Coding: Best Practices

2014 Storm8

Stuff You Already Know

Just a refresher - good to promote

Increment++

- Make it work
  - Prototype it, get the general functionality
- Make it right
  - Clean up memory, test edge cases, etc.
- Make it fast
  - Only optimize once the first two are done

Increment++

- Make it work
  - Prototype it, get the general functionality

Functional wireframes, fake data, method stubs - it doesn’t need to be pretty.

Increment++

- Make it right
  - Clean up memory, test edge cases, etc.

No longer a prototype - refactor, clean up resources, add missing functionality, etc.

Increment++

- Make it fast
  - Only optimize once the first two are done

After profiling and feedback, target specific areas to optimize, then profile again.

Finding Solutions

How do I reverse an NSArray?

Finding Solutions

- Look for a solution on Google/SO
  - But build a **consensus** - don’t just use the first link
- Browse the iOS documentation
- Browse our source (Class+Protocol.h files)
- Ask around if someone has done it
- Share your experience with others!

Finding Solutions

```
- (NSArray *)reversed {
    NSMutableArray *array =
            [NSMutableArray arrayWithCapacity:[self count]];
    NSEnumerator *enumerator = [self reverseObjectEnumerator];
    for (id element in enumerator) {
        [array addObject:element];
    }
    return array;
}
```

Ok, here’s one solution. It works...

Finding Solutions

```
- (NSArray *)reversed {
    return [[self reverseObjectEnumerator] allObjects];
}
```

Here's another. A much simpler solution, but takes a little more research. Simpler = less error prone, more manageable, easier to read, smaller code size, etc.

Writing Code

- Code is written once, but possibly read and copied many, many times
- Code that is easy to read and understand is better than highly optimized code
  - Until the point it needs to be optimized
- Code that follows Apple conventions is more intuitive and easier to grasp

Writing Code

- Use @class in headers
  - Speeds up compilation, fewer files to compile
- Use Apple’s formatting and conventions
  - Text spacing, alignment, and general layout
  - Create a new project and browse Apple’s code
- Use if (count > 0) { } vs if (count) { }
  - Intention is clearer + is not dependent on language

Writing Code

- Use newlines liberally
  - Group blocks of code, return statements
- Comment on what needs commenting
  - A class, a method, a block of code, but **not**

```
x = 10; // Set x to 10
```

- Use new features of the language
  - Use @{} @[] notation, kill @synthesize

Writing Code

- Categories

```
@interface NSArray (NSS8Util)
- (int)intAtIndex:(int)index;
```

- Dynamic method calls

```
if (object respondsToSelector:@selector(foo)) {
    object performSelector:@selector(foo);
}
```

- Notifications

```
[[NSNotificationCenter defaultCenter]
        postNotificationName:@”foo” …]
```

Don’t Make Me Think

Code formatting and API consistency helps reduce mental workload.

Don’t Make Me Think

```
@inteface Foo
@propery int length;
@end

@inteface Bar
@propery int count;
@end

@inteface Baz
@propery int size;
@end
```

Don’t Make Me Think

```
if (baz.??? > 0) {
    ...
}
```

Hmm, does baz use length or count? Or size?

Don’t make me think! It should be intuitive from working with similar code.

Code Reviews

- Code Reviews are meant to:
  - Be a sanity check after staring at code
  - Review designs, suggest improvements, etc.
  - Broaden knowledge base of new code
  - Make incremental changes (add dormant func.)
- Be stingy in code reviews
  - Including spacing, formatting and general readability
  - It’s also your code now. Are you OK with it?

Related

Design Principles and Patterns

Objective-C Style Guide

Objective-C Method Naming Demystified

The Art of Readable Code