

#### **Ease the Pain with Code Generation**

Patrick Goley

@bitsbetweenbits

#### What is Code Generation?

 Automatically generating code that would normally be handwritten. Writing code = bugs.

Usually done as part of a build script or preprocessor before compilation

 Can reduce boilerplate code, provide new language syntax or additional type safety

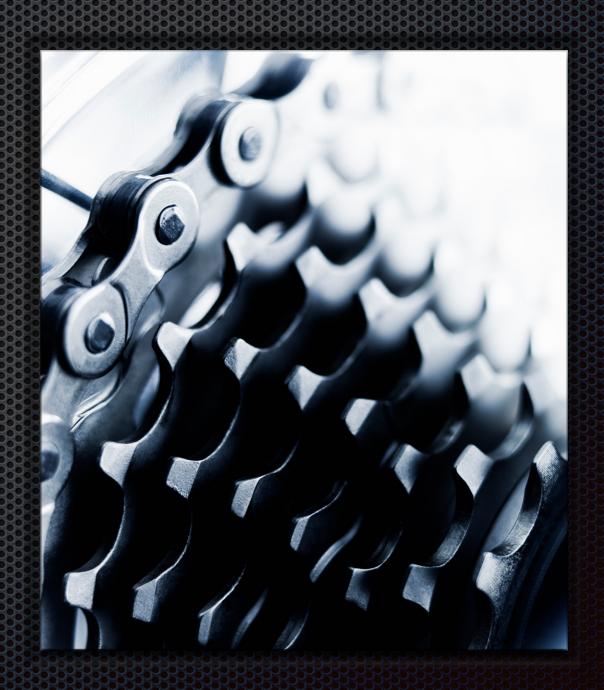
## Synthesized Protocol Conformance provided by Swift compiler

```
struct Flight {
   let airline: String
   let flightNumber: Int
   let departure: Date
extension Flight: Equatable { }
extension Flight: Codable { }
extension Flight: Hashable { }
```

# What about my protocols?

### Sourcery

- Uses SourceKit to read your source files and the types within them
- Uses templates to generate new source files



# What else can we use to generate code?

#### Core Data Classes

 Xcode generates model classes based on entities in a model file

Generates type safe NSFetchRequests

 NSPredicates still not type safe, prone to errors when the model changes

### Can we do better?

# CoreDataQueryInterface (CDQI)

 Swift library for type-safe NSPredicates based on Core Data model

Command line tool to generate Swift code

 Provides type safety on both the attributes being queried and the values they are compared to

# That's great but I write Objective-C

### Clang Preprocessor Macros

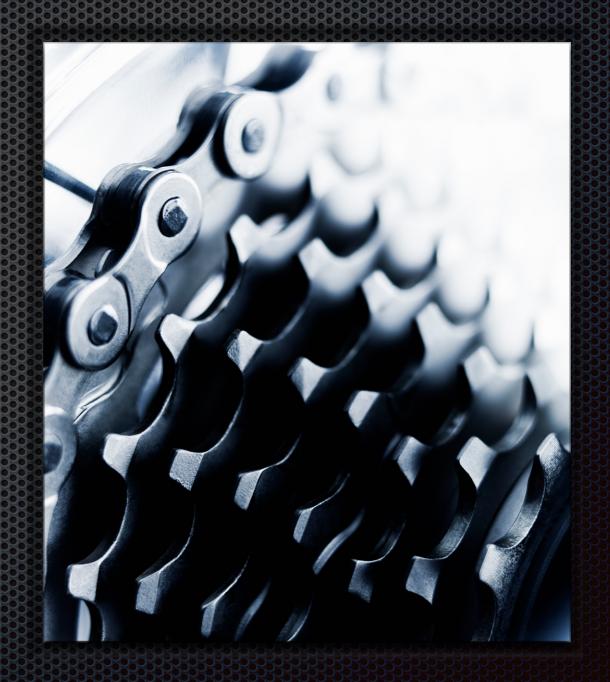
Allows templating within C, C++, Obj-C source files

Conditional compilation (i.e. code for debug builds only)

Not "hygienic" macros. Just string manipulation of your source code

#### Conclusion

- Use code generation to extend your language or development workflow
- Look for "ambient" information to leverage
- Remove boilerplate, repetitive patterns



## Trivia Time!