第4节课内容总结

成员变量存放在类对象的class_ro_t结构体当中。

类方法存在元类当中。

苹果为什么设计元类?

主要的目的是为了复用消息机制。在OC中调用方法，其实是在给某个对象发送某条消息。
消息的发送在编译的时候编译器就会把方法转换为objc_msgSend这个函数。
id objc_msgSend(id self, SEL op, ...) 这个函数有俩个隐式的参数：消息的接收者，消息的方法名。通过这俩个参数就能去找到对应方法的实现。

objc_msgSend函数就会通过第一个参数消息的接收者的isa指针，找到对应的类，如果我们是通过实例对象调用方法，那么这个isa指针就会找到实例对象的类对象，如果是类对象，就会找到类对象的元类对象，然后再通过SEL方法名找到对应的imp，然后就能找到方法对应的实现。

那如果没有元类的话，那这个objc_msgSend方法还得多加俩个参数，一个参数用来判断这个方法到底是类方法还是实例方法。一个参数用来判断消息的接受者到底是类对象还是实例对象。

消息的发送，越快越好。那如果没有元类，在objc_msgSend内部就会有有很多的判断，就会影响消息的发送效率。

所以元类的出现就解决了这个问题，让各类各司其职，实例对象就干存储属性值的事，类对象存储实例方法列表，元类对象存储类方法列表，符合设计原则中的单一职责，而且忽略了对对象类型的判断和方法类型的判断可以大大的提升消息发送的效率，并且在不同种类的方法走的都是同一套流程，在之后的维护上也大大节约了成本。

所以这个元类的出现，最大的好处就是能够复用消息传递这套机制。不管你是什么类型的方法，都是同一套流程。

在objc底层没有类方法和实例方法的区别，都是函数。

ro,rw,rwe

class_ro_t是在编译的时候生成的。当类在编译的时候，类的属性，实例方法，协议这些内容就存在class_ro_t这个结构体里面了，这是一块纯净的内存空间，不允许被修改。

class_rw_t是在运行的时候生成的，类一经使用就会变成class_rw_t，它会先将class_ro_t的内容"拿"过去，然后再将当前类的分类的这些属性、方法等拷贝到class_rw_t里面。它是可读写的。

class_rw_ext_t可以减少内存的消耗。苹果在wwdc2020里面说过，只有大约10%左右的类需要动态修改。所以只有10%左右的类里面需要生成class_rw_ext_t这个结构体。这样的话，可以节约很大一部分内存。

class_rw_ext_t生成的条件：
第一：用过runtime的Api进行动态修改的时候。
第二：有分类的时候，且分类和本类都为非懒加载类的时候。实现了+load方法即为非懒加载类。