ProScala

Getting Scala programming well known

01 Functional Programming

Designed to Save You Resources

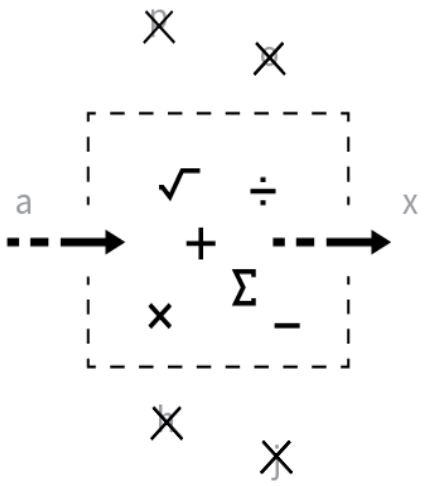
There are many different paradigms in programming [1], which we can put in different groups based on their importance and role in application development and they can be sorted based on their adoption. Considering the evolution of paradigms and the ones that can be considered general purpose, we can pick Imperative/Procedural and Functional as **two ends of a scale in general application development** and treat the others as auxiliary concepts.

What would be the point of inventing programming from the ground up, if we had no resource limitations and would not need to care about the structure of the underlying hardware? Interpreting human's will to instruct a computer would most comfortably result in structured describing text close to human speech [2], not dealing with machine specific operations.

Pure declaritive solutions - though being close to human speech - lack the characteristic of letting us implement anything we want, these are mostly domain specific solutions (eg. SQL [3]).

The most close to nature solution of handling an input to output transformation being the purpose of computing are mathematical functions.

Functions produce their output solely from their input. Functions are composeable.



$$f(g(x))$$

What are the reasons behind that computing was not created this way from its inception?

Nature of computer hardware is Imperative, on the lowest level you won't meet a circuit structure which on it's own has a shape of a function. Lowest level processor operations include accessing specific memory addresses, using the most basic arithmethic operations and doing simple conditional jumps.

```
          global    _start                      syscall

                                                mov       rax, 60
_start:   mov       rax, 1                      xor       rdi, rdi
          mov       rdi, 1                      syscall
          mov       rsi, message
          mov       rdx, 13                     section   .data
                                    message:    db        "Hello, World", 10
```

Example assembly code [4]

Accessing memory is arbitrary in most cases, though there are simple procedures, there's no such thing like hardware isolation to make these structures work exactly like mathematical functions regarding which **referential transparency** [5] (relying solely on the input, therefore producing same result for same input) is the expected natural behavior.

Is Imperative a misconception?

- at least on the lowest level things will be Imperative, but that is their place to live
- on the high level it is, considering **proveably higher mathematical chance of getting into debugging issues**
- some concepts can still be mixed with Functional (like OOP)

Evidence of resource saving using Functional instead of Imperative programming

Key pain points:

- syntax readability and debuggability [6]
- performance by clean code
    - enforced blocks by using constants
    - concise built-in design patterns
    - code ownership [7] issues - by skill (eg. patterns) - by domain getting mixed
    - Functional sense clean code [8] - easy to locate errors, code structuring

Cost of bug location

The relative cost of bug location and fixing exponentially grows based on whether it is discovered in the requirements, design, coding, unit test, system test phase or on field when the project is live.

Functional patterns provide developers the means to be able to go for sure to discover bugs in the unit testing phase the latest, assuming that given a well defined feature set resulting in a precise codebase, the chance is minimized to let coding errors cause problems in the system test phase or later. Still, human error can't be avoided, however at least coding error cost could be minimized, given that, errors from the system test phase will mostly originate from poor overall design or module integration.

Costs

...with a function:

- size of scope to review in case of unexpected outcome:
    - Mathematical function : **function scope**
    - Imperative programmatic function (using variable out of scope of function) : **whole program scope**

...with constant usage enforced and without:

- with constant: **single point** of value assignment
- with variable: **arbitrary points** of value assignment

...with composing mathematical functions:

- with mathematical function composition: **calculable binary steps**
- with Imperative programmatic function (using variables out of scope of functions): **can be infinite**

02 Functional Alternative for Java

OOP-Functional Hybrid Concept

Advantages and disadvantages of a pure Functional [9] language

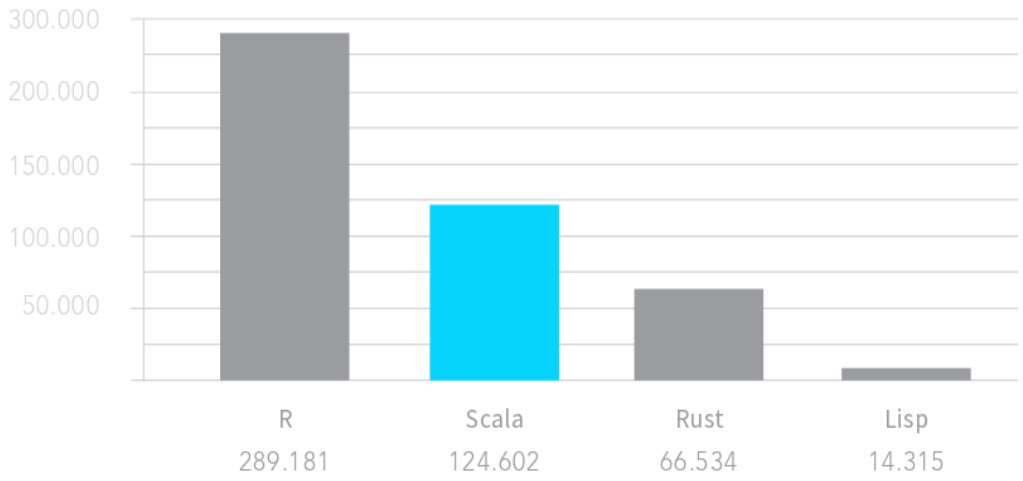
In a pure Functional language, Functional sense clean code is enforced strictly but is less readable and results in a much steeper learning curve for a Java developer. Pure Functional languages do not contain OOP solutions in a way that a Java developer could easily get accustomed to.

OOP is not necessarily contrary to the Functional paradigm, though we should keep some restrictions in mind to use an OOP structure in a Functional way. There comes the concept of OOP-Functional hybrid languages [10], which let us use OOP structures and even utilize them to expand the concept of the Functional paradigm.

Focusing on programming languages being designed based on the Functional paradigm from their inception, picking those which are OOP-Functional hybrids, we have the chance to examine, **which is a good fit for Java**[11] **and other Imperative background developers** to take their knowledge to the next level with an easier entry burden, compared to pure Functional.

Key points in comparing such languages are community activity, design for general usage and vendors using them.

OOP-Functional hybrid languages community activity based on Github search[12]:



General or domain specific usage factors:

- **Scala** - general usage, JVM language, interoperable with Java, Java-ish syntax
- **Rust** - general usage, C++ like syntax
- **Lisp** - general usage
- **R** - domain specific for statistics

Vendors [13] using specific Functional languages:

- **Scala** - Twitter, Foursquare, Apple, UBS, LinkedIn, SoundCloud, Morgan Stanley
- **Rust** - Firefox, Dropbox, Yelp
- **R** - Airbnb, The Economist, Facebook, Google
- **Lisp** - Grammarly, Siscog

Considering the above factors, **Scala** [14] is ProScala's choice to be the language to promote and to illustrate Functional style programming benefits for Java and other developers.

03 Community Goals

Growing Community - More Vendors

Hurdles[15] of getting programming languages like Scala widespread

- Paradigm-bias of education
- Learning curve
- Gravity-effect of incumbent, large languages
- Sunk cost fallacy[16]
- More and more vendors needed

Addressing the latter issues is a huge interest of the community, securing existing Scala/Functional based jobs for fostering a higher adoption of the language. The more means the community has to bring in more vendors, the higher the impact is on the industry.

04 Foundation

Base of Core Activity

ProScala Foundation's aim is to advance vital and self-sustaining activities that can change the paradigm-bias of education [17], dealing with learning curve issues [18], and to provide means for the community to pursuade vendors and developers into adopting advanced and friendly technologies.

The Foundation focuses on promoting content creation that is in harmony with these goals and partners with firms or individuals who are willing to advance community goals in an organized manner.

The Foundation raises funds for supporting authoring, design, audio creative work and marketing activities and the leadership to keeping these pieces together, aimed to bring forth a result in providing a framework that helps advancing the mentioned goals. Support the Foundation to be able to deal with ongoing content creation, delivery, marketing efforts to build up enough traction to have a higher impact, and consulting activity in regard to guidelines for an effective way of implementing Functional and Scala in your organization.

05 Self-sustaining Process

of Spreading Knowledge

Multiple channels to let the community foster advanced tech adoption

By providing the right user experience for the people consuming provided educational content, and the right tools to reach others, also to talk about all the topics that can attract more vendors, it is more likely to reach community goals.

Podcast sponsored by Foundation aiming to be flagship content.

Addressing the issue of making technical content easily consumable is key to reaching **not just technical, but business audiences** and to give an easy to see through guidance about whether it is worth the effort to dive deeper into a specific topic. The podcast aims to fulfill these goals and to be a basis of systematically built other content that can reference it.

Membership site integrated in the process to foster community activity on a business level.

Reaching the right community size while providing ever-green flagship content can be the key to ignite a cycle of positive feedback of the usefulness of advanced technologies, then allowing to remove the above mentioned barriers that recently prevented getting these widespread.

06 Podcast Concepts

Easy to Consume Content Reaching Relevant Audiences

ProScala podcasts aimes to provide easy to consume educational content that helps to involve all stakeholders.

The added value of the material compared to conventional content available on the Internet is to build it up the hard way, meaning, we stick to the aim of addressing **multiple audiences**, while providing useful information in a much shorter time for the listeners compared to traditional means of IT education [19].

It is built top-down providing a big picture about issues and industry practices giving the opportunity for people unfamiliar to Functional, experts and business decision makers to decide whether a specific concept can save them time and money, not to mention stakeholders' delight that advanced technologies can bring to the process and organizations.

07 Membership Site

Facilitator of Growth

Once core content creation activities gain **enough traction** that can be the subscriber source of a membership site aligned with community goals, industry knowledge exchange can be done on a business level bringing in more incentives to the process of **advancing change in the industry**.

Aligned with the goal of bringing in more vendors to the Functional space both including larger companies and greenfield projects, a membership site can facilitate growth by letting various industry players (developers, recruiters, decision makers) meet and exchange their knowledge in a controlled way each getting an incentive for their vital interactions, meanwhile producing community value.

The offer for **developers** addresses the issue that getting useful advanced technical knowledge quicker is most likely to happen by attending costly in-person multi-day trainings or conferences [20]. Our aim is to provide similar value for a fair price compared to the previously believed effective solution, granting the chance for more people to participate.

The offer for **recruiters** is that they can work in a controlled manner with less overhead, though it is expected to create much less noise[21] then what could otherwise be seen in social media channels. Relevant business information is curated, and can be reached in a marketplace like fashion, the aim is to have such business interactions be vetted by the community itself. This can make member recruiters much more likeable for developers compared to their out of membership site competitors.

The offer for **decision makers** is that they can gain industry insights that are resulting from discussions ignited by content especially tailored for their needs, as the content being built top-down in a predecision research manner, therefore the main flow of communication won't be about pure low-level technicals but the business level utility even considering that this will still be dev talk.

08 Business Model Research

Acceptance of Alternative Show Concept

Fulfilling the mission of serving all the relevant community goals to bring the Foundation and it's partnered businesses the chance to be vital to **advance change in the industry** results in the need for an alternative business model compared to incumbent solutions however this raises the question of user acceptance.

Traditional concepts are mostly based on providing a single audience podcast for free with low-cost hands-on coding materials, which does not address the issue of creating a **multi-audience engaging top-down built content** that is easily consumable and can attract people in no time.

For these reasons, research was made to make sure that the target audience would be happy with pairing top-down built evergreen podcast content with a membership site providing premium content. According to research, a **relative majority of developers liked the concept and an absolute majority of them are willing to pay for premium services**. Research was made based on prequel episodes and social media outreach, results were reassured by polling the community about the first broad topic episodes.

09 Expected Effect

The Trajectory This Project is Aimed to Go On

It is expected to attract significant community involvement in interacting with content aiming to ignite discussions and provide valuable possibilities for people keen on Functional programming to tell others why it has an important role in the industry, and it is also likely that the provided materials for reasoning to decision makers will be useful as well.

A community effort requiring job is expected to be tough with building enough traction for sustaining user activity in a membership site, however given that risk a vital outcome is to take place on the short term like simply bringing in a significant portion of new players, and a disruptive outcome on the long run, like having an effect on IT education to bring it toward making the learning curve of Functional less steep.

Getting Scala well known.

AUTHOR

Csaba Kincses

Read more posts by this author.

References

[1] ∧ Programming paradigms http://people.cs.aau.dk/~normark/prog3-03/html/notes/paradigms_themes-paradigm-overview-section.html

[2] ∧ Hints on programming-language design - 13.3.5. Readability https://www.cs.yale.edu/flint/cs428/doc/HintsPL.pdf

[3] ∧ SQL https://www.stat.berkeley.edu/~spector/s296/sql.pdf

[4] ∧ Example for close to machine (low-level) - Assembly language programming "Hello World" example https://cs.lmu.edu/~ray/notes/x86assembly/

[5] ∧ Referential transparency https://www.cs.ucf.edu/~leavens/ComS541Fall97/hw-pages/paradigms/ref_trans.html

[6] ∧ Functional patterns are well aligned practicing debugging strategies Essay about the concept of simply not letting buggy code compile https://crypto.stanford.edu/~blynn/haskell/curry-howard.html

[7] ∧ Code ownership being approached above from a preferred Feature Driven Development point of view

[8] ∧ Clean code in Functional programming https://praveer09.github.io/technology/2016/08/04/how-functional-programming-helps-me-write-clean-code/

[9] ∧ Example of a pure Functional language is Haskell http://www.sci.brooklyn.cuny.edu/~kopec/cis24/spring2003/aslam_pj3.pdf

[10] ∧ Languages sharing both Functional and object oriented traits https://en.wikipedia.org/wiki/List_of_object-oriented_programming_languages, https://en.wikipedia.org/wiki/Category:Functional_languages

[11] ∧ Javaers as target audience

[12] ∧ Github activity related to specific languages as of Nov 2021 https://github.com/search?q=language%3Ar, https://github.com/search?q=language%3Ascala, https://github.com/search?q=language%3Arust, https://github.com/search?q=language%3Alisp

[13] ∧ Vendors using programming languages https://en.wikipedia.org/wiki/Scala_(programming_language)#Companies, https://www.rust-lang.org/, https://www.guru99.com/r-programming-introduction-basics.html, https://lisp-lang.org/

[14] ∧ Scala programming language https://www.scala-lang.org/

[15] ∧ Empirical Analysis of Programming Language Adoption - 5; 5.4; 6; 6.1; http://lmeyerov.github.io/projects/socioplt/papers/oopsla2013.pdf

[16] ∧ Sunk Costs, Rationality, and Acting For the Sake of the Past https://www.princeton.edu/~tkelly/sc.pdf

[17] ∧ The Case for Teaching Functional Programming https://cs.wheaton.edu/~tvandrun/dmfp/case4dmfp.pdf

[18] ∧ Online vs. Traditional Education https://www.rasmussen.edu/student-experience/college-life/online-vs-traditional-education-answer-never-expected/

[19] ∧ Conferences prices https://www.scala-lang.org/events/, https://www.bmc.com/blogs/tech-it-conferences/

[20] ∧ Why Traditional Recruitment Doesn't Apply to Tech - point 4. on noise https://techloop.io/for-companies/why-traditional-recruitment-doesnt-apply-to-tech/

[21] ∧ Details about Functional programming learning curve issues

ng Scala well known.