Refactoring Ugly Objective-C

I am - 🤓💻

😍 - Swift
😍 - Programming
😍 - Swift Package Manager

http://swifthighperformance.com
📙 Swift Hight Performance

Swift

Syntax and Expressiveness

~ 30% less code

Let's talk about Objective-C

Amazing Feature № 1

```
KKOStorage *storage = [KKOStorage new];
[storage saveItem:@"New item"];

KKONetwork *network = [KKONetwork new];
[network uploadItem:@"Some stuff"];
```

Amazing Feature № 1

```
KKOStorage *storage = [KKOStorage new];
[storage saveItem:@"New item"];

KKONetwork *network = [KKONetwork new];
[network uploadItem:@"Some stuff"];

[network saveItem:@"Some stuff"];
❗Error❗
```

BUT

```
[network performSelector: @selector(saveItem:)];

id network = [[KKONetwork alloc] init];
[network saveItem:@"😈"];

[network arrayByAddingObjectsFromArray: @[ @10 ]];
[network URLByAppendingPathComponent:@"/me.com"];
```

But who does ever use `id`?

🍏 Does!

```
// @interface NSObject <NSObject>
// - (id)copy;

NSMutableArray *ar = [NSMutableArray arrayWithArray:@[@1, @1]];
NSMutableArray *aCopy = [ar copy]; // it's a NSArray
[aCopy addObject: @2]; //Crash here!!
```

VS Swift

```
let storage = Storage()
storage.saveItem("Apple")

let network = Network()
network.uploadItem("Item")
// network.saveItem("i") // Error! Always!
```

VS Swift

```
let any: Any = Network()
//  any.saveItem("i") // Can't!!

if let any = any as? Storage {
  any.saveItem("i")
}
```

```
safety += 1
```

Objc `id`
vs
Swift `Any`

Objc `id`

Can do Anything. Whatever you Want!

```
id some;
[some URLCache];
[some stringByAppendingString:@":("];
```

Swift Any
Can't do Nothing

```
let any: Any = "String"
any.
//Any is empty. It has 0 methods

print(any)
```

```
safety += 2
```

3: Initialization

```
- (instancetype)init {
  self = [super init];
  if (self) {
      // Init ivars!
      // Don't access properties!!!
  }

  return self;
}
```

Go Crazy

```
- (instancetype)init {
  return 0; // -1, etc
}

- (instancetype)init {
  return [self init];
}

- (instancetype)init {
  return nil;
}

- (instancetype)init {
  // Do whatever You Want
}
```

```objc
@interface KKOArticle : NSObject

@property (nonatomic, readonly, strong) NSString *title;
@property (nonatomic, readonly, strong) NSString *text;
@property (nonatomic, readonly, strong) NSDate *date;

- (instancetype)initWith:(NSString *)title text:(NSString *)text date:(NSDate *)date;

@end

@implementation KKOArticle

- (instancetype)initWith:(NSString *)title text:(NSString *)text date:(NSDate *)date {
  self = [super init];
  if (self) {
    _title = title;
    _text = text;
    _date = date;
  }
  return self;
}

@end

KKOArticle *article = [[KKOArticle alloc] initWith:@"title" text:@"text" date:[NSDate date]];
```

No boilerplate code!

`init ()`

```
init () {
  // 🎉 Just put code here
}
```

Can't go Crazy :(

```
init () {
  // - must initialize all properties
  // - super.init is required when has a superclass
  // - can't access properties and methods until filly initialized
  // - very safe
}
```

Swift

```
struct Article {
  let title: String
  let text: String
  let date: NSDate
}

Article(title: "title", text: "Text", date: NSDate())
```

Safety += 3

Clean += 2

Optionals

Fun Quiz

$$a + b = c$$

$$b + a = c$$

```
NSString *a;
[a stringByAppendingString:@"b"];
```

```
NSString *a;
[a stringByAppendingString:@"b"]; // Nothing
[@"b" stringByAppendingString:a];
```

```
NSString *a;
[a stringByAppendingString:@"b"]; // Nothing
[@"b" stringByAppendingString:a]; // Crash!
```

WAT ?!

```objc
NSString *a;
[a stringByAppendingString:@"b"]; // Nothing
[@"b" stringByAppendingString:a]; // Crash!

[nil stringByAppendingString:@"b"];
[@"b" stringByAppendingString: nil];
```

```objc
NSString *a;
[a stringByAppendingString:@"b"]; // Nothing
[@"b" stringByAppendingString:a]; // Crash!

[nil stringByAppendingString:@"b"]; // Error
[@"b" stringByAppendingString: nil];
```

```
NSString *a;
[a stringByAppendingString:@"b"]; // Nothing
[@"b" stringByAppendingString:a]; // Crash!

[nil stringByAppendingString:@"b"]; // Error
[@"b" stringByAppendingString: nil]; // Warning
//stringByAppendingString:(nonnull NSString *)
```

```objc
NSString *a;
[a stringByAppendingString:@"b"]; // Nothing
[@"b" stringByAppendingString:a]; // Crash!

[nil stringByAppendingString:@"b"]; // Error
[@"b" stringByAppendingString: nil]; // Warning
//stringByAppendingString:(nonnull NSString *)

[(id)nil stringByAppendingString:a]; // 👍
```

```
let a = "a"

a + "b" /* 👍 */
"b" + a /* 👍 */
```

```
let a = "a"

a + "b" /* 👍 */
"b" + a /* 👍 */
```

```
let a: String?

a + "b" // Error
"b" + a // Error
```

```swift
let a: String?

a + "b" // Error
"b" + a // Error

if let a = a {
  a + "b" // 👍
}
```

```swift
let a: String?

a + "b" // Error
"b" + a // Error
a.stringByAppendingString("b") // Error

if let a = a {
  a + "b" // 👍
}
a?.stringByAppendingString("b")  // 👍
```

```
typedef NS_ENUM(NSInteger, Action) {
  ActionDelete,
  ActionCreate,
  ActionEdit,
  ActionCopy
};
```

```
typedef NS_ENUM(NSInteger, Action) {
  ActionDelete,
  ActionCreate,
  ActionEdit,
  ActionCopy
};

//+ (void)runAction:(Action)action;
[Runner runAction:ActionCreate];
```

```objc
typedef NS_ENUM(NSInteger, Action) {
  ActionDelete,
  ActionCreate,
  ActionEdit,
  ActionCopy
};

//+ (void)runAction:(Action)action;
[Runner runAction:ActionCreate];
[Runner runAction:20]; // WAT ?!
```

```objc
typedef NS_ENUM(NSInteger, Action) {
  ActionDelete,
  ActionCreate,
  ActionEdit,
  ActionCopy
};

//+ (void)runAction:(Action)action;
[Runner runAction:ActionCreate];
[Runner runAction:20]; // WAT ?!

// 😈🤖😈
Action action = ActionEdit;
action += ActionCopy;
action /= 56;
[Runner runAction:action];
```

```
+ (NSString *)actionString:(Action)action {
  switch (action) {
    case ActionDelete:
      return @"Delete";
    case ActionCreate:
      return @"Create";
    case ActionEdit:
      return @"Edit";
    case ActionCopy:
      return @"Copy";
  }
}
```

Swift

```
enum Action {
   case Delete
   case Create
   case Edit
   case Copy
}
```

```
enum Action {
   case Delete
   case Create
   case Edit
   case Copy
}

runAction(.Delete)
//runAction(10) // Error!!
```

```
enum Action: String {
   case Delete
   case Create
   case Edit
   case Copy
}

let action = Action.Delete.rawValue // Delete
```

```swift
enum Action: String {
  case Delete
  case Create
  case Edit
  case Copy

  var isDangerous: Bool {
    return self == .Delete
  }
}

let danger = Action.Delete.isDangerous // true
```

```objc
- (void)setup {
  [self setupWithName:[App name]];
}

- (void)setupWithName:(NSString *)name {
  [self setupWithName:name mode:ModeSqlite];
}

- (void)setupWithName:(NSString *)name mode:(Mode)mode {
  [self setupWithName:name mode:ModeSqlite logeLevel:LogLevelVerbose];
}

- (void)setupWithName:(NSString *)name mode:(Mode)mode logeLevel:(LogLevel)logLevel {
  ...
}

CoreDataStack *stack;

[stack setup];
[stack setupWithName:@"Data" mode:ModeInMemory];
[stack setupWithName:@"Data" mode:ModeInMemory logeLevel:LogLevelNone];

//[stack logeLevel:LogLevelNone]; Error
```

Swift - Parameters default values

```
func setup(name: String = App.name, mode: Mode = .Sqlite,  logLevel: LogLevel = .Verbose) {
  ...
}

let stack = CoreDataStack()

stack.setup()
stack.setup(logLevel: .None)
stack.setup(mode: .InMemory, logLevel: .None)
stack.setup("DB", mode: .InMemory, logLevel: .None)
```

"Is Swift ready for production?"

"Is Objective-C dying?"

Q & A