Programming in Swift on IBM Z

—

Frank Langel
CEO
Scade

Igor Todorovski
Swift on z/OS Lead
IBM Canada

TechU

2018 IBM Systems Technical University

October 10, 2018

Hollywood, Florida

IBM

Please note

IBM’s statements regarding its plans, directions, and intent are subject to change or withdrawal without notice and at IBM’s sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.

The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user’s job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

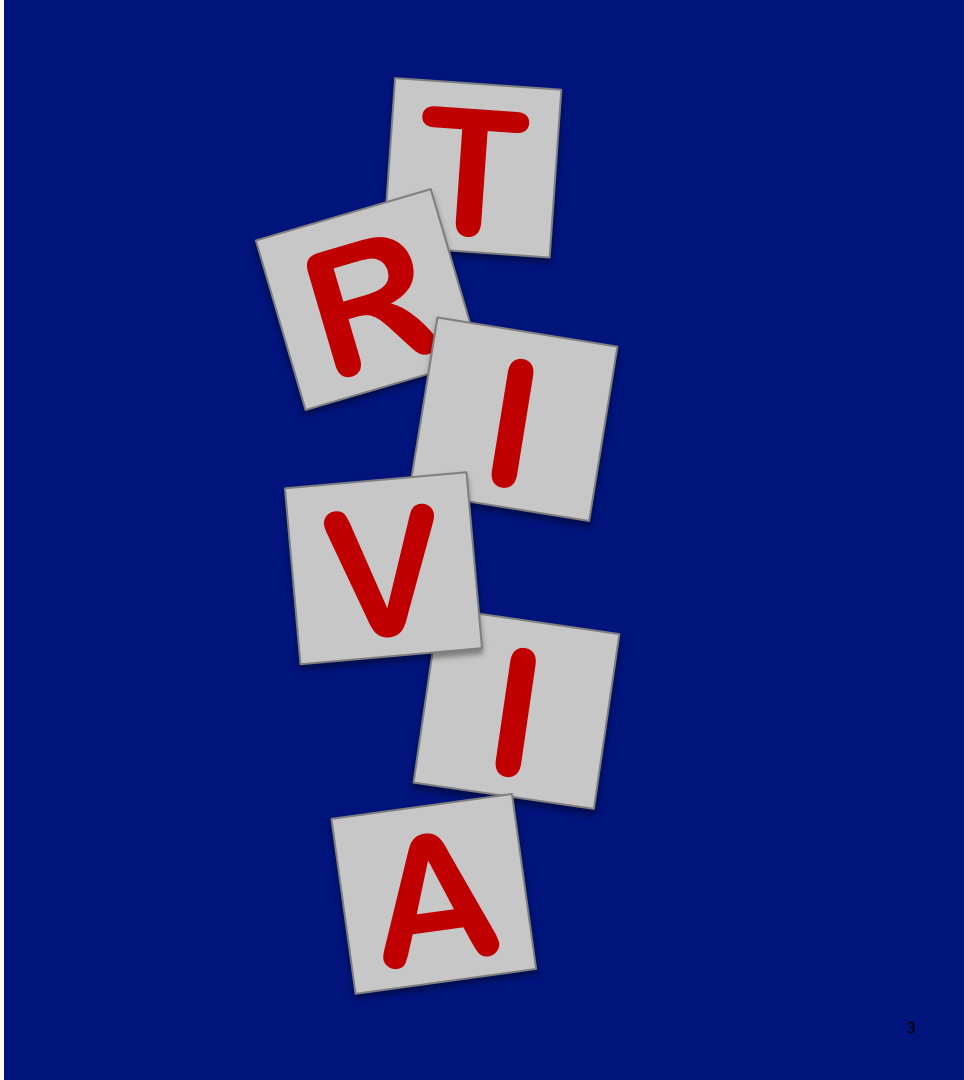
When were the following languages introduced to z Systems?

Assembler (early 1950s)

COBOL (1959)

PL/I (1964)

C/C++ (1972/1983)



Why do we need modern languages on IBM Z?

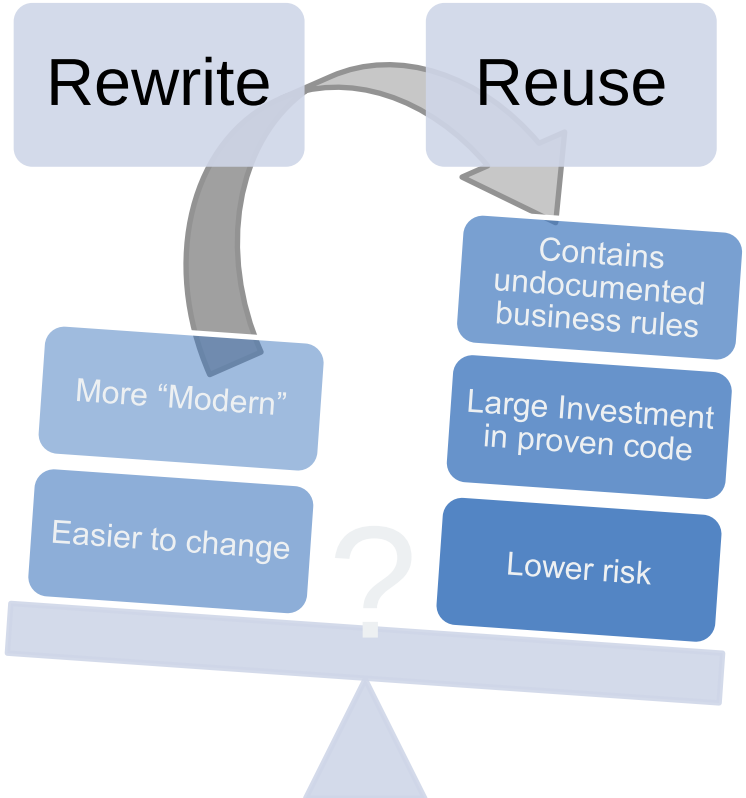
More than

14M

developers are using Java, JavaScript, and Swift, or NodeJS worldwide.

1 Skills: Millions of Available Developers

Remember: Your existing code is a valuable asset!

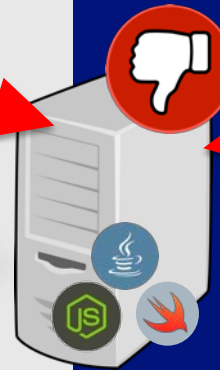


Get the best of the two strategies.

Why do we need modern languages on IBM Z?

2 Leverage best fit language for digital transformation

3 Put your back-end closer to your data



Swift Ecosystem

- ✓ Industry backing
- ✓ Developers acceptance
- ✓ Tools & packages
- ✓ Conferences & meetups

WWDC 2014

Apple releases a new programming language called "Swift"

June 2014

WWDC 2015

Apple announces to open source Swift

June 2015

"Swift is growing faster than anything else we track"

July 2015

RedMonk

Apple releases Swift as **Open Source**.

Nov. 2015

Dec. 3, 2015

Jan. 2016

RedMonk

IBM InterConnect 2016

IBM to bring Swift to the cloud to radically simplify end-to-end development of apps

Feb 2016

Programming Language Rankings *Swift "meteoric rise" continues. "Swift moves up one spot from #18 to #17 Y/Y…and is within hailing distance of our Tier 1 languages".*

try! Swift Tokyo

Mar. 2016

KITURA

stackoverflow Developer Survey Results 2016

Swift #2 "***Most Loved***"

Swift 4 is released

Sep, 2017

The Swift programming language

"Swift's ceiling is virtually unlimited"

June 2018

RedMonk

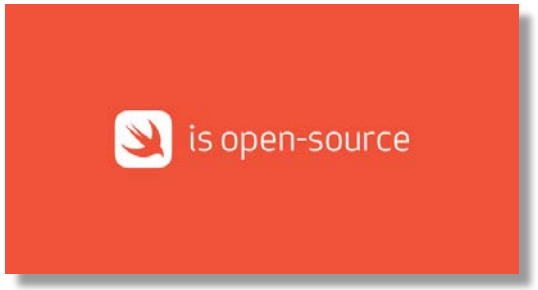
2014	1/2015	6/2015	1/2016	6/2016	1/2017	6/2017	01/2018
	1 JavaScript	1 JavaScript	1 JavaScript	1 JavaScript	1 JavaScript	1 JavaScript	1 JavaScript
	2 Java	2 Java	2 Java	2 Java	2 Java	2 Java	2 Java
	3 PHP	3 PHP	3 PHP	3 PHP	3 Python	3 Python	3 Python
	4 Python	4 Python	4 Python	4 Python	4 PHP	4 PHP	4 PHP
	5 C#	5 C#	5 C#	5 C#	5 C#	5 C#	5 C#
	5 C++	5 C++	5 C++	5 C++	5 C++	6 C++	6 C++
	5 Ruby	5 Ruby	5 Ruby	5 Ruby	7 CSS	7 CSS	7 CSS
	8 CSS	8 CSS	8 CSS	8 CSS	7 Ruby	8 Ruby	8 Ruby
	9 C	9 C	9 C	9 C	9 C	9 C	9 C
	10 Objective-C	10 Objective-C	10 Objective-C	10 Objective-C	10 Objective-C	10 Objective-C	**10 Swift**
	11 Perl	11 Perl	11 Shell	11 Shell	11 Scala	**11 Swift**	10 Objective-C
	11 Shell	11 Shell	12 Perl	12 R	11 Shell	12 Shell	12 Shell
	13 R	13 R	13 R	13 Perl	**11 Swift**	12 Scala	12 R
	14 Scala	14 Scala	14 Scala	14 Scala	14 R	14 R	14 TypeScript
	15 Haskell	15 Go	15 Go	15 Go			
	16 Matlab	15 Haskell	15 Haskell	16 Haskell			
	17 Go	17 Matlab	**17 Swift**	**17 Swift**			
	17 Visual Basic	**18 Swift**	18 Matlab				
	19 Clojure	19 Clojure	19 Clojure				
	19 Groovy	19 Groovy	19 Groovy				
		19 Visual Basic	19 Visual Basic				
	22 Swift						
46 Swift							

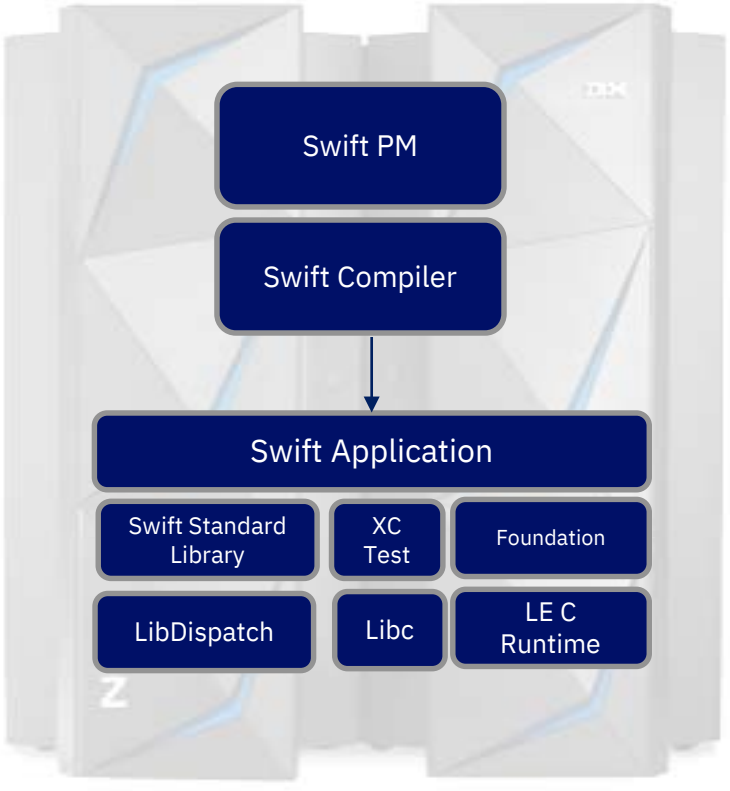
Swift is introduced

Swift became open source

What is Swift?

- Modern language developed by Apple Inc. in **2014**
- **Open sourced** in **2015**
- A client and server side programming language
  - Produces efficient natively compiled binaries (similar to COBOL, C, and PLI)
  - Community driven
    - IBM Swift Kitura Web Framework – Enables API Orchestration





Why Swift?

Safe

- Type Safe
- Memory Safe

Fast

- Natively compiled
- Optimized

Concise

- Type inference
- Little verbiage

Modern

- Concise
- Easy to learn

Community Driven

- Growing # of Libraries

Why Swift? It's *Safe!*

Ensures that **developers** *catch* **issues** *before* **users do!**

- Moves potential run-time errors to compile-time
- Catch errors fast and early!

Type safety with compile-time type inference

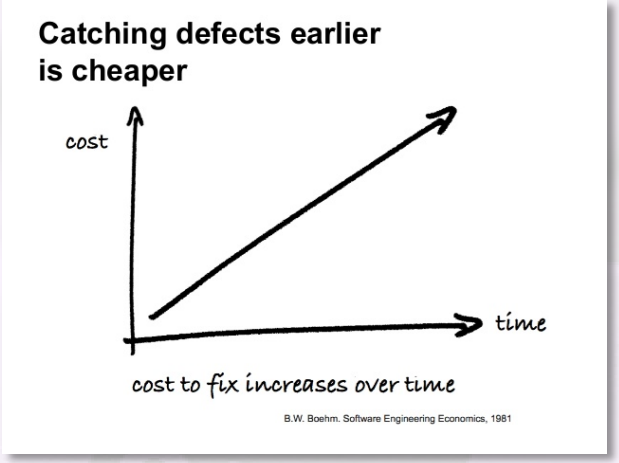
- Catch potential run-time errors at compile-time

Memory management

- Uses ARC (automatic reference counting)
- Variables and constants always **initialized** and array bounds are always **checked**.

Assignments return no value.

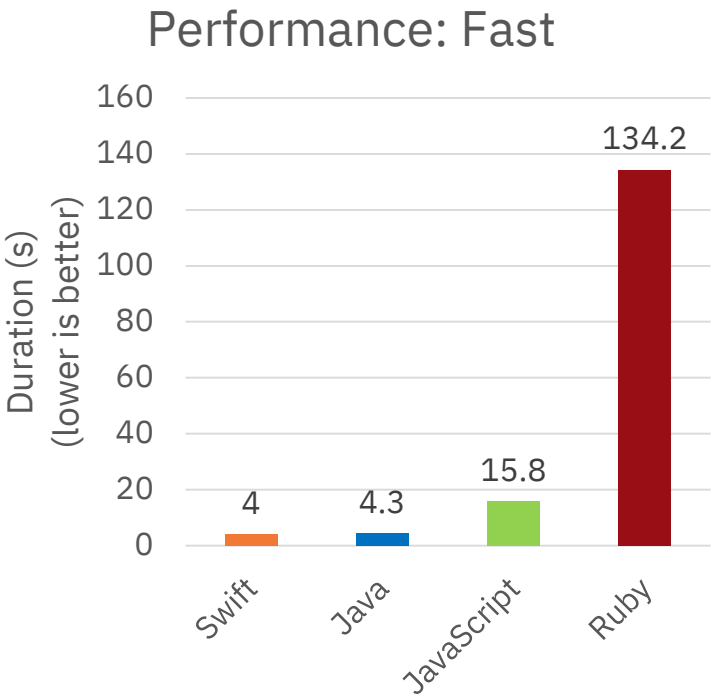
- Prevents error of writing i = 0 instead of i == 0

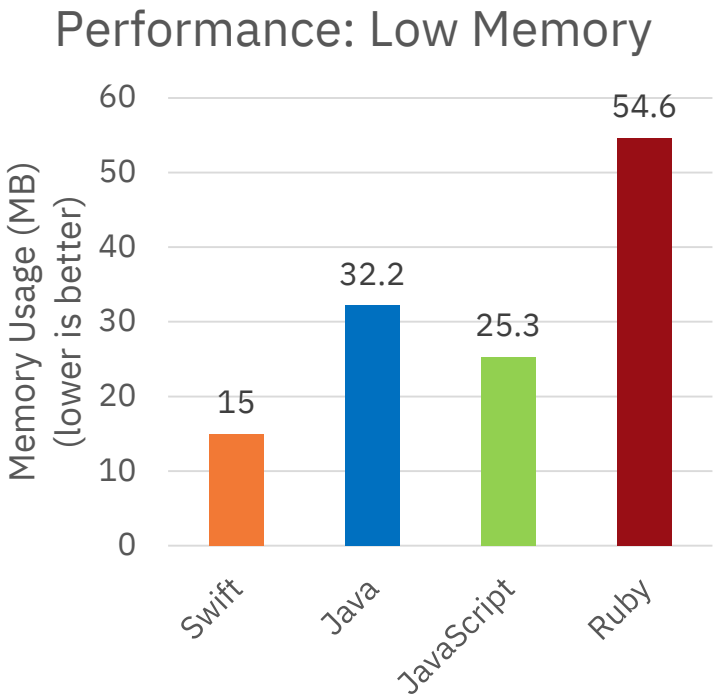



```
var pi = 3.14159
// variable pi is inferred to be of type Double at compile time

pi = "Swift on z/OS" // Error!
```

Why Swift? **Performance**





Source: benchmarksgame.alioth.debian.org/u64q/performance.php?test=spectralnorm

Why Swift? **Performance**

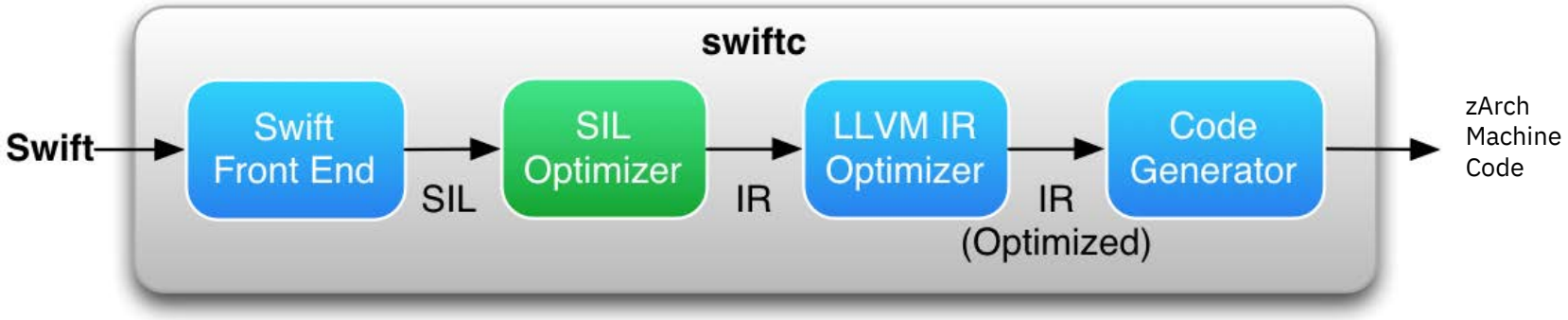
- Compiled to ***native*** code
  - Leverages LLVM Back-end and Optimizer
- Slightly better than Java in ***speed/memory***
- **Concurrency** built-in
- Automatic Code **Optimizer**
  - -O (inlining, loop optimizations, etc)
  - -whole-module-optimization
- Can *directly* call C Libraries via C interface

Invocation: swiftc –O main.swift –o a.out

Why Swift? **Performance**

Based on the well-established **LLVM framework**

- Actively developed compiler and toolchain, IBM/Google/Apple contributing to it
- Geared to work with LLDB
  - LLDB is a next generation, high-performance debugger
- Has a REPL (interactive compiler)



Why Swift? **Modern**

- Easy to learn
  - Concepts are similar in other popular languages (C, C++, Java)

- Concise and straightforward to read
  - Little verbiage
  - Similar syntax to Java and C++

Why Swift? It's *Modern!*

Easy to learn

– Concepts are similar to other popular languages (C, C++, Java)

– Little verbiage

No **semi-colons** needed!

#4 Most Loved Language (Stack Overflow, 2017)

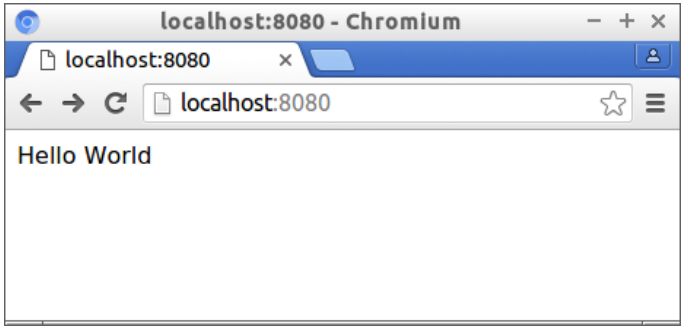
‘Hello World’ Web Server Program

```
import Kitura

let router = Router()

router.get("/") { request, response, next in
    response.send("Hello world")
    next()
}

Kitura.addHTTPServer(onPort: 8080, with: router)
Kitura.run()
```



Why Swift? **Modern**

Language Concepts

Unicode language (variable names and values)

Classes

Initializers, Deinitializers, Inheritance, Methods, Parameters , Setters/Getters

```
var 😀 , 🐶 , A : String
😀 = "Happy ";
🐶 = "Dog ";
A = 😀 + 😀 + 🐶;
print(A);
```

```
Happy Happy Dog
```

```
class Example {
    var a = 0
    var b: String

    init(a: Int) {    // Constructor
        self.a = a
        b = "name"    // An error if a declared property isn't initialized
    }
}
```

Why Swift? **Modern**

Language Concepts

Tuples

```
let http404Error = (404, “Not Found”)
```

Closure Expressions

```
reversedNames = names.sorted(by: { (s1: String, s2: String) -> Bool in
        return s1 > s2
})
```

Protocols (aka interfaces)

```
protocol SupportsToString {
func toString() -> String }
```

Operator Overloading

```
static func +(left: Vector, right: Vector) -> Vector
{ return [left.x + right.x, left.y + right.y, left.z + right.z] }
```

Generics

```
func addTwoValues<T>(_ a: inout T, _ b: inout T) { return a+b }
```

Why Swift? **Modern**

JSON serialization/ deserialization

Logging

Networking

File IO

Containers
Arrays, Sets, Dictionaries, etc

For..in loops

```
let numberOfLegs = ["spider": 8, "ant": 6, "cat": 4]
for (animalName, legCount) in numberOfLegs {
    print("\(animalName)s have \(legCount) legs")
}
```

IBM Toolkit for Swift – Linux on z Systems

- Core tools to develop in Swift:
  - Compiler
  - Swift Runtime
  - Libraries
  - Debugger (lldb)
  - Web framework (Kitura)
  - Package Manager

Community Edition
(free of charge)

Enterprise Edition
(License + S&S)

https://www.ibm.com/marketplace/swift-compiler

IBM Z | Swift@IBM

IBM Toolkit for Swift on z/OS

Community Edition

https://developer.ibm.com/mainframe/products/ibm-toolkit-swift-z-os/

Key features in Swift 4.0

- Swift compiler
- Standard Library
- Core Libraries
- Package Manager
- Sample Swift application based on Kitura

- Interoperability with C, PL/I, assembly, VSAM, and DB2.
- Free of charge

Sample Scenario:

Swift on the IBM HyperProtect Cloud

IBM HyperProtect Cloud

- ✓ Simplified, fast deployment and management of packaged solutions
- ✓ Based on Secure Service Containers (SSC)
- ✓ Only the data owner (client) controls access to data (no sysadmin access)
- ✓ IBM Z QoS reputation (reliability/resilience) combined with security & scaling

KITURA™

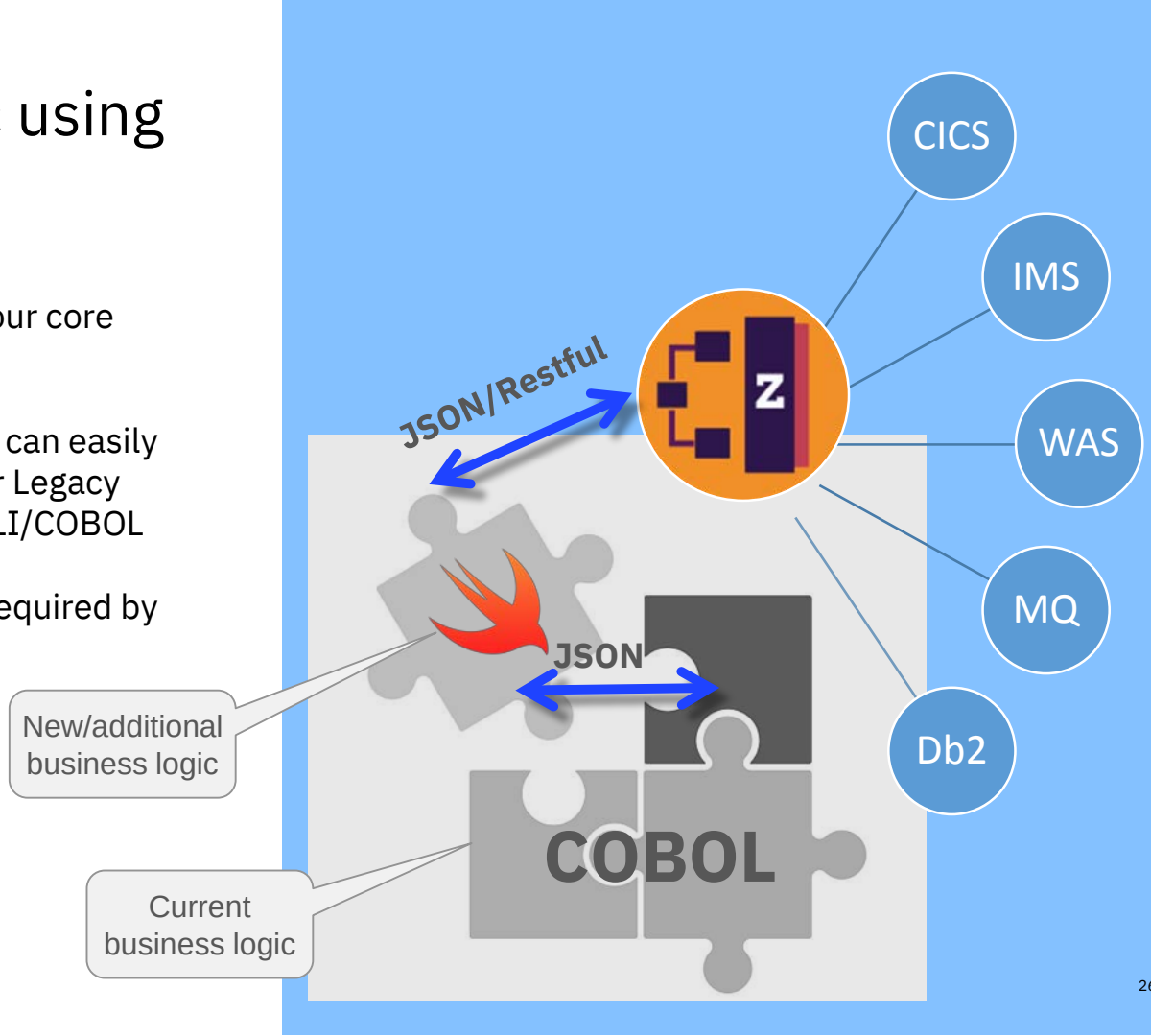
IBM Cloud HyperProtect Platform Services (Apple Starter Kit)

IBM Cloud HyperProtect Containers

Sample Scenario:

Extend business logic using Swift

- High performance language for your core business applications.
- Statically compiled language that can easily fit the current DevOps process for Legacy langauges such as ASM/C/C++/PLI/COBOL
- Conforms to tight audit controls required by many financial institutions.



Sample Scenario:

Call PL/I directly from Swift

Swift supports interlanguage calls to PL/I

Requirements:

1. PL/I procedures compiled as 64-bit (-qlp=64)
2. Swift Module Map to expose PL/I library
3. C bridging header to expose PL/I routines

The same scenario applies for C, C++, and PL/I

```
// Swift Program
import PLITest // Import module
writepair() // C PL/I routine
```

```
// Module Map
module PLITest [system]
{
  header ”interface.h" export *
}
```

```
// C Bridging header to expose PL/I functions
int writepair(void);
```

```
// PL/I procedure
write: procedure ext("writepair")
            Put List( 'Hello world' );
End write;
```

Sample Scenario:

End-to-end Swift application

Front-end

- ✓ Appealing animations and user feedback provide state of the art user experience
- ✓ Cross-platform (Andriod & iOS)

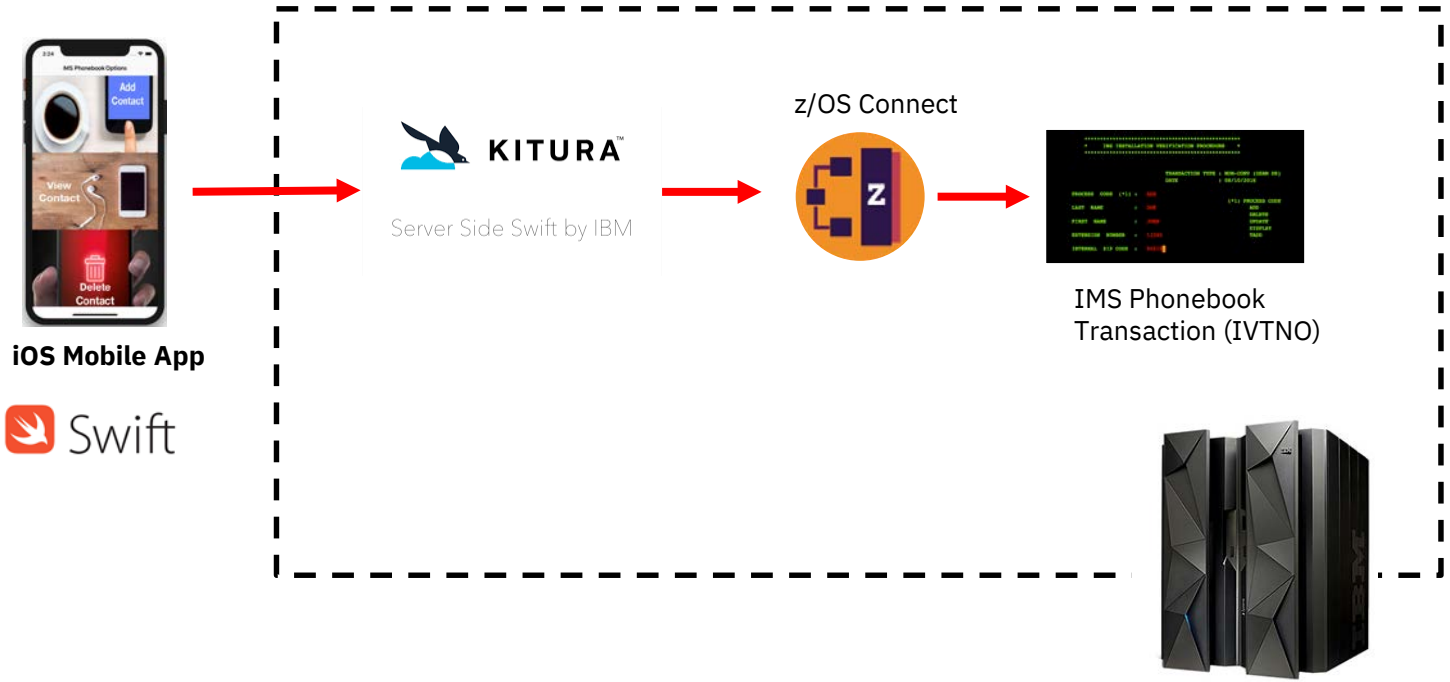
Back-end

- ✓ Reuse Swift code between front-end and back-end
- ✓ Easy access to current Z assets on the most secure platform
- ✓ Free open source web server from IBM (Kitura)

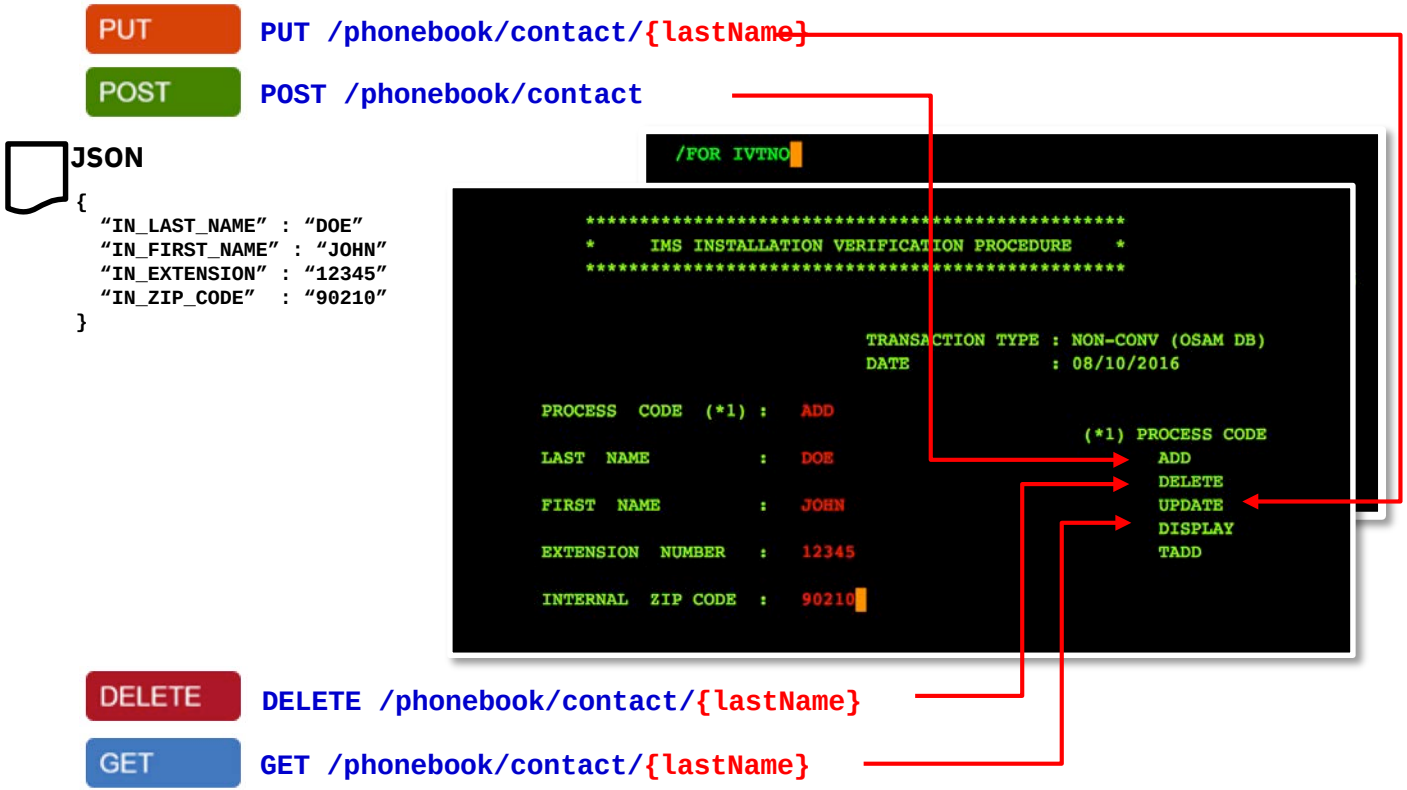
KITURA™

Db2

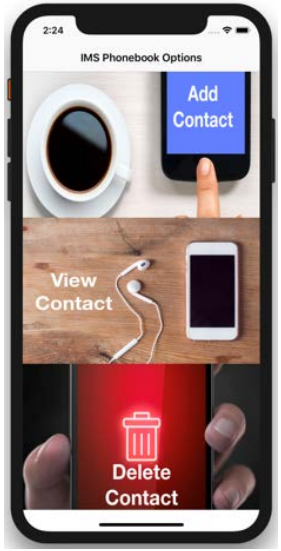
Using Swift to Write a Mobile App and Server Side Code



Using Sample IMS Phonebook Application (IVTNO)



Add Phone Contact Flow



z/OS Connect

KITURA™

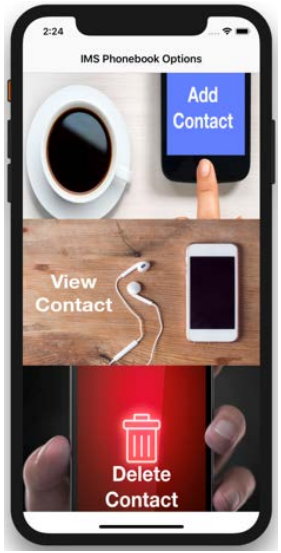
Server Side Swift by IBM

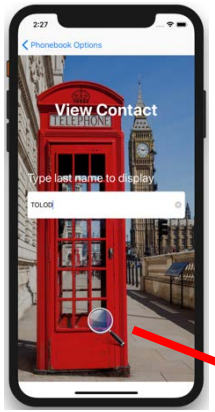
POST **POST /phonebook/contact**

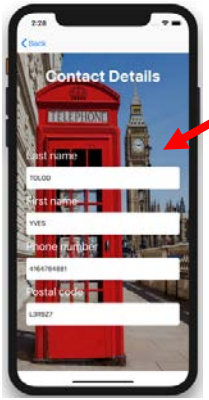
JSON

```
{
  “IN_LAST_NAME” : “DOE”
  “IN_FIRST_NAME” : “JOHN”
  “IN_EXTENSION” : “12345”
  “IN_ZIP_CODE”  : “90210”
}
```

View Phone Contact Flow







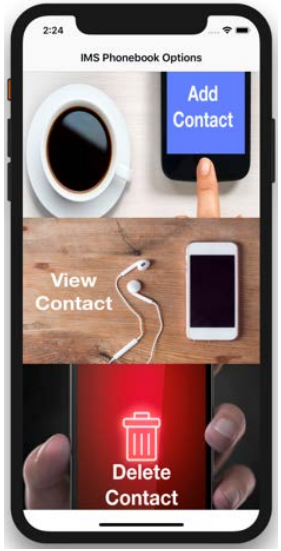
z/OS Connect

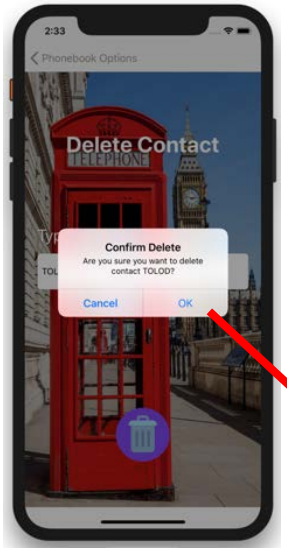
KITURA™

Server Side Swift by IBM

GET `GET /phonebook/contact/{lastName}`

Delete Phone Contact Flow





z/OS Connect


```
*************************************************
*     IMS INSTALLATION VERIFICATION PROCEDURE   *
*************************************************

                          TRANSACTION TYPE : NON-CONV (OSAM DB)
                          DATE             : 08/10/2016

PROCESS  CODE  (*1) :  ADD                    (*1) PROCESS CODE
LAST  NAME          :  DOE                         ADD
FIRST  NAME         :  JOHN                        DELETE
EXTENSION  NUMBER   :  12345                       UPDATE
INTERNAL  ZIP CODE  :  90210                       DISPLAY
                                                   TADD
```

DELETE `DELETE /phonebook/contact/{lastName}`

KITURA™

Server Side Swift by IBM

End-to-end Swift application development with SCADE

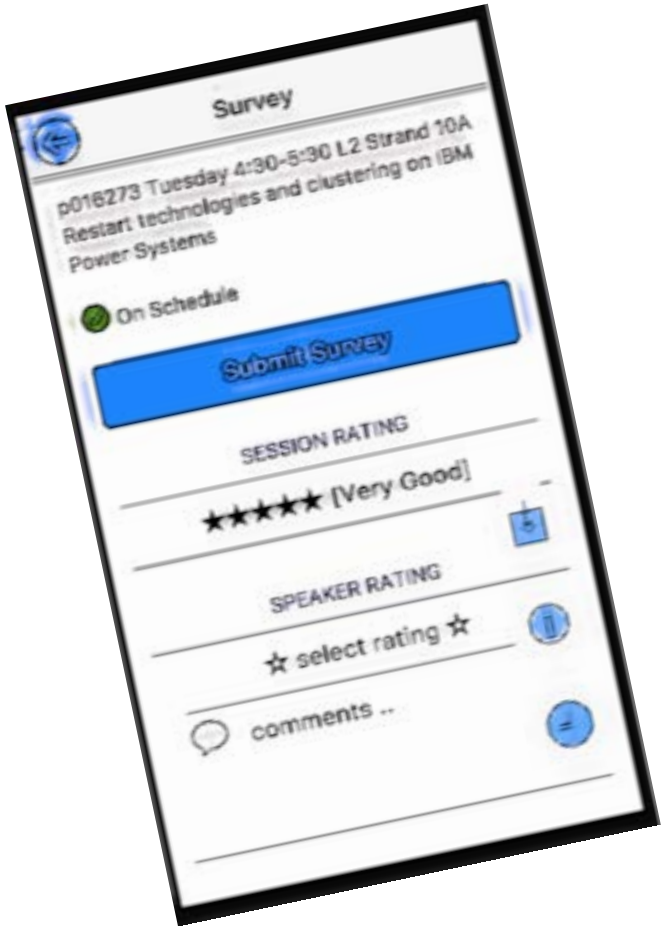
You can impact the future

- We are looking for **innovators and early adopters**
  - Validate user scenarios and get early access to the latest drivers.
  - If interested, please contact: **shereen@ca.ibm.com**

Resources

- Swift
  - IBM Marketplace (Swift on Linux on z): https://www.ibm.com/us-en/marketplace/swift-compiler
  - Swift @ IBM: https://developer.ibm.com/swift/
  - Extending Swift Value(s) to the Server (Free e-book): https://www-01.ibm.com/marketing/iwm/dre/signup?source=mrs-form-10468&S_PKG=ov55459
  - Free online course about server-side Swift: http://blog.udacity.com/2017/06/server-side-swift-with-ibm.html

Please complete the session survey!



Thank you

Frank Langel
CEO
www.scade.io
fll@scade.io

Igor Todorovski
Swift on z/OS, IBM Canada
shereen@ca.ibm.com

Notices and disclaimers

U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts. In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those

customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Notices and disclaimers continued

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

.