What's New in Foundation for Swift

Session 207

Tony Parker Foundation, Apple
Michael LeHew Foundation, Apple

What's New in Foundation for Swift

Swift API design guidelines

Improved Objective-C import

New value types

New Swift-specific API

API Design Guidelines

Safe

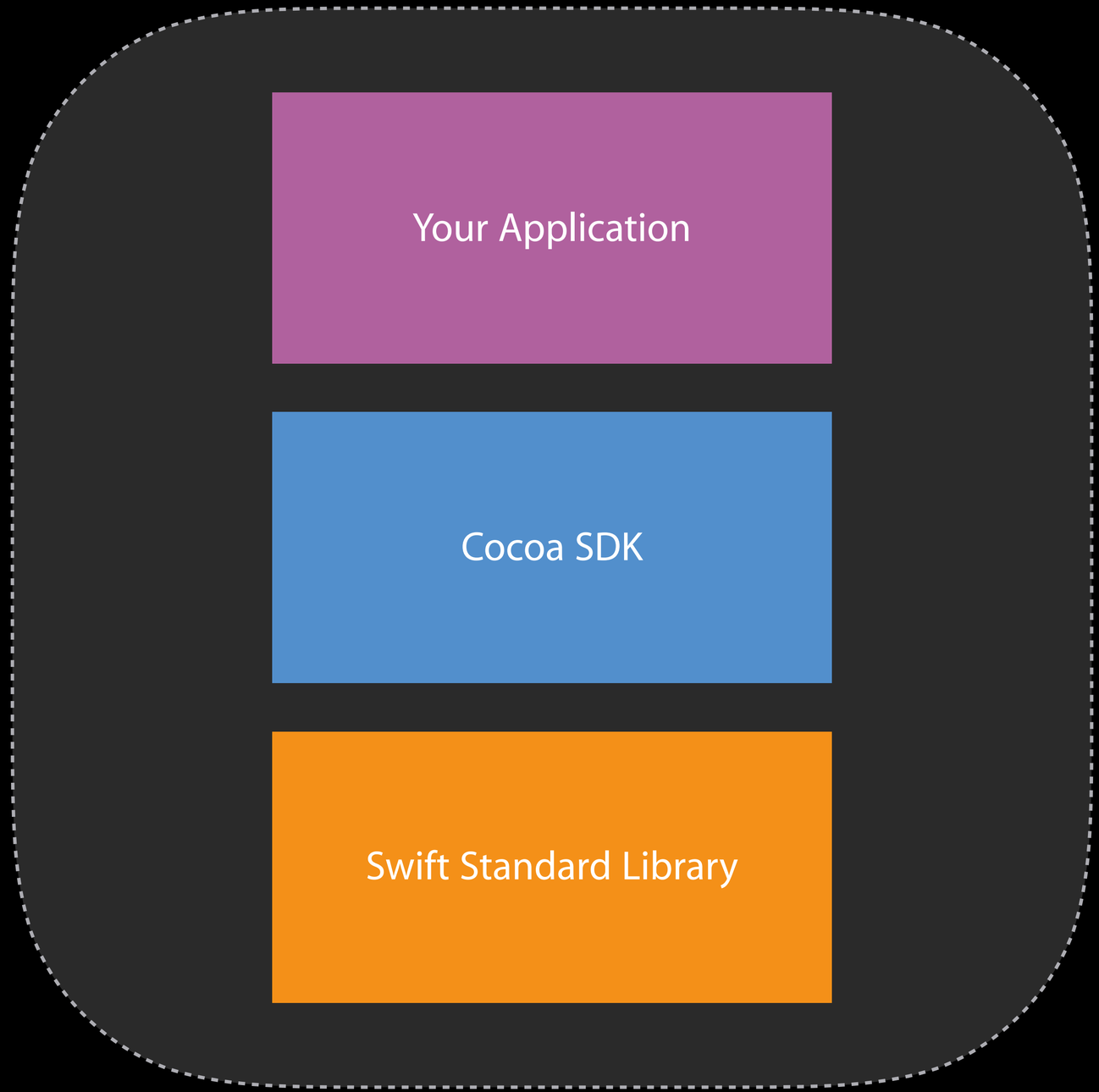
Fast

Expressive

Swift Standard Library

Your Application

Swift Standard Library


Your Application
Cocoa SDK
Swift Standard Library

Safe

Fast

Expressive

Consistent Experience

Consistent Experience

Libraries

Huge number of features

Widespread adoption

Battle-tested implementation

Consistent naming conventions

Continuous development

Consistent Experience

Libraries	Language
Huge number of features	Generics
Widespread adoption	Built-in support for mutation
Battle-tested implementation	Protocol extensions
Consistent naming conventions	Function overloading
Continuous development	Default argument values

Swift Evolution

Swift Evolution

SE-0023 API Design Guidelines

Swift Evolution

SE-0023 API Design Guidelines

SE-0006 Apply API Design Guidelines to Standard Library

Swift Evolution

SE-0023 API Design Guidelines

SE-0006 Apply API Design Guidelines to Standard Library

SE-0005 Better Translation of Objective-C APIs into Swift

Swift Evolution

SE-0023 API Design Guidelines

SE-0006 Apply API Design Guidelines to Standard Library

SE-0005 Better Translation of Objective-C APIs into Swift

Swift API Design Guidelines — Presidio — Tuesday 10:00AM

What’s Next

What's Next

Swift's goals go beyond naming

What's Next

Swift's goals go beyond naming

Mutability model is a key part of language

What's Next

Swift's goals go beyond naming

Mutability model is a key part of language

Turned attention to Foundation

Why Foundation?

Why Foundation?

Unique spot in the SDK

Why Foundation?

Unique spot in the SDK

Low level

- Used everywhere

Why Foundation?

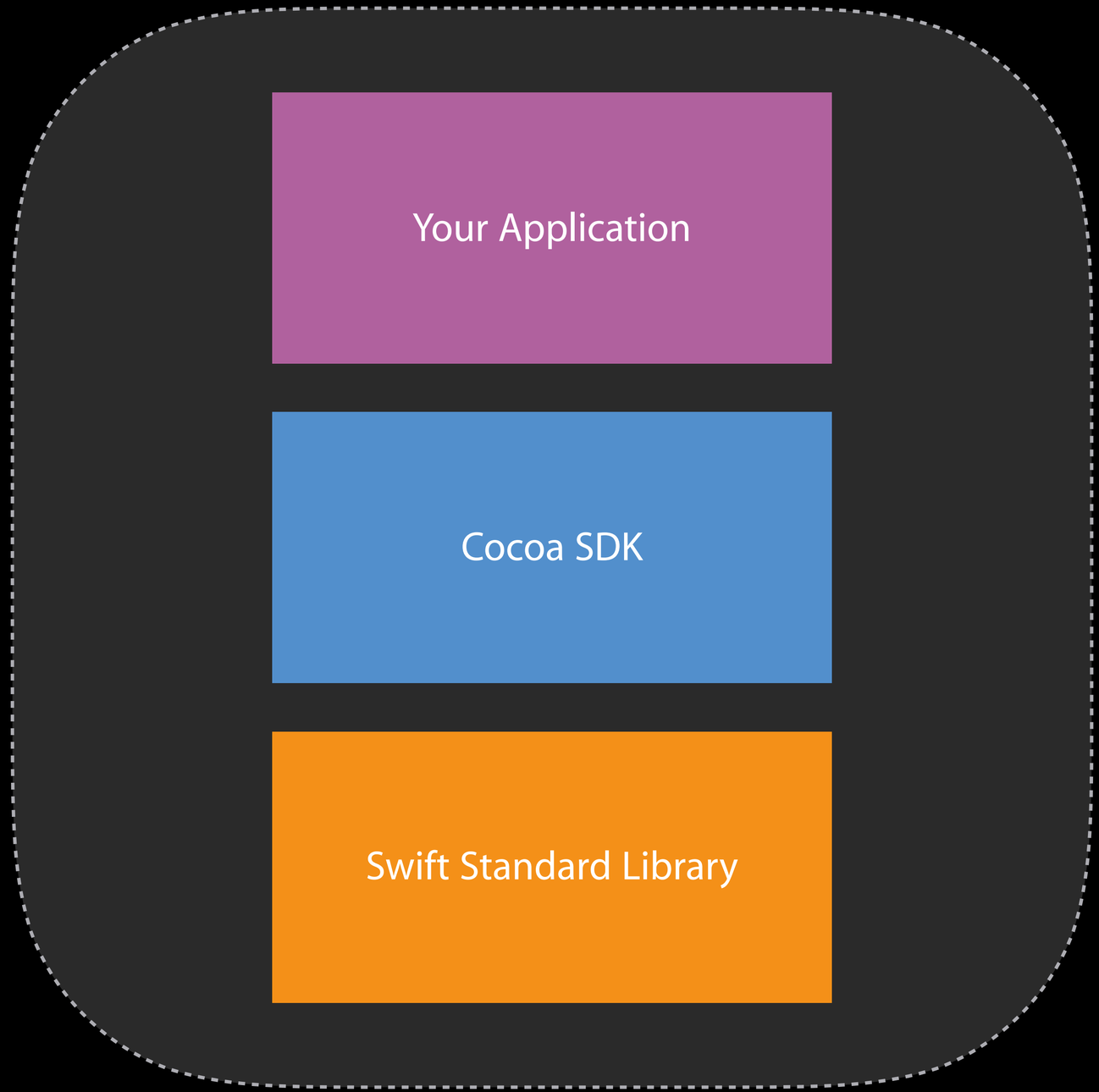
Unique spot in the SDK

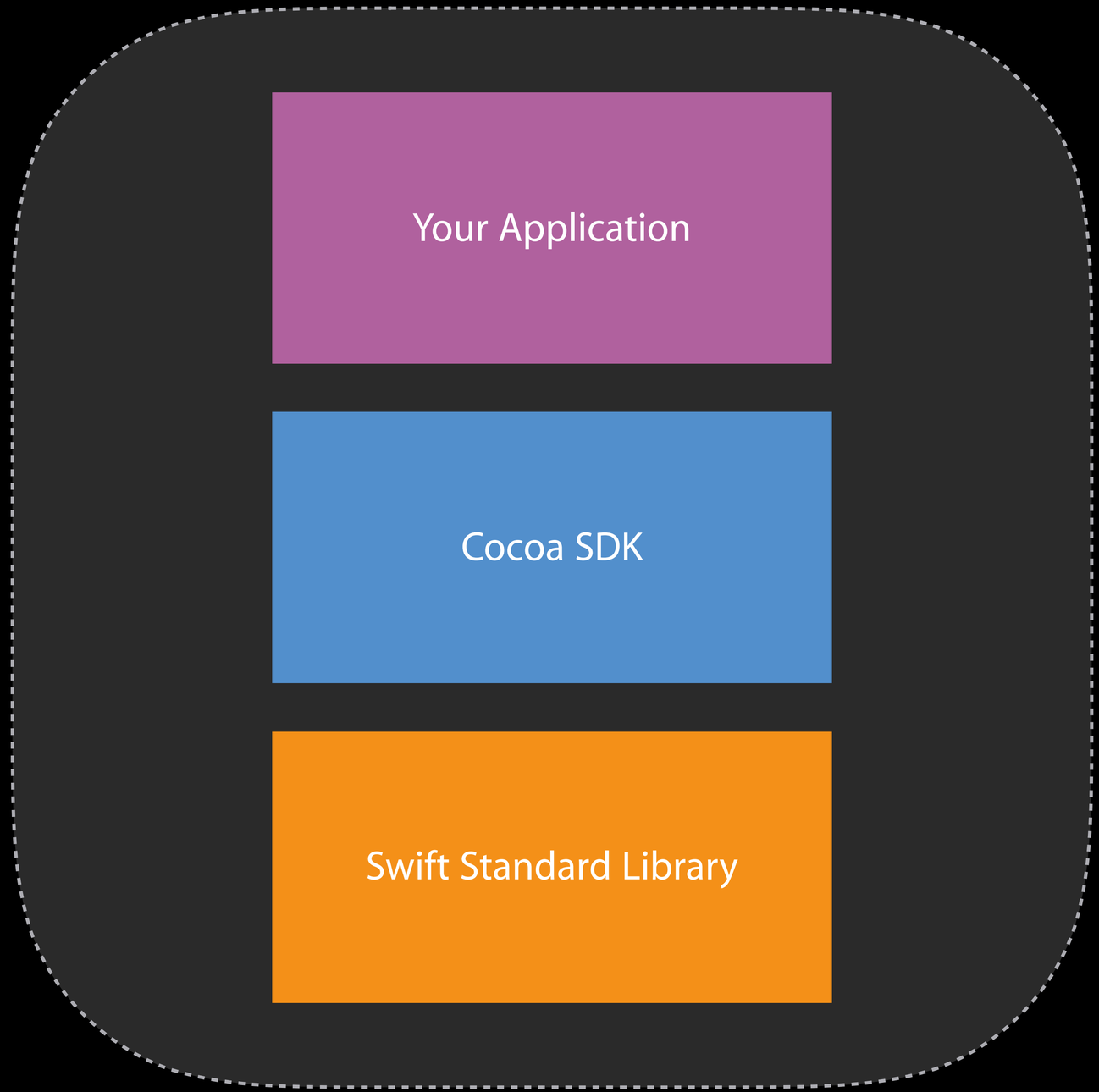
Low level

- Used everywhere

High level

- Establishes common types and design patterns


Your Application
Cocoa SDK
Swift Standard Library


Your Application
Cocoa SDK
Swift Standard Library

Your Application

Swift Standard Library

Your Application

CloudKit | UIKit | WebKit

MapKit | HomeKit | PassKit

SceneKit | HealthKit | WatchKit

SpriteKit | CoreData | MetalKit

Swift Standard Library

Your Application

CloudKit | UIKit | WebKit

MapKit | HomeKit | PassKit

SceneKit | HealthKit | WatchKit

SpriteKit | CoreData | MetalKit

Foundation

Swift Standard Library

Why Foundation?

Leverage point

Why Foundation?

Leverage point

Home of many value types

Why Foundation?

Leverage point

Home of many value types

Evolution over revolution

Foundation Evolution

Foundation Evolution

SE-0069 Mutability and Foundation Value Types

Foundation Evolution

SE-0069 Mutability and Foundation Value Types

SE-0086 Drop NS Prefix in Swift Foundation

Foundation API Improvements

Value semantics

Further naming improvements

Adoption of standard library protocols

Additional type safety

Swift-specific features

Value Types

Value Types

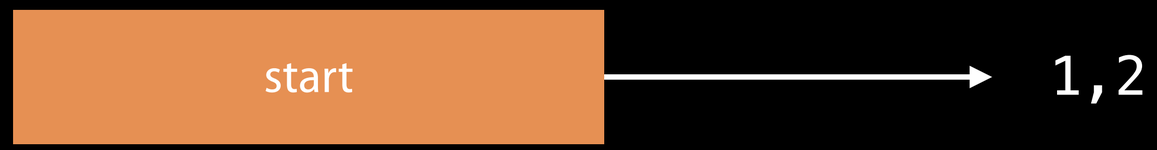
Copy content on assignment or when passed as parameter

```
let start = CGPoint(x: 1, y: 2)
var end = start
end.x += 8
```

Value Types

Copy content on assignment or when passed as parameter

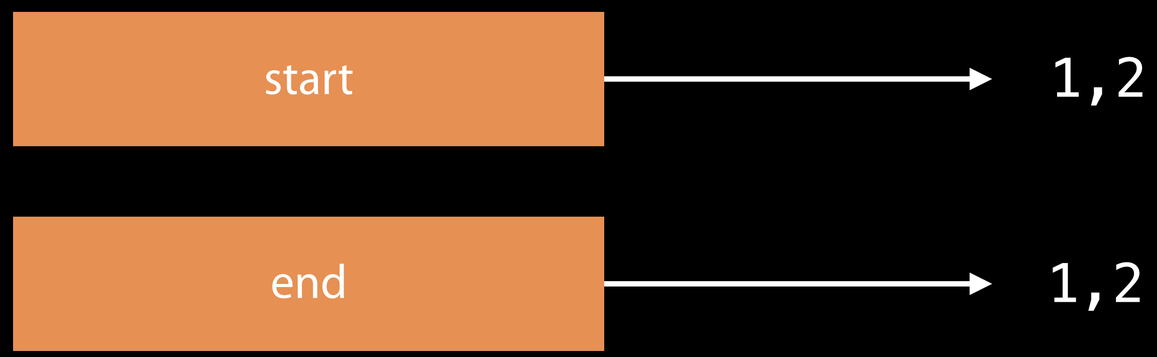
```
let start = CGPoint(x: 1, y: 2)
var end = start
end.x += 8
```



Value Types

Copy content on assignment or when passed as parameter

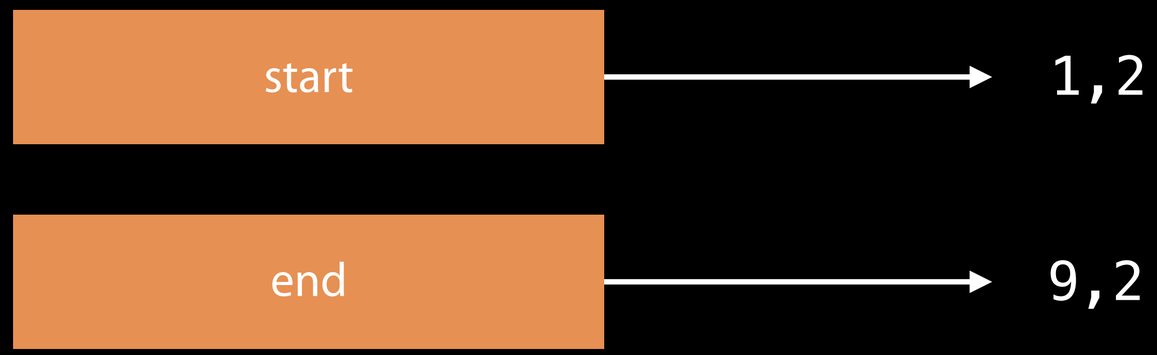
```
let start = CGPoint(x: 1, y: 2)
var end = start
end.x += 8
```



Value Types

Copy content on assignment or when passed as parameter

```
let start = CGPoint(x: 1, y: 2)
var end = start
end.x += 8
```



Reference Types

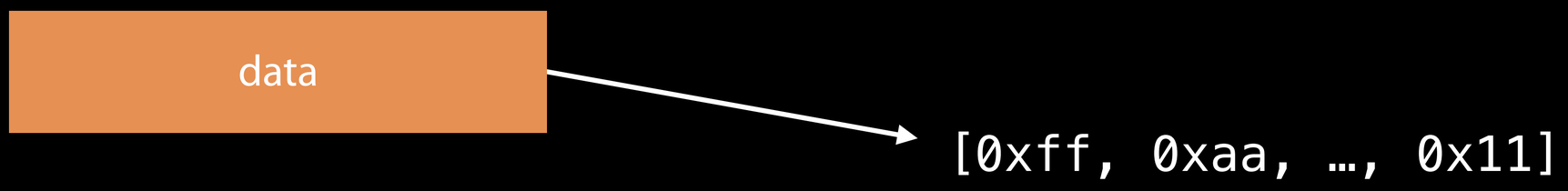
Share content by default

```
let data = NSMutableData(withContentsOf: file1)
var otherData = data
otherData.append(NSData(withContentsOf: file2)
```

Reference Types

Share content by default

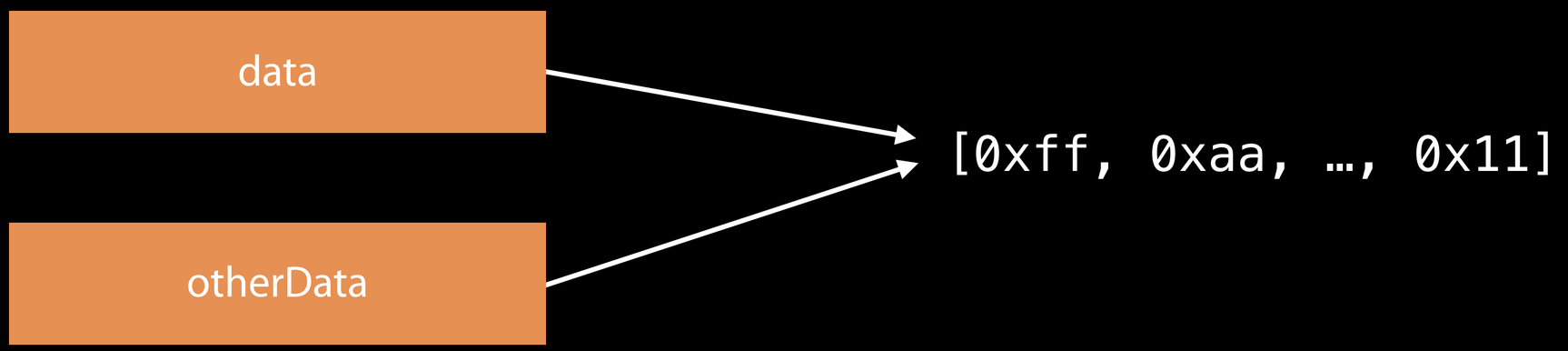
```
let data = NSMutableData(withContentsOf: file1)
var otherData = data
otherData.append(NSData(withContentsOf: file2)
```



Reference Types

Share content by default

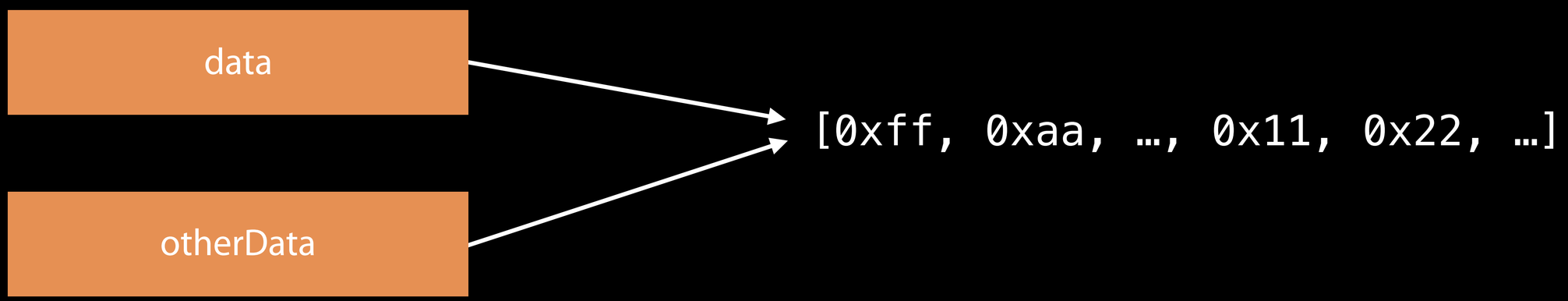
```
let data = NSMutableData(withContentsOf: file1)
var otherData = data
otherData.append(NSData(withContentsOf: file2)
```



Reference Types

Share content by default

```
let data = NSMutableData(withContentsOf: file1)
var otherData = data
otherData.append(NSData(withContentsOf: file2)
```



Value vs. Reference

Neither is better—just used in different ways

Value vs. Reference

Neither is better—just used in different ways

Object identity vs. stored contents

Object Identity

Object Identity

OperationQueue.main

```
class OperationQueue : NSObject {
    class var main: OperationQueue
}
```

Object Identity

OperationQueue.main

```
class OperationQueue : NSObject {
    class var main: OperationQueue
}
```

URLSession.delegate

```
public protocol URLSessionDataDelegate : URLSessionTaskDelegate {
    optional public func urlSession(_ session: URLSession,
                                    dataTask: URLSessionDataTask,
                                    didReceive data: Data)
}
```

Object Identity

OperationQueue.main

```
class OperationQueue : NSObject {
    class var main: OperationQueue
}
```

URLSession.delegate

```
public protocol URLSessionDataDelegate : URLSessionTaskDelegate {
    optional public func urlSession(_ session: URLSession,
                                    dataTask: URLSessionDataTask,
                                    didReceive data: Data)
}
```

Stored Contents

Stored Contents

Date

```
public struct Date : Comparable, Equatable {
   private var _time : Double
}
```

Stored Contents

Date

```
public struct Date : Comparable, Equatable {
   private var _time : Double
}
```

Data

```
public struct Data : Equatable, Hashable, RandomAccessCollection, MutableCollection
```

Copy on Write

Copy on Write

```
let data = Data(contentsOf: file1)
```

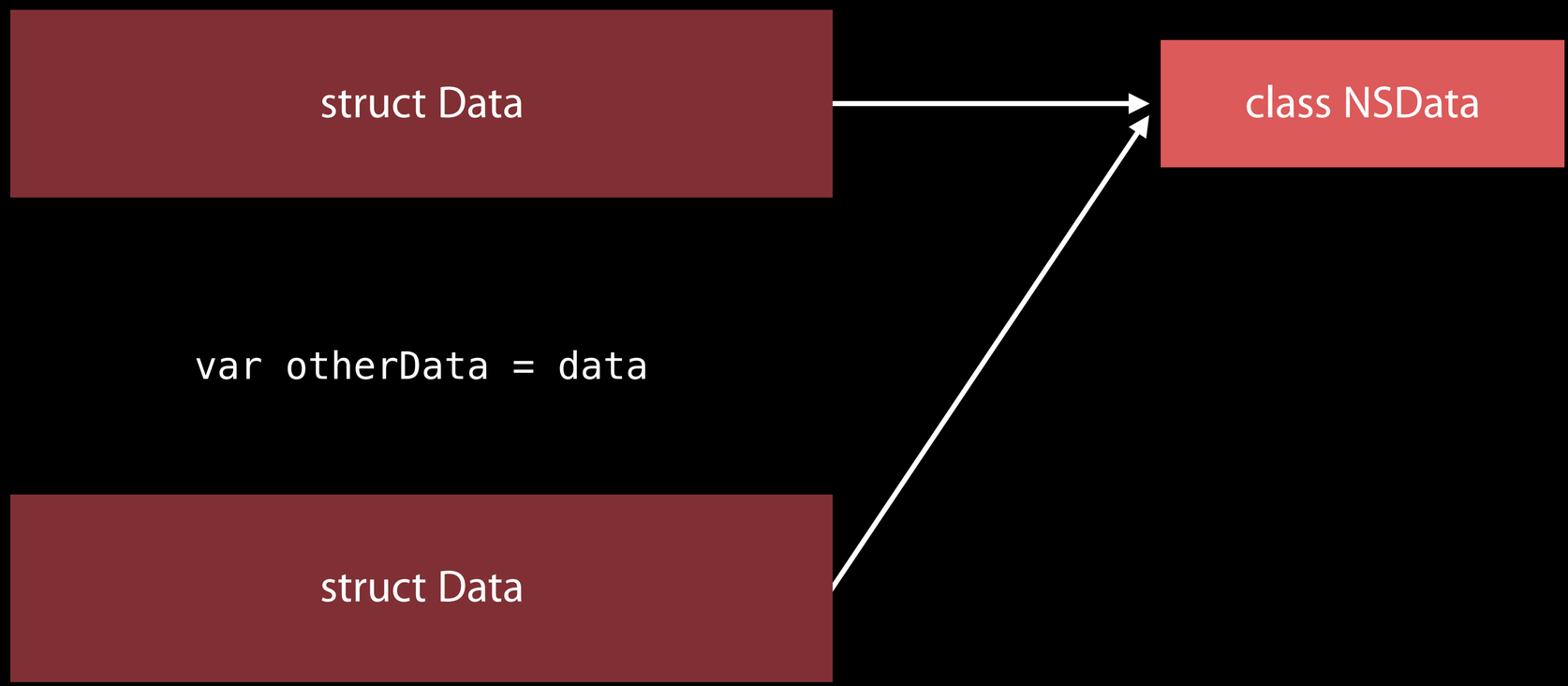
struct Data

Copy on Write

```
let data = Data(contentsOf: file1)
```

Copy on Write

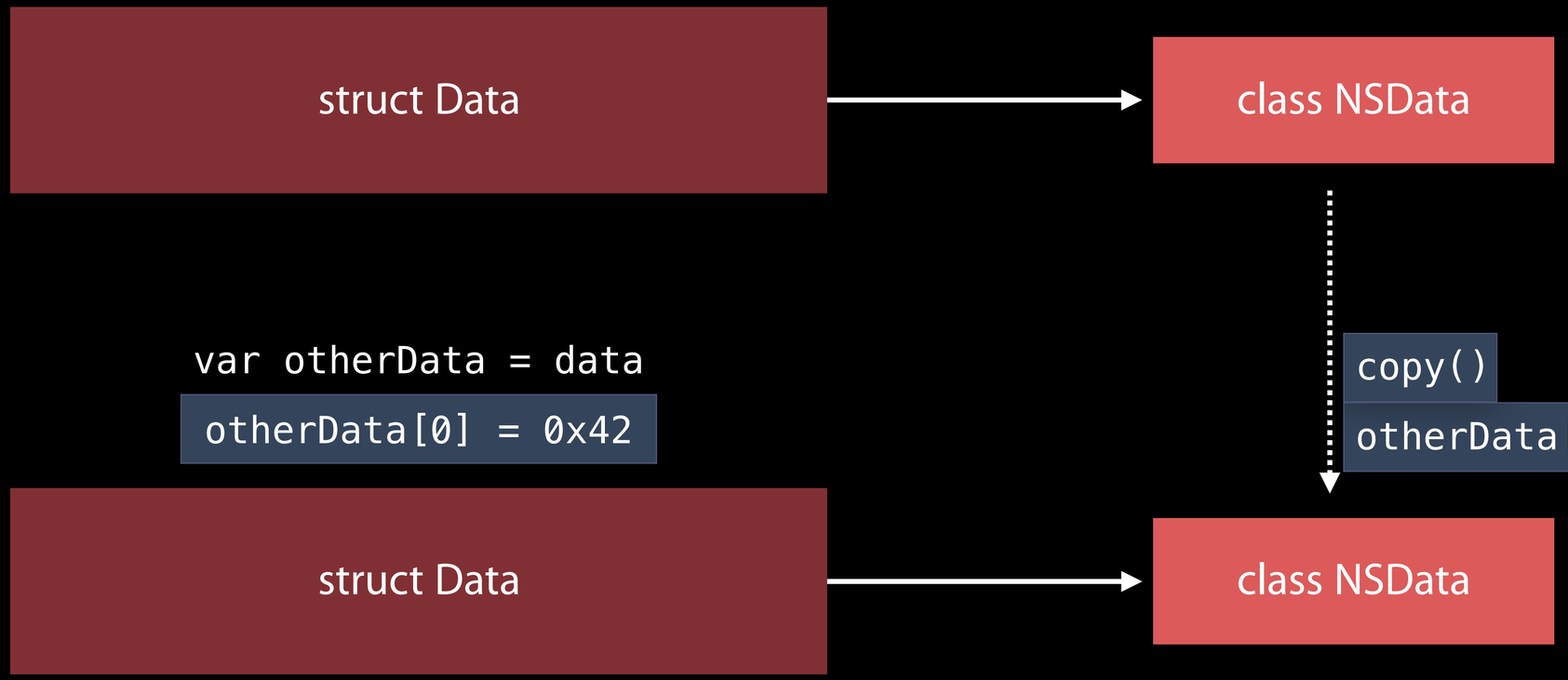
```
let data = Data(contentsOf: file1)
```




```
var otherData = data
```

Copy on Write

```
let data = Data(contentsOf: file1)
```




```
var otherData = data
otherData[0] = 0x42
```

```
let data = Data(contentsOf: file1)
```

struct Data

class NSData

```
var otherData = data
otherData[0] = 0x42
```

```
copy()
otherData[0] = 0x42
```

struct Data

class NSData

```
let data = Data(contentsOf: file1)
```

```
let data = Data(contentsOf: file1)
```


```
otherData[1] = 0x43
otherData[2] = 0x44
```

```
otherData[1] = 0x43
otherData[2] = 0x44
```

```
let data = Data(contentsOf: file1)
```


```
otherData[1] = 0x43
otherData[2] = 0x44
```

```
otherData[1] = 0x43
otherData[2] = 0x44
```

New Value Types

NEW

AffineTransform

CharacterSet

Data

Date

DateComponents

DateInterval *(new)*

Decimal *(improved)*

IndexPath

IndexSet

Measurement *(new)*

Notification

PersonNameComponents

URL

URLComponents

URLRequest

URLQueryItem

UUID

API Exploration

Michael LeHew Foundation, Apple

Nested Enumerations

Objective-C constants namespace by convention

Nested Enumerations

Objective-C constants namespace by convention

```
typedef NS_ENUM(NSUInteger, NSNumberFormatterStyle) { ... }
typedef NS_ENUM(NSUInteger, NSNumberFormatterBehavior) { ... }
typedef NS_ENUM(NSUInteger, NSNumberFormatterPadPosition) { ... }
typedef NS_ENUM(NSUInteger, NSNumberFormatterRoundingMode) { ... }
```

Nested Enumerations

Objective-C constants namespace by convention

```
typedef NS_ENUM(NSUInteger, NSNumberFormatterStyle) { ... }
typedef NS_ENUM(NSUInteger, NSNumberFormatterBehavior) { ... }
typedef NS_ENUM(NSUInteger, NSNumberFormatterPadPosition) { ... }
typedef NS_ENUM(NSUInteger, NSNumberFormatterRoundingMode) { ... }

// Swift 2.2
public enum NSNumberFormatterStyle : UInt { ... }
public enum NSNumberFormatterBehavior : UInt { ... }
public enum NSNumberFormatterPadPosition : UInt { ... }
public enum NSNumberFormatterRoundingMode : UInt { ... }
```

Nested Enumerations

NEW

Swift allows for nested types

Nested Enumerations

NEW

Swift allows for nested types

```
// Swift 3
public class NumberFormatter {
    public enum style { ... }
    public enum behavior { ... }
    public enum padPosition { ... }
    public enum roundingMode { ... }
}
```

Strongly Typed String Enumerations

Many Foundation APIs use families of string constants

Strongly Typed String Enumerations

Many Foundation APIs use families of string constants

```
NSString *const NSProcessInfoThermalStateDidChangeNotification;
NSString *const NSTaskDidTerminateNotification;
NSString *const NSCalendarDayChangedNotification;
```

Strongly Typed String Enumerations

Many Foundation APIs use families of string constants

```
NSString *const NSProcessInfoThermalStateDidChangeNotification;
NSString *const NSTaskDidTerminateNotification;
NSString *const NSCalendarDayChangedNotification;
```

```
NSString *const NSURLIsRegularFileKey;
NSString *const NSURLCreationDateKey;
NSString *const NSURLVolumeMaximumFileSizeKey;
```

Strongly Typed String Enumerations

NEW

Objective-C uses the new types

```
NSString *const NSProcessInfoThermalStateDidChangeNotification;
NSString *const NSTaskDidTerminateNotification;
NSString *const NSCalendarDayChangedNotification;
```

```
NSString *const NSURLIsRegularFileKey;
NSString *const NSURLCreationDateKey;
NSString *const NSURLVolumeMaximumFileSizeKey;
```

Strongly Typed String Enumerations

NEW

Objective-C uses the new types

```
NSNotificationName const NSProcessInfoThermalStateDidChangeNotification;
NSNotificationName const NSTaskDidTerminateNotification;
NSNotificationName const NSCalendarDayChangedNotification;
```

```
NSString *const NSURLIsRegularFileKey;
NSString *const NSURLCreationDateKey;
NSString *const NSURLVolumeMaximumFileSizeKey;
```

Strongly Typed String Enumerations

NEW

Objective-C uses the new types

```
NSNotificationName const NSProcessInfoThermalStateDidChangeNotification;
NSNotificationName const NSTaskDidTerminateNotification;
NSNotificationName const NSCalendarDayChangedNotification;
```

```
NSURLResourceKey const NSURLIsRegularFileKey;
NSURLResourceKey const NSURLCreationDateKey;
NSURLResourceKey const NSURLVolumeMaximumFileSizeKey;
```

Strongly Typed String Enumerations

NEW

String enumerations are extensible

Strongly Typed String Enumerations

NEW

String enumerations are extensible

```
// Objective-C
extern NSNotificationName const MyUserBecameActiveNotification;
```

Strongly Typed String Enumerations

NEW

String enumerations are extensible

```
// Objective-C
extern NSNotificationName const MyUserBecameActiveNotification;
```

```
// Swift 3
public extension Notification.Name {
  public static let userLoggedOut = Notification.Name("UserLoggedOut")
}
```

Strongly Typed String Enumerations

NEW

String enumerations are extensible

```
// Objective-C
extern NSNotificationName const MyUserBecameActiveNotification;
```

```
// Swift 3
public extension Notification.Name {
  public static let userLoggedOut = Notification.Name("UserLoggedOut")
}
```

Strongly Typed String Enumerations

NEW

String enumerations are extensible

```
// Objective-C
extern NSNotificationName const MyUserBecameActiveNotification;
```

```
// Swift 3
public extension Notification.Name {
  public static let userLoggedOut = Notification.Name("UserLoggedOut")
}
let n = Notification(name: .userLoggedOut, object: nil)
```

Class Properties

NEW

Many Foundation APIs associate state with a type

Class Properties

NEW

Many Foundation APIs associate state with a type

```
// Objective-C (conventional class properties)
@interface NSUserDefaults
+ (NSUserDefaults *)standardUserDefaults;
@end
```

Class Properties

NEW

Many Foundation APIs associate state with a type

```
// Objective-C (conventional class properties)
@interface NSUserDefaults
+ (NSUserDefaults *)standardUserDefaults;
@end
```

```
// Objective-C (language supported class properties)
@interface NSUserDefaults
@property (class, readonly, strong) standardUserDefaults;
@end
```

Class Properties

NEW

Many Foundation APIs associate state with a type

```
// Objective-C (conventional class properties)
@interface NSUserDefaults
+ (NSUserDefaults *)standardUserDefaults;
@end
```

```
// Objective-C (language supported class properties)
@interface NSUserDefaults
@property (class, readonly, strong) standardUserDefaults;
@end
```

Class Properties

NEW

Objective-C class properties appear as Swift class properties

```
// Swift 2.2
public class NSUserDefaults {
  public class func standardUserDefaults() -> NSUserDefaults
}
```

Class Properties

NEW

Objective-C class properties appear as Swift class properties

```
// Swift 2.2
public class NSUserDefaults {
  public class func standardUserDefaults() -> NSUserDefaults
}
```

```
// Swift 3 (almost)
public class UserDefaults {
   public class var standardUserDefaults: UserDefaults
}
```

Class Properties

NEW

Objective-C class properties appear to Swift class properties

```
// Swift 2.2
public class NSUserDefaults {
  public class func standardUserDefaults() -> NSUserDefaults
}
```

```
// Swift 3
public class UserDefaults {
   public class var standard: UserDefaults
}
```

Value Types

Date

Measurement

URLComponents

```
// Value Types: Date
```

```
// Value Types: Date


// Swift 2.2
func whenToLeave() -> NSDate { ... }
```

```swift
// Value Types: Date

// Swift 2.2
func whenToLeave() -> NSDate { ... }
let date = whenToLeave()
```

```swift
// Value Types: Date

// Swift 2.2
func whenToLeave() -> NSDate { ... }
let date = whenToLeave()
let reminder = date.dateByAddingTimeInterval(-5.0 * 60.0)
```

```swift
// Value Types: Date

// Swift 2.2
func whenToLeave() -> NSDate { ... }
let date = whenToLeave()
let reminder = date.dateByAddingTimeInterval(-5.0 * 60.0)
```

Allocation Count: 1

```swift
// Value Types: Date

// Swift 2.2
func whenToLeave() -> NSDate { ... }
let date = whenToLeave()
let reminder = date.dateByAddingTimeInterval(-5.0 * 60.0)
```

Allocation Count: 2

```swift
// Date as a Value Type

// Swift 2.2
func whenToLeave() -> NSDate { ... }
let date = whenToLeave()
let reminder = date.dateByAddingTimeInterval(-5.0 * 60.0)



// Swift 3
```

NEW

```swift
// Date as a Value Type

// Swift 2.2
func whenToLeave() -> NSDate { ... }
let date = whenToLeave()
let reminder = date.dateByAddingTimeInterval(-5.0 * 60.0)


// Swift 3
func whenToLeave() -> Date { ... }
```

NEW

```
// Date as a Value Type

// Swift 2.2
func whenToLeave() -> NSDate { ... }
let date = whenToLeave()
let reminder = date.dateByAddingTimeInterval(-5.0 * 60.0)



// Swift 3
func whenToLeave() -> Date { ... }
var date = whenToLeave()
```

NEW

```swift
// Date as a Value Type

// Swift 2.2
func whenToLeave() -> NSDate { ... }
let date = whenToLeave()
let reminder = date.dateByAddingTimeInterval(-5.0 * 60.0)



// Swift 3
func whenToLeave() -> Date { ... }
var date = whenToLeave()
date.addTimeInterval(-5.0 * 60.0)
```

NEW

```swift
// Date as a Value Type


// Swift 3
func whenToLeave() -> Date { ... }
var date = whenToLeave()
date.addTimeInterval(-5.0 * 60.0)
```

NEW

```swift
// Date as a Value Type


// Swift 3
func whenToLeave() -> Date { ... }
let when = whenToLeave().addingTimeInterval(-5.0 * 60.0)
```

NEW

```swift
// Date as a Value Type

// Swift 3
func whenToLeave() -> Date { ... }
let when = whenToLeave().addingTimeInterval(-5.0 * 60.0)

if Date() < when {
} else {
   print("You're late!")
}
```

NEW

NEW

```
// Date as a Value Type

// Swift 3
func whenToLeave() -> Date { ... }
let when = whenToLeave().addingTimeInterval(-5.0 * 60.0)


if Date() < when {
    timer = Timer(fireDate: when, interval: 0, repeats: false) {
       print("Almost time to go!")
    }
    RunLoop.main.add(timer, forMode: .commonModes)
} else {
   print("You're late!")
}
```

NEW

```
// Date as a Value Type

// Swift 3
func whenToLeave() -> Date { ... }
let when = whenToLeave().addingTimeInterval(-5.0 * 60.0)


if Date() < when {
      timer = Timer(fireDate: when, interval: 0, repeats: false) {
         print("Almost time to go!")
      }
      RunLoop.main.add(timer, forMode: .commonModes)
} else {
   print("You're late!")
}
```

Value Types

NEW

Measurement

```
// Swift 3
```

Value Types

NEW

Measurement

```
// Swift 3
let street1 = Measurement(value: 73, UnitLength.meters)
```

Value Types

NEW

Measurement

```
// Swift 3
let street1 = Measurement(value: 73, UnitLength.meters)
let street2 = Measurement(value: 67, UnitLength.meters)
```

Value Types

NEW

Measurement

```
// Swift 3
let street1 = Measurement(value: 73, UnitLength.meters)
let street2 = Measurement(value: 67, UnitLength.meters)
var commuteDistance = street1 + street2
```

Value Types

NEW

Measurement

```
// Swift 3
let street1 = Measurement(value: 73, UnitLength.meters)
let street2 = Measurement(value: 67, UnitLength.meters)
var commuteDistance = street1 + street2
commuteDistance.convert(to: UnitLength.yards)
```

Value Types

NEW

Measurement

```
// Swift 3
let street1 = Measurement(value: 73, UnitLength.meters)
let street2 = Measurement(value: 67, UnitLength.meters)
var commuteDistance = street1 + street2
commuteDistance.convert(to: UnitLength.yards)

let speed = commuteDistance.converted(to: UnitSpeed.metersPerSecond)
```

Value Types

NEW

Measurement

```
// Swift 3
let street1 = Measurement(value: 73, UnitLength.meters)
let street2 = Measurement(value: 67, UnitLength.meters)
var commuteDistance = street1 + street2
commuteDistance.convert(to: UnitLength.yards)

let speed = commuteDistance.converted(to: UnitSpeed.metersPerSecond)
```

Cannot convert value of type 'UnitSpeed' to expected argument type 'UnitLength'

Value Types

NEW

Measurement

```
// Swift 3
let street1 = Measurement(value: 73, UnitLength.meters)
let street2 = Measurement(value: 67, UnitLength.meters)
var commuteDistance = street1 + street2
commuteDistance.convert(to: UnitLength.yards)

let speed = commuteDistance.converted(to: UnitSpeed.metersPerSecond)
```

Cannot convert value of type 'UnitSpeed' to expected argument type 'UnitLength'

Measurements and Units — Presidio — Friday 4:00PM

URL Components

URL Components

```
var template = URLComponents()
template.scheme = "https"
template.host = "www.apple.com"
template.path = "/shop/buy-mac"
template.queryItems = [URLQueryItem(name: "step", value: "detail")]
```

URL Components

```
var template = URLComponents()
template.scheme = "https"
template.host = "www.apple.com"
template.path = "/shop/buy-mac"
template.queryItems = [URLQueryItem(name: "step", value: "detail")]

var urls = Array<URLComponents>()
for product in ["MacBook", "MacBook Pro"]  {
    var components = template
    components.queryItems!.append(URLQueryItem(name: "product", value: product))
    urls.append(components)
}
```

URL Components

```
var template = URLComponents()
template.scheme = "https"
template.host = "www.apple.com"
template.path = "/shop/buy-mac"
template.queryItems = [URLQueryItem(name: "step", value: "detail")]

var urls = Array<URLComponents>()
for product in ["MacBook", "MacBook Pro"]  {
    var components = template
    components.queryItems!.append(URLQueryItem(name: "product", value: product))
    urls.append(components)
}
```

URL Components

```
var template = URLComponents()
template.scheme = "https"
template.host = "www.apple.com"
template.path = "/shop/buy-mac"
template.queryItems = [URLQueryItem(name: "step", value: "detail")]

var urls = Array<URLComponents>()
for product in ["MacBook", "MacBook Pro"]  {
    var components = template
    components.queryItems!.append(URLQueryItem(name: "product", value: product))
    urls.append(components)
}
```

URL Components

```
var template = URLComponents()
template.scheme = "https"
template.host = "www.apple.com"
template.path = "/shop/buy-mac"
template.queryItems = [URLQueryItem(name: "step", value: "detail")]

var urls = Array<URLComponents>()
for product in ["MacBook", "MacBook Pro"]  {
    var components = template
    components.queryItems!.append(URLQueryItem(name: "product", value: product))
    urls.append(components)
}
```

URL Components

```
var template = URLComponents()
template.scheme = "https"
template.host = "www.apple.com"
template.path = "/shop/buy-mac"
template.queryItems = [URLQueryItem(name: "step", value: "detail")]

var urls = Array<URLComponents>()
for product in ["MacBook", "MacBook Pro"]  {
    var components = template
    components.queryItems!.append(URLQueryItem(name: "product", value: product))
    urls.append(components)
}
```

URL Components

```
var template = URLComponents()
template.scheme = "https"
template.host = "www.apple.com"
template.path = "/shop/buy-mac"
template.queryItems = [URLQueryItem(name: "step", value: "detail")]

var urls = Array<URLComponents>()
for product in ["MacBook", "MacBook Pro"]  {
    var components = template
    components.queryItems!.append(URLQueryItem(name: "product", value: product))
    urls.append(components)
}
print(urls)
```

URL Components

```
var template = URLComponents()
template.scheme = "https"
template.host = "www.apple.com"
template.path = "/shop/buy-mac"
template.queryItems = [URLQueryItem(name: "step", value: "detail")]

var urls = Array<URLComponents>()
for product in ["MacBook", "MacBook Pro"]  {
    var components = template
    components.queryItems!.append(URLQueryItem(name: "product", value: product))
    urls.append(components)
}
print(urls)
```

https://www.apple.com/shop/buy-mac?step=detail&product=MacBook

https://www.apple.com/shop/buy-mac?step=detail&product=MacBook%20Pro

Protocol Conformance

NEW

Protocol Conformance

NEW

`CharacterSet` and `IndexSet` conform to `SetAlgebra`

Protocol Conformance

NEW

`CharacterSet` and `IndexSet` conform to `SetAlgebra`

`Data` conforms to

- `MutableCollection` with `ElementType = UInt8`
- `RandomAccessCollection`

```
// Collection API for Data
```

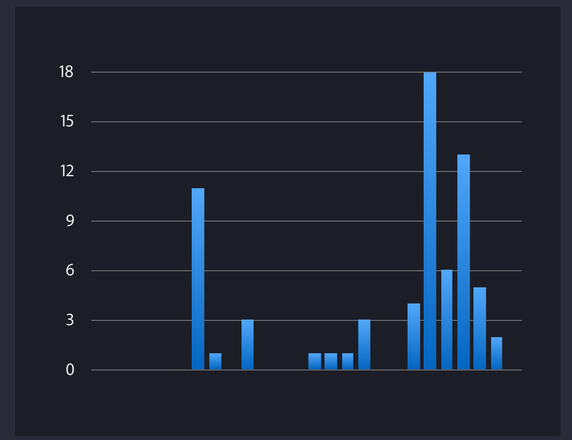
```
 // Collection API for Data

let base64 = "SGVyZSdzIHRvIHRoZSBDcmF6eSBPbmVzISBUaGUgbWlzZm"
             "l0cy4gVGhlIHJlYmVscy4gVGhlIHRyb3VibGVtYWtlcnMu"
```

```
// Collection API for Data

let base64 = "SGVyZSdzIHRvIHRoZSBDcmF6eSBPbmVzISBUaGUgbWlzZm"
             "l0cy4gVGhlIHJlYmVscy4gVGhlIHRyb3VibGVtYWtlcnMu"
let data = Data(base64Encoded: base64)!
```

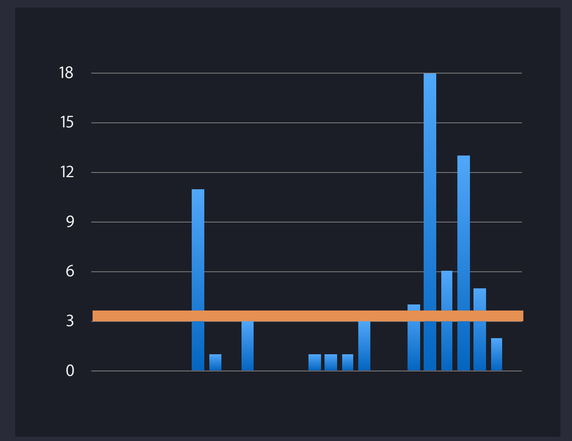
```
 // Collection API for Data

let base64 = "SGVyZSdzIHRvIHRoZSBDcmF6eSBPbmVzISBUaGUgbWlzZm" +
             "l0cy4gVGhlIHJlYmVscy4gVGhlIHRyb3VibGVtYWtlcnMu"
let data = Data(base64Encoded: base64)!

var byteHistogram = Array<Int>(repeating: 0, count: 256)
```

```
 // Collection API for Data

let base64 = "SGVyZSdzIHRvIHRoZSBDcmF6eSBPbmVzISBUaGUgbWlzZm"
           "l0cy4gVGhlIHJlYmVscy4gVGhlIHRyb3VibGVtYWtlcnMu"
let data = Data(base64Encoded: base64)!

var byteHistogram = Array<Int>(repeating: 0, count: 256)
for byte in data {
   byteHistogram[Int(byte)] += 1
}
```

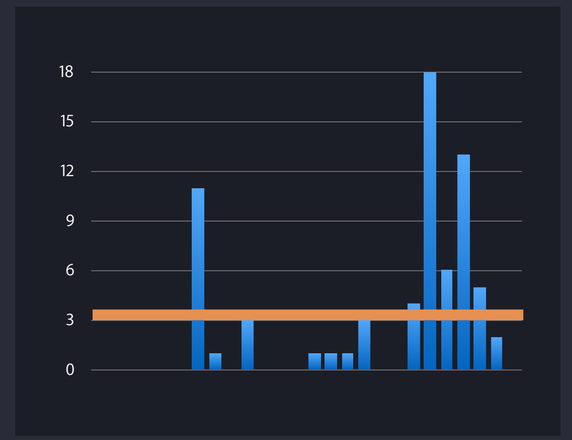
```swift
// Collection API for Data

let base64 = "SGVyZSdzIHRvIHRoZSBDcmF6eSBPbmVzISBUaGUgbWlzZm"
           "l0cy4gVGhlIHJlYmVscy4gVGhlIHRyb3VibGVtYWtlcnMu"
let data = Data(base64Encoded: base64)!

var byteHistogram = Array<Int>(repeating: 0, count: 256)
for byte in data {
   byteHistogram[Int(byte)] += 1
}
```



```swift
// Collection API for Data

let base64 = "SGVyZSdzIHRvIHRoZSBDcmF6eSBPbmVzISBUaGUgbWlzZm"
           "l0cy4gVGhlIHJlYmVscy4gVGhlIHRyb3VibGVtYWtlcnMu"
let data = Data(base64Encoded: base64)!

var byteHistogram = Array<Int>(repeating: 0, count: 256)
for byte in data {
   byteHistogram[Int(byte)] += 1
}
```



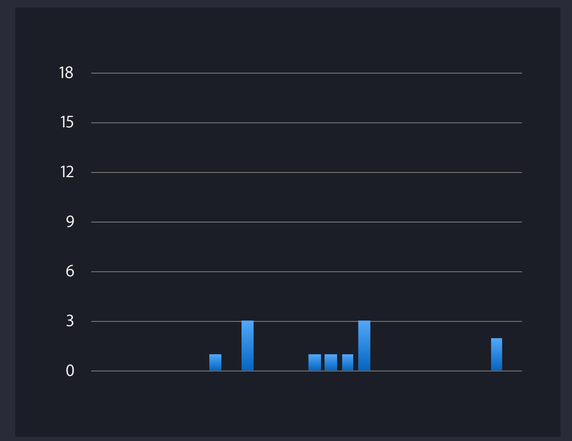
```
 // Collection API for Data

let base64 = "SGVyZSdzIHRvIHRoZSBDcmF6eSBPbmVzISBUaGUgbWlzZm" +
             "l0cy4gVGhlIHJlYmVscy4gVGhlIHRyb3VibGVtYWtlcnMu"
let data = Data(base64Encoded: base64)!

var byteHistogram = Array<Int>(repeating: 0, count: 256)
for byte in data {
   byteHistogram[Int(byte)] += 1
}
```




```
let lessCommonBytes = rawData.filter { (byte : UInt8) in
   return byteHistogram[Int(byte)] <= 3
}
```

```
// Collection API for Data

let base64 = "SGVyZSdzIHRvIHRoZSBDcmF6eSBPbmVzISBUaGUgbWlzZm"
           "l0cy4gVGhlIHJlYmVscy4gVGhlIHRyb3VibGVtYWtlcnMu"
let data = Data(base64Encoded: base64)!

var byteHistogram = Array<Int>(repeating: 0, count: 256)
for byte in data {
   byteHistogram[Int(byte)] += 1
}
```

```
let lessCommonBytes = rawData.filter { (byte : UInt8) in
   return byteHistogram[Int(byte)] <= 3
}
```



```swift
// Collection API for Data

let base64 = "SGVyZSdzIHRvIHRoZSBDcmF6eSBPbmVzISBUaGUgbWlzZm" +
             "l0cy4gVGhlIHJlYmVscy4gVGhlIHRyb3VibGVtYWtlcnMu"
let data = Data(base64Encoded: base64)!

var byteHistogram = Array<Int>(repeating: 0, count: 256)
for byte in data {
   byteHistogram[Int(byte)] += 1
}

let lessCommonBytes = rawData.filter { (byte : UInt8) in
   return byteHistogram[Int(byte)] <= 3
}


var subdata = lessCommonBytes[0..<8]
```

```
[72, 39, 111, 67, 97, 122, 121, 79]
```

```swift
// Collection API for Data

let base64 = "SGVyZSdzIHRvIHRoZSBDcmF6eSBPbmVzISBUaGUgbWlzZm" +
             "l0cy4gVGhlIHJlYmVscy4gVGhlIHRyb3VibGVtYWtlcnMu"
let data = Data(base64Encoded: base64)!

var byteHistogram = Array<Int>(repeating: 0, count: 256)
for byte in data {
   byteHistogram[Int(byte)] += 1
}

let lessCommonBytes = rawData.filter { (byte : UInt8) in
   return byteHistogram[Int(byte)] <= 3
}

var subdata = lessCommonBytes[0..<8]
```

```
[72, 39, 111, 67, 97, 122, 121, 79]
```

```
struct MutableRandomAccessSlice<Data>
```

```swift
// Collection API for Data

let base64 = "SGVyZSdzIHRvIHRoZSBDcmF6eSBPbmVzISBUaGUgbWlzZm" +
             "l0cy4gVGhlIHJlYmVscy4gVGhlIHRyb3VibGVtYWtlcnMu"
let data = Data(base64Encoded: base64)!

var byteHistogram = Array<Int>(repeating: 0, count: 256)
for byte in data {
   byteHistogram[Int(byte)] += 1
}

let lessCommonBytes = rawData.filter { (byte : UInt8) in
   return byteHistogram[Int(byte)] <= 3
}


var subdata = lessCommonBytes[0..<8]
subdata[2] = 42
```

```
[72, 39, 111, 67, 97, 122, 121, 79]
```

```
[72, 39,  42, 67, 97, 122, 121, 79]
```

```
// Value Types and Inheritance
```

```
// Value Types and Inheritance

class AllOnesData : NSData {
```

```
// Value Types and Inheritance

class AllOnesData : NSData {
    override func getBytes(_ buffer: UnsafeMutablePointer<Void>, length: Int) {
        memset(buffer, 1, length)
    }
    ...
}
```

```
// Value Types and Inheritance

class AllOnesData : NSData {
   override func getBytes(_ buffer: UnsafeMutablePointer<Void>, length: Int) {
      memset(buffer, 1, length)
   }
   ...
}
```

```
let ones = Data(reference: AllOnesData(length: 5))
```

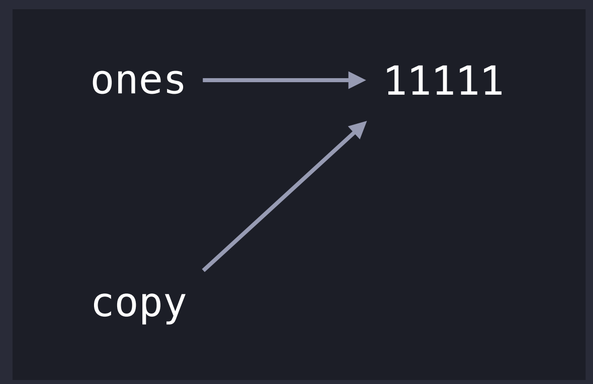
```swift
// Value Types and Inheritance

class AllOnesData : NSData {
   override func getBytes(_ buffer: UnsafeMutablePointer<Void>, length: Int) {
      memset(buffer, 1, length)
   }
   ...
}
```

```swift
let ones = Data(reference: AllOnesData(length: 5))
```

ones ⟶ 11111

```
// Value Types and Inheritance

class AllOnesData : NSData {
   override func getBytes(_ buffer: UnsafeMutablePointer<Void>, length: Int) {
      memset(buffer, 1, length)
   }
   ...
}

let ones = Data(reference: AllOnesData(length: 5))

var copy = ones
```

```
// Value Types and Inheritance

class AllOnesData : NSData {
   override func getBytes(_ buffer: UnsafeMutablePointer<Void>, length: Int) {
      memset(buffer, 1, length)
   }
   ...
}

let ones = Data(reference: AllOnesData(length: 5))

var copy = ones
copy.withUnsafeMutableBytes { (bytes : UnsafeMutablePointer<UInt8>) in
```



```
// Value Types and Inheritance

class AllOnesData : NSData {
   override func getBytes(_ buffer: UnsafeMutablePointer<Void>, length: Int) {
      memset(buffer, 1, length)
   }
   ...
}

let ones = Data(reference: AllOnesData(length: 5))

var copy = ones

copy.withUnsafeMutableBytes { (bytes : UnsafeMutablePointer<UInt8>) in
   bytes.pointee = 0
}
```

Type Safe Access

Historically, many constraints enforced at runtime

Type Safe Access

Historically, many constraints enforced at runtime

```
// Swift 2.2
let url = NSURL.fileURL(withPath: "/my-special-file")
let keys = [NSURLCreationDateKey, NSURLIsRegularFileKey, NSURLVolumeMaximumFileSizeKey]
let values = try url.resourceValues(forKeys: keys)
```

Type Safe Access

Historically, many constraints enforced at runtime

```
// Swift 2.2
let url = NSURL.fileURL(withPath: "/my-special-file")
let keys = [NSURLCreationDateKey, NSURLIsRegularFileKey, NSURLVolumeMaximumFileSizeKey]
let values = try url.resourceValues(forKeys: keys)
```

Type Safe Access

Historically, many constraints enforced at runtime

```
// Swift 2.2
let url = NSURL.fileURL(withPath: "/my-special-file")
let keys = [NSURLCreationDateKey, NSURLIsRegularFileKey, NSURLVolumeMaximumFileSizeKey]
let values = try url.resourceValues(forKeys: keys)
```

```
[String: AnyObject]
```

Type Safe Access

Historically, many constraints enforced at runtime

```
// Swift 2.2
let url = NSURL.fileURL(withPath: "/my-special-file")
let keys = [NSURLCreationDateKey, NSURLIsRegularFileKey, NSURLVolumeMaximumFileSizeKey]
let values = try url.resourceValues(forKeys: keys)

if values[NSURLIsRegularFileKey] as! Boolean { ... }
if let maxSize = (values[NSURLVolumeMaximumFileSizeKey] as? Int) { ... }
```

Type Safe Access

Historically, many constraints enforced at runtime

```
// Swift 2.2
let url = NSURL.fileURL(withPath: "/my-special-file")
let keys = [NSURLCreationDateKey, NSURLIsRegularFileKey, NSURLVolumeMaximumFileSizeKey]
let values = try url.resourceValues(forKeys: keys)

if values[NSURLIsRegularFileKey] as! Boolean { ... }
if let maxSize = (values[NSURLVolumeMaximumFileSizeKey] as? Int) { ... }

var newValues = values
newValues[NSURLIsRegularFileKey] = false
newValues[NSURLCreationDateKey] = "Two Days Ago"
try url.setResourceValues(newValues)
```

Type Safe Access

NEW

New value types express these constraints at compile-time

Type Safe Access

NEW

New value types express these constraints at compile-time

```
// Swift 3
let url = URL(fileURLWithPath: "/my-special-file")
```

Type Safe Access

NEW

New value types express these constraints at compile-time

```
// Swift 3
let url = URL(fileURLWithPath: "/my-special-file")
let keys : Set<URLResourceKey> = [.creationDateKey,
.isRegularFileKey,
.volumeMaximumFileSizeKey]
```

Type Safe Access

NEW

New value types express these constraints at compile-time

```
// Swift 3
let url = URL(fileURLWithPath: "/my-special-file")
let keys : Set<URLResourceKey> = [.creationDateKey,
.isRegularFileKey,
.volumeMaximumFileSizeKey]
```

Type Safe Access

NEW

New value types express these constraints at compile-time

```
// Swift 3
let url = URL(fileURLWithPath: "/my-special-file")
let keys : Set<URLResourceKey> = [.creationDateKey,
                                  .isRegularFileKey,
                                  .volumeMaximumFileSizeKey]
let values = try url.resourceValues(forKeys: keys)
```

Type Safe Access

NEW

New value types express these constraints at compile-time

```
// Swift 3
let url = URL(fileURLWithPath: "/my-special-file")
let keys : Set<URLResourceKey> = [.creationDateKey,
                                  .isRegularFileKey,
                                  .volumeMaximumFileSizeKey]
let values = try url.resourceValues(forKeys: keys)
```

```
struct URLResourceValues
```

Type Safe Access

NEW

```
public struct URLResourceValues {
    ...
    public var creationDate: Date? { get set }
    public var isRegularFile: Bool? { get }
    public var volumeMaximumFileSize: Int? { get }
    ...
```

Type Safe Access

NEW

```
public struct URLResourceValues {
    ...
    public var creationDate: Date? { get set }
    public var isRegularFile: Bool? { get }
    public var volumeMaximumFileSize: Int? { get }
    ...
    public var allValues: [URLResourceKey : AnyObject] { get }
}
```

Type Safe Access

NEW

```
public struct URLResourceValues {
    ...
    public var creationDate: Date? { get set }
    public var isRegularFile: Bool? { get }
    public var volumeMaximumFileSize: Int? { get }
    ...
    public var allValues: [URLResourceKey : AnyObject] { get }
}
```

Properties optional because

- It was not included in the requested keys

Type Safe Access

NEW

```
public struct URLResourceValues {
    ...
    public var creationDate: Date? { get set }
    public var isRegularFile: Bool? { get }
    public var volumeMaximumFileSize: Int? { get }
    ...
    public var allValues: [URLResourceKey : AnyObject] { get }
}
```

Properties optional because

- It was not included in the requested keys
- The data was not present for the resource

Type Safe Access

Properties are strongly typed

```
if values.isRegularFile! {
    ...
}
```

Type Safe Access

Properties are strongly typed

```
if values.isRegularFile! {
   ...
}

if let maxFileSize = values.volumeMaximumFileSize {
  ...
}
```

Type Safe Access

Strongly typed properties help prevent invalid mutation

Type Safe Access

Strongly typed properties help prevent invalid mutation

```
var mutableValues = values
mutableValues.isRegularFile = false
```

Type Safe Access

Strongly typed properties help prevent invalid mutation

```
var mutableValues = values
mutableValues.isRegularFile = false
```

Cannot assign to property: 'isRegularFile' is a get-only property.

Type Safe Access

Strongly typed properties help prevent invalid mutation

```
var mutableValues = values
mutableValues.isRegularFile = false
```

Cannot assign to property: 'isRegularFile' is a get-only property.

```
var mutableValues = values
mutableValues.creationDate = "Two Days Ago"
```

Type Safe Access

Strongly typed properties help prevent invalid mutation

```
var mutableValues = values
mutableValues.isRegularFile = false
```

Cannot assign to property: 'isRegularFile' is a get-only property.

```
var mutableValues = values
mutableValues.creationDate = "Two Days Ago"
```

Cannot assign value of type 'String' to type 'Date'

Native Enumerations

NEW

Native Enumerations

NEW

Data

Native Enumerations

NEW

Data

```
public enum Deallocator {
    case virtualMemory
    case unmap
    case free
    case none
    case custom((UnsafeMutablePointer<UInt8>, Int) -> Void)
}
```

Native Enumerations

NEW

Data

```
public enum Deallocator {
    case virtualMemory
    case unmap
    case free
    case none
    case custom((UnsafeMutablePointer<UInt8>, Int) -> Void)
}
```

Native Enumerations

NEW

Associated values enable expressive, uniform APIs

Native Enumerations

NEW

Associated values enable expressive, uniform APIs

```
let byteCount = 32
var pointer = UnsafeMutablePointer<UInt8>(malloc(byteCount))
let data = Data(bytesNoCopy: pointer, count: count,  deallocator: .free)
```

Native Enumerations

NEW

Associated values enable expressive, uniform APIs

```
let byteCount = 32
var pointer = UnsafeMutablePointer<UInt8>(malloc(byteCount))
let data = Data(bytesNoCopy: pointer, count: count,  deallocator: .free)
```

Native Enumerations

NEW

Associated values enable expressive, uniform APIs

```
let byteCount = 32
var pointer = UnsafeMutablePointer<UInt8>(malloc(byteCount))
let data = Data(bytesNoCopy: pointer, count: count, deallocator: .free)
```

Native Enumerations

NEW

Associated values enable expressive, uniform APIs

```
let byteCount = 32
var pointer = UnsafeMutablePointer<UInt8>(malloc(byteCount))
let data = Data(bytesNoCopy: pointer, count: count,  deallocator: .free)
```

```
var count: Int
var pointer: UnsafeBufferPointer<UInt8> = create_glorious_pointer(&count)
let data = Data(bytesNoCopy: pointer, count: count, deallocator: .custom {
   print("cleaning up allocation at \($0) of \($1) glorious bytes")
   ...
})
```

Native Enumerations

NEW

Associated values enable expressive, uniform APIs

```
let byteCount = 32
var pointer = UnsafeMutablePointer<UInt8>(malloc(byteCount))
let data = Data(bytesNoCopy: pointer, count: count,  deallocator: .free)
```

```
var count: Int
var pointer: UnsafeBufferPointer<UInt8> = create_glorious_pointer(&count)
let data = Data(bytesNoCopy: pointer, count: count, deallocator: .custom {
   print("cleaning up allocation at \($0) of \($1) glorious bytes")
   ...
})
```

Native Enumerations

NEW

Associated values enable expressive, uniform APIs

```
let byteCount = 32
var pointer = UnsafeMutablePointer<UInt8>(malloc(byteCount))
let data = Data(bytesNoCopy: pointer, count: count,  deallocator: .free)
```

```
var count: Int
var pointer: UnsafeBufferPointer<UInt8> = create_glorious_pointer(&count)
let data = Data(bytesNoCopy: pointer, count: count, deallocator: .custom {
   print("cleaning up allocation at \($0) of \($1) glorious bytes")
   ...
})
```

API Exploration

Nested enumerations

Strongly typed string enumerations

Class properties

Value types

Protocol conformance

Type safe access

Value types and inheritance

Native enumerations

Adoption

Tony Parker Foundation, Apple

Bridging

Extends current Objective-C bridging

Imported API uses value types

Bridging

Extends current Objective-C bridging

Imported API uses value types

```
// Swift 2.2
public class NSDatePicker : NSControl {
    @NSCopying public var minDate: NSDate?
    @NSCopying public var maxDate: NSDate?
}
```

Bridging

Extends current Objective-C bridging

Imported API uses value types

```
// Swift 2.2
public class NSDatePicker : NSControl {
    @NSCopying public var minDate: NSDate?
    @NSCopying public var maxDate: NSDate?
}
// Swift 3
public class NSDatePicker : NSControl {
    public var minDate: Date?
    public var maxDate: Date?
}
```

Bridging

Two strategies

Large types hold a reference

Small types create a reference

Bridging Large Types

Bridging Large Types

```
let data = Data(contentsOf: myFile)
```

struct Data

class NSData

Bridging Large Types

```
let data = Data(contentsOf: myFile)
```

struct Data | class NSData

```
myObject.function(data)
```

struct Data | class NSData

Bridging Large Types

```
let data = Data(contentsOf: myFile)
```

struct Data | class NSData

```
myObject.function(data)
```

struct Data | class NSData → class NSData

Bridging Large Types

```
let data = Data(contentsOf: myFile)
```

struct Data | class NSData

```
myObject.property = data
```

struct Data | class NSData

```
ivar = [dataReference copy];
```

class NSData

Bridging Large Types

Bridging Large Types

```
let data = myObject.function()
```

```
[NSData dataWithContentsOfURL:myFile];
```

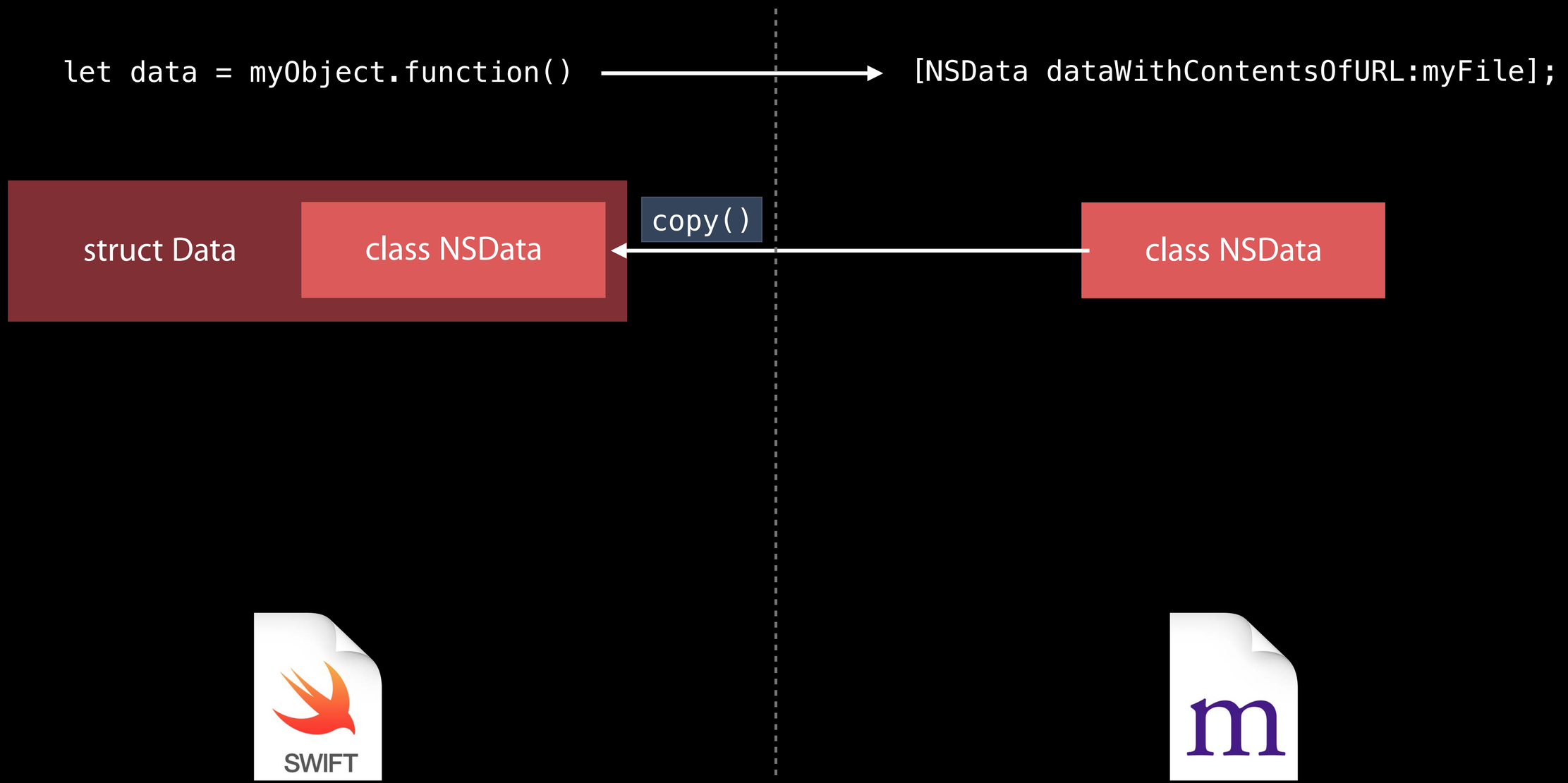
Bridging Large Types

```
let data = myObject.function()
```

```
[NSData dataWithContentsOfURL:myFile];
```

class NSData

Bridging Large Types



Bridging Small Types

Bridging Small Types

```
let now = Date()
```

struct Date

Bridging Small Types

```
let now = Date()
```

struct Date

```
myObject.function(now)
```

struct Date

Bridging Small Types

```
let now = Date()
```

struct Date

```
myObject.function(now)
```

struct Date

```
NSDate(timeInterval)
```

class NSDate

Bridging

Optimized for use in Swift

Safe by default

Potential copy when crossing bridge

Migration

New types exist for all Swift deployment targets

Migration

New types exist for all Swift deployment targets

Migrator helps move code to new API

```swift
// Swift 3 Migration


// Swift 2.2
let date = NSDate()
let laterDate = date.dateByAddingTimeInterval(60)
```

```swift
// Swift 3 Migration

// Swift 2.2
let date = NSDate()
let laterDate = date.dateByAddingTimeInterval(60)

// Swift 3, migration result
let date = Date()
let laterDate = date.addingTimeInterval(60)
```

```swift
// Swift 3 Migration

// Swift 2.2
let date = NSDate()
let laterDate = date.dateByAddingTimeInterval(60)

// Swift 3, migration result
let date = Date()
let laterDate = date.addingTimeInterval(60)
```

```swift
// Swift 3 Migration

// Swift 2.2
let date = NSDate()
let laterDate = date.dateByAddingTimeInterval(60)

// Swift 3, migration result
let date = Date()
let laterDate = date.addingTimeInterval(60)
```

```
// Swift 3 Migration


// Swift 2.2
let d = NSDateComponents()
d.year = 1999
d.month = 12
d.day = 31
```

```
// Swift 3 Migration

// Swift 2.2
let d = NSDateComponents()
d.year = 1999
d.month = 12
d.day = 31

// Swift 3, migration result
var d = DateComponents()
d.year = 1999
d.month = 12
d.day = 31
```

```
// Swift 3 Migration


// Swift 2.2
let d = NSDateComponents()
d.year = 1999
d.month = 12
d.day = 31

// Swift 3, migration result
var d = DateComponents()
d.year = 1999
d.month = 12
d.day = 31
```

```
// Swift 3 Migration


// Swift 2.2
let d = NSDateComponents()
d.year = 1999
d.month = 12
d.day = 31

// Swift 3, migration result
var d = DateComponents()
d.year = 1999
d.month = 12
d.day = 31
```

```
// Swift 3 Migration

// Swift 2.2
let d = NSDateComponents()
d.year = 1999
d.month = 12
d.day = 31

// Swift 3, migration result
var d = DateComponents()
d.year = 1999
d.month = 12
d.day = 31

// Swift 3, manual changes
let d = DateComponents(year: 1999, month: 12, day: 31)
```

```
// Swift 3 Migration

// Swift 2.2
let d = NSDateComponents()
d.year = 1999
d.month = 12
d.day = 31

// Swift 3, migration result
var d = DateComponents()
d.year = 1999
d.month = 12
d.day = 31

// Swift 3, manual changes
let d = DateComponents(year: 1999, month: 12, day: 31)
```

```
// Swift 3 Migration


// Swift 2.2
let data = try NSMutableData(contentsOfURL: url1, options: [])
data.appendData(try NSData(contentsOfURL: url2, options: []))
```

```swift
// Swift 3 Migration

// Swift 2.2
let data = try NSMutableData(contentsOfURL: url1, options: [])
data.appendData(try NSData(contentsOfURL: url2, options: []))

// Swift 3, migration result
let data = try NSMutableData(contentsOf: url1, options: [])
data.append(try Data(contentsOf: url2, options: []))
```

```
// Swift 3 Migration

// Swift 2.2
let data = try NSMutableData(contentsOfURL: url1, options: [])
data.appendData(try NSData(contentsOfURL: url2, options: []))

// Swift 3, migration result
let data = try NSMutableData(contentsOf: url1, options: [])
data.append(try Data(contentsOf: url2, options: []))
```

```swift
// Swift 3 Migration

// Swift 2.2
let data = try NSMutableData(contentsOfURL: url1, options: [])
data.appendData(try NSData(contentsOfURL: url2, options: []))

// Swift 3, migration result
let data = try NSMutableData(contentsOf: url1, options: [])
data.append(try Data(contentsOf: url2, options: []))
```

```swift
// Swift 3 Migration

// Swift 2.2
let data = try NSMutableData(contentsOfURL: url1, options: [])
data.appendData(try NSData(contentsOfURL: url2, options: []))

// Swift 3, migration result
let data = try NSMutableData(contentsOf: url1, options: [])
data.append(try Data(contentsOf: url2, options: []))
```

```swift
// Swift 3 Migration

// Swift 2.2
let data = try NSMutableData(contentsOfURL: url1, options: [])
data.appendData(try NSData(contentsOfURL: url2, options: []))

// Swift 3, migration result
let data = try NSMutableData(contentsOf: url1, options: [])
data.append(try Data(contentsOf: url2, options: []))

// Swift 3, manual changes
var data = try Data(contentsOf: url1)
data.append(try Data(contentsOf: url2))
```

```
// Swift 3 Migration


// Swift 2.2
let data = try NSMutableData(contentsOfURL: url1, options: [])
data.appendData(try NSData(contentsOfURL: url2, options: []))

// Swift 3, migration result
let data = try NSMutableData(contentsOf: url1, options: [])
data.append(try Data(contentsOf: url2, options: []))

// Swift 3, manual changes
var data = try Data(contentsOf: url1)
data.append(try Data(contentsOf: url2))
```

```swift
// Swift 3 Migration

// Swift 2.2
let data = try NSMutableData(contentsOfURL: url1, options: [])
data.appendData(try NSData(contentsOfURL: url2, options: []))

// Swift 3, migration result
let data = try NSMutableData(contentsOf: url1, options: [])
data.append(try Data(contentsOf: url2, options: []))

// Swift 3, manual changes
var data = try Data(contentsOf: url1)
data.append(try Data(contentsOf: url2))
```

```
// Swift 3 Migration

// Swift 2.2
let data = try NSMutableData(contentsOfURL: url1, options: [])
data.appendData(try NSData(contentsOfURL: url2, options: []))

// Swift 3, migration result
let data = try NSMutableData(contentsOf: url1, options: [])
data.append(try Data(contentsOf: url2, options: []))

// Swift 3, manual changes
var data = try Data(contentsOf: url1)
data.append(try Data(contentsOf: url2))
```

Summary

Summary

Improvements to Foundation benefit the whole SDK

- API renaming
- Value types
- Swift-specific API

Summary

Improvements to Foundation benefit the whole SDK

- API renaming
- Value types
- Swift-specific API

Continue to be leverage point in the future

More Information

https://developer.apple.com/wwdc16/207

Related Sessions

Swift API Design Guidelines	Presidio	Tuesday 10:00AM
What's New in Cocoa	Nob Hill	Tuesday 11:00AM
Going Server-Side with Swift Open Source	Mission	Friday 9:00AM
Measurements and Units	Presidio	Friday 4:00PM

Labs

Swift and Foundation Lab	Developer Tools Lab A	Wednesday 9:00AM
Cocoa Lab	Frameworks Lab D	Thursday 2:00PM
Cocoa Lab	Frameworks Lab A	Friday 1:00PM

WWDC16