21.07.2020 / BY ŁUKASZ KUCZERA

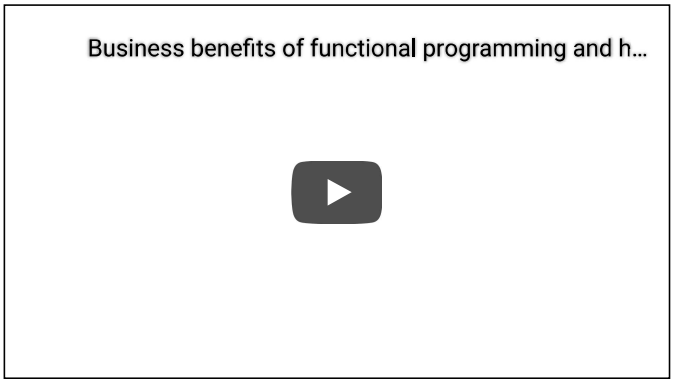
What is functional programming and why it matters for your business?

CTO'S GUIDEBOOK

Since functional programming evolved from lambda calculus and has its roots in academia, it was initially discussed in scholarly contexts. It's no longer the case. Even though significant technologies associated with functional programming (such as Lisp or Scala) were created at universities — FP is making its way to the general "software development discourse," and some might say that it's taking it by storm.

Despite the rising interest, however, not many articles on the web seem to tackle the benefits of functional programming for business. Let's face it: hacking a new technology in spare time is great, but there's always a practical side to business when it comes to trying out new things and having a good reason for that. And that's precisely what we're about to cover in this piece.

Here's what functional programming is, and why does it matter for your business.



What is functional programming in the first place?

Functional programming (FP) is a **programming paradigm**, a certain way of thinking about software development that's ba fining principles. In the case of FP, these key principles are as follows:

1. Binding everything in **pure mathematical functions**. The purpose is to look for simple, repeatable actions that function and then built upon to create more complex features.
2. **Immutable, unchangeable data** — which means that you should rather create new data structures, not altering the ones that are already in place. For example, if you want to modify data in an array, you need to add a new array with the updated value instead of making changes "original" one.
3. **Declarative code**. The main focus here is on what to solve, rather than how to do it.

repeatedly call themselves, until the base case is reached.

Functional programming vs OOP (Object Oriented Programming)

Speaking of programming paradigms, you might already be familiar with another one that's still more popular: object-oriented programming (OOP). This doesn't mean, however, that FP is not gaining any traction. Both functional programming and object-oriented programming are unique in their own ways.

Here's functional programming vs OOP at a glance to help you grasp the differences:

X	Functional programming	Object-oriented programming
Data	Immutable	Mutable
Programming model	Declarative	Imperative
Focus	What you're doing	How you're doing
Parallel programming	Supports parallel programming	Not suitable for parallel programming
Side effects	Its functions have no side effects	Its methods can produce side effects
Iteration	Recursion	"Loop" concept
Application state	Flows through pure functions	Is usually shared and colocated with methods in objects
Key elements of the code	Variables and functions are the main elements of the code	Objects and methods are the key elements

What about programming languages, then?

Pure functional programming languages, such as **Haskell**, are specially designed to serve this particular paradigm and accept only pure functions. Interestingly, you don't need to use a pure functional programming language to bring FP principles to your code.

There are a few languages that still make it easy to write "pure programs", such as **Scala**, **Clojure** or **OCaml**. More popular programming languages like **JavaScript** or **Python** can also support FP one way or the other – either natively or with the right library.

At Scalac, we're big advocates of functional programming and, which shouldn't come as a surprise, we mainly use Scala for this purpose. There are at least a few good reasons for that:

The language is extremely versatile and offers advanced features, clean code, and both functional and object-oriented programming in an open-source, practical package that leverages Java's environment.

Speaking of — **Scala has emerged as one of the most powerful Java alternatives.** Actually, one of the reasons why Scala was created in the first place was to address various concerns developers had with Java. Nowadays, Scala provides interoperability and compatibility with Java, allowing the developers to keep their Java libraries, and leverage the advantages of JVM.

When comparing it to Python, on the other hand, Scala's maintenance is simply much faster. **Despite providing the fluency and flexibility of a dynamic language like Python, it's still a strongly statically typed language.** The performance also tends to be much more efficient, especially in a multi-threaded environment.

Because of its functional aspects and flexibility, **Scala is also useful for expressive code and parallelism**. Basically, with a functional approach and no mutation, there are no race conditions and parallelism gets a lot more straightforward. Then, lambda expressions are also helpful when communicating some aspects of parallelism.

But, the question remains:

Why should you use functional programming (and Scala, for that matter)?

For some reason, this programming paradigm still seems complex for many. When taking a closer look at the benefits of functional programming and languages associated with it, you might have a different impression.

To start with, adopting **functional programming helps you break down every application into smaller, simpler pieces that are both more reliable and easier to understand**. It's mostly because functional code tends to be more concise, predictable, and easier to test. How come?

Since pure functions don't change any states and depend only on the input given, they are much **easier to grasp**.

When there's less code, there are also fewer bugs — not to mention that **actual testing and debugging gets mu** pure functions that take only arguments and produce output.

With the outputs depending only on the arguments passed to the function, **the software can also behave more "predictably"**.

FP adopts **Lazy Evaluation** which avoids repeated evaluation (basically, the value is evaluated and stored only when it is needed) and support Lazy Functional Constructs like Lazy Lists, Lazy Maps, etc.

Speaking of efficiency, functional programs consist of independent units that can **run concurrently** and can be more efficient as a result.

Since Scala supports functional programming, these benefits apply to it as well. In fact, the combination of features in Scala makes it possible to write programs that are concise, elegant, and much easier to debug & maintain.

As a result, increasingly more companies switch to both functional programming and Scala, including well-known tech giants like Twitter or LinkedIn. Interestingly, Scala also made it to the top 10 languages that developers want to learn these days.

The rising interest from both companies and developers are contributing to the recent growth of functional programming and the adoption of Scala as the main language for many applications — also in a business setting.

What do functional programming & Scala mean for your business, though?

There's no doubt that knowing the principles of functional programming can broaden one's horizons and introduce new ways of thinking about software development. But what's in it for the companies?

To start with, **code quality**. When it comes to functional programming — very often, you can do more with less code. It's much **more concise**, which essentially means **fewer bugs, and shorter time** (and/or fewer people) to actually write it. Plus, it's a lot **easier to grasp what the code does**, explain it, and even onboard new team members who can almost immediately join the project and start working on it. In the process, Time to Market often gets shorter.

What's also worth mentioning is that the interest in functional programming and languages that bring its principles to life stems from being **tired of limitations posed by other programming paradigms and technologies**. As already mentioned, Scala as a modern language has plenty of useful features that you won't find in Python and Java. Even implementing it can be a big savings cost of the bottom line.

What's more, **there are already some business cases that both functional programming and Scala have been applied to successfully.** Health care, accounting, banking, advertising — are just a few exemplary industries.

For the time being, functional programming seems to work best with apps aimed at concurrency or parallelism, carrying out mathematical computation, and whenever there are many different operations performed on the same data set. Scala is a good fit if you have a lot of data or complex data structures and algorithms, but FP languages are also often used for artificial intelligence applications like machine learning, language processing, or modelling of speech and vision. Pretty impressive, right?

With the rising interest and so many benefits of functional programming, how come it's not yet as widespread as it should be?

At the end of the day, businesses rather choose the technology based on the skills of their team or the availability of developers, who are more likely to represent more traditional approaches to software development. That's precisely why it matters not only to educate developers about FP but also give them tools to apply it in practice.

For now, "the movement" is all bottom-up: it's starting from developers who are tired of being limited by other programming languages, become interested in functional programming, and then teach their colleagues about it.

As these developers start creating libraries and frameworks that solve specific business problems across multiple areas (such as backend development, analytics, distributed development) with the use of functional programming — eventually decision-makers will also get excited about FP. And, the more companies are interested in functional programming, the more developers skilled at it they want to hire. It's a chain reaction from there.

To facilitate the shift to functional programming, drop some of the jargon around FP and lower the barriers to entry, there are initiatives and projects like ZIO — a library for asynchronous and concurrent programming based on pure functional programming. Taking advantage of it is said to help in solving common business problems, and ensuring higher success rates of projects. This, in turn, brings more attention to functional programming, in the business world as well.

What's also great about ZIO is that it's inclusive for all sorts of developers. It's still fairly new, but it's expected to be a great venture in the long run, mostly due to all the open-source support it gets. It might be difficult to take advantage of ZIO when developing enterprise-level software just yet, but more and more companies are already testing it in production for smaller projects. And that's only one of the reasons why the future of functional programming definitely is bright.

Dig into functional programming yourself

Increasingly more companies are prepared to train development teams on how to use FP and solve business pr
programming, including Scalac.

Switching to functional programming can be done whenever there's a new module to implement and thus, can be done gradually. Start sma
see how big of a difference the main aspects of functional programming can make to your business. *As per usual, we're here to help*.

Why does Scala win against Kotlin? Senior engineer's opinion

Zalando case study: why they chose Scala?

Why Developers Should Pay Attention to ZIO in 2021

The Top Scalac's Posts of 2020

Author

ŁUKASZ KUCZERA

I WROTE THE FIRST COMPUTER PROGRAM WHEN I WAS 8 AND FOUNDED THE FIRST COMPANY WHEN I WAS 16. SINCE THEN, I'M LOOKING FOR OPPORTUNITIES TO IMPROVE THE WORLD AROUND ME. I'M A PEOPLES PERSON, AND I LOVE TO LEAD AND INSPIRE. I COULDN'T FIND A "COMPANY THAT I ALWAYS WANTED TO WORK FOR," SO I DECIDED TO CREATE MY WORKPLACE WHERE I CAN SHARE MY PASSION WITH LIKEMINDED INDIVIDUALS. I HAVE BEEN WEARING DIFFERENT HATS THROUGHOUT MY CAREER AND FOUNDED SEVERAL PROJECTS AND COMPANIES. I WORKED AS A SYSTEM ADMINISTRATOR, SOFTWARE ENGINEER, AND ARCHITECT I HELD THE ROLE OF VP OF ENGINEERING AND CEO. I FOUNDED ISP COMPANY, GAMIFICATION STARTUP, VENTURE FUND, GAMEDEV STUDIO, AND SPECIALIZED SOFTWARE DEVELOPMENT COMPANY.

Latest Blogposts

21.04.2021 / BY MAJA KRYŻAN

How do we build a PRO tech team?

BACKSTAGE CTO'S GUIDEBOOK

Start with "why" We exist to help companies develop amazing IT products. To develop incredible products that are future-proof, stable, fast, and scalable, companies need great teams to do the job. Finding the right people to develop good software is challenging. Recruit IT professionals. Preferably – the best ones. This is where Scalac comes in...

08.04.2021 / BY DARIA KARASEK

How Outsourcing Can Solve Technical Debt

CTO'S GUIDEBOOK

Technical debt has become widely prevalent nowadays. Since technology is constantly evolving, many businesses have to choose between acquiring new solutions or sticking with tried-and-tested ones. No right answer can be given to this hard choice. But even though debt sounds threatening for many, it doesn't always have to be.

Newsletter

EMAIL ADDRESS*

SUBSCRIBE

scalac

ESTIMATE PROJECT

Services

ANALYTICAL DASHBOARDS

BLOCKCHAIN

DATA LABS

DISTRIBUTED SYSTEMS

TRAINING

WORKSHOPS

Company

EU PROJECTS

PRIVACY POLICY

Headquarter

GDAŃSK, POLAND - SCALAC SP. Z O. O.

CZESŁAWA MIŁOSZA 9/9
80-126 GDAŃSK, POLSKA
+48 730 030 492

KRS 0000493511
NIP 5842734834

US office

SAN FRANCISCO USA - SCALAC, INC.

1160 BATTERY ST SUITES 100
SAN FRANCISCO, CA 94111, USA

Got any questions? I'm happy to help.