Speaker background

- Saw first Java presentation in 1996 ACM at UT
- Programming in Java since 1999
- Scala (seriously in November 2012, Martin Odersky, Functional Programming in Scala, Coursera)
- Big data processing Hadoop since 2010, Spark since 2014
- Teaching JHU grad comp sci, Hadoop for UMBC Training, Spark for Databricks

The Scala Programming Language

- Odersky
- Lightbend
- Multi-paradigm
- Statically typed
- Scalable - small to large
- Stretch your mind

Sca(lable) la(nguage)

- Apache Kafka (LinkedIn)
- Apache Spark (Databricks)
- Finagle (Twitter) - Twitter switched from Ruby on Rails to Scala
- Akka (Lightbend)
- Lucid Software - scala.js presentation Fluent 2016 Paul Draper
- Play Web Framework
  - Lichess Online Chess
- Lightbend customers: Walmart, Verizon, Twitter, LinkedIn, Coursera, The Guardian, Airbnb...

Scala to Java bytecode

- JIT
- JVMs for most OSes and Android Dalvik(old) now Android Runtime (ART)
  - Portability
  - Security
  - Garbage collectors
- Full Java interop - leverage Java libraries

Exploration - Scala Shell and Worksheet

- Scala shell
- IDEA Scala Worksheet
- Scala IDE
- sbt console

Typesafe Config Exploration

```
import com.typesafe.config.ConfigFactory
val configStr =
   """analytic {
   |    startTime = 2016103111,
   |    endTime = 2016103115
   |}
   """.stripMargin

val appConfig =
   ConfigFactory.parseString(configStr)
appConfig.hasPath("analytic.timeStart")
appConfig.getString("analytic.startTime")
```

Scala Tour

- Conciseness
  - no semicolons, static type inference, lots of syntactic sugar, functional processing, implicits
- Mixed Paradigms
  - Object Oriented
    - all objects, classes, case classes, traits, mixins
  - Functional

scalatour/01-NoSemicolons

- optional semicolons
- type inference
- vals vs. vars
- higher-order functions on collections

scalatour/02-Functions

- Use def keyword to define function/method
- arg type declaration after variable name
- return type
- body of function
- expressions vs. statements - last expression is returned
- function literals

scalatour/03-AllObjects

- Everything is an object (but might translate to Java primitive)
- Use == for testing equality (eq object reference)

scalatour/04-Tuples

- Most useful as pair/two-tuple (up to 22)
- Strongly typed for each position
- access via _1, _2 methods or pattern matching

scalatour/05-Options

- Avoid null and NullPointerException (NPE)
- Option[T] - Some[T] or None
  - sealed abstract class Option, class Some, object None
- Options act like a collection

scalatour/06-Collections

- Array
- Immutable, mutable data structures
  - List
  - Higher-order functions
    - filter, map, flatMap, reduce, fold...
  - Map
  - Set, Vector...

Scala Docs

- http://www.scala-lang.org/api/2.11.8/
- StringOps
- List - Singleton object vs. class

scalatour/07-MultilineStrings

- Triple quotes
- substitution (f for printf formatting)

scalatour/08-FunctionalPatternMatching

- match construct
- match by type, structure
- default case or MatchError

scalatour/09-ParsingConfig

- Match on regular expressions
- Go Options

scalatour/10-ClassesTraitsMixins

- class - constructor/body
- constructor args - val, var, no modifier
- traits

scalatour/11-CaseClasses

- provide val accessors
- apply/unapply, hashCode, toString
- pattern matching

scalatour/12-Scripting

- In the small
- sys.process
- sys.env
- sys.props

scalatour/13-JavaInterop

- to/from Java/Scala collections
- BeanProperty for getters/setters

scalatour/14-Implicits

- Use sparingly!
- Powerful way to extend closed classes

scalatour/Spark15

- Implemented in Scala
- Powerful functional primitives for scalable cluster processing