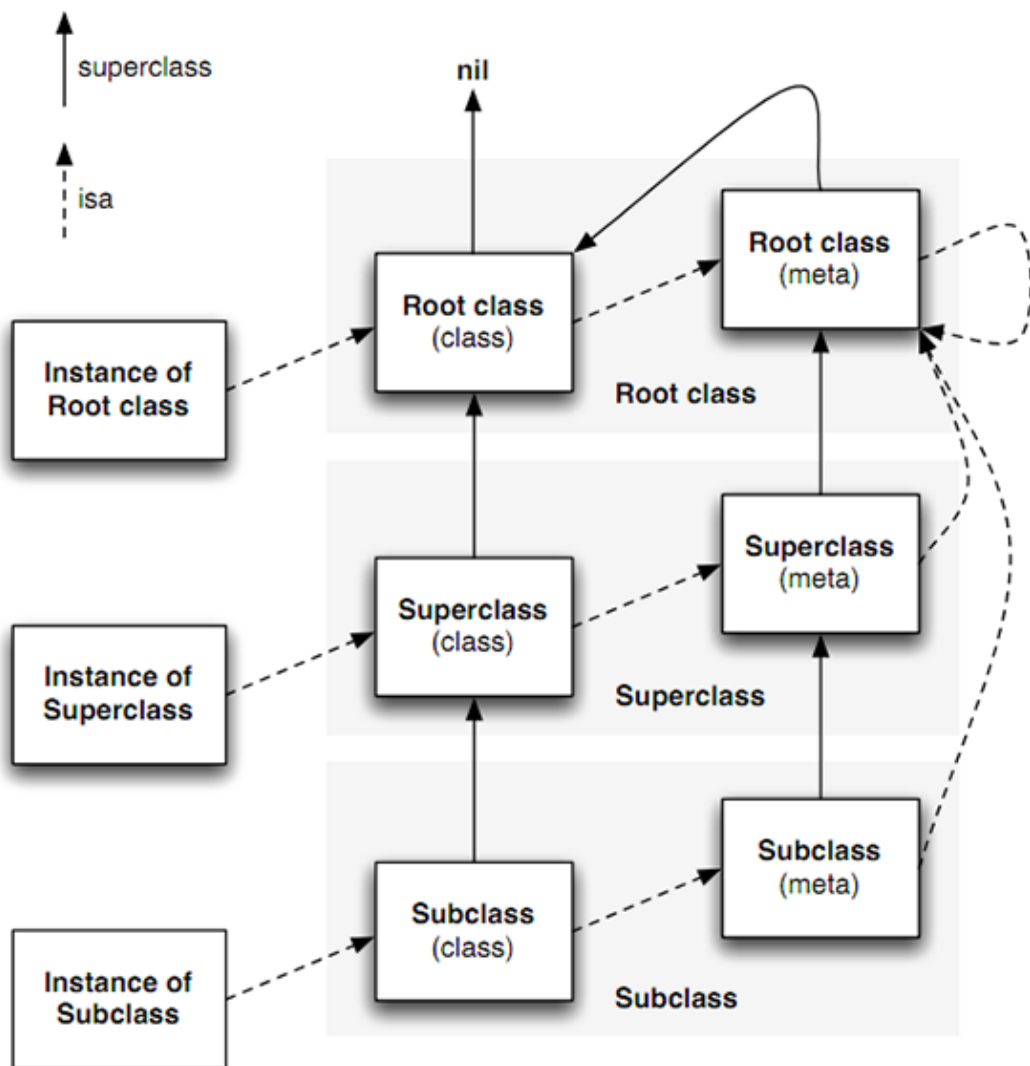


对象、类、元类的关系



Instance	存放 ivars 的值
Class	存放 ivars 、 instance_methods 、 properties
MetaClass	存放 class_methods

Class

```
struct objc_class {
    Class _Nonnull isa OBJC2_ISA_AVAILABILITY;

#if !__OBJC2__
    Class _Nullable super_class OBJC2_UNAVAILABLE;
    const char * _Nonnull name OBJC2_UNAVAILABLE;
    long version OBJC2_UNAVAILABLE;
    long info OBJC2_UNAVAILABLE;
    long instance_size OBJC2_UNAVAILABLE;
    struct objc_ivar_list * _Nullable ivars OBJC2_UNAVAILABLE;
    struct objc_method_list * _Nullable * _Nullable methodLists OBJC2_UNAVAILABLE;
    struct objc_cache * _Nonnull cache OBJC2_UNAVAILABLE;
    struct objc_protocol_list * _Nullable protocols OBJC2_UNAVAILABLE;
#endif

} OBJC2_UNAVAILABLE;
/* Use `Class` instead of `struct objc_class` */
```

▢ ▢ OBJC2e ▢ ▢

```
struct _class_ro_t {
    unsigned int flags;
    unsigned int instanceStart;
    unsigned int instanceSize;
    unsigned int reserved;
    const unsigned char *ivarLayout;
    const char *name;
    const struct _method_list_t *baseMethods;
    const struct _objc_protocol_list *baseProtocols;
    const struct _ivar_list_t *ivars;
    const unsigned char *weakIvarLayout;
    const struct _prop_list_t *properties;
};
```

```
struct _class_t {
    struct _class_t *isa;
    struct _class_t *superclass;
    void *cache;
    void *vtable;
    struct _class_ro_t *ro;
};
```

▢ ▢ OBJC2d Class ▢ ▢

样本代码分析

```
@interface AAA : NSObject

@property (readonly) NSString *name;
@property (class) BOOL abc;

- (void)testA;
+ (void)printA;

@end
```

```
@interface AAA ()
@property NSString *name;
@end

@implementation AAA

- (void)testA {
    self.name = @"123";
}

+ (void)printA {
    puts("AAAAAA");
}

@end
```

```
@interface BBB : AAA

@property (readonly) int age;
@property (class) BOOL xyz;

- (void)testB;
+ (void)printB;

@end
```

```
@interface BBB () {
    BOOL _isSelected;
}
@end

@implementation BBB

- (void)testB {
    _isSelected = !_isSelected;
}

+ (void)printB {
    puts("BBBBBB");
}

@end
```

使用clang重写：

```
clang -rewrite-objc BBB.m -o BBB.cpp
```

对象本质

```
struct NSObject_IMPL {  
    Class isa;  
};
```

```
struct AAA_IMPL {  
    struct NSObject_IMPL NSObject_IVARS;  
    NSString *_name;  
};
```

```
struct BBB_IMPL {  
    struct AAA_IMPL AAA_IVARS;  
    BOOL _isSelected;  
    int _age;  
};
```

根父类 ivars

...

父类 ivars

本类 ivars

1. 实例对象的包含其自身对应类的ivars，以及父类的ivars，父类的父类ivars.....根父类的ivars。

(根父类NSObject的ivars只有isa指针)

2. 实例对象除了ivar，没有别的。

类本质

```
static struct _class_ro_t _OBJC_CLASS_RO_$_BBB __attribute__((used, section
("__DATA,__objc_const"))) = {
    0, __OFFSETOFIVAR__(struct BBB, _isSelected), sizeof(struct BBB_IMPL),
    (unsigned int)0,
    0,
    "BBB",
    (const struct _method_list_t *)&_OBJC_$_INSTANCE_METHODS_BBB,
    0,
    (const struct _ivar_list_t *)&_OBJC_$_INSTANCE_VARIABLES_BBB,
    0,
    (const struct _prop_list_t *)&_OBJC_$_PROP_LIST_BBB,
};
```

```
extern "C" __declspec(dllexport) struct _class_t OBJC_CLASS_$_BBB __attribute__((used, section
("__DATA,__objc_data"))) = {
    0, // &OBJC_METAClass_$_BBB,
    0, // &OBJC_CLASS_$_AAA,
    0, // (void *)&_objc_empty_cache,
    0, // unused, was (void *)&_objc_empty_vtable,
    &_OBJC_CLASS_RO_$_BBB,
};
```

1. 类包含 instance_methods、ivars、properties。

元类本质

```
static struct _class_ro_t_OBJC_METACLASS_RO_$_BBB __attribute__((used, section
("__DATA,__objc_const"))) = {
    1, sizeof(struct _class_t), sizeof(struct _class_t),
    (unsigned int)0,
    0,
    "BBB",
    (const struct _method_list_t *)&_OBJC_$_CLASS_METHODS_BBB,
    0,
    0,
    0,
    0,
};
```

```
extern "C" __declspec(dllexport) struct _class_t_OBJC_METACLASS_$_BBB __attribute__((used,
section("__DATA,__objc_data"))) = {
    0, // &OBJC_METACLASS_$_NSObject,
    0, // &OBJC_METACLASS_$_AAA,
    0, // (void *)&_objc_empty_cache,
    0, // unused, was (void *)&_objc_empty_vtable,
    &_OBJC_METACLASS_RO_$_BBB,
};
```

1. 元类包含 class_methods。

property 的本质

```
struct _prop_t {
    const char *name;
    const char *attributes;
};

static struct /*_prop_list_t*/ {
    unsigned int entsize; // sizeof(struct _prop_t)
    unsigned int count_of_properties;
    struct _prop_t prop_list[1];
} _OBJC_$_PROP_LIST_BBB __attribute__((used, section("__DATA,__objc_const"))) = {
    sizeof(_prop_t),
    1,
    {"age", "Ti,R,V_age"}}
};
```

1. 属性包含名称、描述。

property 的本质是：ivar + getter + setter。

1. 没有 @synthesize、@dynamic 时，编译器会自动编写 ivar、getter、setter。
2. @synthesize：编译器自动编写 ivar、getter、setter。
3. @dynamic：用户手动编写 ivar、getter、setter。

category 的本质

```
@interface BBB (Cate) <NSCopying>
@property NSDate *birthday;
- (NSDate *)bornDate;
+ (void)desc;
@end
```

```
@implementation BBB (Cate)
- (NSDate *)birthday {
    return [NSDate date];
}
- (void)setBirthday:(NSDate *)birthday {
    NSLog(@">>> %@", birthday);
}
- (NSDate *)bornDate {
    return [NSDate date];
}
+ (void)desc {
    NSLog(@">>> %@", [self class]);
}
@end
```

```
struct _category_t {
    const char *name;
    struct _class_t *cls;
    const struct _method_list_t *instance_methods;
    const struct _method_list_t *class_methods;
    const struct _protocol_list_t *protocols;
    const struct _prop_list_t *properties;
};
```

```
static struct _category_t_OBJC_$_CATEGORY_BBB_$_Cate __attribute__((used, section
("__DATA,__objc_const"))) =
{
    "BBB",
    0, // &OBJC_CLASS_$_BBB,
    (const struct _method_list_t *)&_OBJC_$_CATEGORY_INSTANCE_METHODS_BBB_$_Cate,
    (const struct _method_list_t *)&_OBJC_$_CATEGORY_CLASS_METHODS_BBB_$_Cate,
    (const struct _protocol_list_t *)&_OBJC_CATEGORY_PROTOCOLS_$_BBB_$_Cate,
    (const struct _prop_list_t *)&_OBJC_$_PROP_LIST_BBB_$_Cate,
};
```

```
static void OBJC_CATEGORY_SETUP_$_BBB_$_Cate(void) {
    _OBJC_$_CATEGORY_BBB_$_Cate.cls = &OBJC_CLASS_$_BBB;
}
```

1. 分类包含分类所属的类、instance_methods、class_methods、protocols、properties。
2. 分类不包含分类名称。
3. 分类中不存在ivar，所以不能添加成员变量。分类中的属性自动生成的代码只有get、set方法。

protocol 的本质

```
@protocol Proto <NSObject>

@property int numberOfP;

- (void)reqMethod1;
+ (void)reqMethod1;

@optional
- (void)reqMethod2;
+ (void)reqMethod2;

@end
```

```
struct _protocol_t {
    void * isa; // NULL
    const char *protocol_name;
    const struct _protocol_list_t * protocol_list; // super protocols
    const struct method_list_t *instance_methods;
    const struct method_list_t *class_methods;
    const struct method_list_t *optionalInstanceMethods;
    const struct method_list_t *optionalClassMethods;
    const struct _prop_list_t * properties;
    const unsigned int size; // sizeof(struct _protocol_t)
    const unsigned int flags; // = 0
    const char ** extendedMethodTypes;
};
```

```
struct _protocol_t_OBJC_PROTOCOL_Protocol __attribute__((used)) = {
    0,
    "Proto",
    (const struct _protocol_list_t *)&_OBJC_PROTOCOL_REFS_Protocol,
    (const struct method_list_t *)&_OBJC_PROTOCOL_INSTANCE_METHODS_Protocol,
    (const struct method_list_t *)&_OBJC_PROTOCOL_CLASS_METHODS_Protocol,
    (const struct method_list_t *)&_OBJC_PROTOCOL_OPT_INSTANCE_METHODS_Protocol,
    (const struct method_list_t *)&_OBJC_PROTOCOL_OPT_CLASS_METHODS_Protocol,
    (const struct _prop_list_t *)&_OBJC_PROTOCOL_PROPERTIES_Protocol,
    sizeof(_protocol_t),
    0,
    (const char **)&_OBJC_PROTOCOL_METHOD_TYPES_Protocol
};
```

方法缓存

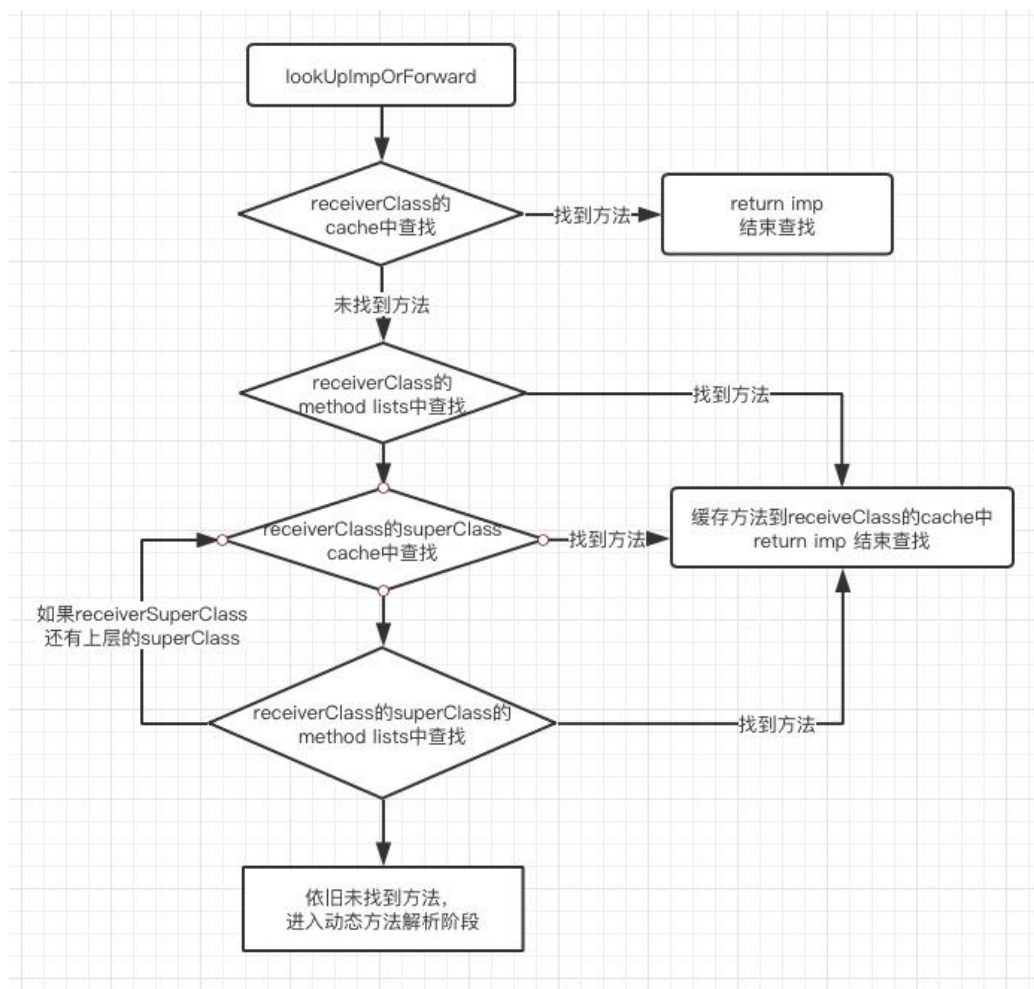
```
struct objc_cache {  
    unsigned int mask /* total = mask + 1 */          OBJC2_UNAVAILABLE;  
    unsigned int occupied                              OBJC2_UNAVAILABLE;  
    Method _Nullable buckets[1]                      OBJC2_UNAVAILABLE;  
};
```

初看以为方法缓存只能缓存一个方法，但实际上是：

```
Method _Nullable buckets[mask + 1];
```

这是C99后的变长数组。

```
static void OBJC_CLASS_SETUP_$_BBB(void) {  
    OBJC_METACLASS_$_BBB.isa = &OBJC_METACLASS_$_NSObject;  
    OBJC_METACLASS_$_BBB.superclass = &OBJC_METACLASS_$_AAA;  
    OBJC_METACLASS_$_BBB.cache = &_objc_empty_cache;  
    OBJC_CLASS_$_BBB.isa = &OBJC_METACLASS_$_BBB;  
    OBJC_CLASS_$_BBB.superclass = &OBJC_CLASS_$_AAA;  
    OBJC_CLASS_$_BBB.cache = &_objc_empty_cache;  
}
```



消息机制

1. 消息发送
2. 动态方法解析
3. 消息转发