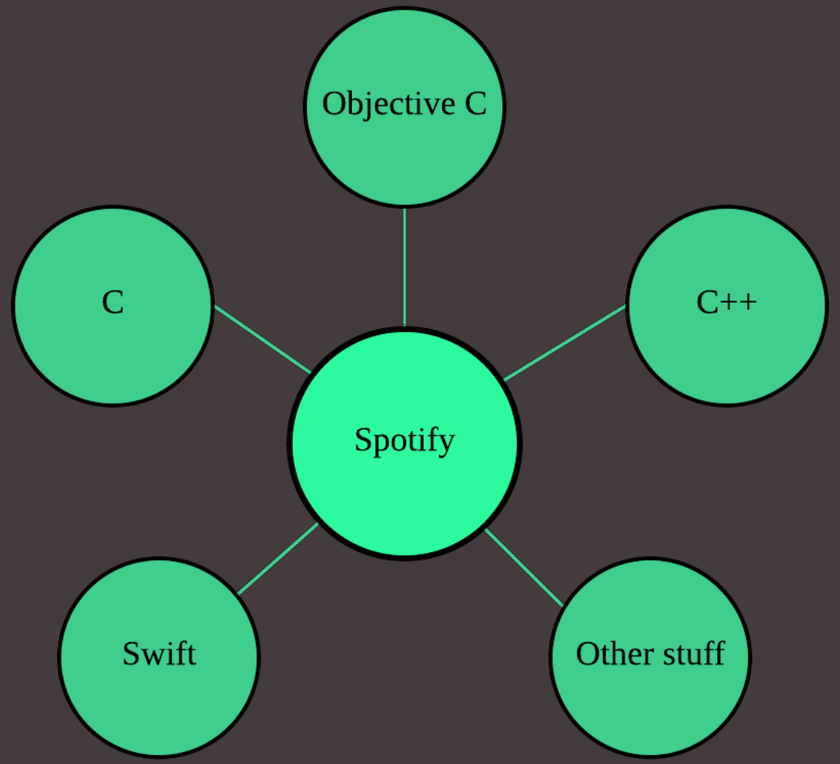
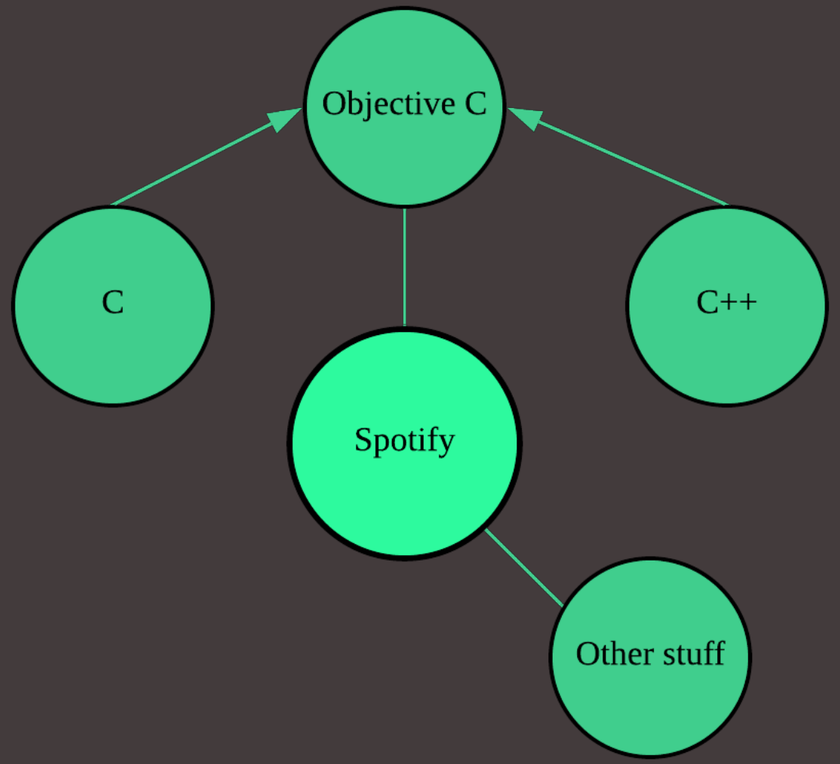
Import C, C++ and Objective-C libraries into Swift Projects

-- Cecilia Humlelu

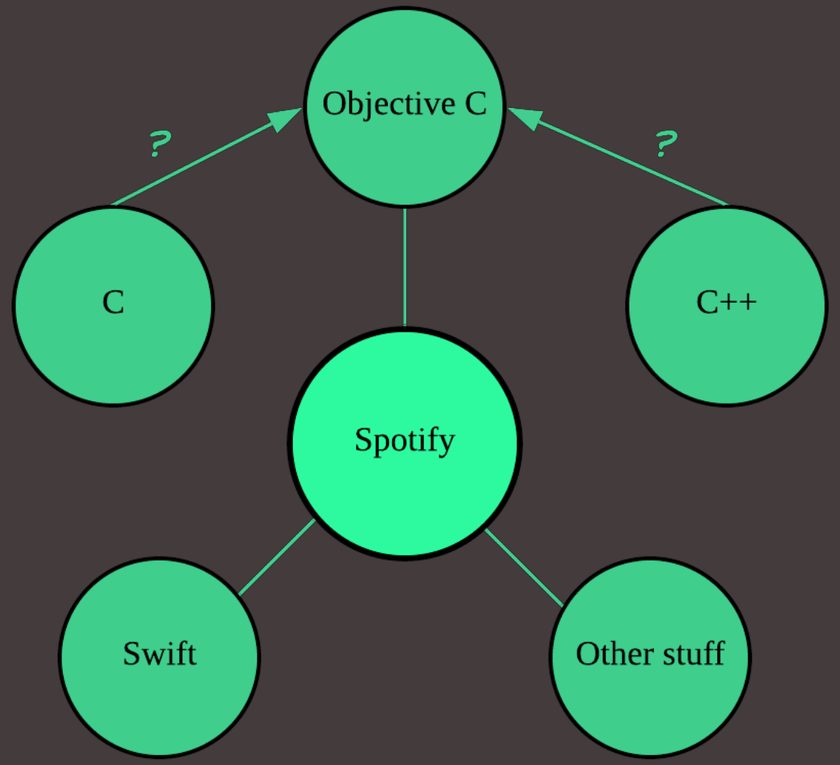
Background



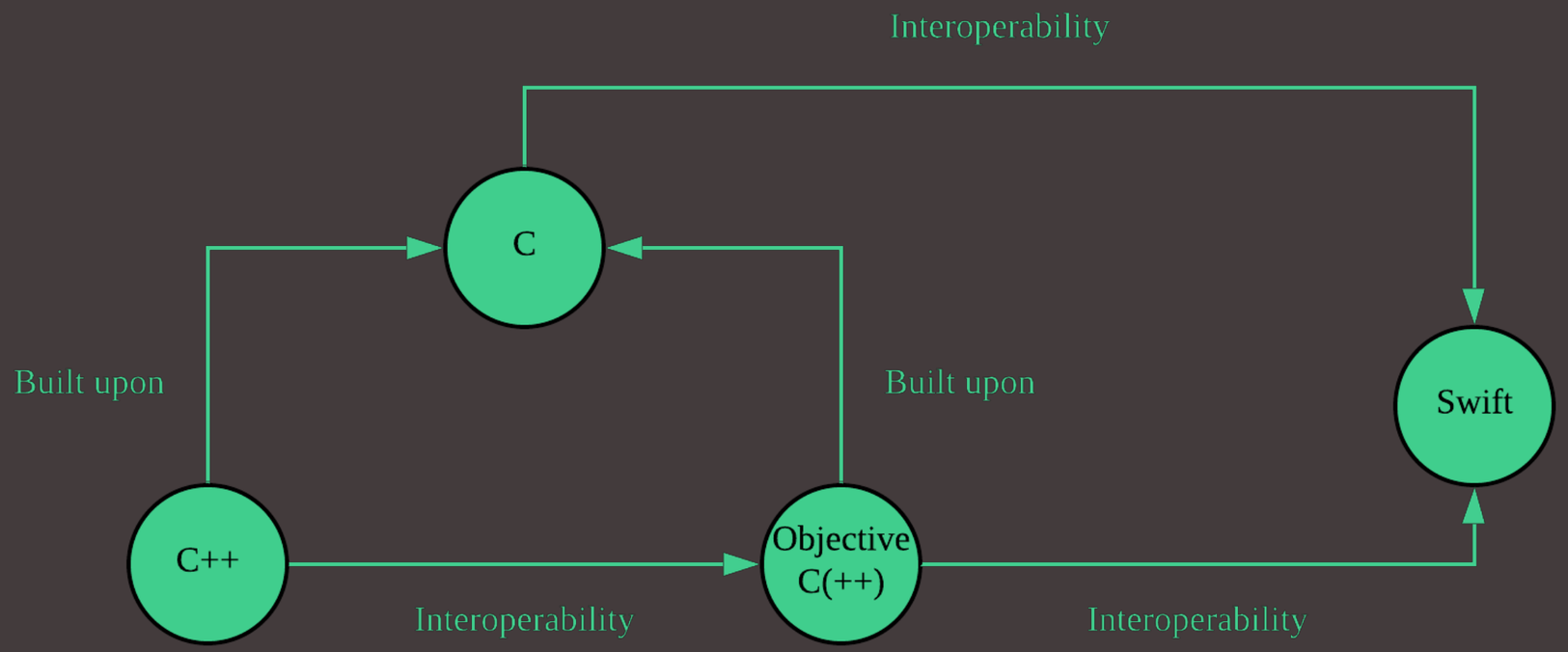
Background



Background



Connections



How to import C, C++ and Objective-C libraries into Swift applications

Import C, C++ and Objective-C libraries in Swift applications

- Dependencies
- Wrappers
- Recap

Objective-C

Drag VanillaObjC.xcodeproj in the swift application

General -> Linked Frameworks and Libraries-> VanillaObjC.framework

```
import VanillaObjC

let vanilla = VanillaObjC()
let result = vanilla.randomString(9)
```

SimpleCppWrapper Project -> Linked Frameworks and Libraries-> libBasicCppLib.a

SimpleCppWrapper.h
SimpleCppWrapper.mm

```
#import "SimpleCppWrapper.h"
#import "BasicCppLib/BasicCppLib.h"

@implementation SimpleCppWrapper

-(NSString *)randomStringWith:(NSUInteger)size {
    //return cpp.randomString();
}
```

Swift Project -> Linked Frameworks and Libraries-> libPureC.a

```
module CWrapper {
    header "../../PureC/PureC/PureC.h"
    export *
}

SWIFT_INCLUDE_PATHS = $(SRCROOT)/CWrapper/

import CWrapper

let result = randomString(10)
print(“print me \(result!)”)
```

Why Wrappers?

Objective-C

```
NS_SWIFT_NAME(OldWrapper)
@interface CHOldWrapper: NSObject
```

C Char *

```
char * randomString(int size);
```

```
public func randomString(_ size: Int32) -> UnsafeMutablePointer<Int8>!

public String.init(cString: UnsafePointer<UInt8>)

public func randomString(size: Int) -> String {
    let variable = CWrapper.randomString(Int32(size))!
    return String(cString: variable)
}
```

C memory management

```
public func randomString(size: Int) -> String {
    let variable = CWrapper.randomString(Int32(size))!
    defer {
        variable.deallocate()
    }
    return String(cString: variable)
}
```

C type casting

```
char * randomString(int size);
```

```
public func randomString(_ size: Int32) -> UnsafeMutablePointer<Int8>!
```

```
public func randomString(size: Int) -> String {
    let str = CWrapper.randomString(Int32(size))!
    ...
}
```

C function pointer

```
int (*getNextValue)(int val)
```

```
int foo(int startValue, int (*getNextValue)(int val));
```

```
public func foo(_ startValue: Int32,
        _ getNextValue: (@convention(c) (Int32) -> Int32)!) -> Int32
```

```
public func foo(startValue: Int,
                getNextValue: @escaping (Int)->Int) ->Int {
    return Int(CWrapper.foo(Int32(startValue), getNextValue))
}
```

```
Cannot convert value of type '(Int) -> Int'
to expected argument type '(@convention(c) (Int32) -> Int32)?'
```

C function pointer

```
public func foo(startValue: Int, getNextValue: @escaping (Int)->Int) ->Int {
    func localClosure(val: Int32) -> Int32 {
        return Int32(getNextValue(Int(val)))
    }
    return Int(CWrapper.foo(Int32(startValue), localClosure))
}
```

A C function pointer cannot be formed from a local function that captures context

C function pointer

```
fileprivate var globalClosure: ((Int)->Int)?

public func foo(startValue: Int,
                getNextValue: @escaping (Int)->Int) ->Int {
    globalClosure = getNextValue

    func localClosure(val: Int32) -> Int32 {
        return Int32(globalClosure!(Int(val)))
    }

    return Int(CWrapper.functionPointer(Int32(startValue), localClosure))
}
```

C++ string via Objective-C

```
public: std::string randomString(int size);
```

```
-(NSString *)randomString:(int)size {
    std::string randomStr = lib.randomString(size);
    const char * chars = randomStr.c_str();
    return [[NSString alloc] initWithCString:chars
                                    encoding:NSUTF8StringEncoding];
}
```

C++ class via Objective-C

```
class BasicCppLibrary {

private: int version;

public: BasicCppLibrary(int version);

};


@implementation SimpleCppWrapper {
    BasicCppLibrary *lib;}

-(instancetype)init {
    if (self = [super init]) {
        lib = new BasicCppLibrary(1);}
    return self;
}

-(void)dealloc { delete lib; }
```

Macros

```
#define GETMAX(x, y) (x>y ? x:y)

int maxInt(int x, int y) {
    return GETMAX(x, y);
}

float maxFloat(float x, float y) {
    return GETMAX(x, y);
}

double maxDouble(double x, double y) {
    return GETMAX(x, y);
}
```

Recap

- Objective-C - Not much effort
- C - Some effort
- C++ - Some effort

ありがとうございました

– github:tokyobirdy