Swift Reversing

Ryan Stortz

Infiltrate 2016

Presentation Overview

Part 1

Swift Introduction

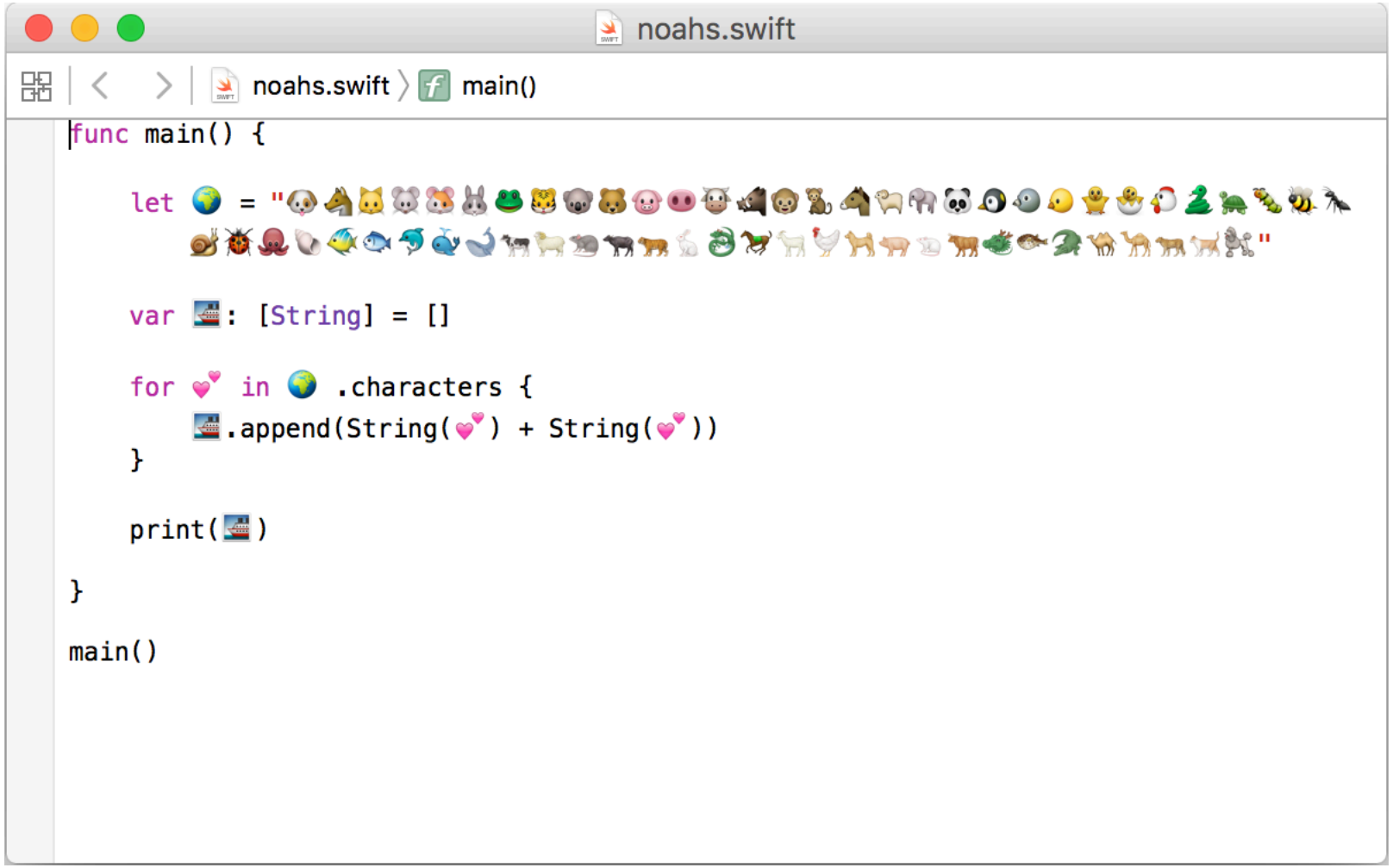
Part 2

Methodology

Part 3

Swift RE

Swift Introduction

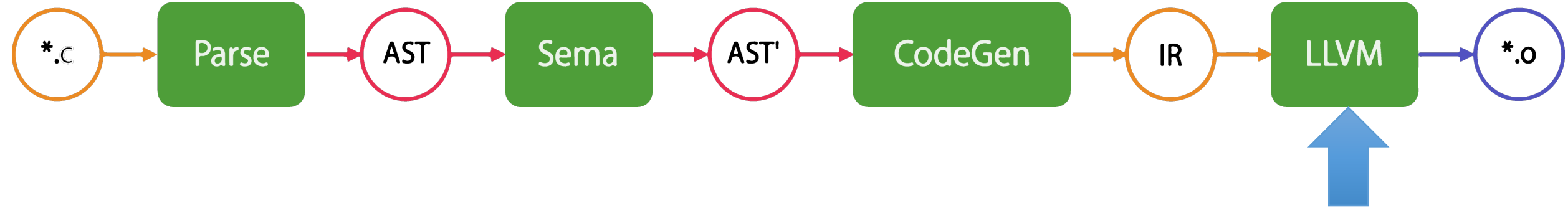

noahs.swift
noahs.swift
main()
func main() {
let 🌎 = "🐶🐴🐱🐭🐹🐰🐸🐯🐨🐻🐷🐽🐮🐗🐵🐒🐴🐑🐘🐼🐧🐦🐤🐥🐣🐔🐍🐢🐛🐝🐜
🐌🐞🐙🐚🐠🐟🐬🐳🐋🐄🐏🐀🐃🐅🐇🐉🐎🐐🐓🐕🐖🐁🐂🐲🐡🐊🐫🐪🐆🐈🐩"
var 🚢: [String] = []
for 💕 in 🌎 .characters {
🚢.append(String(💕) + String(💕))
}
print(🚢)
}
main()

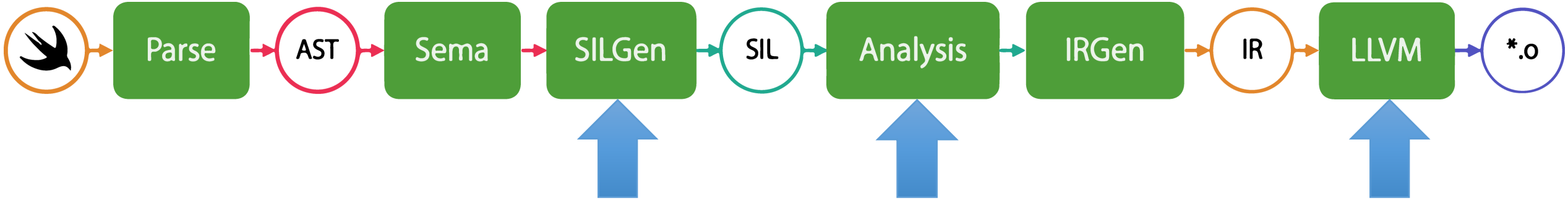

2. bash
ryan@retinabook:~/svn/swi/exa/noah/ $ ./noahs
["🐶", "🐴", "🐱", "🐭", "🐹", "🐰", "🐸", "🐯", "🐨", "🐻", "🐷", "🐽", "🐮", "🐗", "🐵", "🐒", "🐴", "🐑", "🐘", "🐼",
"🐧", "🐦", "🐤", "🐥", "🐣", "🐔", "🐍", "🐢", "🐛", "🐝", "🐜", "🐞", "🐌", "🐙", "🐚", "🐠", "🐟", "🐬", "🐳", "🐋",
"🐄", "🐏", "🐀", "🐃", "🐅", "🐇", "🐉", "🐎", "🐐", "🐓", "🐕", "🐖", "🐁", "🐂", "🐲", "🐡", "🐊", "🐫", "🐪", "🐆",
"🐈", "🐩"]
ryan@retinabook:~/svn/swi/exa/noah/ $

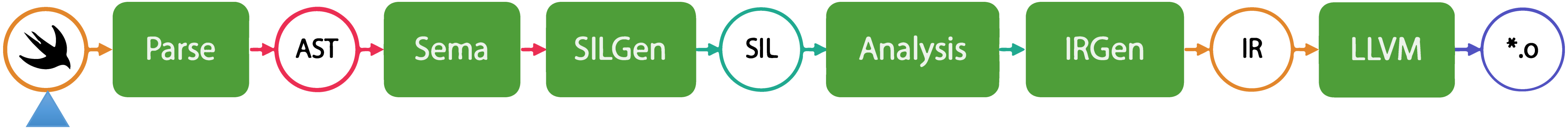
Swift Language

- Safe, fast, and expressive
- Closures and first-class functions
- Tuples and multiple return values
- Generics
- Fast and concise iteration over a range or collection
- Structs that support methods, extensions, and protocols
- Functional programming patterns, e.g., map and filter
- Powerful error handling built-in
- Advanced control flow with do, guard, defer, and repeat keywords

Compiler Architecture



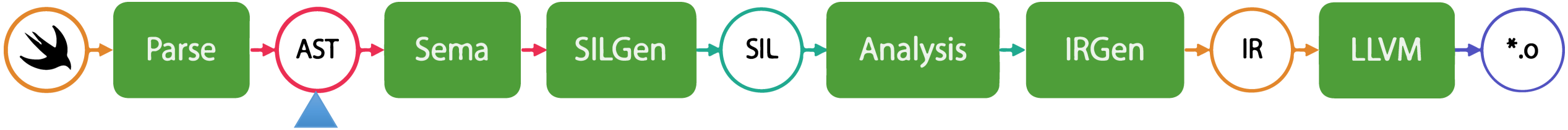




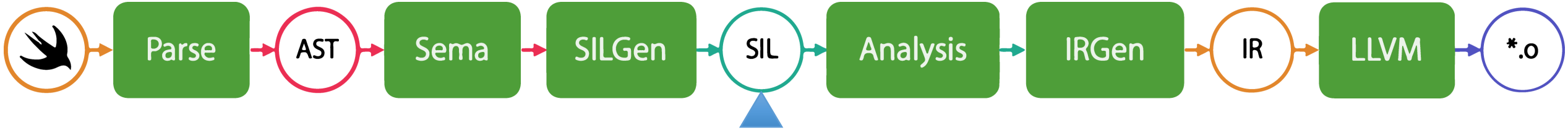
noahs.swift

noahs.swift 〉 main()

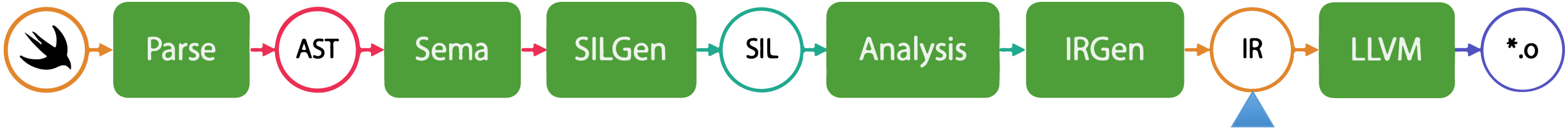
```swift
func main() {

    let 🌍 = "🐶🐺🐱🐭🐹🐰🐸🐯🐨🐻🐷🐽🐮🐗🐵🐒🐴🐑🐘🐼🐧🐦🐤🐥🐣🐔🐍🐢🐛🐝🐜
        🐌🐞🐙🐚🐠🐟🐬🐳🐋🐄🐏🐀🐃🐅🐇🐉🐎🐐🐓🐕🐖🐁🐂🐲🐡🐊🐫🐪🐆🐈🐩"

    var 🚢: [String] = []

    for 💕 in 🌍 .characters {
        🚢.append(String(💕) + String(💕))
    }

    print(🚢)

}

main()
```



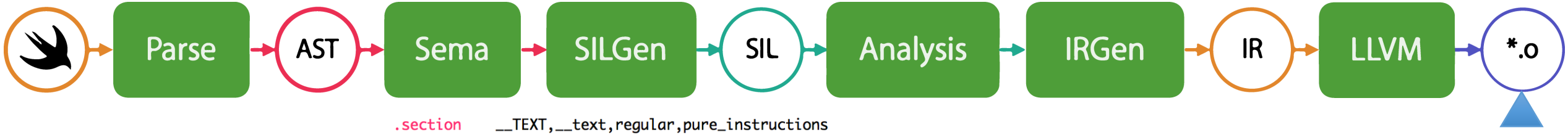

```
(source_file
  (func_decl "main()" type='() -> ()' access=internal
    (parameter_list)
    (brace_stmt
      (pattern_binding_decl
        (pattern_named type='String' '🌎)
        (call_expr implicit type='String' location=noahs.swift:3:13 range=[noahs.swift:3:13 - line:3:13] nothrow
          (constructor_ref_call_expr implicit type='(_builtinUTF16StringLiteral: RawPointer, numberOfCodeUnits: Word) -> String' location=noahs.swift:3:13 range=[noahs.swift:3:13 - line:3:13] nothrow
            (declref_expr implicit type='String.Type -> (_builtinUTF16StringLiteral: RawPointer, numberOfCodeUnits: Word) -> String' location=noahs.swift:3:13 range=[noahs.swift:3:13 - line:3:13] decl=Swift.(file).String.init(_builtinUTF16StringLiteral:numberOfCodeUnits:) specialized=no)
            (type_expr implicit type='String.Type' location=noahs.swift:3:13 range=[noahs.swift:3:13 - line:3:13] typerepr='String'))
          (string_literal_expr type='(_builtinUTF16StringLiteral: Builtin.RawPointer, numberOfCodeUnits: Builtin.Word)' location=noahs.swift:3:13 range=[noahs.swift:3:13 - line:3:13] encoding=utf16 value="🐶🐱🐭🐹🐰🦊🐻🐼🐻‍❄️🐨🐯🦁🐮🐷🐽🐸🐵🙈🙉🙊🐒🐔🐧🐦🐤🐣🐥🦆🦅🦉🦇🐺🐗🐴🦄🐝🐛🦋🐌🐞🐜🦟🦗🕷🕸🦂🐢🐍🦎🦖🦕🐙🦑🦐🦞🦀🐡🐠🐟🐬🐳🐋🦈🐊🐅🐆🦓🦍🦧🐘🦛🦏🐪🐫🦒🦘🐃🐂🐄🐎🐖🐏🐑🦙🐐🦌🐕🐩🦮🐕‍🦺🐈🐈‍⬛🐓🦃🦚🦜🦢🦩🕊🐇🦝🦨🦡🦫🦦🦥🐁🐀🐿🦔🐾🐉🐲🌵🎄🌲🌳🌴🌱🌿☘️🍀🎍🎋🍃🍂🍁🍄🐚🌾💐🌷🌹🥀🌺🌸🌼🌻🌞🌝🌛🌜🌚🌕🌖🌗🌘🌑🌒🌓🌔🌙🌎🌍🌏🪐💫⭐️🌟✨⚡️☄️💥🔥🌪🌈☀️🌤⛅️🌥☁️🌦🌧⛈🌩🌨❄️☃️⛄️🌬💨💧💦🫧☔️☂️🌊🌫"))))

      (var_decl "🌎 type='String' access=private let storage_kind=stored)

      (pattern_binding_decl
        (pattern_typed type='[String]'
          (pattern_named type='[String]' '🏁)
          (type_array
            (type_ident
              (component id='String' bind=Swift.(file).String))))
        (array_expr type='[String]' location=noahs.swift:5:23 range=[noahs.swift:5:23 - line:5:24]))
```




```
// noahs.main () -> ()
sil hidden @_TF5noahs4mainFT_T_ : $@convention(thin) () -> () {
bb0:
  %0 = alloc_stack $Array<String>, var, name "🇺🇸"
  %1 = string_literal utf16 "🐶🐱🐭🐹🐰🦊🐻🐼🐨🐯🦁🐮🐷🐸🐵🐔🐧🐦🐤🐣🐥🦆🦅🦉🦇🐺🐗🐴🦄🐝🐛🦋🐌🐞🐜🦗🕷🦂🐢🐍🦎🦖🦕🐙🦑🦐🦀🐡🐠🐟🐬🐳🐋🦈🐊🐅🐆🦓🦍🐘🦏🦛🐪🐫🦒🐃🐂🐄🐎🐖🐏🐑🐐🦌🐕🐩🐈🐓🦃🕊🐇🐁🐀🐿🦔🐾🐉🐲🌵🎄🌲🌳🌴🌱🌿☘🍀🎍🎋🍃🍂🍁🍄🌾💐🌷🌹🥀🌺🌸🌼🌻🌞🌝🌛🌜🌚🌕🌖🌗🌘🌑🌒🌓🌔🌙🌎🌍🌏🪐💫⭐🌟✨⚡☄💥🔥🌪🌈☀🌤⛅🌥☁🌦🌧⛈🌩🌨❄☃⛄🌬💨💧💦☔☂🌊🌫"
  debug_value %1 : $Builtin.RawPointer
  %3 = struct $COpaquePointer (%1 : $Builtin.RawPointer)
  %4 = enum $Optional<AnyObject>, #Optional.None!enumelt
  debug_value %3 : $COpaquePointer
  debug_value %4 : $Optional<AnyObject>
  %7 = integer_literal $Builtin.Int64, -9223372036854775684
  %8 = struct $UInt (%7 : $Builtin.Int64)
  %9 = struct $_StringCore (%3 : $COpaquePointer, %8 : $UInt, %4 : $Optional<AnyObject>)
  debug_value %9 : $_StringCore
  %11 = integer_literal $Builtin.Int64, 0
  %12 = struct $Int (%11 : $Builtin.Int64)
  debug_value %12 : $Int
  %14 = integer_literal $Builtin.Int1, 0
  debug_value %12 : $Int
  debug_value %12 : $Int
  %17 = integer_literal $Builtin.Int1, -1
  %18 = global_addr @_swiftEmptyArrayStorage : $*_SwiftEmptyArrayStorage
  %19 = address_to_pointer %18 : $*_SwiftEmptyArrayStorage to $Builtin.RawPointer
  %20 = raw_pointer_to_ref %19 : $Builtin.RawPointer to $_EmptyArrayStorage
  %21 = unchecked_ref_cast %20 : $_EmptyArrayStorage to $Builtin.BridgeObject
  debug_value %12 : $Int
  debug_value %12 : $Int
```




```
define hidden void @_TF5noahs4mainFT_T_() #0 {
entry:
  %0 = alloca %Sa, align 8
  %1 = alloca %Vs17IndexingGenerator, align 8
  %2 = alloca %VSS13CharacterView, align 8
  %3 = alloca %Vs17IndexingGenerator, align 8
  %4 = alloca %Sq.20, align 8
  %5 = alloca %SS, align 8
  %6 = bitcast %Sa* %0 to i8*
  call void @llvm.lifetime.start(i64 8, i8* %6)
  %7 = bitcast %Vs17IndexingGenerator* %1 to i8*
  call void @llvm.lifetime.start(i64 64, i8* %7)
  %8 = call { i8*, i64, i64 } @_TFSSCfT26_builtinUTF16StringLiteralBp17numberOfCodeUnitsBw_SS(i8* bitcast ([125 x i16]* @0 to i8*), i64 124)
  %9 = extractvalue { i8*, i64, i64 } %8, 0
  %10 = extractvalue { i8*, i64, i64 } %8, 1
  %11 = extractvalue { i8*, i64, i64 } %8, 2
  %12 = call { %swift.bridge*, i8* } @_TTSg5SS___TFs27_allocateUninitializedArrayurFBwTGSax_Bp_(i64 0)
  %13 = extractvalue { %swift.bridge*, i8* } %12, 0
  %14 = extractvalue { %swift.bridge*, i8* } %12, 1
  %15 = bitcast i8* %14 to %SS*
  %16 = call %swift.bridge* @_TTSg5SS___TFSaCft12arrayLiteralGSax__GSax_(%swift.bridge* %13)
  %._buffer = getelementptr inbounds %Sa, %Sa* %0, i32 0, i32 0
  %._buffer._storage = getelementptr inbounds %Vs12_ArrayBuffer, %Vs12_ArrayBuffer* %._buffer, i32 0, i32 0
  %._buffer._storage.rawValue = getelementptr inbounds %Vs14_BridgeStorage, %Vs14_BridgeStorage* %._buffer._storage, i32 0, i32 0
  store %swift.bridge* %16, %swift.bridge** %._buffer._storage.rawValue, align 8
  %17 = call { i8*, i64, i64 } @_TFSSg10charactersVSS13CharacterView(i8* %9, i64 %10, i64 %11)
  %18 = extractvalue { i8*, i64, i64 } %17, 0
  %19 = extractvalue { i8*, i64, i64 } %17, 1
  %20 = extractvalue { i8*, i64, i64 } %17, 2
  %21 = bitcast %VSS13CharacterView* %2 to i8*
  call void @llvm.lifetime.start(i64 24, i8* %21)
```




```
    .section    __TEXT,__text,regular,pure_instructions
    .macosx_version_min 10, 9
    .globl  _main
    .align  4, 0x90
_main:
    .cfi_startproc
    pushq   %rbp
Ltmp0:
    .cfi_def_cfa_offset 16
Ltmp1:
    .cfi_offset %rbp, -16
    movq    %rsp, %rbp
Ltmp2:
    .cfi_def_cfa_register %rbp
    subq    $16, %rsp
    movq    _globalinit_33_1BDF70FFC18749BAB495A73B459ED2F0_token5@GOTPCREL(%rip), %rax
    movq    __TZvOs7Process5_argcVs5Int32@GOTPCREL(%rip), %rcx
    movl    %edi, (%rcx)
    cmpq    $-1, (%rax)
    movq    %rsi, -8(%rbp)
    je  LBB0_2
    movq    _globalinit_33_1BDF70FFC18749BAB495A73B459ED2F0_token5@GOTPCREL(%rip), %rdi
    movq    _globalinit_33_1BDF70FFC18749BAB495A73B459ED2F0_func5@GOTPCREL(%rip), %rax
    movq    %rax, %rsi
    callq   _swift_once
LBB0_2:
    movq    __TZvOs7Process11_unsafeArgvGSpGSpVs4Int8__@GOTPCREL(%rip), %rax
    movq    -8(%rbp), %rcx
    movq    %rcx, (%rax)
    callq   __TF5noahs4mainFT_T_
    xorl    %eax, %eax
    addq    $16, %rsp
    popq    %rbp
    retq
    .cfi_endproc
```

Local variables after SILGen

SILGen emits all local ‘var’iables as heap boxes with alloc_box

```
func f() -> Int {
  var x = 42

  return x
}
```

```
%0 = alloc_box $Int  // var x
%4 = ...
store %4 to %0#1 : $*Int

%6 = load %0#1 : $*Int
strong_release %0#0
return %6 : $Int
```

Box-to-stack promotes heap boxes to stack allocations

All closure captures are by reference

- Not acceptable to leave them on the heap!

IDA View-A | Pseudocode-A | Hex View-1 | Structures

```
__int64 v93; // [rsp+1F8h] [rbp-78h]@1
__int64 v94; // [rsp+200h] [rbp-70h]@1
__int64 v95; // [rsp+208h] [rbp-68h]@1
__int64 v96; // [rsp+210h] [rbp-60h]@1
__int64 v97; // [rsp+218h] [rbp-58h]@1
__int64 v98; // [rsp+220h] [rbp-50h]@1
__int64 v99; // [rsp+228h] [rbp-48h]@1
__int64 v100; // [rsp+230h] [rbp-40h]@1
__int64 v101; // [rsp+238h] [rbp-38h]@1
__int64 v102; // [rsp+240h] [rbp-30h]@1

v0 = _TFSSCfMSSFT26_builtinUTF16StringLiteralBp17numberOfCodeUnitsBw_SS(&unk_100000E90, 124LL
v2 = v1;
v76 = v3;
v4 = _TTSg5SS___TFSs27_allocateUninitializedArrayurFBwTGSaq__Bp_(0LL);
v102 = _TTSg5SS___TFSaCurfMGSaq__Ft12arrayLiteralGSaq___GSaq__(v4);
v91 = _TFSSg10charactersVSS13CharacterView(v0, v2, v76);
v92 = v5;
v93 = v6;
v75 = _TMaGVSs17IndexingGeneratorVSS13CharacterView_(v0, v2);
v49 = _TWPSiSs18_SignedIntegerTypeSs_ptr;
v50 = _TMdSi_ptr;
v51 = _TWPSiSs33_BuiltinIntegerLiteralConvertibleSs_ptr;
v52 = _TMdVSs20_DisabledRangeIndex__ptr;
v53 = _TMdVSS13CharacterView_ptr;
v54 = _TWPVSS13CharacterViewSs9IndexableSs_ptr;
v55 = _TWPVSS13CharacterViewSs12SequenceTypeSs_ptr;
v56 = v75;
v57 = _TWPuRq_Ss9Indexable_GVSs17IndexingGeneratorq__Ss13GeneratorTypeSs_ptr;
v58 = _TMdVSs9Character_ptr;
v59 = _TMdVVSS13CharacterView5Index_ptr;
v60 = _TWPVVSS13CharacterView5IndexSs16ForwardIndexTypeSs_ptr;
v61 = _TMdSi_ptr;
v62 = _TWPSiSs18_SignedIntegerTypeSs_ptr;
v63 = _TMdSi_ptr;
v64 = _TWPSiSs33_BuiltinIntegerLiteralConvertibleSs_ptr;
v65 = _TMdVSs20_DisabledRangeIndex__ptr;
v66 = _TMdVSS13CharacterView_ptr;
v67 = _TMdVSs9Character_ptr;
v68 = _TMdVSs9Character_ptr;
v69 = &v91;
_TFeRq_Ss14CollectionTypezqq_S_9GeneratorGVSs17IndexingGeneratorq__zqq_Ss9Indexable8_Elementq
  &v83,
  _TMdVSS13CharacterView_ptr,
  _TWPVSS13CharacterViewSs14CollectionTypeSs_ptr,
  _TMdVVSS13CharacterView5Index_ptr,
  _TWPVVSS13CharacterView5IndexSs16ForwardIndexTypeSs_ptr);
swift_unknownRelease(v93);
v94 = v83;
v95 = v84;
v96 = v85;
v97 = v86;
v98 = v87;
v99 = v88;
v100 = v89;
v101 = v90;
while ( 1 )
{
  _TFVSs17IndexingGenerator4nextuRq_Ss9Indexable_fRGS_q__FT_GSqqq_S0_8_Element_(&v80, v75, &v94);
  v74 = v80;
  v73 = v81;
  if ( !((v82 ^ 1) & 1) )
    break;
  if ( ~v81 & 1 )
    swift_retain_noresult(v80);
  v7 = _TFSSCfMSSFVSs9CharacterSS(v74, v73);
  v72 = v8;
  v71 = v7;
  v70 = v9;
  if ( ~v73 & 1 )
```

_UNKNOWN

000007A2 __TF5noahs4mainFT_T_:106

noahs.swift

noahs.swift 〉 main()

```
func main() {

    let 🌍 = "🐶🐴🐱🐭🐹🐰🐸🐯🐨🐻🐷🐽🐮🐗🐵🙈🙉🙊🐒🐔🐧🐦🐤🐣🐥🦆🦅🦉🦇🐺🐴🦄🐝🐛🦋🐌🐞🐜🦗🕷🦂🐢🐍🦎🦖🦕🐙🦑🦐🦞🦀🐡🐠🐟🐬🐳🐋🦈🐊🐅🐆🦓🦍🐘🦏🦛🐪🐫🦒🦘🐃🐂🐄🐎🐖🐏🐑🦙🐐🦌🐕🐩🐈🐓🦃🕊🐇🐁🐀🐿🦔🐾🐉🐲🌵🎄🌲🌳🌴🌱🌿☘🍀🎍🎋🍃🍂🍁🍄🌾💐🌷🌹🥀🌺🌸🌼🌻🌞🌝🌛🌜🌚🌕🌖🌗🌘🌑🌒🌓🌔🌙🌎🌍🌏💫⭐🌟✨⚡🔥💥☄🌪🌈☀🌤⛅🌥☁🌦🌧⛈🌩🌨❄☃⛄🌬💨💧💦☔🌊🌫"

    var 🚂: [String] = []

    for 💕 in 🌍.characters {
        🚂.append(String(💕) + String(💕))
    }

    print(🚂)

}

main()
```

Methodology

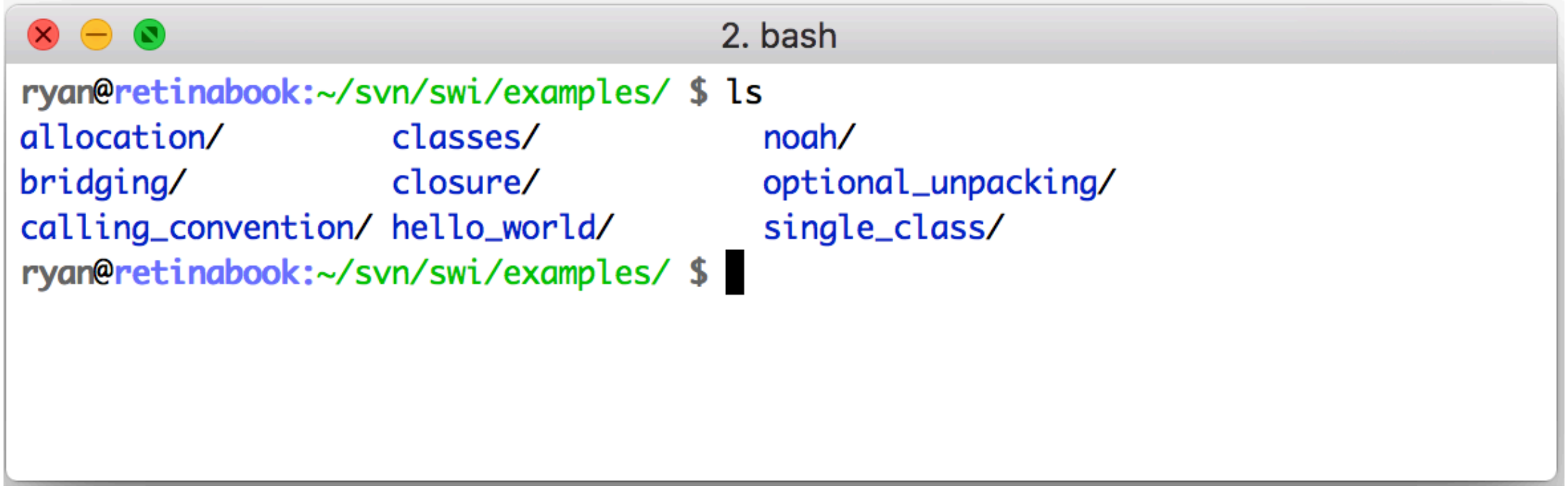
Motivation

- Application Penetration Testing
- Exploit Development
- Re-implementation
- Interoperability
- Build Character

Initial Questions

- Toolchain
  - What tools are available now?
- Language Core
  - Is it message based like Objective-C or does it look more like C/C++?
  - Is it lazy like Haskell?
  - What native types are available?
  - Which storage backs which types of variables?
  - What does class instantiation look like?
  - How are Optionals unwrapped?
- ABI
  - How does Swift bridge into Objective-C?
  - How does it represent virtual method calls under the hood?
  - How are classes and structures laid out in memory?
  - What is the Swift calling convention?

Methodology: Examples



classes.swift

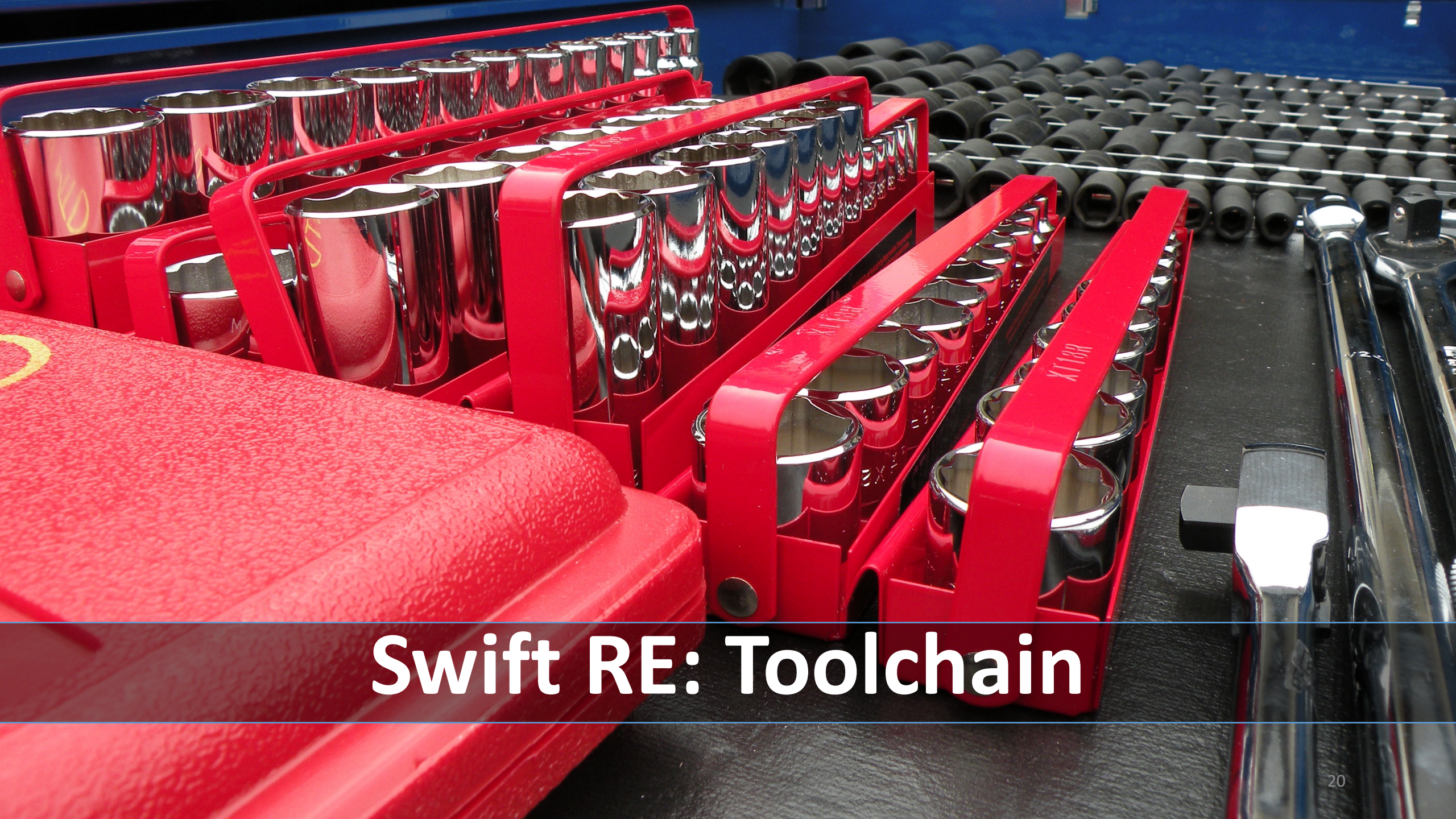
classes.swift 〉 Vehicle

```swift
class Vehicle {
    var currentSpeed = 0.0
    var description: String {
        return "traveling at \(currentSpeed) miles per hour"
    }
    func makeNoise() {
        // do nothing - an arbitrary vehicle doesn't necessarily make a noise
    }
}

class Bicycle: Vehicle {
    var hasBasket = false

    override var description: String {
        if hasBasket {
            return "Bicycles have two wheels this one does have a basket!"
        } else {
            return "Bicycles have two wheels this one doesn't have a basket!"
        }
    }
}

class Tandem: Bicycle {
    var currentNumberOfPassengers = 0
}

class Train: Vehicle {
    override func makeNoise() {
        print("Choo Choo")
    }
}

class Car: Vehicle {
    var gear = 1
    override var description: String {
        return super.description + " in gear \(gear)"
    }
}

func main()
{
    let choice : Int? = Int(Process.arguments[1])

    switch choice
```

closure.swift

closure.swift 〉 No Selection

```
func close(let numbers : [Int])
{
    let mappedNumbers = numbers.map({ number in 3 * number })
    print(mappedNumbers)
}

func main()
{

    let args = Process.arguments
    let tail = args[1...args.count-1]

    let int_args = tail.map{ x in Int(x)! }

    close(int_args)
}

main()
```



Swift RE: Toolchain

swift reversing

All Images Videos Shopping News More Search tools

About 479,000 results (0.43 seconds)

How to reverse a string using reverse() – Swift 2 example code

https://www.hackingwithswift.com/.../how-to-reverse-a-string-using-reve...

How Taylor Swift Reversed Female Opinion To Become The ...

www.buzzfeed.com/.../how-taylor-swift-reversed-female-opinion-to-bec...

Jan 8, 2016 - In 2013 **Swift** was a regular fixture on "Most Hated Celebrities" lists. Then came a carefully strategised comeback in 2014 – and with it, a...

and then initialize a String from them. **Swift** 2.0: let array = ["lizard", "Rhino", ...

Reverse a String in Swift - Ravi Shankar

rshankar.com/**reverse**-a-string-in-**swift**/

Jul 2, 2014 - Here is a simple code snippet written in **Swift** programming language for **reversing** a string. import Cocoa //Assigning a value to a String ...

How to Reverse an array in Swift | iSwift Cookbook

iswift.org/cookbook/**reverse**-an-array

Toolchain

- swiftc
  - The compiler
- swift
  - The compiler REPL
- swift-demangle
  - A name demangler

```
ryan@retinabook:~/svn/swiftre/ $ swift
Welcome to Apple Swift version 2.2 (swiftlang-703.0.18.1 clang-703.0.29). Type :help for
assistance.
  1> let v : Int? = 2
v: Int? = 2
  2> let x : Int? = nil
x: Int? = nil
  3> v!
$R0: Int = 2
  4> x!
fatal error: unexpectedly found nil while unwrapping an Optional value
Execution interrupted. Enter code to recover and continue.
Enter LLDB commands to investigate (type :help for assistance.)
  5> :reg read
General Purpose Registers:
       rax = 0x00007fff74e6a420  libsystem_platform.dylib`_os_lock_type_spin
       rbx = 0x0000000000000000
       rcx = 0x00000000000fc080
       rdx = 0x00000000000f8b10
       rdi = 0x00000001004a2600
       rsi = 0x00000001004a2600
       rbp = 0x00007fff5fbffd10
       rsp = 0x00007fff5fbffd00
        r8 = 0x0000000000000007
        r9 = 0x0000000000000000
       r10 = 0x0000000000000008
       r11 = 0x0000000100500000
       r12 = 0x0000000000000000
       r13 = 0x0000000000000000
       r14 = 0x0000000000000000
       r15 = 0x0000000000000000
       rip = 0x0000000100173728  libswiftCore.dylib`function signature specialization <Ar
g[0] = Exploded, Arg[1] = Exploded, Arg[2] = Dead, Arg[3] = Dead> of Swift._fatalErrorMes
sage (Swift.StaticString, Swift.StaticString, Swift.StaticString, Swift.UInt) -> () + 40
    rflags = 0x0000000000010206
        cs = 0x000000000000002b
        fs = 0x0000000000000000
        gs = 0x0000000000000000
```

```
  5> :x/8i $pc
->  0x100000f60: 55                pushq  %rbp
    0x100000f61: 48 89 e5          movq   %rsp, %rbp
    0x100000f64: 31 c0             xorl   %eax, %eax
    0x100000f66: 5d                popq   %rbp
    0x100000f67: c3                retq
    0x100000f68: ff 25 c2 00 00 00 jmpq   *0xc2(%rip)              ; (void *)0x00007fff
9af4a333: mach_absolute_time
    0x100000f6e: ff 25 c4 00 00 00 jmpq   *0xc4(%rip)              ; (void *)0x00007fff
9af4b07c: mach_timebase_info
    0x100000f74: ff 25 c6 00 00 00 jmpq   *0xc6(%rip)              ; (void *)0x00000001
00233aa0: swift_once
  5>
```

swift-demangle

```
$ echo
'__TFeRq_Ss14CollectionTypezqq_S_9GeneratorGVSs17IndexingGeneratorq__zqq_Ss9Indexabl
e8_Elementqqq_S_9GeneratorSs13GeneratorType7Element_SsS_8generateuRq_S_zqq_S_9Genera
torGS0_q__zqq_S1_8_Elementqqq_S_9GeneratorS2_7Element_fq_FT_GS0_q__' | xcrun swift-
demangle
```

```
_ext.Swift.Swift.CollectionType<A where A: Swift.CollectionType, A.Generator ==
Swift.IndexingGenerator<A>, A._Element == A.Generator.Element>.generate <A where A:
Swift.CollectionType, A.Generator == Swift.IndexingGenerator<A>, A._Element ==
A.Generator.Element> (A)() -> Swift.IndexingGenerator<A>
```

```
$ echo
‘_TTSf4n_d___TTSg5C11CommandLine6Option___TZFSa28_allocateBufferUninitializedurfMGSa
q__FSiGVSs12_ArrayBufferq__' | xcrun swift-demangle
```

```
function signature specialization <Arg[1] = Dead> of generic specialization
<CommandLine.Option> of static Swift.Array._allocateBufferUninitialized <A>
([A].Type)(Swift.Int) -> Swift._ArrayBuffer<A>
```

```
$ echo
'__TFeRq_Ss14CollectionTypezqq_S_9GeneratorGVSs17I
ndexingGeneratorq__zqq_Ss9Indexable8_Elementqqq_S_
9GeneratorSs13GeneratorType7Element_SsS_8generateu
Rq_S_zqq_S_9GeneratorGS0_q__zqq_S1_8_Elementqqq_S_
9GeneratorS2_7Element_fq_FT_GS0_q__' | xcrun
swift-demangle -expand


_Demangling for
_TFeRq_Ss14CollectionTypezqq_S_9GeneratorGVSs17IndexingGeneratorq__zqq_Ss9Indexable8
_Elementqqq_S_9GeneratorSs13GeneratorType7Element_SsS_8generateuRq_S_zqq_S_9Generato
rGS0_q__zqq_S1_8_Elementqqq_S_9GeneratorS2_7Element_fq_FT_GS0_q__

kind=Global
  kind=Function
    kind=Extension
      kind=Module, text="Swift"
      kind=Protocol
        kind=Module, text="Swift"
        kind=Identifier, text="CollectionType"
      kind=DependentGenericSignature
        kind=DependentGenericParamCount, index=1
        kind=DependentGenericConformanceRequirement
          kind=Type
            kind=DependentGenericParamType, text="A"
              kind=Index, index=0
              kind=Index, index=0
          kind=Type
            kind=Protocol
              kind=Module, text="Swift"
              kind=Identifier, text="CollectionType"
        kind=DependentGenericSameTypeRequirement
          kind=Type
            kind=DependentMemberType, text="Generator"
              kind=Type
                kind=DependentGenericParamType, text="A"
                  kind=Index, index=0
                  kind=Index, index=0
              kind=Type
                kind=Protocol
                  kind=Module, text="Swift"
                  kind=Identifier, text="CollectionType"
          kind=Type
            kind=BoundGenericStructure
              kind=Type
                kind=Structure
                  kind=Module, text="Swift"
                  kind=Identifier, text="IndexingGenerator"
              kind=TypeList
                kind=Type
                  kind=DependentGenericParamType, text="A"
                    kind=Index, index=0
                    kind=Index, index=0
        kind=DependentGenericSameTypeRequirement
          kind=Type
            kind=DependentMemberType, text="_Element"
              kind=Type
                kind=DependentGenericParamType, text="A"
                  kind=Index, index=0
                  kind=Index, index=0
              kind=Type
                kind=Protocol
                  kind=Module, text="Swift"
                  kind=Identifier, text="Indexable"
        kind=Type
          kind=DependentMemberType, text="Element"
            kind=Type
              kind=DependentMemberType, text="Generator"
                kind=Type
                  kind=DependentGenericParamType, text="A"
                    kind=Index, index=0
                    kind=Index, index=0
                kind=Type
                  kind=Protocol
                    kind=Module, text="Swift"
                    kind=Identifier, text="CollectionType"
              kind=Type
                kind=Protocol
                  kind=Module, text="Swift"
                  kind=Identifier, text="GeneratorType"
    kind=Identifier, text="generate"
    kind=Type
      kind=DependentGenericType
        kind=DependentGenericSignature
          kind=DependentGenericParamCount, index=1
          kind=DependentGenericConformanceRequirement
            kind=Type
              kind=DependentGenericParamType, text="A"
                kind=Index, index=0
                kind=Index, index=0
            kind=Type
              kind=Protocol
                kind=Module, text="Swift"
                kind=Identifier, text="CollectionType"
          kind=DependentGenericSameTypeRequirement
            kind=Type
              kind=DependentMemberType, text="Generator"
                kind=Type
                  kind=DependentGenericParamType, text="A"
                    kind=Index, index=0
                    kind=Index, index=0
                kind=Type
                  kind=Protocol
                    kind=Module, text="Swift"
                    kind=Identifier, text="CollectionType"
            kind=Type
              kind=BoundGenericStructure
                kind=Type
                  kind=Structure
                    kind=Module, text="Swift"
                    kind=Identifier, text="IndexingGenerator"
                kind=TypeList
                  kind=Type
                    kind=DependentGenericParamType, text="A"
                      kind=Index, index=0
                      kind=Index, index=0
          kind=DependentGenericSameTypeRequirement
            kind=Type
              kind=DependentMemberType, text="_Element"
                kind=Type
                  kind=DependentGenericParamType, text="A"
                    kind=Index, index=0
                    kind=Index, index=0
                kind=Type
                  kind=Protocol
                    kind=Module, text="Swift"
                    kind=Identifier, text="Indexable"
          kind=Type
            kind=DependentMemberType, text="Element"
              kind=Type
                kind=DependentMemberType, text="Generator"
                  kind=Type
                    kind=DependentGenericParamType, text="A"
                      kind=Index, index=0
                      kind=Index, index=0
                  kind=Type
                    kind=Protocol
                      kind=Module, text="Swift"
                      kind=Identifier, text="CollectionType"
                kind=Type
                  kind=Protocol
                    kind=Module, text="Swift"
                    kind=Identifier, text="GeneratorType"
  kind=Type
    kind=UncurriedFunctionType
      kind=ArgumentTuple
        kind=Type
          kind=DependentGenericParamType, text="A"
            kind=Index, index=0
            kind=Index, index=0
      kind=ReturnType
        kind=Type
          kind=FunctionType
            kind=ArgumentTuple
              kind=Type
                kind=NonVariadicTuple
            kind=ReturnType
              kind=Type
                kind=BoundGenericStructure
                  kind=Type
                    kind=Structure
                      kind=Module, text="Swift"
                      kind=Identifier, text="IndexingGenerator"
                  kind=TypeList
                    kind=Type
                      kind=DependentGenericParamType, text="A"
                        kind=Index, index=0
                        kind=Index, index=0

ext.Swift.Swift.CollectionType<A where A: Swift.CollectionType,
A.Generator == Swift.IndexingGenerator<A>, A._Element ==
A.Generator.Element>.generate <A where A: Swift.CollectionType,
A.Generator == Swift.IndexingGenerator<A>, A._Element ==
A.Generator.Element> (A)() -> Swift.IndexingGenerator<A>
```

Initial Questions: Revisited (Toolchain)

- Toolchain
  - What tools are available now? swift-demangle

Swift RE: Language Core

Language Core

- Native types
  - String, Bool, Int, Int8, Int16, Int32, Int64, UInt, UInt8, UInt16, UInt32, UInt64, Float, Float80, Double
  - No tagged pointers in Swift (but will be in the Objc bridges)
- Control Flow
- Optionals
- Class Instantiation

```
v16 = function signature specialization
v17 = (_QWORD *)v16;
*(_QWORD *)(v16 + 16) = 3LL;
v113 = "traveling at ";
v114 = 13LL;
v115 = 0LL;
*(_QWORD *)(v16 + 32) = generic speciali
v17[5] = v18;
v17[6] = v19;
v112 = *(_QWORD *)(v14 + 16);
```

Messages? Laziness?

```
if ( 2 == choice )
{
  v20 = type metadata accessor for classes.Train();
  v21 = classes.Train.__allocating_init (classes.Train.Type)() -> classes.Train((_TtMC7classes5Train)v20).pointer;
  ((void (__fastcall *)(classes::Train *, __int64))v21->pointer[13].pointer)(v21, v50);
  return swift_release(v21);
}
```

```
 8  v3 = a3;
 9  v4 = (_QWORD *)type metadata accessor for __ObjC.BridgeTest();
10  v5 = __ObjC.BridgeTest.__allocating_init (__ObjC.BridgeTest.Type) -> () -> __ObjC.BridgeTest(v4);
11  objc_msgSend(v5, selRef_class_method1);
12  objc_release(v5);
13  if ( !(v3 & 1) )
14  {
15    if ( (a2 & 0x8000000000000000LL) != 0LL )
16    {
17      swift_unknownRelease(a2 & 0x7FFFFFFFFFFFFFFFLL);
18    }
19    else if ( ~(a2 >> 63) & 1 )
20    {
21      swift_release(a2);
22    }
23  }
24  objc_release(a1);
25  return 1;
26}
```

Optionals

- Swift has optionals which alleviates a lot of null/nil pointer problems.

optional.swift

optional.swift 〉 main(_:)

```
func main(let optionalName: String?)
{
    print(optionalName == nil)

    var greeting = "Hello!"
    if let name = optionalName {
        greeting = "Hello, \(name)"
    }

    print(greeting)
}

main(nil)
main("Ryan")
```

classes.swift

classes.swift 〉 Vehicle

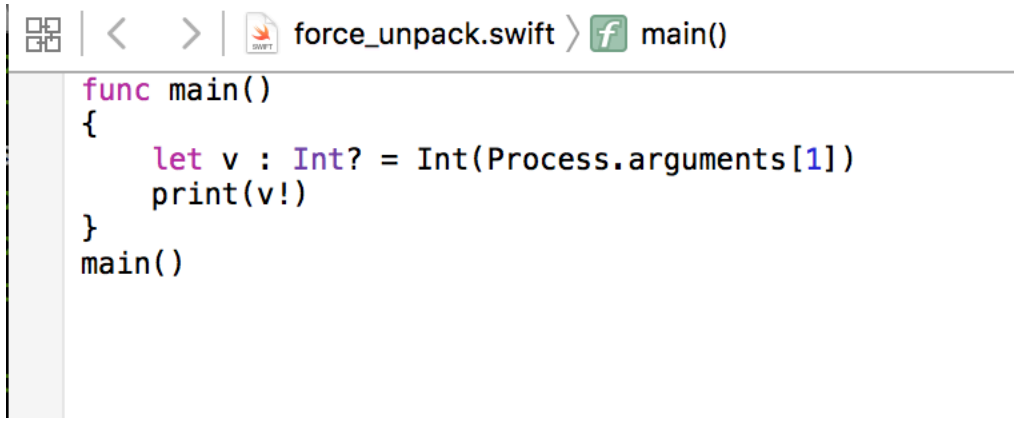
```
func main()
{
    let choice : Int? = Int(Process.arguments[1])

    switch choice
    {
    case .Some(1):
        let bike = Bicycle()
        bike.hasBasket = true
        print(bike.description)
    case .Some(2):
        let train = Train()
        train.makeNoise()
    case .Some(3):
        let car = Car()
        print(car.description)
    default:
        print("Invalid choice!")

    }
}

main()
```

```
00000000`00000002  00                00000000`00000000  01
[  Value = 2     ] [Op]              [  Value = nil   ] [Op]
```

```
(lldb) list
   50  =>            case .Some(2):
   51                        let train = Train()
   52                        train.makeNoise()
   53            case .Some(3):
   54                        let car = Car()
   55                        print(car.description)
   56            default:
   57                        print("Invalid choice!")
   58
   59            }
(lldb) reg read
General Purpose Registers:
       rax = 0x0000000000000002
       rbx = 0x0000000000000000
       rcx = 0x0000000000000002
       rdx = 0x0000000000000002
       rdi = 0x0000000100702b80
       rsi = 0x000000000000000a
       rbp = 0x00007fff5fbff9b0
       rsp = 0x00007fff5fbff840
        r8 = 0x0000000000000000
        r9 = 0x0000000000000000
       r10 = 0x00000001002ad201
       r11 = 0x00000001000dfcc0
       r12 = 0x0000000000000000
       r13 = 0x0000000000000000
       r14 = 0x0000000000000000
       r15 = 0x0000000000000000
       rip = 0x000000010000148e  classes`classes.main () -> () + 446 at
classes.swift:50
    rflags = 0x0000000000000297
```

```
        cs = 0x000000000000002b
        fs = 0x0000000000000000
        gs = 0x0000000000000000

(lldb) x/8i $pc
->  0x10000148e: 48 39 d1              cmpq   %rdx, %rcx
    0x100001491: 75 40                 jne    0x1000014d3
    0x100001493: e8 e8 02 00 00        callq  0x100001780
    0x100001498: 48 89 c7              movq   %rax, %rdi
    0x10000149b: e8 b0 fb ff ff        callq  0x100001050
    0x1000014a0: 48 89 45 b0           movq   %rax, -0x50(%rbp)
    0x1000014a4: 48 8b 38              movq   (%rax), %rdi
    0x1000014a7: 48 89 bd 20 ff ff ff  movq   %rdi, -0xe0(%rbp)
(lldb) x/40xg $rbp-0x28
0x7fff5fbff988: 0x0000000000000002 0x0000000000000300
```

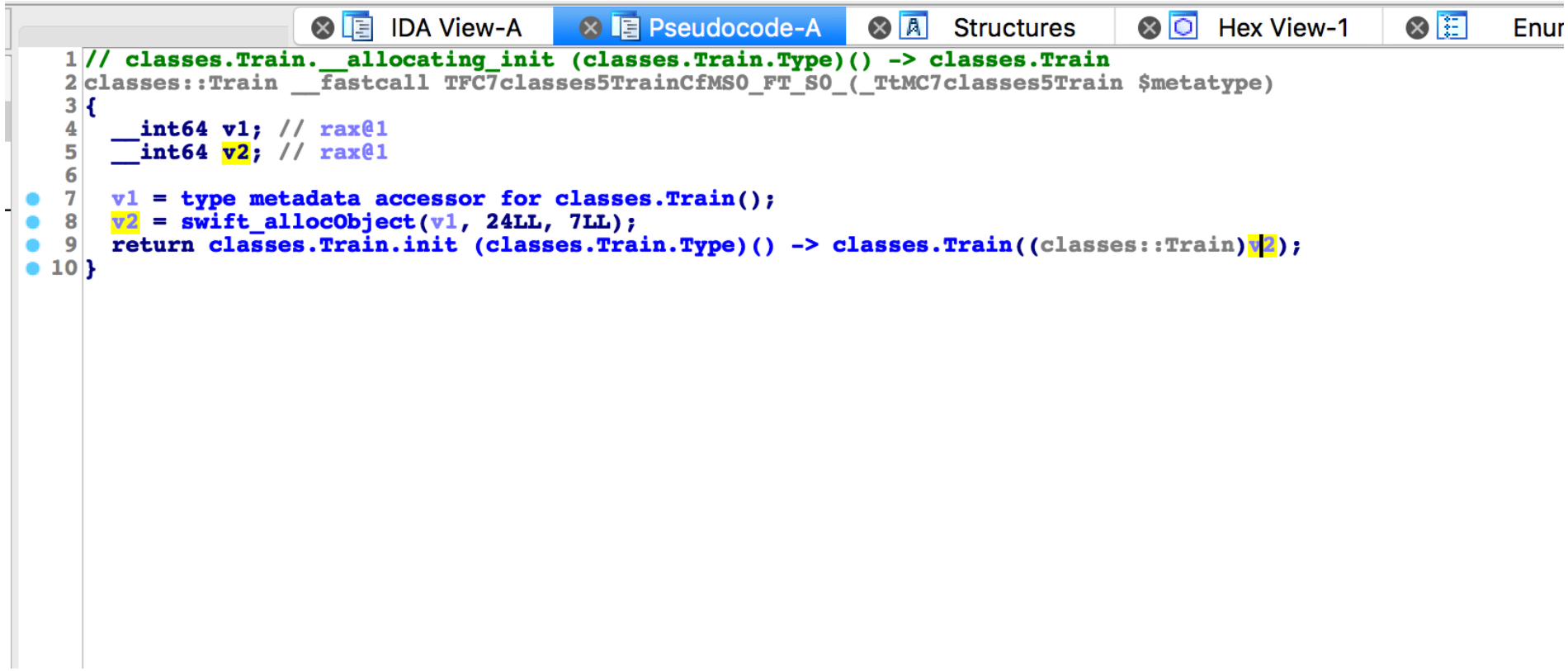
```
if ( !(v51 & 1) )
  swift_unknownRetain(v52);
if ( !(v51 & 1) )
{
  v12 = generic specialization <Swift.String>
  v14 = v13;
```




```
52  v10[2] = 1LL;
53  v10[7] = direct type metadata for Swift.Int;
54  if ( v9 & 1 )
55LABEL_11:
56    BUG();
57  v10[4] = v7;
58  v11 = Swift.(print (Swift.Array<protocol<>>, separa
```

```
ryan@retinabook:~/svn/swi/exa/optional_unpacking/ $ ./force_unpack 1
1
ryan@retinabook:~/svn/swi/exa/optional_unpacking/ $ ./force_unpack INFILTRATE
Illegal instruction: 4
[132] ryan@retinabook:~/svn/swi/exa/optional_unpacking/ $
```

Dynamic Allocation and Class Instantiation




```
RefCounted *swift_allocObject(Metadata *type, size_t size, size_t alignMask);
```

```
/// Allocates a new heap object.  The returned memory is
/// uninitialized outside of the heap-object header.  The object
/// has an initial retain count of 1, and its metadata is set to
/// the given value.
///
/// At some point "soon after return", it will become an
/// invariant that metadata->getSize(returnValue) will equal
/// requiredSize.
///
/// Either aborts or throws a swift exception if the allocation fails.
///
/// \param requiredSize - the required size of the allocation,
///   including the header
/// \param requiredAlignmentMask - the required alignment of the allocation;
///   always one less than a power of 2 that's at least alignof(void*)
/// \return never null
///
/// POSSIBILITIES: The argument order is fair game.  It may be useful
/// to have a variant which guarantees zero-initialized memory.
SWIFT_RUNTIME_EXPORT
extern "C" HeapObject *swift_allocObject(HeapMetadata const *metadata,
                                         size_t requiredSize,
                                         size_t requiredAlignmentMask);
```

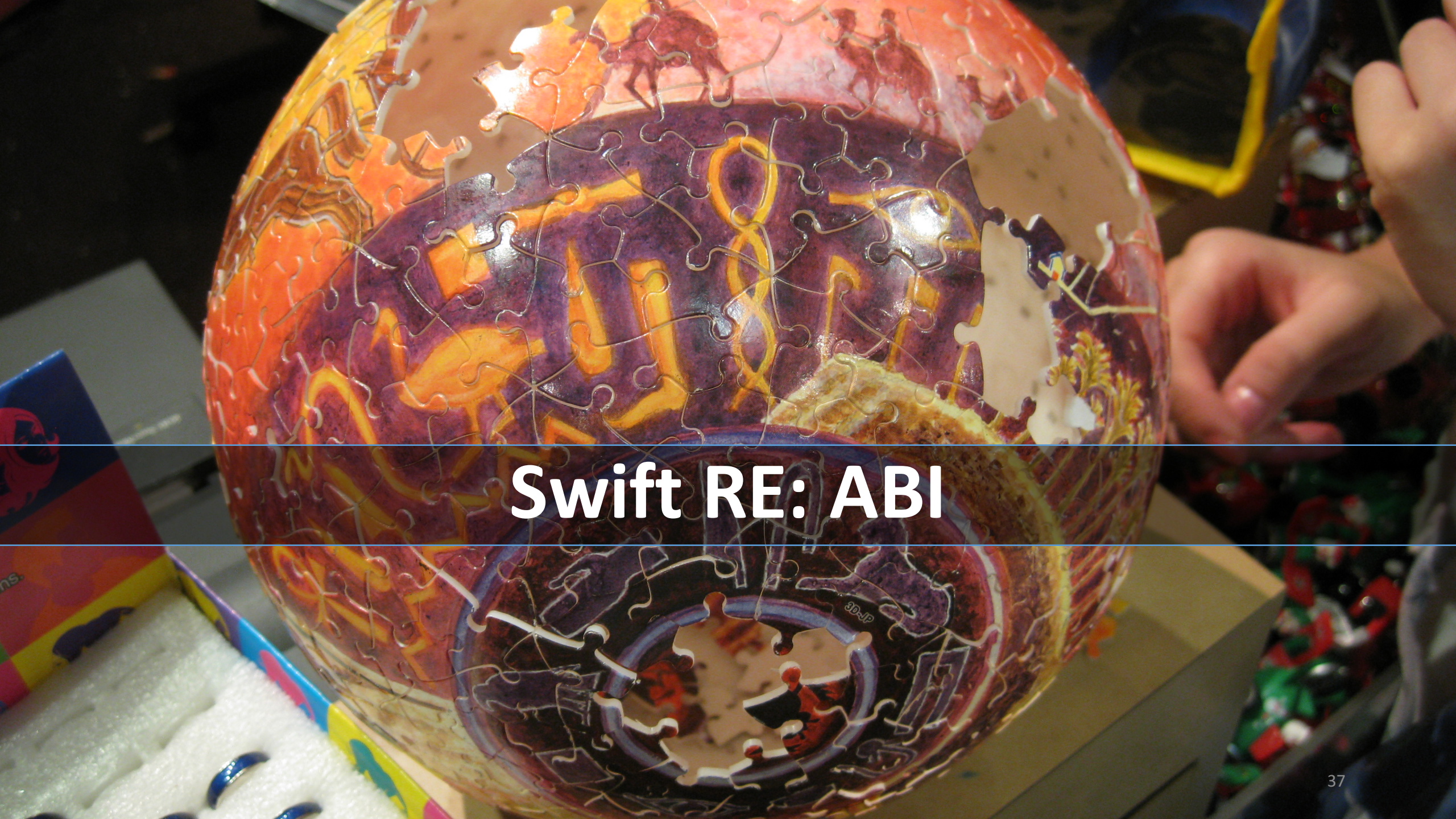
```
1 // classes.Train.init (classes.Train.Type)() -> classes.Train
2 classes::Train __fastcall TFC7classes5TraincfMS0_FT_S0_(classes::Train self)
3 {
4   classes::Train result; // rax@1
5
6   result.pointer = (classes::Train *)classes.Vehicle.init (classes.Vehicle.Type)() -> classes.Vehicle((classes::Vehicle)self.pointer).pointer;
7   return result;
8 }
```

IDA View-A | Pseudocode-A | Structures | Hex View-1 | Enums | Imports | Expo

```
1  // type metadata accessor for classes.Train
2  __int64 _TMaC7classes5Train()
3  {
4    __int64 v1; // [rsp+8h] [rbp-8h]@1
5
6    v1 = lazy cache variable for type metadata for classes.Train;
7    if ( !lazy cache variable for type metadata for classes.Train )
8    {
9      lazy cache variable for type metadata for classes.Train = swift_getInitializedObjCClass(&full type metadata for classes.Train + 2);
10     v1 = lazy cache variable for type metadata for classes.Train;
11   }
12   return v1;
13 }
```

Initial Questions: Revisited (Language Core)

- Language Core
  - Is it message based like Objective-C or does it look more like C/C++? C++
  - Is it lazy like Haskell? No, thank God
  - What native types are available? The usuals
  - Which storage backs which types of variables? Stack, Heap, depends on lifetime
  - What does class instantiation look like? Slightly different than C++
  - How are Optionals unwrapped? With a bitwise AND



Swift RE: ABI

ABI

- Objective-C Bridging
- Virtual function calls
- Ownership rules
- Calling convention

Objective-C Bridging

```
func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
    // Override point for customization after application launch.

    let obj = BridgeTest()
    obj.class_method1()

    return true
}
```

```
 8   v3 = a3;
 9   v4 = (_QWORD *)type metadata accessor for __ObjC.BridgeTest();
10   v5 = __ObjC.BridgeTest.__allocating_init (__ObjC.BridgeTest.Type) -> () -> __ObjC.BridgeTest(v4);
11   objc_msgSend(v5, selRef_class_method1);
12   objc_release(v5);
13   if ( !(v3 & 1) )
14   {
15     if ( (a2 & 0x8000000000000000LL) != 0LL )
16     {
17       swift_unknownRelease(a2 & 0x7FFFFFFFFFFFFFFFLL);
18     }
19     else if ( ~(a2 >> 63) & 1 )
20     {
21       swift_release(a2);
22     }
23   }
24   objc_release(a1);
25   return 1;
26 }
```

```objc
#import "objc.h"

@implementation BridgeTest : NSObject

- (void) class_method1
{
    NSLog(@"called class_method!");
}

@end
```

```
1 // BridgeTest - (void)class_method1
2 void __cdecl -[BridgeTest class_method1](struct BridgeTest *self, SEL a2)
3 {
4   NSLog(&cfstr_CalledClass_me);
5 }
```

Virtual Function Calls

```
if ( 2 == choice )
{
  v20 = type metadata accessor for classes.Train();
  v21 = classes.Train.__allocating_init (classes.Train.Type)() -> classes.Train((_TtMC7classes5Train)v20).pointer;
  ((void (__fastcall *)(classes::Train *, __int64))v21->pointer[13].pointer)(v21, v50);
  return swift_release(v21);
}
```

classes.swift

classes.swift 〉 Vehicle

```
func main()
{
    let choice : Int? = Int(Process.arguments[1])

    switch choice
    {
    case .Some(1):
        let bike = Bicycle()
        bike.hasBasket = true
        print(bike.description)
    case .Some(2):
        let train = Train()
        train.makeNoise()
    case .Some(3):
        let car = Car()
        print(car.description)
    default:
        print("Invalid choice!")
```

Ownership and Ownership Rules

- Swift is full ARC
  - Automatic Reference Counting
  - Everything is derived from a few base types, which include the reference counts.
- Functions understand their argument ownership rules
  - Dead
  - Guaranteed
  - Exploded
  - Guaranteed and Exploded

Calling Convention

- Swift’s approach:
  - YOLO
  - External calls are RAX:RDX:RCX:R8
- __swiftcall is not supported in HexRays
- Scattered return values
  - Hexrays has a lot of trouble with them :(

IDA Pro Documentation

Since compilers can use such complex calling conventions, IDA needs some mechanism to describe them. Scattered argument locations are used for that. The above calling convention can be described like this:

```
void __usercall myfunc(struc_1 s@<0:rdi.1, 2:rdi^2.2, 4:rdi^4.1, 8:rsi.4>);
```

It reads:

```
1 byte  at offset 0 of the argument is  passed in the byte 0 of RDI
2 bytes at offset 2 of the argument are passed in the byte 1,2 of RDI
1 byte  at offset 4 of the argument is  passed in the byte 3 of RDI
4 bytes at offset 8 of the argument are passed starting from the byte 0 of RSI
```

In other words, the following syntax is used:

```
argoff:register^regoff.size
```

where

```
argoff - offset within the argument
register - register name used to pass part of the argument
regoff - offset within the register
size - number of bytes
```

__swiftcall

```
Swift::String __usercall __spoils<rax,rdx,rcx,r8> func@<0:rdx,
8:rax, 16:rcx>(void *a1, void *a2)
```

```
Swift::String *__cdecl func(Swift::String *__return_ptr
__struct_ptr retstr, void *a1, void *a2);
```

Initial Questions: Revisited (ABI)

- ABI
  - How does Swift bridge into Objective-C? Seamlessly
  - How does it represent virtual method calls under the hood? Similar to C++
  - How are classes and structures laid out in memory? Exactly like Objective-c
  - What is the Swift calling convention? Yolo

Tools

swift.py

- IDA and HexRays plugin
  - Rewrites Hex-Rays output to demangle names
  - Annotates IDA with demangled names
  - Class body recovery
  - Type propagation (Coming Soon)
  - Witness table recovery (Coming soon – Hopefully)

Demo

Questions?

Ryan Stortz

- Principal Security Researcher at Trail of Bits
- Previously at Raytheon SIGOVS

Contact Information:

- @withzombies
- ryan@trailofbits.com