# 语法基础

## isa指针

每个对象都包含一个isa指针，指向当前对象所属的类，每次给对象发送消息时，对象会顺着isa指针找到存储于类中的方法，并执行。通过isa指针可以找到当前对象所属的类。

## self关键字

self类似于c++里的this指针

### 类方法中的self

- 整个程序运行过程中，一个类有且仅有一个类对象，通过类名调用方法，就是给这个类对象发消息
- 类方法里的self就是这个类对象，在类方法里通过self，可以调用其他类方法
- 不能在类方法里调用对象方法或成员变量，因为它们都是属于实例对象的

### 对象方法中的self

- 整个程序运行过程中，对象可以有0至多个
- 对象方法里的self就是调用当前对象
- 可以通过self来访问成员变量和本对象的其他方法

### 总结

- 谁调用的self，谁就是self
- self只能在方法中使用，不要使用self调用函数，也不可以在函数内部使用self
- 使用self调用本方法，会导致死循环

## super关键字

super是一个编译器指示符，告诉编译器在执行时去调用谁的方法
子类重写父类方法时，想保留父类的一些行为，就使用super

## CGFloat

```

```

## NSInteger

```

```

## id

一个可指向任意对象类型的指针，动态类型，直到执行时才确定对象所属的类
不能使用点语法，因为点语法是编译时特性，而id是运行时特性

### 动态类型判断类型

```

```

## 构造函数

### 重写init方法

```

```

## SEL类型

对象签名类型

```

```

## 类属性定义@property

简书博客

这篇进阶不错

- 控制setter方法的内存管理

retain：release旧值，retain新值（适用于oc对象）
assign：直接赋值，不做内存管理（适用于非oc对象类型）
copy：release旧值，copy新值（一般用于NSString *）

  - copy的对象被修改时，不会影响原对象，相当于复制了个副本

- 需不需要setter的方法

readwrite：同时生成set方法和get方法（默认的）
readonly：只有get方法

- 多线程

atomic：（默认）
nonatomic

- 控制set方法和get方法的名称

setter：设置set方法的名称
getter：设置get方法的名称

## 类方法定义

对象方法是一个实例对象的行为，比如张三具有吃的行为，这种行为有可能影响自身的某些状态；
类方法是一个类的行为，可以直接通过类名调用，需要使用某些数据必须通过参数传入，不能访问成员变量。

## @class的作用

告诉编译器这是一个类；并不会包含这个类的所有内容

- 在.h文件使用@class引用一个类
- 在.m文件使用#import包含一个类的.h文件

对于循环依赖关系来说，嵌套包含会出错，这时使用@class在两个类之间相互声明就好了

## 函数调用

- 不涉及面向对象的时候，跟C一样



- 涉及面向对象的时候



- oc里面没有私有方法的概念，一般写在.h的是public的，写在.m的是private的

## 消息传递

- 发送消息（调用函数）用方括号
- 获取对象用点

## 协议 protocol

- https://blog.csdn.net/developer_jiangqq/article/details/18980969 //基本概念

### 使用场景

- 需要由别的类实现的方法
- 声明未知类的接口
- 类与类之间的通信

### 基本特点

- 协议本身不是一个类，而是定义了一个其他类可以实现的接口
- 也就是说，protocol就是一系列方法的声明组成的

### 关键字

- @required 必须实现的方法
- @optional 选择性实现的方法

### 例

```
```

### 类型限制的使用方式

### Tips

- 协议只能声明不能实现
- 父类遵守了某协议，子类就也要遵守
- oc不可多继承，但可以多遵守协议
- 协议可以遵守协议
- 基协议
  NSObject是一个基类，所有类都要最终继承它；NSObject是一个最基本的协议

## 分类 Category

### 分类的作用

- 可以在不修改原来类的基础上，扩充一些方法
- 一个庞大的类可以分模块开发，多人协作

### 在.h文件中声明类别

- 新添加的方法必须写在@interface和@end之间
- className 现有类的类名(要为哪个类扩展方法)
- 待声明的类别名称
- 新添加的方法（不允许在声明类别的时候定义变量）

### 在.m文件中实现类别

- 写在@implementation和@end之间
- 新添加方法要实现

- 注意不要和其他分类，父类有重名的方法，否则会有重叠覆盖，按照特定顺序覆盖

## 类扩展 extension

- 扩展是分类的一个特例，也有人称它为匿名分类
- 可以为某个类扩充一些私有的成员方法和变量
  - 写在.m文件里
- 书写格式

## Block

- block是一种应用广泛的数据类型：动画、多线程、集合遍历、网络请求回调等
- 用来保存某一段代码，可以在恰当的时候取出来调用
- 功能类似于函数和方法

### 基础

### 进阶

如果block内部使用外部声明的强引用访问对象A，那么block内部会自动产生一个强引用指向对象A；
如果block内部使用外部声明的弱引用访问对象A，那么block内部会自动产生一个弱引用指向对象A；
如果参数是局部变量，Block是值传递；
如果参数是全局变量，静态变量，__block修饰的局部变量，Block是指针传递；

- ARC管理Block
  - 只要block引用外部局部变量，block放在堆空间
  - block使用strong，最好不要使用copy

### 实际使用

- 使用typedef定义一个block

- 这时myBlock就成为了一种Block类型，可以定义类属性

- 截获局部变量值

对于block外的变量引用，默认将其复制到其数据结构中实现访问的，默认情况下block只能访问不能修改局部变量的值。

- __block修饰的外部变量

对于用block修饰的外部变量引用，block是复制其地址来实现访问的，所以可以修改外部变量的值。

- block的三种类型
  - 全局块(_NSConcreteGlobalBlock) 存在于全局内存，相当于单例
  - 栈块(_NSConcreteStackBlock) 超出其作用域，马上被销毁
  - 堆块(_NSConcreteMallocBlock) 是一个带引用计数的对象，需要自行管理内存

如果block不访问外部变量，为全局块；

在ARC环境下，访问外界变量的block默认存储在堆中（本来在栈里，但是会被自动copy到堆中，自动释放）。

### 循环引用

某个类将block作为自己的属性变量，然后该类在block的方法体里面又使用了该类本身，只要block中用到了对象的属性或者函数，block就会持有该对象。

### 实例

- block作为变量

- block作为属性

- block作为方法参数

- block回调（留坑）

block回调是关于block最常用的内容，比如网络下载，可以用block实现下载成功和失败的反馈。

简单的例子：在B中输入字符串，通过回调返回给A

有一些小问题，一是循环引用，二是没有__block修饰的赋值。

## OC Copy

- 特点
  - 修改源文件的内容，不会影响副本文件，反之亦然
- 使用
  - 一个对象可以使用copy或者mutableCopy创建对象
  - copy创建的是不可变的副本，如NSString、NSArray、NSDictionary
  - mutableCopy创建的是可变副本，如NSMutableString、NSMutableArray、NSMutableDictionary
- 前提
  - 遵守NSCopying协议，实现copyWithZone方法
  - 遵守NSMutableCopying协议，实现mutableCopyWithZone方法

- 只有源对象和副本对象都是不可变时，采用的是浅复制，否则都是深复制