#WWDC19

Advances in Foundation

I-Ting Tina Liu, Foundation

New API Highlights

Combine

Ordered Collection Diffing

Data

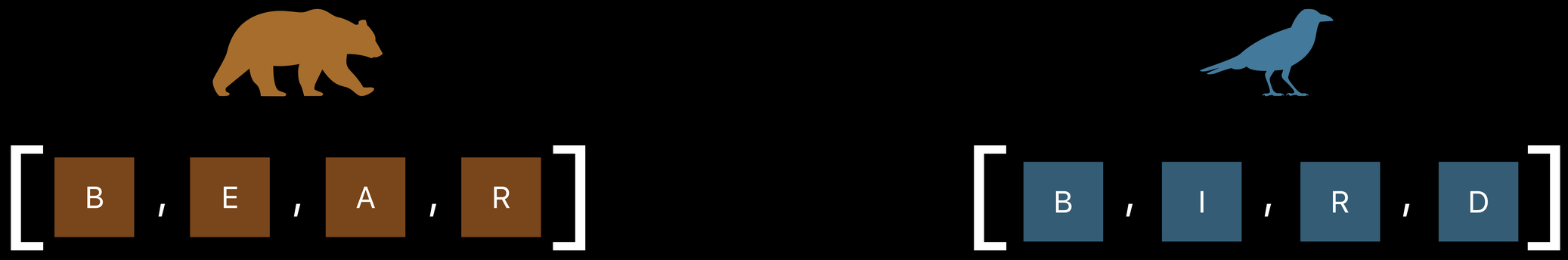
Units and Formatters

OperationQueue

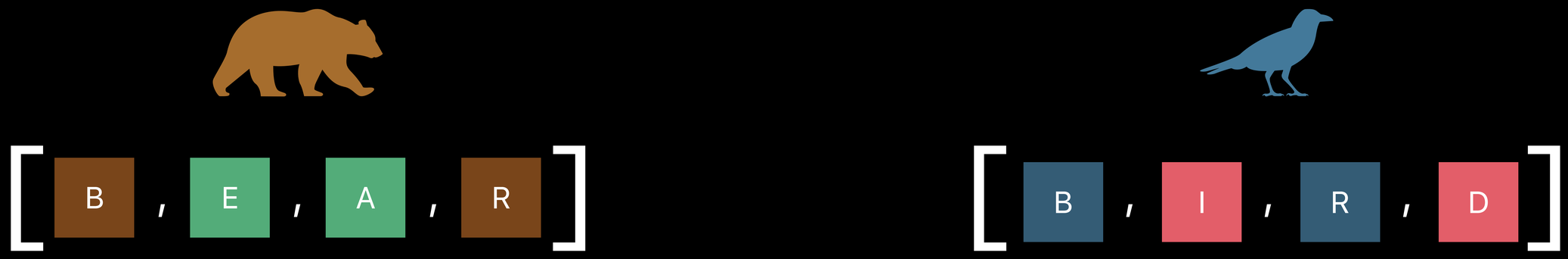
USB and SMB on iOS

Swift Update

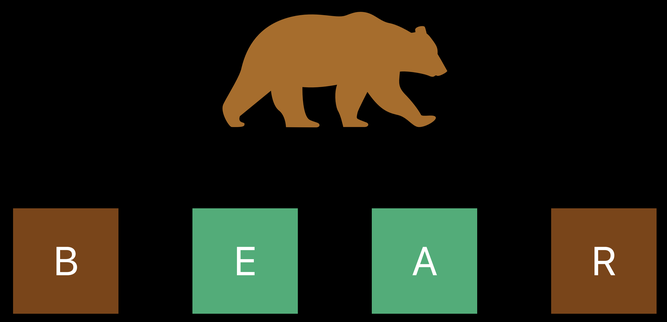
Ordered Collection Diffing



Ordered Collection Diffing

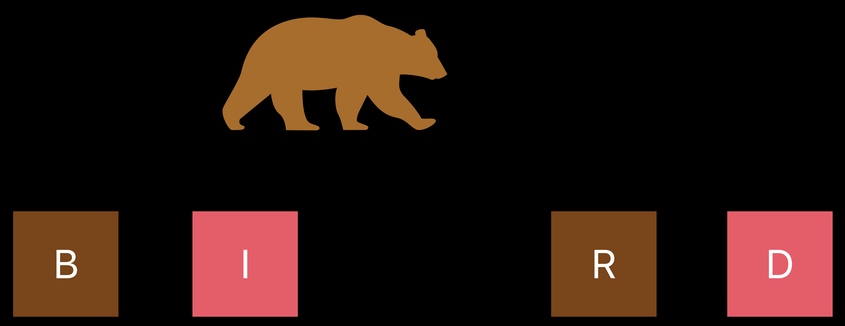


Ordered Collection Diffing

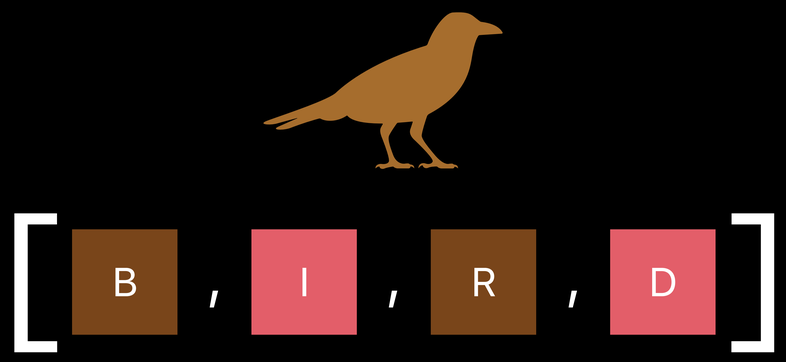


Ordered Collection Diffing

Ordered Collection Diffing



Ordered Collection Diffing



Ordered Collection Diffing

```
let diff = bird.difference(from: bear)

let newBird = bear.applying(diff)                    // [b, i, r, d]
```

Ordered Collection Diffing

```
let diff = bird.difference(from: bear)

let newBird = bear.applying(diff)                    // [b, i, r, d]
```

Ordered Collection Diffing

```
let diff = bird.difference(from: bear)

let newBird = bear.applying(diff)                    // [b, i, r, d]
```

Ordered Collection Diffing

```
let diff = bird.difference(from: bear)

let newBird = bear.applying(diff)                    // [b, i, r, d]
```

Ordered Collection Diffing

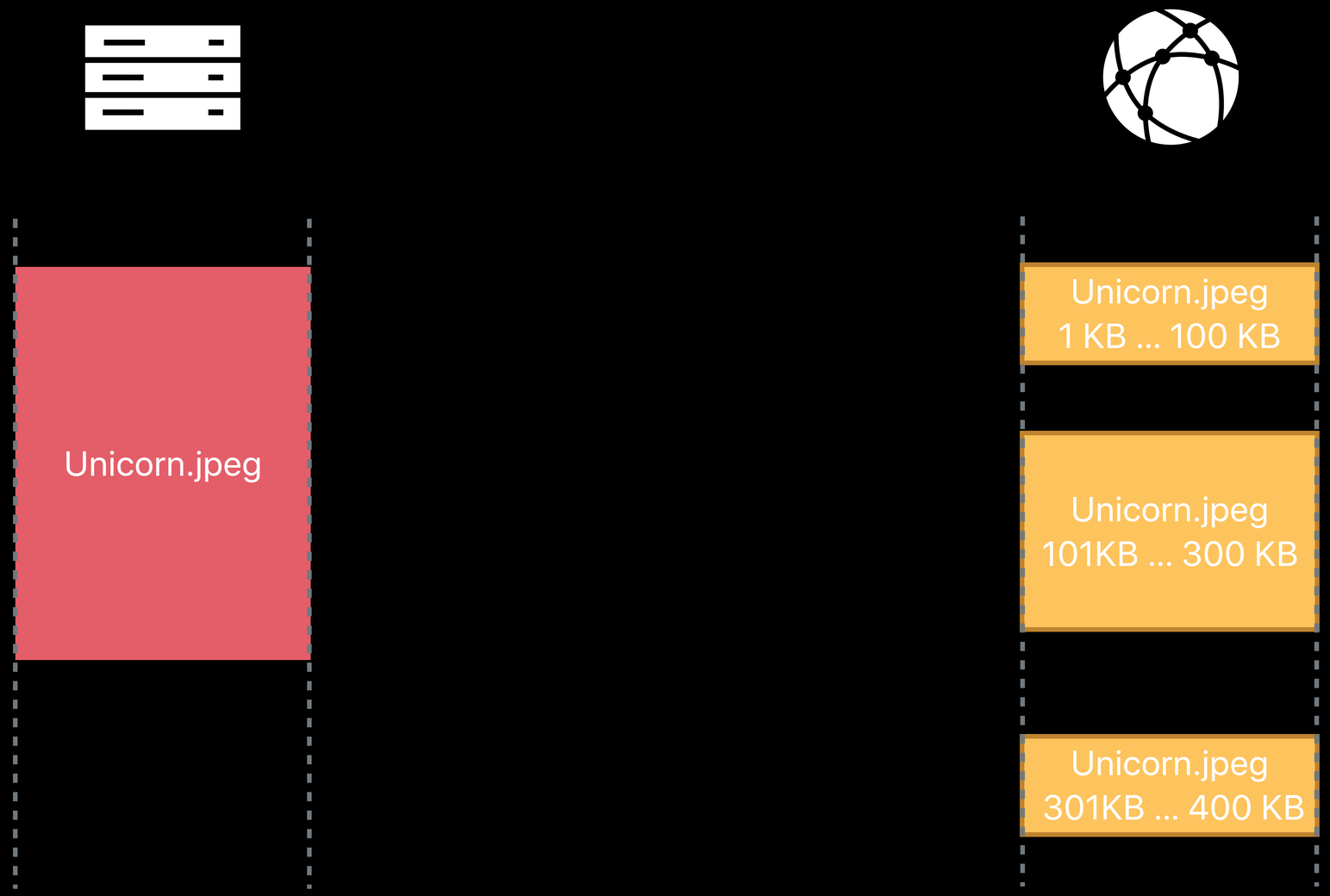
```
let diff = bird.difference(from: bear)

let newBird = bear.applying(diff)                    // [b, i, r, d]
```

Data

Contiguity

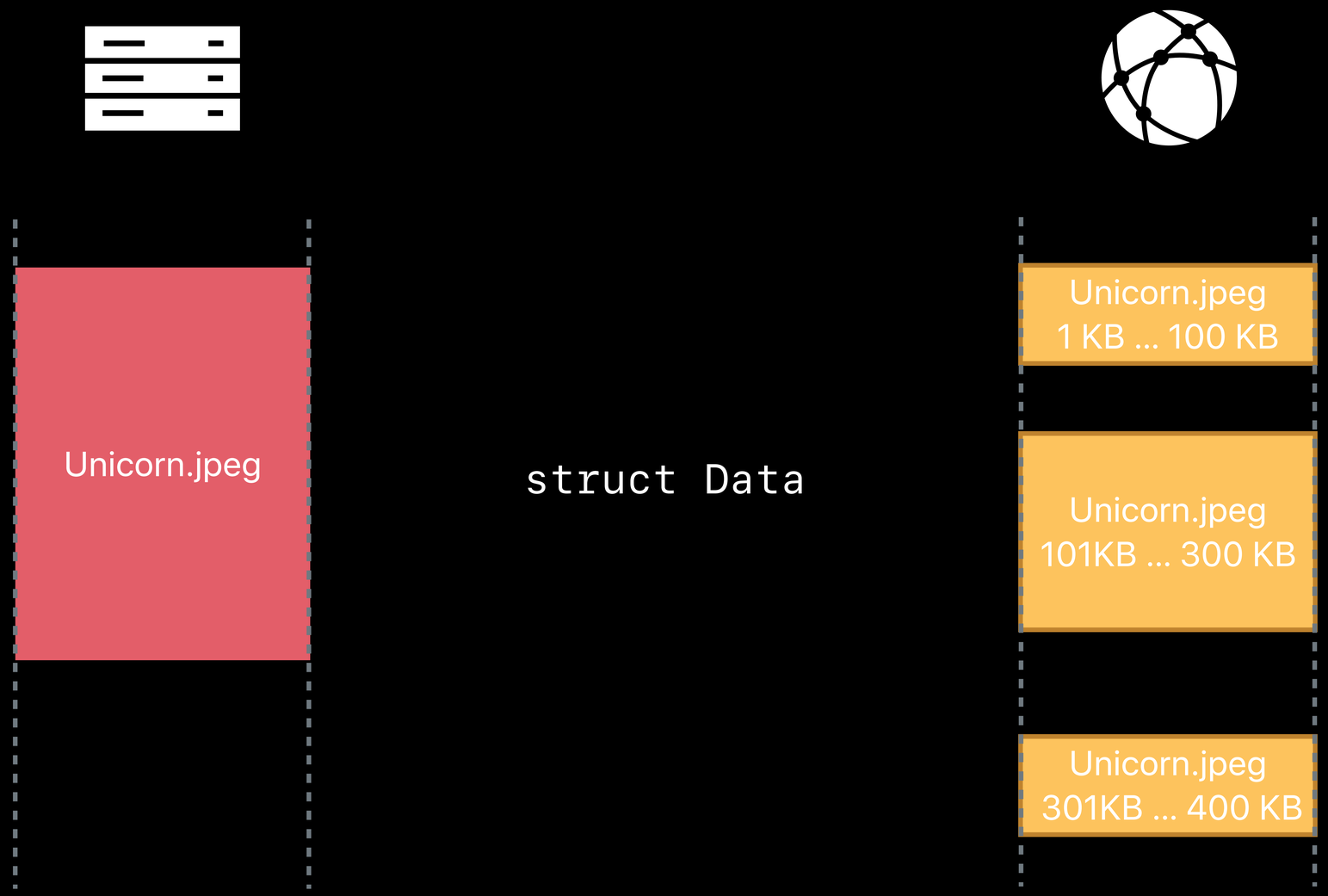
Data

Contiguity



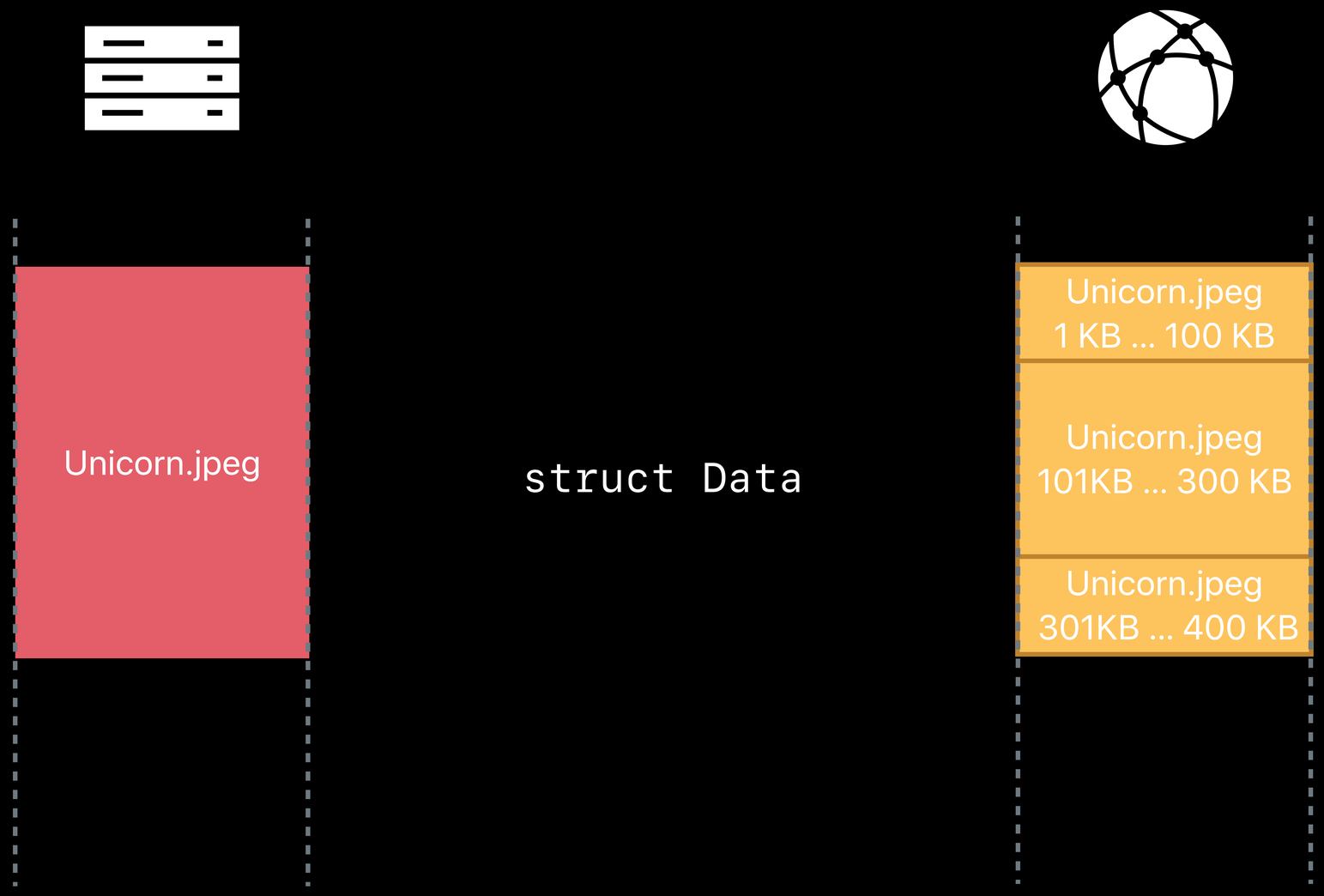
Data

Contiguity



Data

Contiguity



Data

NEW

`Data` is contiguous

Data

NEW

`Data` is contiguous

```
public protocol ContiguousBytes {
    func withUnsafeBytes<R>(_ body: (UnsafeRawBufferPointer) throws -> R) rethrows -> R
}
```

Data

NEW

Work with potentially discontiguous types

```
public protocol DataProtocol: RandomAccessCollection where Element == UInt8, … { }

public protocol MutableDataProtocol : DataProtocol,
    MutableCollection, RangeReplaceableCollection { }
```

Data

NEW

Work with potentially discontiguous types

```
public protocol DataProtocol: RandomAccessCollection where Element == UInt8, … { }

public protocol MutableDataProtocol : DataProtocol,
    MutableCollection, RangeReplaceableCollection { }
```

Data

NEW

Work with potentially discontiguous types

```
public protocol DataProtocol: RandomAccessCollection where Element == UInt8, … { }

public protocol MutableDataProtocol : DataProtocol,
    MutableCollection, RangeReplaceableCollection { }
```

Data

NEW

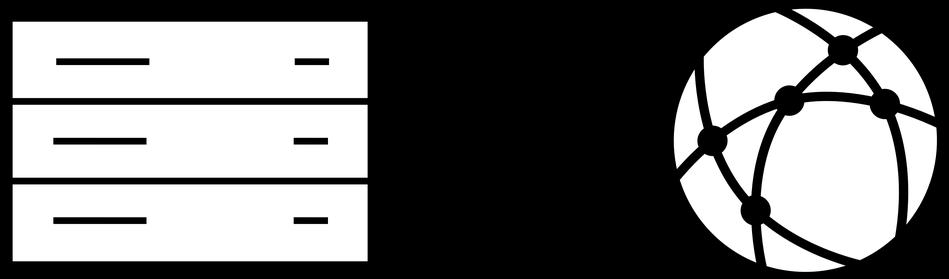
Protocols adopted by

- Foundation: `Data`
- Swift Standard Library: `[UInt8]`
- Dispatch: `DispatchData`

Consider using `DataProtocol` as a generic constraint

Data

Compression



Data

Compression

```
let compressed = try data.compressed(using: .lzfse)

public enum CompressionAlgorithm : Int {
    case lzfse
    case lz4
    case lzma
    case zlib
}
```

Data

Compression

```
let compressed = try data.compressed(using: .lzfse)

public enum CompressionAlgorithm : Int {
    case lzfse
    case lz4
    case lzma
    case zlib
}
```

Data

Compression

```
let compressed = try data.compressed(using: .lzfse)
```

```
public enum CompressionAlgorithm : Int {
    case lzfse
    case lz4
    case lzma
    case zlib
}
```

Units

`UnitDuration`

- Added `milliseconds, microseconds, nanoseconds,` and `picoseconds`

`UnitFrequency`

- Added `framesPerSecond`

Units

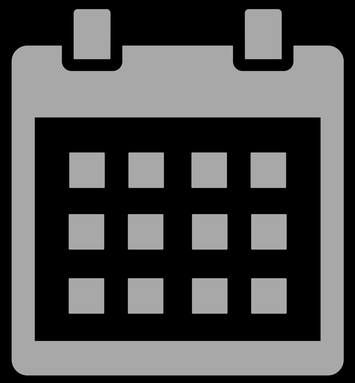
NEW

`UnitInformationStorage`

- `bits, bytes, nibbles` for common usage
- SI- and binary-prefixed units `(kilo, kibi, ... yotta, yobi)`
- Format with `MeasurementFormatter` and `ByteCountFormatter`

Displaying a Date or Time

" Read: 1 hour ago "



" Payment due: Tomorrow "

Relative Date Time Formatter

```
let formatter = RelativeDateTimeFormatter()
let dateString = formatter.localizedString(for: aDate, relativeTo: now)

// en_US:      "2 weeks ago"
// es_ES:      "hace 2 semanas"
// zh_TW:      "2 週前"
```

List Formatter

```
let string = ListFormatter.localizedString(byJoining: ["🐶", "🐷", "🦄"])

// en_US:    "🐶, 🐷, and 🦄"
// es_ES:    "🐶, 🐷 y 🦄"
// zh_TW:    "🐶、🐷和🦄"
```

List Formatter

8/15/2019 9/13/2019 2/1/2020

“8/15/19, 9/13/19, and 2/1/20”

“Aug 15, 2019, Sep 13, 2019, and Feb 1, 2020”

List Formatter

[8/15/2019 , 9/13/2019 , 2/1/2020]

“8/15/19, 9/13/19, and 2/1/20”

“Aug 15, 2019, Sep 13, 2019, and Feb 1, 2020”

“15/8/19, 13/9/19 y 1/2/20”

“15 ago 2019, 13 sept 2019 y 1 feb 2020”

List Formatter

```
let listFormatter = ListFormatter()
let dateFormatter = DateFormatter()


listFormatter.itemFormatter = dateFormatter
let string = listFormatter.string(from: dates)

// en_US:    "8/15/19, 9/13/19, and 2/1/20"
// es_ES:    "15/8/19, 13/9/19 y 1/2/20”
```

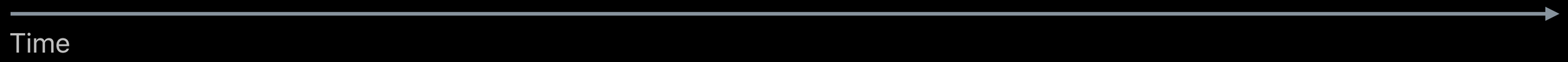
List Formatter

```
let listFormatter = ListFormatter()
let dateFormatter = DateFormatter()


listFormatter.itemFormatter = dateFormatter
let string = listFormatter.string(from: dates)

// en_US:    "8/15/19, 9/13/19, and 2/1/20"
// es_ES:    "15/8/19, 13/9/19 y 1/2/20"
```

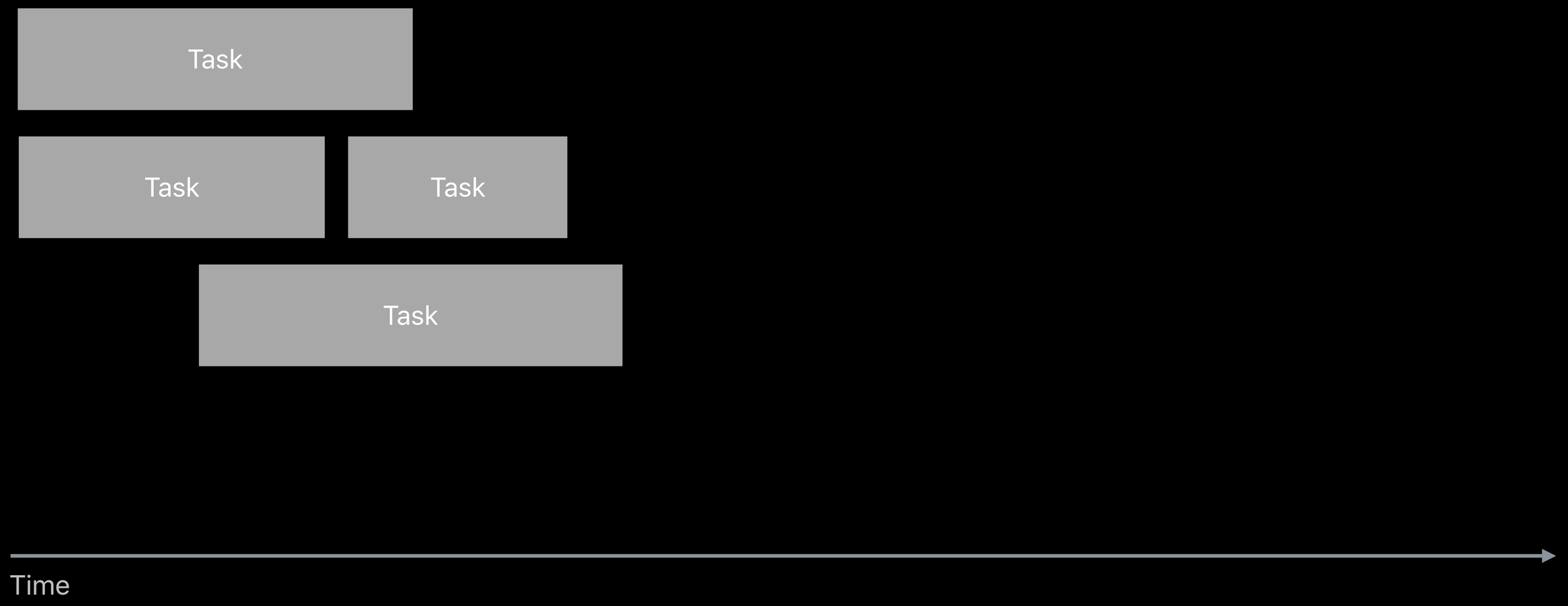
List Formatter

```
let listFormatter = ListFormatter()
let dateFormatter = DateFormatter()
dateFormatter.dateStyle = .medium

listFormatter.itemFormatter = dateFormatter
let string = listFormatter.string(from: dates)

// en_US:    "Aug 15, 2019, Sep 13, 2019, and Feb 1, 2020"
// es_ES:    "15 ago 2019, 13 sept 2019 y 1 feb 2020”
```

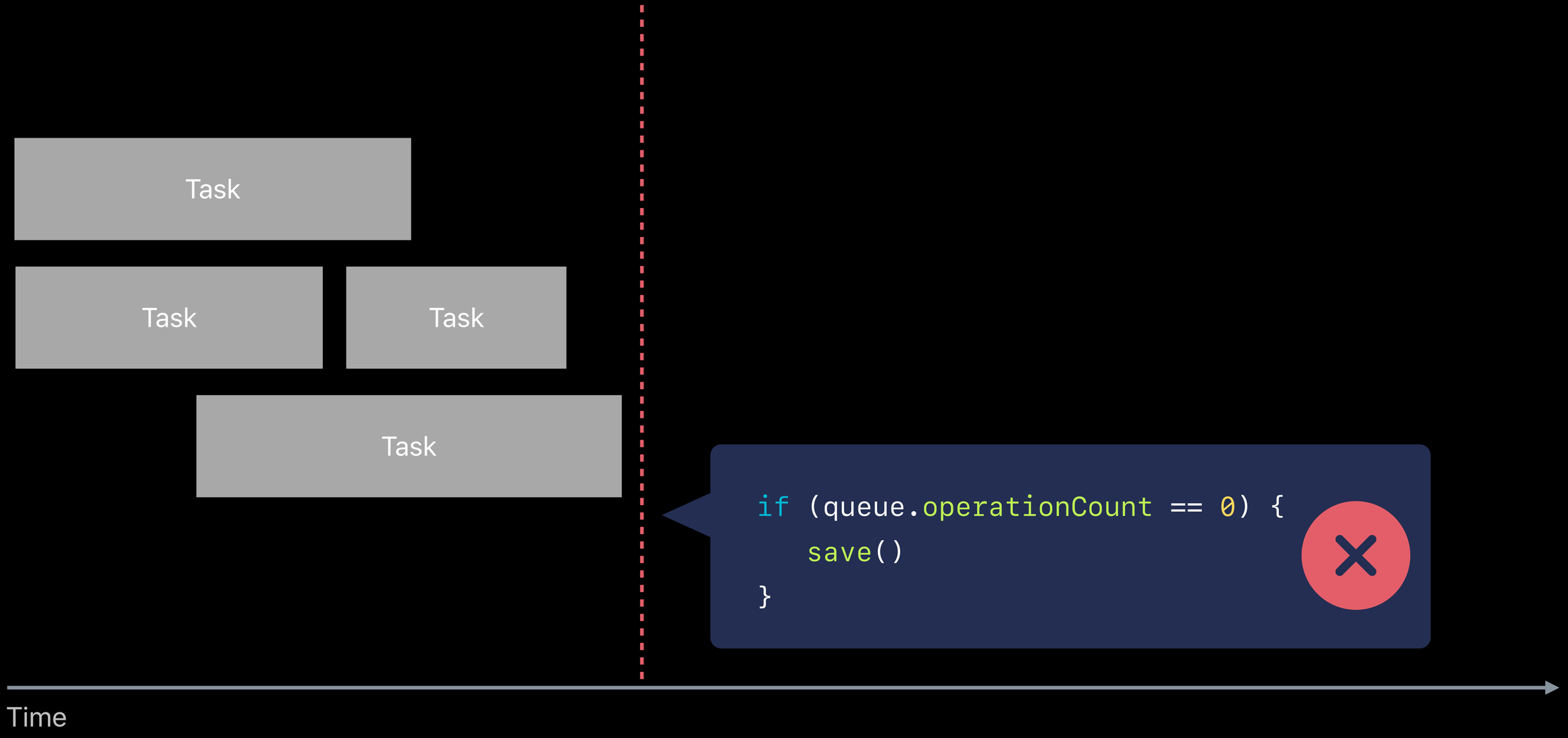
Operation Queue



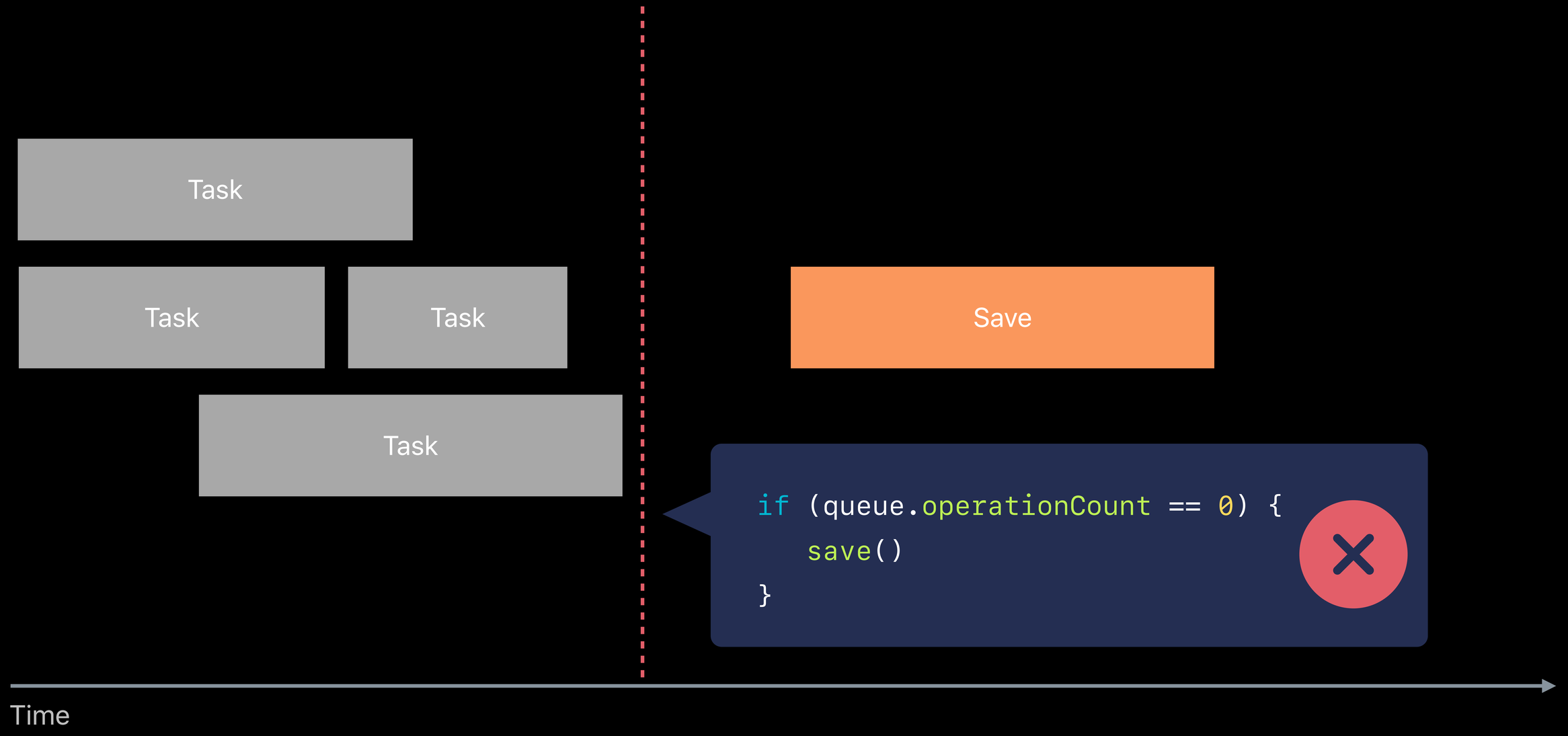
Operation Queue



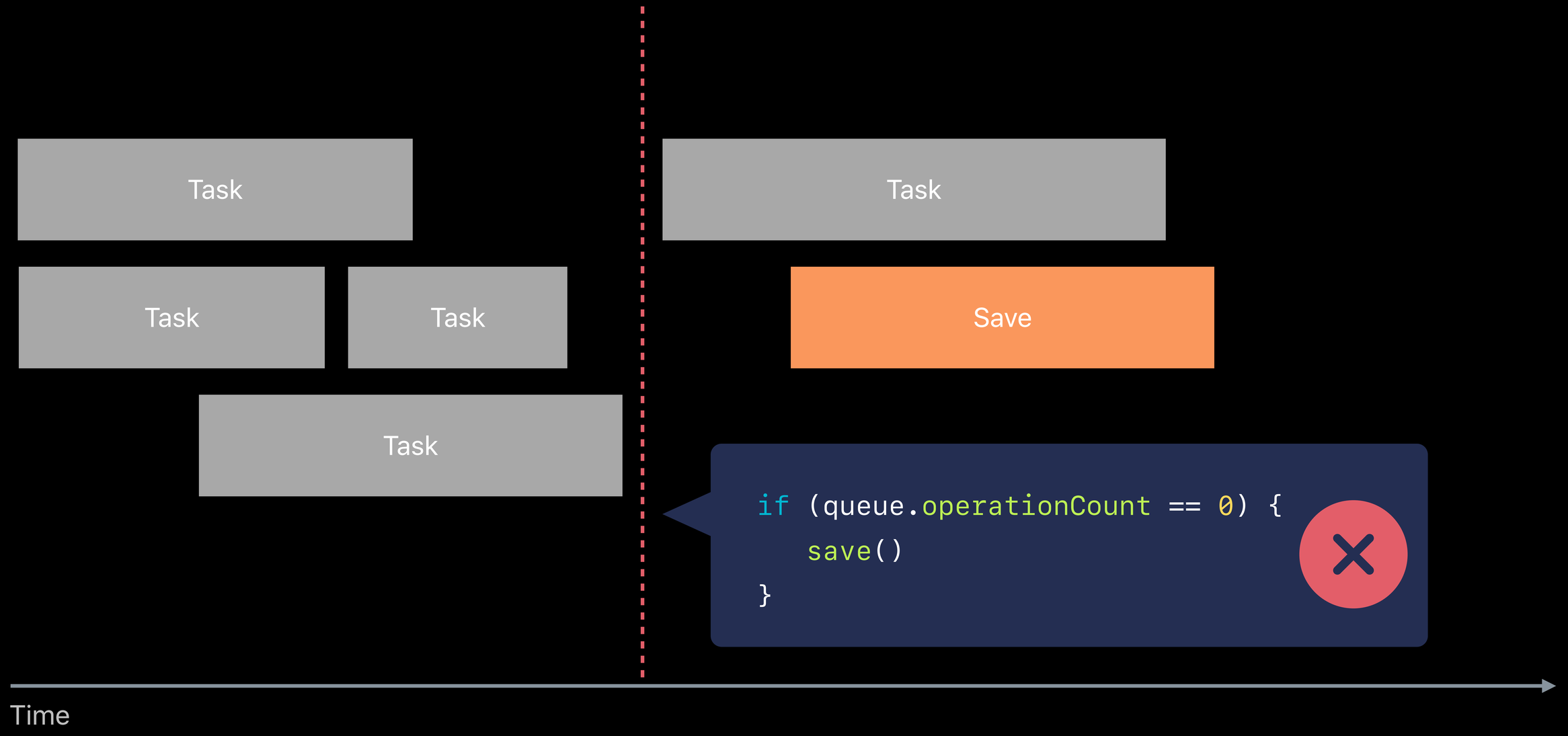
Operation Queue



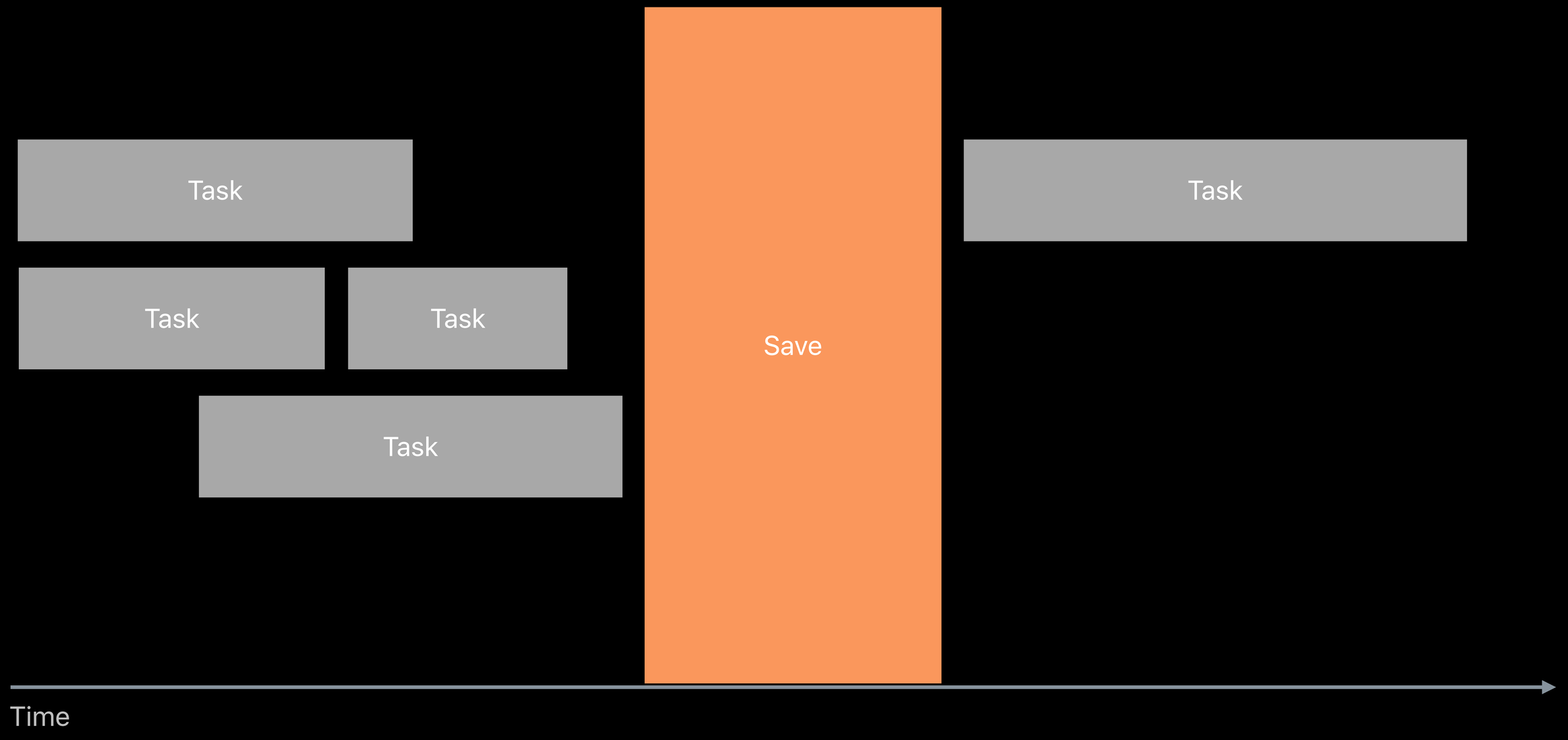
Operation Queue



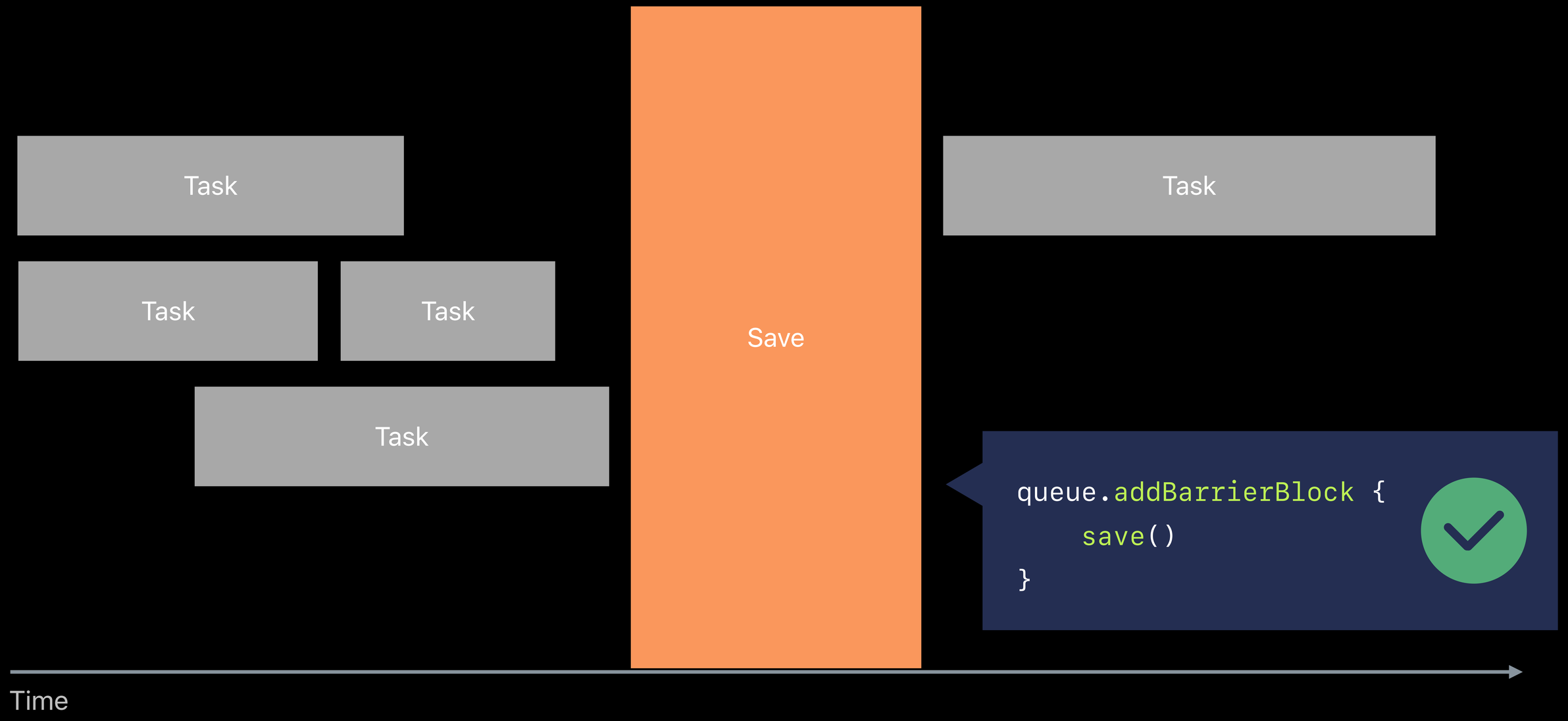
Operation Queue



Operation Queue



Operation Queue



Operation Queue

Progress reporting

```
let queue = OperationQueue()
queue.progress.totalUnitCount = 3

queue.addOperation {
    task1()                                    // Finished task: 1 / 3
}
queue.addOperation {
    task2()                                    // Finished task: 2 / 3
}
queue.addOperation {
    task3()                                    // Finished task: 3 / 3
}
```

Operation Queue

Progress reporting

```
let queue = OperationQueue()
queue.progress.totalUnitCount = 3

queue.addOperation {
    task1()                                    // Finished task: 1 / 3
}
queue.addOperation {
    task2()                                    // Finished task: 2 / 3
}
queue.addOperation {
    task3()                                    // Finished task: 3 / 3
}
```

Operation Queue

Progress reporting

```
let queue = OperationQueue()
queue.progress.totalUnitCount = 3

queue.addOperation {
    task1()                                    // Finished task: 1 / 3
}
queue.addOperation {
    task2()                                    // Finished task: 2 / 3
}
queue.addOperation {
    task3()                                    // Finished task: 3 / 3
}
```

Be Prepared for USB and SMB on iOS

Multiple volumes

- Use `FileManager.SearchPathDirectory.itemReplacementDirectory`

Disappearing volumes

- Use `Data.ReadingOptions.mappedIfSafe`

Be Prepared for USB and SMB on iOS

Multiple volumes

- Use `FileManager.SearchPathDirectory.itemReplacementDirectory`

Disappearing volumes

- Use `Data.ReadingOptions.mappedIfSafe`

Be Prepared for USB and SMB on iOS

Multiple volumes

- Use `FileManager.SearchPathDirectory.itemReplacementDirectory`

Disappearing volumes

- Use `Data.ReadingOptions.mappedIfSafe`

Be Prepared for USB and SMB on iOS

Slower file system operations

- Defer access to non-main thread

Varying capabilities

- Test capabilities with `URLResourceKey`, e.g. `volumeSupportsFileCloningKey`
- Handle errors

Swift Update

Scanner

```
// Swift 4
var nameNSString: NSString?
if scanner.scanUpToCharacters(from: .newlines, into: &nameNSString) {
   let name = nameNSString! as String
}
```

```
// Swift 5.1
let nameString = scanner.scanUpToCharacters(from: .newlines)
```

Swift Update

Scanner

```swift
// Swift 4
var nameNSString: NSString?
if scanner.scanUpToCharacters(from: .newlines, into: &nameNSString) {
    let name = nameNSString! as String
}
```

```swift
// Swift 5.1
let nameString = scanner.scanUpToCharacters(from: .newlines)
```

Swift Update

Scanner

```
// Swift 4
var nameNSString: NSString?
if scanner.scanUpToCharacters(from: .newlines, into: &nameNSString) {
    let name = nameNSString! as String
}
```

```
// Swift 5.1
let nameString = scanner.scanUpToCharacters(from: .newlines)
```

Swift Update

Scanner

```
// Swift 4
var nameNSString: NSString?
if scanner.scanUpToCharacters(from: .newlines, into: &nameNSString) {
   let name = nameNSString! as String
}
```

```
// Swift 5.1
let nameString = scanner.scanUpToCharacters(from: .newlines)
```

Swift Update

Scanner

```
// Swift 4
var nameNSString: NSString?
if scanner.scanUpToCharacters(from: .newlines, into: &nameNSString) {
   let name = nameNSString! as String
}
```

```
// Swift 5.1
let nameString = scanner.scanUpToCharacters(from: .newlines)
let matchedString = scanner.scanString(string: "hi, 🏠")
```

Swift Update

FileHandle

Error-based API

```
let fileHandle = FileHandle()
let data = try fileHandle.readToEnd()
```

Works with `DataProtocol`

```
extension FileHandle {
    public func write<T: DataProtocol>(contentsOf data: T) throws
}
```

Try It

Use `DataProtocol` instead of `[UInt8]`

Format dates and lists with `Formatter`

Use `OperationQueue`'s barrier and progress reporting

More Information

developer.apple.com/wwdc19/723

WWDC19