Handling Non-Object Values

Typically, your key-value coding compliant object relies on the default implementation of key-value coding to automatically wrap and unwrap non-object properties, as described in Representing Non-Object Values. However, you can override the default behavior. The most common reason to do so is to handle attempts to store a `nil` value on non-object properties.

> NOTE
> Because all properties in Swift are objects, this section only applies to Objective-C properties.

If your key-value coding compliant object receives a `setValue:forKey:` message with `nil` passed as the value for a non-object property, the default implementation has no appropriate, generalized course of action. It therefore sends itself a `setNilValueForKey:` message, which you can override. The default implementation of `setNilValueForKey:` raises an `NSInvalidArgumentException` exception, but you can provide an appropriate, implementation-specific behavior.

For example, the code in Listing 10-1 responds to an attempt to set a person's age to a `nil` value by instead setting the age to 0, which is more appropriate for a floating point value. Notice that the override method calls upon its object's superclass for any keys that it does not explicitly handle.

Listing 10-1 Example implementation of setNilValueForKey:

```
- (void)setNilValueForKey:(NSString *)key
{
    if ([key isEqualToString:@"age"]) {
        [self setValue:@(0) forKey:@”age”];
    } else {
        [super setNilValueForKey:key];
    }
}
```

> NOTE
> For backward compatibility, when an object has overridden the deprecated `unableToSetNilForKey:` method, `setValue:forKey:` invokes that method instead of `setNilValueForKey:`.