];
    NSLog(@"手机系统版本: %@", phoneVersion);
    //手机型号
    NSString* phoneModel = [[UIDevice currentDevice] model];
    NSLog(@"手机型号: %@",phoneModel );
    //地方型号  （国际化区域名称）
    NSString* localPhoneModel = [[UIDevice currentDevice] localizedModel];
    NSLog(@"国际化区域名称: %@",localPhoneModel );

    NSDictionary *infoDictionary = [[NSBundle mainBundle] infoDictionary];
    // 当前应用名称
    NSString *appCurName = [infoDictionary
objectForKey:@"CFBundleDisplayName"];
    NSLog(@"当前应用名称：%@",appCurName);
    // 当前应用软件版本  比如：1.0.1
    NSString *appCurVersion = [infoDictionary
objectForKey:@"CFBundleShortVersionString"];
    NSLog(@"当前应用软件版本:%@",appCurVersion);
    // 当前应用版本号码   int类型
    NSString *appCurVersionNum = [infoDictionary
objectForKey:@"CFBundleVersion"];
    NSLog(@"当前应用版本号码：%@",appCurVersionNum);
```

54、获取一个类的所有属性

```
id LenderClass = objc_getClass("Lender");
unsigned int outCount, i;
objc_property_t *properties = class_copyPropertyList(LenderClass, &outCount);
for (i = 0; i < outCount; i++) {
    objc_property_t property = properties[i];
    fprintf(stdout, "%s %s\n", property_getName(property),
property_getAttributes(property));
}
```

56、image拉伸

```
+ (UIImage *)resizableImage:(NSString *)imageName
{
    UIImage *image = [UIImage imageNamed:imageName];
    CGFloat imageW = image.size.width;
    CGFloat imageH = image.size.height;
    return [image resizableImageWithCapInsets:UIEdgeInsetsMake(imageH * 0.5,
imageW * 0.5, imageH * 0.5, imageW * 0.5)
resizingMode:UIImageResizingModeStretch];
}
```

61、拿到当前正在显示的控制器，不管是push进去的，还是present进去的都能拿到

```
- (UIViewController *)getVisibleViewControllerFrom:(UIViewController*)vc {
    if ([vc isKindOfClass:[UINavigationController class]]) {
        return [self getVisibleViewControllerFrom:[((UINavigationController*) vc)
visibleViewController]];
    }else if ([vc isKindOfClass:[UITabBarController class]]){
        return [self getVisibleViewControllerFrom:[((UITabBarController*) vc)
selectedViewController]];
    } else {
        if (vc.presentedViewController) {
            return [self getVisibleViewControllerFrom:vc.presentedViewController];
        } else {
            return vc;
        }
    }
}
```

81、一个字符串是否包含另一个字符串

```
// 方法1
if ([str1 containsString:str2]) {
        NSLog(@"str1包含str2");
    } else {
        NSLog(@"str1不包含str2");
    }

// 方法2
if ([str1 rangeOfString: str2].location == NSNotFound) {
        NSLog(@"str1不包含str2");
    } else {
        NSLog(@"str1包含str2");
    }
```

88、移除字符串中的空格和换行

```
+ (NSString *)removeSpaceAndNewline:(NSString *)str {
    NSString *temp = [str stringByReplacingOccurrencesOfString:@" "
withString:@""];
    temp = [temp stringByReplacingOccurrencesOfString:@"\r" withString:@""];
    temp = [temp stringByReplacingOccurrencesOfString:@"\n" withString:@""];
    return temp;
}
```

89、判断字符串中是否有空格

```
+ (BOOL)isBlank:(NSString *)str {
    NSRange _range = [str rangeOfString:@" "];
    if (_range.location != NSNotFound) {
        //有空格
        return YES;
    } else {
        //没有空格
        return NO;
    }
}
```

112、scrollView滚动到最下边

```
CGPoint bottomOffset = CGPointMake(0, scrollView.contentSize.height -
scrollView.bounds.size.height);
[scrollView setContentOffset:bottomOffset animated:YES];
```

115、为UIView某个角添加圆角

```
// 左上角和右下角添加圆角
UIBezierPath *maskPath = [UIBezierPath
bezierPathWithRoundedRect:view.bounds byRoundingCorners:
(UIRectCornerTopLeft | UIRectCornerBottomRight)
cornerRadii:CGSizeMake(20, 20)];
    CAShapeLayer *maskLayer = [CAShapeLayer layer];
    maskLayer.frame = view.bounds;
    maskLayer.path = maskPath.CGPath;
    view.layer.mask = maskLayer;
```

117、将一个view放置在其兄弟视图的最上面

```
[parentView bringSubviewToFront:yourView]
```

118、将一个view放置在其兄弟视图的最下面

```
[parentView sendSubviewToBack:yourView]
```

128、获取一个view所属的控制器

```
// view分类方法
- (UIViewController *)belongViewController {
    for (UIView *next = [self superview]; next; next = next.superview) {
        UIResponder* nextResponder = [next nextResponder];
        if ([nextResponder isKindOfClass:[UIViewController class]]) {
            return (UIViewController *)nextResponder;
        }
    }
    return nil;
}
```