OCamello: A Course and Summer School with Learn-OCaml

ROBERTO BLANCO, MPI-SP, Germany
RICARDO J. RODRÍGUEZ, University of Zaragoza, Spain

We report on an (at the time of this writing, forthcoming) week-long summer school on functional programming and OCaml, entitled *Advanced Programming Techniques: The Functional Paradigm*, part of the 95th Annual Edition of the interdisciplinary summer university of the University of Zaragoza. We develop new custom learning materials using Learn-OCaml as an integrated learning platform and bring together academic and industrial members of the OCaml community for an associated outreach event.

Additional Key Words and Phrases: OCaml, summer school, learning outcomes

ACM Reference Format:
Roberto Blanco and Ricardo J. Rodríguez. 2018. *OCamello*: A Course and Summer School with Learn-OCaml. *J. ACM* 37, 4, Article 111 (August 2018), 3 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

There is agreement among experts, and some supporting empirical data, that modern functional programming languages result in simpler and less buggy programs [7], whose correctness is easier to assess and even guarantee. The influence of the functional paradigm has become widespread in recent years, driving ongoing adoption of functional idioms by many conventional imperative languages [2–5], which make it one of the most useful tools for anyone interested in the development of quality software.

It is therefore unfortunate, in our opinion, that these techniques remain little known in certain areas, as is the case of Spanish higher education, where computer science curricula recommendations make no explicit mention of them [1]. The general trend in recent times has been to reduce foundational exposure to strong and statically typed programming languages (such as Ada or Java) in favor of more dynamic or unsafe languages (such as Python or C++). While the latter play an important role in a balanced education, we also believe it is unwise to neglect the former.

With this in mind, we argue that OCaml is an ideal language to fill these gaps: on the one hand, increasing exposure to a strongly-typed, static, memory-safe language; on the other, studying the functional paradigm, its foundations and applications. To this end, and to address the scarcity of related learning materials in the standard language of instruction, we have developed an introductory course to functional programming, which we hope will facilitate access to this knowledge and serve as basis for future teaching efforts. We intend to release all materials under a permissive open license.

The original run of the course is organized as a week-long summer school[1], presented under the name *Advanced Programming Techniques: The Functional Paradigm*. This course is part of the

[1]Note to reviewers: at the time this writing, the course is scheduled to be held between 4–8 July 2022.

Authors' addresses: Roberto Blanco, MPI-SP, Bochum, Germany, roberto.blanco@mpi-sp.org; Ricardo J. Rodríguez, University of Zaragoza, Zaragoza, Spain, rjrodriguez@unizar.es.

0004-5411/2018/8-ART111 $15.00
https://doi.org/XXXXXXX.XXXXXXX

95th annual edition of the interdisciplinary summer university of the University of Zaragoza, the oldest of its kind in Spain. The initiative is sponsored by the OCaml Software Foundation and according to university officials, this is the first time a course has been sponsored by an international organization.

2 COURSE DESCRIPTION

The first edition of the course covers 30 hours of lectures, divided into 24 hours of standard lectures and 6 hours dedicated to the outreach event (see Section 3). In addition to these, the course load includes 10 hours of personal study and another 10 hours reserved for the development of a small programming project outside the classroom, required for participants taking the class for 1.5 European Credit Transfer System credits. Enrollment is open to anyone with a basic understanding of programming, from intermediate to advanced undergraduate or graduate students (particularly in computer science, engineering, mathematics and related disciplines) to IT professionals.

The foundation of the project are the introductory lectures to the OCaml language and its software ecosystem. Each lecture combines a tutorial-style unit and exercises, some solved interactively and others individually by the participants. To these activities we add interactive questions (via clickers) and discussions among participants. We use Learn-OCaml for all class materials. Inside the Learn-OCaml framework, we favor the tutorial over the lesson environment, and supplement it with the exercise environment for class exercises and homework. In particular, we seek to emulate and adapt the highly interactive style of lectures based on [6]. We do some light pre-processing on the original sources to improve formatting and introduce a gradual presentation of the contents, a bit closer to a standard slide deck, while preserving the OCaml top-level view at all times.

The lectures introduce OCaml from scratch and cover the core of the language, with emphasis on the various components of expressions and types, higher-order programming and the module system. As the course is aimed at people with some programming experience, we also include the imperative features of OCaml, its connection to systems programming and model of interaction with other languages and tools, as well as brief introductions to software quality (aspects such as secure programming, testing and verification) and theoretical foundations. We de-emphasize some of more advanced features (particularly the object system) and topics (such as monads and effects). The main goal is that at the end of the course the participants are able to use OCaml to solve their programming problems in a clean and effective way.

3 A PUBLIC OUTREACH EVENT

One of our main goals is to help dispel ingrained misconceptions of functional programming as primarily an academic endeavor, and to showcase the OCaml language as a project that is actively used in both academia and industry. For this, we complete the course lectures with an outreach event, open to the public and to remote participants, where we bring together members of the OCaml community to offer a holistic view of its diversity and impact.

This event is organized in two parts. First, we invite members of the OCaml team to give an account of the language, its history, development, and future prospects. Second, we invite companies using OCaml as their primary language to present their work and explain how the language helps them in their business operations. Each of these parts is followed by a round table discussion.

ACKNOWLEDGMENTS

We wish to thank the OCaml Software Foundation for their generous support, and to Gabriel Scherer for his helpful advice.

REFERENCES

[1] ANECA. 2005. Libro Blanco del Título de Grado en Ingeniería Informática. http://www.aneca.es/var/media/150388/libroblanco_jun05_informatica.pdf. In Spanish.

[2] Alex Gyori, Lyle Franklin, Danny Dig, and Jan Lahoda. 2013. Crossing the gap from imperative to functional programming through refactoring. In *Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE'13, Saint Petersburg, Russian Federation, August 18-26, 2013*, Bertrand Meyer, Luciano Baresi, and Mira Mezini (Eds.). ACM, 543–553. https://doi.org/10.1145/2491411.2491461

[3] Jaakko Järvi and John Freeman. 2010. C++ lambda expressions and closures. *Sci. Comput. Program.* 75, 9 (2010), 762–772. https://doi.org/10.1016/j.scico.2009.04.003

[4] Davood Mazinanian, Ameya Ketkar, Nikolaos Tsantalis, and Danny Dig. 2017. Understanding the use of lambda expressions in Java. *Proc. ACM Program. Lang.* 1, OOPSLA (2017), 85:1–85:31. https://doi.org/10.1145/3133909

[5] Brian McNamara and Yannis Smaragdakis. 2000. Functional programming in C++. In *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP '00), Montreal, Canada, September 18-21, 2000*, Martin Odersky and Philip Wadler (Eds.). ACM, 118–129. https://doi.org/10.1145/351240.351251

[6] Benjamin C. Pierce et al. 2013. Software Foundations. https://softwarefoundations.cis.upenn.edu/.

[7] Baishakhi Ray, Daryl Posnett, Premkumar T. Devanbu, and Vladimir Filkov. 2017. A large-scale study of programming languages and code quality in GitHub. *Commun. ACM* 60, 10 (2017), 91–100. https://doi.org/10.1145/3126905