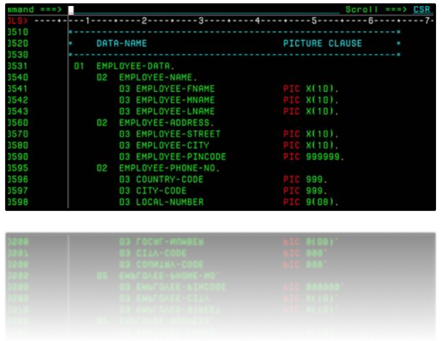
Modern Language:
A Pathway to Modernize the Enterprise
Extending IMS applications with REST APIs and Modern Language

GET /ibm/presenter
{
“name”: “Yves Tolod”
“title” : “z API and Modern Language on z/OS Specialist”
“email”: “yves.tolod@ca.ibm.com”
“phone”: “(416) 605-8936”
}

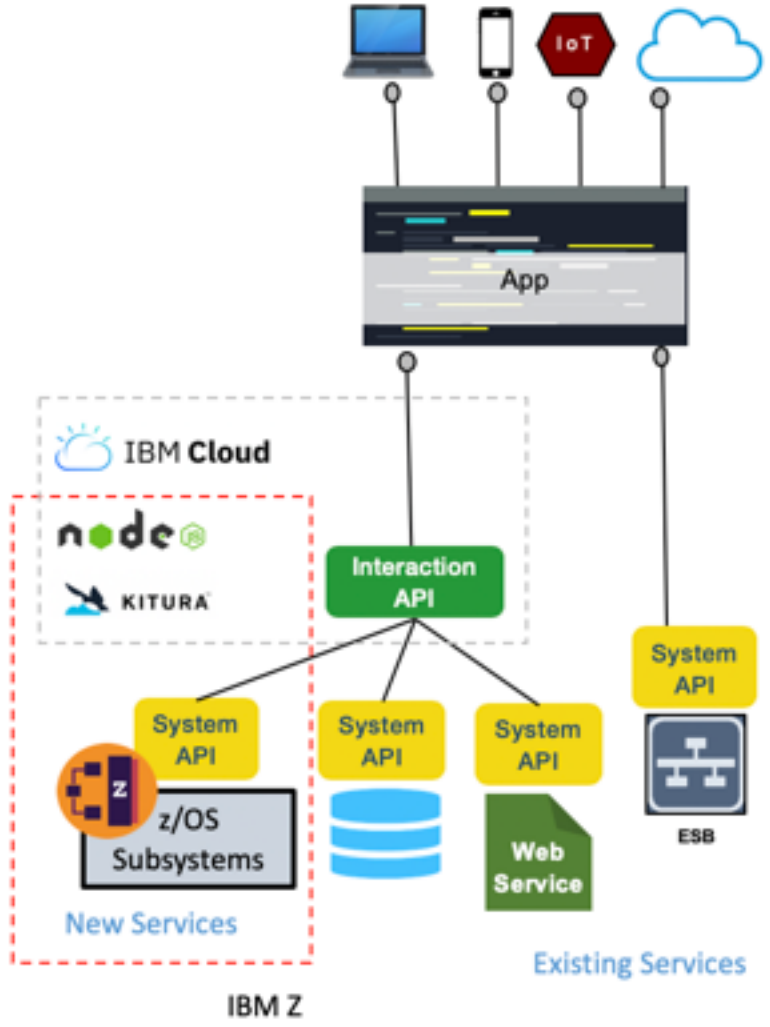


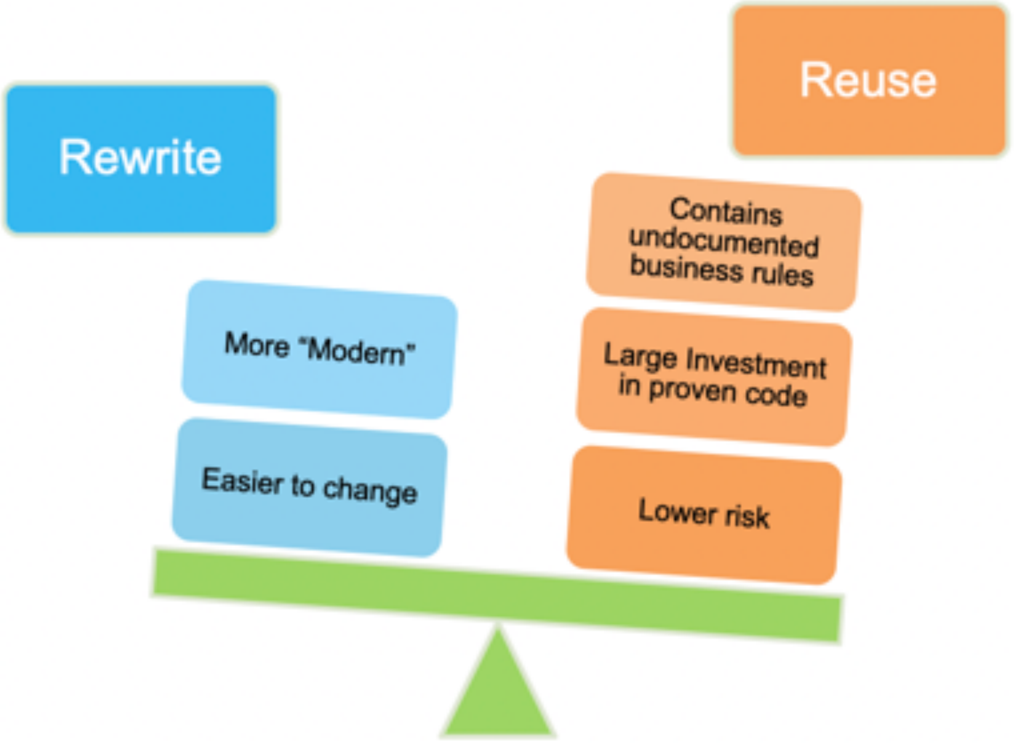
```
mmand ===>                                                    Scroll ===> CSR
LS> ----+----1----+----2----+----3----+----4----+----5----+----6----+----7-
510        *---------------------------------------------------------------*
520        *    DATA-NAME                           PICTURE CLAUSE        *
530        *---------------------------------------------------------------*
531         01  EMPLOYEE-DATA.
540             02  EMPLOYEE-NAME.
541                 03 EMPLOYEE-FNAME               PIC X(10).
542                 03 EMPLOYEE-MNAME               PIC X(10).
543                 03 EMPLOYEE-LNAME               PIC X(10).
560             02  EMPLOYEE-ADDRESS.
570                 03 EMPLOYEE-STREET              PIC X(10).
580                 03 EMPLOYEE-CITY                PIC X(10).
590                 03 EMPLOYEE-PINCODE             PIC 999999.
595             02  EMPLOYEE-PHONE-NO.
596                 03 COUNTRY-CODE                 PIC 999.
597                 03 CITY-CODE                    PIC 999.
598                 03 LOCAL-NUMBER                 PIC 9(08).
```

Modern Language

A Pathway to Modernize the Enterprise

Combine Modern Language with Existing Core Assets to Accelerate Digital Transformation




Rewrite
Reuse
More "Modern"
Easier to change
Contains undocumented business rules
Large Investment in proven code
Lower risk

Existing core assets on the host are built with a lot of due diligence and over a long period of time

Why do we need Modern Languages on IBM Z?

Skills: Millions of available developers

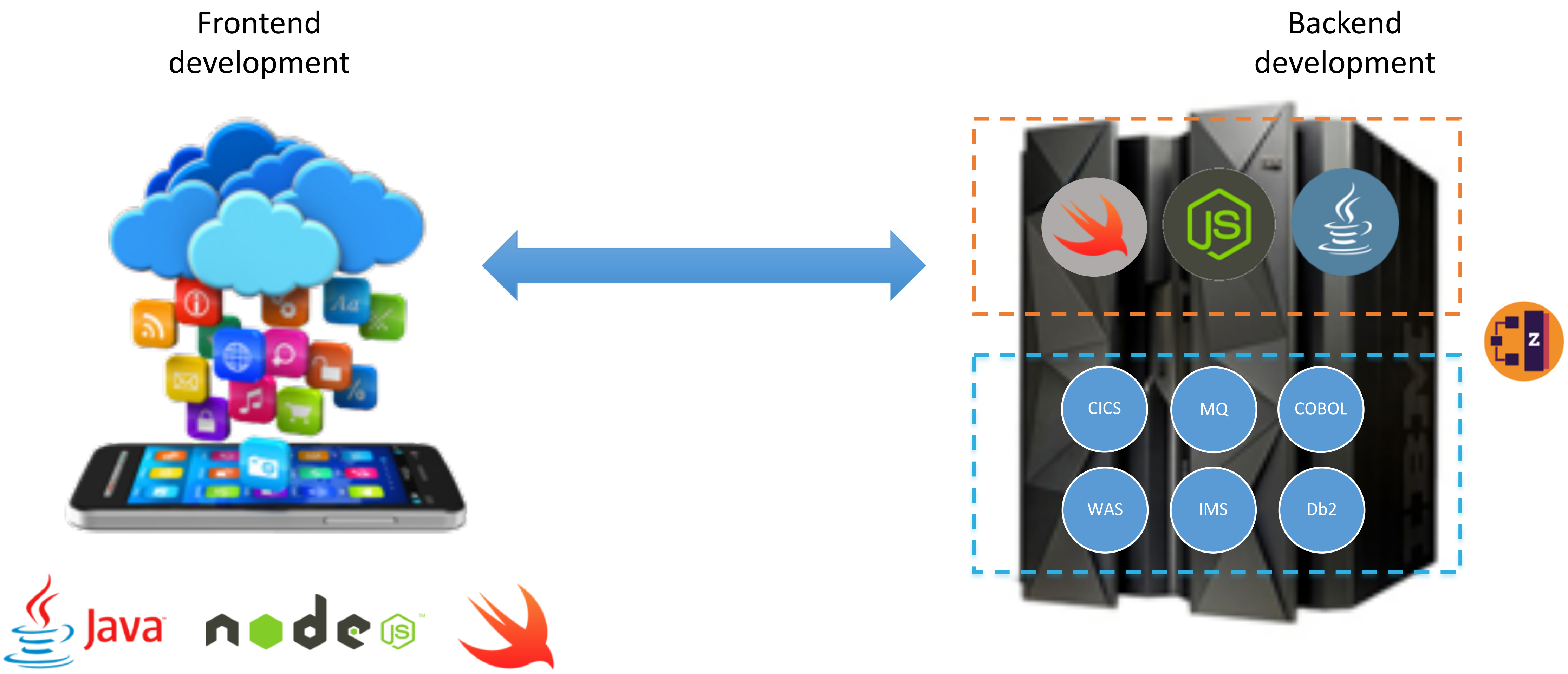
Frontend
Development

Full stack developer

Backend
Development

Why do we need Modern Languages on IBM Z?

Leverage best fit language for digital transformation



Frontend development

Backend development

CICS

MQ

COBOL

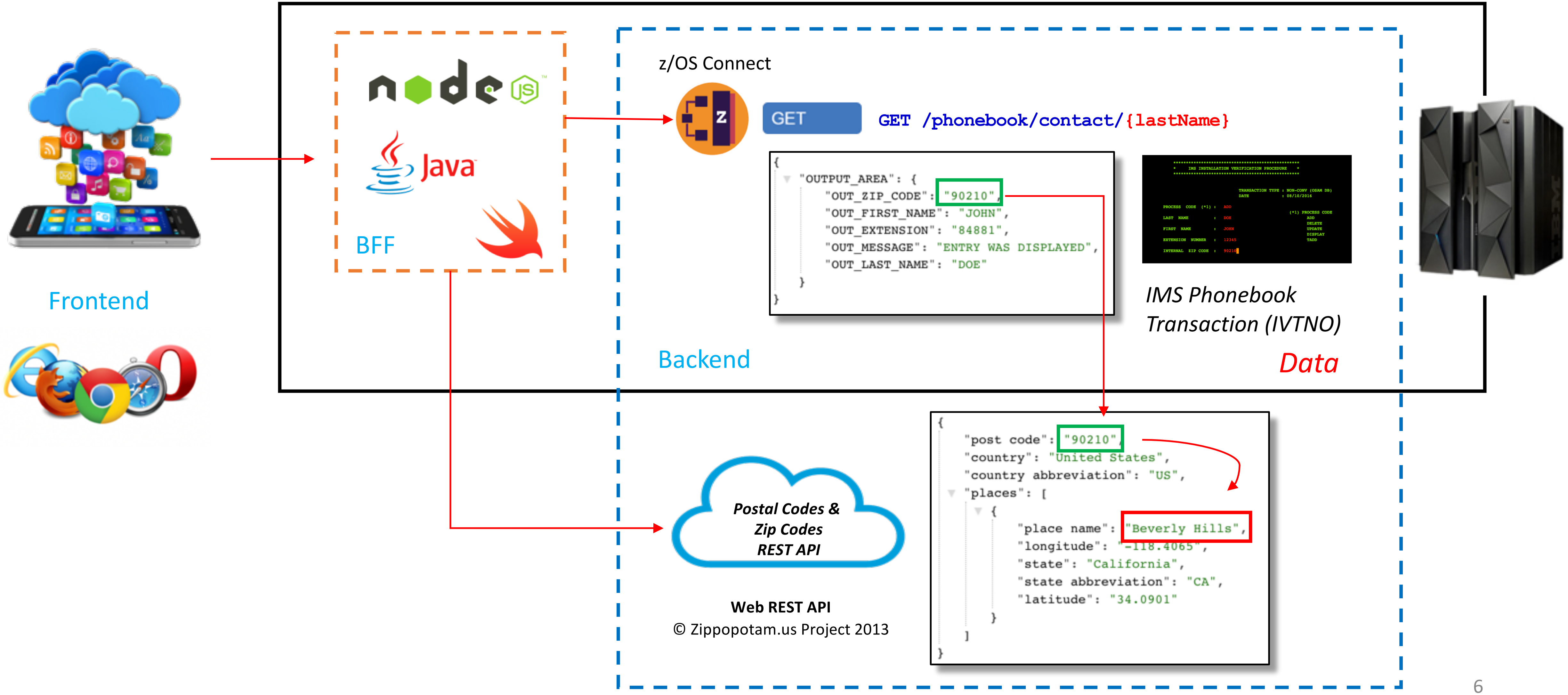
WAS

IMS

Db2

Why do we need Modern Languages on IBM Z?

Put your *Backend For Frontend* Closer to your data



`GET /phonebook/contact/{lastName}`

```
{
  "OUTPUT_AREA": {
    "OUT_ZIP_CODE": "90210",
    "OUT_FIRST_NAME": "JOHN",
    "OUT_EXTENSION": "84881",
    "OUT_MESSAGE": "ENTRY WAS DISPLAYED",
    "OUT_LAST_NAME": "DOE"
  }
}
```

```
  ***********************************************
  *    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
                                               DELETE
FIRST  NAME         :  JOHN                    UPDATE
                                               DISPLAY
EXTENSION  NUMBER   :  12345                   TADD

INTERNAL  ZIP CODE  :  90210
```

IMS Phonebook Transaction (IVTNO)

Postal Codes & Zip Codes REST API

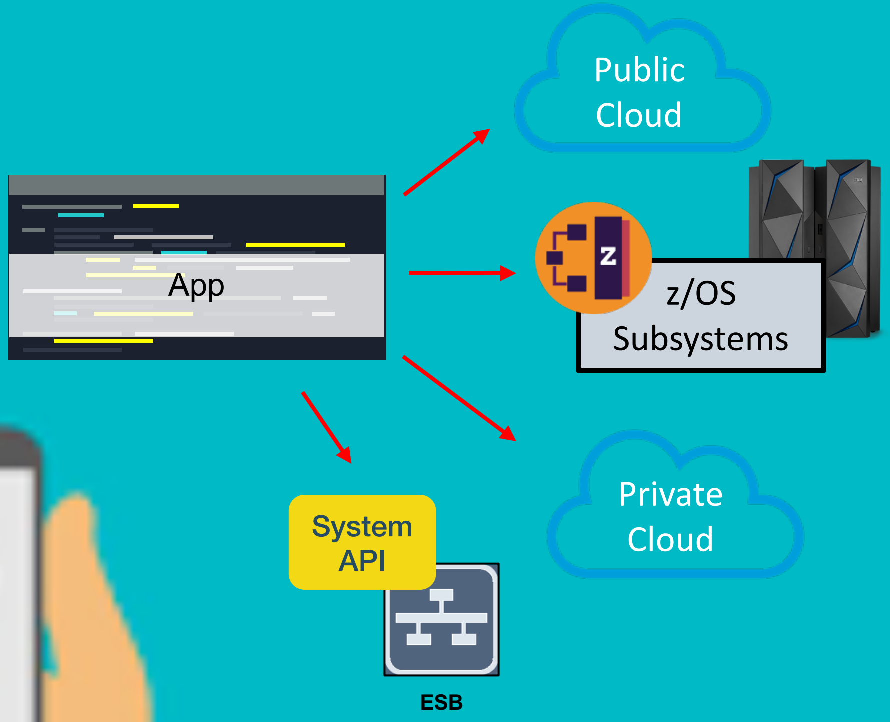
Web REST API

© Zippopotam.us Project 2013

```
{
  "post code": "90210",
  "country": "United States",
  "country abbreviation": "US",
  "places": [
    {
      "place name": "Beverly Hills",
      "longitude": "-118.4065",
      "state": "California",
      "state abbreviation": "CA",
      "latitude": "34.0901"
    }
  ]
}
```

Why Modernize with APIs?

APIs enable the future of collaboration and engagement



App

Public Cloud

z/OS Subsystems

Private Cloud

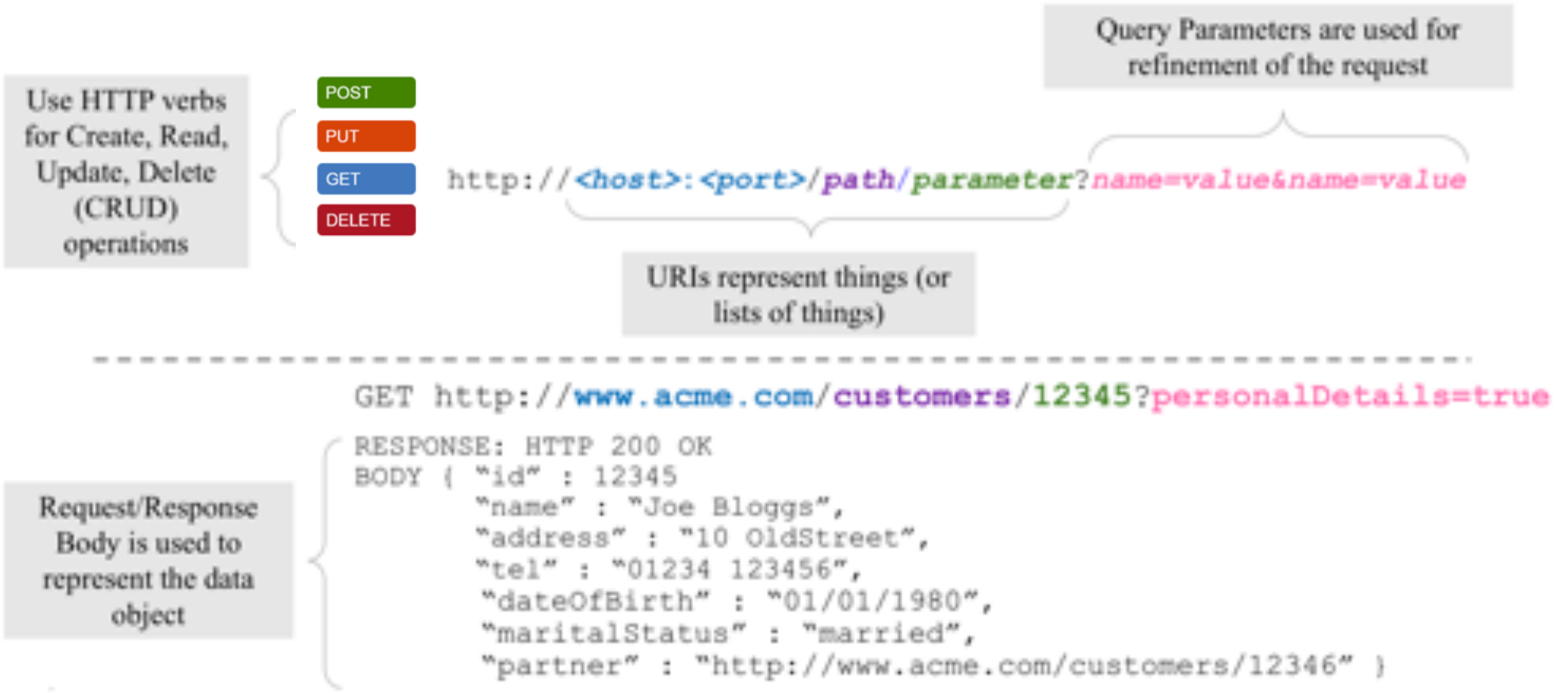
System API

ESB

Quick Overview
What is a REST API?

REST stands for **Re**presentational **S**tate **T**ransfer.

- *An architectural style for **accessing** and **updating** data.*
- *Typically using HTTP… but not all HTTP interfaces are “RESTful”.*
- *Simple and intuitive for the end consumer (the developer).*



Query Parameters are used for refinement of the request

Use HTTP verbs for Create, Read, Update, Delete (CRUD) operations

```
http://<host>:<port>/path/parameter?name=value&name=value
```

URIs represent things (or lists of things)

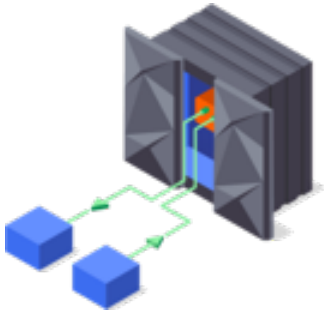
Request/Response Body is used to represent the data object

```
GET http://www.acme.com/customers/12345?personalDetails=true

RESPONSE: HTTP 200 OK
BODY { "id" : 12345
       "name" : "Joe Bloggs",
       "address" : "10 OldStreet",
       "tel" : "01234 123456",
       "dateOfBirth" : "01/01/1980",
       "maritalStatus" : "married",
       "partner" : "http://www.acme.com/customers/12346" }
```

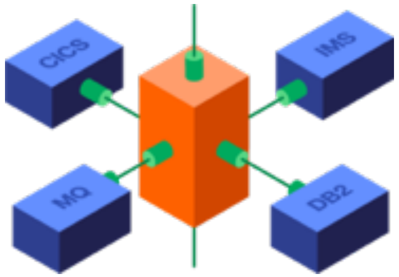
z/OS Connect Enterprise Edition

How do I expose my z/OS Assets as REST APIs?

z/OS Connect Enterprise Edition is IBM's Strategic solution for enabling natural REST APIs for z/OS assets in a unified manner across different subsystems with integrated auditing, security and scalability... **without writing z/OS application code.**



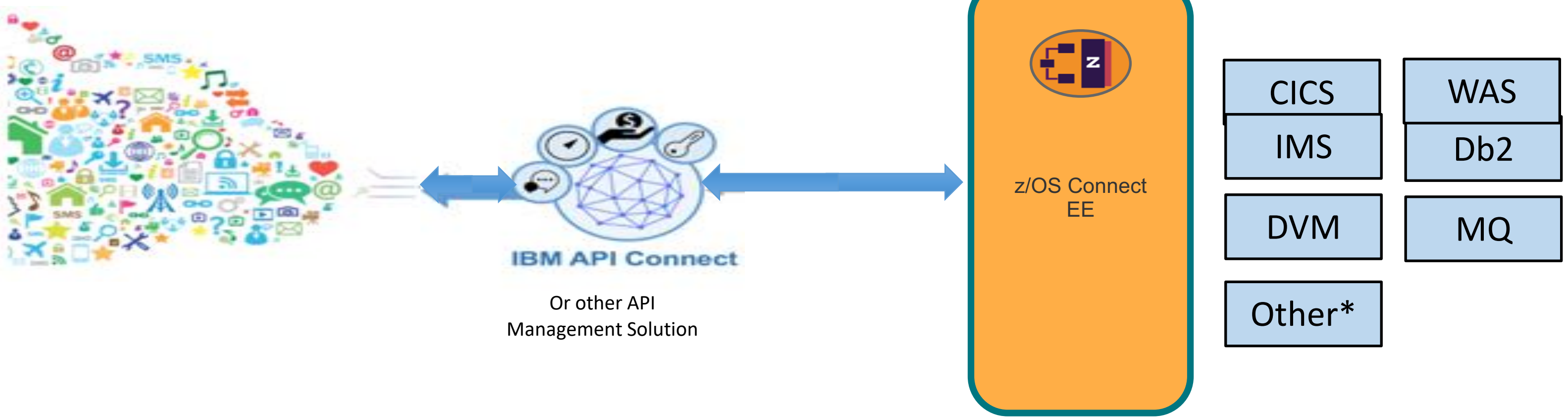
APIs **to** and **from** the mainframe



Comprehensive subsystem support and **unified** tooling

Point-and-click API creation

Z applications can now easily participate in the API economy

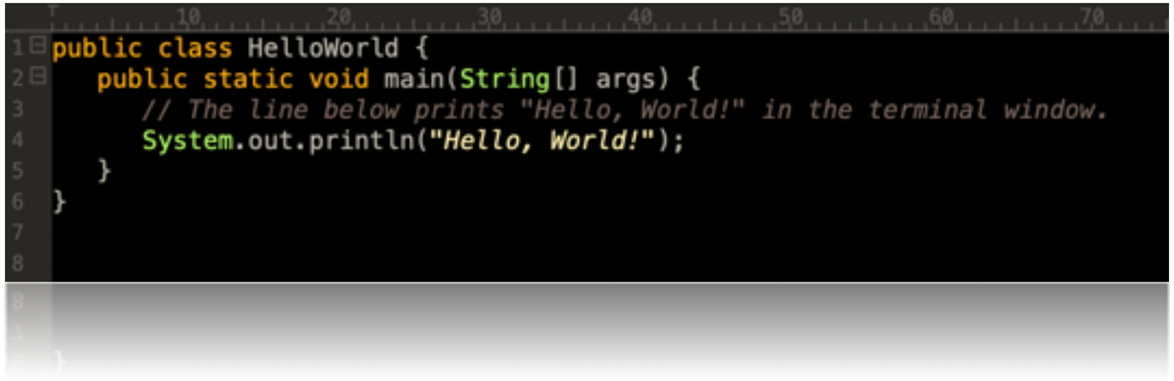


* Other Vendors or your own implementation

Quick Overview

What is Java?

- **Object-oriented** programming language designed to produce programs that will run on any computer system (supports the logic ***"Write once, run anywhere"***)
- Consists of the Java Development Kit (SDK), Java Virtual Machine (JVM) and the Java Runtime Environment (JRE)
- Java workload on z/OS can be offloaded to specialty processor **(zIIP engine)** which do not count towards Monthly License Charges (MLC)



```java
public class HelloWorld {
   public static void main(String[] args) {
      // The line below prints "Hello, World!" in the terminal window.
      System.out.println("Hello, World!");
   }
}
```

Compile: javac HelloWorld.java

Execute: java HelloWorld

Output: Hello, World!

Java Execution Environments and Interoperability
Capitalize on pre-existing assets, artifacts, platform strengths

IBM Java Execution Offerings

Transactional / Interactive

- WebSphere Application Server for z/OS (WAS z/OS and Liberty)
- CICS Transaction Server (JCICS and Liberty)
- *IMS (Java Message Processing regions)*
- *IMS Java*
- DB2 Stored Procedures

Batch oriented

- Compute Grid (WAS-z/OS Java batch)
- WebSphere Liberty Java batch (JSR 352)
- *IMS Java Batch regions (JMP)*
- JZOS component of z/OS SDK

- **Open Source or non-IBM vendor Application Server and Frameworks**
  - Tomcat, JBoss
  - iBatis, Hibernate, Spring
  - Ant

- **COBOL / Native Interoperability**
  - *COBOL Invoke maps to JNI*
  - IDz and JZOS have tooling to map COBOL copy books to Java classes
  - JCICS
  - *IMS Java, JMP/JBP*
  - WAS CG, WOLA
  - etc

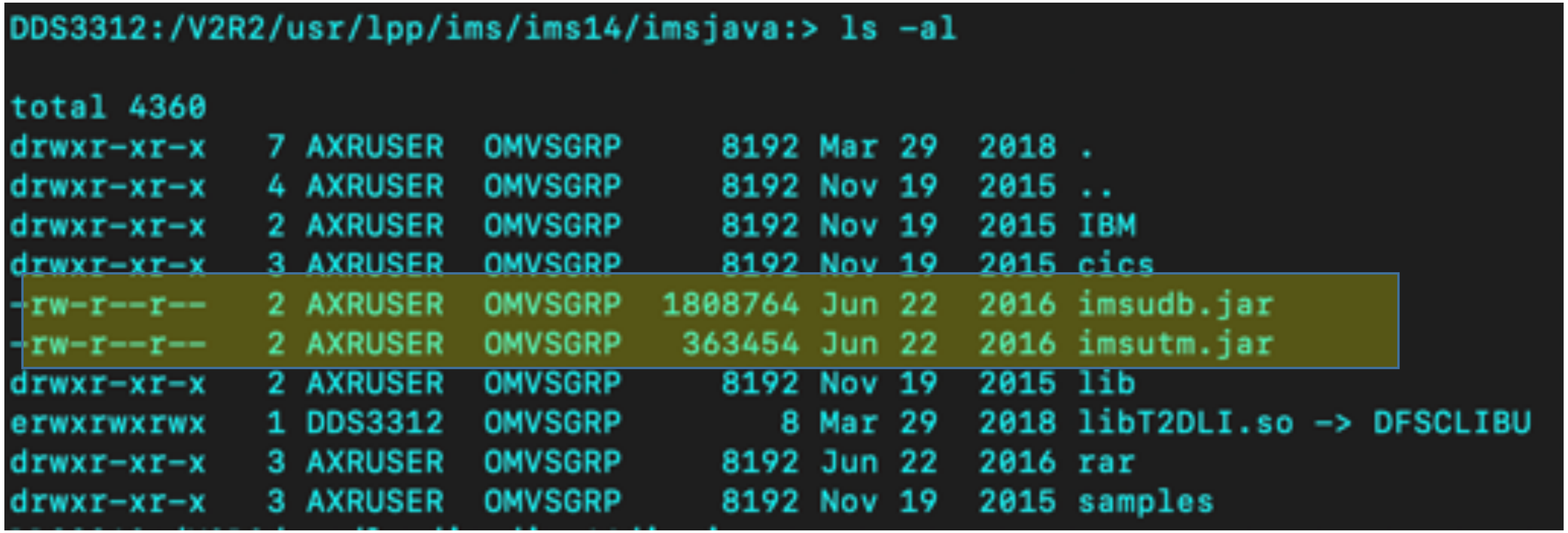
Java in IMS

Setting up Java Environment in IMS

- **Step 1.** Install the IMS Java dependent region resource adapter (**`imsutm.jar`**) and IMS universal drivers (**`imsudb.jar`**) provided through the *Java On Demand Feature* FMID.

ALEX

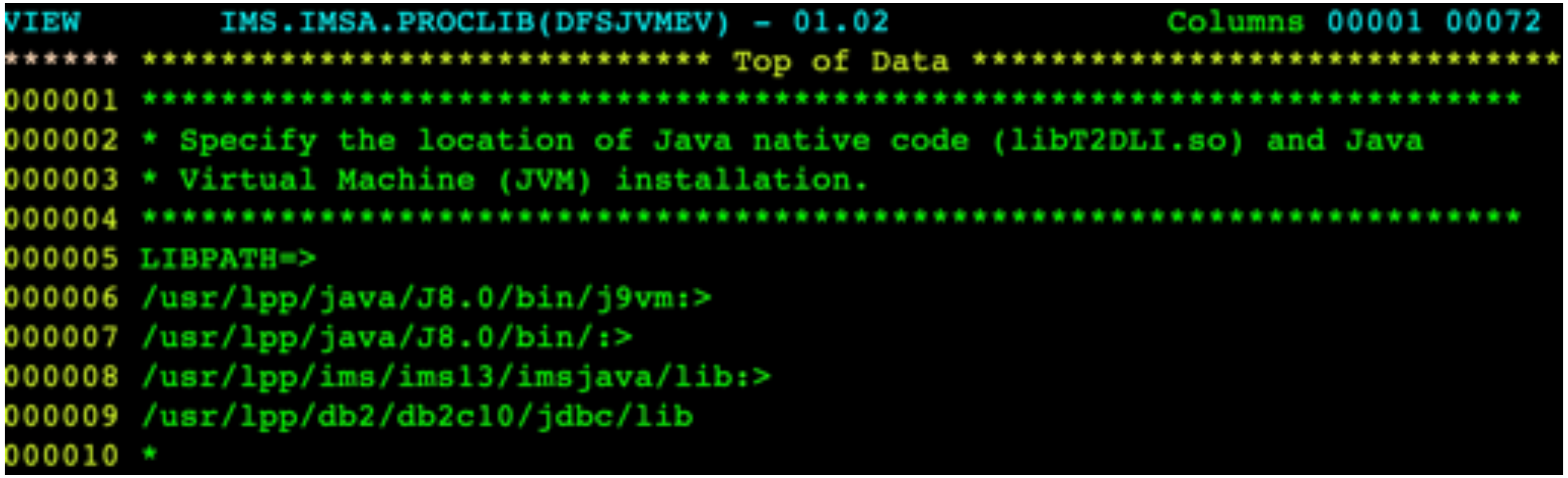
SYSTEMS
PROGRAMMER



```
DDS3312:/V2R2/usr/lpp/ims/ims14/imsjava:> ls -al

total 4360
drwxr-xr-x    7 AXRUSER   OMVSGRP      8192 Mar 29   2018 .
drwxr-xr-x    4 AXRUSER   OMVSGRP      8192 Nov 19   2015 ..
drwxr-xr-x    2 AXRUSER   OMVSGRP      8192 Nov 19   2015 IBM
drwxr-xr-x    3 AXRUSER   OMVSGRP      8192 Nov 19   2015 cics
-rw-r--r--    2 AXRUSER   OMVSGRP   1808764 Jun 22   2016 imsudb.jar
-rw-r--r--    2 AXRUSER   OMVSGRP    363454 Jun 22   2016 imsutm.jar
drwxr-xr-x    2 AXRUSER   OMVSGRP      8192 Nov 19   2015 lib
erwxrwxrwx    1 DDS3312   OMVSGRP         8 Mar 29   2018 libT2DLI.so -> DFSCLIBU
drwxr-xr-x    3 AXRUSER   OMVSGRP      8192 Jun 22   2016 rar
drwxr-xr-x    3 AXRUSER   OMVSGRP      8192 Nov 19   2015 samples
```

Java in IMS
Setting up Java Environment in IMS

- **Step 2.** Setup the IMS Java Environment (**`DFSJVMEV`** member in IMS PROCLIB) and Java native code (`libT2DLI.so`).



```
VIEW       IMS.IMSA.PROCLIB(DFSJVMEV) - 01.02              Columns 00001 00072
****** ***************************** Top of Data ******************************
000001 **********************************************************************
000002 * Specify the location of Java native code (libT2DLI.so) and Java
000003 * Virtual Machine (JVM) installation.
000004 **********************************************************************
000005 LIBPATH=>
000006 /usr/lpp/java/J8.0/bin/j9vm:>
000007 /usr/lpp/java/J8.0/bin/:>
000008 /usr/lpp/ims/ims13/imsjava/lib:>
000009 /usr/lpp/db2/db2c10/jdbc/lib
000010 *
```

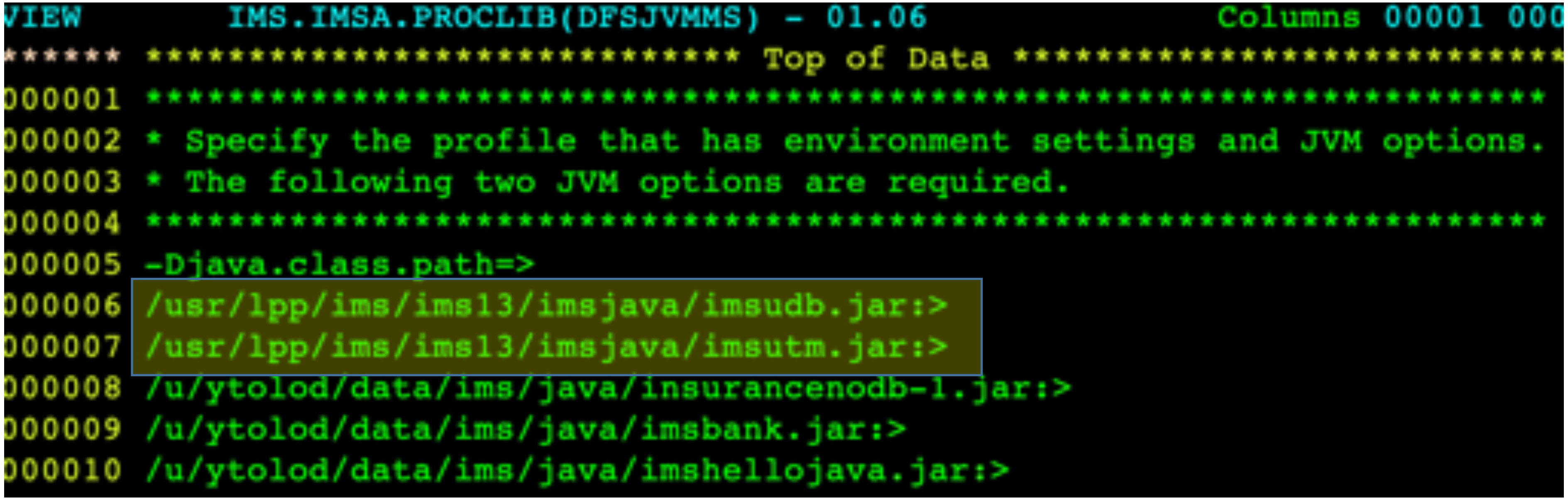
ALEX

SYSTEMS
PROGRAMMER

Java in IMS

Setting up Java Environment in IMS

- **Step 3.** Setup the Java `CLASSPATH` to specify the location of the `.jar` files for the IMS Java region resource adapter, IMS Universal drivers, and the Java applications.



```
VIEW       IMS.IMSA.PROCLIB(DFSJVMMS) - 01.06             Columns 00001 000
****** ***************************** Top of Data ******************************
000001 ***********************************************************************
000002 * Specify the profile that has environment settings and JVM options.
000003 * The following two JVM options are required.
000004 ***********************************************************************
000005 -Djava.class.path=>
000006 /usr/lpp/ims/ims13/imsjava/imsudb.jar:>
000007 /usr/lpp/ims/ims13/imsjava/imsutm.jar:>
000008 /u/ytolod/data/ims/java/insurancenodb-1.jar:>
000009 /u/ytolod/data/ims/java/imsbank.jar:>
000010 /u/ytolod/data/ims/java/imshellojava.jar:>
```

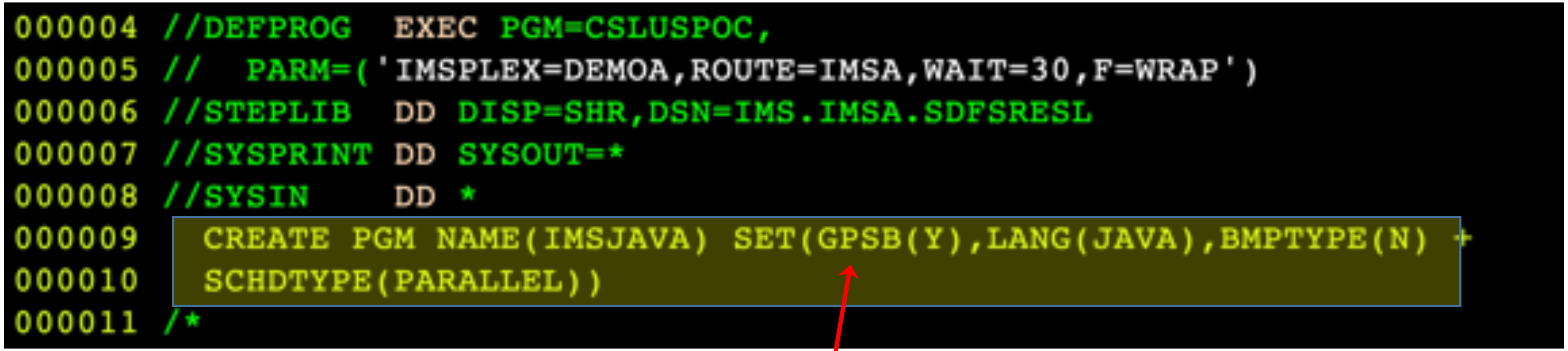
ALEX

SYSTEMS
PROGRAMMER

Java in IMS

Setting up Java program in IMS

- **Step 4.** Create the IMS program to make the PSB available online



```
//DEFPROG  EXEC PGM=CSLUSPOC,
//  PARM=('IMSPLEX=DEMOA,ROUTE=IMSA,WAIT=30,F=WRAP')
//STEPLIB  DD DISP=SHR,DSN=IMS.IMSA.SDFSRESL
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  CREATE PGM NAME(IMSJAVA) SET(GPSB(Y),LANG(JAVA),BMPTYPE(N) +
  SCHDTYPE(PARALLEL))
/*
```

Generated PSB (GPSB) – does not require PSBGEN or ACBGEN

ALEX

SYSTEMS
PROGRAMMER

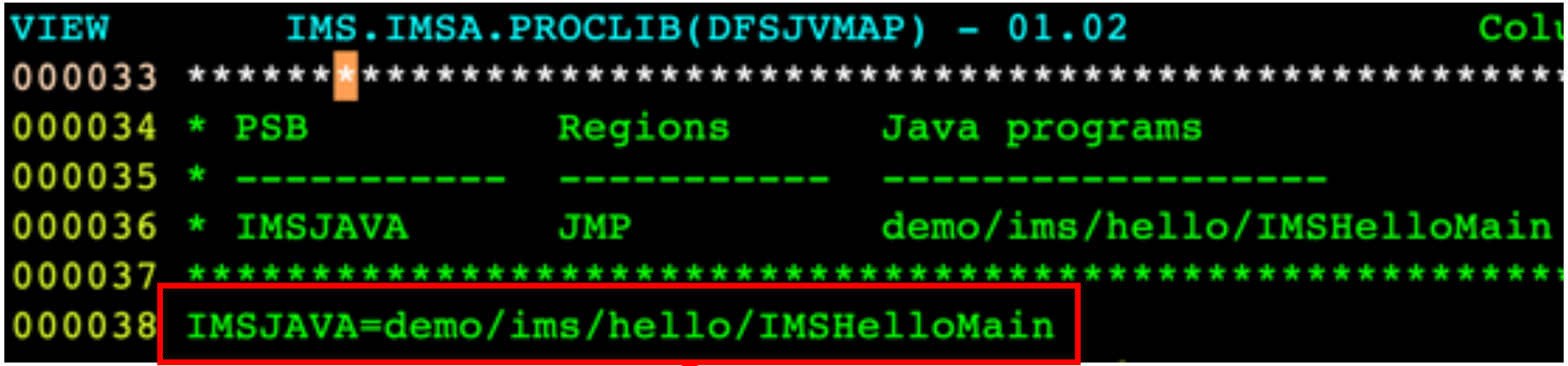
Java in IMS

Setting up Java program in IMS

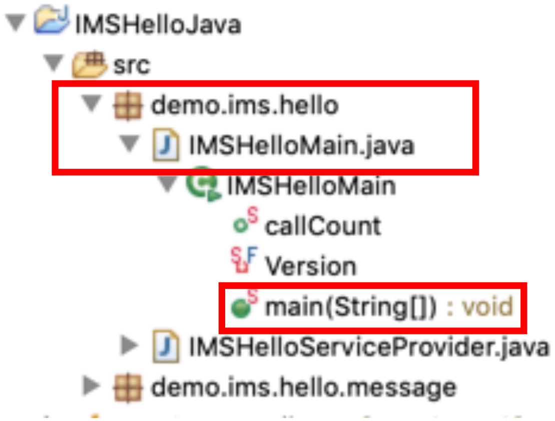
- **Step 5.** Map the IMS program (PSB name) to the Main Java class (**`DFSJVMAP`** member in IMS PROCLIB)

ALEX

SYSTEMS PROGRAMMER



```
VIEW       IMS.IMSA.PROCLIB(DFSJVMAP) - 01.02          Colu
000033 **********************************************************
000034 * PSB          Regions      Java programs
000035 * -----------  -----------  -------------------
000036 * IMSJAVA      JMP          demo/ims/hello/IMSHelloMain
000037 **********************************************************
000038 IMSJAVA=demo/ims/hello/IMSHelloMain
```

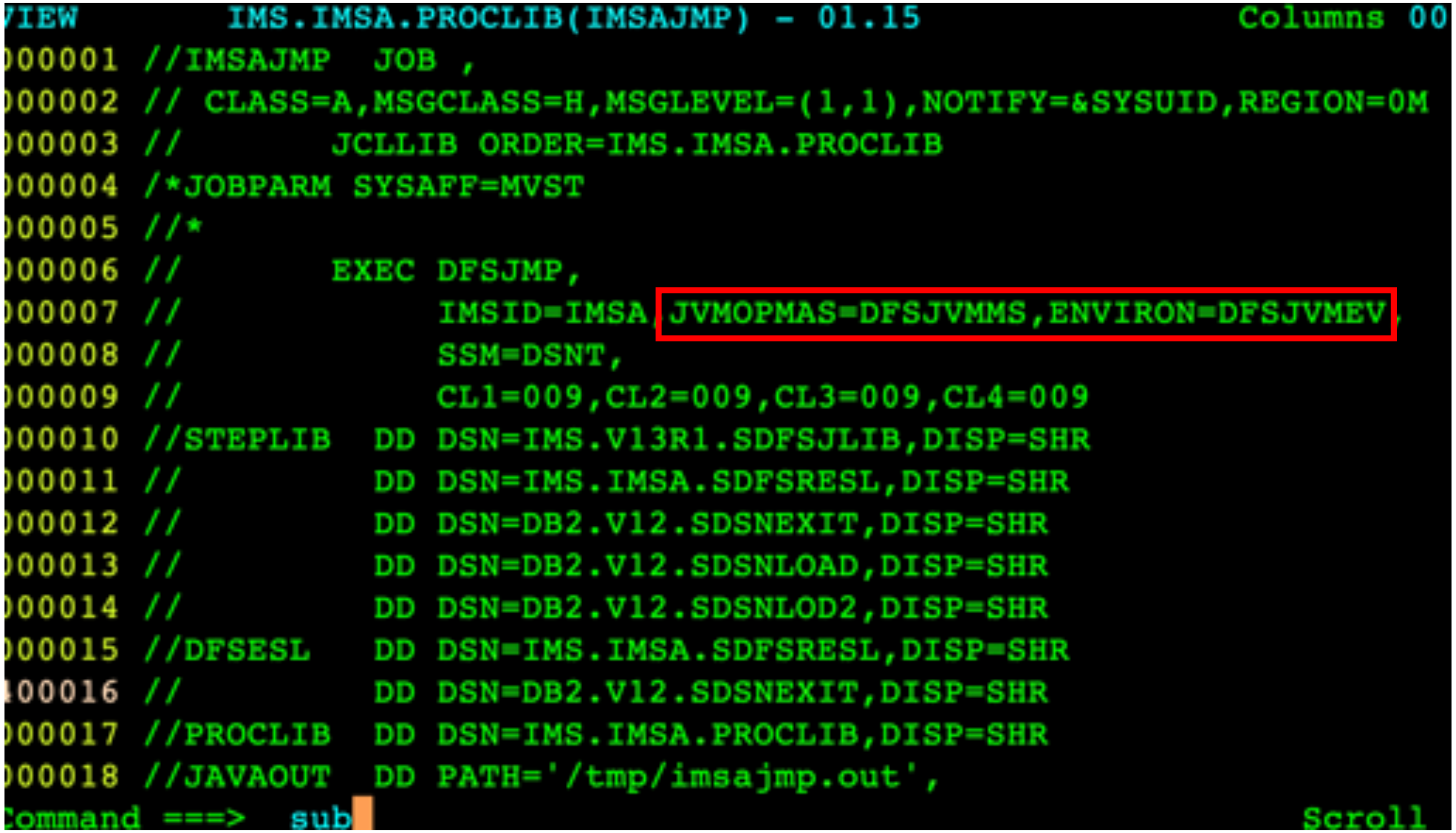


- IMSHelloJava
  - src
    - demo.ims.hello
      - IMSHelloMain.java
        - IMSHelloMain
          - callCount
          - Version
          - main(String[]) : void
      - IMSHelloServiceProvider.java
    - demo.ims.hello.message

Java in IMS

Setting up Java program in IMS

- **Step 6.** Customize the IMS JMP procedure and start the JMP region



```
VIEW        IMS.IMSA.PROCLIB(IMSAJMP) - 01.15              Columns 00
//IMSAJMP  JOB ,
// CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),NOTIFY=&SYSUID,REGION=0M
//         JCLLIB ORDER=IMS.IMSA.PROCLIB
/*JOBPARM SYSAFF=MVST
//*
//         EXEC DFSJMP,
//              IMSID=IMSA,JVMOPMAS=DFSJVMMS,ENVIRON=DFSJVMEV,
//              SSM=DSNT,
//              CL1=009,CL2=009,CL3=009,CL4=009
//STEPLIB  DD DSN=IMS.V13R1.SDFSJLIB,DISP=SHR
//         DD DSN=IMS.IMSA.SDFSRESL,DISP=SHR
//         DD DSN=DB2.V12.SDSNEXIT,DISP=SHR
//         DD DSN=DB2.V12.SDSNLOAD,DISP=SHR
//         DD DSN=DB2.V12.SDSNLOD2,DISP=SHR
//DFSESL   DD DSN=IMS.IMSA.SDFSRESL,DISP=SHR
//         DD DSN=DB2.V12.SDSNEXIT,DISP=SHR
//PROCLIB  DD DSN=IMS.IMSA.PROCLIB,DISP=SHR
//JAVAOUT  DD PATH='/tmp/imsajmp.out',
Command ===>  sub                                          Scroll
```

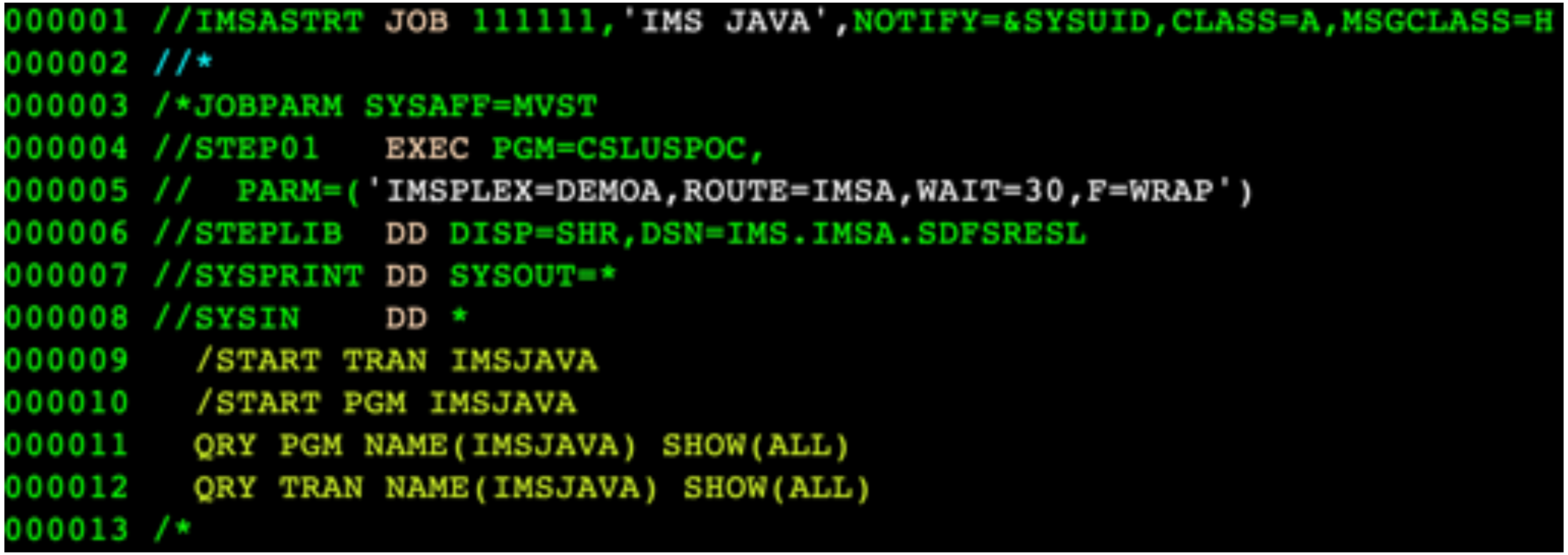
ALEX

SYSTEMS
PROGRAMMER

Java in IMS

Setting up Java program in IMS

- **Step 7.** Start the IMS program / transaction



```
//IMSASTRT JOB 111111,'IMS JAVA',NOTIFY=&SYSUID,CLASS=A,MSGCLASS=H
//*
/*JOBPARM SYSAFF=MVST
//STEP01   EXEC PGM=CSLUSPOC,
//  PARM=('IMSPLEX=DEMOA,ROUTE=IMSA,WAIT=30,F=WRAP')
//STEPLIB  DD DISP=SHR,DSN=IMS.IMSA.SDFSRESL
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  /START TRAN IMSJAVA
  /START PGM IMSJAVA
  QRY PGM NAME(IMSJAVA) SHOW(ALL)
  QRY TRAN NAME(IMSJAVA) SHOW(ALL)
/*
```

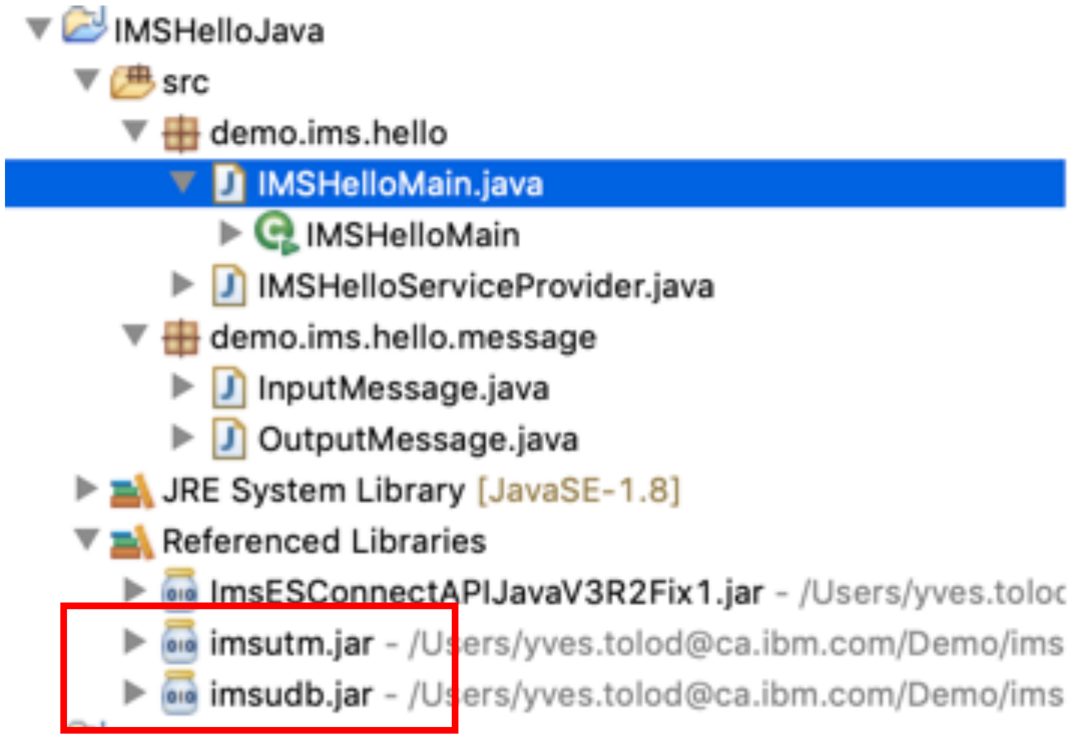
ALEX

SYSTEMS
PROGRAMMER

Java in IMS

Writing your first Java Program in IMS

- **Step 1.** Import the IMS universal drivers (`imsutm.jar` and `imsudb.jar`) into your Java project



- IMSHelloJava
  - src
    - demo.ims.hello
      - IMSHelloMain.java
        - IMSHelloMain
      - IMSHelloServiceProvider.java
    - demo.ims.hello.message
      - InputMessage.java
      - OutputMessage.java
  - JRE System Library [JavaSE-1.8]
  - Referenced Libraries
    - ImsESConnectAPIJavaV3R2Fix1.jar - /Users/yves.toloc
    - imsutm.jar - /Users/yves.tolod@ca.ibm.com/Demo/ims
    - imsudb.jar - /Users/yves.tolod@ca.ibm.com/Demo/ims

CAROL

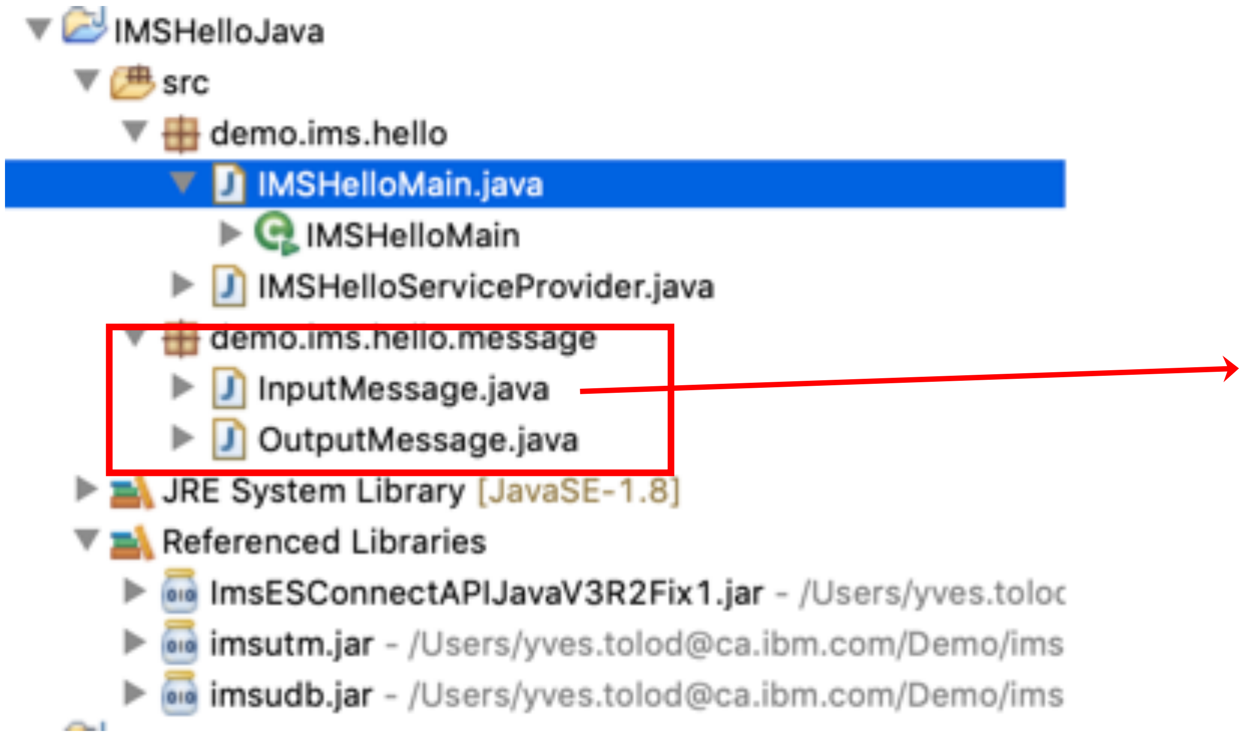
JAVA
PROGRAMMER

Java in IMS
Writing your first Java Program in IMS

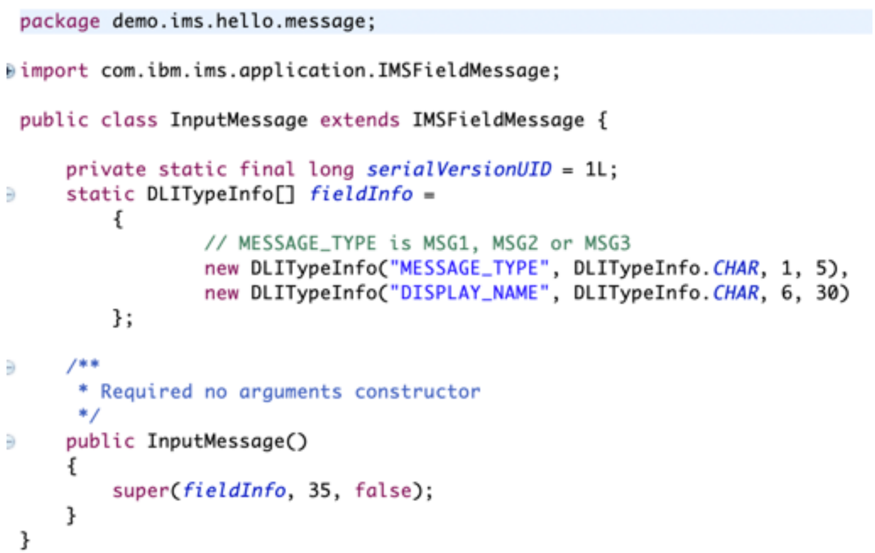
- **Step 2.** Define the layout of the input and output message

CAROL

JAVA
PROGRAMMER



```
▼ IMSHelloJava
  ▼ src
    ▼ demo.ims.hello
      ▼ IMSHelloMain.java
        ▶ IMSHelloMain
      ▶ IMSHelloServiceProvider.java
    ▼ demo.ims.hello.message
      ▶ InputMessage.java
      ▶ OutputMessage.java
  ▶ JRE System Library [JavaSE-1.8]
  ▼ Referenced Libraries
    ▶ ImsESConnectAPIJavaV3R2Fix1.jar - /Users/yves.toloc
    ▶ imsutm.jar - /Users/yves.tolod@ca.ibm.com/Demo/ims
    ▶ imsudb.jar - /Users/yves.tolod@ca.ibm.com/Demo/ims
```

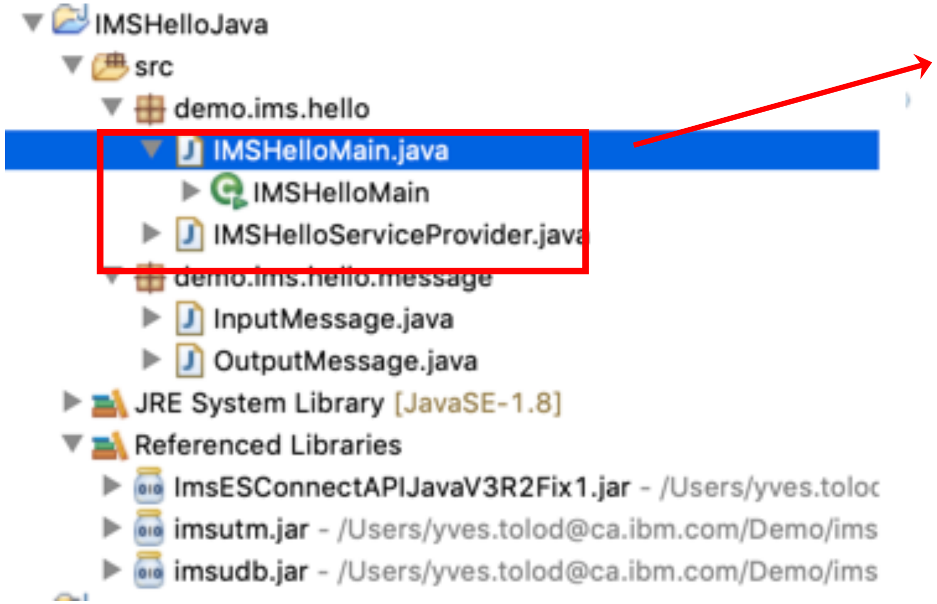


```java
package demo.ims.hello.message;

import com.ibm.ims.application.IMSFieldMessage;

public class InputMessage extends IMSFieldMessage {

    private static final long serialVersionUID = 1L;
    static DLITypeInfo[] fieldInfo =
        {
                // MESSAGE_TYPE is MSG1, MSG2 or MSG3
                new DLITypeInfo("MESSAGE_TYPE", DLITypeInfo.CHAR, 1, 5),
                new DLITypeInfo("DISPLAY_NAME", DLITypeInfo.CHAR, 6, 30)
        };

    /**
     * Required no arguments constructor
     */
    public InputMessage()
    {
        super(fieldInfo, 35, false);
    }
}
```

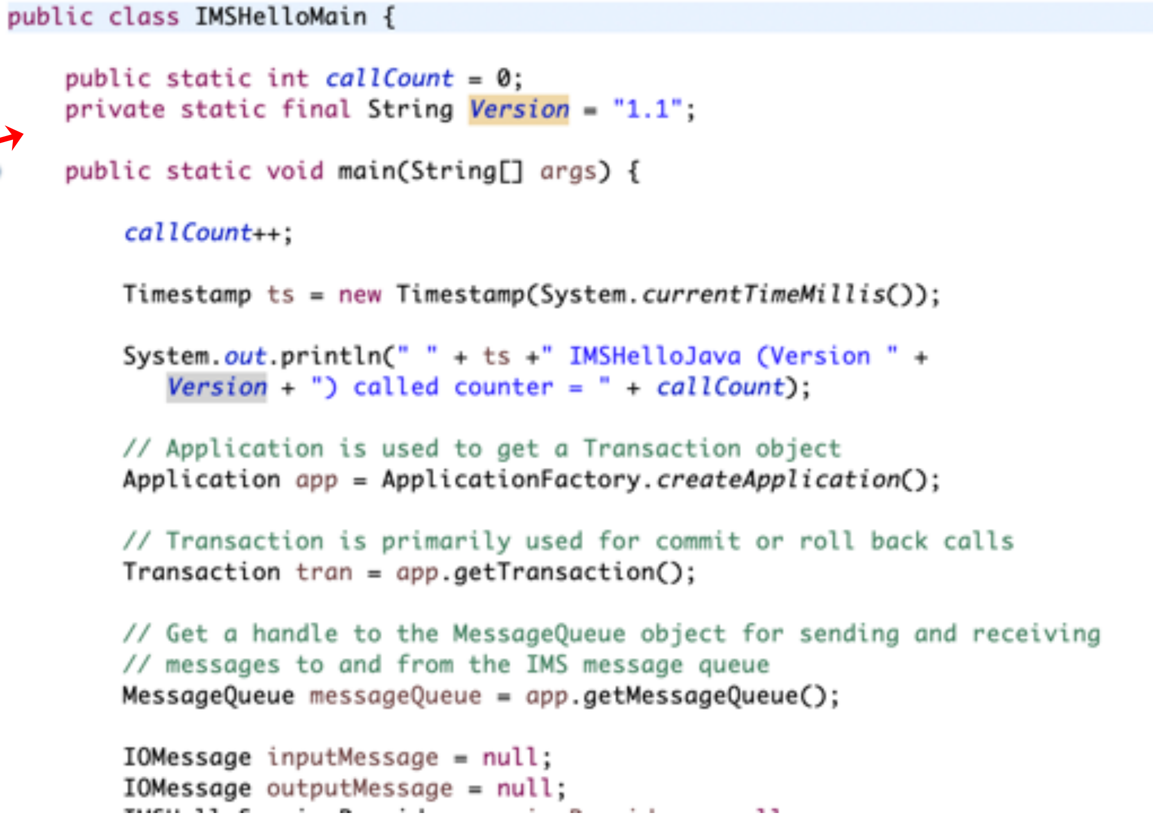
Java in IMS

Writing your first Java Program in IMS

- **Step 3.** Develop the application that accesses the IMS message queues and performs the business logic.



- IMSHelloJava
  - src
    - demo.ims.hello
      - IMSHelloMain.java
        - IMSHelloMain
      - IMSHelloServiceProvider.java
    - demo.ims.hello.message
      - InputMessage.java
      - OutputMessage.java
  - JRE System Library [JavaSE-1.8]
  - Referenced Libraries
    - ImsESConnectAPIJavaV3R2Fix1.jar - /Users/yves.toloc
    - imsutm.jar - /Users/yves.tolod@ca.ibm.com/Demo/ims
    - imsudb.jar - /Users/yves.tolod@ca.ibm.com/Demo/ims



```
public class IMSHelloMain {

    public static int callCount = 0;
    private static final String Version = "1.1";

    public static void main(String[] args) {

        callCount++;

        Timestamp ts = new Timestamp(System.currentTimeMillis());

        System.out.println(" " + ts +" IMSHelloJava (Version " +
            Version + ") called counter = " + callCount);

        // Application is used to get a Transaction object
        Application app = ApplicationFactory.createApplication();

        // Transaction is primarily used for commit or roll back calls
        Transaction tran = app.getTransaction();

        // Get a handle to the MessageQueue object for sending and receiving
        // messages to and from the IMS message queue
        MessageQueue messageQueue = app.getMessageQueue();

        IOMessage inputMessage = null;
        IOMessage outputMessage = null;
```

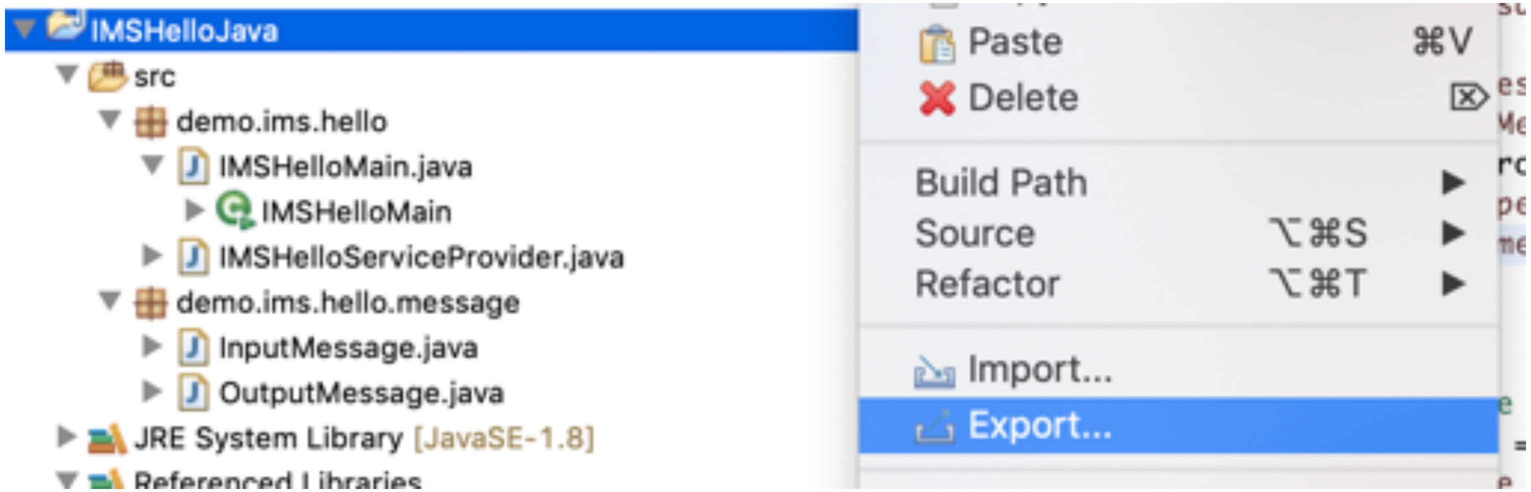
CAROL

JAVA
PROGRAMMER

Java in IMS

Writing your first Java Program in IMS

- **Step 4.** Compile and export the application `.jar` file and upload to z/OS UNIX System Services file system in binary mode.



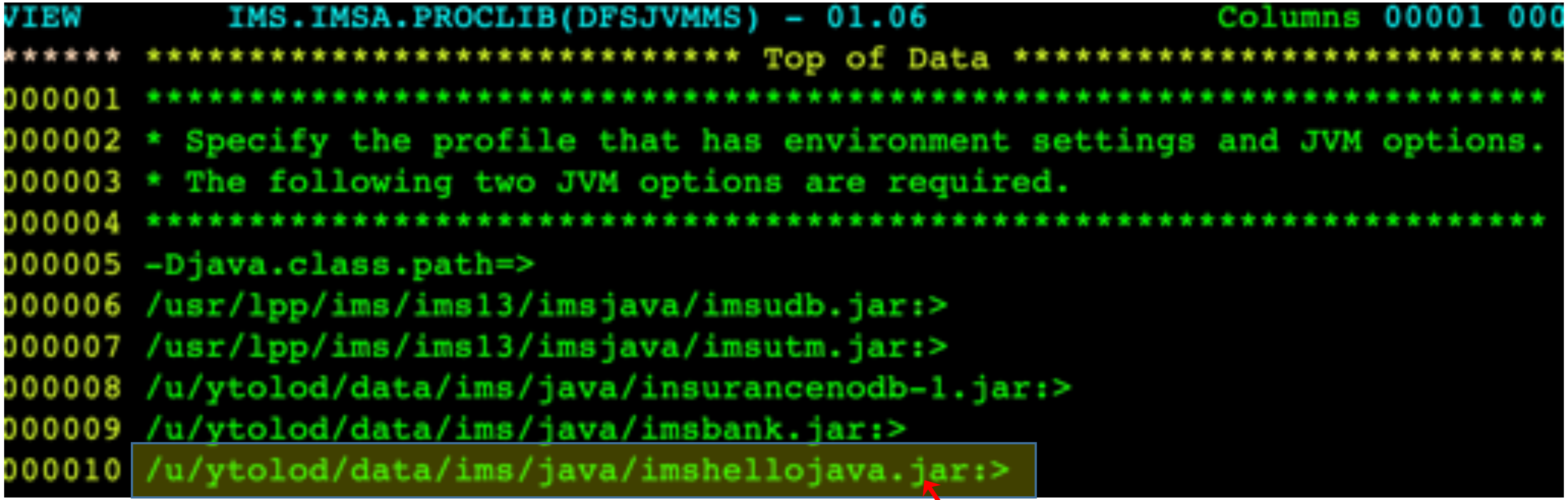
CAROL

JAVA
PROGRAMMER

Java in IMS

Writing your first Java Program in IMS

- **Step 5.** Deploy and Test your application



```
VIEW       IMS.IMSA.PROCLIB(DFSJVMMS) - 01.06                Columns 00001 000
****** ***************************** Top of Data *****************************
000001 ***********************************************************************
000002 * Specify the profile that has environment settings and JVM options.
000003 * The following two JVM options are required.
000004 ***********************************************************************
000005 -Djava.class.path=>
000006 /usr/lpp/ims/ims13/imsjava/imsudb.jar:>
000007 /usr/lpp/ims/ims13/imsjava/imsutm.jar:>
000008 /u/ytolod/data/ims/java/insurancenodb-1.jar:>
000009 /u/ytolod/data/ims/java/imsbank.jar:>
000010 /u/ytolod/data/ims/java/imshellojava.jar:>
```

Contains the Java IMS application

CAROL

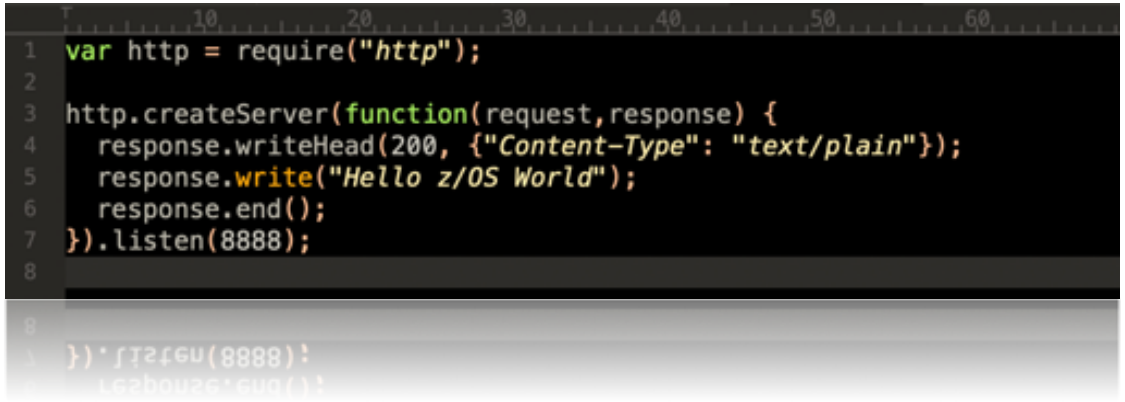
JAVA
PROGRAMMER

Quick Overview

What is Node.js?

- **Server side JavaScript platform**
- Designed to build scalable network applications
  - Lightweight and efficient
- Uses an event-driven, single-threaded, non-blocking I/O model
  - Best suited for data-intensive (i.e. I/O bound) applications
- Provides a module-driven, highly scalable approach to application design and development that encourages agile practices

‘Hello World’ Web Application



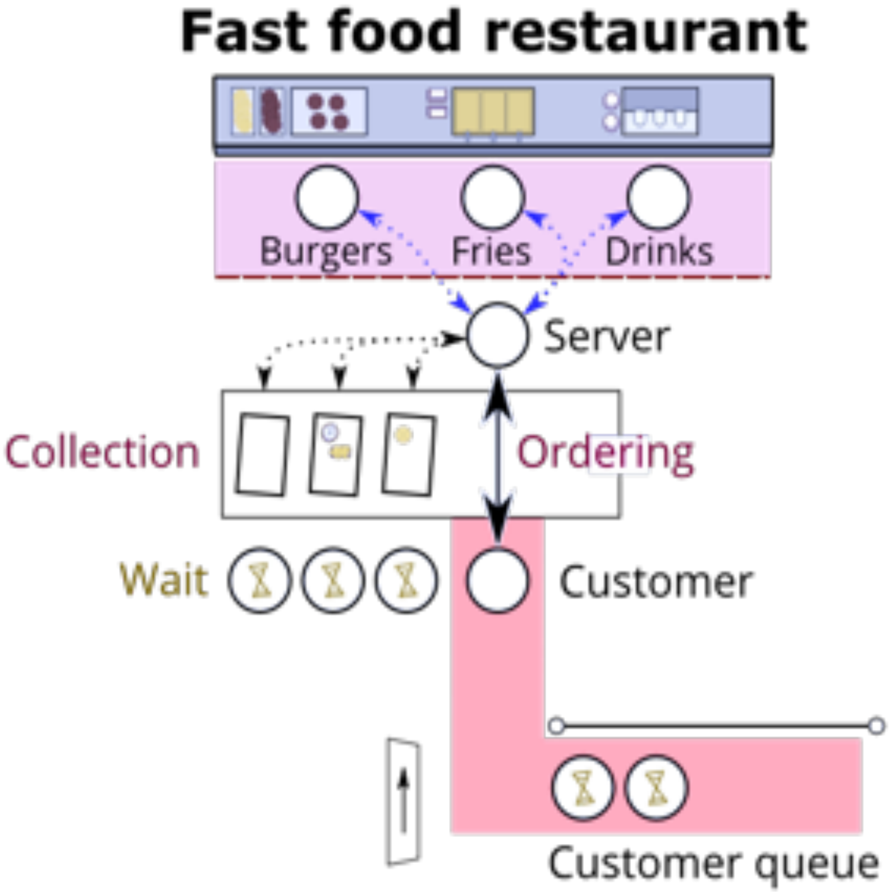
```
var http = require("http");

http.createServer(function(request,response) {
  response.writeHead(200, {"Content-Type": "text/plain"});
  response.write("Hello z/OS World");
  response.end();
}).listen(8888);
```

Emerging as the favored choice for digital transformation - Steadily establishing its place within enterprises

Quick Overview

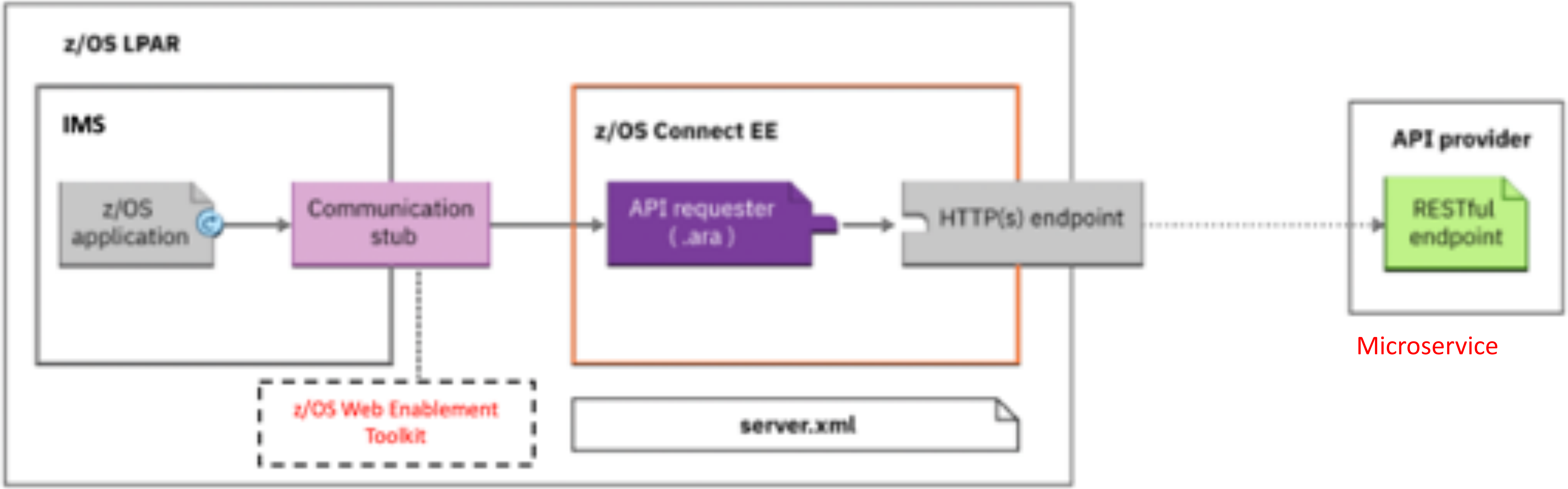
How Node.js works? *The Event Loop*

- **One thread multiplexes for multiple requests**
  - No waiting for a response
  - Handles return from I/O when notified
- **Scalability determined by**
  - CPU Usage
  - “Back end” responsiveness
- **Concurrency** determined by how fast the food server can work



Using Node.js with IMS

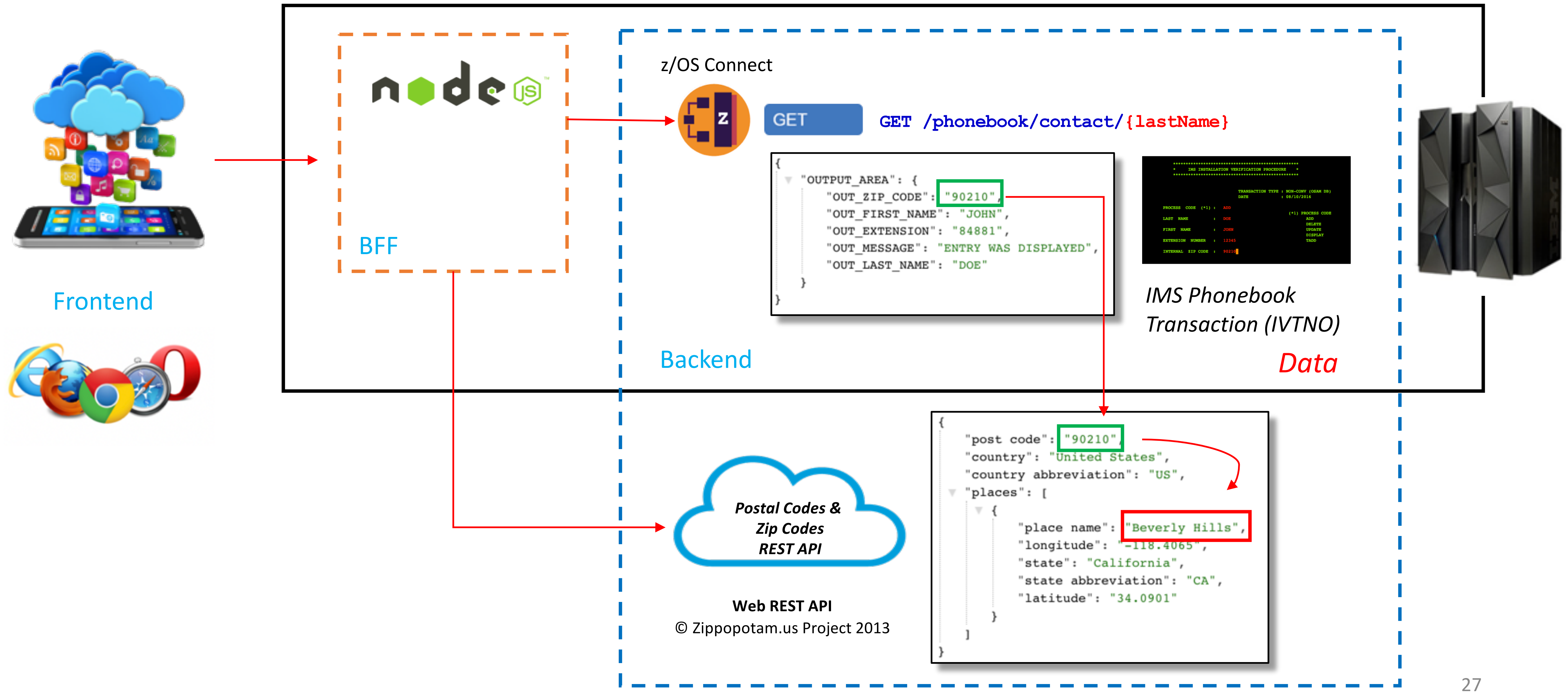
Extending your IMS Program with Node.js



Microservice

Using Node.js with IMS

Consolidating Multiple IMS Calls to a single Orchestration API



GET /phonebook/contact/{lastName}

```
{
  "OUTPUT_AREA": {
    "OUT_ZIP_CODE": "90210",
    "OUT_FIRST_NAME": "JOHN",
    "OUT_EXTENSION": "84881",
    "OUT_MESSAGE": "ENTRY WAS DISPLAYED",
    "OUT_LAST_NAME": "DOE"
  }
}
```

IMS Phonebook Transaction (IVTNO)

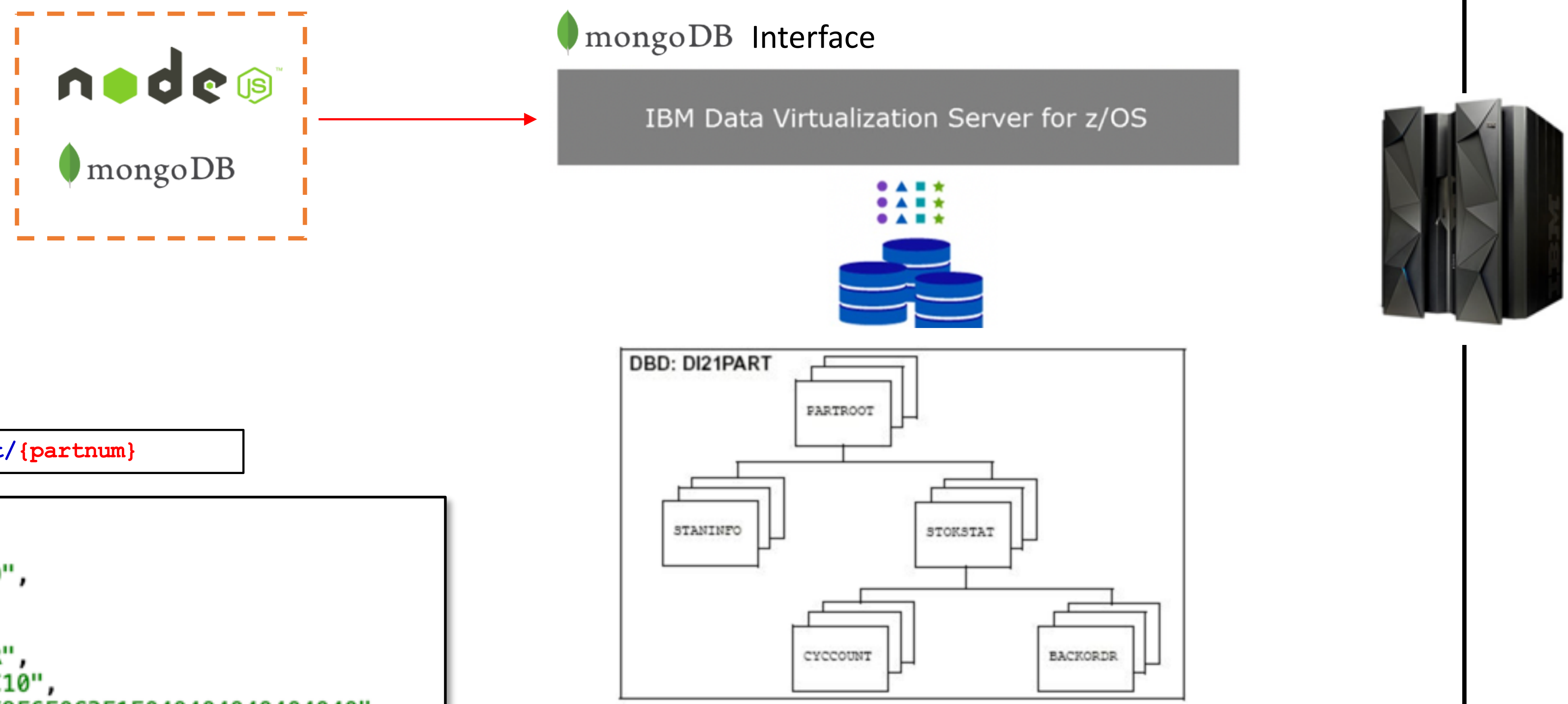
```
{
  "post code": "90210",
  "country": "United States",
  "country abbreviation": "US",
  "places": [
    {
      "place name": "Beverly Hills",
      "longitude": "-118.4065",
      "state": "California",
      "state abbreviation": "CA",
      "latitude": "34.0901"
    }
  ]
}
```

Web REST API

© Zippopotam.us Project 2013

Using Node.js with IMS

REST API Node + MongoDB using DVM to access IMS Data



GET

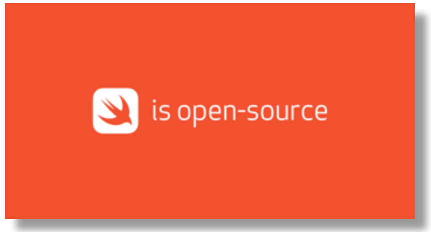
```
GET /dvm/ims/part/{partnum}
```

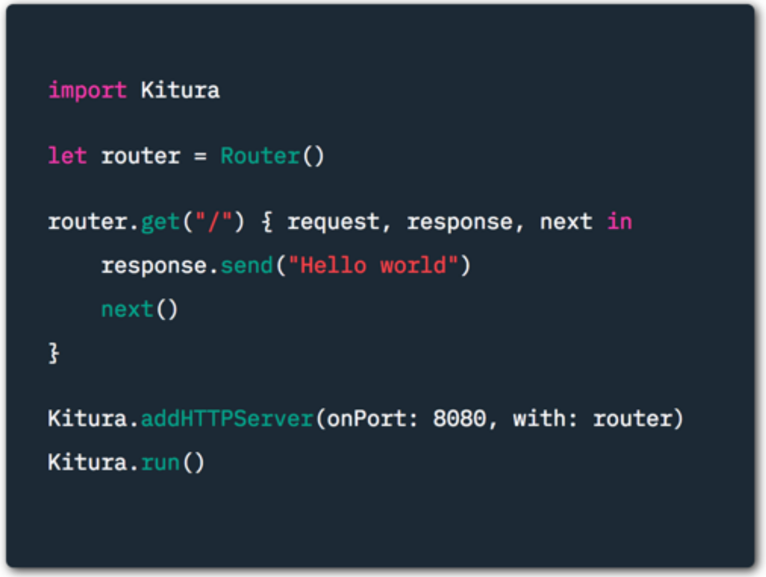
```
[
  {
    "partkey": "02AN960C10",
    "prefix": "02",
    "partno": "AN960C10",
    "description": "WASHER",
    "record_id": "02AN960C10",
    "child_id": "F0F2C1D5F9F6F0C3F1F040404040404040"
  }
]
```

Quick Overview

What is Swift?

- Modern language developed by Apple Inc. in **2014**
- Compiled programming language for iOS, macOS, watchOS, tvOS, Linux, **including Linux on z Systems and z/OS**
- **Open sourced** in **2015**
- A modern 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





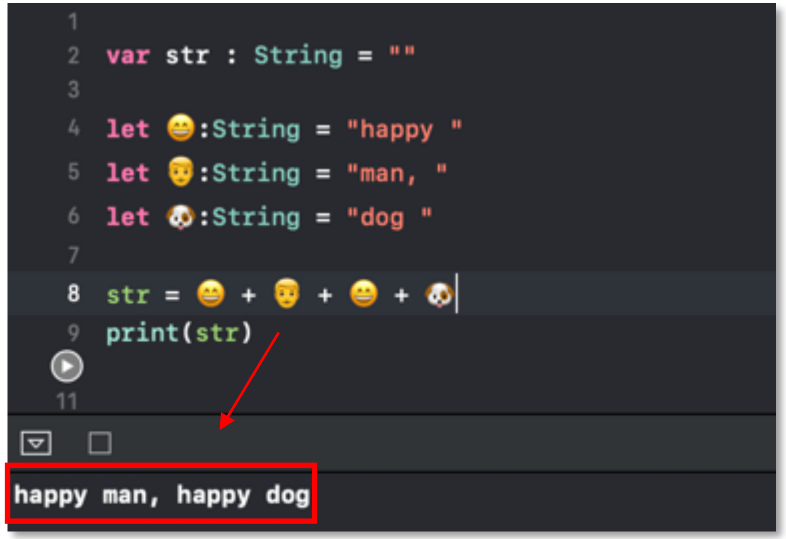
```
import Kitura

let router = Router()

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

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

‘Hello World’ Web Server Program

Quick Overview

Why Swift? *It's fast, it's safe, it's modern, it's open-source*

- Modern language developed by Apple Inc. in **2014**
- Compiled programming language for iOS, macOS, watchOS, tvOS, Linux, **including Linux on z Systems and z/OS**
- **Open sourced** in **2015**
- A modern 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



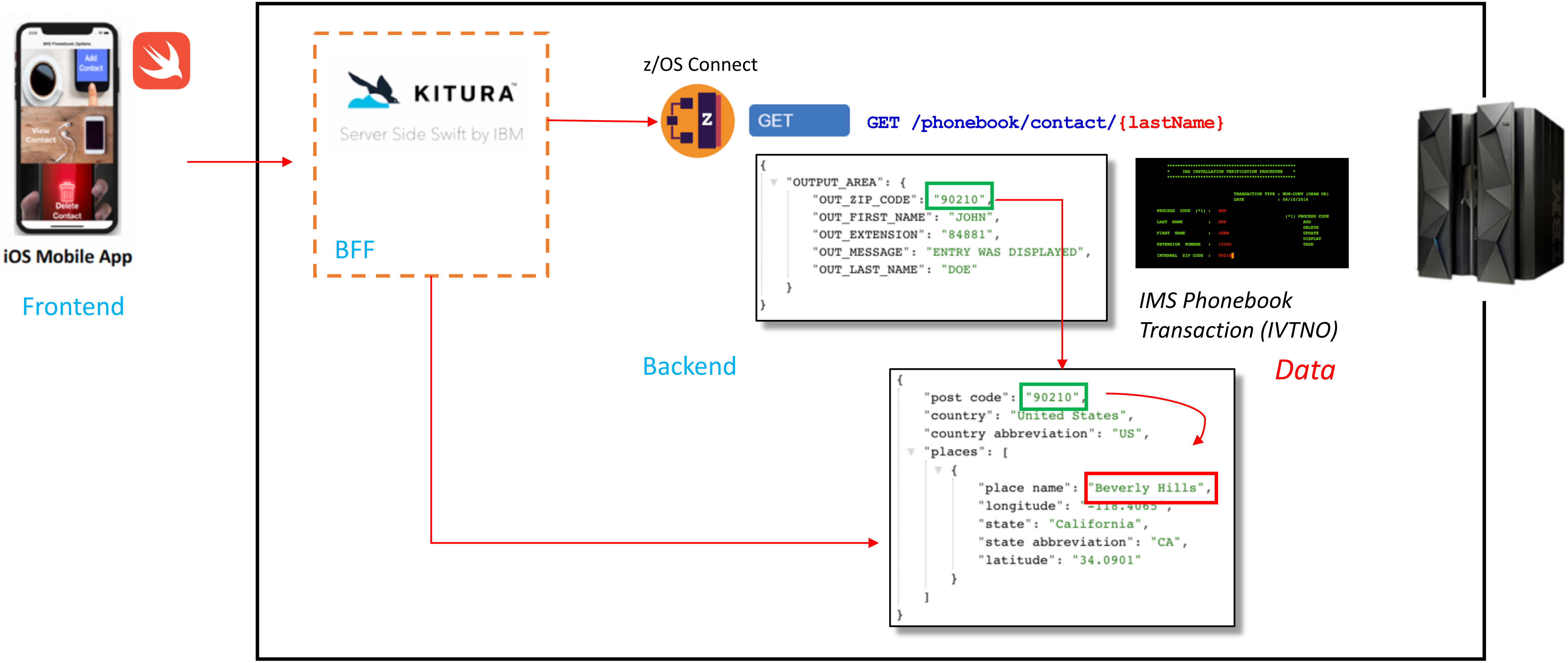
```swift

var str : String = ""

let 😁:String = "happy "
let 👱:String = "man, "
let 🐶:String = "dog "

str = 😁 + 👱 + 😁 + 🐶
print(str)

```

```
happy man, happy dog
```

is open-source

Using Swift with IMS

Augmenting data returned by your IMS program



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

```
{
  "OUTPUT_AREA": {
    "OUT_ZIP_CODE": "90210",
    "OUT_FIRST_NAME": "JOHN",
    "OUT_EXTENSION": "84881",
    "OUT_MESSAGE": "ENTRY WAS DISPLAYED",
    "OUT_LAST_NAME": "DOE"
  }
}
```

```
     **************************************************
     *     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
                                                     DELETE
   FIRST  NAME         :   JOHN                      UPDATE
                                                     DISPLAY
   EXTENSION  NUMBER   :   12345                     TADD

   INTERNAL  ZIP CODE  :   90210
```

```
{
    "post code": "90210",
    "country": "United States",
    "country abbreviation": "US",
    "places": [
        {
            "place name": "Beverly Hills",
            "longitude": "-118.4065",
            "state": "California",
            "state abbreviation": "CA",
            "latitude": "34.0901"
        }
    ]
}
```

Additional Information

Get Started on IBM SDK for Node.js – z/OS today

No cost download

Based on Node.js V6.x

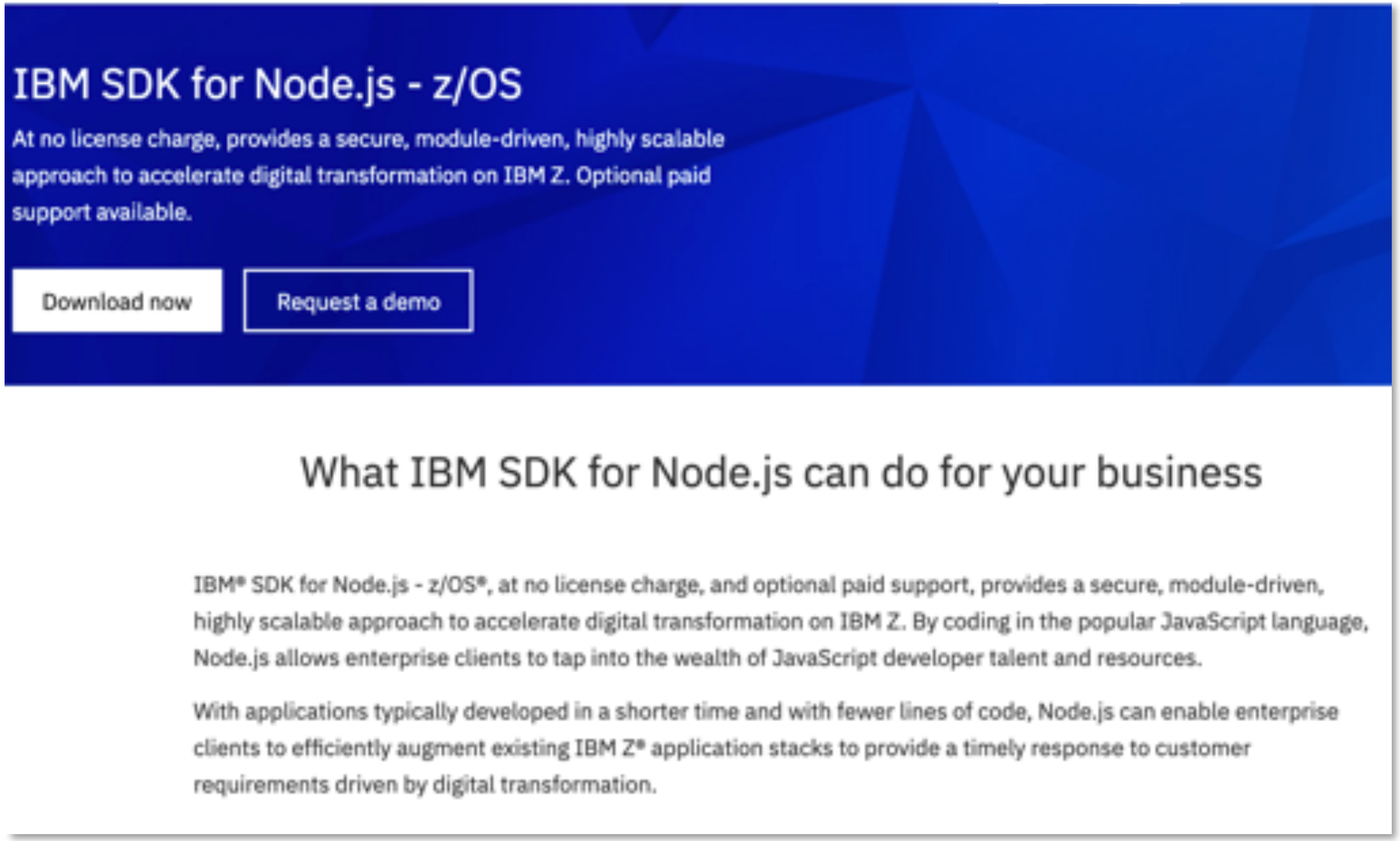
SMP/E Install

Software Requirements:

- z/OS v2.2
- z/OS v2.3

Hardware Requirements

- IBM z14
- IBM z13 / z13s
- IBM zEnterprise EC12 / BC12
- IBM zEnterprise 196 / 114



IBM SDK for Node.js - z/OS

At no license charge, provides a secure, module-driven, highly scalable approach to accelerate digital transformation on IBM Z. Optional paid support available.

What IBM SDK for Node.js can do for your business

IBM® SDK for Node.js - z/OS®, at no license charge, and optional paid support, provides a secure, module-driven, highly scalable approach to accelerate digital transformation on IBM Z. By coding in the popular JavaScript language, Node.js allows enterprise clients to tap into the wealth of JavaScript developer talent and resources.

With applications typically developed in a shorter time and with fewer lines of code, Node.js can enable enterprise clients to efficiently augment existing IBM Z® application stacks to provide a timely response to customer requirements driven by digital transformation.

https://www.ibm.com/ca-en/marketplace/sdk-nodejs-compiler-zos

Additional Information

Learn More About Node.js on z/OS

IBM Marketplace (Node.js on z/OS):
https://www.ibm.com/ca-en/marketplace/sdk-nodejs-compiler-zos

Node.js on z/OS GitHub Samples:
https://github.com/zosconnect/sample-nodejs-clients

Mainframe DEV:
https://developer.ibm.com/mainframe/2018/01/19/reasons-host-node-js-applications-zos/

Node.js in CICS
https://developer.ibm.com/cics/2018/07/03/node-js-developers-introduction-node-js-cics/

Calling REST APIs from z/OS Programs using z/OS Connect
https://www.ibm.com/support/knowledgecenter/en/SS4SVW_3.0.0/facilitating/facilitating.html

Additional Information

Get Started on IBM Toolkit for Swift on z/OS today

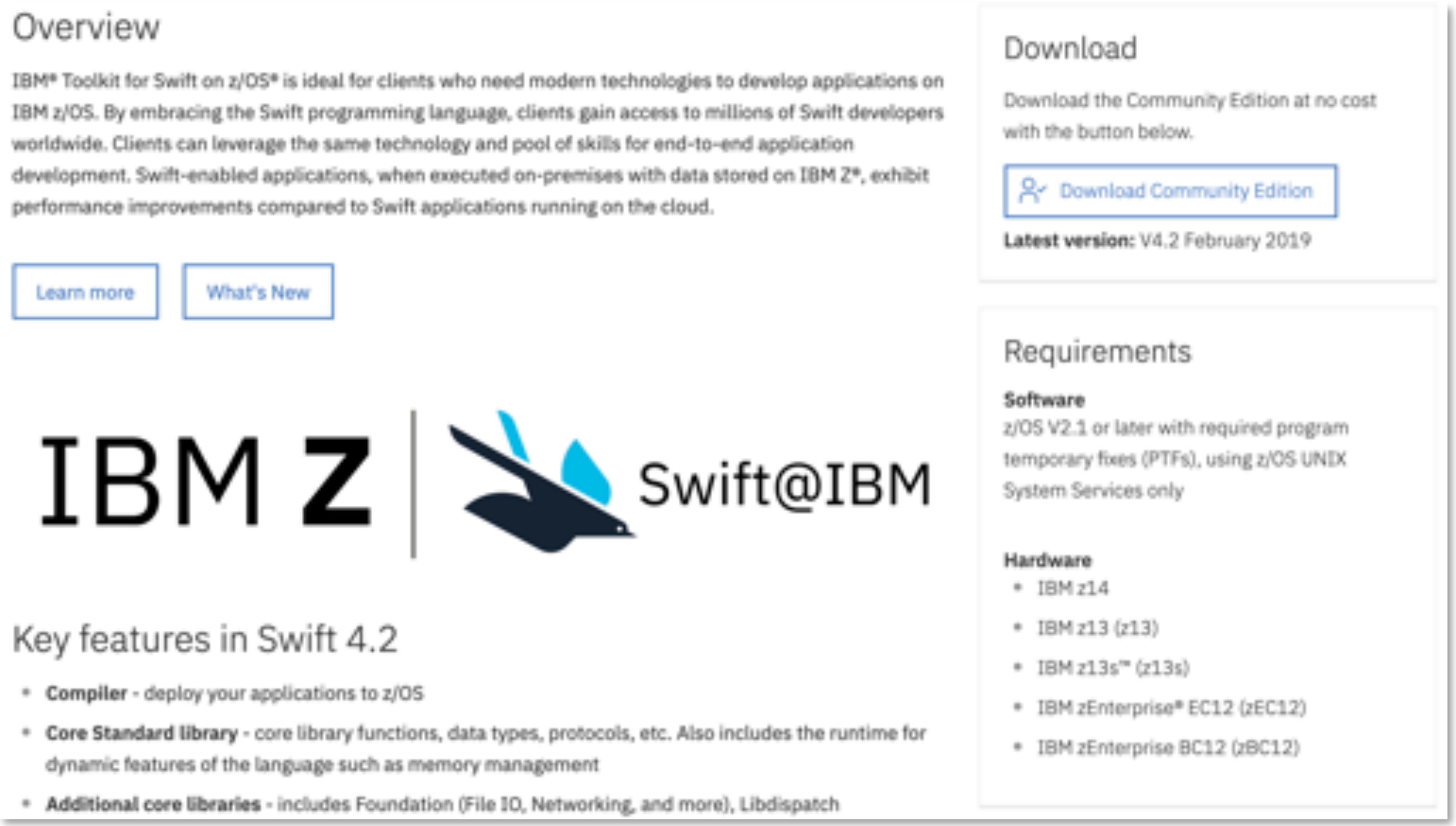
No cost download

Software Requirements:

- z/OS v2.1 or later with required PTFs
- Using z/OS UNIX System Services only

Hardware Requirements

- IBM z14
- IBM z13
- IBM z13s
- IBM zEnterprise EC12
- IBM zEnterprise BC12



Overview

IBM® Toolkit for Swift on z/OS® is ideal for clients who need modern technologies to develop applications on IBM z/OS. By embracing the Swift programming language, clients gain access to millions of Swift developers worldwide. Clients can leverage the same technology and pool of skills for end-to-end application development. Swift-enabled applications, when executed on-premises with data stored on IBM Z®, exhibit performance improvements compared to Swift applications running on the cloud.

Key features in Swift 4.2

- **Compiler** - deploy your applications to z/OS
- **Core Standard library** - core library functions, data types, protocols, etc. Also includes the runtime for dynamic features of the language such as memory management
- **Additional core libraries** - includes Foundation (File IO, Networking, and more), Libdispatch

Download

Download the Community Edition at no cost with the button below.

Latest version: V4.2 February 2019

Requirements

Software

z/OS V2.1 or later with required program temporary fixes (PTFs), using z/OS UNIX System Services only

Hardware

- IBM z14
- IBM z13 (z13)
- IBM z13s™ (z13s)
- IBM zEnterprise® EC12 (zEC12)
- IBM zEnterprise BC12 (zBC12)

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

Additional Information

Learn More About 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

Have you heard?

IBM IMS

The IMS team will soon be delivering a regular IMS eNews straight to your inbox!

It will include Announcements, Events, Education,
New Offerings, Videos, and much more...
including the return of **I am IMS**!

Register to get on the mailing list:
ibm.biz/IMS_eNews

IBM