Functional Programming

Lecture 13: FP in the Real World

Viliam Lisý

Artificial Intelligence Center
Department of Computer Science
FEE, Czech Technical University in Prague

viliam.lisy@fel.cvut.cz

Mixed paradigm languages

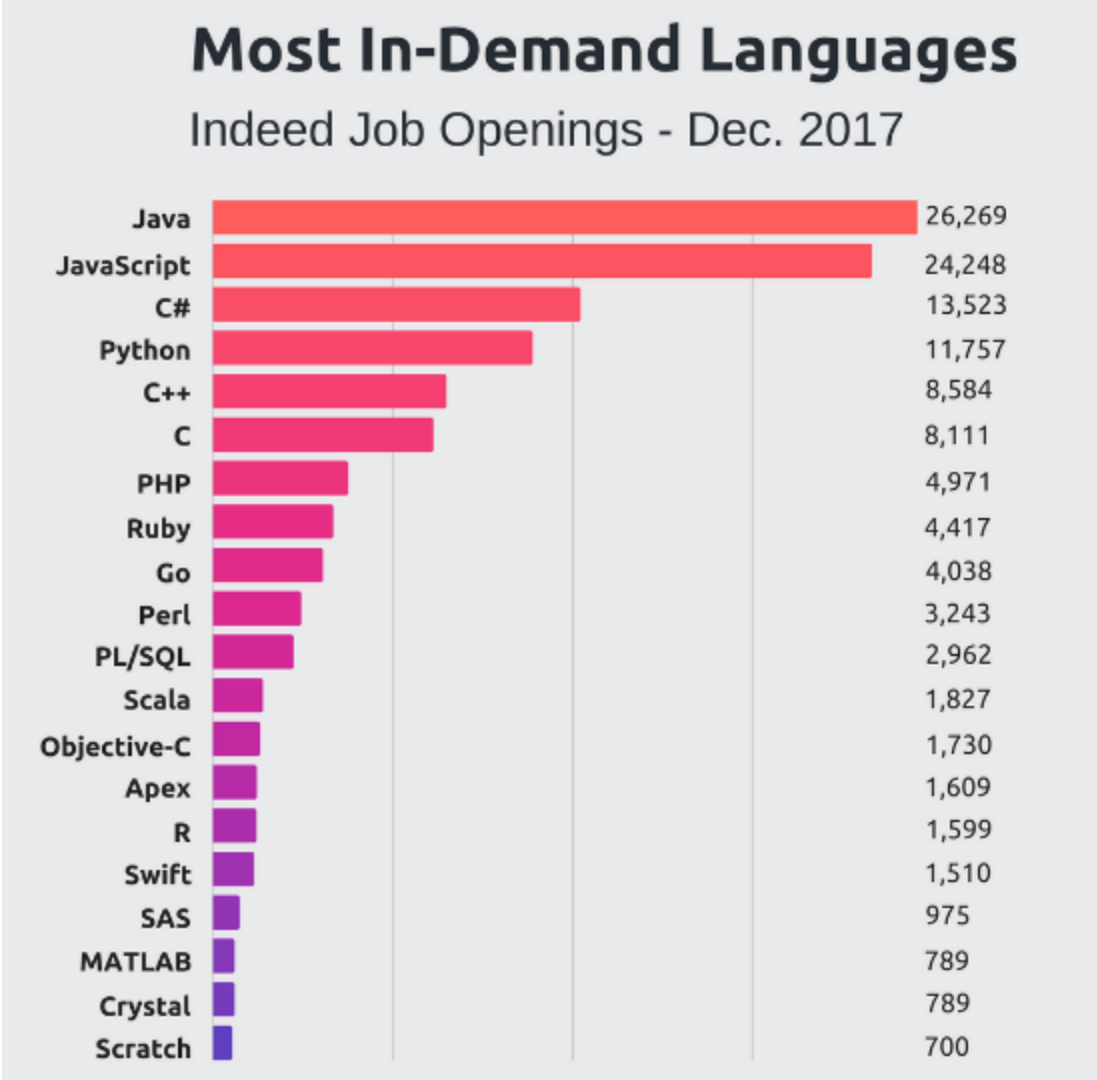
Functional programming is great

- easy parallelism and concurrency
- referential transparency, encapsulation
- compact declarative code

Imperative programming is great

- more convenient I/O
- better performance in certain tasks

There is no reason not to combine paradigms


Most In-Demand Languages
Indeed Job Openings - Dec. 2017
Java 26,269
JavaScript 24,248
C# 13,523
Python 11,757
C++ 8,584
C 8,111
PHP 4,971
Ruby 4,417
Go 4,038
Perl 3,243
PL/SQL 2,962
Scala 1,827
Objective-C 1,730
Apex 1,609
R 1,599
Swift 1,510
SAS 975
MATLAB 789
Crystal 789
Scratch 700

Back-end (Server-side) table in most popular websites

Websites	C#	C	C++	D	Erlang	Go	Hack	Java	JavaScript	Perl	PHP	Python	Ruby	Scala	Xhp
Google.com	No	Yes	Yes	No	No	Yes	No	Yes	No	No	Yes	Yes	No	No	No
YouTube.com	No	Yes	Yes	No	No	Yes	No	Yes	No	No	No	Yes	No	No	No
Facebook.com	No	No	Yes	Yes	Yes	No	Yes	Yes	No	No	Yes	Yes	No	No	Yes
Yahoo	No	No	No	No	No	No	No	No	Yes	No	Yes	No	No	No	No
Amazon.com	No	No	Yes	No	No	No	No	Yes	No	Yes	No	No	No	No	No
Wikipedia.org	No	No	No	No	No	No	Yes	No	No	No	Yes	No	No	No	No
Twitter.com	No	No	Yes	No	No	No	No	Yes	No	No	No	No	Yes	Yes	No
Bing	Yes	No	No	No	No	No	No	No	No	No	No	No	No	No	No
eBay.com	No	No	No	No	No	No	No	Yes	Yes	No	No	No	No	Yes	No
MSN.com	Yes	No	No	No	No	No	No	No	No	No	No	No	No	No	No
Microsoft	Yes	No	No	No	No	No	No	No	No	No	No	No	No	No	No
Linkedin.com	No	No	No	No	No	No	No	Yes	Yes	No	No	No	No	Yes	No
Pinterest	No	No	No	No	Yes	No	No	No	No	No	No	Yes	No	No	No
WordPress.com	No	No	No	No	No	No	No	No	Yes	No	Yes	No	No	No	No

Source: Wikipedia

Scala

Quite popular with industry

Multi-paradigm language

- simple parallelism/concurrency
- able to build enterprise solutions

Runs on JVM

Scala vs. Haskell

- Adam Szlachta's slides

Is Java 8 a Functional Language?

Based on:

https://jlordiales.me/2014/11/01/overview-java-8/

Functional language

- first class functions
- higher order functions
- **pure functions** (referential transparency)
- recursion
- closures
- currying and partial application

First class functions

Previously, you could pass only classes in Java

```
File[] directories = new File(".").listFiles(new FileFilter() {
    @Override
    public boolean accept(File pathname) {
      return pathname.isDirectory();
    }
});
```

Java 8 has the concept of method reference

```
File[] directories = new File(".").listFiles(File::isDirectory);
```

Lambdas

Sometimes we want a single-purpose function

```
File[] csvFiles = new File(".").listFiles(new FileFilter() {
    @Override
    public boolean accept(File pathname) {
      return pathname.getAbsolutePath().endsWith("csv");
    }
});
```

Java 8 has lambda functions for that

```
File[] csvFiles = new File(".")
  .listFiles(pathname -> pathname.getAbsolutePath().endsWith("csv"));
```

Streams

We want a list of adult users grouped by sex

```
public Map<Sex, List<User>> groupUsers(List<User> allUsers) {
  Map<Sex, List<User>> result = new HashMap<>();
  for (User user : allUsers) {
    if (user.getAge() >= 18) {
      List<User> currentUsers = result.get(user.getSex());
      if (currentUsers == null) {
          currentUsers = new ArrayList<>();
          result.put(user.getSex(),currentUsers);}
      currentUsers.add(user);
    }}
  return result;}
```

Streams

In Java 8, we can use higher order functions

```
public Map<Sex, List<User>> groupUsers(List<User> allUsers) {
  return allUsers
    .parallelStream()
    .filter(user -> user.getAge() >= 18)
    .collect(groupingBy(User::getSex));
}
```

Declarative style (and lazy)

- easier to understand
- easier to parallelize

Is Java 8 a Functional Language?

Functional language

- first class functions — Yes
- higher order functions — Yes
- **pure functions** (referential transparency) — No
- recursion — No tail recursion optimization by default
- closures — Only values, variables become final
- currying and partial application — Yes

No, but it provides many of the nice FP features

FP aspect in mainstream languages

	First class functions	Higher order functions	Lambda	Closures	List comprehensions	Referential transparency	Currying/partial application	Data immutability	Pattern matching	Lazy evaluation
Haskell	+	+	+	+	+	+	+	+	+	+
Java 8	(+)	+	+	+/-	-	-	(+)	(+)	-	(+)
C++14	+	+	+	+	-	-	(+)	(+)	(+)	(+)
Python	+	+	+	+	+	-	+	(+)	(+/-)	(+)
JavaScript	+	+	+	+	+	-	+	(+)	(+/-)	(+)
MATLAB	+	+	+	+	-	-	+	(+)	-	(+)

Erlang

Haskell – complex types + concurrency support

- Immutable data
- Pattern matching
- Functional programming
- Distributed
- Fault-tolerant

Map Reduce

Distributed parallel big data processing inspired by functional programming

- John Hughes's slides

Lisp for Scripting in SW Tools

- Emacs: extensible text editor
- AutoCAD: technical drawing software
- Gimp: gnu image manipulation program

Gimp

User scripts in: ~/.gimp-2.8/scripts

Register the function by

- script-fu-register
- script-fu-menu-register
- Filters → Script-Fu → Refresh Scripts

See example source code in a separate file.

(example form Gimp documentation)

TAKE-AWAYS FROM FP

Declarative programming

- write what should be done and leave how to the **optimizer**
  - particularly interesting in distributed setting
- easier to understand, no need to go back from how to what

Minimizing Side Effects

- reusability
- predictability
- concurrency
- lower mental load (modularity/encapsulation)

It is easier than it seems!

Immutability

You can use it in any programming language to

- ease parallelization
- avoid defensive copying
- avoid bugs in hashmaps / sets
- consistent state even with exceptions
- allows easier caching

It is not as inefficient as it seems!

Recursion

- Many problems are naturally recursive
  - easier to understand / analyze
  - less code, less bugs
  - combines well with immutability

- A great universal tool

Exam

Schedule

- 45 min test
  - anything hard to evaluate by programming
- 15 min break
- 2h of programming at computers (>50% points)
  - ~2 Haskell and ~2 Scheme tasks
  - upload system, otherwise no internet
  - school computers with Linux (tool requests?)

Dates (tentative): 31.5. 9:00; 6.6. 9:00; …