Principles of Programming Languages

Tom Flaherty

Version 0.0, November 9, 2016

Table of Contents

Introduction . . . 1

- Organization . . . 1
- Guilding Principles . . . 1
- Selecting a Language . . . 1

Concepts . . . 2

- Paradigm . . . 2
  - What is a Paradigm? . . . 2
  - Paradigm shifts . . . 2
  - Multiple Paradigms . . . 2
- Principles . . . 2
  - What is a Principle? . . . 2
  - Evolution of Principles . . . 2
  - Core Concepts . . . 2
- Languages . . . 2
  - Productivity . . . 2
  - Intrinsic . . . 2
  - Tracing Principles . . . 2
- Applications . . . 2
  - What is Illustrated . . . 2
  - Language Strength . . . 2
  - Polyglot . . . 2

Composition . . . 3

- Classes . . . 3
  - Classification . . . 3
  - Don’t Repeat Yourself . . . 3
  - Uniform Hierarchies . . . 3
  - Mixins . . . 3
- Typing . . . 3
  - Static . . . 3
  - Dynamic . . . 3
  - Specialized . . . 3
  - Inferred . . . 3
  - Generic . . . 3
  - Type classes . . . 3
- Meta Programming . . . 3
  - Modifying Structure . . . 3
  - Interceding Behavior . . . 3
  - Macros at Compile Time . . . 3

- Construction 3
  - Factories 3
  - Builders 3
  - Reflection 3
  - Cloning with Prototypes 4
- Dependency Injection 4
  - Runtime Construction 4
  - Interface Injection 4
- Summary 4
- References 4

Orchestration 5
- Polymorphic 5
  - Universal 5
  - Ad Hoc 5
- Behavior 5
  - Association 5
  - Scenarios 5
  - Design by Contract 5
  - Pre and Post Conditions 5
- Distributed 5
  - Resource 5
  - REST 5
  - The Tier Context 5
  - Messaging 5
- Parallelism 5
  - Concurrency 5
  - Functional Concurrency 6
  - Dependencies 6
- Summary 6
- References 6

Preservation 7
- Scope 7
  - Lexical 7
  - Libraries 7
  - Packaging 7
- Build 7
- State 7
  - Fallacies 7
  - Continuations 7
- Test Harness 7
  - Isolating Components 7

Enabling Diagnostics 7
Debugging 7
Test Granularity 7
Test Provisioning 7
Defense In Depth 7
Detecting Incidents 7
Failing Fast 7
Containment 7
Summary 7
References 8
Transformation 9
Representation 9
Function Shaping 9
Wrapping Unwrapping 9
Semantic Consistency 9
Emphemeral Mistakes 9
Data Driven APIs 9
Transformation Preferred 9
Multiple Futures 9
Pattern Matching 9
Traversal 9
Sequential 9
Recursive 9
Heuristics 9
Higher Orders 9
Closures 9
Currying 9
Partial Functions 9
Laziness 9
Monads 9
Category Theory 9
Shape Wrap and Roll 10
Duality 10
Unification 10
ReactiveX 10
Observer Patterh 10
Create 10
Combine 10
Listen 10
Summary 10
References 10

Expression 11
Convention 11
Regularity 11
Naming Conventions 11
Interpretation 11
Evaluation 11
DSL 11
Continuity 11
Wholeness 11
Transparency 11
Declarative 11
Impediance Mismatch 11
Adaptation 11
Hot Spots 11
Indirection 11
Decoupling 11
Refactoring 11
Comprehension 11
Simplicity 11
Communication 11
Git 12
Code Documentation 12
ReactiveX 12
Better Code Bases 12
Functional 12
Reactive Operators 12
Async 12
Concurrency 12
Summary 12
References 12
A tour of Groovy 13
Expressive Syntax 13
Command line 13
Parsing XML 13
Meta Object Protocol 14
Meta Class 14
Expanding and Invoking 14
Capabilities 14
Multi-paradigm 15
Object Oriented 15
Dynamic 15

Closures 15
Principles in Groovy 16
Composition 16
Orchestration 16
Preservation 16
Transformation 16
Expression 16
Grails and Ola’s Pyramid 17
Grails Architecture 17
Ola’s Pyramid Layers 17
Plugins for Propagation 17
REST with Grails 17
Principles in Grails 17
Summary 18
References 19
A Tour of Clojure 20
Lisp Revitalized 20
JVM Influence 20
Software Transactions 20
Polymorphism 21
Multi-Methods 21
Macros 21
Principles in Clojure 22
Composition 22
Orchestration 22
Preservation 22
Transformation 22
Expression 22
Ring and Composure 23
HTTP Request Response 23
JavaScript and Clojure 23
HTML5 and CSS3 23
Page Navigation 23
Data Interchange 23
Principles 23
Summary 24
References 25
A Tour of JavaScript 26
JavaScript Revitalized 26
Influences 26
Functional 26

Dialects 27
    CoffeeScript 27
    EcmaScript 27
    TypeScript 27
Principles in JavaScript 28
    Composition 28
    Orchestration 28
    Preservation 28
    Transformation 28
    Expression 28
NodeJS 29
    Packages 29
    MEAN 29
    Electron 29
Summary 30
References 31
A Tour of Scala 32
    Compilation Hints 32
    Java 7 in Scala 32
Objected Oriented Scala 33
    Uniform Class Hierarchy 33
Typing 34
    Type Inference 34
    Type Classes 34
    Generics 34
        Types to the Next Level 34
Traits 35
    Emergence of Traits 35
        PI Calculus 35
        Liskov Substitution 35
    Thick and Thin 35
    Self Type 35
    Interface Injection 35
    Stackable Modifications 35
Post Functional Scala 36
    For Comprehensions 36
    Monads 36
        Composability 36
        Shape Wrap and Poll 36
DSLs 37
    Operators and Implicits 37

- Parsing Combinators 37
- Case Classes 37
- Pattern Matching 37

Principles in Scala 38

- Composition 38
- Orchestration 38
- Preservation 38
- Transformation 38
- Expression 38

Akka 39

- Architecture 39
- Concurrency 39
- Persistence 39
- Apache Camel 39
- Principles 39

Summary 40

References 41

Representation 42

- Domain Knowledge 42
- Transformation 42
- Combination 42

Propagation 43

- APIs 43
- Visitation 43
- Pattern Matching 43

External DSLs 44

- Parsing options 44
- Combinatorial Parsers 44
- Parsers 44
  - Groovy 44
  - Clojure 44
  - Scala 44

Internal DSLs 45

- Arranging Syntax 45
- Overloading Operators 45
- Implicit Type Conversion 45
- Favorite Language 45

Summary 46

References 47

Picking Language 48

- Selecting Principles 48

  Tracing Principles 48
  DSL Enhancements 48
  Using Principles 48
Myths 49
  XML is a Language 49
  Encapsulating State 49
  Command and Control 49
  Single Paradigms 49
Human Nature 50
  They don’t Get It 50
  Open Source 50
  DSLs for Humans 50
Decision Criteria 51
  Discovering principles 51
  Paradigms 51
    Blending 51
    Synthesis 51
    Convergence 51
  Arrival of the Fittest 51
Summary References 52
Appendix A Language Survey 53
Appendix B Principles 54
Appendix C Paradigms 55
Appendix D Languages 56
Appendix E Traceability 57
Glossary 58
Index 59

Introduction

Organization

Guilding Principles

Selecting a Language

Concepts

Paradigm

What is a Paradigm?

Paradigm shifts

Multiple Paradigms

Principles

What is a Principle?

Evolution of Principles

Core Concepts

Languages

Productivity

Intrinsic

Tracing Principles

Applications

What is Illustrated

Language Strength

Polyglot

Composition

Classes

Classification

Don’t Repeat Yourself

Uniform Hierarchies

Mixins

Typing

Static

Dynamic

Specialized

Inferred

Generic

Type classes

Meta Programming

Modifying Structure

Interceding Behavior

Macros at Compile Time

Construction

Factories

Builders

Reflection

Cloning with Prototypes

Dependency Injection

Runtime Construction

Interface Injection

Summary

References

Orchestration

Polymorphic

Universal

Parametric

Inclusion

Ad Hoc

Overloading

Coercion

Behavior

Association

Scenarios

Class Responsibility Collaborator

Design by Contract

Pre and Post Conditions

Distributed

Resource

REST

The Tier Context

Messaging

Parallelism

Concurrency

Functional Concurrency

Dependencies

Summary

References

Preservation

Scope

Lexical

Libraries

Packaging

Build

State

Fallacies

Continuations

Test Harness

Isolating Components

Enabling Diagnostics

Debugging

Test Granularity

Test Provisioning

Defense In Depth

Detecting Incidents

Failing Fast

Containment

Summary

References

Transformation

Representation

Function Shaping

Wrapping Unwrapping

Semantic Consistency

Emphemeral Mistakes

Data Driven APIs

Transformation Preferred

Multiple Futures

Pattern Matching

Traversal

Sequential

Recursive

Heuristics

Higher Orders

Closures

Currying

Partial Functions

Laziness

Monads

Category Theory

Shape Wrap and Roll

Duality

Unification

ReactiveX

Observer Patterh

Create

Combine

Listen

Summary

References

Expression

Convention

Regularity

Naming Conventions

Interpretation

Evaluation

DSL

Continuity

Wholeness

Transparency

Declarative

Impediance Mismatch

Adaptation

Hot Spots

Indirection

Decoupling

Refactoring

Comprehension

Simplicity

Communication

Git

Code Documentation

ReactiveX

Better Code Bases

Functional

Reactive Operators

Async

Concurrency

Summary

References

A tour of Groovy

Expressive Syntax

Command line

Parsing XML

Meta Object Protocol

Meta Class

Expanding and Invoking

Capabilities

Multi-paradigm

Object Oriented

Dynamic

Closures

Principles in Groovy

Composition

Orchestration

Preservation

Transformation

Expression

Grails and Ola’s Pyramid

Grails Architecture

Ola’s Pyramid Layers

Plugins for Propagation

REST with Grails

Principles in Grails

Summary

References

A Tour of Clojure

Lisp Revitalized

JVM Influence

Software Transactions

Polymorphism

Multi-Methods

Macros

Principles in Clojure

Composition

Orchestration

Preservation

Transformation

Expression

Ring and Composure

HTTP Request Response

JavaScript and Clojure

HTML5 and CSS3

Page Navigation

Data Interchange

Principles

Summary

References

A Tour of JavaScript

JavaScript Revitalized

Influences

Functional

Dialects

CoffeeScript

EcmaScript

TypeScript

Principles in JavaScript

Composition

Orchestration

Preservation

Transformation

Expression

NodeJS

Packages

MEAN

Electron

Summary

References

A Tour of Scala

Compilation Hints

Java 7 in Scala

Objected Oriented Scala

Uniform Class Hierarchy

Typing

Type Inference

Type Classes

Generics

Types to the Next Level

Traits

Emergence of Traits

PI Calculus

Liskov Substitution

Thick and Thin

Self Type

Interface Injection

Stackable Modifications

Post Functional Scala

For Comprehensions

Monads

Composability

Shape Wrap and Poll

DSLs

Operators and Implicits

Parsing Combinators

Case Classes

Pattern Matching

Principles in Scala

Composition

Orchestration

Preservation

Transformation

Expression

Akka

Architecture

Concurrency

Persistence

Apache Camel

Principles

Summary

References

Representation

Domain Knowledge

Transformation

Combination

Propagation

APIs

Visitation

Pattern Matching

External DSLs

Parsing options

Combinatorial Parsers

Parsers

Groovy

Clojure

Scala

Internal DSLs

Arranging Syntax

Overloading Operators

Implicit Type Conversion

Favorite Language

Summary

References

Picking Language

Selecting Principles

Tracing Principles

DSL Enhancements

Using Principles

Myths

XML is a Language

Encapsulating State

Command and Control

Single Paradigms

Human Nature

They don’t Get It

Open Source

DSLs for Humans

Decision Criteria

Discovering principles

Paradigms

Blending

Synthesis

Convergence

Arrival of the Fittest

Summary References

Appendix A Language Survey

```
Fortran, Cobal, C and C++, C#, Objective-C
Algol, ADA, Pascal
APL, Erlang, ML,  F#, OZ,OCAML, Python
```

Appendix B Principles

Appendix C Paradigms

Appendix D Languages

Appendix E Traceability

Glossary

Index