Working with C/C++ & Swift

Richard Topchii

Pragma Conference 6 October 2023

Motivation

Motivation

Motivation

1. Performance

Sep 2023	Sep 2022	Change	Programming Language	Ratings	Change
1	1		Python	14.16%	-1.58%
2	2		C	11.27%	-2.70%
3	4	↑	C++	10.65%	+0.90%
4	3	↓	Java	9.49%	-2.23%
5	5		C#	7.31%	+2.42%
6	7	↑	JavaScript	3.30%	+0.48%
7	6	↓	Visual Basic	2.22%	-2.18%
8	10	↑	PHP	1.55%	-0.13%
9	8	↓	Assembly language	1.53%	-0.96%
10	9	↓	SQL	1.44%	-0.57%
11	15	⇈	Fortran	1.28%	+0.26%
12	12		Go	1.19%	+0.03%

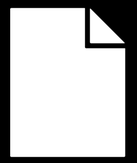
Sep 2023	Sep 2022	Change	Programming Language	Ratings	Change
1	1		Python	14.16%	-1.58%
2	2		C	11.27%	-2.70%
3	4	↑	C++	10.65%	+0.90%
4	3	↓	Java	9.49%	-2.23%
5	5		C#	7.31%	+2.42%
6	7	↑	JavaScript	3.30%	+0.48%
7	6	↓	Visual Basic	2.22%	-2.18%
8	10	↑	PHP	1.55%	-0.13%
9	8	↓	Assembly language	1.53%	-0.96%
10	9	↓	SQL	1.44%	-0.57%
11	15	⇈	Fortran	1.28%	+0.26%
12	12		Go	1.19%	+0.03%

4	3	↓		Java	9.49%	-2.23%
5	5			C#	7.31%	+2.42%
6	7	↑		JavaScript	3.30%	+0.48%
7	6	↓		Visual Basic	2.22%	-2.18%
8	10	↑		PHP	1.55%	-0.13%
9	8	↓		Assembly language	1.53%	-0.96%
10	9	↓		SQL	1.44%	-0.57%
11	15	⇈		Fortran	1.28%	+0.26%
12	12			Go	1.19%	+0.03%
13	14	↑		MATLAB	1.19%	+0.13%
14	22	⇈		Scratch	1.08%	+0.51%
15	13	↓		Delphi/Object Pascal	1.02%	-0.07%
16	16			Swift	1.00%	+0.02%
17	26	⇈		Rust	0.97%	+0.47%
18	18			R	0.97%	+0.02%

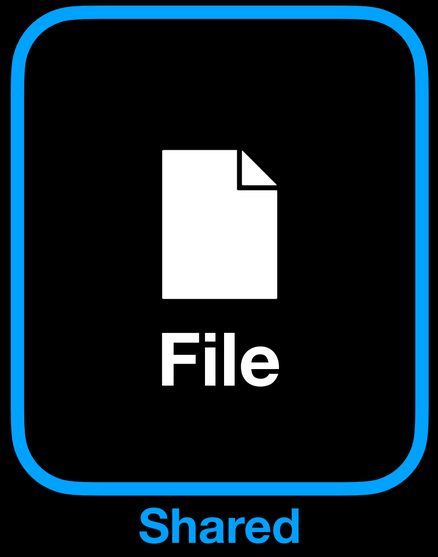
4	3	↓		Java	9.49%	-2.23%
5	5			C#	7.31%	+2.42%
6	7	↑		JavaScript	3.30%	+0.48%
7	6	↓		Visual Basic	2.22%	-2.18%
8	10	↑		PHP	1.55%	-0.13%
9	8	↓		Assembly language	1.53%	-0.96%
10	9	↓		SQL	1.44%	-0.57%
11	15	⇈		Fortran	1.28%	+0.26%
12	12			Go	1.19%	+0.03%
13	14	↑		MATLAB	1.19%	+0.13%
14	22	⇈		Scratch	1.08%	+0.51%
15	13	↓		Delphi/Object Pascal	1.02%	-0.07%
16	16			Swift	1.00%	+0.02%
17	26	⇈		Rust	0.97%	+0.47%
18	18			R	0.97%	+0.02%

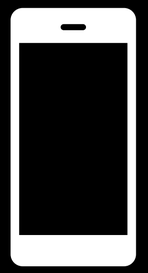
Motivation

1. Code, Data Model and Tests reuse (cross-platform)
2. … without sacrificing the performance
3. … and native-like tooling support

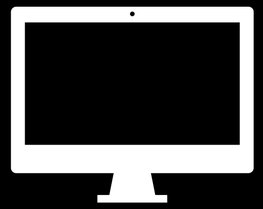


File

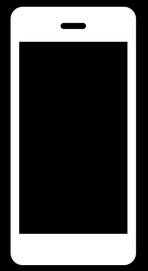

File
Shared



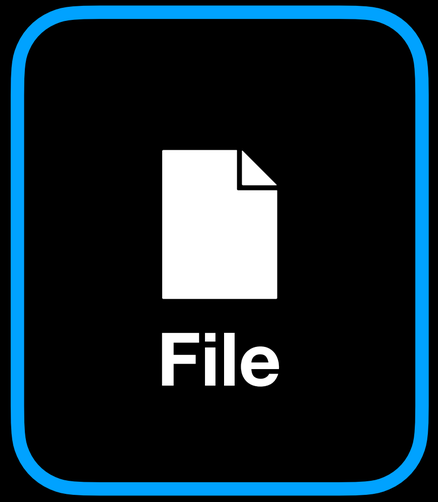
iOS



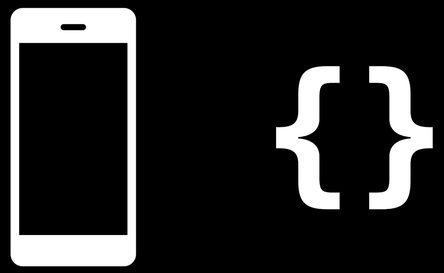
macOS



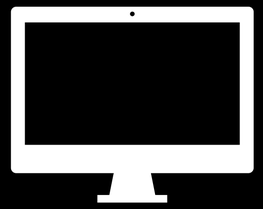
Android



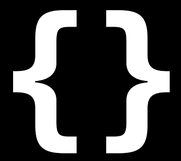
Shared

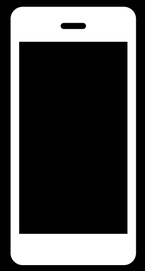


iOS

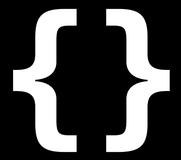


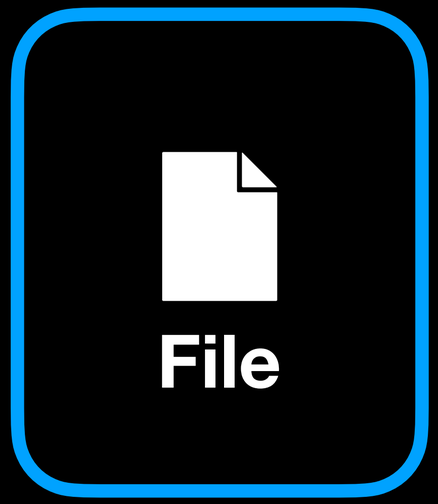
macOS

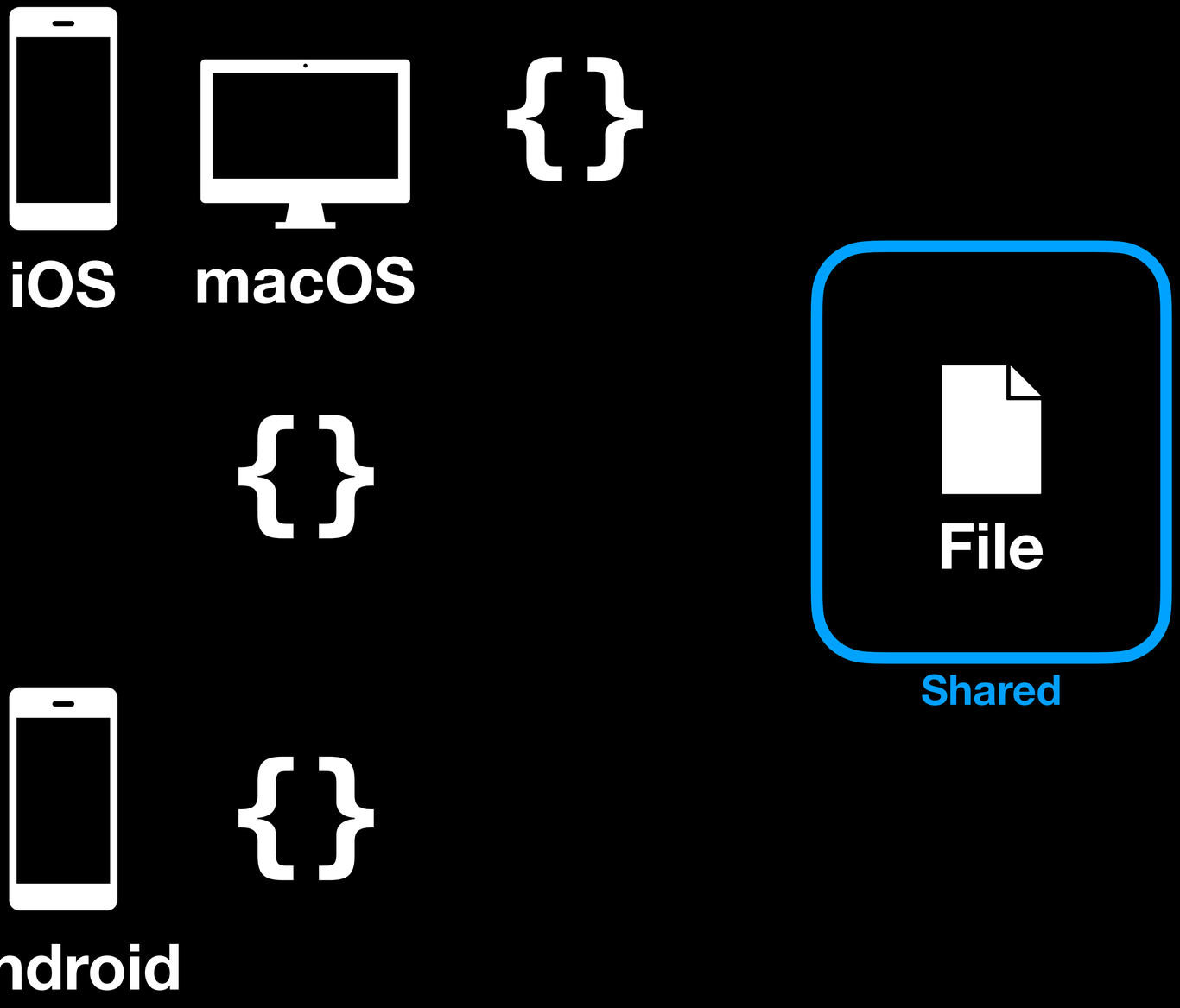


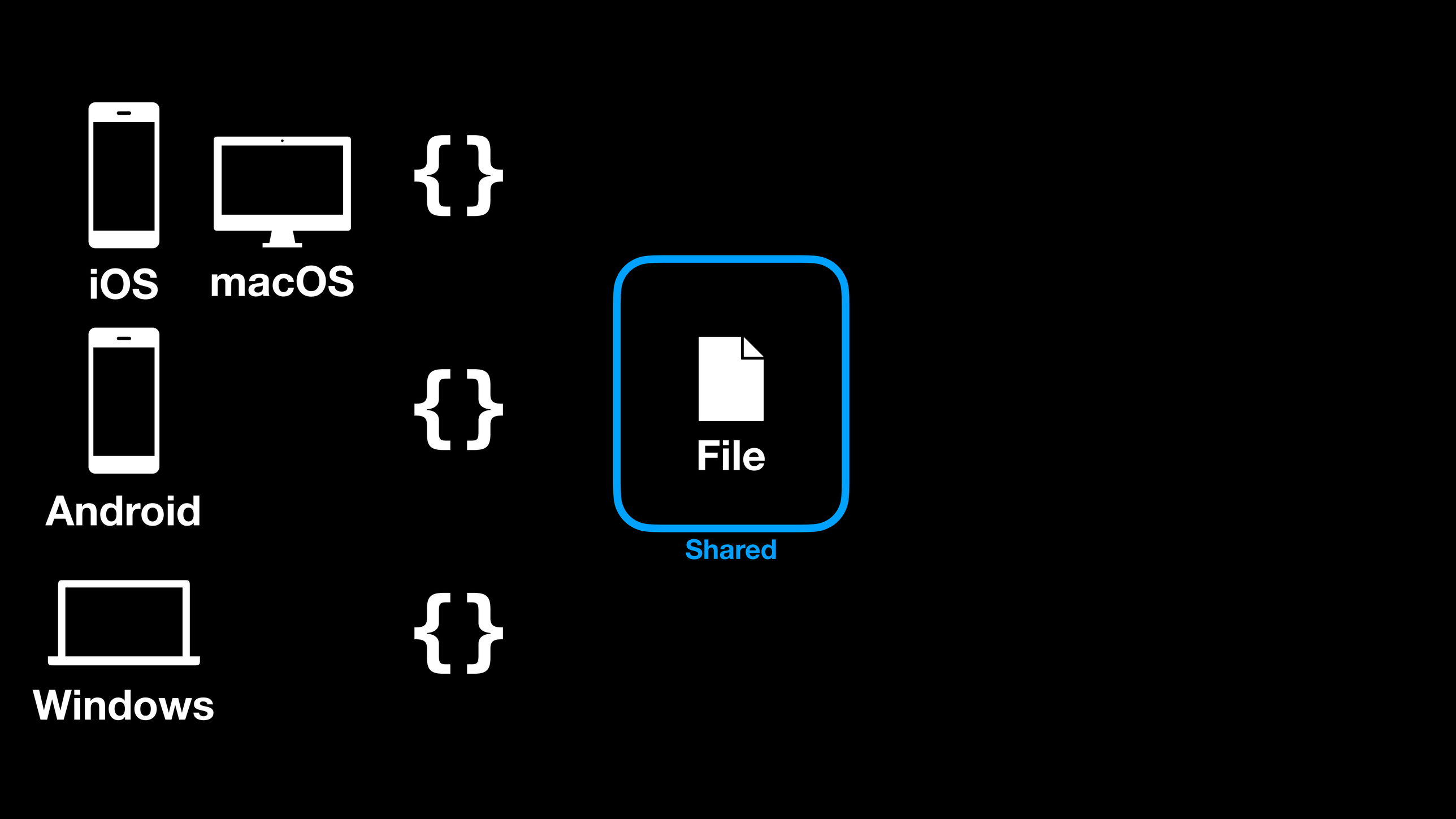


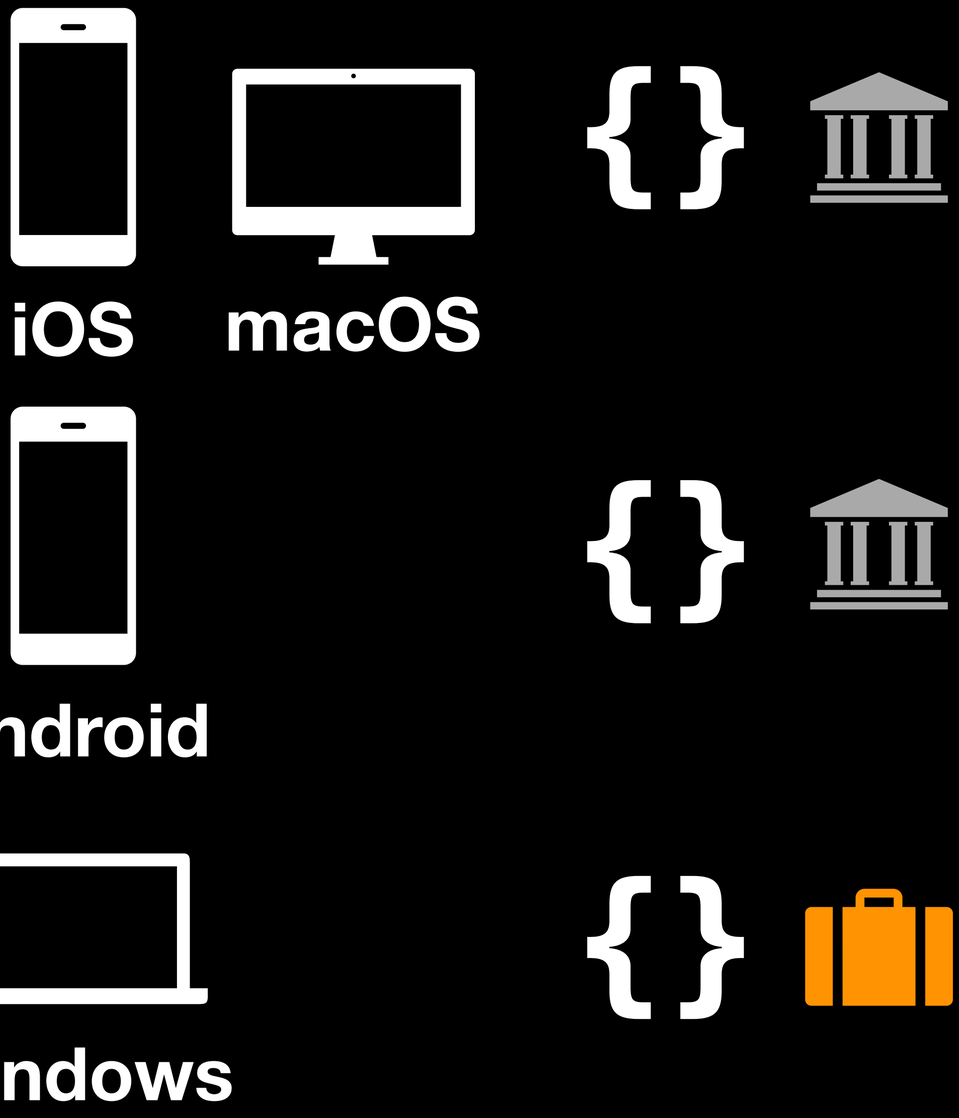
Android

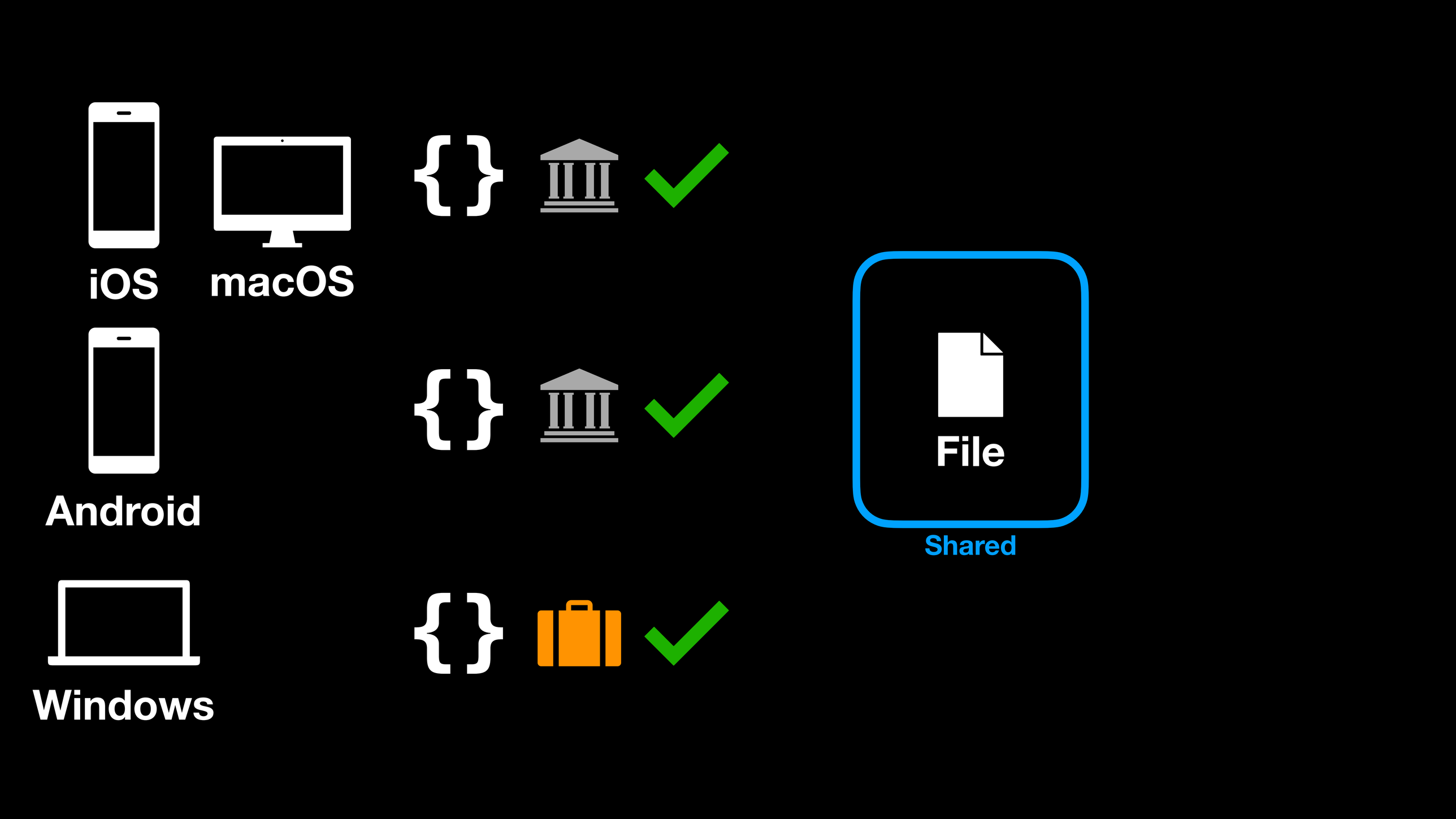


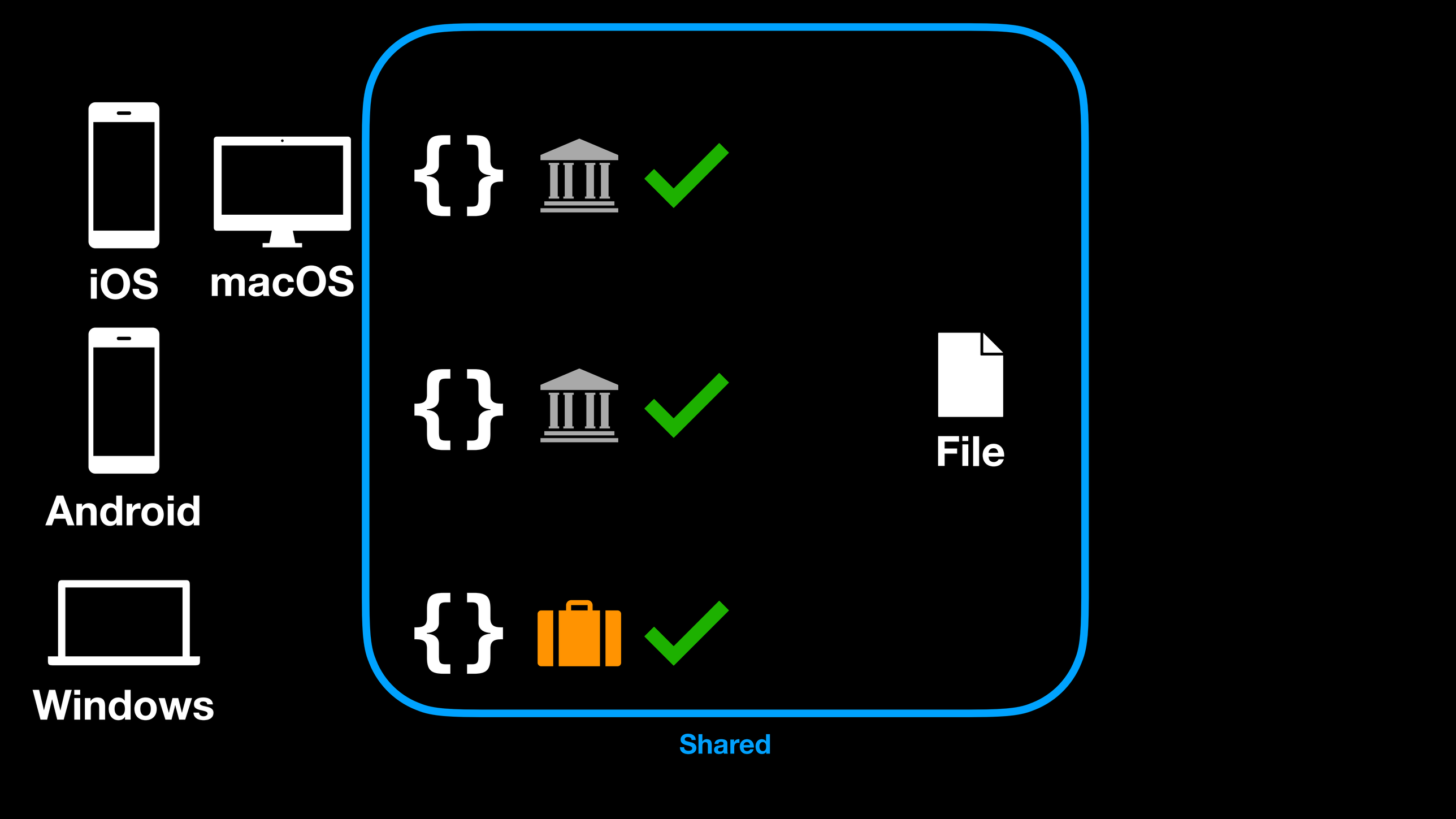


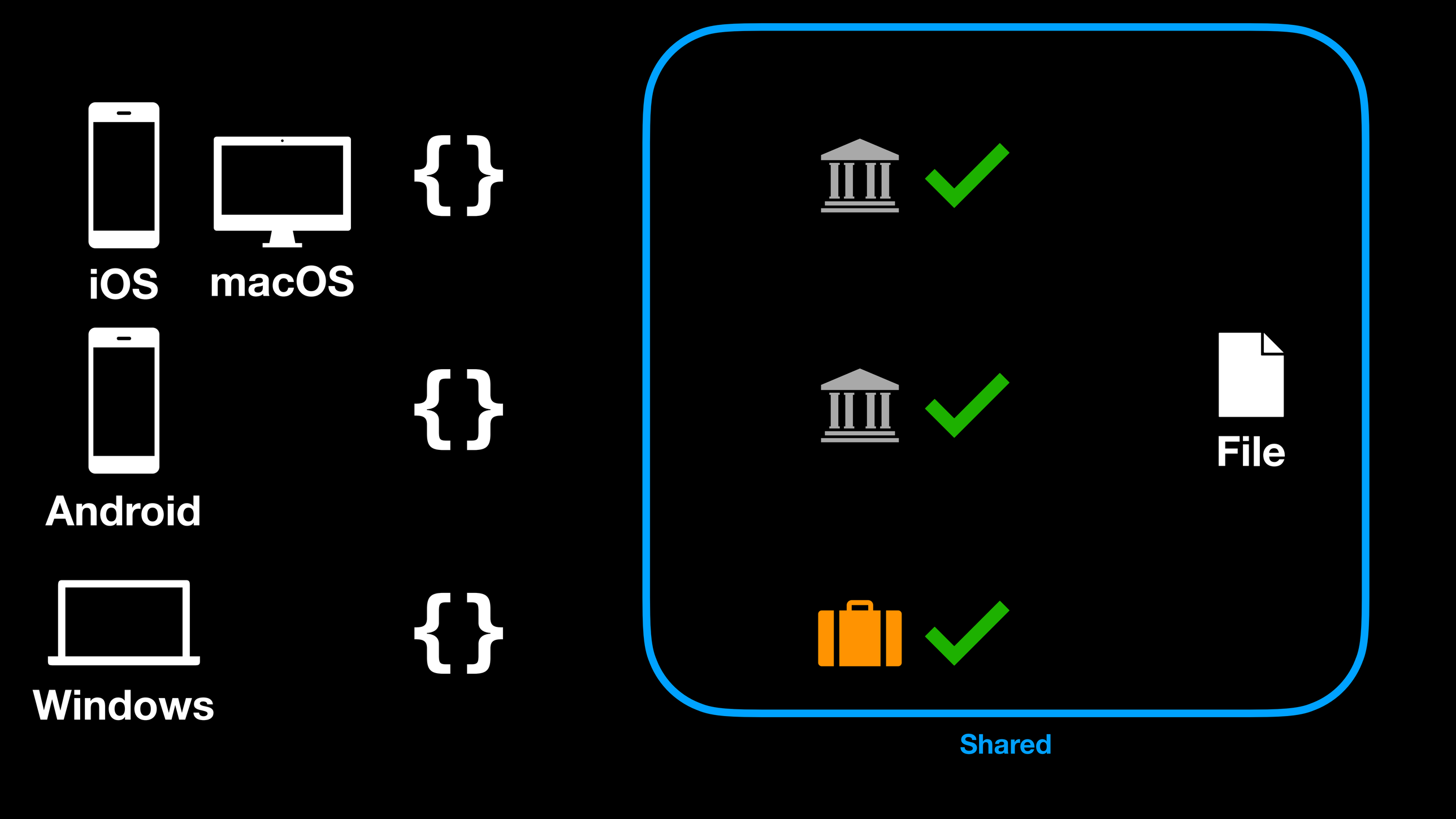

iOS
macOS
File
Shared
Android

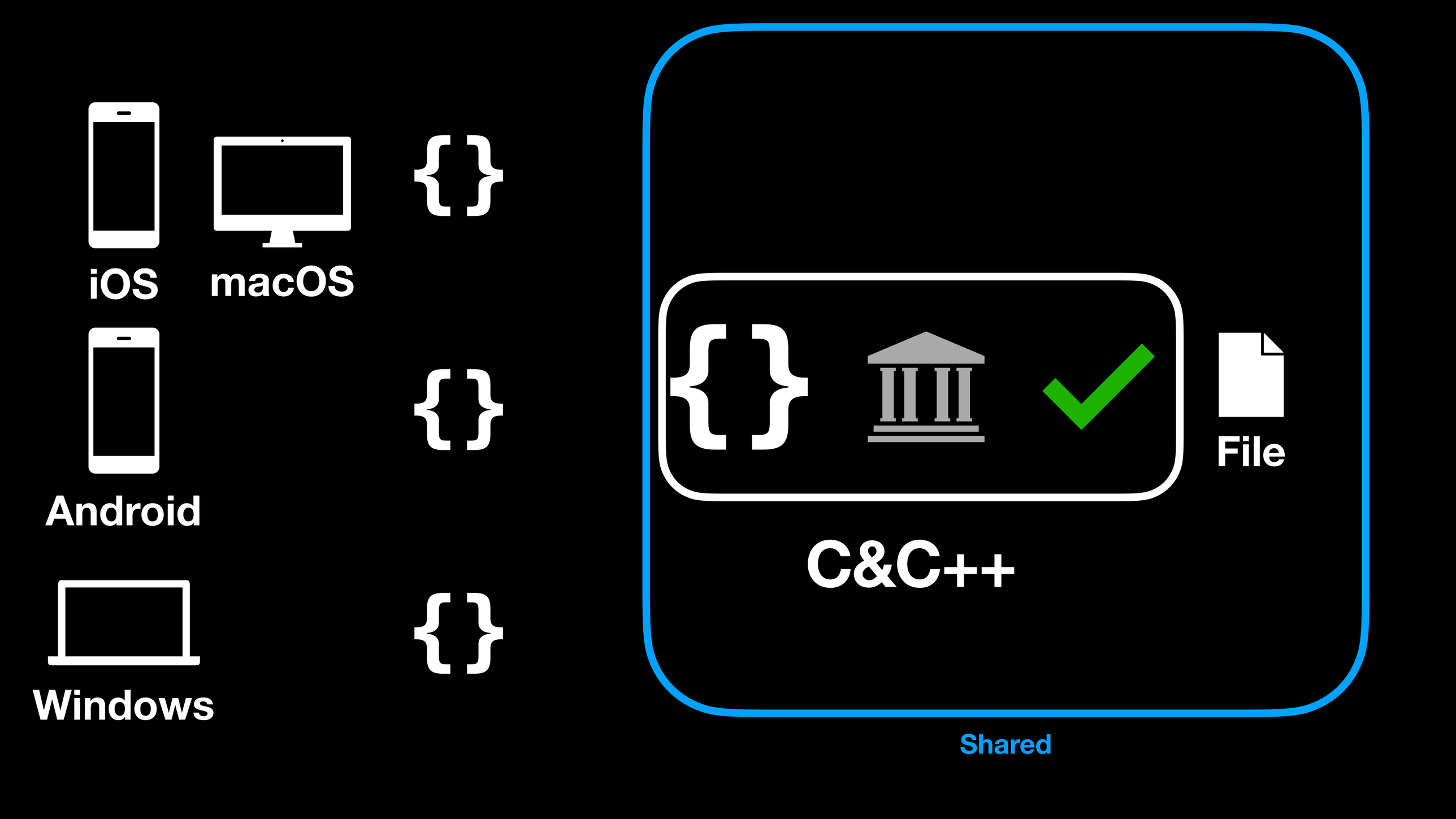

iOS
macOS
Android
Windows
File
Shared


iOS
macOS
Android
Windows
{}
{}
{}
File
Shared


iOS
macOS
Android
Windows
File
Shared


iOS
macOS
Android
Windows
File
Shared


iOS
macOS
Android
Windows
File
Shared


iOS
macOS
Android
Windows
File
C&C++
Shared


WHEN DO WE GET STARTED?

You’ve already used C in Swift

C-only Frameworks

Keychain

C-only Frameworks

```
var query = [String: Any]()
query[kSecAttrAccount as String] = key
query[kSecReturnData as String] = kCFBooleanTrue
query[kSecMatchLimit as String] = kSecMatchLimitOne
query[kSecReturnAttributes as String] = kCFBooleanTrue
query[kSecUseAuthenticationContext as String] = context
```

```
var queryResult: CFTypeRef?
let status = SecItemCopyMatching(query as CFDictionary, &queryResult)

guard [errSecItemNotFound, errSecSuccess]
    .contains(status) else { return nil }

if let result = queryResult as? [String: Any] {
    return result[kSecValueData as String] as? Data
}
```

Keychain

C-only Frameworks

```
var query = [String: Any]()
query[kSecAttrAccount as String] = key
query[kSecReturnData as String] = kCFBooleanTrue
query[kSecMatchLimit as String] = kSecMatchLimitOne
query[kSecReturnAttributes as String] = kCFBooleanTrue
query[kSecUseAuthenticationContext as String] = context


var queryResult: CFTypeRef?
let status = SecItemCopyMatching(query as CFDictionary, &queryResult)

guard [errSecItemNotFound, errSecSuccess]
    .contains(status) else { return nil }

if let result = queryResult as? [String: Any] {
    return result[kSecValueData as String] as? Data
}
```

Keychain

C-only Frameworks

```
var query = [String: Any]()
query[kSecAttrAccount as String] = key
query[kSecReturnData as String] = kCFBooleanTrue
query[kSecMatchLimit as String] = kSecMatchLimitOne
query[kSecReturnAttributes as String] = kCFBooleanTrue
query[kSecUseAuthenticationContext as String] = context


var queryResult: CFTypeRef?
let status = SecItemCopyMatching(query as CFDictionary, &queryResult)

guard [errSecItemNotFound, errSecSuccess]
    .contains(status) else { return nil }

if let result = queryResult as? [String: Any] {
    return result[kSecValueData as String] as? Data
}
```

Endpoint Security

C-only Frameworks

```
// Create the client.
es_client_t *client = NULL;
es_new_client_result_t newClientResult =
es_new_client(&client,
              ^(es_client_t * client, const es_message_t * message) {
    switch (message->event_type) {
        case ES_EVENT_TYPE_AUTH_EXEC:
            es_respond_auth_result(client, message, ES_AUTH_RESULT_ALLOW, true);
            break;
        default:
            panic("Found unexpected event type: %i", message->event_type);
            break;
    }
});
```

Endpoint Security

C-only Frameworks

```
// Create the client.
es_client_t *client = NULL;
es_new_client_result_t newClientResult =
es_new_client(&client,
              ^(es_client_t * client, const es_message_t * message) {
    switch (message->event_type) {
        case ES_EVENT_TYPE_AUTH_EXEC:
            es_respond_auth_result(client, message, ES_AUTH_RESULT_ALLOW, true);
            break;
        default:
            panic("Found unexpected event type: %i", message->event_type);
            break;
    }
});
```

C-only Frameworks

- Core Audio
- Core Midi
- Core Text
- Search Kit
- Accelerate
- libDispatch

C-only Frameworks

Work with Swift seamlessly

Multiple ways of working

C Interop

Objective-C++

ObjC

C++ Interop

The most stable

Outline

Outline

1. Dependencies
2. Compile Sources
3. Linking

Outline

1. Dependencies
2. Compile Sources
3. Linking

Outline

1. Dependencies

XCFramework

2. Compile Sources

3. Linking

Outline

1. Dependencies

XCFramework

2. Compile Sources

3. Linking

Outline

1. Dependencies

XCFramework

2. Compile Sources

3. Linking

Outline

1. Dependencies

XCFramework

2. Compile Sources

3. Linking

Demo

json Public

Sponsor | Watch 755 | Fork 6.3k | Starred 36.8k

develop | 9 branches | 55 tags | Go to file | Add file | Code

scribam Fix MinGW CI (#4175) edffad0 yesterday 4,554 commits

.github	Fix MinGW CI (#4175)	yesterday
.reuse	Use official Clang/GCC containers (#3703)	last year
LICENSES	Use REUSE framework (#3546)	last year
cmake	Fix CI + new Doctest (#3985)	5 months ago
docs	Fix source highlighting in user defined type macros docs (#4169)	4 days ago
include/nlohmann	Fix spellcheck issue (#4173)	2 days ago
single_include/nlohmann	Fix spellcheck issue (#4173)	2 days ago
tests	Fix spellcheck issue (#4173)	2 days ago
tools	Correct a typo in serve_header/README.md (#4143)	last month
.cirrus.yml	Prevent memory leak when exception is thrown in adl_serializer::to_j...	7 months ago
.clang-format	add Clang-Tidy configuration	3 years ago
.clang-tidy	Fix CI (#4160)	2 weeks ago
.gitignore	More documentation updates for 3.11.0 (#3553)	last year
.lgtm.yml	Miscellaneous small fixes (#3643)	last year

About

JSON for Modern C++

json.nlohmann.me

json | json-serialization | msgpack | cbor | json-parser | header-only | messagepack | json-pointer | json-patch | stl-containers | rfc-6901 | rfc-6902 | rfc-7159 | rfc-7049 | json-diff | bson | ubjson | json-merge-patch | rfc-8259

Readme
MIT license
Code of conduct
Security policy
Cite this repository
Activity
36.8k stars
755 watching
6.3k forks
Report repository

Demo
Json


json / single_include / nlohmann /
mwestphal Fix spellcheck issue (#4173)
Name
Last commit message
..
json.hpp
Fix spellcheck issue (#4173)
json_fwd.hpp
Fix CI (#4160)

Package.swift: JSON.hpp

```
// swift-tools-version:5.4
// The swift-tools-version declares the minimum version of Swift required to build this packa

import PackageDescription

let package = Package(
    name: "json",
    platforms: [
        .iOS(.v9), .macOS(.v10_10), .tvOS(.v9), .watchOS(.v2)
    ],
    products: [
        .library(
            name: "json",
            targets: ["json"]),
    ],
    targets: [
        .target(
            name: "json",
            dependencies: [],
            path: ".",
            exclude:
                [
                    "appveyor.yml",
                    "benchmarks",
                    "cmake",
                    "ChangeLog.md",
                    "doc",
```

```
// swift-tools-version:5.4
// The swift-tools-version declares the minimum version of Swift required to build this packa

import PackageDescription

let package = Package(
    name: "json",
    platforms: [
        .iOS(.v9), .macOS(.v10_10), .tvOS(.v9), .watchOS(.v2)
    ],
    products: [
        .library(
            name: "json",
            targets: ["json"]),
    ],
    targets: [
        .target(
            name: "json",
            dependencies: [],
            path: ".",
            exclude:
                [
                    "appveyor.yml",
                    "benchmarks",
                    "cmake",
                    "ChangeLog.md",
                    "doc",
```

```
        dependencies: [],
        path: ".",
        exclude:
            [
                "appveyor.yml",
                "benchmarks",
                "cmake",
                "ChangeLog.md",
                "doc",
                "include",
                "test",
                "third_party",
                "CMakeLists.txt",
                "CODE_OF_CONDUCT.md",
                "LICENSE.MIT",
                "Makefile",
                "meson.build",
                "nlohmann_json.natvis",
                "README.md",
                "wsjcpp.yml",
            ],
        sources:
            [

            ],
        publicHeadersPath: "./single_include/")
],
```

```
        dependencies: [],
        path: ".",
        exclude:
            [
                "appveyor.yml",
                "benchmarks",
                "cmake",
                "ChangeLog.md",
                "doc",
                "include",
                "test",
                "third_party",
                "CMakeLists.txt",
                "CODE_OF_CONDUCT.md",
                "LICENSE.MIT",
                "Makefile",
                "meson.build",
                "nlohmann_json.natvis",
                "README.md",
                "wsjcpp.yml",
            ],
        sources:
            [

            ],
        publicHeadersPath: "./single_include/")
],
```

```
        dependencies: [],
        path: ".",
        exclude:
            [
                "appveyor.yml",
                "benchmarks",
                "cmake",
                "ChangeLog.md",
                "doc",
                "include",
                "test",
                "third_party",
                "CMakeLists.txt",
                "CODE_OF_CONDUCT.md",
                "LICENSE.MIT",
                "Makefile",
                "meson.build",
                "nlohmann_json.natvis",
                "README.md",
                "wsjcpp.yml",
            ],
        sources:
            [

            ],
        publicHeadersPath: "./single_include/")
],
```

```
            [
                "appveyor.yml",
                "benchmarks",
                "cmake",
                "ChangeLog.md",
                "doc",
                "include",
                "test",
                "third_party",
                "CMakeLists.txt",
                "CODE_OF_CONDUCT.md",
                "LICENSE.MIT",
                "Makefile",
                "meson.build",
                "nlohmann_json.natvis",
                "README.md",
                "wsjcpp.yml",
            ],
        sources:
            [

            ],
        publicHeadersPath: "./single_include/")
    ],
    cxxLanguageStandard: .cxx11
)
```

Package.swift: JSON.hpp: fix

```
            [
                "appveyor.yml",
                "benchmarks",
                "cmake",
                "ChangeLog.md",
                "doc",
                "include",
                "test",
                "third_party",
                "CMakeLists.txt",
                "CODE_OF_CONDUCT.md",
                "LICENSE.MIT",
                "Makefile",
                "meson.build",
                "nlohmann_json.natvis",
                "README.md",
                "wsjcpp.yml",
            ],
            sources:
                [
                    "_SwiftPackageManagerFile.cpp"
                ],
            publicHeadersPath: "./single_include/")
    ],

    cxxLanguageStandard: .cxx11
)
```

```
            ],
            sources:
                [
                    "_SwiftPackageManagerFile.cpp"
                ],
            publicHeadersPath: "./single_include/")
    ],

    cxxLanguageStandard: .cxx11
)

// This file exists only to trigger the json module to build; without
// it swiftpm won't build anything, and then the package is not available for
// the modules that need it.

#include <nlohmann/json.hpp>
```

```
            ],
            sources:
                [
                    "_SwiftPackageManagerFile.cpp"
                ],
            publicHeadersPath: "./single_include/")
    ],

    cxxLanguageStandard: .cxx11
)

// This file exists only to trigger the json module to build; without
// it swiftpm won't build anything, and then the package is not available for
// the modules that need it.

#include <nlohmann/json.hpp>
```

Files

main

Go to file t

Sources
ComplexModule
IntegerUtilities
Numerics
RealModule
_NumericsShims
include
CMakeLists.txt
README.md
_NumericsShims.c
_TestSupport
CMakeLists.txt
Tests
cmake
.gitignore

swift-numerics / Sources / _NumericsShims / _NumericsShims.c

stephentyrone Back to trailing 'Module'

Code Blame 19 lines (16 loc) · 791 Bytes

```
1   //===--- NumericsShims.c ---------------------------------*- swift -*-===//
2   //
3   // This source file is part of the Swift Numerics open source project
4   //
5   // Copyright (c) 2019 Apple Inc. and the Swift Numerics project authors
6   // Licensed under Apache License v2.0 with Runtime Library Exception
7   //
8   // See https://swift.org/LICENSE.txt for license information
9   //
10  //===----------------------------------------------------------------------===//
11
12  // This file exists only to trigger the NumericShims module to build; without
13  // it swiftpm won't build anything, and then the shims are not available for
14  // the modules that need them.
15
16  // If any shims are added that are not pure header inlines, whatever runtime
17  // support they require can be added to this file.
18
19  #include "_NumericsShims.h"
```

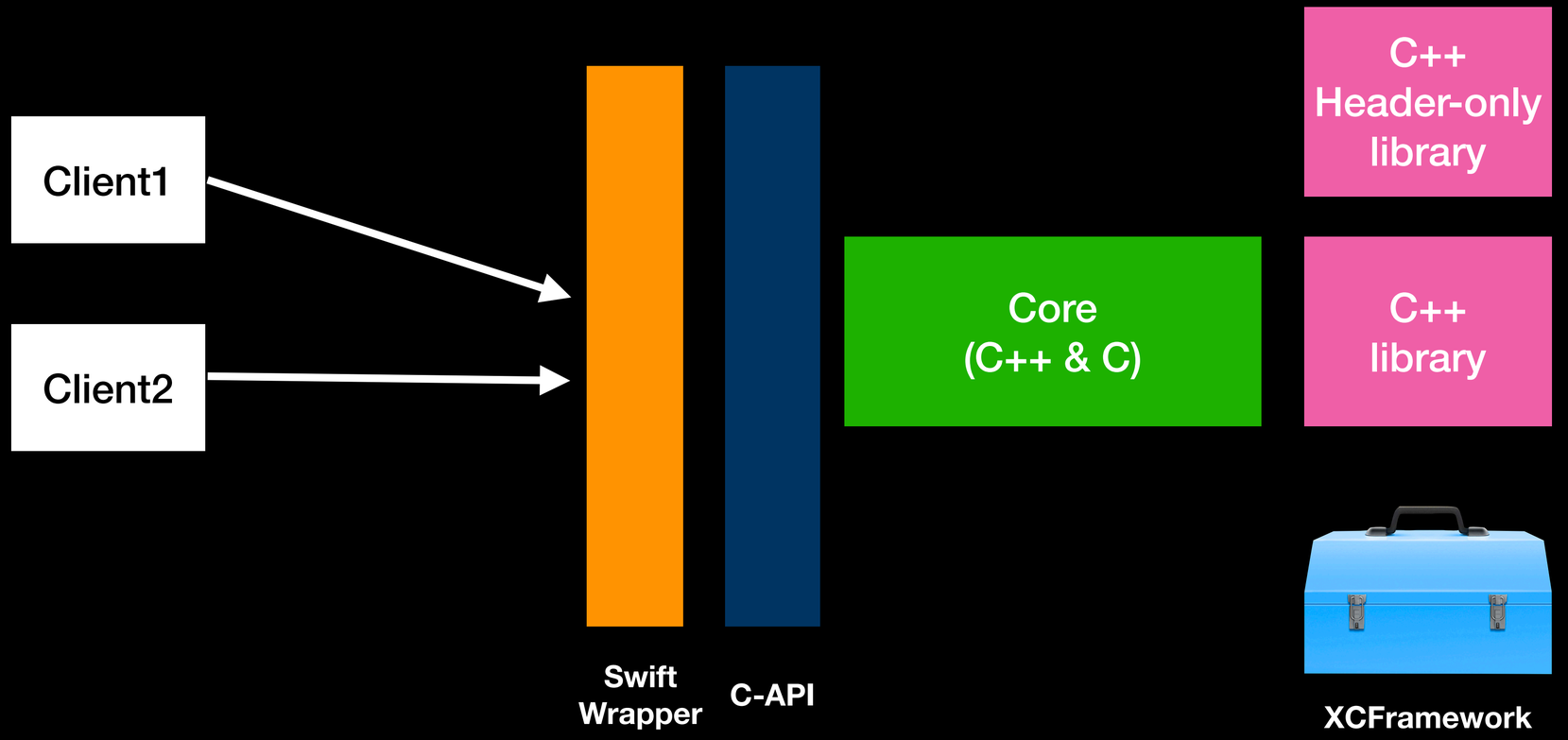
Real-World Example

Real-World Example

- A C++ library (Cross-platform)
- …with C interface
- Tested with GoogleTest
- Uses pre-built frameworks
- Outside looks like just a Swift library

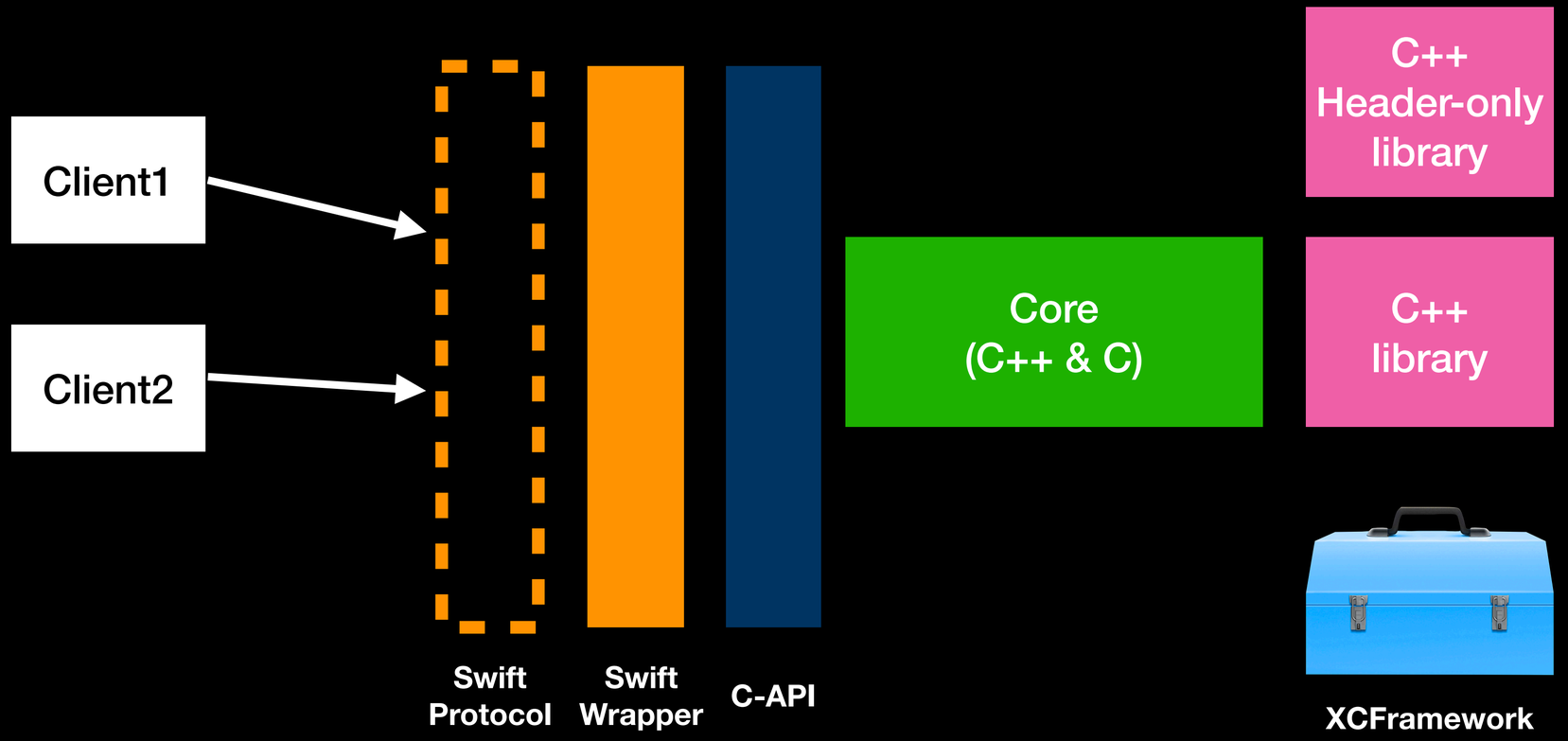
Real-World Example

Architecture



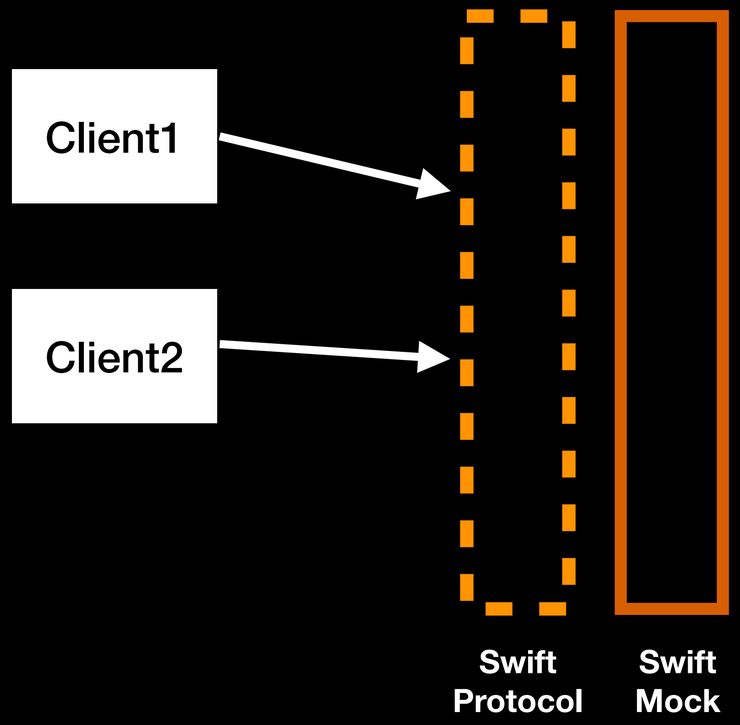
Real-World Example

Architecture



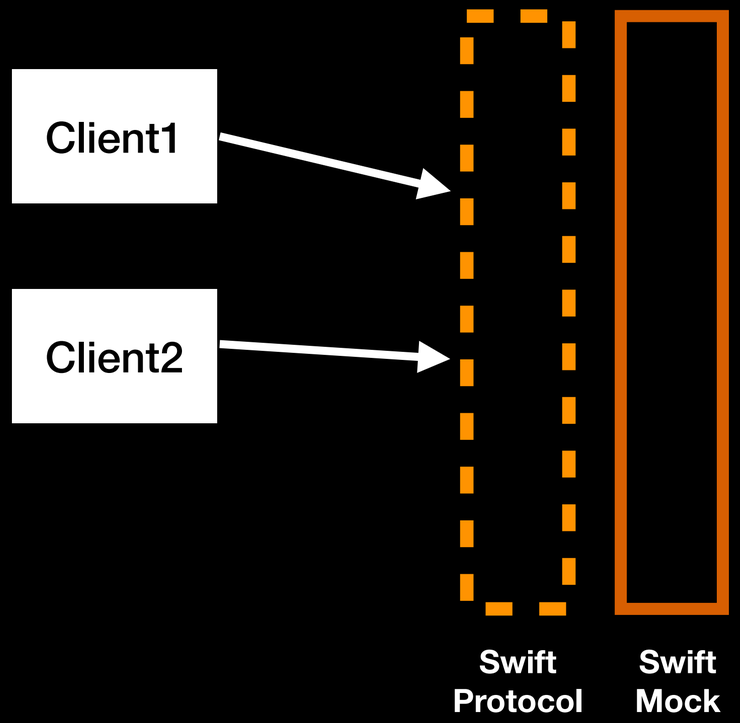
Real-World Example

Architecture



Real-World Example

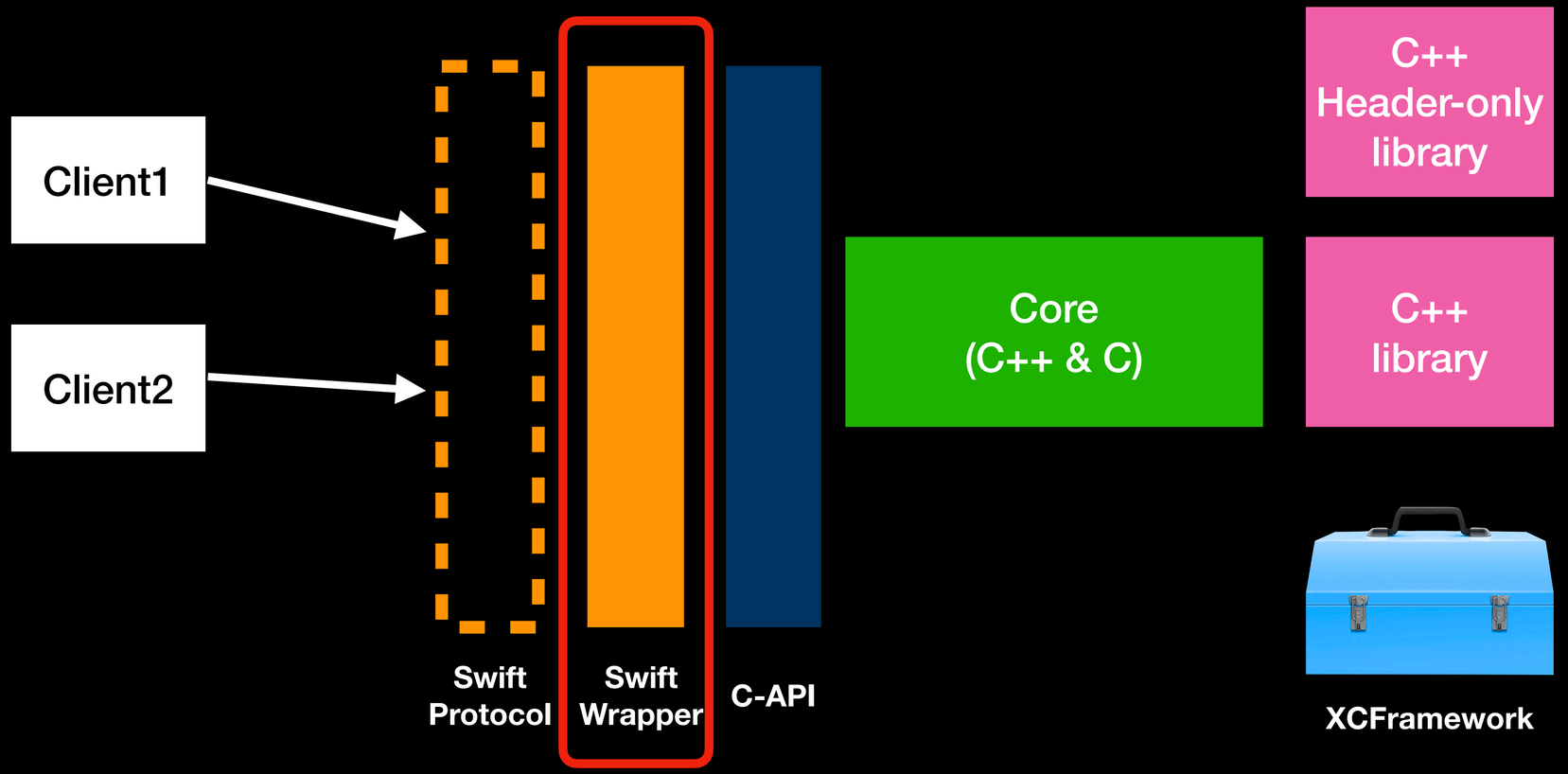
Architecture



No C++ → Faster build times

Real-World Example

Architecture



Swift Wrapper

Hides C/C++ Complexity

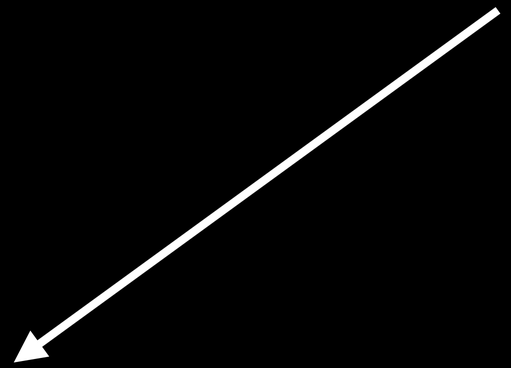
```swift
public func save(_ value: String) throws -> String {
    var returnValue: UnsafePointer<CChar>?
    defer {
        FreeString(returnValue)
    }
    let result = withUnsafeMutablePointer(to: &returnValue) {
        Core_Save(handle, value, $0)
    }
    if result != 0 {
        throw NSError(domain: "Core", code: -1)
    }

    return String(cString: returnValue)
}
```

C-API Call

Swift Wrapper

Hides C/C++ Complexity

Allows writing to pointer



```swift
public func save(_ value: String) throws -> String {
    var returnValue: UnsafePointer<CChar>?
    defer {
        FreeString(returnValue)
    }
    let result = withUnsafeMutablePointer(to: &returnValue) {
        Core_Save(handle, value, $0)
    }
    if result != 0 {
        throw NSError(domain: "Core", code: -1)
    }

    return String(cString: returnValue)
}
```

C-API Call

Swift Wrapper

Hides C/C++ Complexity

Memory Management

```
public func save(_ value: String) throws -> String {
    var returnValue: UnsafePointer<CChar>?
    defer {
        FreeString(returnValue)
    }
    let result = withUnsafeMutablePointer(to: &returnValue) {
        Core_Save(handle, value, $0)
    }
    if result != 0 {
        throw NSError(domain: "Core", code: -1)
    }

    return String(cString: returnValue)
}
```

C-API Call

Combine / Asynchronous code

- C Code can call the Swift code back (function pointer)
- Function pointers are imported as a closures with **@convention(c)**

Combine / Asynchronous code

```
typedef void(CALLBACK_TYPE *Callback)(
    void *context,
    const char *outputValue,
    ErrorCode result);

ErrorCode AsyncOperation(CoreHandle handle, Callback cb, void *context, const char
*inputValue);
```

Combine / Asynchronous code

```
typedef void(CALLBACK_TYPE *Callback)(
    void *context,
    const char *outputValue,
    ErrorCode result);

ErrorCode AsyncOperation(CoreHandle handle, Callback cb, void *context, const char
*inputValue);
```

Code

Combine / Asynchronous code

```
typedef void(CALLBACK_TYPE *Callback)(
    void *context,
    const char *outputValue,
    ErrorCode result);

ErrorCode AsyncOperation(CoreHandle handle, Callback cb, void *context, const char
*inputValue);
```

Data

Combine / Asynchronous code

```
typedef void(CALLBACK_TYPE *Callback)(
    void *context,
    const char *outputValue,
    ErrorCode result);

ErrorCode AsyncOperation(CoreHandle handle, Callback cb, void *context, const char
*inputValue);
```

Combine / Asynchronous code

Helper tool

```
private final class PromiseWrapper<Output, Failure: Error> {
    let promise: Future<Output, Failure>.Promise

    init(_ value: @escaping Future<Output, Failure>.Promise) {
        self.promise = value
    }
}
```

Just stores the promise

```
public func asyncOperation(_ inputValue: String) -> Future<Void, Error> {
    let handle = self.handle
    return Future { promise in
        typealias CCallback = @convention(c)(UnsafeMutableRawPointer?, CoreError) -> Void
        let callback: CCallback = {context, result in
            if let context {
                let promise = Unmanaged<PromiseWrapper<Void, Error>>
                    .fromOpaque(context)
                    .takeRetainedValue()
                    .promise
                promise(.success(()))
            }
        }

        let wrapper = PromiseWrapper(promise)
        let context = Unmanaged
            .passRetained(wrapper)
            .toOpaque()
        Core_AsyncOperation(handle,
                            callback,
                            context,
                            inputValue)
    }
}
```

1. Function invocation

```
public func asyncOperation(_ inputValue: String) -> Future<Void, Error> {
    let handle = self.handle
    return Future { promise in
        typealias CCallback = @convention(c)(UnsafeMutableRawPointer?, CoreError) -> Void
        let callback: CCallback = {context, result in
            if let context {
                let promise = Unmanaged<PromiseWrapper<Void, Error>>
                    .fromOpaque(context)
                    .takeRetainedValue()
                    .promise
                promise(.success(()))
            }
        }

        let wrapper = PromiseWrapper(promise)
        let context = Unmanaged
            .passRetained(wrapper)
            .toOpaque()
        Core_AsyncOperation(handle,
                            callback,
                            context,
                            inputValue)
    }
}
```

1. Function invocation

```
public func asyncOperation(_ inputValue: String) -> Future<Void, Error> {
    let handle = self.handle
    return Future { promise in
        typealias CCallback = @convention(c)(UnsafeMutableRawPointer?, CoreError) -> Void
        let callback: CCallback = {context, result in
            if let context {
                let promise = Unmanaged<PromiseWrapper<Void, Error>>
                    .fromOpaque(context)
                    .takeRetainedValue()
                    .promise
                promise(.success(()))
            }
        }

        let wrapper = PromiseWrapper(promise)
        let context = Unmanaged
            .passRetained(wrapper)
            .toOpaque()
        Core_AsyncOperation(handle,
                            callback,
                            context,
                            inputValue)
    }
}
```

2. Callback invocation

1. Function invocation

```
public func asyncOperation(_ inputValue: String) -> Future<Void, Error> {
    let handle = self.handle
    return Future { promise in
        typealias CCallback = @convention(c)(UnsafeMutableRawPointer?, CoreError) -> Void
        let callback: CCallback = {context, result in
            if let context {
                let promise = Unmanaged<PromiseWrapper<Void, Error>>
                    .fromOpaque(context)
                    .takeRetainedValue()
                    .promise
                promise(.success(()))
            }
        }

        let wrapper = PromiseWrapper(promise)
        let context = Unmanaged
            .passRetained(wrapper)
            .toOpaque()
        Core_AsyncOperation(handle,
                            callback,
                            context,
                            inputValue)
    }
}
```

```
public func asyncOperation(_ inputValue: String) -> Future<Void, Error> {
    let handle = self.handle
    return Future { promise in
        typealias CCallback = @convention(c)(UnsafeMutableRawPointer?, CoreError) -> Void
        let callback: CCallback = {context, result in
            if let context {
                let promise = Unmanaged<PromiseWrapper<Void, Error>>
                    .fromOpaque(context)
                    .takeRetainedValue()
                    .promise
                promise(.success(()))
            }
        }

        let wrapper = PromiseWrapper(promise)
        let context = Unmanaged
            .passRetained(wrapper)
            .toOpaque()
        Core_AsyncOperation(handle,
                            callback,
                            context,
                            inputValue)
    }
}
```

```
public func asyncOperation(_ inputValue: String) -> Future<Void, Error> {
    let handle = self.handle
    return Future { promise in
        typealias CCallback = @convention(c)(UnsafeMutableRawPointer?, CoreError) -> Void
        let callback: CCallback = {context, result in
            if let context {
                let promise = Unmanaged<PromiseWrapper<Void, Error>>
                    .fromOpaque(context)
                    .takeRetainedValue()
                    .promise
                promise(.success(()))
            }
        }

        let wrapper = PromiseWrapper(promise)
        let context = Unmanaged
            .passRetained(wrapper)
            .toOpaque()
        Core_AsyncOperation(handle,
                            callback,
                            context,
                            inputValue)
    }
}
```

```
public func asyncOperation(_ inputValue: String) -> Future<Void, Error> {
    let handle = self.handle
    return Future { promise in
        typealias CCallback = @convention(c)(UnsafeMutableRawPointer?, CoreError) -> Void
        let callback: CCallback = {context, result in
            if let context {
                let promise = Unmanaged<PromiseWrapper<Void, Error>>
                    .fromOpaque(context)
                    .takeRetainedValue()
                    .promise
                promise(.success(()))
            }
        }

        let wrapper = PromiseWrapper(promise)
        let context = Unmanaged
            .passRetained(wrapper)
            .toOpaque()
        Core_AsyncOperation(handle,
                            callback,
                            context,
                            inputValue)
    }
}
```

+1

```
public func asyncOperation(_ inputValue: String) -> Future<Void, Error> {
    let handle = self.handle
    return Future { promise in
        typealias CCallback = @convention(c)(UnsafeMutableRawPointer?, CoreError) -> Void
        let callback: CCallback = {context, result in
            if let context {
                let promise = Unmanaged<PromiseWrapper<Void, Error>>
                    .fromOpaque(context)
                    .takeRetainedValue()
                    .promise
                promise(.success(()))
            }
        }

        let wrapper = PromiseWrapper(promise)
        let context = Unmanaged
            .passRetained(wrapper)
            .toOpaque()
        Core_AsyncOperation(handle,
                            callback,
                            context,
                            inputValue)
    }
}
```

-1

+1

```
public func asyncOperation(_ inputValue: String) -> Future<Void, Error> {
    let handle = self.handle
    return Future { promise in
        typealias CCallback = @convention(c)(UnsafeMutableRawPointer?, CoreError) -> Void
        let callback: CCallback = {context, result in
            if let context {
                let promise = Unmanaged<PromiseWrapper<Void, Error>>
                    .fromOpaque(context)
                    .takeRetainedValue()
                    .promise
                promise(.success(()))
            }
        }

        let wrapper = PromiseWrapper(promise)
        let context = Unmanaged
            .passRetained(wrapper)
            .toOpaque()
        Core_AsyncOperation(handle,
                            callback,
                            context,
                            inputValue)
    }
}
```

```
public func asyncOperation(_ inputValue: String) -> Future<Void, Error> {
    let handle = self.handle
    return Future { promise in
        typealias CCallback = @convention(c)(UnsafeMutableRawPointer?, CoreError) -> Void
        let callback: CCallback = {context, result in
            if let context {
                let promise = Unmanaged<PromiseWrapper<Void, Error>>
                    .fromOpaque(context)
                    .takeRetainedValue()
                    .promise
                promise(.success(()))
            }
        }

        let wrapper = PromiseWrapper(promise)
        let context = Unmanaged
            .passRetained(wrapper)
            .toOpaque()
        Core_AsyncOperation(handle,
                            callback,
                            context,
                            inputValue)
    }
}
```

```
public func asyncOperation(_ inputValue: String) -> Future<Void, Error> {
    let handle = self.handle
    return Future { promise in
        typealias CCallback = @convention(c)(UnsafeMutableRawPointer?, CoreError) -> Void
        let callback: CCallback = {context, result in
            if let context {
                let promise = Unmanaged<PromiseWrapper<Void, Error>>
                    .fromOpaque(context)
                    .takeRetainedValue()
                    .promise
                promise(.success(()))
            }
        }

        let wrapper = PromiseWrapper(promise)
        let context = Unmanaged
            .passRetained(wrapper)
            .toOpaque()
        Core_AsyncOperation(handle,
                            callback,
                            context,
                            inputValue)
    }
}
```

```
public func asyncOperation(_ inputValue: String) -> Future<Void, Error> {
    let handle = self.handle
    return Future { promise in
        typealias CCallback = @convention(c)(UnsafeMutableRawPointer?, CoreError) -> Void
        let callback: CCallback = {context, result in
            if let context {
                let promise = Unmanaged<PromiseWrapper<Void, Error>>
                    .fromOpaque(context)
                    .takeRetainedValue()
                    .promise
                promise(.success(()))
            }
        }

        let wrapper = PromiseWrapper(promise)
        let context = Unmanaged
            .passRetained(wrapper)
            .toOpaque()
        Core_AsyncOperation(handle,
                            callback,
                            context,
                            inputValue)
    }
}
```

```
public func asyncOperation(_ inputValue: String) -> Future<Void, Error> {
    let handle = self.handle
    return Future { promise in
        typealias CCallback = @convention(c)(UnsafeMutableRawPointer?, CoreError) -> Void
        let callback: CCallback = {context, result in
            if let context {
                let promise = Unmanaged<PromiseWrapper<Void, Error>>
                    .fromOpaque(context)
                    .takeRetainedValue()
                    .promise
                promise(.success(()))
            }
        }

        let wrapper = PromiseWrapper(promise)
        let context = Unmanaged
            .passRetained(wrapper)
            .toOpaque()
        Core_AsyncOperation(handle,
                            callback,
                            context,
                            inputValue)
    }
}
```

C++ & Swift Interop

New in 2023

0:00 / 17:44

Overview Transcript Code

Mix Swift and C++

Learn how you can use Swift in your C++ and Objective-C++ projects to make your code safer, faster, and easier to develop. We'll show you how to use C++ and Swift APIs to incrementally incorporate Swift into your app.

WWDC 2023 session 10172

C++ Interop

- Available starting 2023
- Newest API
- Allows directly calling C++ from Swift and vice versa
- Not everything can be imported / called from Swift

C++ Interop

```
class FibonacciCalculatorCplusplus {
public:
    FibonacciCalculatorCplusplus(bool printInvocation);
    double fibonacci(double value) const;
private:
    bool printInvocation;
};
```

```
// Create the C++ `FibonacciCalculatorCplusplus` class and invoke its `fibonacci` method.
let cxxCalculator = FibonacciCalculatorCplusplus(printInvocation)
return cxxCalculator.fibonacci(value - 1.0) + cxxCalculator.fibonacci(value - 2.0)
```

C++ Interop

```
class FibonacciCalculatorCplusplus {
public:
    FibonacciCalculatorCplusplus(bool printInvocation);
    double fibonacci(double value) const;
private:
    bool printInvocation;
};
```

```
// Create the C++ `FibonacciCalculatorCplusplus` class and invoke its `fibonacci` method.
let cxxCalculator = FibonacciCalculatorCplusplus(printInvocation)
return cxxCalculator.fibonacci(value - 1.0) + cxxCalculator.fibonacci(value - 2.0)
```

References

- Documentation: Swift Unmanaged
- Documentation: Using Imported C Functions in Swift
- Documentation: Swift: C Interoperability

Q&A