

A Deep Dive Into Sendable

Tim Condon

@0xTim



BROKENHANDS



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 
- Organise ServerSide.swift, NSManchester, Vapor London
- @0xTim     



Why

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



Undefined Behaviour





**RUNTIME
ERRORS**



**COMPILE
TIME ERRORS**



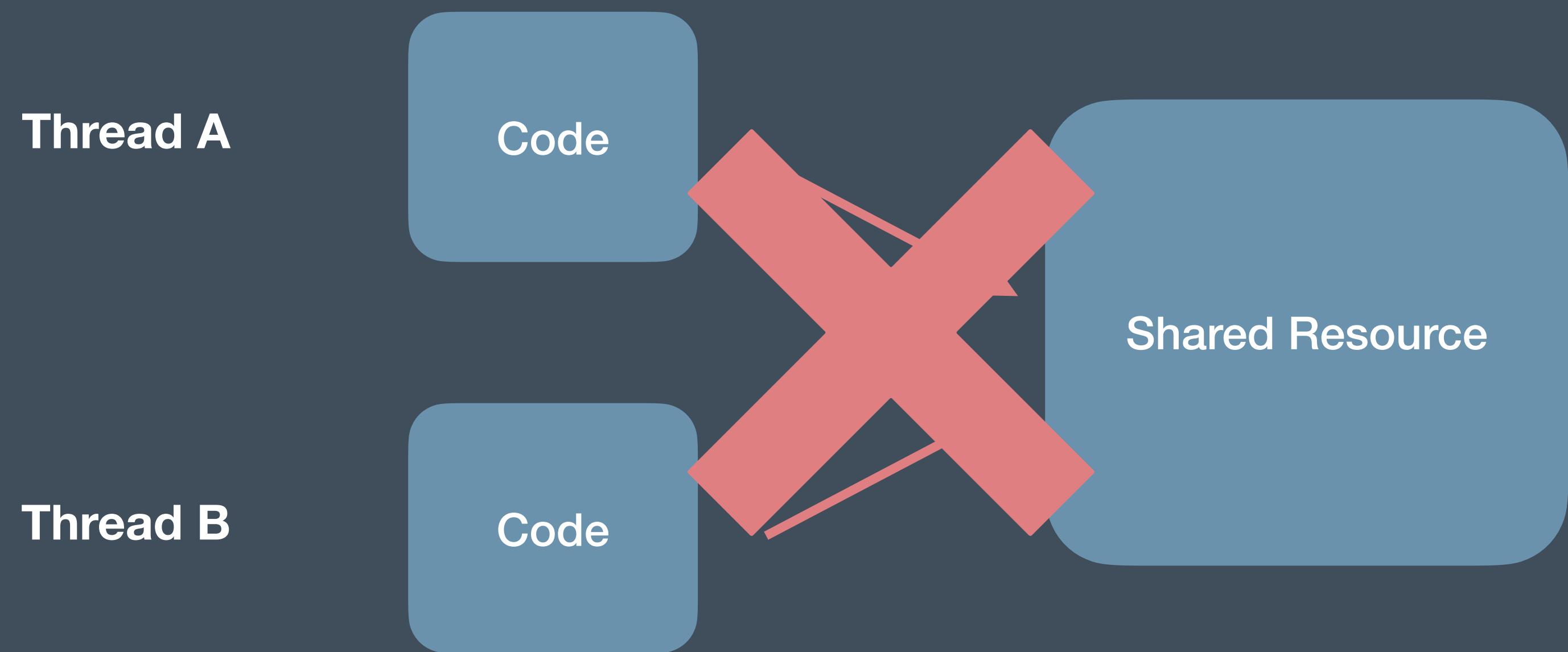
SHIP IT

Data Races

```
d 1 Queue: com.apple.main-thread (serial)
spatch_async_swift_job
Swift_task_enqueueGlobal
Swift_task_switchImpl(swift::AsyncContext*, void (swift:...
closure #1 in ViewModel.run()
partial apply for closure #1 in ViewModel.run()
thunk for @escaping @callee_guaranteed @Sendable @...
partial apply for thunk for @escaping @callee_guarante...
d 2 Queue: com.apple.root....operative (concurrent) ⚠
swift_release_dealloc
bool swift::RefCounts<swift::RefCountBitsT<(swift::RefC...
array.append(_:)
repository.add(value:)
operation.execute(value:)
closure #1 in ViewModel.run()
partial apply for closure #1 in ViewModel.run()
thunk for @escaping @callee_guaranteed @Sendable @...
partial apply for thunk for @escaping @callee_guarante...
d 3 Queue: com.apple.root....operative (concurrent) ⚠
swift_release_dealloc
bool swift::RefCounts<swift::RefCountBitsT<(swift::RefC...
array.append(_:)
repository.add(value:)
operation.execute(value:)
closure #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



The key word arguments Examples or Waiters include:

```
# 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](#)

ON THIS PAGE

- Overview
- Identifiers and attributes
- Actions
- References
- Sub-resources
- Waiters
- Multithreading or multiprocessing with resources**

Feedback

Did you find this page useful?
improve this website or boto3?
[Give us feedback.](#)

Quickstart

A Sample Tutorial

Code Examples

Developer Guide

Configuration

Credentials

Low-level clients

Resources

Session

Collections

Paginator

Error handling

Retries

Extensibility guide

Copyright © 2023, Amazon Web Services, Inc
Made with Sphinx and @pradyunsg's Furo
[Privacy](#) | [Site Terms](#) | [Cookie preferences](#)



**READ
THE DOCS**

**INHERENT
KNOWLEDGE**

**CODE
REVIEW**

**COMPILER
ERRORS**



What

Sendable

A screenshot of a GitHub repository page for the Swift Evolution project. The repository is named `apple / swift-evolution`. The active tab is `Code`, showing the file `swift-evolution / proposals / 0302-concurrent-value-and-concurrent-closures.md`. A pull request by `JTurcotti` titled "Fix typo in 0302-concurrent-value-and-concurrent-closures.md (#2060)" is visible, with a green checkmark indicating it has been merged. The status bar shows "3ab911e · last year" and a link to "History". The page includes a "Preview" tab, a "Code" tab, and a "Blame" tab. The main content area features a large heading "Sendable and @Sendable closures" and a bulleted list of proposal details. Below this is a "Contents" section with a hierarchical list of links.

- Proposal: [SE-0302](#)
- Authors: [Chris Lattner](#), [Doug Gregor](#)
- Review Manager: [John McCall](#)
- Status: **Implemented (Swift 5.7)**
- Implementation: [apple/swift#35264](#)
- Major Contributors: Dave Abrahams, Paul Cantrell, Matthew Johnson, John McCall
- Review: ([first review](#)) ([revision announcement](#)) ([second review](#)) ([acceptance](#))

Contents

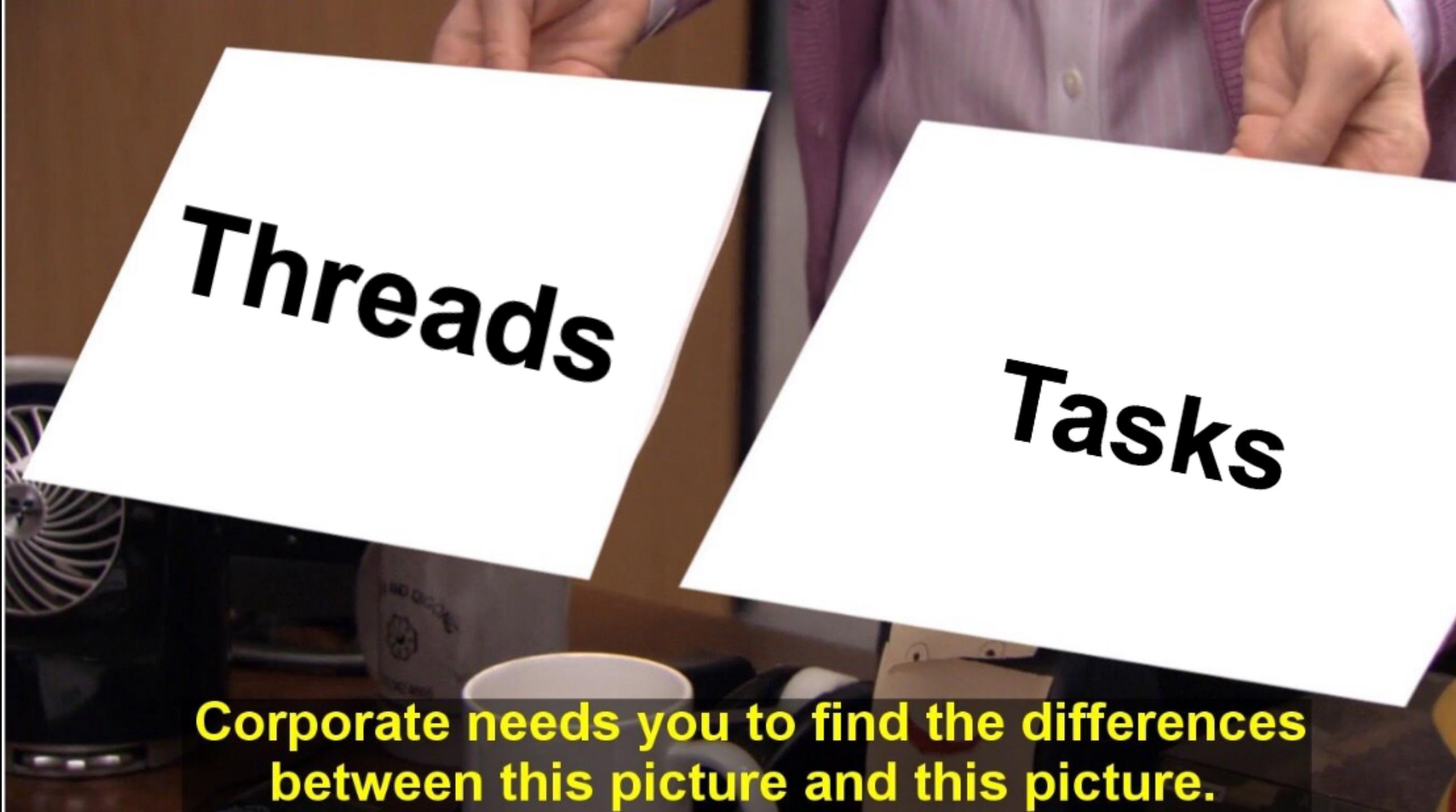
- [Introduction](#)
- [Motivation](#)
 - [Swift Value Semantics](#)
 - [Value Semantic Composition](#)
 - [Higher Order Functional Programming](#)
 - [Immutable Classes](#)
 - [Internally Synchronized Reference Types](#)
 - ["Transferring" Objects Between Concurrency Domains](#)
 - [Deep Copying Classes](#)

SE-0302

Sendable provides compiler guarantees of data safety in concurrent environments

Sendable ≈ Rust's Memory Safety

Sendable == Thread Safety



Threads

Tasks

Corporate needs you to find the differences
between this picture and this picture.



Made with Piñata Farms **They're the same picture.**



**RUNTIME
ERRORS**



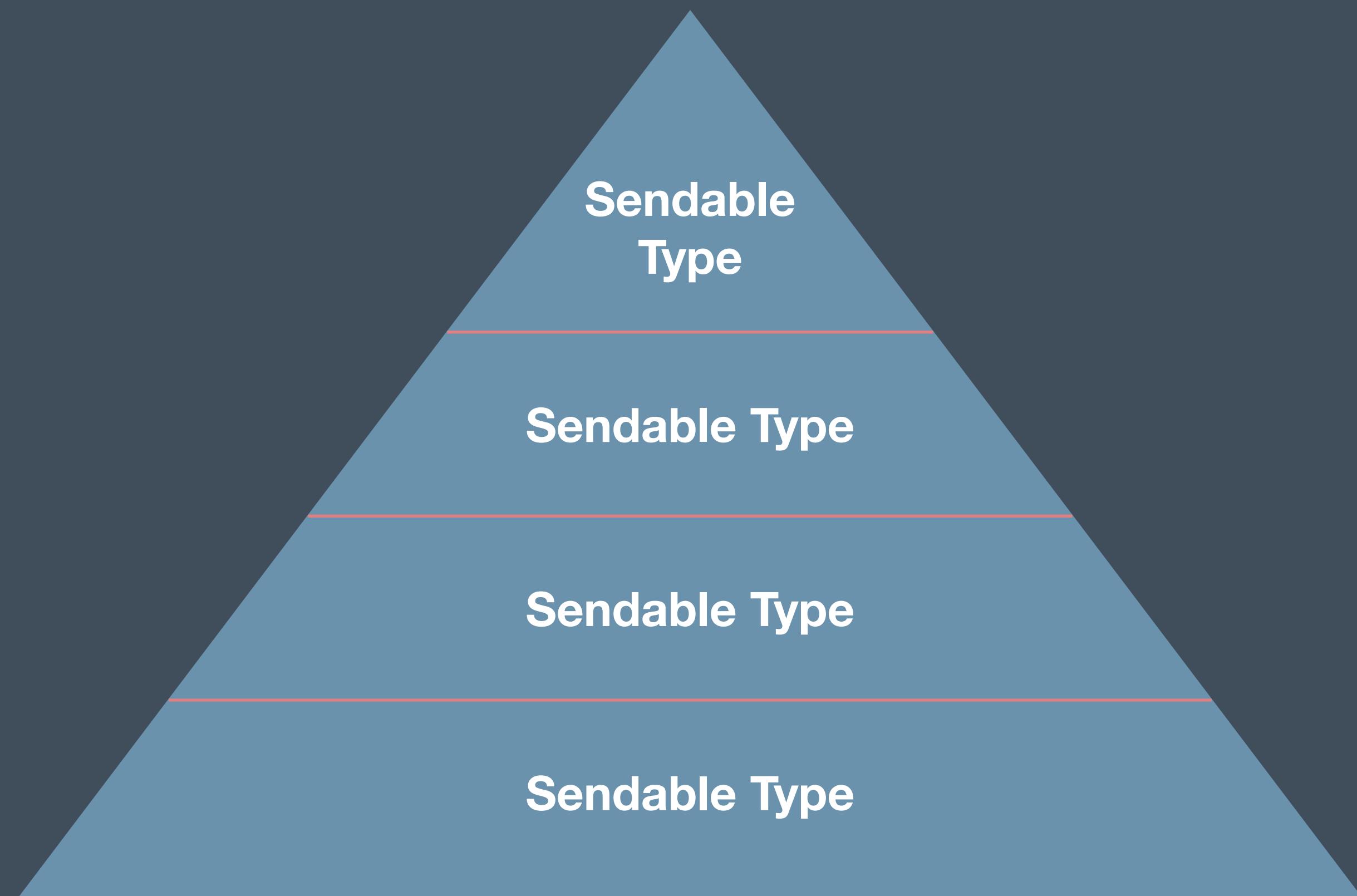
**COMPILE
TIME ERRORS**

How

A screenshot of a GitHub browser window displaying the `Sendable.swift` file from the `swift / stdlib / public / core` repository. The window has a dark theme. At the top, there are navigation icons, a search bar with the text "github.com", and a tab bar with "main". Below the header, the file path "swift / stdlib / public / core / Sendable.swift" is shown, along with a "Top" link. The main content area contains tabs for "Code" (which is selected) and "Blame", and a set of download and edit buttons. The code itself is annotated with numerous triple slashes (///) providing documentation and notes about the protocol's requirements and behavior.

```
128     /// To satisfy the requirements of the `Sendable` protocol,  
129     /// all of the elements of the tuple must be sendable.  
130     /// Tuples that satisfy the requirements implicitly conform to `Sendable`.  
131     ///  
132     /// ### Sendable Metatypes  
133     ///  
134     /// Metatypes such as `Int.Type` implicitly conform to the `Sendable` protocol.  
135     #if $NoncopyableGenerics && $NonescapableTypes  
136     @_marker public protocol Sendable: ~Copyable, ~Escapable { }  
137     #elseif $NoncopyableGenerics  
138     @_marker public protocol Sendable: ~Copyable { }  
139     #else  
140     @_marker public protocol Sendable { }  
141     #endif  
142     ///  
143     /// A type whose values can safely be passed across concurrency domains by copying,  
144     /// but which disables some safety checking at the conformance site.  
145     ///  
146     /// Use an unchecked conformance to `Sendable` instead --- for example:  
147     ///  
148     ///     struct MyStructure: @unchecked Sendable { ... }
```

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
}
```



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

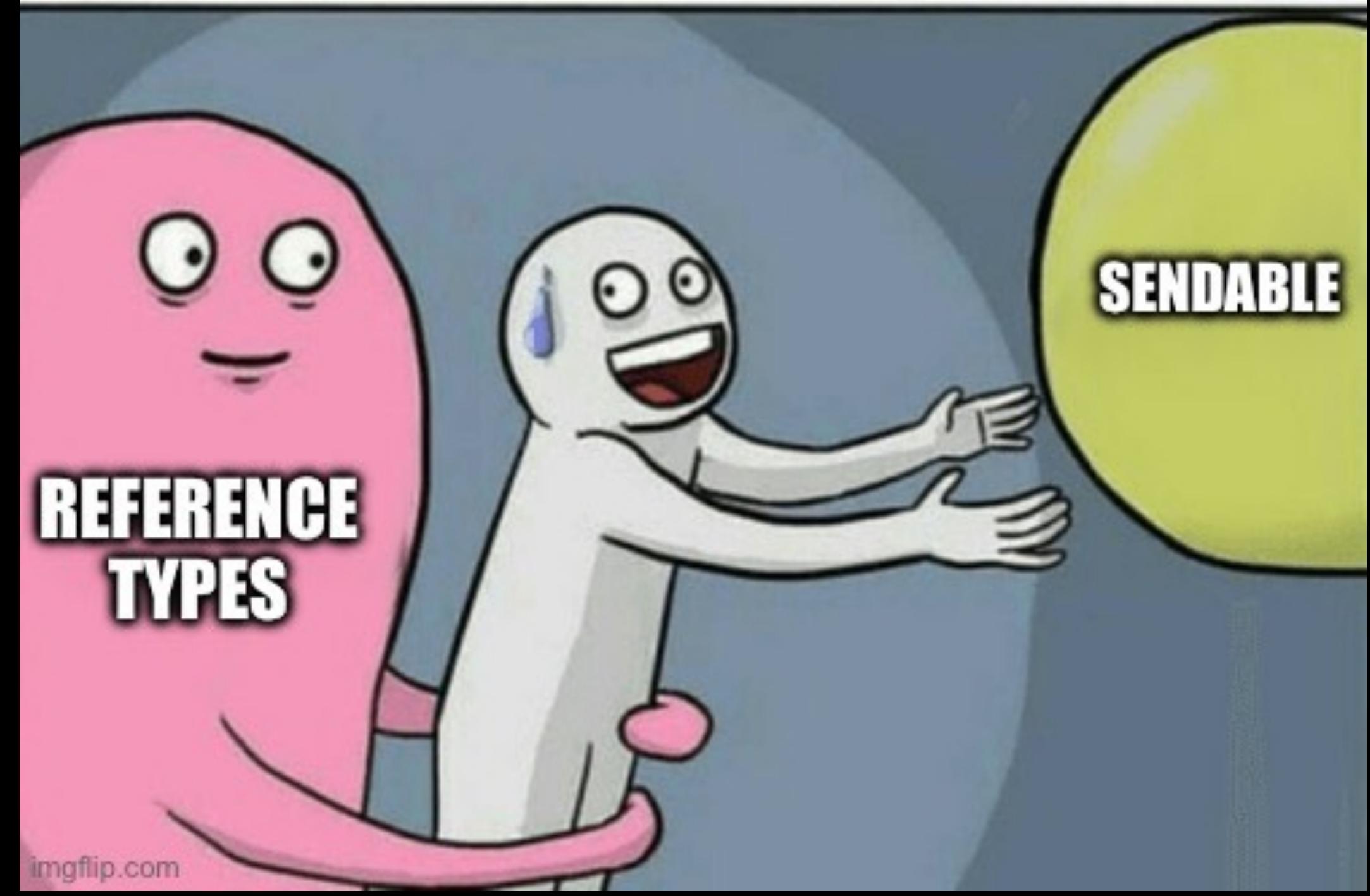
Sendable Reference Types

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

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

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





Sendable Reference Types

```
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  
    }  
}
```



BLAWN 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 = []
    }
}
```



```
// 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



```
// 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.addPerson(person)
await storeAddressBookAsync(addressBook)

person.name = "Alice" // ⚡
```

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

    init() {
        self.people = []
    }

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

    func createPerson(in addressBook: AddressBook) async {
        let person = Person(name: "tim", age: 29, phoneNumber: "1234")
        await addressBook.addPerson(person)
    }
}
```



⚠️ Passing argument of non-sendable type 'Person' into actor-isolated context may introduce data races

Region based Isolation

- Proposal: [SE-0414](#)
- Authors: [Michael Gottesman](#) [Joshua Turcotti](#)
- Review Manager: [Holly Borla](#)
- Status: Accepted
- Implementation: On `main` gated behind `-enable-experimental-feature RegionBasedIsolation`
- Review: ([first pitch](#)), ([second pitch](#)), ([first review](#)), ([revision](#)), ([second review](#)), ([acceptance](#))

Introduction

Swift Concurrency assigns values to *isolation domains* determined by actor and task boundaries. Code running in distinct isolation domains can execute concurrently, and `Sendable` checking defines away concurrent access to shared mutable state by preventing non-`Sendable` values from being passed across isolation boundaries full stop. In practice, this is a significant semantic restriction, because it forbids natural programming patterns that are free of data races.

In this document, we propose loosening these rules by introducing a new control flow sensitive diagnostic that determines whether a non-`Sendable` value can safely be transferred over an isolation boundary. This is done by introducing the concept of *isolation regions* that allows the compiler to reason conservatively if two values can affect each other. Through the usage of isolation regions, the language can prove that transferring a non-`Sendable` value over an isolation boundary cannot result in races because the value (and any other value that might reference it) is not used in the caller after the point of transfer.

Motivation

SE-0414

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

    init() {
        self.people = []
    }

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

func createPerson(in addressBook: AddressBook) async {
    let person = Person(name: "tim", age: 29, phoneNumber: "1234")
    await addressBook.addPerson(person)
}
```

Safe Mutable Values

```
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

final class Person: Sendable {
    let name: NIOLockedValueBox<String>
    let age: NIOLockedValueBox<Int>
    let phoneNumber: NIOLockedValueBox<String>

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



Sendable Reference Types

```
// Option 4

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

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



```
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

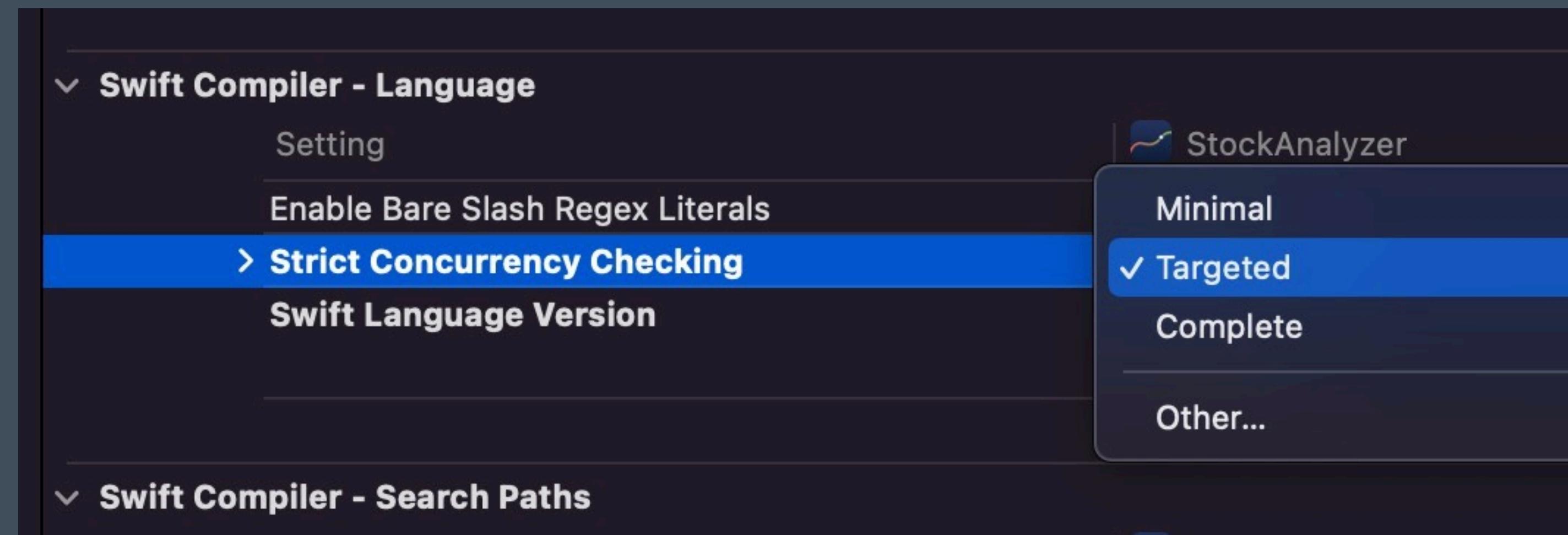
final class AddressBook: Sendable {
    private let people: NIOLoopBoundBox<[Person]>

    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 Filter

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...
 - > ⚠️ Reference to static property 'order' is not concurrency-safe because it involves sh...
 - > ⚠️ Reference to static property 'order' is not concurrency-safe because it involves sh...
 - > ⚠️ Reference to static property 'order' is not concurrency-safe because it involves sh...
 - > ⚠️ Reference to static property 'order' is not concurrency-safe because it involves sh...
 - > ⚠️ Reference to static property 'order' is not concurrency-safe because it involves sh...
 - > ⚠️ Reference to static property 'order' is not concurrency-safe because it involves sh...
 - > ⚠️ Reference to static property 'order' is not concurrency-safe because it involves sh...
 - > ⚠️ Reference to static property 'ops' is not concurrency-safe because it involves shared...
 - > ⚠️ Reference to static property 'ops' is not concurrency-safe because it involves shared...
 - > ⚠️ Reference to static property 'ops' is not concurrency-safe because it involves shared...
 - > ⚠️ Reference to static property 'ops' is not concurrency-safe because it involves shared...
 - > ⚠️ Reference to static property 'ops' is not concurrency-safe because it involves shared...
 - > ⚠️ Reference to static property 'ops' is not concurrency-safe because it involves shared...
 - > ⚠️ Reference to static property 'ops' is not concurrency-safe because it involves shared...
 - > ⚠️ Reference to static property 'ops' is not concurrency-safe because it involves shared...
- AsyncSessionTests
 - > ⚠️ Reference to static property 'ops' is not concurrency-safe because it involves shared...
 - > ⚠️ Reference to static property 'ops' is not concurrency-safe because it involves shared...
 - > ⚠️ Reference to static property 'ops' is not concurrency-safe because it involves shared...
 - > ⚠️ Reference to static property 'ops' is not concurrency-safe because it involves shared...
 - > ⚠️ Reference to static property 'ops' is not concurrency-safe because it involves shared...
 - > ⚠️ Reference to static property 'ops' is not concurrency-safe because it involves shared...
 - > ⚠️ Reference to static property 'ops' is not concurrency-safe because it involves shared...
 - > ⚠️ Reference to static property 'ops' is not concurrency-safe because it involves shared...
- vapor 2834 issues
- Abort

vapor > Sources > Vapor > Response > Response > No Selection

```
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

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(⚠ Reference to captured var 'client' in concurrently-executing code)
        self?._on(ping: client)
        self?.on(ping: client)
    }
}
```

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  
, http1ChannelConfig in  
    channel.pipeline.addVaporHTTP1Handlers(  
        application: application!, ⚠ Reference to captured var 'application' in concurrently-executing...  
        responder: responder,  
        configuration: configuration ⚠ Capture of 'configuration' with non-sendable type 'HTTPServer.Configuration' in a  
        `@Sendable` closure  
    )  
}  
}
```



Sendable

swift-nio ▾ Discuss Swift NIO 412 ↗

5 Pinned + Add a bookmark

```
frame #50: 0x000000010078002c XCTestCore`-[XCTest runTest] + 48
frame #51: 0x00000001007b1a18 XCTestCore`-[XCTestSuite _usedOnRepetitionPolicy: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-[XCTTestDriver _runTests]_block_invoke.272 + 56
frame #74: 0x000000010078a460 XCTestCore`-[XCTestObservationCenter _observeTestExecutionForBlock:] + 288
frame #75: 0x00000001007abd24 XCTestCore`-[XCTTestDriver _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 }
    }
}
```



Recap

Sendable provides compiler guarantees of data safety in concurrent environments

A screenshot of a GitHub repository interface showing multiple pull requests and their details.

The main repository is `swift-evolution / proposals`.

Open pull requests:

- `0422-caller-side-default-argument-macro-expression.md` by `dempseyatgithub`: Correct the chronological order of proposal review discussions (#2402). Status: Merged (green checkmark). Last commit: `27c506c · 3 weeks ago`. History: [View History](#).
- `0430-transferring-parameters-and-results.md` by `DougGregor`: Accept SE-0431 (#2437). Status: Accepted (green checkmark). Last commit: `20f0408 · 4 days ago`. History: [View History](#).

Other files visible in the sidebar include:

- `main`
- `0416-keypath-function-subtyping.md`
- `0417-task-executor-preference.md`
- `0418-inferring-sendable-for-me...`
- `0419-backtrace-api.md`
- `0420-inheritance-of-actor-isolat...`

Content of the accepted pull request (0430-transferring-parameters-and-results.md):

@isolated(any) Function Types

- Proposal: [SE-0431](#)
- Authors: [John McCall](#)
- Review Manager: [Doug Gregor](#)
- Status: Accepted
- Implementation: [apple/swift#71433](#), [apple/swift#71574](#), available on `main` with `-enable-experimental-feature IsolatedAny`.
- Previous revision: [1](#)
- Review: ([pitch](#)) ([review](#)) ([acceptance](#))

Introduction

The actor isolation of a function is an important part of how it's used. Swift can reason precisely about the isolation of a specific function *declaration*, but when functions are passed around as *values*, Swift's function types are not expressive enough to keep up. This proposal adds a new kind of function type that carries its function's actor isolation dynamically. This solves a variety of expressivity problems in the language. It also allows features such as the standard library's task-creation APIs to be implemented more efficiently and with stronger semantic guarantees.

Regions of parameter and result values

TransferringArgsAndResults

it `transferring` annotation to denote when a parameter or result value is boundary. This allows the callee or the caller, respectively, to transfer a non-boundary or merge the value into an actor-isolated region.

sferring non- `Sendable` values over isolation boundaries. In most cases, function same region for any given call. This means that non- `Sendable` parameter values

developer.apple.com

WWDC24

Overview Special Event

WWDC24

Join us online for the biggest developer event of the year. Be there for the unveiling of the latest Apple platforms, technologies, and tools. Learn how to create and elevate your apps and games. Engage with Apple designers and engineers and connect with the worldwide developer community. All online and at no cost.

June 10–14, 2024

Add to calendar

Thank you!

@0xTim



BROKENHANDS



ServerSide.swift

26-27 September 2024, London, UK



Questions?

@0xTim



BROKENHANDS