What The IndexStore Has To Say

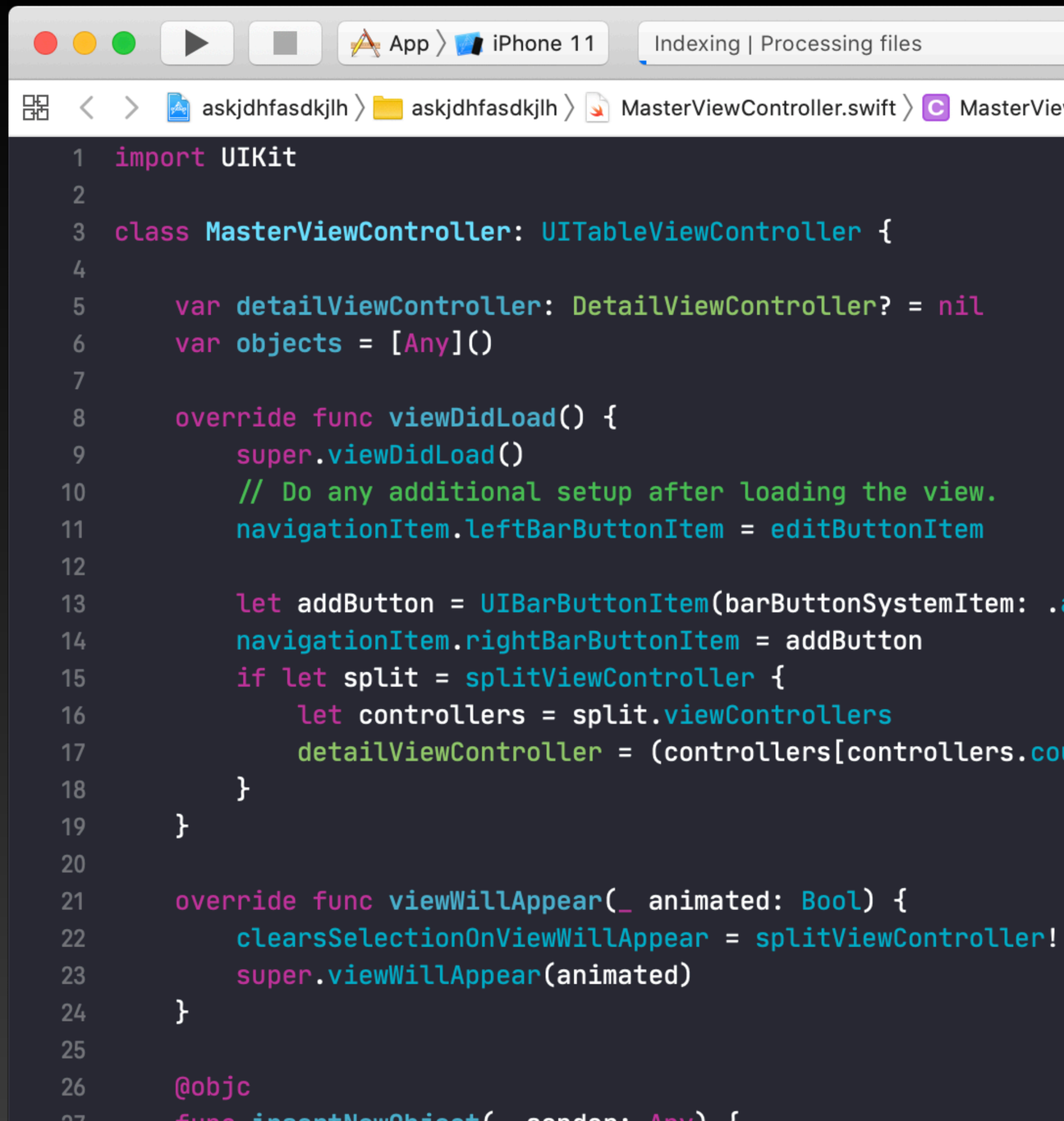
A Code is Data Talk

Dave Lee @kastiglione

IndexStore

Only Partially an Index

- Xcode Indexing…
- "Index"Store
- SymbolAttributeGraphStore



Code is Data

Implement this Function

```
class Project {
    func callers(of f: Function) -> [Function] {
        // …
    }
}
```

Code is Data
Available Options

- % grep | perl | sed | awk | ...
- Custom ad hoc scripts and tools
- SourceKit (SourceKitten)
- SwiftSyntax
- **IndexStore**

} Complimentary

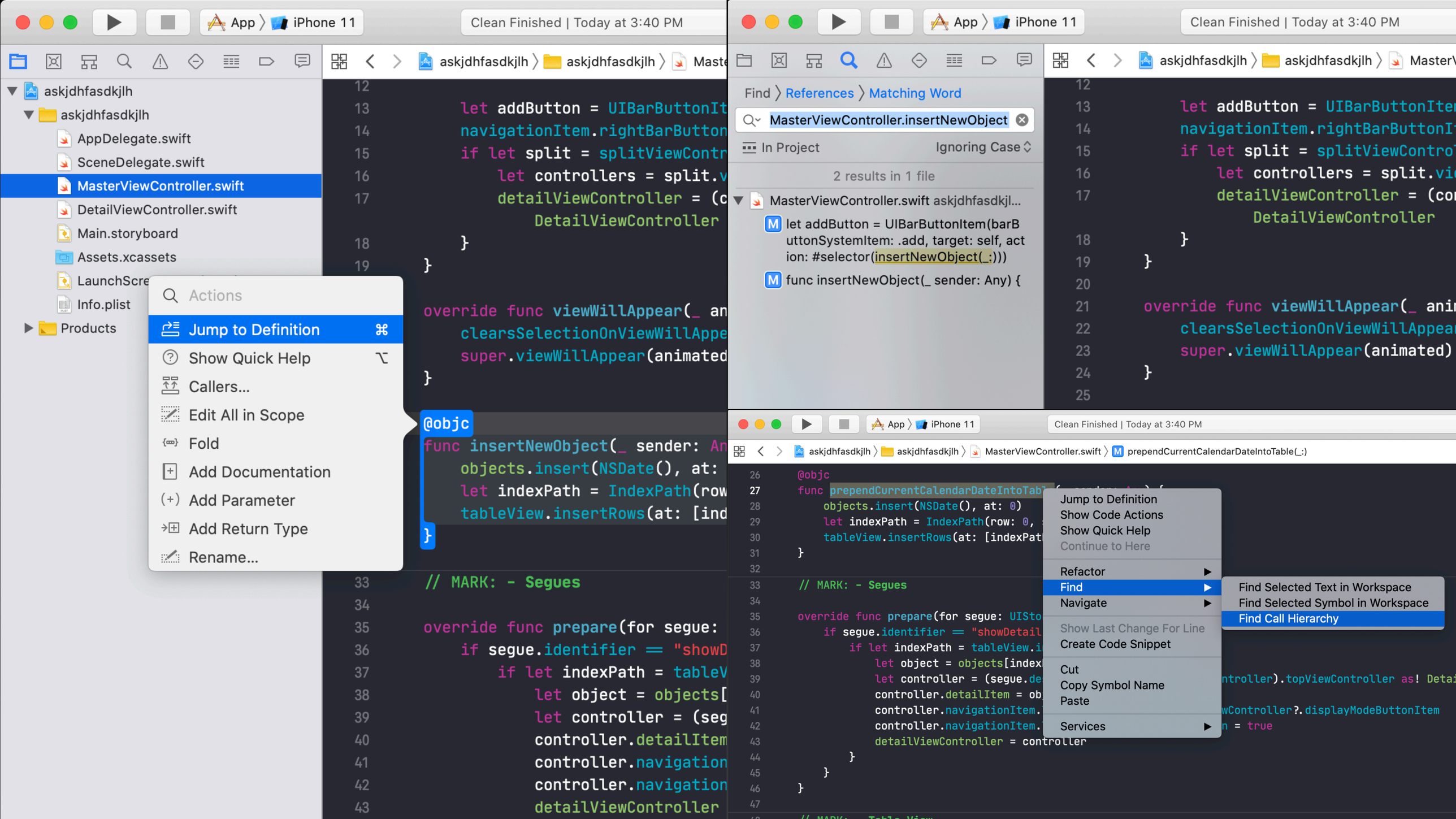
Code is Data

Available Options

	Syntax Aware	Scope	Representation	Relationships
Shell commands	🚫	¯_(ツ)_/¯	¯_(ツ)_/¯	¯_(ツ)_/¯
Ad hoc scripts	🚫	¯_(ツ)_/¯	¯_(ツ)_/¯	¯_(ツ)_/¯
SourceKit	✅	**token**	**json objects**	**explicit(?)**
SwiftSyntax	✅	**file**	**AST**	**derived**
IndexStore	✅	**project**	**~database**	**explicit**

“The best camera is the one with you”

IndexStore Crash Course


App 〉 iPhone 11
Clean Finished | Today at 3:40 PM
askjdhfasdkjlh
askjdhfasdkjlh
AppDelegate.swift
SceneDelegate.swift
MasterViewController.swift
DetailViewController.swift
Main.storyboard
Assets.xcassets
Info.plist
Products
Actions
Jump to Definition ⌘
Show Quick Help ⌥
Callers...
Edit All in Scope
Fold
Add Documentation
Add Parameter
Add Return Type
Rename...
Find 〉 References 〉 Matching Word
MasterViewController.insertNewObject
In Project
Ignoring Case
2 results in 1 file
MasterViewController.swift askjdhfasdkjl...
let addButton = UIBarButtonItem(barButtonSystemItem: .add, target: self, action: #selector(insertNewObject(_:)))
func insertNewObject(_ sender: Any) {
Jump to Definition
Show Code Actions
Show Quick Help
Continue to Here
Refactor
Find
Navigate
Show Last Change For Line
Create Code Snippet
Cut
Copy Symbol Name
Paste
Services
Find Selected Text in Workspace
Find Selected Symbol in Workspace
Find Call Hierarchy

IndexStore Crash Course

Xcode Indexing

- Background Indexing
- Index while Building

IndexStore Crash Course

Background Indexing

- ObjC: performed by Xcode, using libclang
- Swift: performed by swiftc

IndexStore Crash Course

Index while Building

- Costs roughly 5-15% (could be more in some cases)
- COMPILER_INDEX_STORE_ENABLE (applies to non-optimized builds)
- Both swiftc and clang support -index-store-path
  - Apple's clang only (currently)

IndexStore Crash Course

Logging

- defaults write com.apple.dt.Xcode IDEIndexShowLog YES

IndexStore Crash Course

Files

- DerivedData: Index/DataStore/v5
  - units
  - records
- Open source format
  - LLVM Bitstream (container format for bitcode, swiftmodules, etc)

IndexStore Crash Course

Record Files

- Record files contain data about source ***code***
  - Contains no paths
- Record file names contain a hash of its ***contents***
  - records/1J/AppDelegate.swift-2KRFZ3APSBU1J
- Powers many IDE features

IndexStore Crash Course

Unit Files

- Unit files contain data about source ***files***
  - IndexStore's book keeping
- Unit file names contain a hash of the ***object file path***
  - units/AppDelegate.o-1GHOD2MQ1C369
- Coupled to Xcode's build directory layout
- Unit files are the index into the… index
  - source file -> object file -> unit file -> record file

IndexStore Crash Course

Open Source

- Ability to create custom tools using IDE technology
- Java/Android IDEs don't have open source indexes
- Provides unique opportunities

IndexStore Crash Course
Xcode API

- XcodeDefault.xctoolchain/usr/lib/libIndexStore.dylib
  - Exports C functions, easily wrapped
  - Read only API
  - Treated as ABI
  - Xcode does not include its headers
- github.com/apple/llvm-project
  - clang/include/indexstore/indexstore.h

IndexStore Crash Course

IndexStore-DB

- IndexStore-Indexed
- In-memory database layer over IndexStore
  - Swift API
  - Read only API
  - SPM enabled
  - Best interface for long lived clients, like LSP
- github.com/apple/indexstore-db

IndexStore Crash Course
Clang API

- Reference implementation
  - C++ API
  - Read-write API
  - CMake enabled
- Use it for writing index files
  - github.com/lyft/index-import

IndexStore Data Model

IndexStore Has No Docs

person

person, woman

person, woman, man

person, woman, man, camera

person, woman, man, camera, tv

symbol

symbol, kind

symbol, kind, occurrence

symbol, kind, occurrence, role

symbol, kind, occurrence, role, relation

What You're About To See
Is not Completely Real

IndexStore Data Model

Symbol

```
struct Symbol {
    var usr: String
    var humanName: String
    var moduleName: String
    var kind: SymbolKind
    var subkind: SymbolSubkind
    var properties: [Property]
}
```

USR

Unified Symbol Resolution: a string that globally identifies a symbol within a ***program***

USRs
Swift vs ObjC

- Swift USRs are Swift mangled symbols
  - s:10Foundation9IndexPathV5UIKitE3rowSivp
- ObjC USRs are custom in clang
  - c:objc(cs)UIView
  - c:objc(cs)UIView(im)addSubview:

IndexStore Data Model

Kinds of Symbols

```
enum SymbolKind {
    case `enum`, `struct`, `class`, `protocol`
    case function
    case variable
    case instanceMethod
    case instanceProperty
    // many more…
}
```

IndexStore Data Model

Symbol Occurrence

```
struct SymbolOccurrence {
    var symbol: Symbol
    var roles: Set<SymbolRole>
    var sourceFile: String
    var location: (line: Int, column: Int)
    var relations: [SymbolRelation]
}
```

IndexStore Data Model

Uses of Symbols

```
emum SymbolRole {
    case definition
    case reference
    case read
    case write
    case call
    // more…
}
```

IndexStore Data Model

Symbol

```
struct SymbolRelation {
    var relation: SymbolRole // Role's second use
    var relatedSymbol: Symbol
}
```

IndexStore Data Model

Uses of Symbols

```
enum SymbolRole {
     case childOf
    case baseOf
    case overrideOf
    case receivedBy
    case calledBy
    case extendedBy
    case containedBy
   // more…
}
```

IndexStore Data Model
Missing Piece

- Swift Demangler
  - Available via Swift C++ API
- ObjC USR parser
  - No public implementations

IndexStore Use Cases

IndexStore Use Cases

Leverage Points

- Symbol kind and role
  - Focused searches
- Symbol (and file) relationships
  - Type hierarchy
  - Function call graph
  - Container/context aware searches
- Flatten abstractions, indirect relationships

IndexStore Use Cases

Categories

- Export IndexStore to other formats
- Ad hoc developer queries
- Purpose built tools for searches/queries
  - Augment Xcode symbol search
- Static analysis
- Visualizations
- Refactoring

IndexStore Use Cases

Export Data

- IndexStore's data model can be mapped to a SQLite schema
  - Prolog is a really nice alternative
- IndexStore's relationships can be exported into directed graphs
  - Analyze using graph libraries
- Source code annotations

IndexStore Use Cases

Ad hoc Queries

- The IndexStore data model can be mapped to database schemas
  - SQL, NoSQL, Graph
- nshipster.com/as-we-may-code/
  - SPAQRQL, Cypher

IndexStore Use Cases

nshipster.com/as-we-may-code/

Answering Questions About Your Code

"What can you do with a knowledge graph?" That's kind of like asking, *"What can you do with Swift?"* The answer — *"Pretty much anything"* — is as true as it is unhelpful.

Perhaps a better framing would be to consider the kinds of questions that a knowledge graph of code symbols can help answer:

- Which methods in Foundation produce a `Date` value?
- Which public types in my project *don't* conform to `Codable`?
- Which methods does `Array` inherit default implementations from `RandomAccessCollection`?
- Which APIs have documentation that includes example code?
- What are the most important APIs in `MapKit`?
- Are there any unused APIs in my project?
- What's the oldest version of iOS that my app could target based on my current API usage?
- What APIs were added to Alamofire between versions 4.0 and 4.2?
- What APIs in our app are affected by a CVE issued for a 3rd-party dependency?

IndexStore Use Cases

Ad hoc Queries

```
with recursive views (symbol) as (
    select symbol from symbols where name = 'UIView'
    UNION
    select relations.related_symbol
    from relations join views using (symbol)
    where relations.relation_role = 'baseOf'
)
select symbols.name, occurrences.file
from symbols join occurrences using (symbol)
where symbol in views and occurrences.role = 'definition'
```

IndexStore Use Cases

Developer Tools

- Supplement Xcode's feature set
  - Search symbol references by caller (ex viewDidLoad)
- Show more info about types
  - List all ancestor classes, protocols
  - List all descendent types
  - Display full API of a type including all inherited functions etc

IndexStore Use Cases

Static Analysis

- Validate xib/storyboard actions and outlets
- Identify unused code, unused imports
- Catch unwanted code patterns
- API diffs
  - For releases or code review (requires extra CI builds)

IndexStore Use Cases

Visualizations

- Graph any of the relationships using graphviz, d3, gephi, etc
  - Project type hierarchy
  - Function call graph
- Visualize relationship between source files
  - dependencies, cycles, or clusters
  - Aid understanding of unfamiliar code

IndexStore Use Cases

Refactoring

- Combine with SwiftSyntax
  - or whatever works
- Automate Xcode refactorings
- Delete unused code
  - github.com/woshiccm/Pecker
- Reorder source file contents to some standard
  - github.com/krzyzanowskim/reorder

Limitations

IndexStore Limitations

it has some

- Designed for Xcode's use cases
- USR parsing/demangling batteries not included
- Refactoring/editing requires other libraries
- Symbol occurrences include only the start location
- No local variable info
- No access control levels
- more...

so, what comes to mind?

thank you