The copy Attribute

* The copy attribute is an alternative to strong.
* Instead of taking ownership of the existing object, it creates a copy of whatever you assign to the property, then takes ownership of that
* Only objects that conform to the [NSCopying protocol](https://developer.apple.com/library/ios/documentation/cocoa/Reference/Foundation/Protocols/NSCopying_Protocol/Reference/Reference.html#//apple_ref/occ/intf/NSCopying) can use this attribute.
* Properties that represent values (opposed to connections or relationships) are good candidates for copying. For example, developers usually copy NSString properties instead of strongly reference them:

// Car.h

@property (nonatomic, copy) NSString \*model;

* Now, Car will store a brand new instance of whatever value we assign to model.
* If you’re working with mutable values, this has the added perk of freezing the object at whatever value it had when it was assigned. This is demonstrated below:

// main.m

#import <Foundation/Foundation.h>

#import "Car.h"

int main(int argc, const char \* argv[]) {

@autoreleasepool {

Car \*honda = [[Car alloc] init];

NSMutableString \*model = [NSMutableString stringWithString:@"Honda Civic"];

honda.model = model;

NSLog(@"%@", honda.model);

[model setString:@"Nissa Versa"];

NSLog(@"%@", honda.model); // Still "Honda Civic"

}

return 0;

}

Summary

The goal of all these attributes is to help us to focus on what data needs to be recorded by letting the compiler automatically determine how it’s represented. They are summarized below.

| **Attribute** | **Description** |
| --- | --- |
| getter= | Use a custom name for the getter method. |
| setter= | Use a custom name for the setter method. |
| readonly | Don’t synthesize a setter method. |
| nonatomic | Don’t guarantee the integrity of accessors in a multi-threaded environment. This is more efficient than the default atomic behavior. |
| strong | Create an owning relationship between the property and the assigned value. This is the default for object properties. |
| weak | Create a non-owning relationship between the property and the assigned value. Use this to prevent retain cycles. |
| copy | Create a copy of the assigned value instead of referencing the existing instance. |

Now that we’ve got properties out of the way, we can take an in-depth look at the other half of Objective-C classes: methods. We’ll explore everything from the quirks behind their naming conventions to dynamic method calls.