A Deep Dive Into `Sendable`

Tim Condon
@0xTim

BROKENHANDS

Introduction

- Founder of Broken Hands 
- Vapor Core Team 
- SSWG and SWWG Member 
- Server-side Swift Team @ Kodeco 
- @0xTim     
- Organise ServerSide.swift, NSManchester, Vapor London

Swift


Fast
Modern
Safe
Interactive

Swift Safety

Optionals

Memory Overflows


C BUFFER
YOUR MEMORY

Swift Safety

Optionals

Memory Overflows

Explicit Error Handling

Type Checking

Swift Safety

- Stop undefined behaviour
- (Not all crashes are bad)
- Move run time issues to compile time

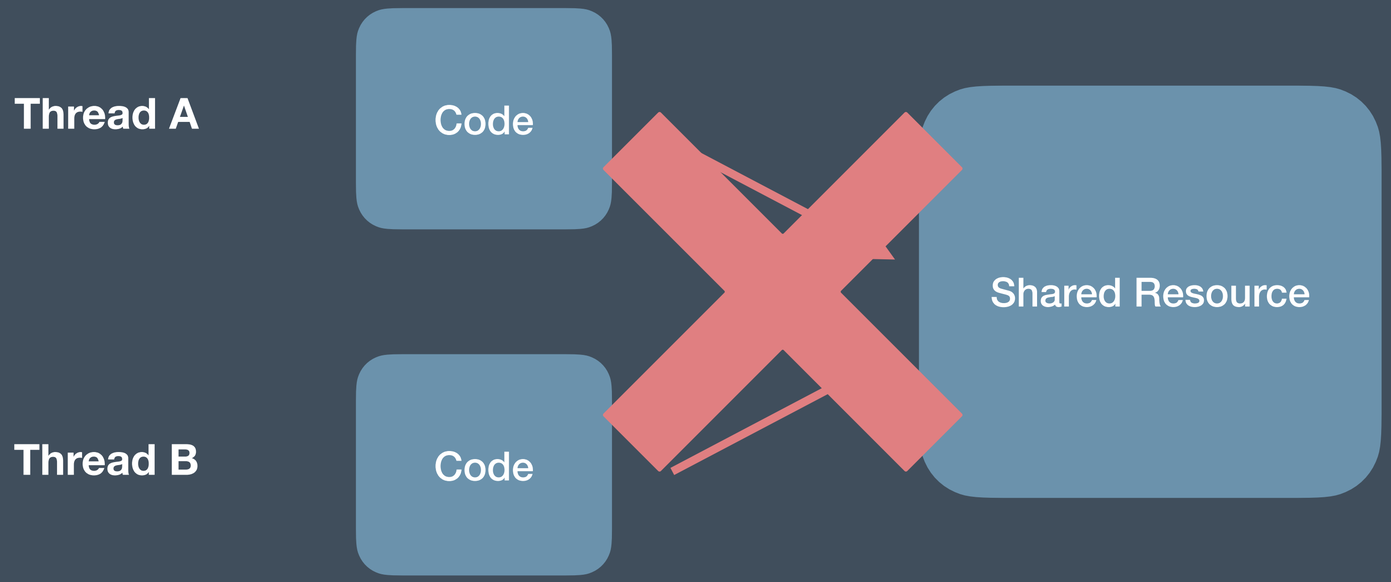

SHIP IT

Data Races

```
d 1 Queue: com.apple.main-thread (serial)
spatch_async_swift_job
ift_task_enqueueGlobal
ift_task_switchImpl(swift::AsyncContext*, void (swift:...
osure #1 in ViewModel.run()
artial apply for closure #1 in ViewModel.run()
unk for @escaping @callee_guaranteed @Sendable @...
artial apply for thunk for @escaping @callee_guarante...
d 2 Queue: com.apple.root....ooperative (concurrent)
swift_release_dealloc
ol swift::RefCounts<swift::RefCountBitsT<(swift::RefC...
rray.append(_:)
epository.add(value:)
peration.execute(value:)
osure #1 in ViewModel.run()
artial apply for closure #1 in ViewModel.run()
unk for @escaping @callee_guaranteed @Sendable @...
artial apply for thunk for @escaping @callee_guarante...
d 3 Queue: com.apple.root....ooperative (concurrent)
swift_release_dealloc
ol swift::RefCounts<swift::RefCountBitsT<(swift::RefC...
rray.append(_:)
epository.add(value:)
peration.execute(value:)
osure #1 in ViewModel.run()
```

```
1
2  import SwiftUI
3
4  class Repository {
5      var results: [Int] = []
6
7      func add(value: Int) {
8          results.append(value)        Thread 2: EXC_BAD_ACCESS (code=1, address=0xf
9      }
10 }
11
12 @MainActor
13 class ViewModel: ObservableObject {
14     var repository = Repository()
15
16     func run() async {
17         for value in 0..<1000 {
18             Task {
19                 await Operation(repository: self.repository).execute(value: va
20             }
21         }
22     }
23 }
24
25 struct Operation {
26     let repository: Repository
27
28     func execute(value: Int) async {
29         repository.add(value: value)
30     }
31 }
32
```

Data Races



```
# S3: Wait for a bucket to exist.
bucket.wait_until_exists()

# EC2: Wait for an instance to reach the running state.
instance.wait_until_running()
```

Multithreading or multiprocessing with resources

Resource instances are **not** thread safe and should not be shared across threads or processes. These special classes contain additional meta data that cannot be shared. It's recommended to create a new Resource for each thread or process:

```
import boto3
import boto3.session
import threading

class MyTask(threading.Thread):
    def run(self):
        # Here we create a new session per thread
        session = boto3.session.Session()

        # Next, we create a resource client using our thread's session object
        s3 = session.resource('s3')

        # Put your thread-safe code here
```

In the example above, each thread would have its own Boto3 session and its own instance of the S3 resource. This is a good idea because resources contain shared data when loaded and calling actions, accessing properties, or manually loading or reloading the resource can modify this data.

Previous Low-level clients

Next Session

Read The Documentation

Use the Compiler

Sendable

Sendable

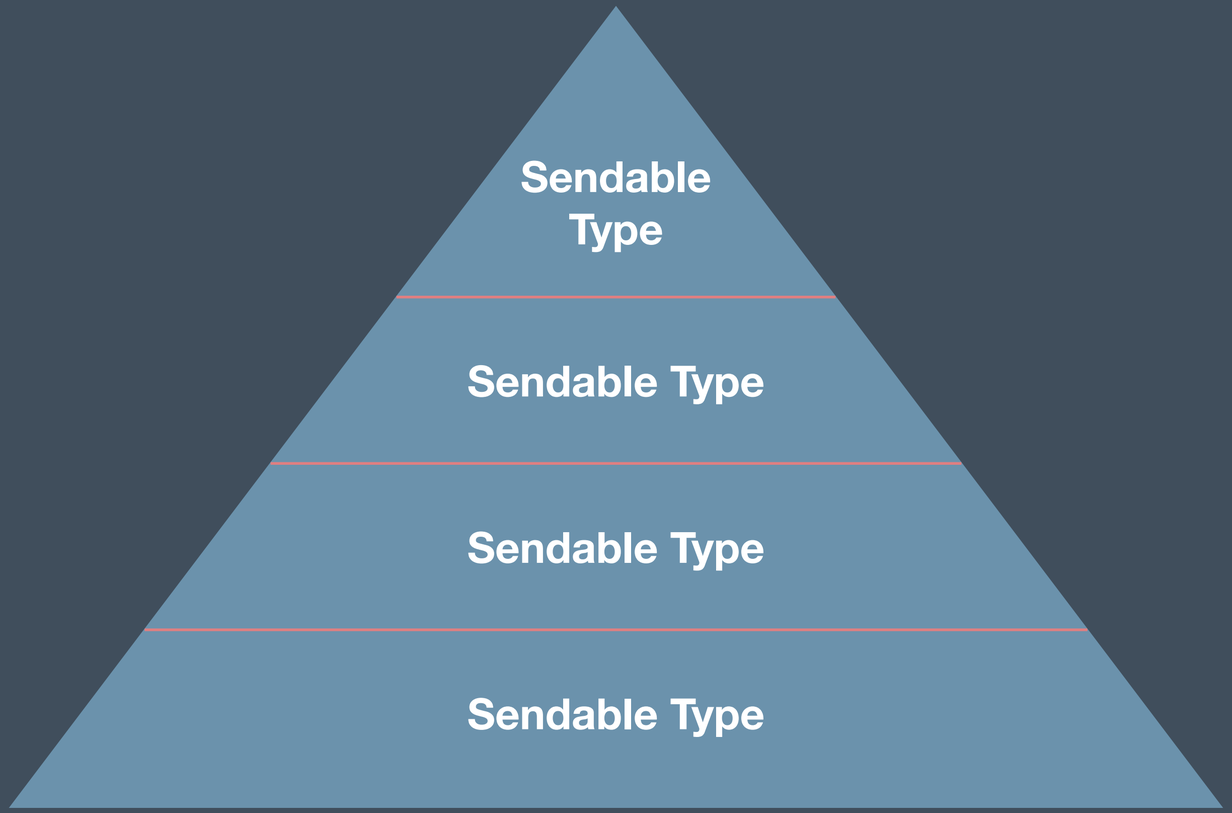
- SE-0302
- Part of Swift Concurrency
- Provides compiler guarantees of data safety in concurrent environments
- *Similar* to Rust's memory guarantees
- Think thread safety!
- Moves data race errors from runtime to compile time 🎉


Threads
Tasks
Corporate needs you to find the differences between this picture and this picture.
They're the same picture.
Made with Piñata Farms

Sendable

- SE-0302
- Part of Swift Concurrency
- Provides compiler guarantees of data safety in concurrent environments
- *Similar* to Rust's memory guarantees
- Think thread safety!
- Moves data race errors from runtime to compile time 🎉

Sendable



```
struct PersonData: Codable, Sendable {
    let name: String
    let age: Int
    let phoneNumber: String
}
```


Sendable Types

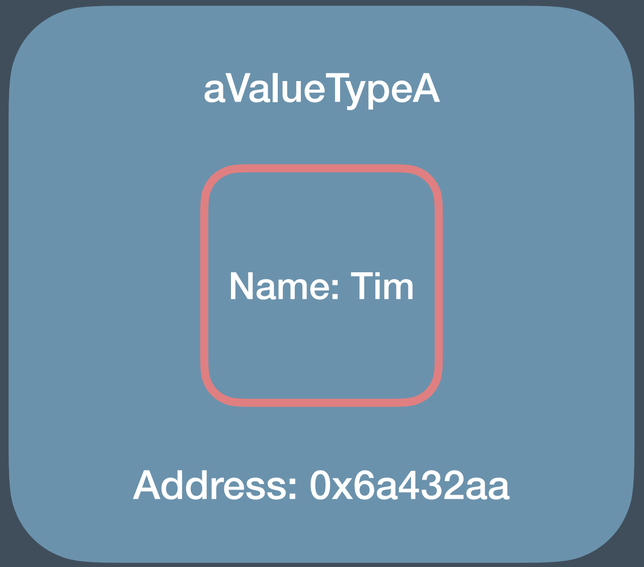
```
/// Most base types are Sendable

public struct Date : Comparable,
Hashable, Equatable, Sendable {
    /// ...
}
```


Sendable Types

```
// Value types are Sendable

struct PersonData: Codable, Sendable {
    var name: String
    var age: String
    var phoneNumber: String
}
```


Copy On Write

```
let aValueTypeA = AValueType(name: "Tim")
var aValueTypeB = aValueTypeA
```



Copy On Write

```
let aValueTypeA = AValueType(name: "Tim")
var aValueTypeB = aValueTypeA
aValueTypeB.name = "Alice"
```

Sendable Types

```
// This is implicit in the same target
extension PersonData: Sendable {}
```


Sendable Types

```
// Anything conforming to this must be Sendable

protocol SuperAwesomeThing: Sendable {
    func doSomethingSafe()
}
```


Sendable Types

```
// All properties must be Sendable

struct SomeGenericThing<T> {
    let theThing: T
}

extension SomeGenericThing: Sendable where T: Sendable {}
```


Sendable Closures

```
public struct Provider: Sendable {
    let run: @Sendable (Application) -> ()

    public init(_ run: @Sendable @escaping (Application) -> ()) {
        self.run = run
    }
}
```


Sendable Reference Types

Sendable Reference Types

```
// MUST be final
final class AReferenceType: Sendable {

    // Cannot be mutable - no CoW
    let nonMutableSendableThing: SomethingSendable

    init(_ aThing: SomethingSendable) {
        self.nonMutableSendableThing = aThing
    }
}
```



SENDABLE
REFERENCE TYPES
SENDABLE
imgflip.com

Sendable Reference Types

```
class Person {
    var name: String
    var age: Int
    var phoneNumber: String

    init(name: String, age: Int, phoneNumber: String) {
        self.name = name
        self.age = age
        self.phoneNumber = phoneNumber
    }
}
```



BLOWN AWAY

Sendable Reference Types

```
class AddressBook {
    var people: [Person]

    init() {
        self.people = []
    }
}
```


Sendable Reference Types

```
// Option 1

@MainActor
class AddressBook {
    var people: [Person]

    init() {
        self.people = []
    }
}
```


Sendable Reference Types

```
// Option 2

actor AddressBook {
    var people: [Person]

    init() {
        self.people = []
    }
}
```


```swift
// Option 3

class AddressBook: @unchecked Sendable {
    private var people: [Person]
    private let lock = NIOLock()

    func addPerson(_ person: Person) {
        lock.withLockVoid {
            people.append(person)
        }
    }

    func getPerson(at index: Int) -> Person {
        lock.withLock {
            people[index]
        }
    }

    init() {
        self.people = []
    }
}
```

@unchecked Sendable

@unchecked Sendable

```swift
// Option 3

class AddressBook: @unchecked Sendable {
    private var people: [Person]
    private let lock = NIOLock()

    func addPerson(_ person: Person) {
        lock.withLockVoid {
            people.append(person)
        }
    }

    func getPerson(at index: Int) -> Person {
        lock.withLock {
            people[index]
        }
    }

    init() {
        self.people = []
    }
}
```


Person is a reference type

```
let addressBook = AddressBook()

let person = Person(name: "Tim", age: 99,
                    phoneNumber: "07123456789")

addressBook.people.append(person)
await storeAddressBookAsync(addressBook)

person.name = "Alice" // 💥
```

Safe Mutable Values

```swift
public struct NIOLockedValueBox<Value> {

    @usableFromInline
    internal let _storage: LockStorage<Value>

    /// Initialize the `Value`.
    @inlinable
    public init(_ value: Value) {
        self._storage = .create(value: value)
    }

    /// Access the `Value`, allowing mutation of it.
    @inlinable
    public func withLockedValue<T>(_ mutate: (inout Value) throws -> T)
      rethrows -> T {
        return try self._storage.withLockedValue(mutate)
    }
}

extension NIOLockedValueBox: Sendable where Value: Sendable {}
```

Sendable Reference Types

```
// Option 4 - IF Person is Sendable

final class AddressBook: Sendable {
    private let people: NIOLockedValueBox<[Person]>

    init() {
        self.people = .init([])
    }
}
```

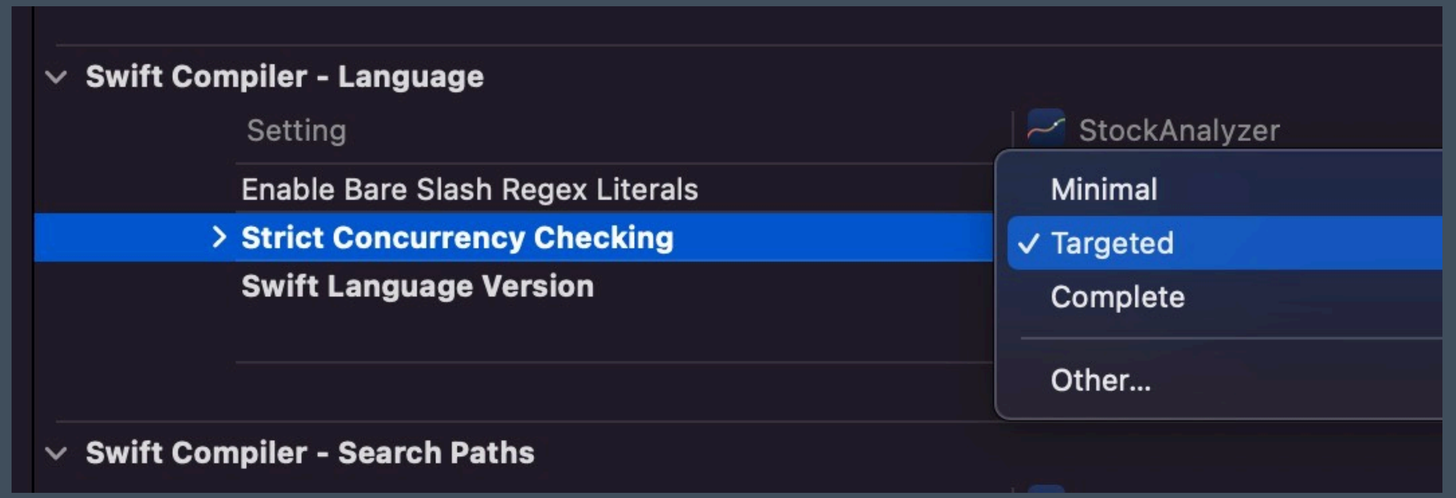

```swift
public struct NIOLoopBound<Value>: @unchecked Sendable {
    public let _eventLoop: EventLoop

    @usableFromInline
    /* private */ var _value: Value

    /// Initialise a ``NIOLoopBound`` to `value` with the precondition that the code is running on `eventLoop`.
    @inlinable
    public init(_ value: Value, eventLoop: EventLoop) {
        eventLoop.preconditionInEventLoop()
        self._eventLoop = eventLoop
        self._value = value
    }

    /// Access the `value` with the precondition that the code is running on `eventLoop`.
    /// - note: ``NIOLoopBound`` itself is value-typed, so any writes will only affect the current value.
    @inlinable
    public var value: Value {
        get {
            self._eventLoop.preconditionInEventLoop()
            return self._value
        }
        set {
            self._eventLoop.preconditionInEventLoop()
            self._value = newValue
        }
    }
}
```

Sendable Reference Types

```
// Option 5

final class AddressBook: Sendable {
    private let people: NIOLoopBoundBox<[Person]>

    init(eventLoop: EventLoop) {
        self.people = .init([], eventLoop: eventLoop)
    }
}
```


The Sendable Journey

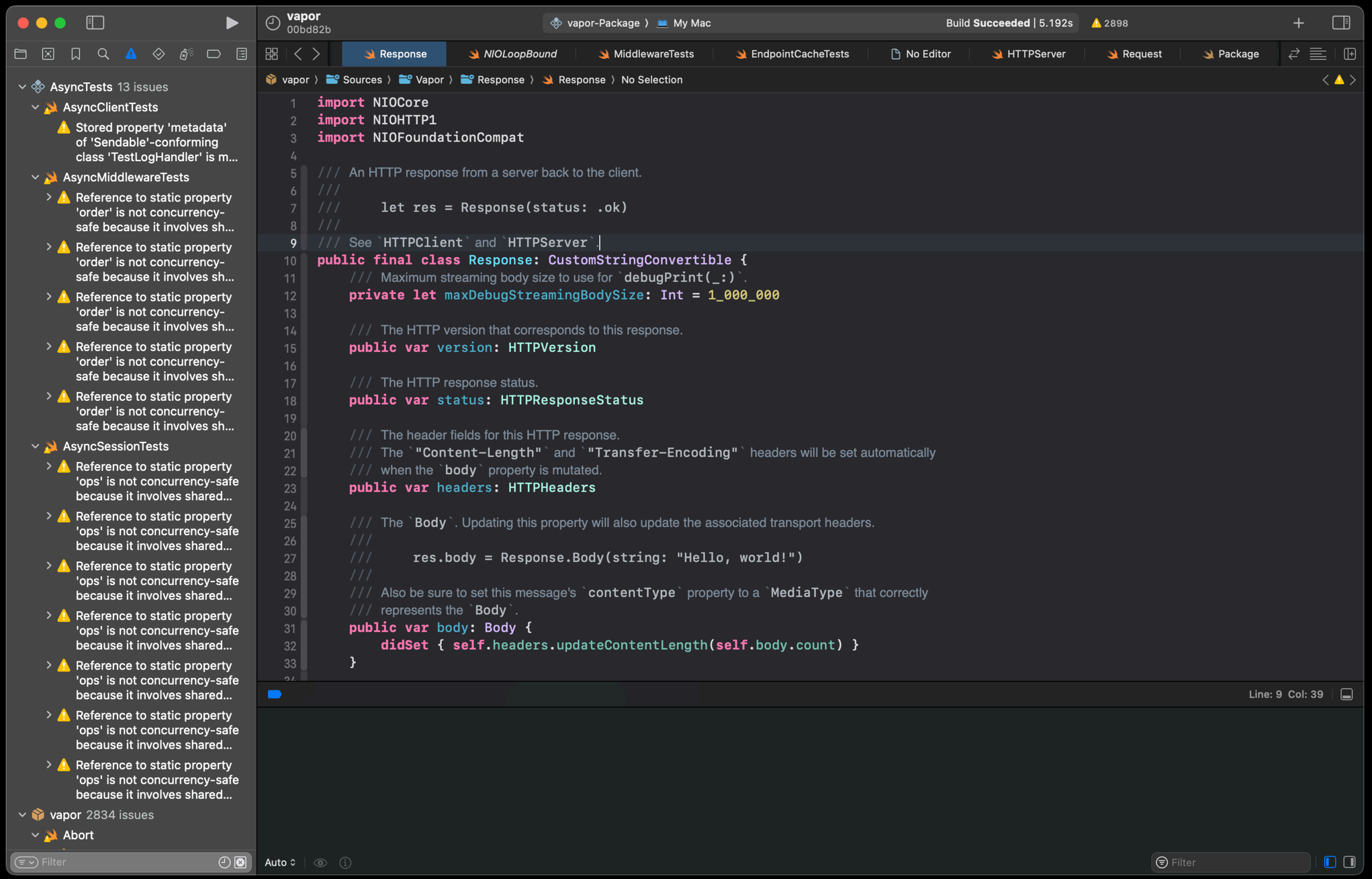
The Sendable Journey



Credit: Antoine van der Lee

The Sendable Journey

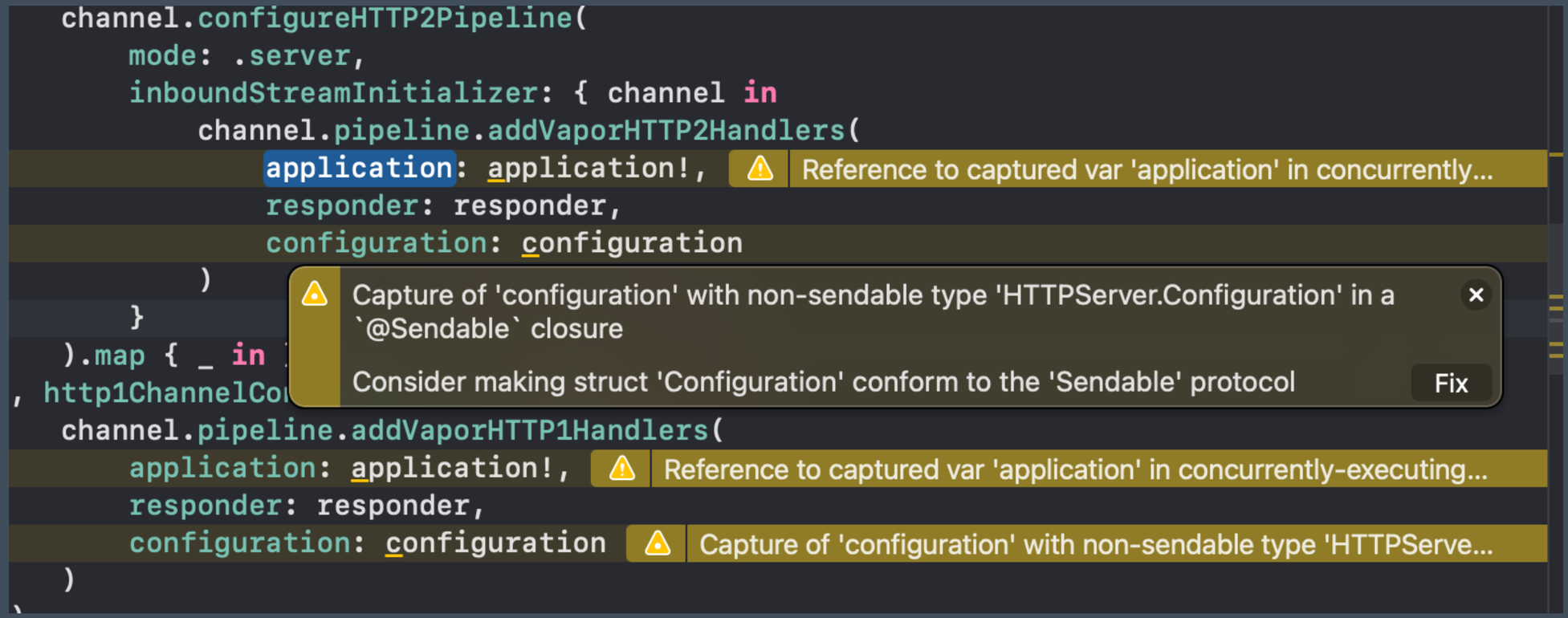
```
// swift-tools-version:5.9
import PackageDescription

let package = Package(
    name: "vapor",
    dependencies: [
        // ..
    ],
    targets: [
        .target(name: "Vapor", dependencies: [
          // ..
        ],
        swiftSettings: [
          .enableExperimentalFeature("StrictConcurrency=complete")
        ]),
    ]
)
```



vapor
00bd82b
vapor-Package ⟩ My Mac
Build Succeeded | 5.192s
2898
Response
NIOLoopBound
MiddlewareTests
EndpointCacheTests
No Editor
HTTPServer
Request
Package
vapor ⟩ Sources ⟩ Vapor ⟩ Response ⟩ Response ⟩ No Selection
AsyncTests 13 issues
AsyncClientTests
Stored property 'metadata' of 'Sendable'-conforming class 'TestLogHandler' is m...
AsyncMiddlewareTests
Reference to static property 'order' is not concurrency-safe because it involves sh...
AsyncSessionTests
Reference to static property 'ops' is not concurrency-safe because it involves shared...
vapor 2834 issues
Abort
import NIOCore
import NIOHTTP1
import NIOFoundationCompat
/// An HTTP response from a server back to the client.
///
///   let res = Response(status: .ok)
///
/// See `HTTPClient` and `HTTPServer`.
public final class Response: CustomStringConvertible {
/// Maximum streaming body size to use for `debugPrint(_:)`.
private let maxDebugStreamingBodySize: Int = 1_000_000
/// The HTTP version that corresponds to this response.
public var version: HTTPVersion
/// The HTTP response status.
public var status: HTTPResponseStatus
/// The header fields for this HTTP response.
/// The `"Content-Length"` and `"Transfer-Encoding"` headers will be set automatically
/// when the `body` property is mutated.
public var headers: HTTPHeaders
/// The `Body`. Updating this property will also update the associated transport headers.
///
///   res.body = Response.Body(string: "Hello, world!")
///
/// Also be sure to set this message's `contentType` property to a `MediaType` that correctly
/// represents the `Body`.
public var body: Body {
didSet { self.headers.updateContentLength(self.body.count) }
}
Line: 9 Col: 39
Auto
Filter
Filter

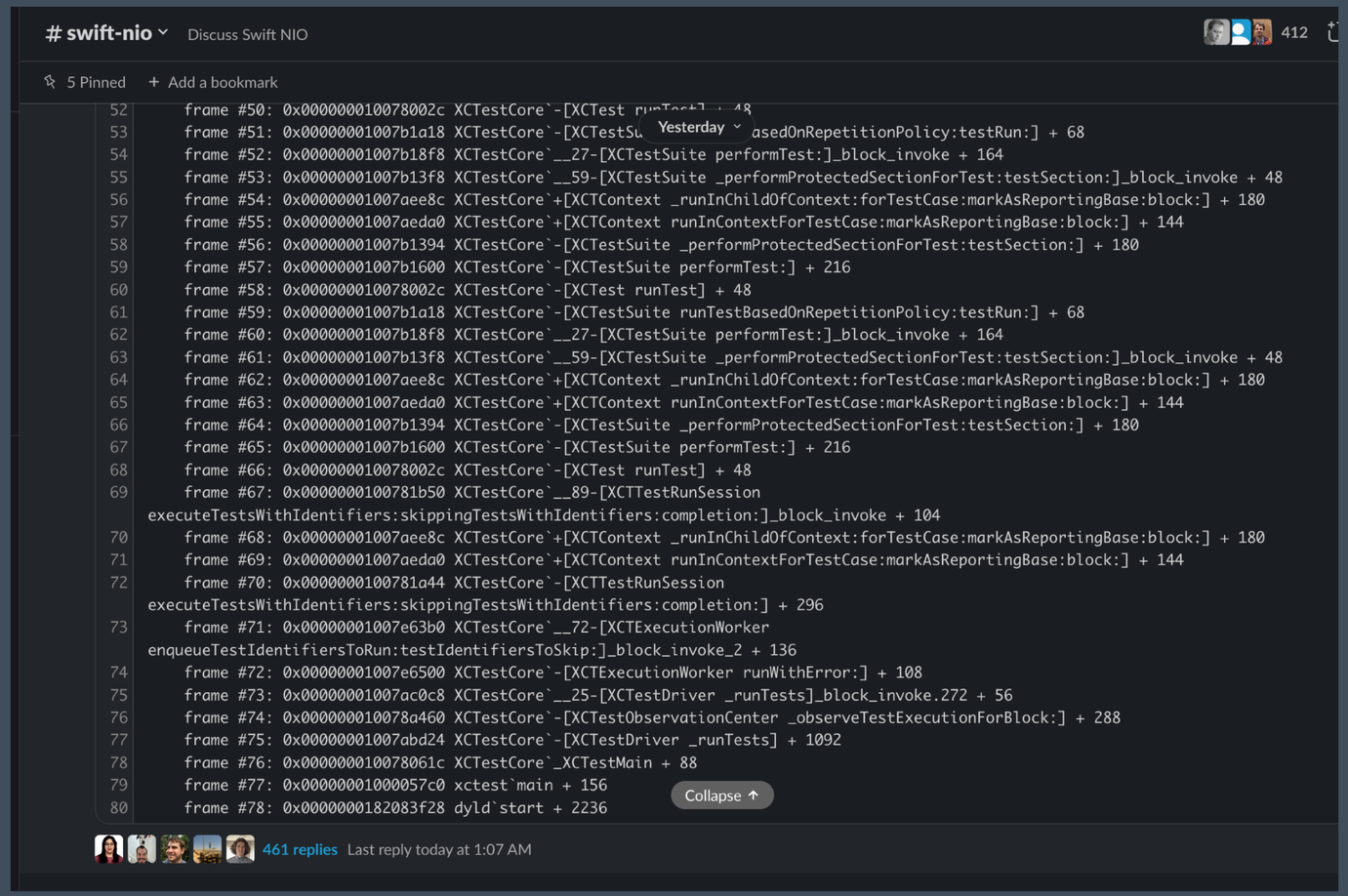
```
func handle(_ req: Request, _ ws: WebSocketKit.WebSocket) {
    var client: Client?

    self.application.ws.knownEventLoop.submit { [weak self] in
        guard let self = self else { return }
        let c = Client(self, req, ws, logger: self.logger)
        client = c
        self._clients.append(c)
    }.whenComplete { [weak self] _ in
        guard
            let client = client,
            let self = self
        else { return }
        self._on(open: client)
        self.on(open: client)
        self.logger.info("[⚡] 🟢 new connection \(client.id)")
    }

    ws.onPing { [weak self] _ in
        guard let client = client else { return }
        self?.log
        self?._or
        self?.on(ping: client)
    }
```

Reference to captured var 'client' in concurrently-executing code

Sendable


channel.configureHTTP2Pipeline(
mode: .server,
inboundStreamInitializer: { channel in
channel.pipeline.addVaporHTTP2Handlers(
application: application!,
Reference to captured var 'application' in concurrently...
responder: responder,
configuration: configuration
)
}
).map { _ in
, http1ChannelCo
Capture of 'configuration' with non-sendable type 'HTTPServer.Configuration' in a `@Sendable` closure
Consider making struct 'Configuration' conform to the 'Sendable' protocol
Fix
channel.pipeline.addVaporHTTP1Handlers(
application: application!,
Reference to captured var 'application' in concurrently-executing...
responder: responder,
configuration: configuration
Capture of 'configuration' with non-sendable type 'HTTPServe...
)

Sendable


swift-nio
Discuss Swift NIO
412
5 Pinned
Add a bookmark
Yesterday
frame #50: 0x000000010078002c XCTestCore`-[XCTest runTest] + 48
frame #51: 0x00000001007b1a18 XCTestCore`-[XCTestSu asedOnRepetitionPolicy:testRun:] + 68
frame #52: 0x00000001007b18f8 XCTestCore`__27-[XCTestSuite performTest:]_block_invoke + 164
frame #53: 0x00000001007b13f8 XCTestCore`__59-[XCTestSuite _performProtectedSectionForTest:testSection:]_block_invoke + 48
frame #54: 0x00000001007aee8c XCTestCore`+[XCTContext _runInChildOfContext:forTestCase:markAsReportingBase:block:] + 180
frame #55: 0x00000001007aeda0 XCTestCore`+[XCTContext runInContextForTestCase:markAsReportingBase:block:] + 144
frame #56: 0x00000001007b1394 XCTestCore`-[XCTestSuite _performProtectedSectionForTest:testSection:] + 180
frame #57: 0x00000001007b1600 XCTestCore`-[XCTestSuite performTest:] + 216
frame #58: 0x000000010078002c XCTestCore`-[XCTest runTest] + 48
frame #59: 0x00000001007b1a18 XCTestCore`-[XCTestSuite runTestBasedOnRepetitionPolicy:testRun:] + 68
frame #60: 0x00000001007b18f8 XCTestCore`__27-[XCTestSuite performTest:]_block_invoke + 164
frame #61: 0x00000001007b13f8 XCTestCore`__59-[XCTestSuite _performProtectedSectionForTest:testSection:]_block_invoke + 48
frame #62: 0x00000001007aee8c XCTestCore`+[XCTContext _runInChildOfContext:forTestCase:markAsReportingBase:block:] + 180
frame #63: 0x00000001007aeda0 XCTestCore`+[XCTContext runInContextForTestCase:markAsReportingBase:block:] + 144
frame #64: 0x00000001007b1394 XCTestCore`-[XCTestSuite _performProtectedSectionForTest:testSection:] + 180
frame #65: 0x00000001007b1600 XCTestCore`-[XCTestSuite performTest:] + 216
frame #66: 0x000000010078002c XCTestCore`-[XCTest runTest] + 48
frame #67: 0x0000000100781b50 XCTestCore`__89-[XCTTestRunSession executeTestsWithIdentifiers:skippingTestsWithIdentifiers:completion:]_block_invoke + 104
frame #68: 0x00000001007aee8c XCTestCore`+[XCTContext _runInChildOfContext:forTestCase:markAsReportingBase:block:] + 180
frame #69: 0x00000001007aeda0 XCTestCore`+[XCTContext runInContextForTestCase:markAsReportingBase:block:] + 144
frame #70: 0x0000000100781a44 XCTestCore`-[XCTTestRunSession executeTestsWithIdentifiers:skippingTestsWithIdentifiers:completion:] + 296
frame #71: 0x00000001007e63b0 XCTestCore`__72-[XCTExecutionWorker enqueueTestIdentifiersToRun:testIdentifiersToSkip:]_block_invoke_2 + 136
frame #72: 0x00000001007e6500 XCTestCore`-[XCTExecutionWorker runWithError:] + 108
frame #73: 0x00000001007ac0c8 XCTestCore`__25-[XCTestDriver _runTests]_block_invoke.272 + 56
frame #74: 0x000000010078a460 XCTestCore`-[XCTestObservationCenter _observeTestExecutionForBlock:] + 288
frame #75: 0x00000001007abd24 XCTestCore`-[XCTestDriver _runTests] + 1092
frame #76: 0x000000010078061c XCTestCore`_XCTestMain + 88
frame #77: 0x00000001000057c0 xctest`main + 156
frame #78: 0x0000000182083f28 dyld`start + 2236
Collapse
461 replies
Last reply today at 1:07 AM

Maintaining Your API

Backwards Compatibility

```
// With @preconcurrency, adopters don’t get warnings
// unless complete currency checking is turned on

@preconcurrency public func onPong(
   _ callback: @Sendable @escaping (WebSocket) -> ()) {
     self.onPongCallback.value = callback
}
```


Backwards Compatibility

```
// With @preconcurrency import, remove warnings
// when using non-Sendable types from dependencies

@preconcurrency import ConsoleKit
```


Maintaining the API

```
// What about mutable properties in a class?

public var method: HTTPMethod
```


Maintaining the API

```
private let _method: NIOLockedValueBox<HTTPMethod>

public var method: HTTPMethod {
    get {
        return _method.withLockedValue { $0 }
    }
    set {
        _method.withLockedValue { $0 = newValue }
    }
}
```


Gotchas

- It’s still evolving including helpers, Objective-C interop, frameworks, dependencies
- Each version of Swift will add more checking
- Sendable conformances ***must*** be done in the same file
- Sendable functions currently don’t work

Recap

- Sendable provides code annotations for the compiler to guarantee data safety in concurrent environments
- Don’t use `@unchecked Sendable` unless you’re *really* sure
- Start adopting now!

Thank you!

@0xTim

BROKENHANDS