Binary Frameworks in Swift

Harlan Haskins, Swift Team
Jordan Rose, Swift Team
Swift Packages
Swift Packages
Swift Packages

Dependencies managed by Xcode
Swift Packages

Dependencies managed by Xcode

Versions managed by Xcode
Swift Packages

- Dependencies managed by Xcode
- Versions managed by Xcode
- No binary compatibility requirements
Swift Packages

XCFrameworks
Agenda
Agenda

Introducing XCFrameworks
Agenda

Introducing XCFrameworks
Creating an XCFramework
Agenda

Introducing XCFrameworks
Creating an XCFramework
Framework author considerations
Demo
Using an XCFramework
Using Frameworks
Trust
Adopting Swift Packages in Xcode

Creating Swift Packages

WWDC 2019
Creating an XCFramework
import UIKit

public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
                    to destination: Location,
                    speed: Speed) {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
}

public struct Location {
    public var coordinates: Coordinates
}
import UIKit

public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed)
    {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
}

public struct Location {
    public var coordinates: Coordinates
}

import UIKit

public class Spaceship {
  public let name: String
  private var currentLocation: Location

  public init(name: String) {
    self.name = name
    currentLocation = .launchpad
  }

  public func fly(
    to destination: Location,
    speed: Speed)
  {
    currentLocation = destination
  }
}

public enum Speed {
  case leisurely
  case fast
}

public struct Location {
  public var coordinates: Coordinates
}
import UIKit

public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed)
    {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
}

public struct Location {
    public var coordinates: Coordinates
}
import UIKit

public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed
    ) {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
}

public struct Location {
    public var coordinates: Coordinates
}
import UIKit

public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location, 
        speed: Speed) {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
}

public struct Location {
    public var coordinates: Coordinates
}

import UIKit

public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed)
    { currentLocation = destination
    }

    public enum Speed {
        case leisurely
        case fast
    }

    public struct Location {
        public var coordinates: Coordinates
    }
}
import UIKit

public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed)
    {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
}

public struct Location {
    public var coordinates: Coordinates
}

New Build Setting in Xcode 11

Build Options

Setting: FlightKit

Build Libraries for Distribution: Yes
Compiled module was created by a newer version of the compiler
Importing a Module

exec
Importing a Module

```swift
import MyFramework

func doSomething()
```

- `import MyFramework`
- `func doSomething()`
Compiled Module Format

.swiftmodule
Compiled Module Format

Serialized, binary format

.swiftmodule
Compiled Module Format

Serialized, binary format

Internal compiler data structures

.swiftmodule
Compiled Module Format

Serialized, binary format

Internal compiler data structures

Changes between compiler versions
Swift Module Interfaces
Swift Module Interfaces

Textual listing of public API
Swift Module Interfaces

Textual listing of public API

Compatible across compiler versions
Swift Module Interfaces

Textual listing of public API

Compatible across compiler versions

Enabled with “Build Libraries for Distribution”
import UIKit

public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed)
    {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
}

public struct Location {
    public var coordinates: Coordinates
}
import UIKit

public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed)
    }

    public enum Speed {
        case leisurely
        case fast
    }

    public struct Location {
        public var coordinates: Coordinates
    }
}

import Swift
import UIKit

public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed)
    }

    public enum Speed {
        case leisurely
        case fast
    }

    public struct Location {
        public var coordinates: Coordinates
    }
}
import UIKit

public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(to destination: Location, speed: Speed) {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
}

public struct Location {
    public var coordinates: Coordinates
}

// swift-interface-format-version: 1.0
// swift-compiler-version: Swift version 5.1
// swift-module-flags: -target arm64-apple-ios13.0
// -enable-library-evolution
// -swift-version 5 -O
// -module-name FlightKit

import Swift
import UIKit

public class Spaceship {
    public let name: Swift.String
    public init(name: Swift.String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(to destination: FlightKit.Location, speed: FlightKit.Speed) {
        currentLocation = destination
    }
}

public enum Speed: Swift.Hashable {
    case leisurely
    case fast
}

public struct Location {
    public var coordinates: Coordinates
}

@objc deinit

public static func ==(a: FlightKit.Speed, b: FlightKit.Speed) -> Swift.Bool

public func hash(

import UIKit

public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(to destination: Location, speed: Speed) {
        currentLocation = destination
    }

    public enum Speed {
        case leisurely
        case fast
    }

    public struct Location {
        public var coordinates: Coordinates
    }
}
import UIKit

public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed)
    {
        currentLocation = destination
    }

    public enum Speed {
        case leisurely
        case fast
    }

    public struct Location {
        public var coordinates: Coordinates
    }
}
```swift
import UIKit
public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed)
        currentLocation = destination
    }

    public enum Speed: Swift.Hashable {
        case leisurely
        case fast
    }

    public struct Location {
        public var coordinates: Coordinates
    }
}
```

```swift
import Swift
import UIKit
public class Spaceship {
    public let name: Swift.String
    private var currentLocation: Location

    public init(name: Swift.String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: FlightKit.Location,
        speed: FlightKit.Speed)
        currentLocation = destination
    }

    @objc deinit

    public enum Speed: Swift.Hashable {
        case leisurely
        case fast
    }

    public struct Location {
        public var coordinates: FlightKit.Coordinates
    }
}
```
import UIKit

public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed)
    { currentLocation = destination }
}

public enum Speed {
    case leisurely
    case fast
}

public struct Location {
    public var coordinates: Coordinates
}

import Swift
import UIKit

public class Spaceship {
    public let name: Swift.String

    public init(name: Swift.String)
    {
        self.name = name
    }

    public func fly(
        to destination: FlightKit.Location,
        speed: FlightKit.Speed)
    {
        currentLocation = destination
    }

    @objc deinit
}

public enum Speed: Swift.Hashable {
    case leisurely
    case fast

    public static func ==(a: FlightKit.Speed, b: FlightKit.Speed) -> Swift.Bool
    public func hash(into hasher: inout Swift.Hasher)
}

public struct Location {
    public var coordinates: FlightKit.Coordinates
}

import UIKit

public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed)
    {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
}

public struct Location {
    public var coordinates: Coordinates
}

import Swift
import UIKit

public class Spaceship {
    public let name: String

    public init(name: String)
    {
        self.name = name
    }

    public func fly(
        to destination: Location,
        speed: Speed)
    {
        currentLocation = destination
    }

    @objc deinit
}

public enum Speed: Swift.Hashable {
    case leisurely
    case fast

    public static func ==(a: FlightKit.Speed, b: FlightKit.Speed) -> Swift.Bool
    {
    }

    public func hash(
        into hasher: inout Swift.Hasher)
    {
    }
}

public struct Location {
    public var coordinates: Coordinates
}

public struct Coordinates {
    public var x: Int
    public var y: Int
}
import UIKit

public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed
    ) {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
}

public struct Location {
    public var coordinates: Coordinates
}

import Swift
import UIKit

public class Spaceship {
    public let name: String

    public init(name: String) {
        self.name = name
    }

    public func fly(
        to destination: Location,
        speed: Speed
    ) {
    }
}

public enum Speed: Swift.Hashable {
    case leisurely
    case fast

    public static func ==(a: FlightKit.Speed, b: FlightKit.Speed) -> Swift.Bool

    public func hash(
        into hasher: inout Swift.Hasher
    )
}

public struct Location {
    public var coordinates: FlightKit.Coordinates
}

import FlightKit
import UIKit

public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed)
    }

    public enum Speed {
        case leisurely
        case fast
    }

    public struct Location {
        public var coordinates: Coordinates
    }
}

import Swift
import UIKit

public class Spaceship {
    public let name: Swift.String

    public init(name: Swift.String) {
    }

    public func fly(
        to destination: FlightKit.Location,
        speed: FlightKit.Speed)
    }

    @objc deinit
}

public enum Speed: Swift.Hashable {
    case leisurely
    case fast

    public static func ==(a: FlightKit.Speed, b: FlightKit.Speed) -> Swift.Bool

    public func hash(
        into hasher: inout Swift.Hasher)
    }

    public struct Location {
        public var coordinates: FlightKit.Coordinates
    }
}
import UIKit

public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed)
    {
        currentLocation = destination
    }

    public enum Speed {
        case leisurely
        case fast
    }

    public struct Location {
        public var coordinates: Coordinates
    }
}

import Swift
import UIKit

public class Spaceship {
    public let name: Swift.String
    public init(name: Swift.String)
    {
        self.name = name
    }

    public func fly(
        to destination: FlightKit.Location,
        speed: FlightKit.Speed)
    {
        currentLocation = destination
    }

    @objc deinit
}

public enum Speed: Swift.Hashable {
    case leisurely
    case fast

    public static func ==(a: FlightKit.Speed, b: FlightKit.Speed) -> Swift.Bool
    public func hash(
        into hasher: inout Swift.Hasher)
    }

    public struct Location {
        public var coordinates: FlightKit.Coordinates
    }
}
import UIKit

public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed)
    
    public enum Speed {
        case leisurely
        case fast
    }

    public struct Location {
        public var coordinates: Coordinates
    }
}

import Swift
import UIKit

public class Spaceship {
    public let name: Swift.String
    public init(name: Swift.String) {
    }

    public func fly(
        to destination: FlightKit.Location,
        speed: FlightKit.Speed)
    
    public enum Speed: Swift.Hashable {
        case leisurely
        case fast
    }

    public static func ==(a: FlightKit.Speed, b: FlightKit.Speed) -> Swift.Bool

    public func hash(
        into hasher: inout Swift.Hasher)
    
    public struct Location {
        public var coordinates: FlightKit.Coordinates
    }
}
import UIKit

public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed)
    {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
}

public struct Location {
    public var coordinates: Coordinates
}

import Swift
import UIKit

public class Spaceship {
    public let name: Swift.String
    public init(name: Swift.String)
    {
        self.name = name
    }

    public func fly(
        to destination: FlightKit.Location,
        speed: FlightKit.Speed)
    {
        currentLocation = destination
    }

    @objc deinit
}

public enum Speed: Swift.Hashable {
    case leisurely
    case fast

    public static func ==(a: FlightKit.Speed, b: FlightKit.Speed) -> Swift.Bool

    public func hash(
        into hasher: inout Swift.Hasher)
    }

public struct Location {
    public var coordinates: FlightKit.Coordinates
}
import UIKit

public class Spaceship {
public let name: String
private var currentLocation: Location

public init(name: String) {
    self.name = name
    currentLocation = .launchpad
}

public func fly(
    to destination: Location,
    speed: Speed)
    currentLocation = destination
}

public enum Speed {
case leisurely
case fast
}

public struct Location {
    public var coordinates: Coordinates
}

import Swift
import UIKit

public class Spaceship {
public let name: Swift.String

public init(name: Swift.String) {
    self.name = name
}

public func fly(
    to destination: FlightKit.Location,
    speed: FlightKit.Speed)
{
    currentLocation = destination
}

public struct Location {
    public var coordinates: FlightKit.Coordinates
}
public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed) {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
}

public struct Location {
    public var coordinates: Coordinates
}

import Swift
import UIKit

public class Spaceship {
    public let name: Swift.String
    public init(name: Swift.String) {
        self.name = name
    }

    public func fly(
        to destination: FlightKit.Location,
        speed: FlightKit.Speed) {
        currentLocation = destination
    }

    @objc deinit
}

public enum Speed: Swift.Hashable {
    case leisurely
    case fast

    public static func ==(a: FlightKit.Speed, b: FlightKit.Speed) -> Swift.Bool
    public func hash(into hasher: inout Swift.Hasher)
}

public struct Location {
    public var coordinates: FlightKit.Coordinates
}
public class Spaceship {

    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed)
    {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
}

public struct Location {
    public var coordinates: Coordinates
}

import Swift
import UIKit

public class Spaceship {

    public let name: Swift.String
    public init(name: Swift.String) {
    }

    public func fly(
        to destination: FlightKit.Location,
        speed: FlightKit.Speed)
    {
        currentLocation = destination
    }
}

@objc deinit

public enum Speed: Swift.Hashable {
    case leisurely
    case fast

    public static func ==(a: FlightKit.Speed, b: FlightKit.Speed) -> Swift.Bool

    public func hash(
        into hasher: inout Swift.Hasher)
    }

public struct Location {
    public var coordinates: FlightKit.Coordinates
}

```swift
public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed
    ) {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
}

public struct Location {
    public var coordinates: Coordinates
}
```

```swift
import Swift
import UIKit

public class Spaceship {
    public let name: String

    public init(name: String) {
        self.name = name
    }

    public func fly(
        to destination: Location,
        speed: Speed
    ) {
        currentLocation = destination
    }
}

public enum Speed: Hashable {
    case leisurely
    case fast

    public static func ==(a: Speed, b: Speed) -> Bool
    public func hash(into hasher: inout Hasher)
}

public struct Location {
    public var coordinates: Coordinates
}
```
import UIKit

public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed
    ) {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
}

public struct Location {
    public var coordinates: Coordinates
}

import Swift
import UIKit

public class Spaceship {
    public let name: String

    public init(name: String)
    {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed
    ) {
        currentLocation = destination
    }
}

public enum Speed: Hashable {
    case leisurely
    case fast

    public static func ==(a: Speed, b: Speed) -> Bool
    public func hash(into hasher: inout Hasher)
}

public struct Location {
    public var coordinates: Coordinates
}
import UIKit

public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed)
    {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
}

public struct Location {
    public var coordinates: Coordinates
}

import Swift
import UIKit

public class Spaceship {
    public let name: Swift.String
    public init(name: Swift.String)
    public func fly(
        to destination: FlightKit.Location,
        speed: FlightKit.Speed)
    {
        currentLocation = destination
    }
}

public enum Speed: Swift.Hashable {
    case leisurely
    case fast

    public static func ==(a: FlightKit.Speed, b: FlightKit.Speed) -> Swift.Bool

    public func hash(
        into hasher: inout Swift.Hasher)
    }
}

public struct Location {
    public var coordinates: FlightKit.Coordinates
}
import UIKit

public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed)
    {
        currentLocation = destination
    }

    public enum Speed {
        case leisurely
        case fast
    }

    public struct Location {
        public var coordinates: Coordinates
    }
}

import Swift
import UIKit

public class Spaceship {
    public let name: Swift.String
    private var currentLocation: FlightKit.Location

    public init(name: Swift.String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: FlightKit.Location,
        speed: FlightKit.Speed)
    {
        currentLocation = destination
    }

    public enum Speed: Swift.Hashable {
        case leisurely
        case fast
    }

    public static func ==(a: FlightKit.Speed,
                           b: FlightKit.Speed) -> Swift.Bool

    public func hash(
                    a: FlightKit.Speed)
    {
    }

    @objc deinit
}

import UIKit

public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed)
    {
        currentLocation = destination
    }

    public enum Speed {
        case leisurely
        case fast
    }

    public struct Location {
        public var coordinates: Coordinates
    }
}
Building an XCFramework
Archiving Your Framework
Archiving Your Framework

Builds framework in Release
Archiving Your Framework

Builds framework in Release

Available in Xcode Organizer
Archiving Your Framework

Builds framework in Release

Available in Xcode Organizer

Keeps track of dSYMs
Archiving Your Framework

xcodebuild archive
Archiving Your Framework

xcodebuild archive
   -scheme FlightKit
Archiving Your Framework

```bash
xcodebuild archive
   -scheme FlightKit
   -destination "..."
   -destination "..."
   ...
   -destination "...
```
Archiving Your Framework

```bash
xcodebuild archive
    -scheme FlightKit
    -destination "...
    -destination "...
    ...
    -destination "...
SKIP_INSTALL=NO
```
Archiving Your Framework
Archiving Your Framework
Archiving Your Framework
Archiving Your Framework
Building an XCFramework
Building an XCFramework
Building an XCFramework

xcodebuild -create-xcframework
Building an XC Framework

```
xcodebuild -create-xcframework
  -framework [path]
  -framework [path]
...
  -framework [path]
  -output FlightKit.xcframework
```
Building an XCFramework
Building an XCFramework

1. Enable “Build Libraries for Distribution”
Building an XCFramework

1. Enable “Build Libraries for Distribution”
2. `xcodebuild archive`
Building an XC.Framework

1. Enable “Build Libraries for Distribution”
2. `xcodebuild archive`
3. `xcodebuild -create-xcframework`
Framework Author Considerations

Jordan Rose, Swift Team
Evolving your framework

Trading flexibility for optimizability

Helping your clients
Evolving Your Framework
Evolving Your Framework

1.0

1.1
Why Is Binary Compatibility Important?
Why Is Binary Compatibility Important?
Why Is Binary Compatibility Important?
Why Is Binary Compatibility Important?
Why Is Binary Compatibility Important?
Why Is Binary Compatibility Important?
Why Is Binary Compatibility Important?

1.0

1.1

2.1
Framework Versions
# Framework Versions

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Information Property List</td>
<td>Dictionary</td>
<td>(8 items)</td>
</tr>
<tr>
<td>Localization native development re...</td>
<td>String</td>
<td>$(DEVELOPMENT_LANGUAGE)</td>
</tr>
<tr>
<td>Executable file</td>
<td>String</td>
<td>$(EXECUTABLE_NAME)</td>
</tr>
<tr>
<td>Bundle identifier</td>
<td>String</td>
<td>$(PRODUCT_BUNDLE_IDENTIFIER)</td>
</tr>
<tr>
<td>InfoDictionary version</td>
<td>String</td>
<td>6.0</td>
</tr>
<tr>
<td>Bundle name</td>
<td>String</td>
<td>$(PRODUCT_NAME)</td>
</tr>
<tr>
<td>Bundle OS Type code</td>
<td>String</td>
<td>$(PRODUCT_BUNDLE_PACKAGE_TYPE)</td>
</tr>
<tr>
<td>Bundle versions string, short</td>
<td>String</td>
<td>1.0</td>
</tr>
<tr>
<td>Bundle version</td>
<td>String</td>
<td>$(CURRENT_PROJECT_VERSION)</td>
</tr>
</tbody>
</table>
Semantic Versioning
semver.org
1.2.4

Patch Version
Bug fixes
Minor Version
Compatible additions

1.2.4
1.2.4

Major Version
Breaking changes (source, binary, semantic)
public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(to destination: Location, speed: Speed) {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
}

public struct Location {
    public var coordinates: Coordinates
}
public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed) {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
}

public struct Location {
    public var coordinates: Coordinates
}

public class Spaceship {
    public let name: String
    private static var defaultLocation: Location?
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = Spaceship.defaultLocation ?? .launchpad
    }

    public func doABarrelRoll() {
        // …
    }

    public func fly(
        to destination: Location,
        speed: Speed,
        stealthily: Bool = false) {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
    case ludicrous
}
public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(to destination: Location, speed: Speed, stealthily: Bool = false) {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
    case ludicrous
}

public struct Location {
    public var coordinates: Coordinates
}
```swift
public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(to destination: Location, speed: Speed, stealthily: Bool = false) {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
    case ludicrous
}

public struct Location {
    public var coordinates: Coordinates
}
```
public class Spaceship {
    public let name: String

    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(to destination: Location, speed: Speed) {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
}

1.0.0
class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func doABarrelRoll() {}

    public func fly(to destination: Location, speed: Speed) {
        currentLocation = destination
    }

    public enum Speed {
        case leisurely
        case fast
    }
}
```swift
public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed
    ) {
        currentLocation = destination
    }

    public enum Speed {
        case leisurely
        case fast
    }
}
```
public class Spaceship {
    public let name: String

    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed,
        stealthily: Bool = false
    ) {
        currentLocation = destination
    }

    public enum Speed {
        case leisurely
        case fast
    }
}
public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed,
        stealthily: Bool = false
    ) {
        currentLocation = destination
    }

    public enum Speed {
        case leisurely
        case fast
    }
}
public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(to destination: Location, speed: Speed, stealthily: Bool = false) {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
}
```swift
public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed,
        stealthily: Bool = false
    ) {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
}
```
```swift
public class Spaceship {
    public let name: String

    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(to destination: Location, speed: Speed) {
        currentLocation = destination
    }

    public enum Speed {
        case leisurely
        case fast
    }
}
```
public class Spaceship {
    public let name: String
    private var currentLocation: Location

    public init(name: String) {
        self.name = name
        currentLocation = .launchpad
    }

    public func fly(
        to destination: Location,
        speed: Speed
        ) {
        currentLocation = destination
    }
}

public enum Speed {
    case leisurely
    case fast
}
```swift
public func fly(to destination: Location, speed: Speed) {
    currentLocation = destination
}

public enum Speed {
    case leisurely
    case fast
}

public struct Location: Hashable {
    public var coordinates: Coordinates
    public var label: String
}
```
```swift
public enum Speed {
    case leisurely
    case fast
    case ludicrous
}

public struct Location: Hashable {
    public var coordinates: Coordinates
    public var label: String
}

public func fly(to destination: Location, speed: Speed) {
    currentLocation = destination
}
```

```
1.0.0
```

```
to destination: Location,
    speed: Speed) {
    currentLocation = destination
}
```
public func fly(to destination: Location, speed: Speed, stealthily: Bool = false) {
    currentLocation = destination
}

public enum Speed {
    case leisurely
    case fast
    case ludicrous
}

public struct Location: Hashable {
    public var coordinates: Coordinates
    public var label: String
}
public func fly(to destination: Location, speed: Speed, stealthily: Bool = false) {
    currentLocation = destination
}

public enum Speed {
    case leisurely
    case fast
    case ludicrous
}

public struct Location: Hashable {
    public var coordinates: Coordinates
    public var label: String
}

1.0.0

to destination: Location, speed: Speed) {
currentLocation = destination
}

case leisurely

case fast

case ludicrous

color: Coordinates

public var label: String
}

2.0.0
to destination: Location, speed: Speed, stealthily: Bool = false) {
currentLocation = destination
}

case leisurely

case fast

case ludicrous

color: Coordinates

public var label: String
}
public func fly(
to destination: Location,
speed: Speed,
stealthily: Bool = false) {
currentLocation = destination
}

public enum Speed {
case leisurely
case fast
case ludicrous
}

public struct Location: Hashable {
public var coordinates: Coordinates
public var label: String
}
Implications for API Design
Implications for API Design

Keep your interface small — easy to add, hard to remove!
Implications for API Design

Keep your interface small — easy to add, hard to remove!

Choose names and requirements carefully
Implications for API Design

Keep your interface small — easy to add, hard to remove!

Choose names and requirements carefully

Avoid unnecessary extensibility (open classes, arbitrary callbacks, etc.)
Behind the Scenes
Extra indirection at module boundaries

```java
public class Spaceship {
    public init()
    public func fly(
        to destination: Location,
        speed: Speed)
}
```
Behind the Scenes
Extra indirection at module boundaries

```swift
public class Spaceship {
    public init()
    public func fly(
        to destination: Location,
        speed: Speed)
}

spaceship.fly(to: home, speed: .fast)
```
Behind the Scenes
Extra indirection at module boundaries

```swift
public class Spaceship {
    public init()
    public func fly(
        to destination: Location,
        speed: Speed)
}

spaceship.fly(to: home, speed: .fast)

fly(to:speed:)?
```
Behind the Scenes

Extra indirection at module boundaries

```swift
public class Spaceship {
    public init()
    public func fly(
        to destination: Location,
        speed: Speed)
}
```

`spaceship.fly(to: home, speed: .fast)`

fly(to:speed:)?

Method #2!
Behind the Scenes
Extra indirection when clients use the framework’s structs and enums

```swift
public enum Speed {
    case leisurely
    case fast
}

spaceship.fly(to: home, speed: .fast)
```
Behind the Scenes
Extra indirection when clients use the framework’s structs and enums

```swift
public enum Speed {
    case leisurely
    case fast
}

spaceship.fly(to: home, speed: .fast)
```
Behind the Scenes
Extra indirection when clients use the framework’s structs and enums

```swift
public enum Speed {
    case leisurely
    case fast
}
```

```swift
spaceship.fly(to: home, speed: .fast)
```

How big?
Behind the Scenes
Extra indirection when clients use the framework’s structs and enums

```swift
public enum Speed {
    case leisurely
    case fast
}
```

```
spaceship.fly(to: home, speed: .fast)
```

How big?

1 byte!
Behind the Scenes
Extra indirection when clients use the framework’s structs and enums

```swift
public enum Speed {
    case leisurely
    case fast
}
```

```swift
spaceship.fly(to: home, speed: .fast)
```

Clean it up!
Behind the Scenes
Extra indirection when clients use the framework’s structs and enums

```swift
public enum Speed {
    case leisurely
    case fast
}
```

```swift
spaceship.fly(to: home, speed: .fast)
```

Clean it up!
Okay!
Trading Flexibility for Optimizability
Flexibility Optimizability

Build Options

- Build Libraries for Distribution: Yes
Flexibility Optimizability

Build Options

- Build Libraries for Distribution: Yes
Flexibility

Optimizability

Build Libraries for Distribution: Yes
@inlinable functions
@frozen enums
@frozen structs
public class CargoShip: Spaceship {
    @inlinable
    public func canCarry(
        _ cargo: Cargo) -> Bool {
        return cargo.mass <= self.capacity
    }
}

@usableFromInline
internal var capacity: Measurement<UnitMass>

internal var currentCargo: Cargo?
public class CargoShip: Spaceship {

    @inlinable
    public func canCarry(
        _ cargo: Cargo) -> Bool {
        return cargo.mass <= self.capacity
    }

    @usableFromInline
    internal var capacity: Measurement<UnitMass>

    internal var currentCargo: Cargo?
}
public class CargoShip: Spaceship {
    @inlinable
    public func canCarry(_ cargo: Cargo) -> Bool {
        return cargo.mass <= self.capacity
    }

    @usableFromInline
    internal var capacity: Measurement<UnitMass>

    internal var currentCargo: Cargo?
}
public class CargoShip: Spaceship {
    @inlinable
    public func canCarry(_ cargo: Cargo) -> Bool {
        return cargo.mass <= self.capacity
    }

    @usableFromInline
    internal var capacity: Measurement<UnitMass>

    internal var currentCargo: Cargo?
}
public class CargoShip: Spaceship {
    @inlinable
    public func canCarry(_ cargo: Cargo) -> Bool {
        return cargo.mass <= self.capacity
    }

    @usableFromInline
    internal var capacity: Measurement<UnitMass>

    internal var currentCargo: Cargo?
}

// swift-interface-format-version: 1.0
// swift-compiler-version: Swift version 5.1
// swift-module-flags: -target arm64-apple-ios13.0
// -enable-library-evolution
// -swift-version 5 -O
// -module-name FlightKit
public class CargoShip: Spaceship {
    @inlinable
    public func canCarry(_ cargo: Cargo) -> Bool {
        return cargo.mass <= self.capacity
    }

    @usableFromInline
    internal var capacity: Measurement<UnitMass>

    internal var currentCargo: Cargo?
}
public class CargoShip: Spaceship {
    @inlinable
    public func canCarry(_ cargo: Cargo) -> Bool {
        return cargo.mass <= self.capacity
    }

    @usableFromInline
    internal var capacity: Measurement<UnitMass>

    internal var currentCargo: Cargo?
}
public class CargoShip: Spaceship {
    @inlinable
    public func canCarry(_ cargo: Cargo) -> Bool {
        return cargo.mass <= self.capacity
    }

    @usableFromInline
    internal var capacity: Measurement<UnitMass>

    internal var currentCargo: Cargo?
}
“But what happens if I change it?”
```swift
public class CargoShip: Spaceship {
    @inlinable
    public func canCarry(_ cargo: Cargo) -> Bool {
        return cargo.mass <= self.capacity
    }

    @usableFromInline
    internal var capacity: Measurement<UnitMass>

    internal var currentCargo: Cargo?
}
```
```swift
public class CargoShip: Spaceship {
    @inlinable
    public func canCarry(_ cargo: Cargo) -> Bool {
        return cargo.mass <= self.capacity
    }

    @usableFromInline
    internal var capacity: Measurement<UnitMass>

    internal var currentCargo: Cargo?
}
```
public class CargoShip: Spaceship {
    @inlinable
    public func canCarry(_ cargo: Cargo) -> Bool {
        return cargo.mass <= self.capacity &&
            !cargo.isRadioactive
    }
}

@usableFromInline
internal var capacity: Measurement<UnitMass>

internal var currentCargo: Cargo?
cargo.mass <= self.capacity && !cargo.isRadioactive

cargo.mass <= self.capacity
cargo.mass <= self.capacity && !cargo.isRadioactive

cargo.mass <= self.capacity
cargo.mass <= self.capacity && !cargo.isRadioactive

cargo.mass <= self.capacity
cargo.mass \leq \text{self.capacity} \land \neg \text{cargo.isRadioactive}
cargo.mass <= self.capacity && !cargo.isRadioactive
cargo.mass <= self.capacity && !cargo.isRadioactive
cargo.mass <= self.capacity && !cargo.isRadioactive

cargo.mass <= self.capacity
cargo.mass <= self.capacity && !cargo.isRadioactive

cargo.mass <= self.capacity
cargo.mass <= self.capacity && !cargo.isRadioactive

cargo.mass <= self.capacity
cargo.mass <= self.capacity && !cargo.isRadioactive

cargo.mass <= self.capacity
cargo.mass <= self.capacity && !cargo.isRadioactive

cargo.mass <= self.capacity
Changing an `@inlinable` Function

Rule of thumb — don’t change the output or observable behavior
• Better algorithms are okay
@frozen Enums

```java
public enum FlightPlanKind {
    case oneWay
    case roundTrip
}
```
public enum FlightPlanKind {
    case oneWay
    case roundTrip
}

switch kind {
    case .oneWay:
        print("See you there!")
    case .roundTrip:
        print("See you in a few weeks!")
    @unknown default:
        print("Have a good flight!")
}

public enum FlightPlanKind {
    case oneWay
    case roundTrip
}

switch kind {
    case .oneWay:
        print("See you there!")
    case .roundTrip:
        print("See you in a few weeks!")
    @unknown default:
        print("Have a good flight!")
}
@frozen Enums

```swift
public enum FlightPlanKind {
    case oneWay
    case roundTrip
}
```

```swift
switch kind {
    case .oneWay:
        print("See you there!")
    case .roundTrip:
        print("See you in a few weeks!")
    @unknown default:
        print("Have a good flight!")
}
```
public enum FlightPlanKind {
    case oneWay
    case roundTrip
}

switch kind {
    case .oneWay:
        print("See you there!")
    case .roundTrip:
        print("See you in a few weeks!")
    @unknown default:
        print("Have a good flight!")
}
public enum FlightPlanKind {
    case oneWay
    case roundTrip
}

switch kind {
    case .oneWay:
        print("See you there!")
    case .roundTrip:
        print("See you in a few weeks!")
    @unknown default:
        print("Have a good flight!")
}
@frozen public enum FlightPlanKind {
    case oneWay
    case roundTrip
}

switch kind {
    case .oneWay:
        print("See you there!")
    case .roundTrip:
        print("See you in a few weeks!")
    @unknown default:
        print("Have a good flight!")
}
```swift
@frozen public enum FlightPlanKind {
    case oneWay
    case roundTrip
}

switch kind {
    case .oneWay:
        print("See you there!")
    case .roundTrip:
        print("See you in a few weeks!")
    @unknown default:
        print("Have a good flight!")
}
```
@frozen Enums

```swift
@frozen public enum FlightPlanKind {
    case oneWay
    case roundTrip
}
```

Promise not to add new cases

```swift
switch kind {
    case .oneWay:
        print("See you there!")
    case .roundTrip:
        print("See you in a few weeks!")
    @unknown default:
        print("Have a good flight!")
}
```
@frozen Enums

```swift
@frozen public enum FlightPlanKind {
    case oneWay
    case roundTrip
}
```

Promise not to add new cases

Clients won’t write default in switches

```
switch kind {
    case .oneWay:
        print("See you there!")
    case .roundTrip:
        print("See you in a few weeks!")
    @unknown default:
        print("Have a good flight!")
}
```
@frozen Enums

```swift
@frozen public enum FlightPlanKind {
    case oneWay
    case roundTrip
}
```

```
switch kind {
    case .oneWay:
        print("See you there!")
    case .roundTrip:
        print("See you in a few weeks!")
}
```

Promise not to add new cases

Clients won’t write `default` in switches
**Enums**

Promise not to add new cases

Clients won’t write `default` in switches

Better performance, especially when no cases have associated values

```swift
@frozen public enum FlightPlanKind {
    case oneWay
    case roundTrip
}

switch kind {
    case .oneWay:
        print("See you there!")
    case .roundTrip:
        print("See you in a few weeks!")
}
```
Changing `@frozen` Enums

```swift
@frozen public enum FlightPlanKind {
    case oneWay
    case roundTrip
    case multiHop([Location])
}

switch kind {
    case .oneWay:
        print("See you there!")
    case .roundTrip:
        print("See you in a few weeks!")
}
```
@frozen public enum FlightPlanKind {
    case oneWay
    case roundTrip
    case multiHop([Location])
}

switch kind {
    case .oneWay:
        print("See you there!")
    case .roundTrip:
        print("See you in a few weeks!")
}

Changing `@frozen` Enums

```swift
@frozen public enum FlightPlanKind {
    case oneWay
    case roundTrip
    case multiHop([Location])
}
```

```swift
switch kind {
    case .oneWay:
        print("See you there!")
    case .roundTrip:
        print("See you in a few weeks!")
}
```
Changing `@frozen` Enums

```swift
@frozen public enum FlightPlanKind {
    case oneWay
    case roundTrip
    case multiHop([Location])
}
```

```swift
switch kind {
    case .oneWay:
        print("See you there!"
    case .roundTrip:
        print("See you in a few weeks!")
}
```
public struct Coordinates {
    public var radius, azimuth, inclination: Double
    public init(r: Double, alpha: Double, theta: Double) {
    ...
    }
}
public struct Coordinates {
    public var radius, azimuth, inclination: Double
    public init(r: Double, alpha: Double, theta: Double) {...}
}

@frozen Structs

How big?
public struct Coordinates {
  public var radius, azimuth, inclination: Double
  public init(r: Double, alpha: Double, theta: Double) {...}
}
public struct Coordinates {
    public var radius, azimuth, inclination: Double
    public init(r: Double, alpha: Double, theta: Double) {...}
}
@frozen public struct Coordinates {
    public var radius, azimuth, inclination: Double
    public init(r: Double, alpha: Double, theta: Double) {...}
}

@frozen public struct Coordinates {
    public var radius, azimuth, inclination: Double
    public init(r: Double, alpha: Double, theta: Double) {...}
}
@frozen public struct Coordinates {
    public var radius, azimuth, inclination: Double
    public init(r: Double, alpha: Double, theta: Double) {…}
}

Promise not to add or reorder stored properties
@frozen public struct Coordinates {
    public var radius, azimuth, inclination: Double
    public init(r: Double, alpha: Double, theta: Double) {...}
}

Promise not to add or reorder stored properties

Types of stored properties must be public or @usableFromInline
@frozen public struct Coordinates {
    public var radius, azimuth, inclination: Double
    @inlinable public init(r: Double, alpha: Double, theta: Double) {
    }
}

Promise not to add or reorder stored properties

Types of stored properties must be public or @usableFromInline

Allows inlinable initializers
@frozen public struct Coordinates {
    public var radius, azimuth, inclination: Double
    @inlinable public init(r: Double, alpha: Double, theta: Double) {...}
}

Promise not to add or reorder stored properties

Types of stored properties must be public or @usableFromInline

Allows inlinable initializers
Flexibility Is the Default
Flexibility Is the Default

Breaking changes are inconvenient
Flexibility Is the Default

Breaking changes are inconvenient

These attributes only affect client code
Flexibility Is the Default

Breaking changes are inconvenient.

These attributes only affect client code.

Profile before reaching for `@inlinable` or `@frozen`
Helping Your Clients
Entitlements
Entitlements
Entitlements

Document your requirements
Entitlements

Document your requirements

Minimize entitlement requests
Entitlements

Document your requirements
Minimize entitlement requests
Handle denial gracefully
Dependencies
Dependencies

Document your dependencies
Dependencies

Document your dependencies

Minimize your dependencies
Dependencies

Document your dependencies

Minimize your dependencies

Dependencies must use “Build Libraries for Distribution”
Binary Frameworks Cannot Depend on Packages
Binary Frameworks Cannot Depend on Packages
Binary Frameworks Cannot Depend on Packages
Binary Frameworks Cannot Depend on Packages
Binary Frameworks Cannot Depend on Packages
Your Objective-C Interface

@import FlightKit;
Your Objective-C Interface

@import FlightKit;

FlightKit.h
FlightKit-Swift.h
If Your Swift Code Has No Objective-C API

```swift
@import FlightKit;
```

- FlightKit.h
- FlightKit-Swift.h
If Your Swift Code Has No Objective-C API

@import FlightKit;

FlightKit.h
FlightKit-Swift.h

**Swift Compiler - General**

<table>
<thead>
<tr>
<th>Setting</th>
<th>FlightKit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install Objective-C Compatibility Header</td>
<td>No</td>
</tr>
</tbody>
</table>
If Your Swift Code Has No Objective-C API

@import FlightKit;
If Your Framework Has No Objective-C API

@import FlightKit;
If Your Framework Has No Objective-C API

```objective-c
@import FlightKit;
```

FlightKit.h

<table>
<thead>
<tr>
<th>Packaging</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Setting</strong></td>
</tr>
<tr>
<td>Defines Module</td>
</tr>
</tbody>
</table>
If Your Framework Has No Objective-C API

@import FlightKit;

FlightKit.h

Packaging

<table>
<thead>
<tr>
<th>Setting</th>
<th>FlightKit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defines Module</td>
<td>No</td>
</tr>
</tbody>
</table>
If Your Framework Has No Objective-C API

```Objective-C
@import FlightKit;
```
Summary
Summary

XCFrameworks — for distributing multiple framework variants

"Build Libraries for Distribution" build setting

Framework owners — know your responsibilities
## Swift Open Hours

<table>
<thead>
<tr>
<th>Event</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Building, Signing, and Distributing Lab</td>
<td>Thursday, 3:00</td>
</tr>
<tr>
<td></td>
<td>Friday, 9:00</td>
</tr>
</tbody>
</table>

More Information

developer.apple.com/wwdc19/416