第二周学习报告

OC学习

懒加载

```
// 懒加载 重写你的get方法
- (NSMutableArray*)items {
    if (_items == nil) {
        _items = [NSMutableArray new];
    }
    return _items;
}

@end
```

非必要是不去初始化以节省内存资源

NSInteger和NSNumber

前者是基本数据类型

后者是对象

OC中的NSArray只能存放对象类型，不能存放基本数据类型

可以用一个@

```
NSArray *arr = @[@1,@2,@3,@4,@5];
```

NSDate

一个特定时间点的表示，独立于任何日历或时区。

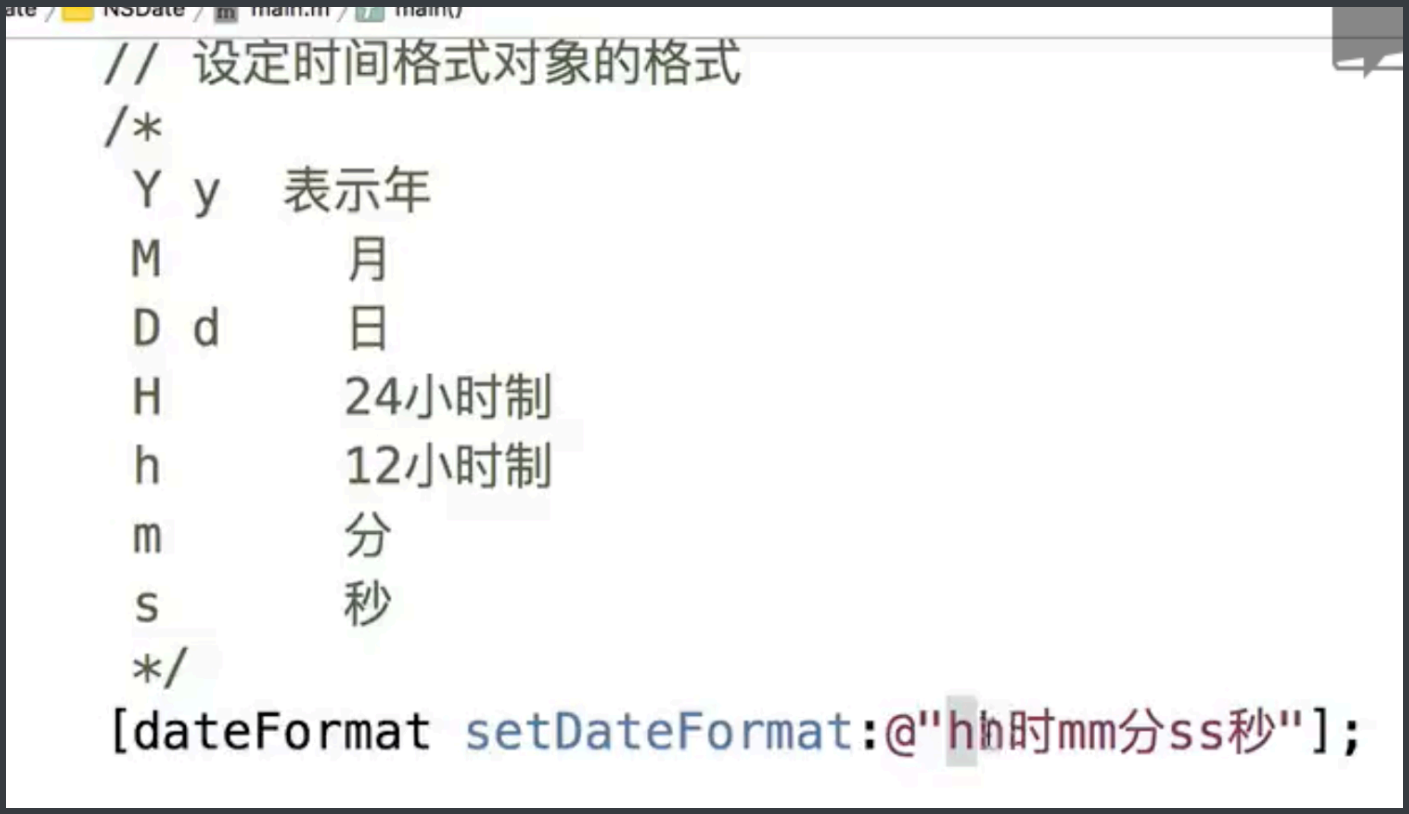
常用 date 类方法获取包含当前时间的NSDate实例

也常用。timeinterval　实例方法来计算时间间隔

格式化NSDate用

```
NSDateFormatter *formatter = [[NSDateFormatter alloc] dateFromString:
(nonnull NSString *)];
```

格式



把NSDate转换为字符串

```
[formatter stringFromDate:(nonnull NSDate *)]
```

数组

普通数组：NSArray

可变数组：NSMutableArray

可以对数组中的所有对象发送消息

```
Array makeObjectsperformSelector:…..
```

内省

询问ID类型 ， 一种自我保护的方法

```
isKindOfClass
IsMemberOfClass
if ([[accentList objectAtIndex:0] isKindOfClass:[BankAccent class]])
```

类作为属性

当一个类的属性为一个对象时，在实例化大类的时候需要将小类 alloc init

否则大类的小类属性只是一个nil指针，没有指向有效的对象

成员变量

OC最开始的时候用的在@interface后面直接中括号写成员变量，然后外部不能访问这些成员变量所以手动写get和set方法来间接访问，再后来苹果推荐用@property的这种用属性的写法，自动写好get和set方法，再后来为了方便提供了一个用 . 的方式访问，之前的也就几乎不用了

重写init方法

固定格式：

```
- (instancetype)init{
  self = [super init];
  if (self){
  ...
  }
  return self;
}
```

Get方法

在获取对象的成员变量的方法

在OC中可以用 “点” 来访问成员变量，这实际上也是消息发送

比如重写Get方法后用 “点”调用的是重写过的get方法

类扩展

在m文件中

```
@interface 名字()

@end
```

用于对一些不可继承的类添加功能或属性

用于多态的实现?

description

%@就是替换description返回的字符串

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

```
@property (readonly, copy) NSString *description;
- (NSString *)descriptionWithLocale:(nullable id)local
```

细节问题

头文件循环引用

解决方案@class 然后再在.m文件里面导入需要使用的.h文件

```
@class XXX
```

告诉编译器 我有一个名为XXX的类， 项目没有错误， 请继续～

OC动态类型识别

类的识别不是在编译时进行的，而是在运行时进行的，

动态绑定 确定类型后，再确定消息能否被响应。

OC类本质上也是一个被实例化过的对象， 也是占据内存的

所以可以向对象发送消息

他们都保存了isa指针， isa指针的类型是 class 类

NSLog不输出时间戳

```
#define NSLog(FORMAT, ...) printf("%s\n", [[NSString
stringWithFormat:FORMAT, ##__VA_ARGS__] UTF8String])
```