CocoaHeads Aachen November 2017

Build Your Own Swift Refactoring Action

Refactoring ≠ Refactoring

- Syntactic changes
  - Indentation, Code formatting, Renaming
- Semantic changes
  - Extract method , for-loop ↔ functional style
- Code generation
  - Localise string, add missing case statements

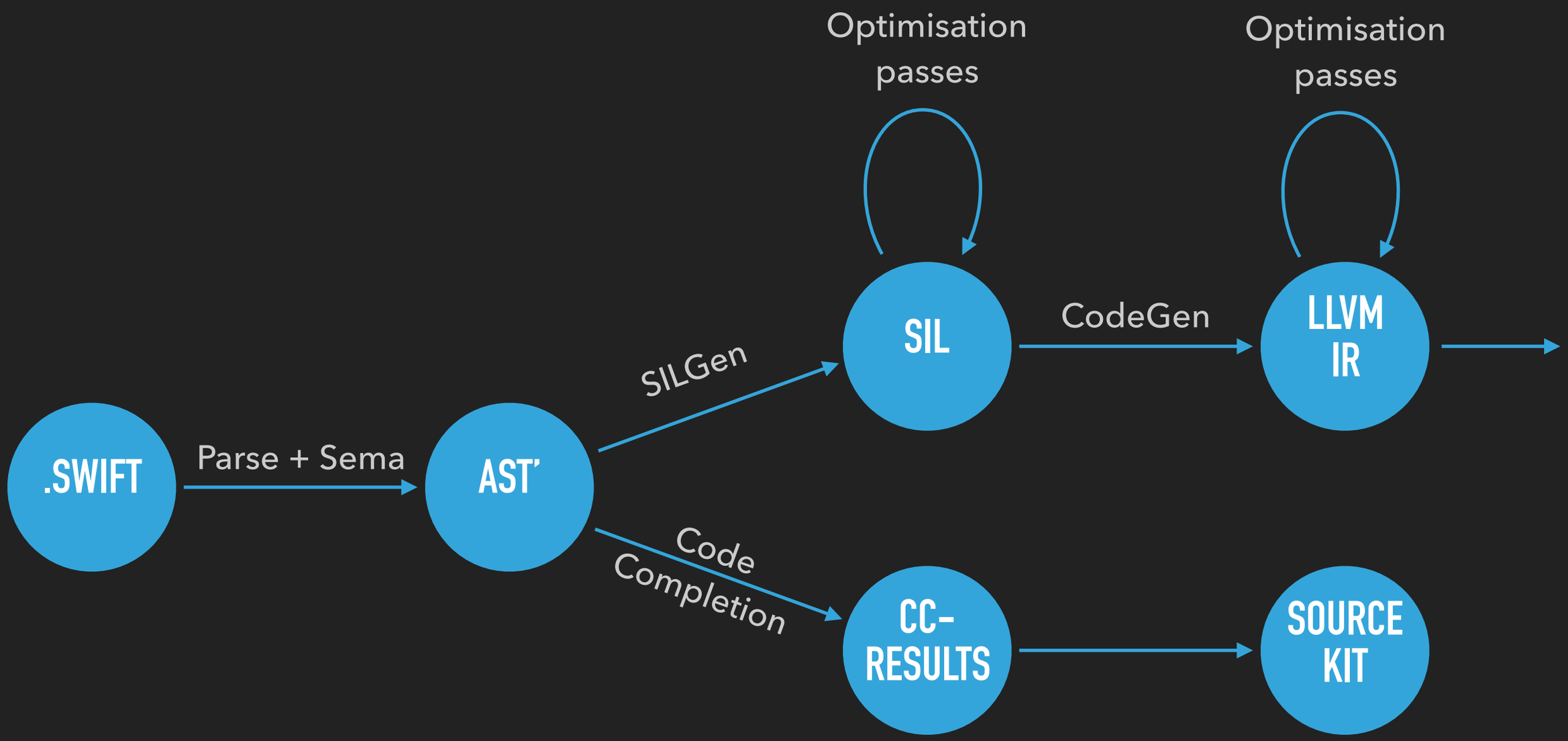
The Toolbox

- Compiler-based refactoring
- SourceKit
- libSyntax
- ~~Search & Replace + Regular Expressions~~

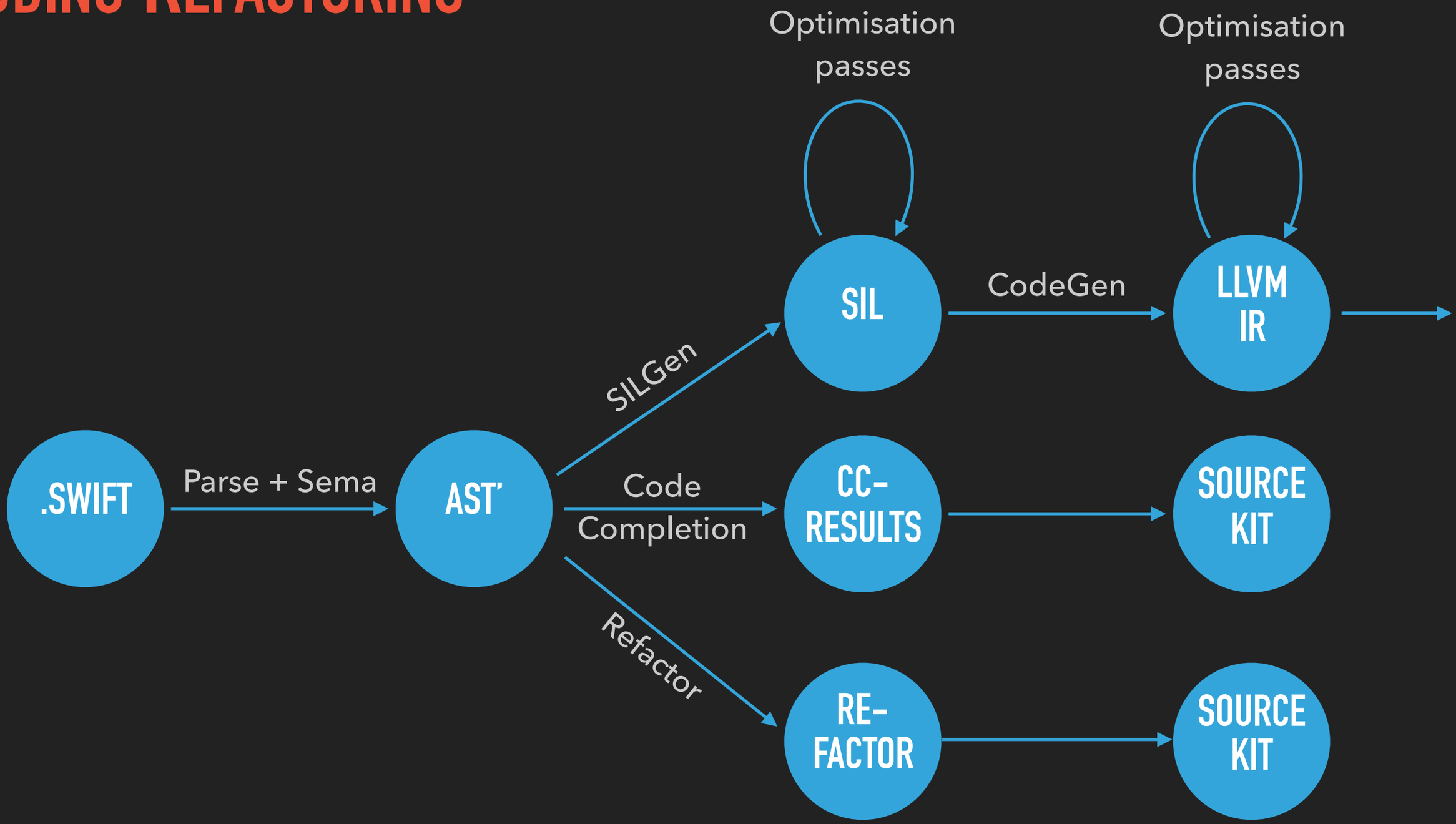
The Full Power at Your Fingertips

Compiler-Based Refactoring

Recap: Stages of the Swift compiler



ADDING REFACTORING



Compiler-Integrated Refactoring

- Integrated into the Swift compiler
- Basis of operation: Abstract syntax tree (AST)
  - Abstracts away from source formatting
  - Symbols and overloads resolved
- Access: within the compiler in C++
- AST designed for compiling, not refactoring
- Result: Shipped with next version of the compiler / Xcode

Refactoring Driver

- Parses Swift file
- Parses selection: Selected expression, referenced decls, …
- Determines applicable refactoring actions
- Invokes refactoring action

Compiler Refactoring Actions

Demo

Demo – Content

- Refactoring Action to move members of a type to a new extension
- Pull Request to integrate the action into the compiler available on GitHub (Link)
  - Interesting changes in Refactoring.cpp (Link)
- Toolchain with this refactoring action can be downloaded here
  - Install instructions can be found here

Some Notes

- AST not primarily designed to perform refactoring
  - Some information lost or not easily accessible
    - “Trivial” operations like “get parent node” not possible
- However: Basically all semantic information available

Taming The Compiler

SourceKit

SourceKit

- External access to compiler data
  - Code completion, syntax highlighting, structure, …
- Official way Xcode communicates with the compiler
- Action to output code structure

sourcekitten structure --file myFile.swift

Demo

The Lightweight Option

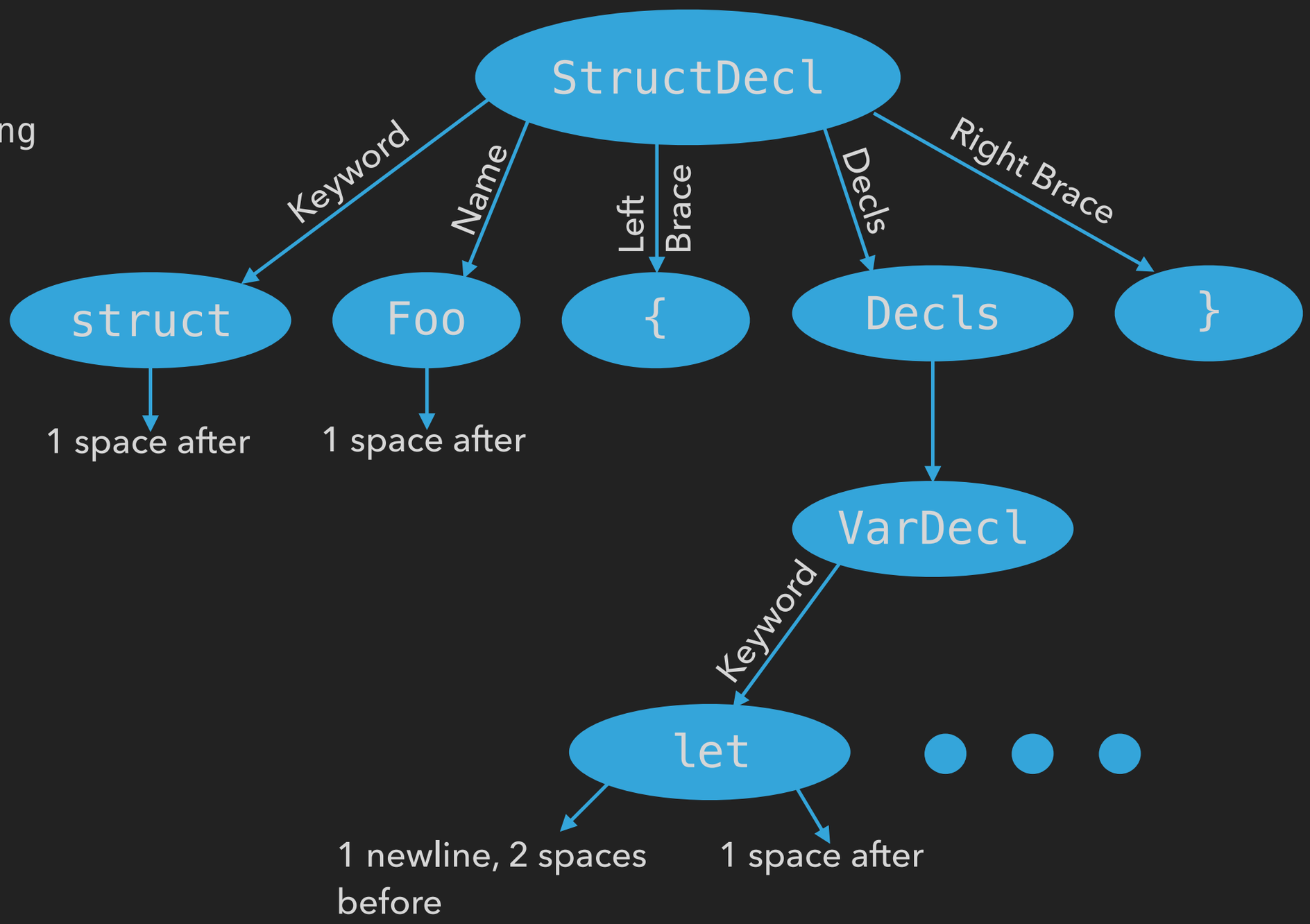
LibSyntax

LIBSYNTAX

- Framework
- Basis of operation: Source-preserving syntax tree
  - Focus on “trivia” (spaces, newlines, comments, …)
  - Symbols not resolved
- Access: via Swift API
- Still very pre-alpha-state
- Result: Custom Tool

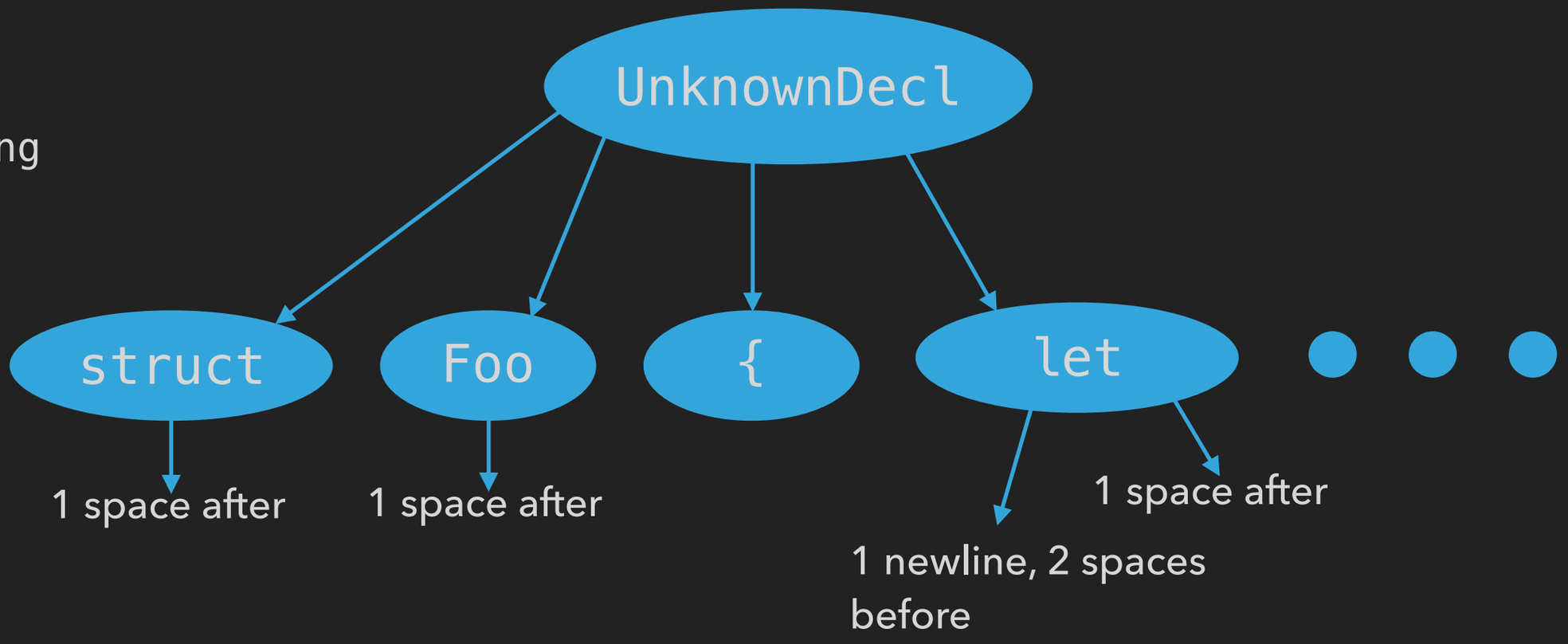
libSyntax – How It Should Be

```
struct Foo {
  let bar: String
}
```



libSyntax – The Reality

```
struct Foo {
  let bar: String
}
```



Indentation Fixer With SwiftSyntax

Demo

SwiftSyntax – Build Instructions

- Download Swift Toolchain from swift.org
- Add in Build Settings
  - Library Search Path: "/Library/Developer/Toolchains/swift-DEVELOPMENT-XXXX-XX-XX-a.xctoolchain/usr/lib/swift/macosx"
  - Runpath Search Path: "/Library/Developer/Toolchains/swift-DEVELOPMENT-XXXX-XX-XX-a.xctoolchain/usr/lib/swift/macosx"

Thank You

Alex Hoppen

@alex_hoppen