Swift Explicitly-Built Modules

Artem Chikin

LLVM Dev Meeting 2024

Modules in Swift

By-design, Swift programs are composed of a number of modules which represent units of code distribution and correspond to binary products.

SDelegate.swift

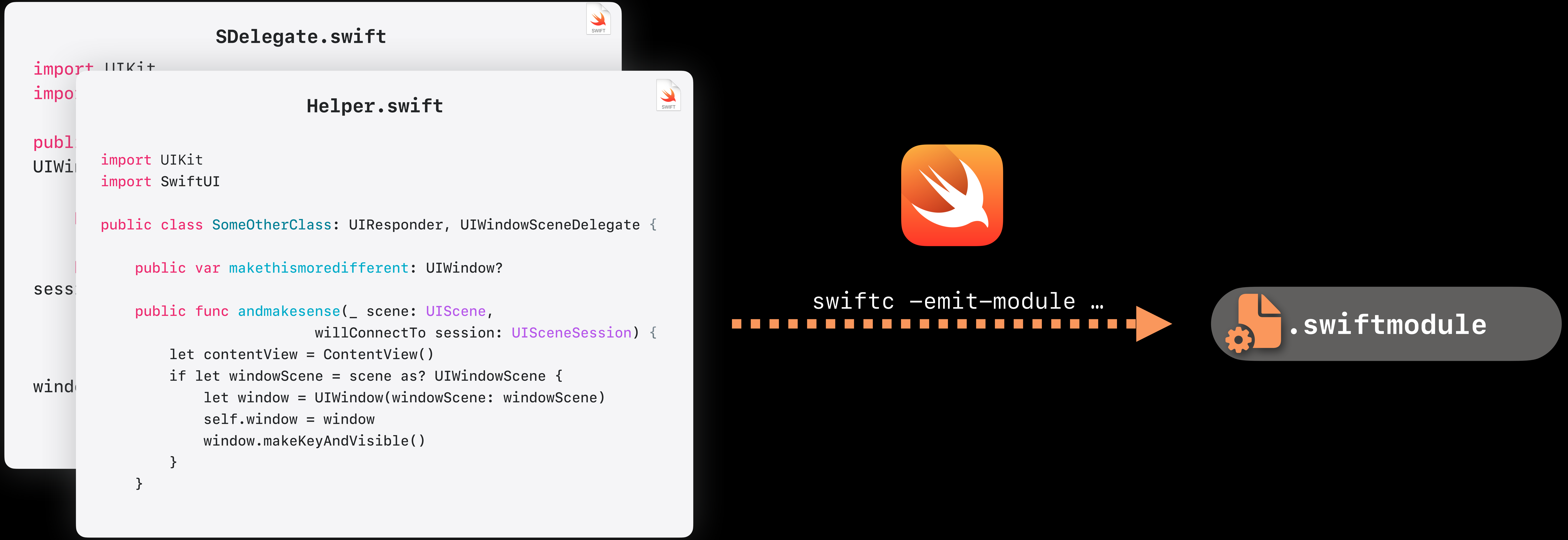
```swift
import UIKit
import SwiftUI

public class SceneDelegate: UIResponder, UIWindowSceneDelegate {

    public var window: UIWindow?

    public func scene(_ scene: UIScene,
                      willConnectTo session: UISceneSession) {
        let contentView = ContentView()
        if let windowScene = scene as? UIWindowScene {
            let window = UIWindow(windowScene: windowScene)
            self.window = window
            window.makeKeyAndVisible()
        }
    }
}
```

Helper.swift

```swift
import UIKit
import SwiftUI

public class SomeOtherClass: UIResponder, UIWindowSceneDelegate {

    public var makethismoredifferent: UIWindow?

    public func andmakesense(_ scene: UIScene,
                             willConnectTo session: UISceneSession) {
        let contentView = ContentView()
        if let windowScene = scene as? UIWindowScene {
            let window = UIWindow(windowScene: windowScene)
            self.window = window
            window.makeKeyAndVisible()
        }
    }
}
```

Modules in Swift

By-design, Swift programs are composed of a number of modules which represent units of code distribution and correspond to binary products.



SDelegate.swift

```
import UIKit
impo

publ
UIWi


sess


wind
```

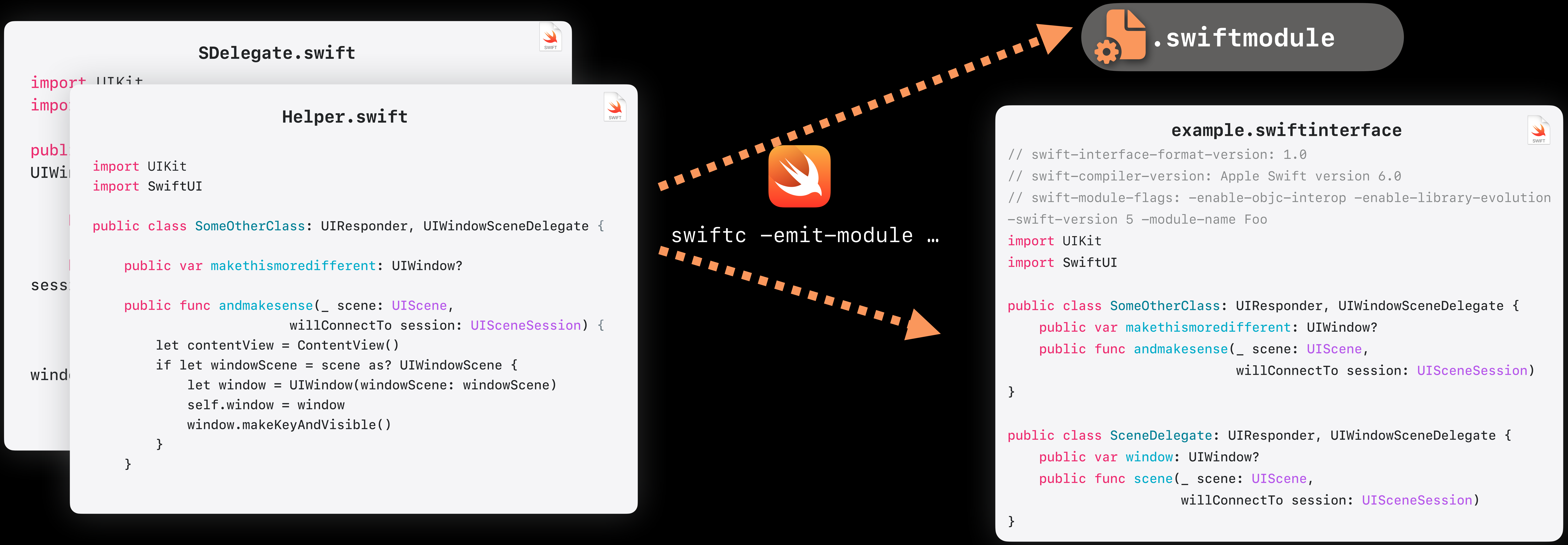
Helper.swift

```
import UIKit
import SwiftUI

public class SomeOtherClass: UIResponder, UIWindowSceneDelegate {

    public var makethismoredifferent: UIWindow?

    public func andmakesense(_ scene: UIScene,
                             willConnectTo session: UISceneSession) {
        let contentView = ContentView()
        if let windowScene = scene as? UIWindowScene {
            let window = UIWindow(windowScene: windowScene)
            self.window = window
            window.makeKeyAndVisible()
        }
    }
```

```
swiftc -emit-module …
```

Modules in Swift

By-design, Swift programs are composed of a number of modules which represent units of code distribution and correspond to binary products.



SDelegate.swift

```
import UIKit
impo

publ
UIWi

sess

wind
```

Helper.swift

```
import UIKit
import SwiftUI

public class SomeOtherClass: UIResponder, UIWindowSceneDelegate {

    public var makethismoredifferent: UIWindow?

    public func andmakesense(_ scene: UIScene,
                             willConnectTo session: UISceneSession) {
        let contentView = ContentView()
        if let windowScene = scene as? UIWindowScene {
            let window = UIWindow(windowScene: windowScene)
            self.window = window
            window.makeKeyAndVisible()
        }
    }
```

```
swiftc -emit-module …
```

example.swiftinterface

```
// swift-interface-format-version: 1.0
// swift-compiler-version: Apple Swift version 6.0
// swift-module-flags: -enable-objc-interop -enable-library-evolution
-swift-version 5 -module-name Foo
import UIKit
import SwiftUI

public class SomeOtherClass: UIResponder, UIWindowSceneDelegate {
    public var makethismoredifferent: UIWindow?
    public func andmakesense(_ scene: UIScene,
                             willConnectTo session: UISceneSession)
}

public class SceneDelegate: UIResponder, UIWindowSceneDelegate {
    public var window: UIWindow?
    public func scene(_ scene: UIScene,
                      willConnectTo session: UISceneSession)
}
```

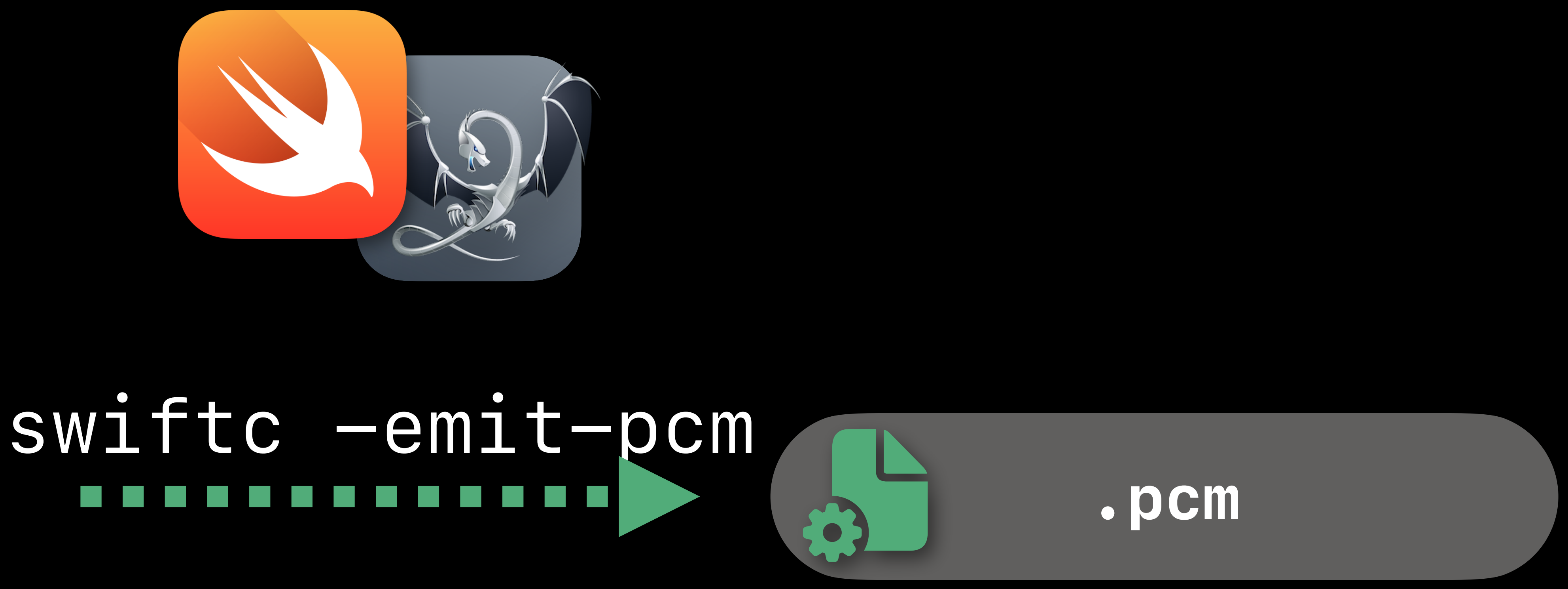
C/Obj-C/C++ Modules in Swift

Swift directly interoperates with C/Objective-C/C++ code. Doing so at scale requires modularized header interfaces.

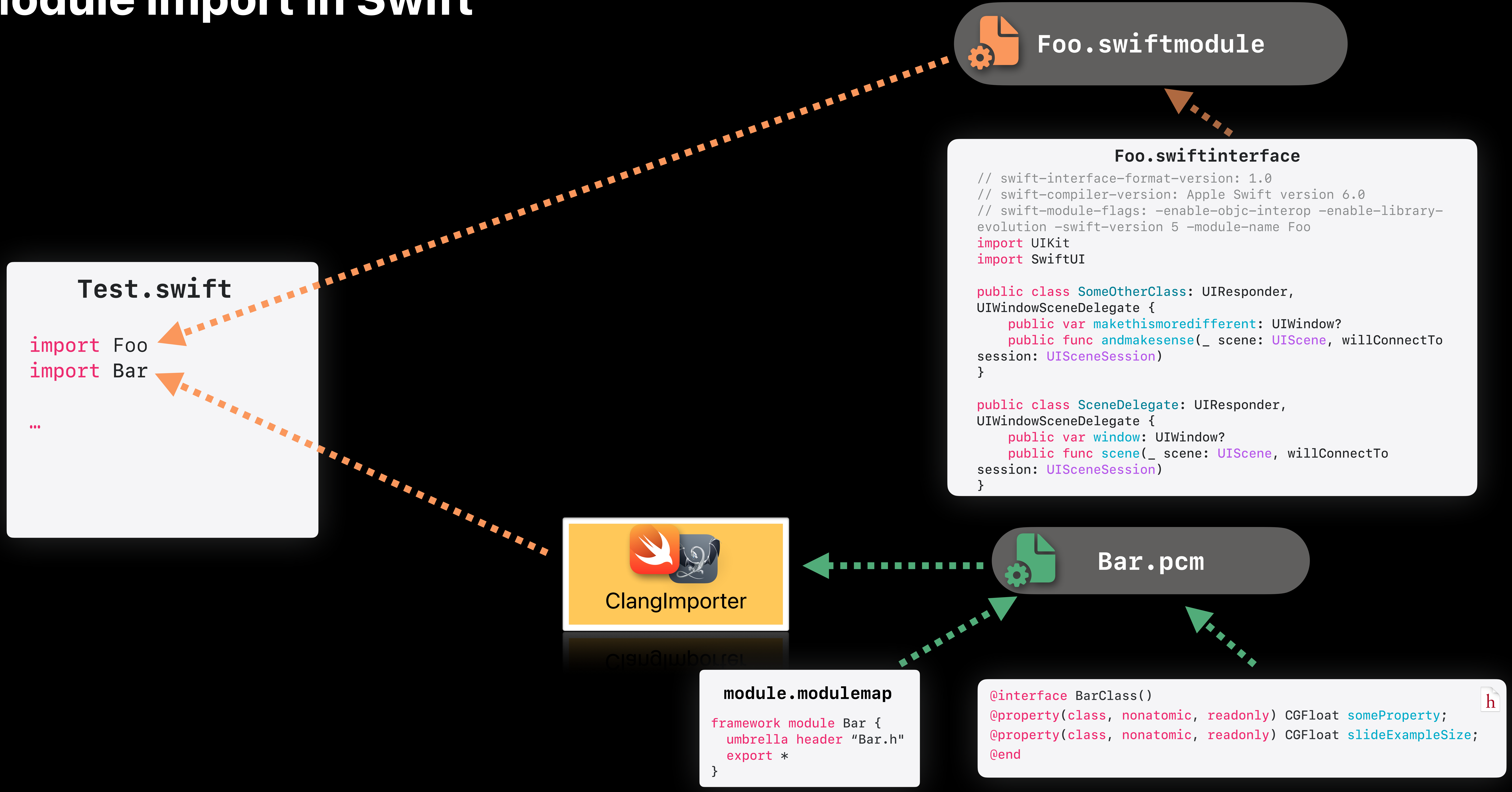
module.modulemap

```
framework module UIKit {
  umbrella header "UIKit.h"
  export *
}
```

```
@interface UIFont (UIFontSystemFonts)
@property(class, nonatomic, readonly) CGFloat labelFontSize;
@property(class, nonatomic, readonly) CGFloat buttonFontSize;
@property(class, nonatomic, readonly) CGFloat smallSystemFontSize;
@property(class, nonatomic, readonly) CGFloat systemFontSize;
@property(class, nonatomic, readonly) CGFloat defaultFontSize;
@property(class, nonatomic, readonly) CGFloat systemMinimumFontSize;
@end
```



Module import in Swift



Test.swift

```
import Foo
import Bar

…
```

Foo.swiftinterface

```
// swift-interface-format-version: 1.0
// swift-compiler-version: Apple Swift version 6.0
// swift-module-flags: -enable-objc-interop -enable-library-
evolution -swift-version 5 -module-name Foo
import UIKit
import SwiftUI

public class SomeOtherClass: UIResponder,
UIWindowSceneDelegate {
    public var makethismoredifferent: UIWindow?
    public func andmakesense(_ scene: UIScene, willConnectTo
session: UISceneSession)
}

public class SceneDelegate: UIResponder,
UIWindowSceneDelegate {
    public var window: UIWindow?
    public func scene(_ scene: UIScene, willConnectTo
session: UISceneSession)
}
```

module.modulemap

```
framework module Bar {
  umbrella header "Bar.h"
  export *
}
```

```
@interface BarClass()
@property(class, nonatomic, readonly) CGFloat someProperty;
@property(class, nonatomic, readonly) CGFloat slideExampleSize;
@end
```

Module Resolution in Swift

Each imported named module may require compilation into its associated binary product consumable by the client compiler.

Module Resolution in Swift

Implicit discovery and compilation

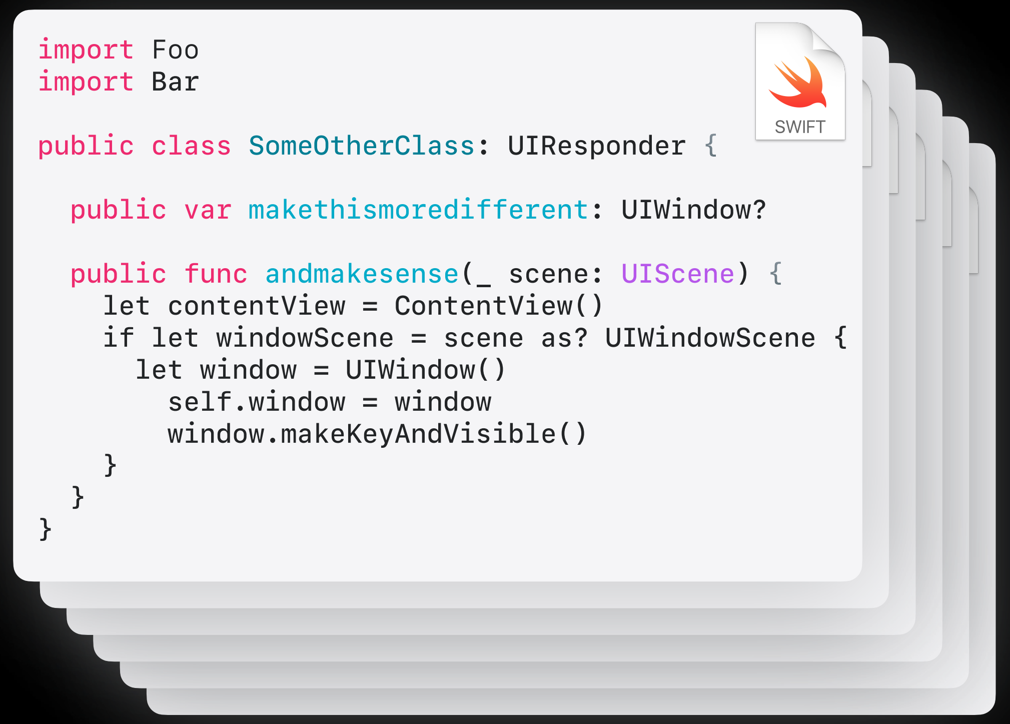
Each imported named module may require compilation into its associated binary product consumable by the client compiler.

```swift
import Foo
import Bar

public class SomeOtherClass: UIResponder {

  public var makethismoredifferent: UIWindow?

  public func andmakesense(_ scene: UIScene) {
    let contentView = ContentView()
    if let windowScene = scene as? UIWindowScene {
      let window = UIWindow()
        self.window = window
        window.makeKeyAndVisible()
    }
  }
}
```

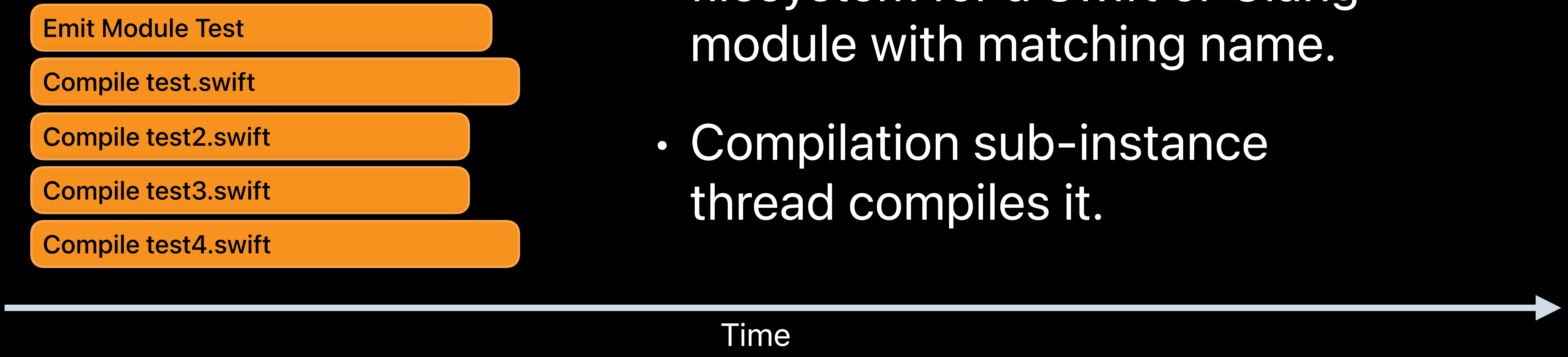
Module Resolution in Swift

Implicit discovery and compilation

Each imported named module may require compilation into its associated binary product consumable by the client compiler.



```swift
import Foo
import Bar

public class SomeOtherClass: UIResponder {

  public var makethismoredifferent: UIWindow?

  public func andmakesense(_ scene: UIScene) {
    let contentView = ContentView()
    if let windowScene = scene as? UIWindowScene {
      let window = UIWindow()
        self.window = window
        window.makeKeyAndVisible()
    }
  }
}
```

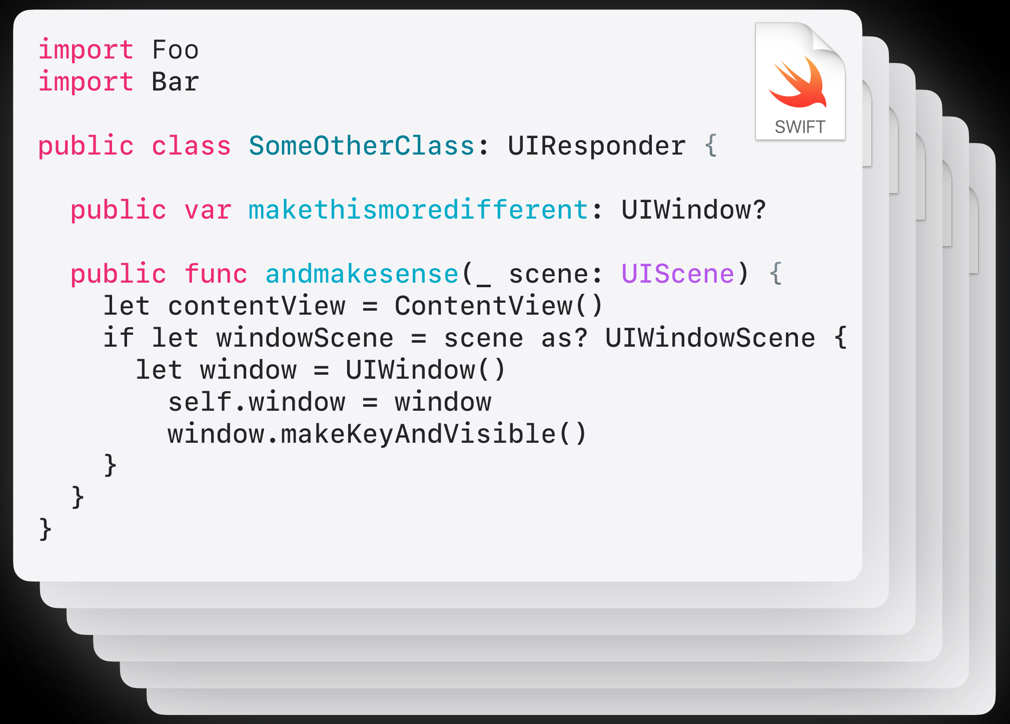


- Compiler searches the filesystem for a Swift or Clang module with matching name.
- Compilation sub-instance thread compiles it.

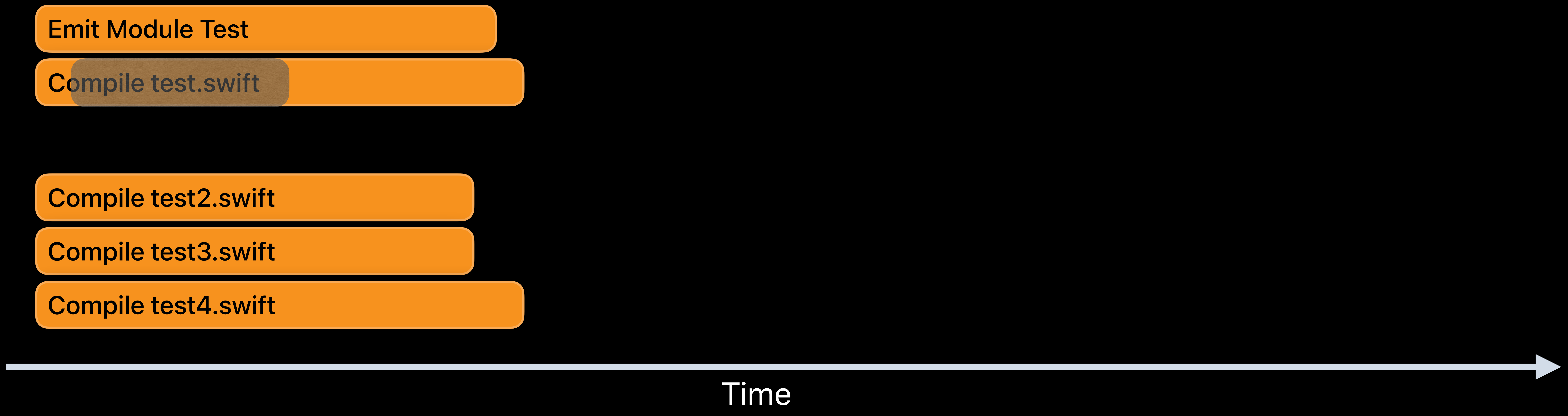
Module Resolution in Swift

Implicit discovery and compilation

Each imported named module may require compilation into its associated binary product consumable by the client compiler.



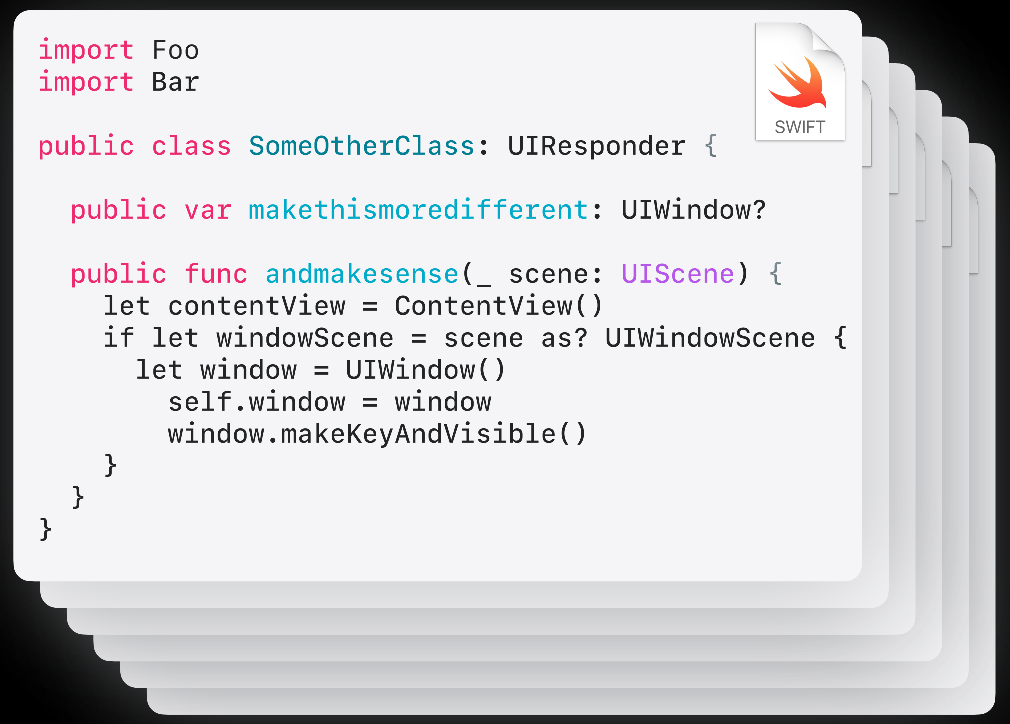
```swift
import Foo
import Bar

public class SomeOtherClass: UIResponder {

  public var makethismoredifferent: UIWindow?

  public func andmakesense(_ scene: UIScene) {
    let contentView = ContentView()
    if let windowScene = scene as? UIWindowScene {
      let window = UIWindow()
        self.window = window
        window.makeKeyAndVisible()
    }
  }
}
```



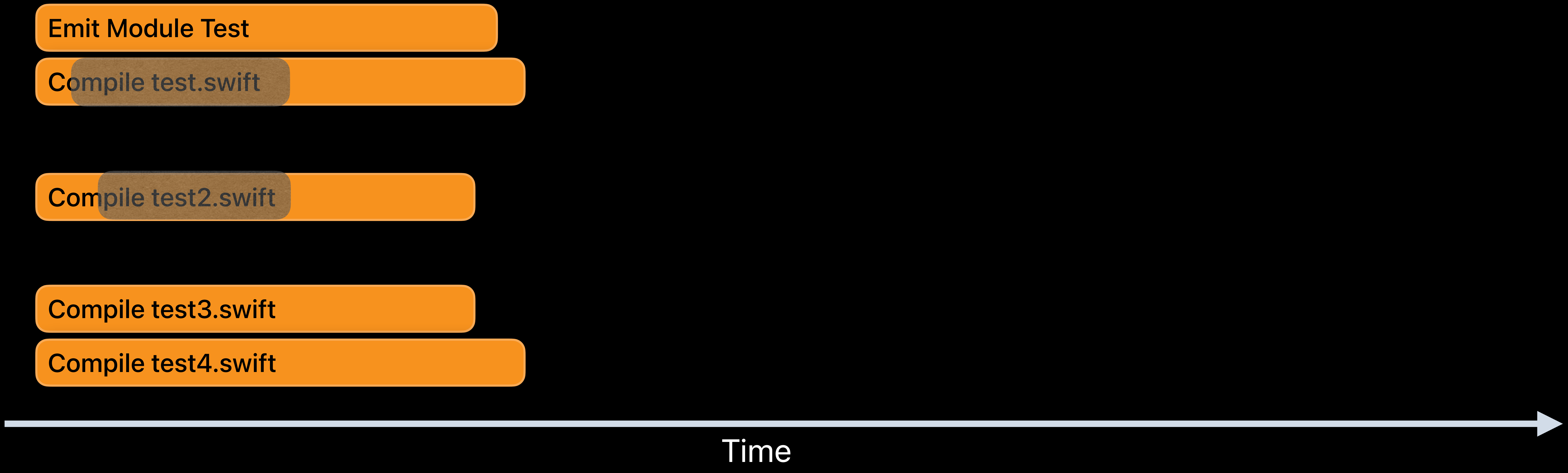
Module Resolution in Swift

Implicit discovery and compilation

Each imported named module may require compilation into its associated binary product consumable by the client compiler.



```
import Foo
import Bar

public class SomeOtherClass: UIResponder {

  public var makethismoredifferent: UIWindow?

  public func andmakesense(_ scene: UIScene) {
    let contentView = ContentView()
    if let windowScene = scene as? UIWindowScene {
      let window = UIWindow()
        self.window = window
        window.makeKeyAndVisible()
    }
  }
}
```



Module Resolution in Swift

Implicit discovery and compilation

Each imported named module may require compilation into its associated binary product consumable by the client compiler.

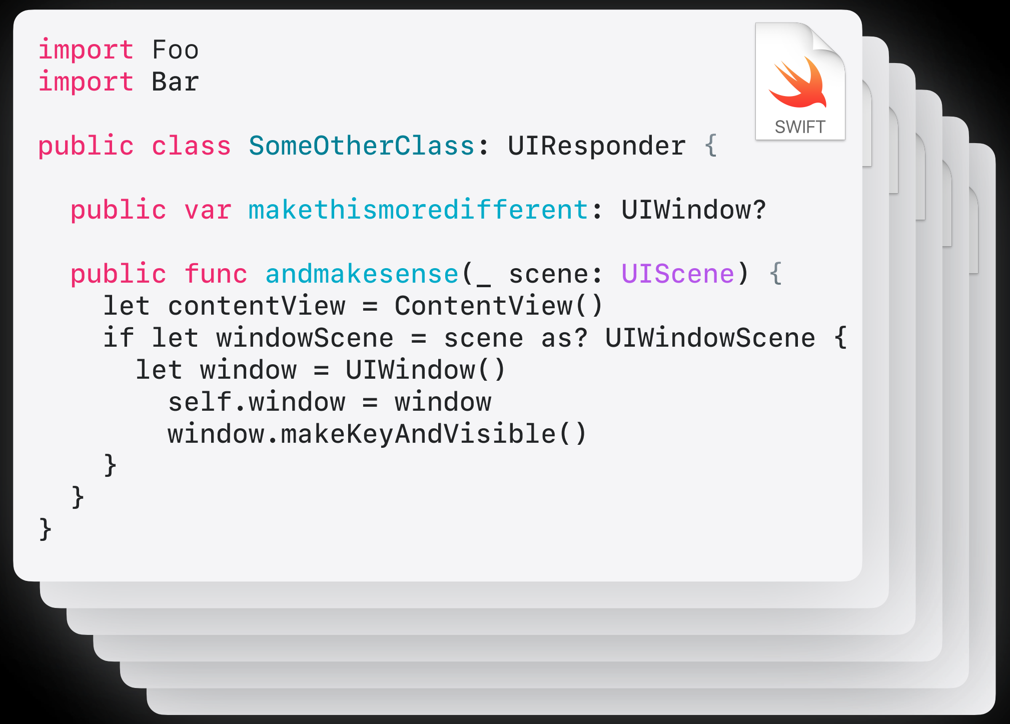
SWIFT
Emit Module Test
Compile test.swift
Compile test2.swift
Compile test3.swift
Compile test4.swift
Time

```swift
import Foo
import Bar

public class SomeOtherClass: UIResponder {

  public var makethismoredifferent: UIWindow?

  public func andmakesense(_ scene: UIScene) {
    let contentView = ContentView()
    if let windowScene = scene as? UIWindowScene {
      let window = UIWindow()
        self.window = window
        window.makeKeyAndVisible()
    }
  }
}
```

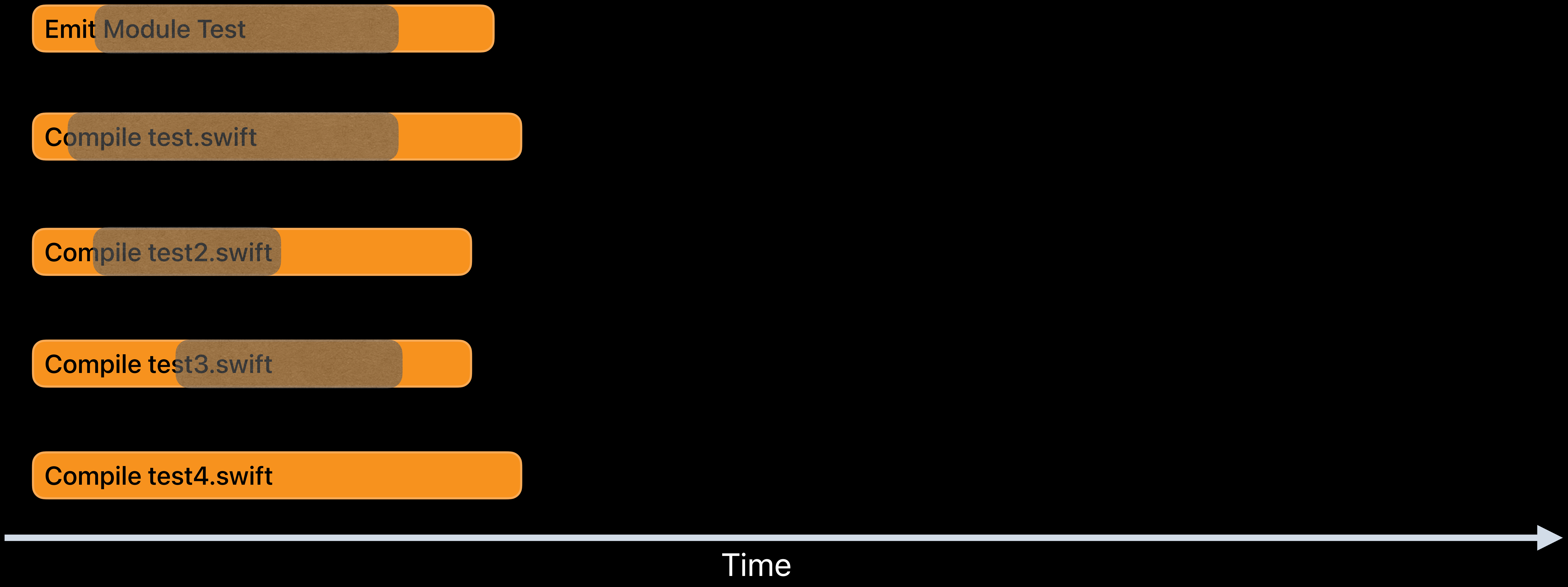
Module Resolution in Swift

Implicit discovery and compilation

Each imported named module may require compilation into its associated binary product consumable by the client compiler.



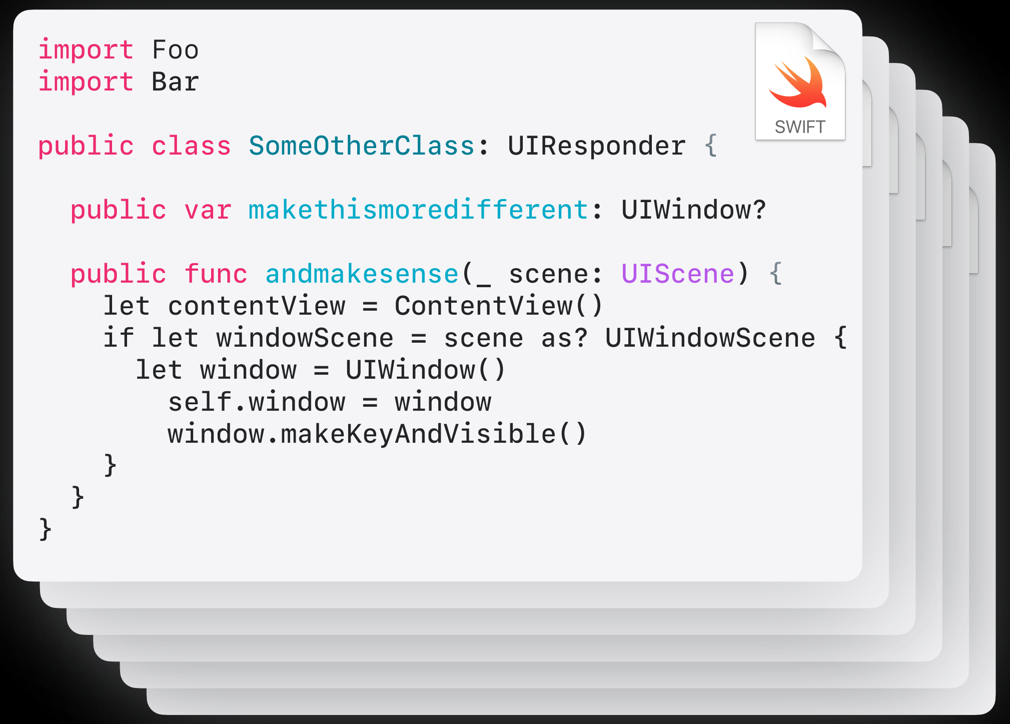
```
import Foo
import Bar

public class SomeOtherClass: UIResponder {

  public var makethismoredifferent: UIWindow?

  public func andmakesense(_ scene: UIScene) {
    let contentView = ContentView()
    if let windowScene = scene as? UIWindowScene {
      let window = UIWindow()
        self.window = window
        window.makeKeyAndVisible()
    }
  }
}
```



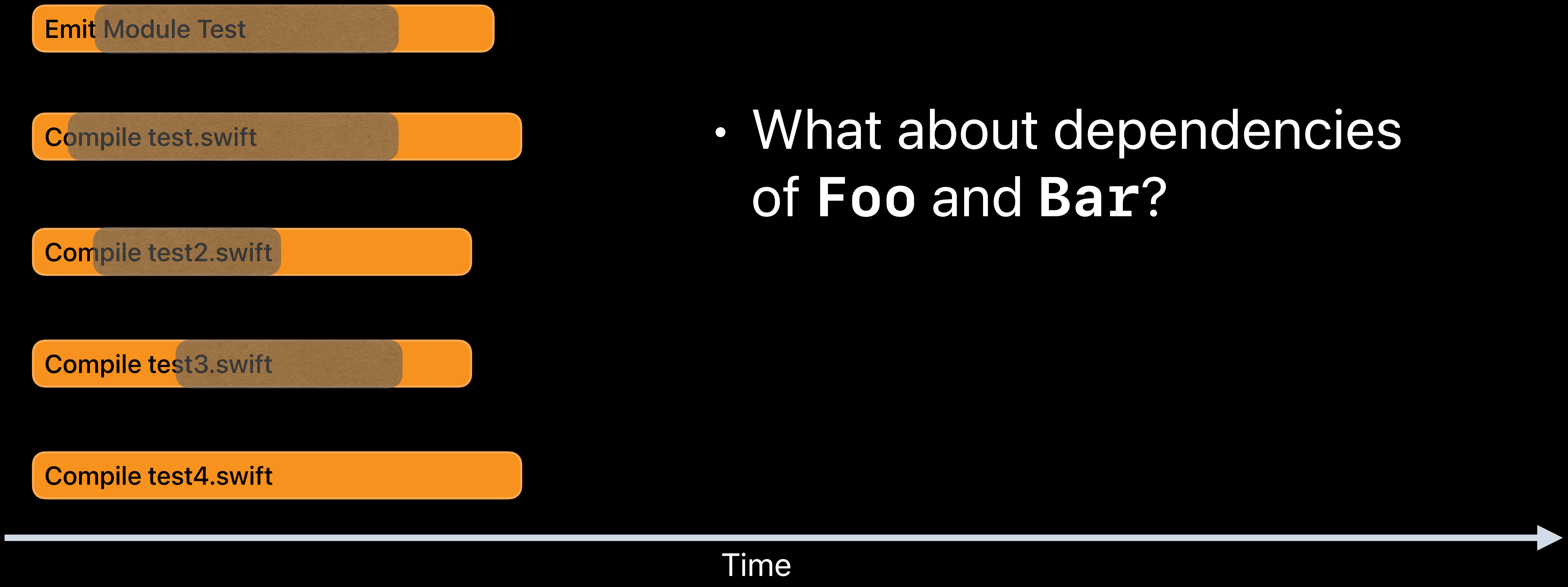
Module Resolution in Swift

Implicit discovery and compilation

Each imported named module may require compilation into its associated binary product consumable by the client compiler.



```swift
import Foo
import Bar

public class SomeOtherClass: UIResponder {

  public var makethismoredifferent: UIWindow?

  public func andmakesense(_ scene: UIScene) {
    let contentView = ContentView()
    if let windowScene = scene as? UIWindowScene {
      let window = UIWindow()
        self.window = window
        window.makeKeyAndVisible()
    }
  }
}
```



- What about dependencies of **`Foo`** and **`Bar`**?

Module Resolution in Swift

Implicit discovery and compilation

Each imported named module may require compilation into its associated binary product consumable by the client compiler.

Emit Module Test
Compile test.swift
Compile test2.swift
Compile test3.swift
Compile test4.swift
SWIFT
Time

```swift
import Foo
import Bar

public class SomeOtherClass: UIResponder {

  public var makethismoredifferent: UIWindow?

  public func andmakesense(_ scene: UIScene) {
    let contentView = ContentView()
    if let windowScene = scene as? UIWindowScene {
      let window = UIWindow()
        self.window = window
        window.makeKeyAndVisible()
    }
  }
}
```

- What about dependencies of **`Foo`** and **`Bar`**?

Sub-instance thread creation for dependency compilation is recursive.

Module Resolution in Swift

Implicit discovery and compilation

Emit Module Test
Compile test.swift
Compile test2.swift
Compile test3.swift
Compile test4.swift
Time
SWIFT

```
import Foo
import Bar

public class SomeOtherClass: UIResponder {

  public var makethismoredifferent: UIWindow?

  public func andmakesense(_ scene: UIScene) {
    let contentView = ContentView()
    if let windowScene = scene as? UIWindowScene {
      let window = UIWindow()
        self.window = window
        window.makeKeyAndVisible()
    }
  }
}
```

Opaque to the Build System

Peer task waiting with no forward progress

Non-isolated compilation tasks

Nested compilation context hard to reason about

Filesystem locking

Late error discovery

Module Resolution in Swift

Explicitly Built Modules

Clang 💙 Explicit Modules **Swift ❤️ Explicit Modules**

[1] *J. Svoboda: Implicitly discovered, explicitly built Clang modules (EuroLLVM 2022)*

Module Resolution in Swift

Explicitly Built Modules

Dependency Scan

Build Modules

Build Source

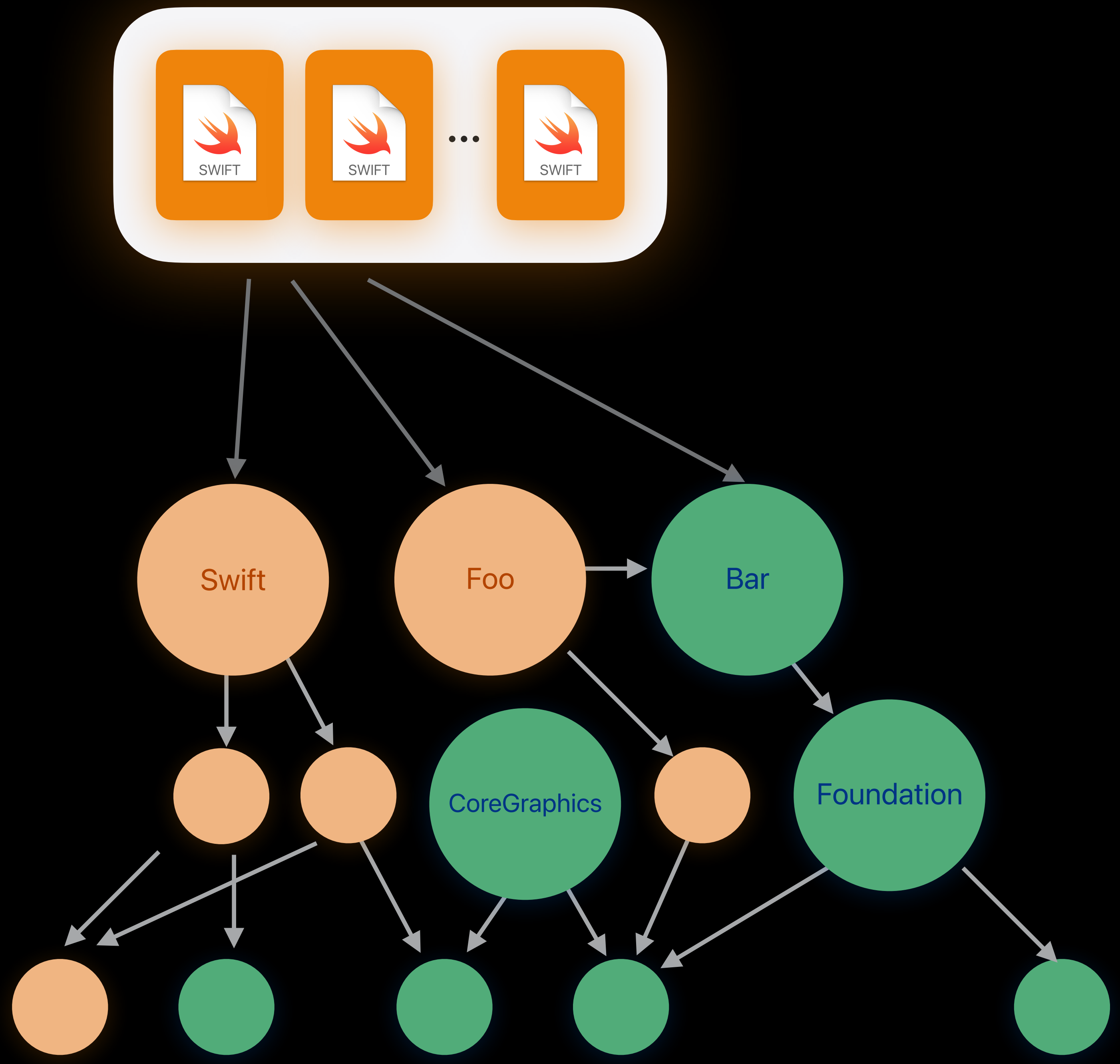
Module Resolution in Swift

Explicitly Built Modules

Dependency Scan

Build Modules

Build Source



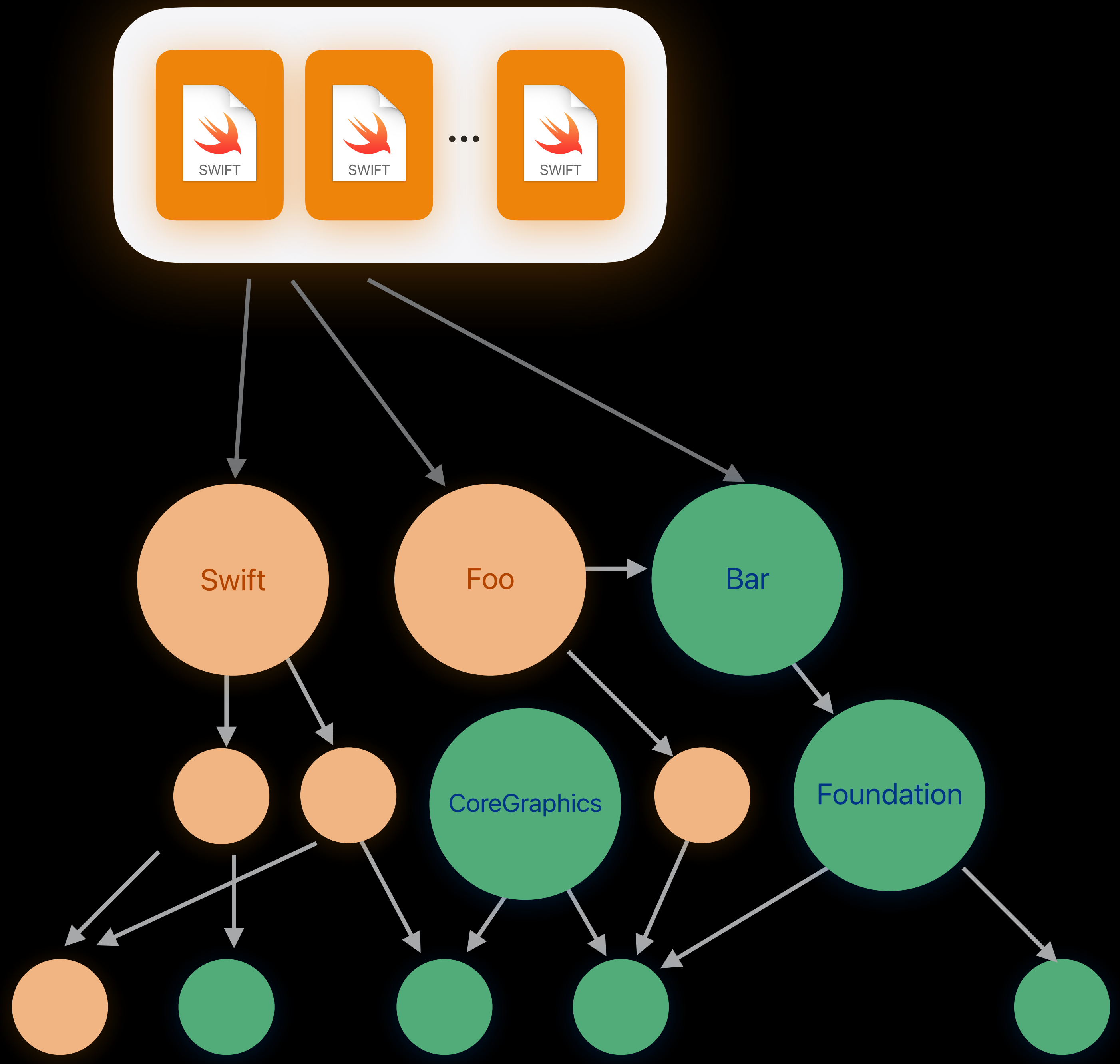
Module Resolution in Swift

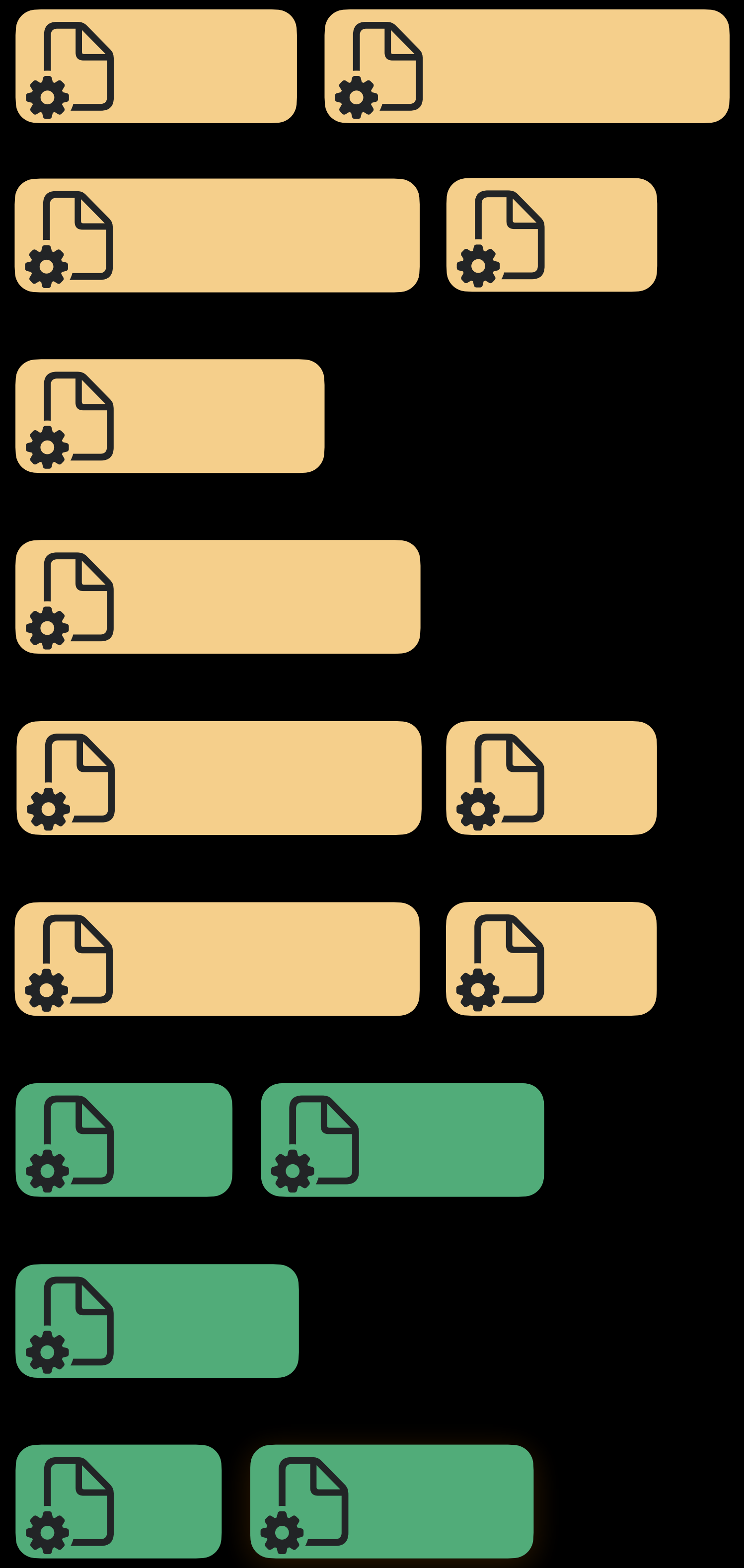
Explicitly Built Modules

Dependency Scan

Build Modules

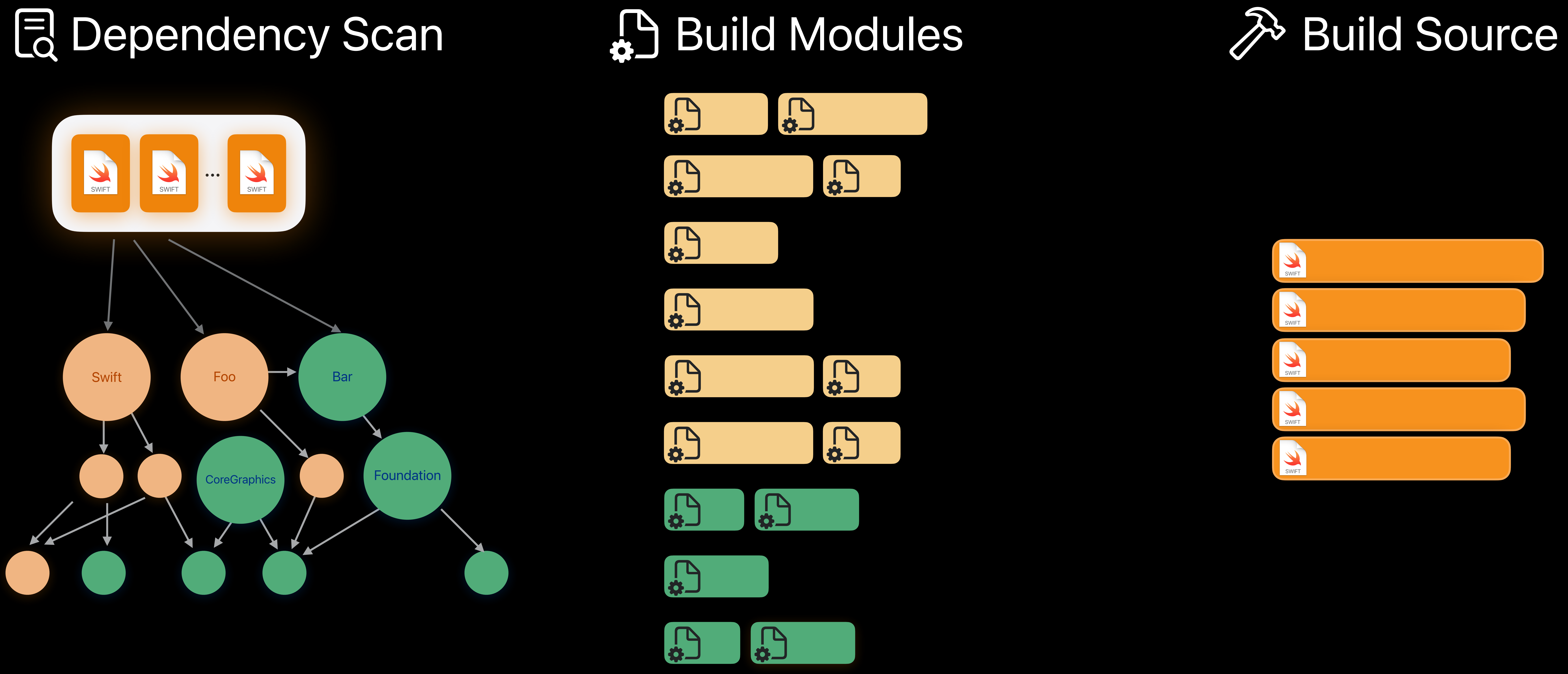
Build Source





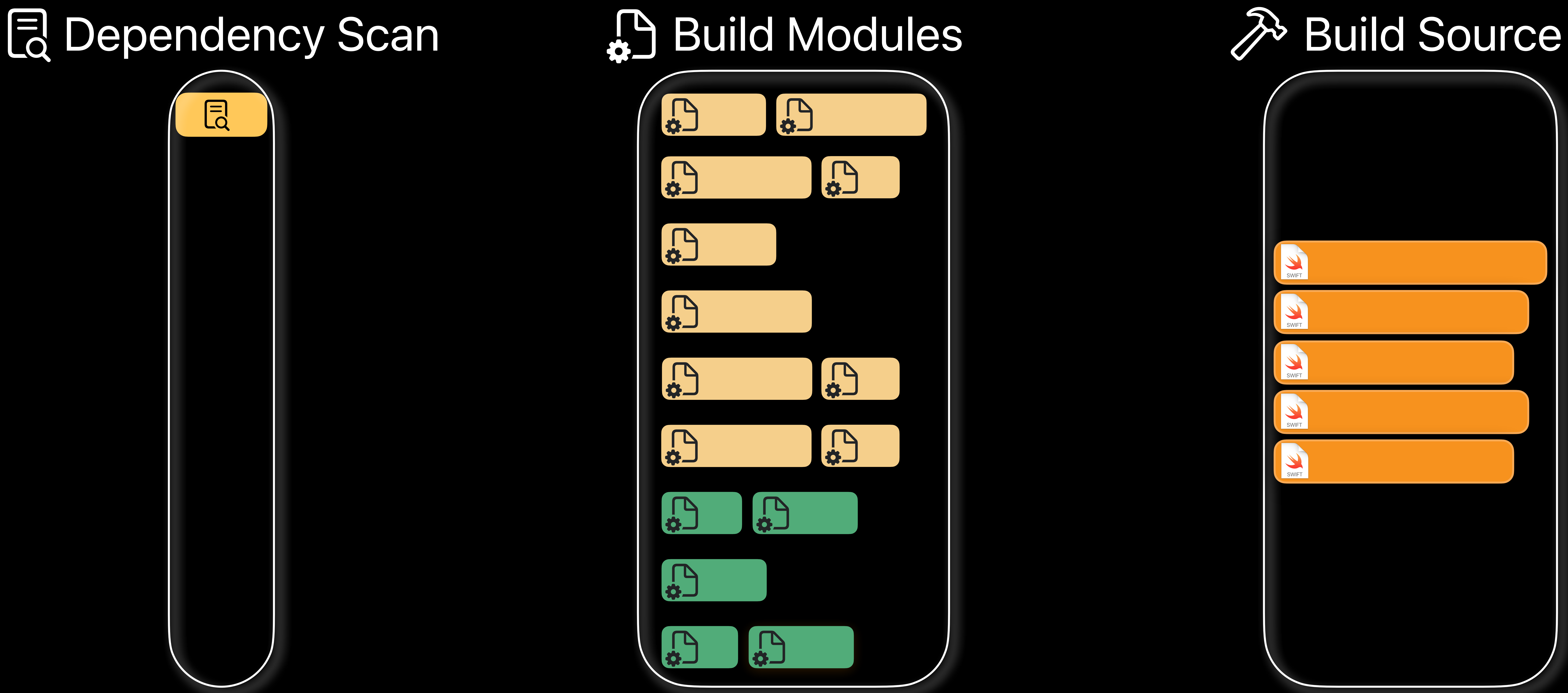
Module Resolution in Swift

Explicitly Built Modules



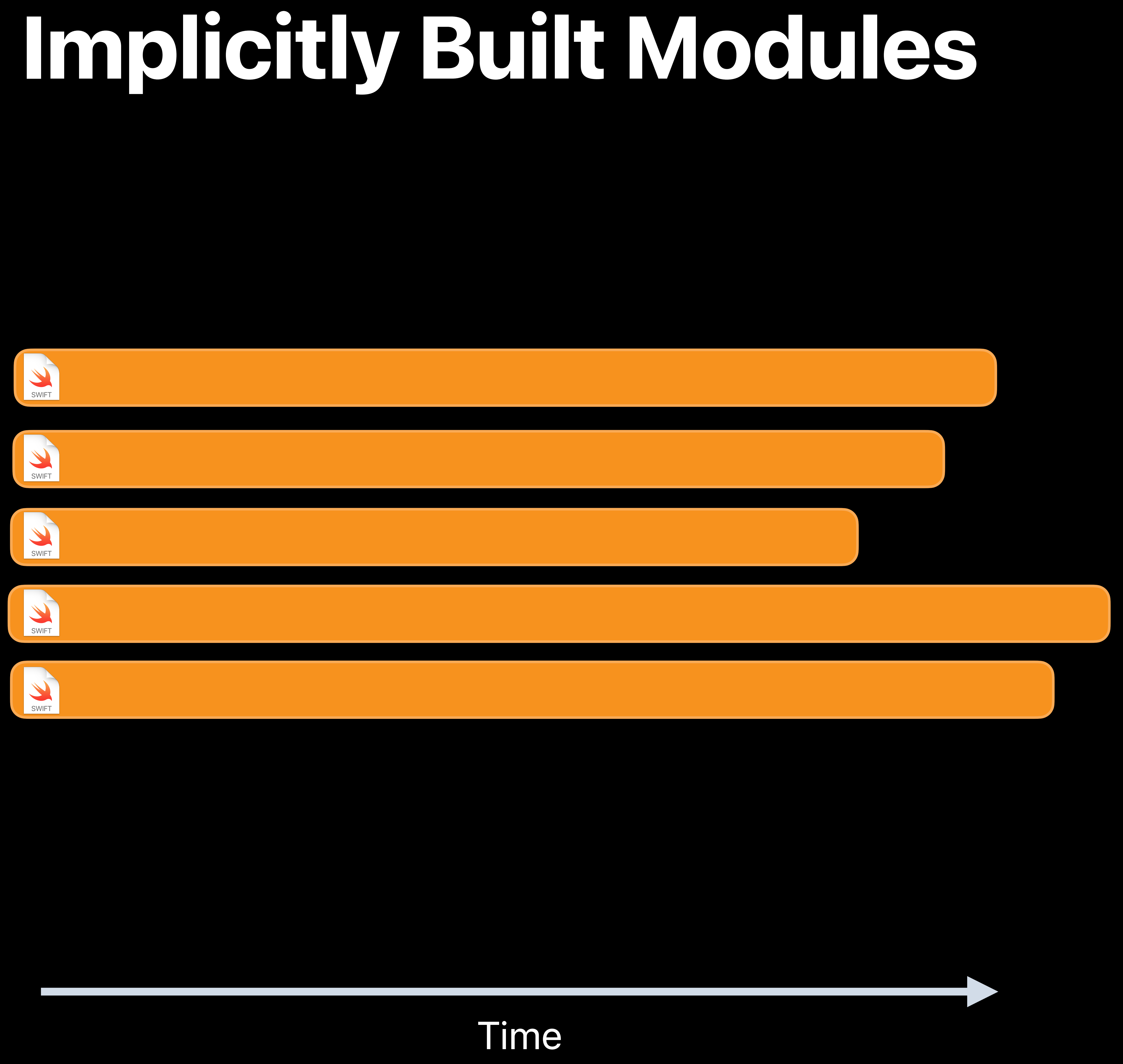
Module Resolution in Swift

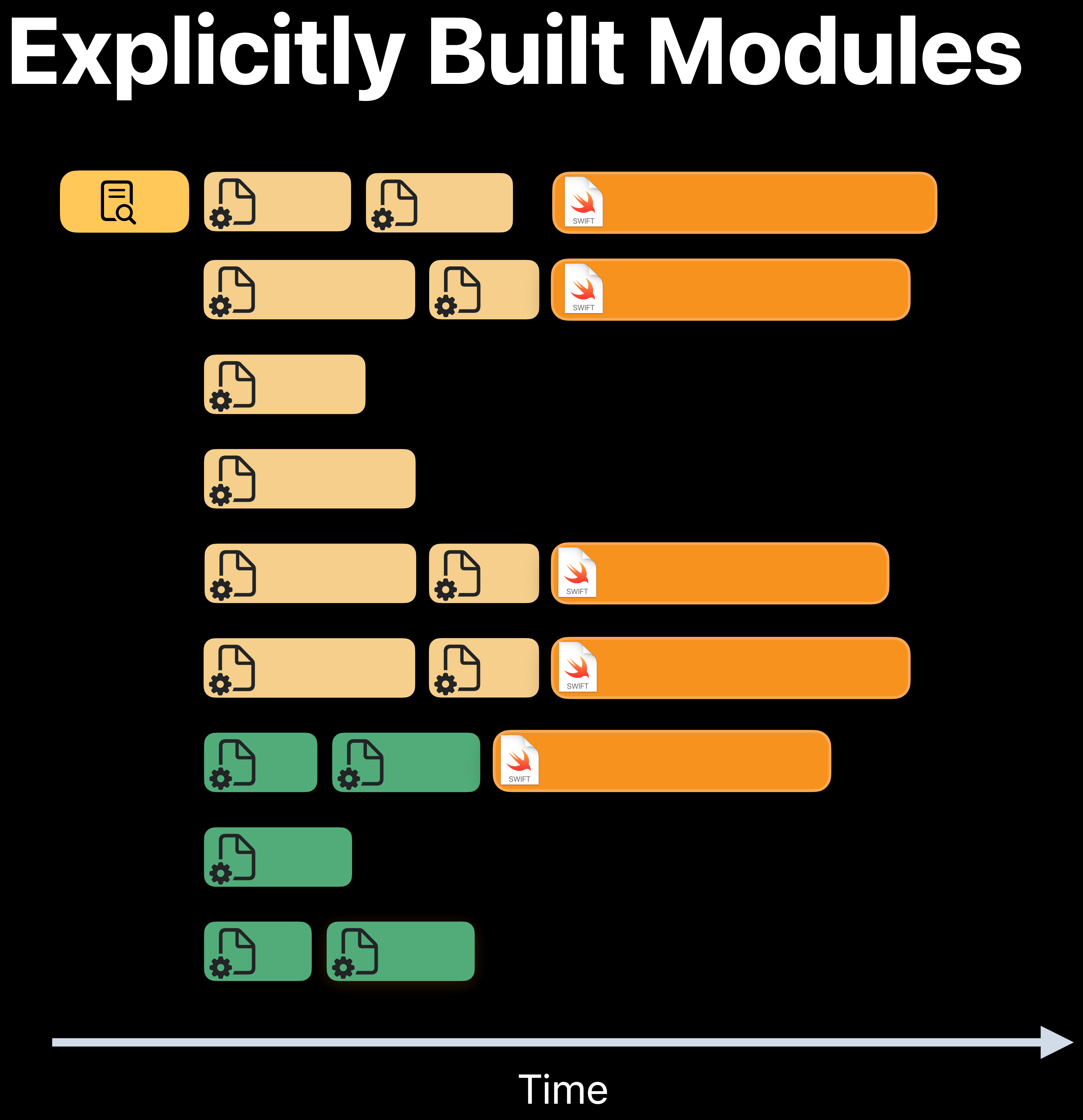
Explicitly Built Modules



Module Resolution in Swift

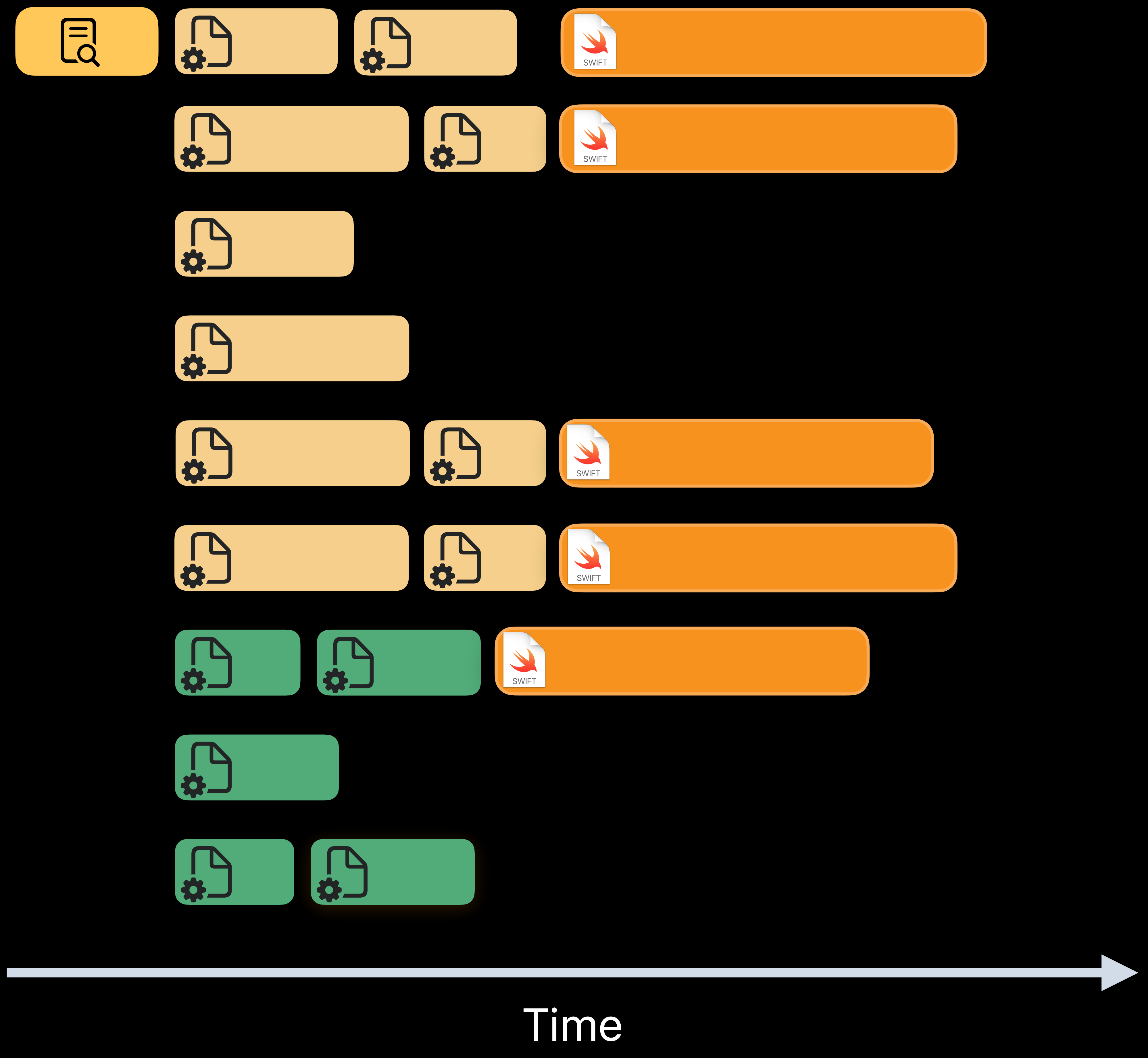
Explicitly Built Modules





Module Resolution in Swift

Explicitly Built Modules

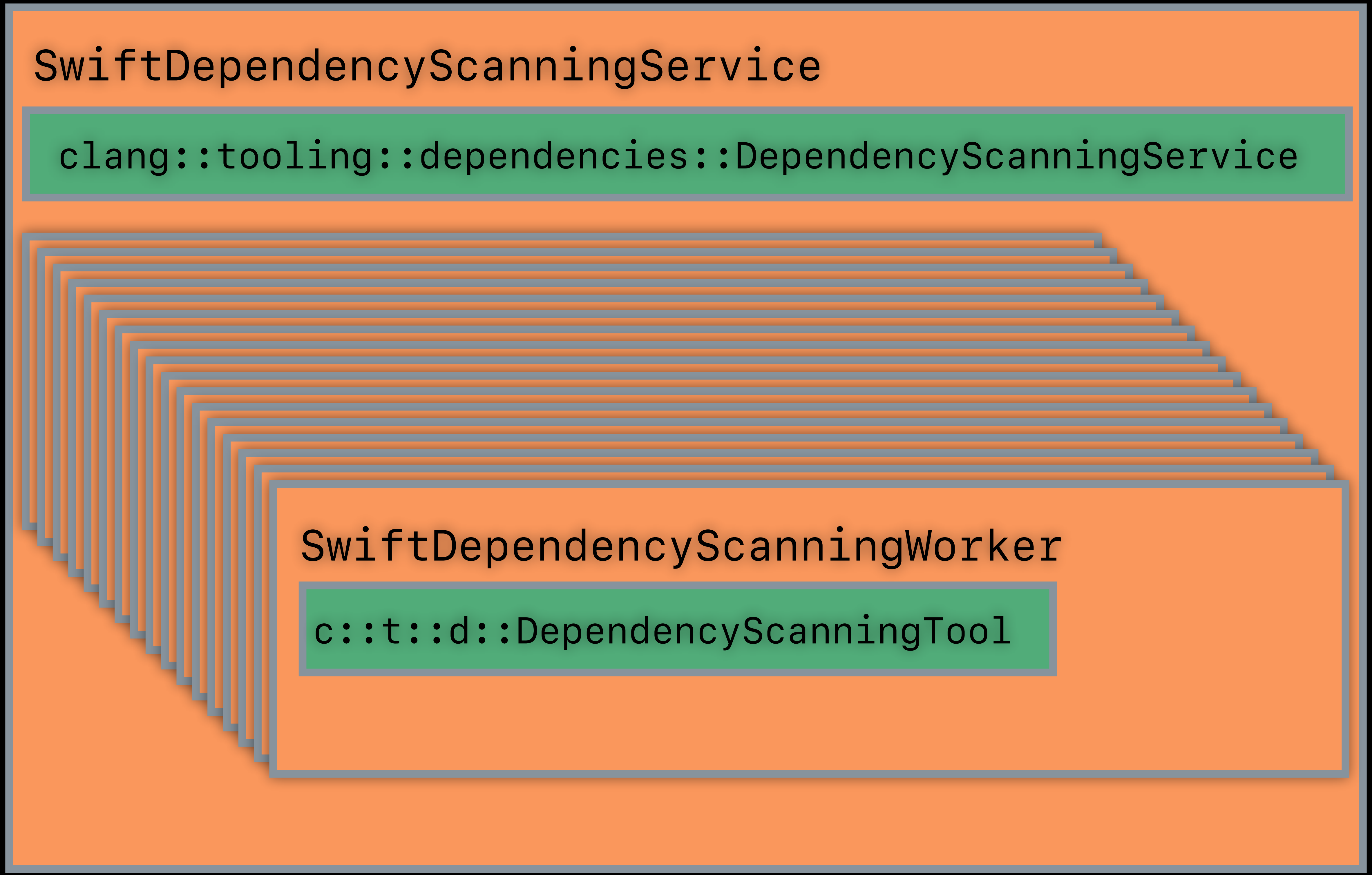


- More scheduling parallelism opportunities
- Distributable Build
- Deterministic, explainable
- Isolated Compilation Tasks
- Simpler Debugging
- Earlier actionable error detection

Module Resolution in Swift

Explicitly Built Modules

Dependency Scanner

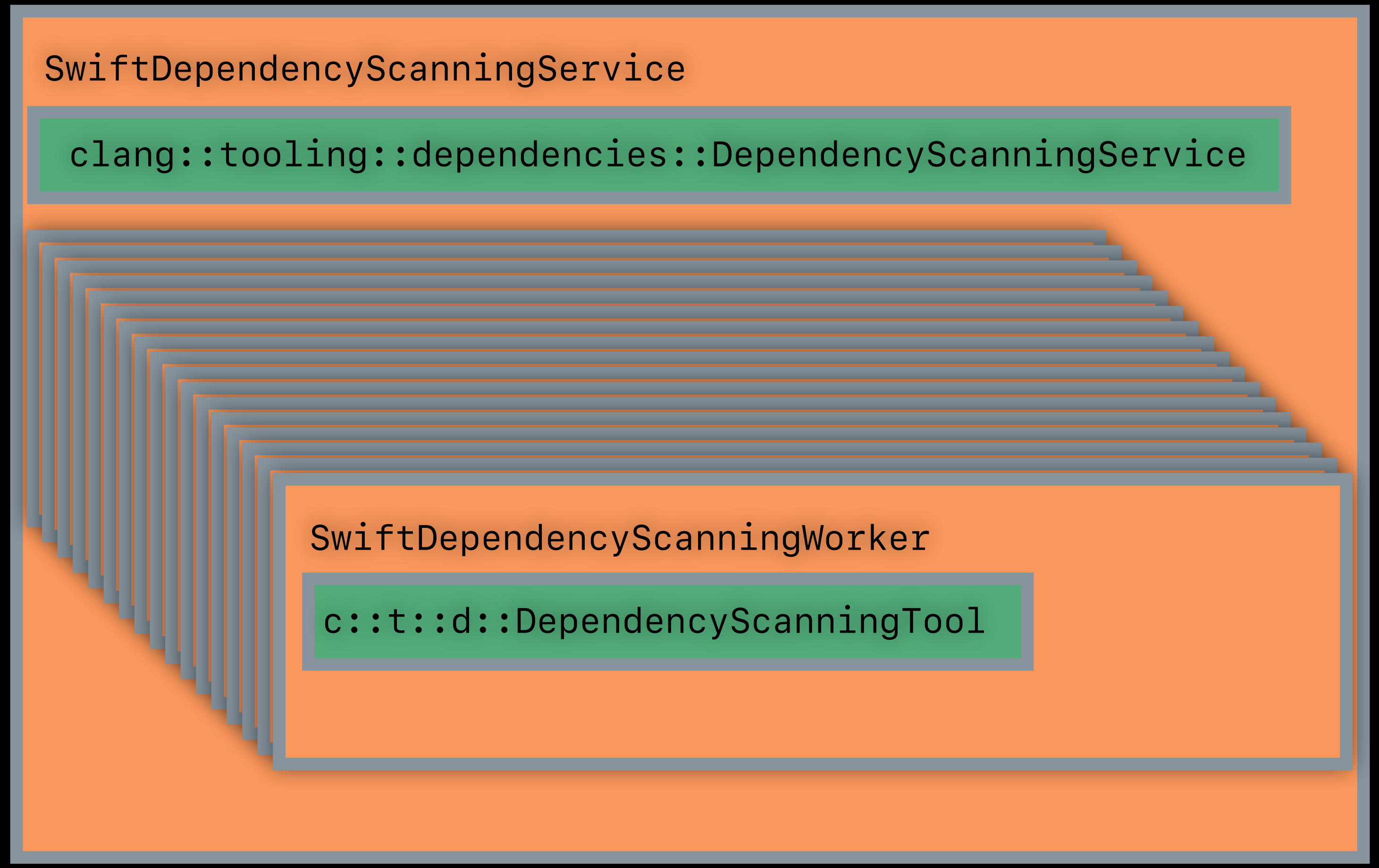

SwiftDependencyScanningService
clang::tooling::dependencies::DependencyScanningService
SwiftDependencyScanningWorker
c::t::d::DependencyScanningTool

[2] *A. Lorenz & M.Spencer: clang-scan-deps: Fast Dependency Scanning For Explicit Modules (EuroLLVM 2019)*

Module Resolution in Swift

Explicitly Built Modules

Dependency Scanner

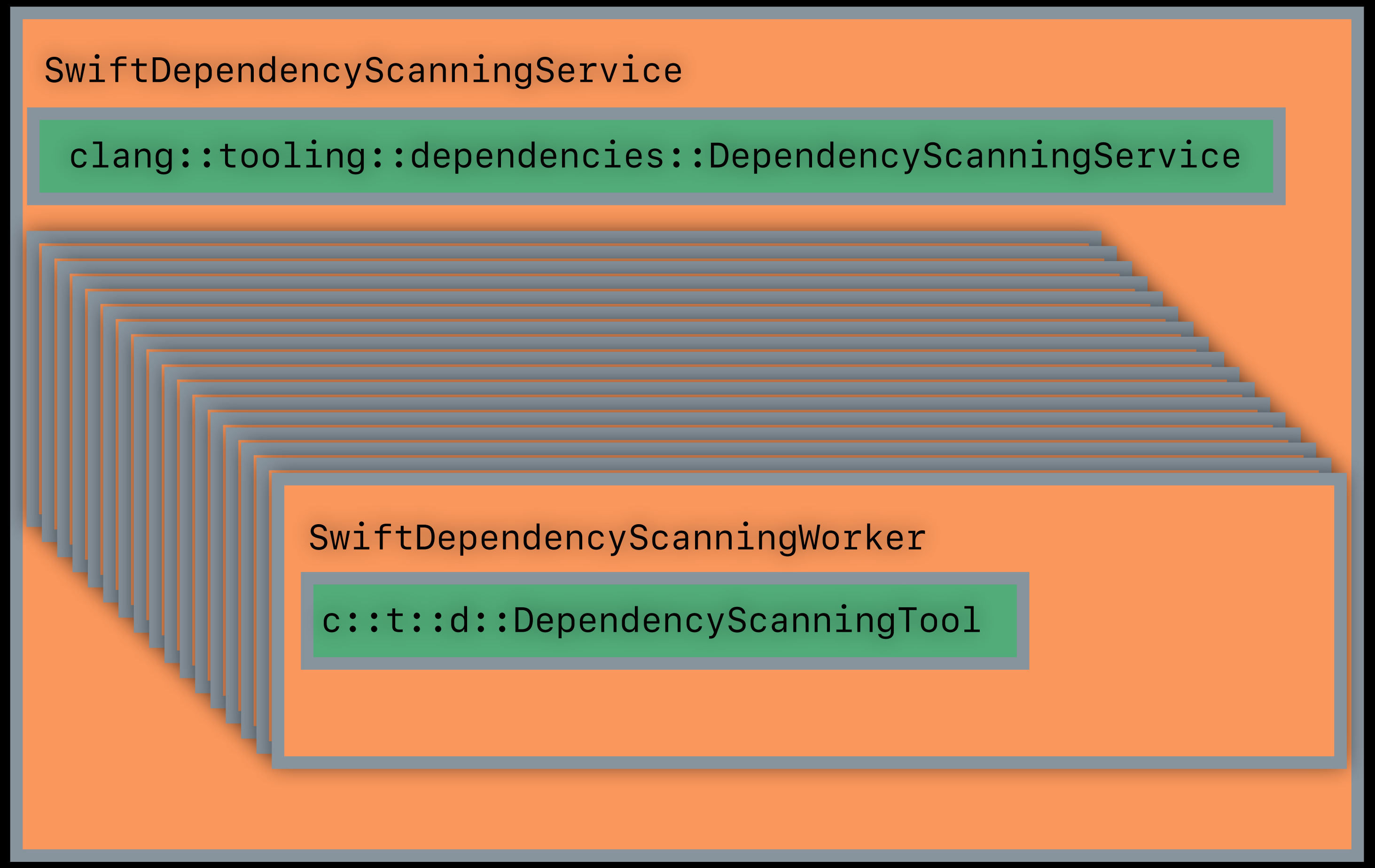

SwiftDependencyScanningService
clang::tooling::dependencies::DependencyScanningService
SwiftDependencyScanningWorker
c::t::d::DependencyScanningTool

```
let worklist = get_source_imports()
for id in worklist {
  if (dep = worker->findSwiftModule(id))
    worklist.add(dep->getImports)
  else if (dep = worker->clangTool->findClangModule(id))
    Record dep and entire sub-graph
  else
    error("Module not found: \(id)")
}
```

Module Resolution in Swift

Explicitly Built Modules

Dependency Scanner


SwiftDependencyScanningService
clang::tooling::dependencies::DependencyScanningService
SwiftDependencyScanningWorker
c::t::d::DependencyScanningTool

```
let worklist = get_source_imports()
for id in worklist {
  if (dep = worker->findSwiftModule(id))
    worklist.add(dep->getImports)
  else if (dep = worker->clangTool->findClangModule(id))
    Record dep and entire sub-graph
  else
     error("Module not found: \(id)")
}
```

```
class DependencyScanningTool {
public:
…
  llvm::Expected<ModuleDepsGraph> getModuleDependencies(
      StringRef ModuleName, const std::vector<std::string> &CommandLine,
      StringRef CWD, const llvm::DenseSet<ModuleID> &AlreadySeen,
      LookupModuleOutputCallback LookupModuleOutput);
```

Module Resolution in Swift

Explicitly Built Modules

Dependency Scanner Performance

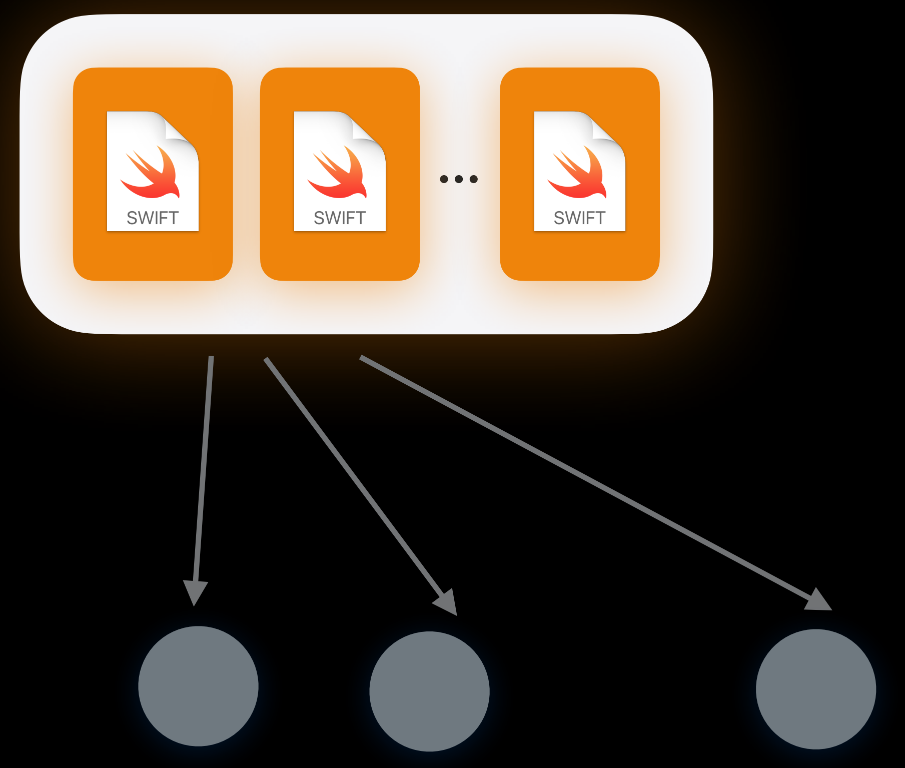
- Scanning performance crucial -> on the build critical path
- Swift module discovery relatively very cheap
- Clang module discovery and recipe formulation expensive:
  - Must parse all headers
  - Must identify full transitive dependency closure
- *Insight*: Once a dependency graph of Swift dependencies has been built, all unresolved imports are Clang modules -> Can be resolved as one large batch

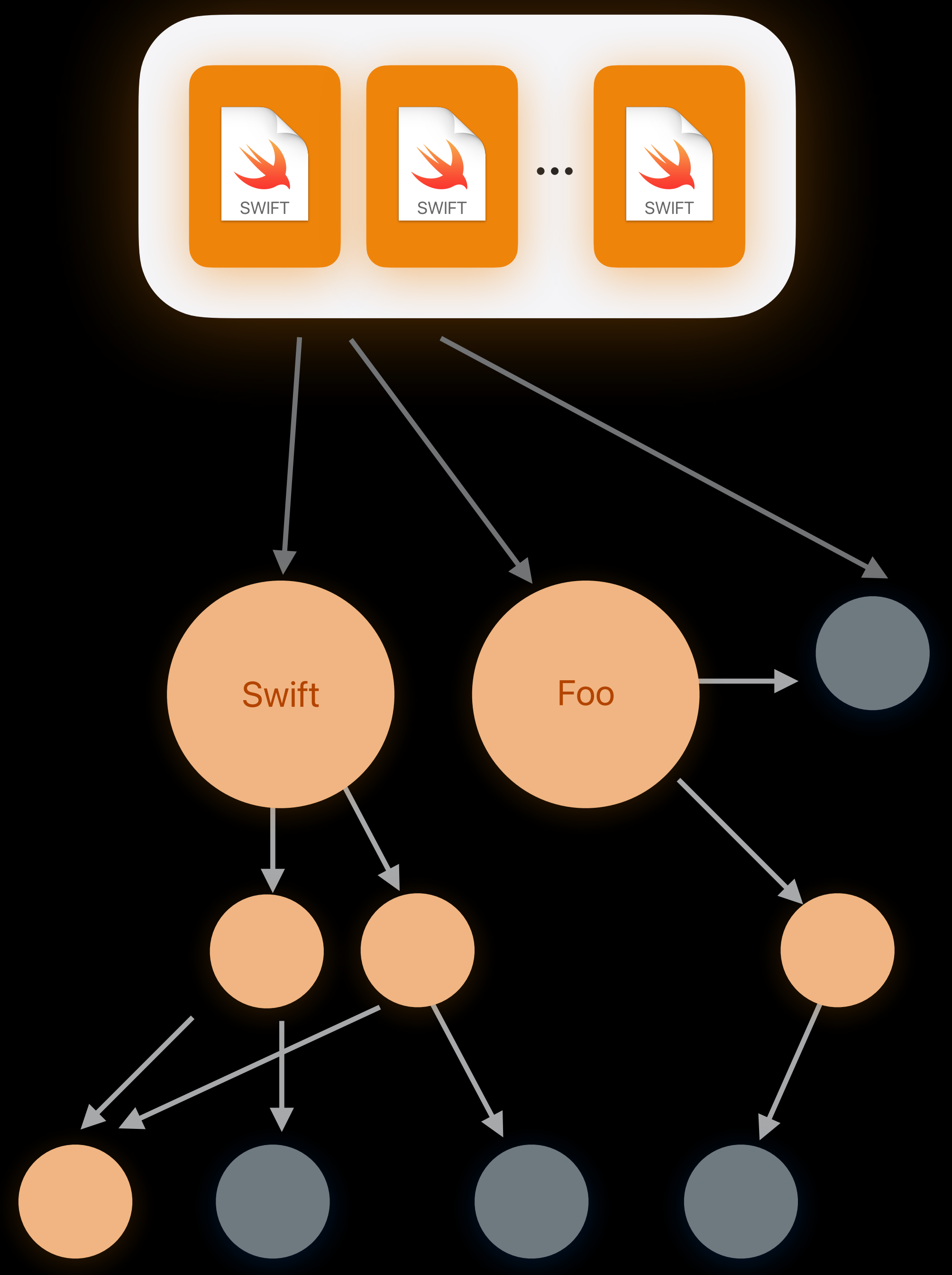
Module Resolution in Swift

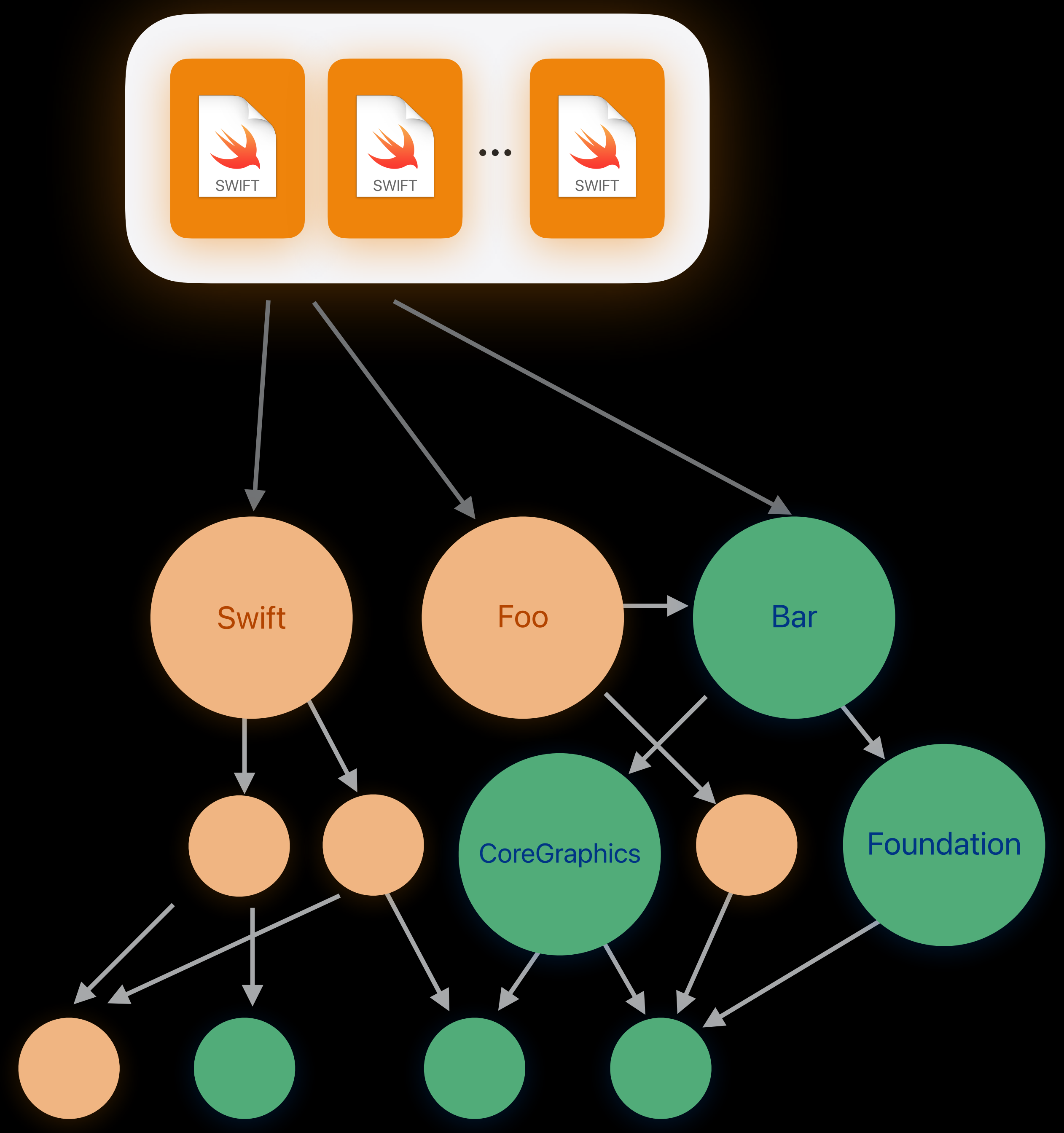
Explicitly Built Modules

Dependency Scanner Performance

Batch resolution of Clang dependencies


SWIFT
SWIFT
SWIFT


SWIFT
SWIFT
SWIFT
Swift
Foo


SWIFT
SWIFT
SWIFT
Swift
Foo
Bar
CoreGraphics
Foundation