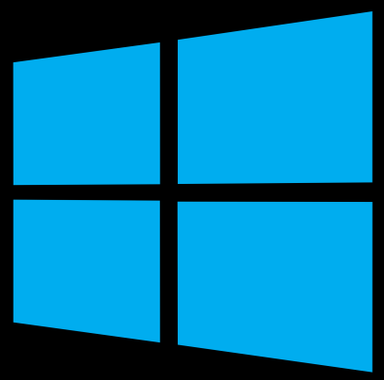
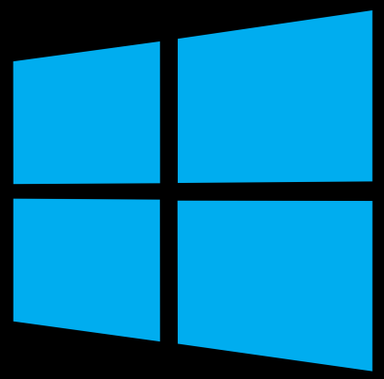
# Death by a 1000 Cuts: Bringing Swift to Windows

Saleem Abdulrasool (@compnerd)



# Porting by a 1000 Patches: Bringing Swift to Windows

Saleem Abdulrasool (@compnerd)



# Why Swift?

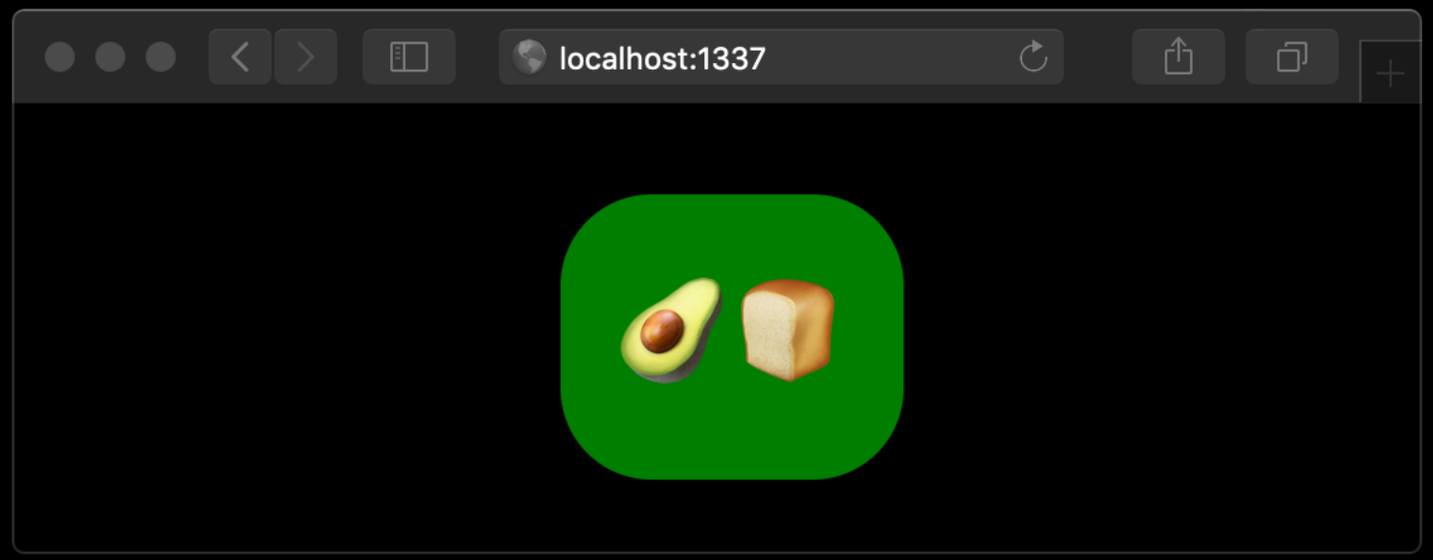
# Why Swift?

- Safe

# Why Swift?

- Safe
- Flexible

```
import SwiftWebUI

struct MainPage: View {
  var body: some View {
    VStack {
      Text("🥑🍞")
        .padding(.all)
        .background(.green, cornerRadius: 12)
        .foregroundColor(.white)
    }
  }
}
```



# Why Swift?

- Safe
- Flexible
- Multi-paradigm

# Why Swift?

- Safe
- Flexible
- Multi-paradigm
- Compiled Language

# Why Swift?

- Safe
- Flexible
- Multi-paradigm
- Compiled Language
- Break from legacy codebase

# Why Swift?

- Safe
- Flexible
- Multi-paradigm
- Compiled Language
- Break from legacy codebase
- Systems Development

# Why Windows?

# Why Windows?

- Access to developers and users

# Why Windows?

- Access to developers and users
- Enables Portable System and Application Code

# Why Windows?

- Access to developers and users
- Enables Portable System and Application Code
- Improves the Swift and LLVM projects codebases

# Why Windows?

- Access to developers and users
- Enables Portable System and Application Code
- Improves the Swift and LLVM projects codebases
- Interesting Challenge

# Pawn Takes Queen

# Pawn Takes Queen

- Compiler

# Pawn Takes Queen

- Compiler
- Runtime/Standard Library

# Pawn Takes Queen

- Compiler
- Runtime/Standard Library
- Core Libraries (libdispatch, Foundation, XCTest)

# Pawn Takes Queen

- Compiler
- Runtime/Standard Library
- Core Libraries (libdispatch, Foundation, XCTest)
- Debugger (lldb)

# Pawn Takes Queen

- Compiler
- Runtime/Standard Library
- Core Libraries (libdispatch, Foundation, XCTest)
- Debugger (lldb)
- Developer Tools (SourceKit-LSP, swift-package-manager)

# The Tortoise & The Hare

# The Tortoise & The Hare

- The Windows community **is** interested

# The Tortoise & The Hare

- The Windows community **is** interested
- Previous Attempts

# The Tortoise & The Hare

- The Windows community **is** interested
- Previous Attempts
  - cygwin
  - MinGW

# The Tortoise & The Hare

- The Windows community **is** interested
- Previous Attempts
  - cygwin
  - MinGW
  - WSL

# The Tortoise & The Hare

- The Windows community **is** interested
- Previous Attempts
  - cygwin
  - MinGW
  - WSL
- Windows Swift

# Instructions Not Included

# Instructions Not Included

- CMake

# Instructions Not Included

- CMake
- autotools

# Instructions Not Included

- CMake
- autotools
- custom build systems

# Instructions Not Included

- CMake
- autotools
- custom build systems
- build-script

# An Alien Planet

# An Alien Planet

- ~~bash~~ cmd
- ~~make~~ nmake

# An Alien Planet

- ~~bash~~ cmd
- ~~make~~ nmake
- Windows' VFS is slower than Linux's VFS

# An Alien Planet

- ~~bash~~ cmd
- ~~make~~ nmake
- Windows' VFS is slower than Linux's VFS
- cross-compilation conveniently solves these problems

# All I Have is a Hammer

# All I Have is a Hammer

- compiler
  - clang, clang-cl - VFS

# All I Have is a Hammer

- compiler
  - clang, clang-cl - VFS
- assembler
  - IAS - AT&T vs Intel

# All I Have is a Hammer

- compiler
  - clang, clang-cl - VFS
- assembler
  - IAS - AT&T vs Intel
- linker
  - gold, bfd - ELF only, lack of MS SDK support

# All I Have is a Hammer

- compiler
  - clang, clang-cl - VFS
- assembler
  - IAS - AT&T vs Intel
- linker
  - gold, bfd - ELF only, lack of MS SDK support
  - link - must build on Windows

# All I Have is a Hammer

- compiler
  - clang, clang-cl - VFS
- assembler
  - IAS - AT&T vs Intel
- linker
  - gold, bfd - ELF only, lack of MS SDK support
  - link - must build on Windows
  - lld - couldn’t generate import libraries

# All I Have is a Hammer

- compiler
  - clang, clang-cl - VFS
- assembler
  - IAS - AT&T vs Intel
- linker
  - gold, bfd - ELF only, lack of MS SDK support
  - link - must build on Windows
  - lld - couldn’t generate import libraries
  - Symlink Forest

# 99 Standards on The Wall

# 99 Standards on The Wall

- C++ is dark and full of terrors

```
-         size = Builder.CreateAdd(
-             Builder.CreateAnd(Builder.CreateAdd(heapHeaderSize, alignmentMask),
-                               Builder.CreateNot(alignmentMask)),
-             size);
+         auto *Add = Builder.CreateAdd(heapHeaderSize, alignmentMask);
+         auto *Not = Builder.CreateNot(alignmentMask);
+         size = Builder.CreateAdd(Builder.CreateAnd(Add, Not), size);
```

# 99 Standards on The Wall

- C++ is dark and full of terrors

```
-       return OwnedString(StringRef(OwnedPtr->getText(), Str.size()),
-                          std::move(OwnedPtr));
+       // Allocate the StringRef on the stack first.  This is to ensure that the
+       // order of evaluation of the arguments is specified.  The specification
+       // does not specify the order of evaluation for the arguments.  Itanium
+       // chose to evaluate left to right, while Windows evaluates right to left.
+       // As such, it is possible that the OwnedPtr has already been `std::move`d
+       // by the time that the StringRef is attempted to be created.  In such a
+       // case, the offset of the field (+4) is used instead of the pointer to
+       // the text, resulting in invalid memory references.
+       StringRef S(OwnedPtr->getText(), Str.size());
+       return OwnedString(S, std::move(OwnedPtr));
```

# 99 Standards on The Wall

- C++ is dark and full of terrors
  - clang-tidy
- libstdc++ vs libc++ vs msvcprt

# 99 Standards on The Wall

```
+#if os(Windows)
+public typealias ThreadHandle = HANDLE
+#else
+public typealias ThreadHandle = pthread_t
+#endif

-public func _stdlib_pthread_create_block<Argument, Result>(
+public func _stdlib_thread_create_block<Argument, Result>(
   _ start_routine: @escaping (Argument) -> Result, _ arg: Argument
-) -> (CInt, pthread_t?) {
+) -> (CInt, ThreadHandle?) {
   let context = ThreadBlockContextImpl(block: start_routine, arg: arg)
   let contextAsVoidPointer = Unmanaged.passRetained(context).toOpaque()
+#if os(Windows)
+  var threadID =
+      _beginthreadex(nil, 0, { invokeBlockContext($0)!
+                                  .assumingMemoryBound(to: UInt32.self).pointee },
+                     contextAsVoidPointer, 0, nil)
+  return threadID == 0 ? (errno, nil)
+                       : (0, UnsafeMutablePointer<ThreadHandle>(&threadID).pointee)
+#else
```

# 99 Standards on The Wall

- C++ is dark and full of terrors
  - clang-tidy
- libstdc++ vs libc++ vs msvcprt
- libSystem/BSD libc vs glibc vs msvcrt/ucrt vs bionic

# Objective Evaluation

# Objective Evaluation

- Weak Linking

# Objective Evaluation

- Weak Linking

```
+  if (Context.LangOpts.Target.isOSBinFormatCOFF()) {
+    if (DK == DAK_WeakLinked) {
+      diagnose(Loc, diag::attr_unsupported_on_target, AttrName,
+               Context.LangOpts.Target.str());
+      DiscardAttribute = true;
+    }
+  }
```

# Objective Evaluation

- Weak Linking

```
     encodeForceLoadSymbolName(buf, linkLib.getName());
     auto ForceImportThunk =
         Module.getOrInsertFunction(buf, llvm::FunctionType::get(VoidTy, false));
-    ApplyIRLinkage(IRLinkage::ExternalWeakImport)
-        .to(cast<llvm::GlobalValue>(ForceImportThunk));
+
+    const IRLinkage IRL =
+        llvm::Triple(Module.getTargetTriple()).isOSBinFormatCOFF()
+            ? IRLinkage::ExternalImport
+            : IRLinkage::ExternalWeakImport;
+    ApplyIRLinkage(IRL).to(cast<llvm::GlobalValue>(ForceImportThunk));
```

# Objective Evaluation

- Weak Linking
- DLL Storage

# Objective Evaluation

- Weak Linking
- DLL Storage

```
  if (auto fn = dyn_cast<llvm::Function>(cache)) {
    fn->setCallingConv(cc);

+    bool IsExternal =
+        fn->getLinkage() == llvm::GlobalValue::AvailableExternallyLinkage ||
+        (fn->getLinkage() == llvm::GlobalValue::ExternalLinkage &&
+         fn->isDeclaration());
+
+    if (!isStandardLibrary(Module) && IsExternal &&
+        ::useDllStorage(llvm::Triple(Module.getTargetTriple())))
+      fn->setDLLStorageClass(llvm::GlobalValue::DLLImportStorageClass);
```

# Objective Evaluation

- Weak Linking
- DLL Storage
- Multiple Definitions

```
+  case SILLinkage::PublicNonABI:
+    return isDefinition ? RESULT(WeakODR, Hidden, Default)
+                        : RESULT(External, Hidden, Default);
```

# Objective Evaluation

- Weak Linking
- DLL Storage
- Multiple Definitions
- COMDAT Groups

# What did you call me?

# What did you call me?

- Calling Conventions

# What did you call me?

- Calling Conventions
  - PreserveMost
  - SwiftCall

# What did you call me?

- Calling Conventions
  - PreserveMost
  - SwiftCall
- Name Decoration

```
void C();
void __attribute__((__swiftcall__)) Swift();
void __attribute__((__preserve_most__)) PreserveMost();

int CC(void (&)());
template <typename T> int CC(T &);

int r = CC(C)             // ?CC@@YAHA6AXXZ@Z
      + CC(Swift)         // ??$CC@$$A6SXXZ@@YAHP6SXXZ@Z
      + CC(PreserveMost); // ??$CC@$$A6UXXZ@@YAHP6UXXZ@Z
```

# What did you call me?

- Calling Conventions
  - PreserveMost
  - SwiftCall
- Name Decoration
  - Vendor Controlled Platform

# What did you call me?

- Calling Conventions
  - PreserveMost
  - SwiftCall
- Name Decoration
  - Vendor Controlled Platform
  - clang extensions

# What did you call me?

- Calling Conventions
  - PreserveMost
  - SwiftCall
- Name Decoration
  - Vendor Controlled Platform
  - clang extensions
- Language Extensions

# You‘ve Got Mail

# You‘ve Got Mail

- Calling conventions and language boundaries

# You‘ve Got Mail

- Calling conventions and language boundaries

```
#if !SWIFT_OBJC_INTEROP // __SwiftValue is a native class
SWIFT_CC(swift) SWIFT_RUNTIME_STDLIB_INTERNAL
-bool swift_swiftValueConformsTo(const Metadata *);
+bool swift_swiftValueConformsTo(const Metadata *, const Metadata *);
```

```
@_silgen_name("swift_swiftValueConformsTo")
public func _swiftValueConformsTo<T>(_ type: T.Type) -> Bool {
  if let foundationType = _foundationSwiftValueType {
    return foundationType is T.Type
  } else {
    return __SwiftValue.self is T.Type
  }
}
```

# Heisen-Jigsaw Puzzles

# Heisen-Jigsaw Puzzles

- Multiple bugs interact in complicated ways

# Heisen-Jigsaw Puzzles

- Multiple bugs interact in complicated ways

SWIFT

```
   auto addr = getAddrOfLLVMVariable(*entity, ConstantInit(), DbgTy, refKind,
                                     defaultVarTy);
+  if (auto *GV = dyn_cast<llvm::GlobalVariable>(addr.getValue()))
+    GV->setComdat(nullptr);

   // FIXME: MC breaks when emitting alias references on some platforms
   // (rdar://problem/22450593 ). Work around this by referring to the aliasee
```

LLVM

```
          Sym->Aux[0].AuxType = ATWeakExternal;
          Sym->Aux[0].Aux.WeakExternal.TagIndex = 0;
          Sym->Aux[0].Aux.WeakExternal.Characteristics =
-             COFF::IMAGE_WEAK_EXTERN_SEARCH_LIBRARY;
+             COFF::IMAGE_WEAK_EXTERN_SEARCH_ALIAS;
      } else {
          if (!Base)
            Sym->Data.SectionNumber = COFF::IMAGE_SYM_ABSOLUTE;
```

# Creepy Crawlers

# Creepy Crawlers

- PDB Support

# Creepy Crawlers

- PDB Support
- Cross Language Boundaries

# Creepy Crawlers

- PDB Support
- Cross Language Boundaries
- Swift’s Debugging Model

# Aperture Science Lab

# Aperture Science Lab

- IRGen

# Aperture Science Lab

- IRGen
- lit

# Aperture Science Lab

- IRGen
- lit
- Paths

# It’s a Marathon

# It’s a Marathon

Pipelines - Run toolchain

https://dev.azure.com/compnerd/windows-swift/_build/results?buildid=9003

Azure DevOps / Pipelines / x64 Toolchain (VS2019) / toolchain

Search

Sign in

windows-swift

Overview

Boards

Repos

Pipelines

Pipelines

Releases

## #toolchain Update linux-sdk.yml

on x64 Toolchain (VS2019)

Summary

**Scheduled**

compnerd/windows-s... master 8885a77

Sep 18 at 5:00 PM

| | |
|---|---|
| Duration: | 4h 58m 35s |
| Tests: | Get started |
| Changes: | 6 commits |
| Work items: | - |
| Artifacts: | 1 published |

Jobs

| Name | Status | Duration |
|---|---|---|
| windows | Success | 4h 58m 31s |

# It’s a Marathon

- CI
- Testing

# It’s a Marathon

- CI
- Testing
- Components

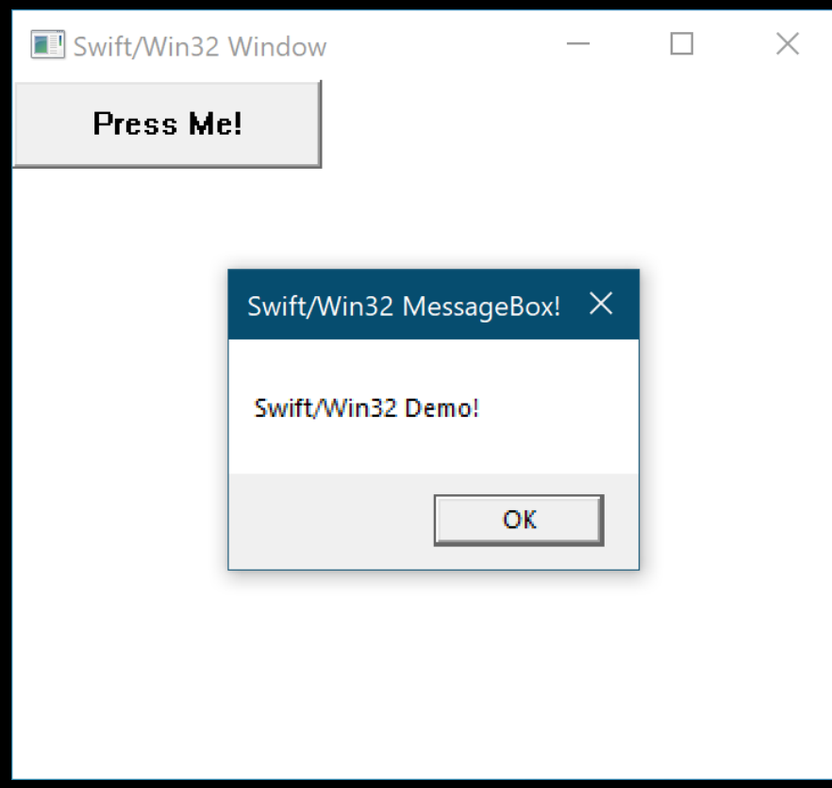
# It’s a Marathon

- CI
- Testing
- Components
- Distributions

# The Old, New Thing

# The Old, New Thing

```swift
import WinSDK
import SwiftWin32     // https://github.com/compnerd/swift-win32

class EventHandler: WindowDelegate {
  func OnDestroy(_ hWnd: HWND?, _ wParam: WPARAM, _ lParam: LPARAM)
      -> LRESULT {
    PostQuitMessage(0)
    return 0
  }

  func OnCommand(_ hWnd: HWND?, _ wParam: WPARAM, _ lParam: LPARAM)
      -> LRESULT {
    MessageBoxW(nil, "Swift/Win32 Demo!".LPCWSTR,
                "Swift/Win32 MessageBox!".LPCWSTR, UINT(MB_OK))
    return 0
  }
}

class SwiftApplicationDelegate: ApplicationDelegate {
  var window = Window(title: "Swift/Win32 Window")
  var button = Button(frame: .zero, title: "Press Me!")
  var delegate = EventHandler()

  func application(_: Application,
                   didFinishLaunchingWithOptions options: [Application.LaunchOptionsKey:Any]?) -> Bool {
    window.addSubview(button)
    window.delegate = delegate
    return true
  }
}

ApplicationMain(CommandLine.argc, CommandLine.unsafeArgv, nil, SwiftApplicationDelegate())
```



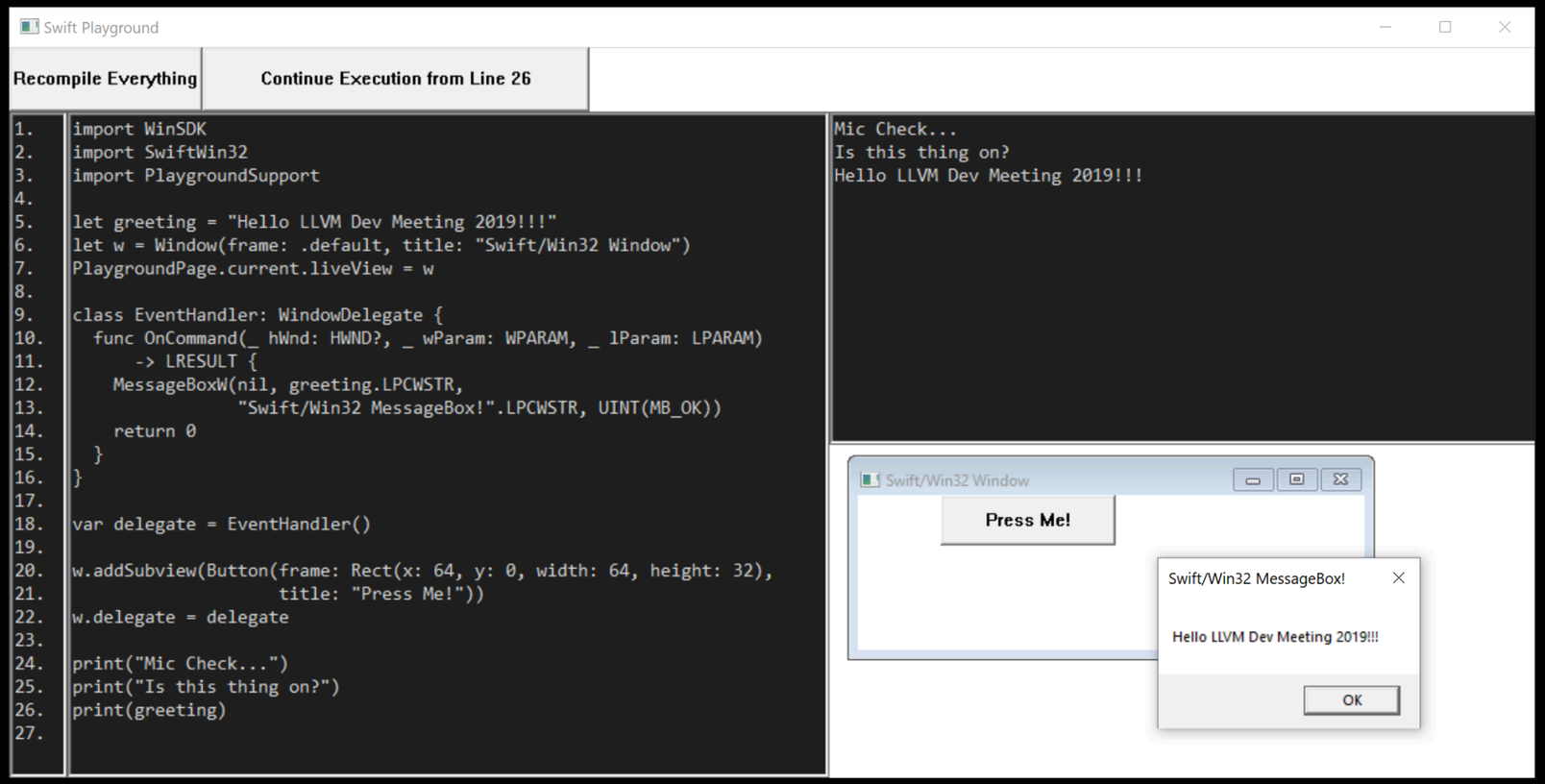
# Quickly Now

# Quickly Now

- Immediate Feedback

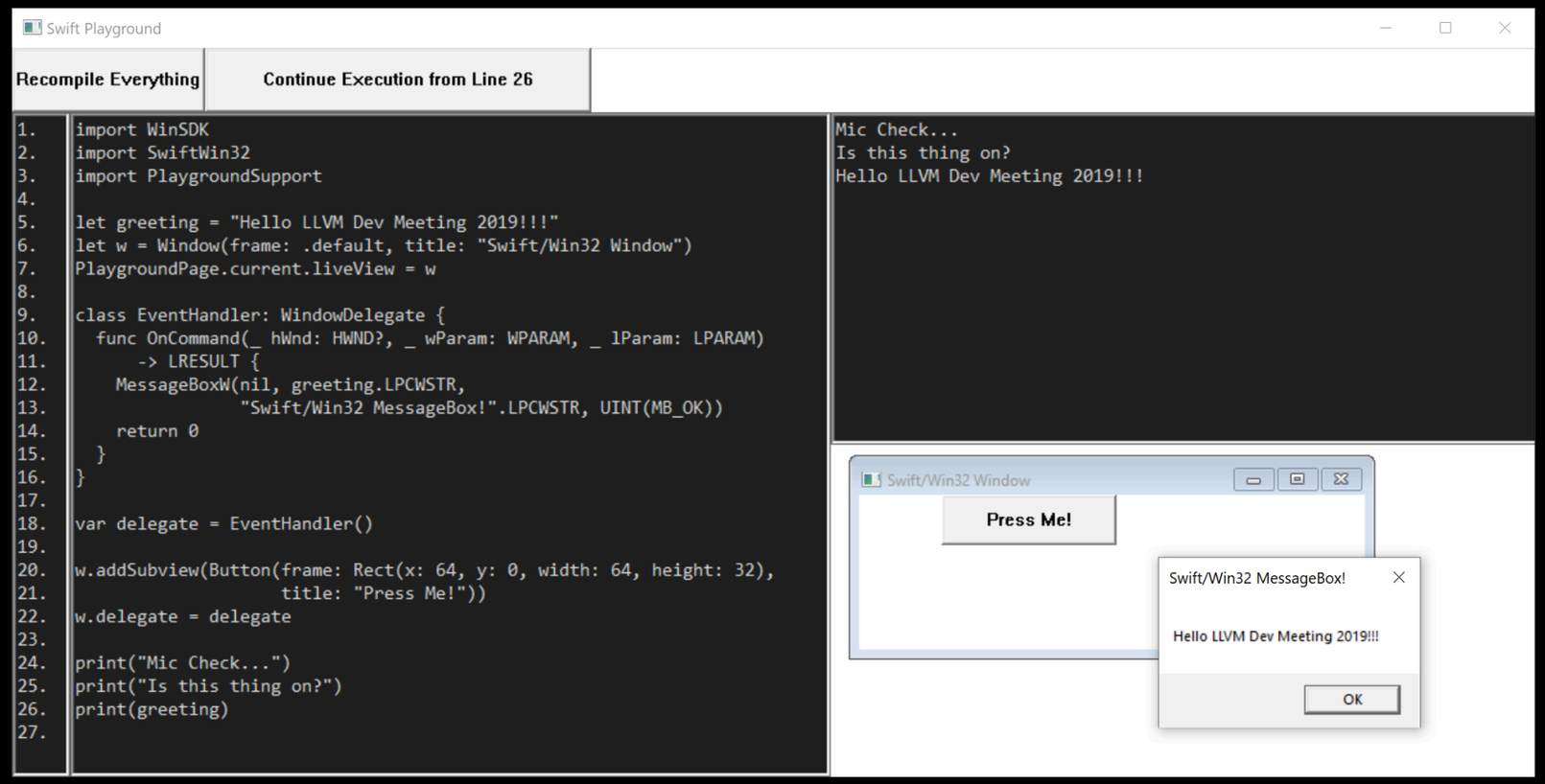
# Quickly Now

- Immediate Feedback
- REPL



# Quickly Now

- Immediate Feedback
- REPL
- Rapid Prototyping



# Future Work

# Future Work

- Simplifications to SDK

# Future Work

- Simplifications to SDK
- Improved debugging

# Future Work

- Simplifications to SDK
- Improved debugging
- Porting higher level frameworks (e.g. Swift-NIO, swift-log)

# Thanks

- Ted Kremenek, Michael Gottesman
- Jordan Rose
- John McCall, Doug Gregor, Slava Pestov, Arnold Schwaighofer
- Mike Ash, Andrew Trick
- Davide Italiano, Jonas Devlieghere
- Kim Topley, Pierre Habouzit
- Lily Vulcano, Gwynne Raskind
- Ankit Agarwal
- Mishal Shah
- The Swift community